

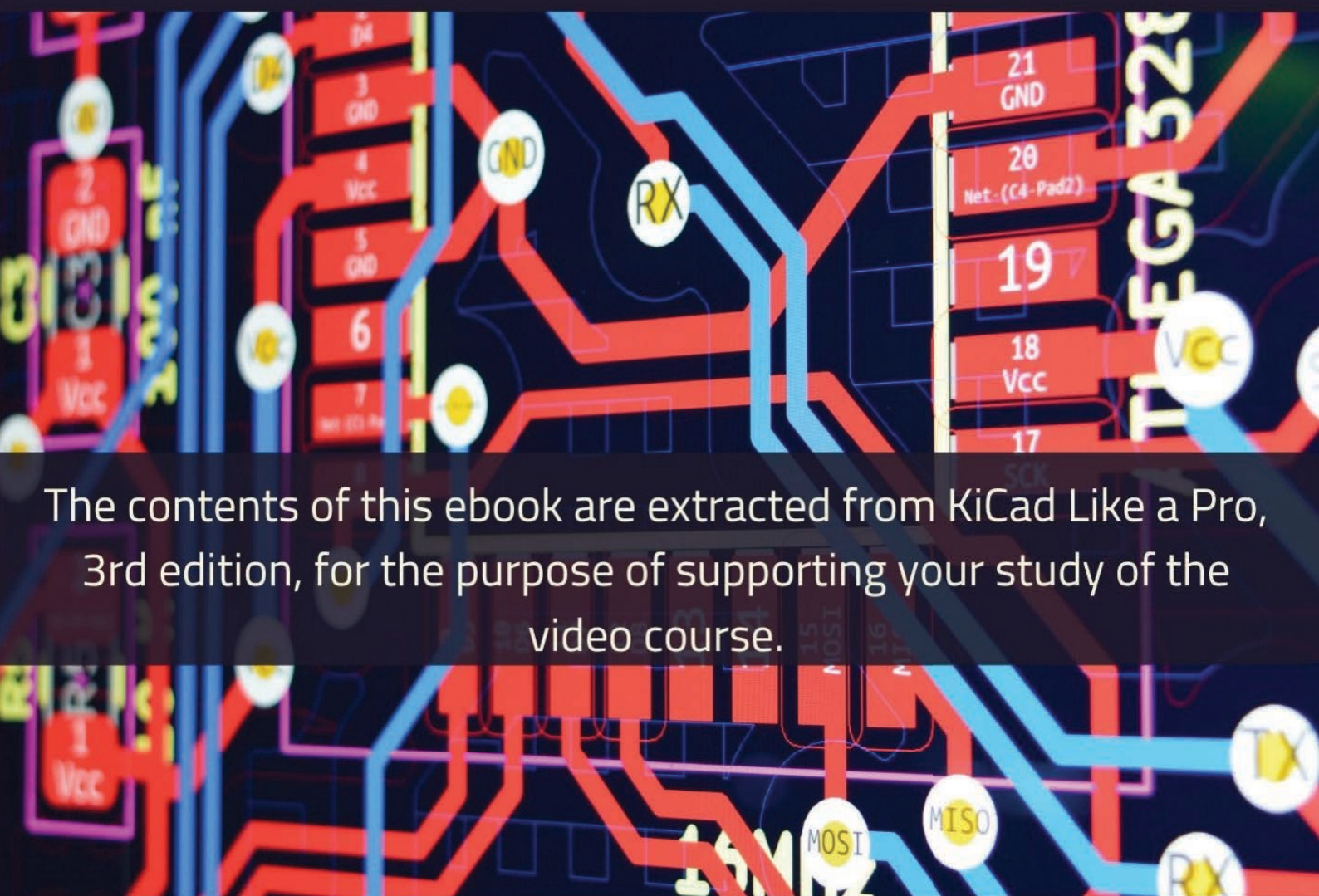
3RD EDITION

# KICAD LIKE A PRO

## FREE VIDEO COURSE COMPANION

---

DR PETER DALMARIS



The contents of this ebook are extracted from KiCad Like a Pro, 3rd edition, for the purpose of supporting your study of the video course.

# KICAD LIKE A PRO, THIRD EDITION

## **KiCad Like a Pro, 3<sup>rd</sup> Edition**

By Dr Peter Dalmaris

Copyright © 2021 by Tech Explorations™

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher except for the use of brief quotations in a book review.

Printed in Australia

First Printing, 2018

ISBN (PDF) : 978-1-68489-093-4

ISBN (epub): 978-1-68489-094-1

ISBN (mobi): 978-1-68489-089-7

Tech Explorations Publishing  
PO Box 22, Berowra 2081 NSW  
Australia

[www.techexplorations.com](http://www.techexplorations.com)

**Cover designer:** Michelle Dalmaris

### **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.

Version 1

## 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).

Supporting our students through their learning journey is our priority, and we do this through our dedicated online community and course forums.

Founded in 2013 by Peter Dalmaris, Tech Explorations was created after Peter realised how difficult it was to find high-quality definitive guides for the Arduino, written or produced by creators who responded to their reader questions.

Peter was frustrated having to search for Youtube videos and blog articles that almost never seemed to be made for the purpose of conveying knowledge.

He decided to create Teach Explorations so that he could produce the educational content that he wished he could find back then.

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 that 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.

We want to help you take this journey without delays and frustrations.

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.

Here's what some of them had to say:

*"I'm about half way through this course and I am learning so much. Peter is an outstanding instructor. I recommend this course if you really want to learn about the versatility of the amazing Raspberry Pi" -- Scott*

*"The objectives of this course are uniquely defined and very useful. The instructor explains the material very clearly." -- Huan*

*"Logical for the beginner. Many things that I did not know so far about Arduino but easy to understand. Also the voice is easy to understand which is unlike many courses about microcontrollers that I have STARTED in the past. Thanks" -- Anthony*

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

## From the back cover

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.

# How to read this book

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

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 go straight to Parts 7 and 8, zoom through them quickly, and then proceed to the projects in Part 9.

Once you have the basic KiCad concepts and skills confidently learned, you can use the recipes in Parts 7, 8 and 13 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.

Throughout this book, 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 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 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.

# 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 [txplo.re/kicadr](http://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.

# Why all the ?? in the TOC?

The version of the book you are looking at is the video course companion.

This version of the book contains specific chapters from the full book - not all of them.

For this reason, any chapter from the full book that is not available in this companion is marked with “??” Instead of an actual page number in the Table of Contents.

All the pages and chapters from the full book that are present in the companion book will have the actual page number correctly listed in the TOC.

# Table of Contents

Introduction.....	29
1. What is a PCB?.....	30
2. The PCB design process.....	36
3. Fabrication.....	41
4. Get KiCad for your operating system.....	43
5. Example KiCad projects.....	47
Part 2: Getting started with KiCad 6.....	56
1. Introduction.....	57
2. KiCad Project Manager (main window).....	58
3. Overview of the individual KiCad apps.....	64
4. Paths and Libraries.....	72
5. Create a new project from scratch.....	76
6. Create a new project from a template.....	78
7. KiCad 6 on Mac OS, Linux, Windows.....	82
8. Differences between KiCad 6 and 5.....	87
Part 3: Project - A hands-on tour of KiCad - Schematic Design.....	89
1. Introduction to schematic design and objective of this section.....	90
Design workflows summary.....	??
The finished KiCad project and directory.....	??
Start Kicad and create a new project.....	??
1 - Start Eeschema, setup Sheet.....	??
2 - Add symbols.....	??
3 - Arrange, annotate, associate.....	??
4 - Wiring.....	??
5 - Nets.....	??
6 - The Electrical Rules Check.....	??
7 - Comments with text and graphics.....	??
Part 4: Project- A hands-on tour of KiCad - Layout.....	93
1. Introduction to layout design and objective of this section.....	94
1 - Start Pcbnew, import footprints.....	??
2 - Outline and constraints (edge cut).....	??
3 - Move footprints in place.....	??
4 - Route (add tracks).....	??
5 - Refine the outline.....	??
6 - Silkscreen (text and graphics).....	??

7 - Design rules check.....	??
8 - Export Gerbers and order.....	??
The manufactured PCB.....	??
Part 5: Design principles and PCB terms.....	??
Introduction.....	??
Schematic symbols.....	??
PCB key terms.....	??
FR4.....	??
Traces.....	??
Pads and holes.....	??
Via.....	??
Annular ring.....	??
Soldermask.....	??
Silkscreen.....	??
Drill bit and drill hit.....	??
Surface mounted devices.....	??
Gold Fingers.....	??
Keep-out areas.....	??
Panel.....	??
Solder paste and paste stencil.....	??
Pick-and-place.....	??
Part 6: PCB design workflows.....	??
The KiCad Schematic Design Workflow.....	??
Schematic Design Step 1: Setup.....	??
Schematic Design Step 2: Symbols.....	??
Schematic Design Step 3: AAA (Arrange, Annotate, Associate).....	??
Schematic Design Step 4: Wire.....	??
Schematic Design Step 5: Nets.....	??
Schematic Design Step 6: Electrical Rules Check.....	??
Schematic Design Step 7: Comments and Graphics.....	??
The KiCad Layout Design Workflow.....	??
Layout Design Step 1: Setup.....	??
Layout Design Step 2: Outline and constraints.....	??
Layout Design Step 3: Place footprints.....	??
Layout Design Step 4a: Route.....	??
Layout Design Step 4b: Copper fills.....	??
Layout Design Step 5: Silkscreen.....	??
Layout Design Step 6: Design rules check.....	??



Layout Design Step 7: Export & Manufacture.....	??
Part 7: Fundamental Kicad how-to: Symbols and Eeschema.....	??
Introduction.....	??
Left menu bar overview.....	??
Top menu bar overview.....	??
Right menu bar overview.....	??
Schematic editor preferences.....	??
How to find a symbol with the Chooser.....	??
How to find schematic symbols on the Internet.....	??
How to install symbol libraries in bulk.....	??
How to create a custom symbol.....	??
How to associate a symbol with a footprint.....	??
Net labels.....	??
Net classes.....	??
Hierarchical sheets.....	??
Global labels.....	??
Hierarchical labels and import sheet pin.....	??
Electrical rules and customization.....	??
Bulk editing of schematic elements.....	??
Part 8: Fundamental Kicad how-to: Footprints and Pcbnew.....	??
Introduction.....	??
Left toolbar.....	??
Top toolbar.....	??
Top toolbar Row 1.....	??
Top toolbar Row 2.....	??
Right toolbar.....	??
Right toolbar main buttons.....	??
Right toolbar - Appearance.....	??
Layout editor preferences.....	??
Board Setup.....	??
Board Setup - Board Stackup.....	??
Board Setup - Text & Graphics.....	??
Board Setup - Design Rules and net classes.....	??
Board Setup - Design Rules - Custom Rules and violation severity.....	??
How to find and use a footprint.....	??
Footprint sources on the Internet.....	??
How to install footprint libraries.....	??
Filled zones.....	??

Keep-out zones.....	??
Interactive router.....	??
Length measuring tools.....	??
Bulk editing.....	??
Create a custom footprint, introduction.....	??
Create a new library and footprint.....	??
Create a footprint, 1, Fabrication layer.....	??
Create a footprint, 2, Pads.....	??
Create a footprint, 3, Courtyard layer.....	??
Create a footprint, 4, Silkscreen layer.....	??
Use the new footprint.....	??
Finding and using a 3D shape for a footprint.....	??
How to export and test Gerber files.....	??
Part 9: Project - Design a simple breadboard power supply PCB.....	96
1. Introduction.....	97
Schematic design editing.....	??
1 - Setup.....	??
2 - Symbols.....	??
2 - Edit Component values.....	??
3 - Arrange, Annotate.....	??
3 - Associate.....	??
4 - Wiring.....	??
5 & 6 - Nets and Electrical Rules Check.....	??
7 - Comments.....	??
Layout design editing.....	??
1 - Setup.....	??
2 - Outline and constraints.....	??
3 - Place footprints.....	??
2 - Refine the outline.....	??
4 - Route.....	??
5 - Copper fills.....	??
6 - Silkscreen.....	??
7 - Design Rules Check.....	??
8 - Export and Manufacture.....	??
Part 10: Project - A 4 x 8 x 8 LED matrix array.....	102
1. Introduction.....	103
Schematic design.....	??
1 - Setup.....	

Schematic design	
Schema 1 - Setup	
Schema 2 - Symbols	
Schema 3 - Arrange, Annotate	
Schema 3 - Associate	
Schema 4 - Wiring	
Schema 5 - Nets	
Schema 6 - Electrical Rules Check	
Schema 7 - Comments	
Schema - Last-minute edits	
Layout design	
Layout 1 - Setup	
Layout 2 - Outline and constraints	
Layout 3 - Place components	
Layout 2 supplemental - Refine outline	
Layout 3 supplemental - Move footprints to back layer	
Layout 4 - Route	
Layout 4 - Copper fills	
Layout 5 - Silkscreen	
Layout 6 - Design Rules Check	
Layout 7 - Manufacture	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
2 - Symbols.....	??
3 - Arrange, Annotate.....	??
3 - Associate.....	??
4 - Wiring.....	??
5 - Nets.....	??
6 - Electrical Rules Check.....	??
7 - Comments.....	??
Last-minute edits.....	??
Layout design editing.....	??
1 - Setup.....	??
2 - Outline and constraints.....	??
3 - Place components.....	??
2 - Refine outline.....	??
3 - Move footprints.....	??

4 - Route.....	??
4 - Copper fills.....	??
5 - Silkscreen.....	??
6 - Design Rules Check.....	??
7 - Manufacture.....	??
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
Bonus - Found a bug in the schematic! (and fix).....	??
The assembled and working PCB	
Schematic design	
Schema 1 - Setup	
Schema 2 - Symbols	
Schema 3 - Arrange, Annotate	
Schema 3 - Associate	
Schema 4 - Wiring	
Schema 5 - Nets	
Schema 6 - Electrical Rules Check	
Schema 7 - Comments	
Schema - Last-minute edits	
Layout design	
Layout 1 - Setup	
Layout 2 - Outline and constraints	
Layout 3 - Place components	
Layout 2 supplemental - Refine outline	
Layout 3 supplemental - Move footprints to back layer	
Layout 4 - Route	
Layout 4 - Copper fills	
Layout 5 - Silkscreen	
Layout 6 - Design Rules Check	
Layout 7 - Manufacture	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
Part 11 : Project - MCU datalogger.....	108
1. Project - Introduction.....	109
Create the new project and Git repository.....	??

Schematic design.....	??
Schema 1 - Setup.....	??
Schema 2 - Symbols.....	??
Schema 2 - Sheet two.....	??
Schema 3 - Arrange, Annotate.....	??
Edit component values.....	??
Schema 3 - Associate.....	??
Schema 4 - Wiring of sheet 1.....	??
Schema 4 - Wiring of sheet 2.....	??
Schema 5 - Nets.....	??
Schema 6 - Electrical Rules Check.....	??
Schema 7 - Comments.....	??
Create the 2-layer branch in Git.....	??
Layout design.....	??
Layout 1 - Setup.....	??
Layout 2 - Outline and constraints.....	??
Layout 3 - Place components.....	??
Layout 2 - Outline refinement.....	??
Layout 4 - Route.....	??
Layout 4 - Copper fills.....	??
Layout 4 - Routing improvements.....	??
Layout 5 - Silkscreen.....	??
Layout 4 - Routing violations and complete silkscreen.....	??
Layout 6 - Design Rules Check.....	??
Layout 7 - Manufacture.....	??
3D shapes.....	??
Merge 2-layer branch to main.....	??
Design 4 Layer PCB in new Git branch.....	??
Four-layer PCB routing.....	??
Four-layer PCB manufacturing.....	??
Updating layout from changes to the schematic with Git.....	??
Part 12 : Project - An ESP32 clone.....	??
1. Project - Introduction.....	113
Schematic design.....	??
Schema 1 - New KiCad project and Schematic Setup.....	??
Schema 2 - Symbols.....	??
Schema 3 - Annotate and set component values.....	??
Schema 3 - Arrange.....	??

Schema 3 - Associate.....	??
Schema 4 - Wiring.....	??
Schema 5 - Nets and Net Classes.....	??
Schema 6 - Electrical Rules Check.....	??
Schema 7 - Comments.....	??
Layout design.....	??
Layout 1 - Setup.....	??
Layout 2 - Outline and constraints.....	??
Layout 3 - Place components.....	??
Layout 2 supplemental - refine outline.....	??
Layout 4 - Route.....	??
Layout 4 - Copper fills and keep out areas.....	??
Layout 5 - Silkscreen.....	??
Layout 4 - Routing improvements.....	??
Layout 6 - Design Rules Check.....	??
Layout 7 - Manufacture.....	??
3D shapes.....	??
Part 13: Recipies.....	??
Create a custom silkscreen or copper graphic.....	??
Change a symbols and footprints in bulk.....	??
Change a symbol in bulk.....	??
Change a footprint in bulk.....	??
Interactive delete.....	??
Find and Replace (Eeschema).....	??
Edit Text & Graphics Properties.....	??
Edit Track & Via Properties (Pcbnew).....	??
Text variables.....	??
Board Setup - pre-defined sizes for tracks and vias.....	??
Board Setup - Design rules violation severity.....	??
Board Setup - Custom design rules.....	??
Schematic Setup - Electrical Rules and violation severity.....	??
Schematic Setup - Electrical Rules and Pin conflicts map.....	??
Field name templates.....	??
Bill of Materials.....	??
Build-in BOM in Pcbnew.....	??
Build-in BOM in Eeschema.....	??
A plug-in for BOM.....	??
Import components from Snapeda.....	??



The Freerouting autorouter.....	??
Install and start FreeRouting on MacOS.....	??
Install and start FreeRouting on Linux Kubuntu.....	??
Install and start FreeRouting on Windows.....	??
How to use the Freerouting autorouter 2-layer example.....	??
How to use the Freerouting autorouter 4-layer example.....	??
Pcbnew Inspection menu.....	??
Single track and differential pair routing.....	??
Track length tuning.....	??
Differential pair skew tuning.....	??
Interactive router modes.....	??
The footprint wizard.....	??
Pin and wire highlighter tool.....	??
Pcbnew Origins.....	??
KiCad project management with Git.....	??
Install Git.....	??
Git configuration.....	??
Create a new KiCad project Git repository.....	??
How to ignore files.....	??
Basic Git commands: add, commit.....	??
Basic Git commands: branch.....	??
Basic Git commands: merge.....	??
Sharing your KiCad project on GitHub.....	??
Customize the editor color scheme.....	??
Import an EAGLE, Altium, or Cadstar project.....	??
The circuit simulator.....	??
Prepare the circuit for simulation.....	??
Configure the simulator.....	??
Simulate.....	??
Import a KiCad 5 project.....	??
KiCad project templates.....	??
Using a system project template.....	??
Create a user project template.....	??
Archive/unarchive and share a project.....	??
Buses.....	??
Calculate the width of a trace.....	??
Design a custom schematic sheet.....	??

# Detailed Table of Contents

Introduction.....	29
1. What is a PCB?.....	30
2. The PCB design process.....	36
3. Fabrication.....	41
4. Get KiCad for your operating system.....	43
5. Example KiCad projects.....	47
Part 2: Getting started with KiCad 6.....	56
1. Introduction.....	57
2. KiCad Project Manager (main window).....	58
3. Overview of the individual KiCad apps.....	64
4. Paths and Libraries.....	72
5. Create a new project from scratch.....	76
6. Create a new project from a template.....	78
7. KiCad 6 on Mac OS, Linux, Windows.....	82
8. Differences between KiCad 6 and 5.....	87
Part 3: Project - A hands-on tour of KiCad - Schematic Design.....	89
1. Introduction to schematic design and objective of this section.....	90
Design workflows summary.....	??
The finished KiCad project and directory.....	??
Start Kicad and create a new project.....	??
1 - Start Eeschema, setup Sheet.....	??
2 - Add symbols.....	??
3 - Arrange, annotate, associate.....	??
4 - Wiring.....	??
5 - Nets.....	??
6 - The Electrical Rules Check.....	??
7 - Comments with text and graphics.....	??
Part 4: Project- A hands-on tour of KiCad - Layout.....	93
1. Introduction to layout design and objective of this section.....	94
1 - Start Pcbnew, import footprints.....	??
2 - Outline and constraints (edge cut).....	??
3 - Move footprints in place.....	??
4 - Route (add tracks).....	??
5 - Refine the outline.....	??
6 - Silkscreen (text and graphics).....	??

7 - Design rules check.....	??
8 - Export Gerbers and order.....	??
The manufactured PCB.....	??
Part 5: Design principles and PCB terms.....	??
Introduction.....	??
Schematic symbols.....	??
PCB key terms.....	??
FR4.....	??
Traces.....	??
Pads and holes.....	??
Via.....	??
Annular ring.....	??
Soldermask.....	??
Silkscreen.....	??
Drill bit and drill hit.....	??
Surface mounted devices.....	??
Gold Fingers.....	??
Keep-out areas.....	??
Panel.....	??
Solder paste and paste stencil.....	??
Pick-and-place.....	??
Part 6: PCB design workflows.....	??
The KiCad Schematic Design Workflow.....	??
Schematic Design Step 1: Setup.....	??
Schematic Design Step 2: Symbols.....	??
Schematic Design Step 3: AAA (Arrange, Annotate, Associate).....	??
Schematic Design Step 4: Wire.....	??
Schematic Design Step 5: Nets.....	??
Schematic Design Step 6: Electrical Rules Check.....	??
Schematic Design Step 7: Comments and Graphics.....	??
The KiCad Layout Design Workflow.....	??
Layout Design Step 1: Setup.....	??
Layout Design Step 2: Outline and constraints.....	??
Layout Design Step 3: Place footprints.....	??
Layout Design Step 4a: Route.....	??
Layout Design Step 4b: Copper fills.....	??
Layout Design Step 5: Silkscreen.....	??
Layout Design Step 6: Design rules check.....	??

Layout Design Step 7: Export & Manufacture.....	??
Part 7: Fundamental Kicad how-to: Symbols and Eeschema.....	??
Introduction.....	??
Left menu bar overview.....	??
Top menu bar overview.....	??
Right menu bar overview.....	??
Schematic editor preferences.....	??
How to find a symbol with the Chooser.....	??
How to find schematic symbols on the Internet.....	??
How to install symbol libraries in bulk.....	??
How to create a custom symbol.....	??
How to associate a symbol with a footprint.....	??
Net labels.....	??
Net classes.....	??
Hierarchical sheets.....	??
Global labels.....	??
Hierarchical labels and import sheet pin.....	??
Electrical rules and customization.....	??
Bulk editing of schematic elements.....	??
Part 8: Fundamental Kicad how-to: Footprints and Pcbnew.....	??
Introduction.....	??
Left toolbar.....	??
Top toolbar.....	??
Top toolbar Row 1.....	??
Top toolbar Row 2.....	??
Right toolbar.....	??
Right toolbar main buttons.....	??
Right toolbar - Appearance.....	??
Layout editor preferences.....	??
Board Setup.....	??
Board Setup - Board Stackup.....	??
Board Setup - Text & Graphics.....	??
Board Setup - Design Rules and net classes.....	??
Board Setup - Design Rules - Custom Rules and violation severity.....	??
How to find and use a footprint.....	??
Footprint sources on the Internet.....	??
How to install footprint libraries.....	??
Filled zones.....	??

Keep-out zones.....	??
Interactive router.....	??
Length measuring tools.....	??
Bulk editing.....	??
Create a custom footprint, introduction.....	??
Create a new library and footprint.....	??
Create a footprint, 1, Fabrication layer.....	??
Create a footprint, 2, Pads.....	??
Create a footprint, 3, Courtyard layer.....	??
Create a footprint, 4, Silkscreen layer.....	??
Use the new footprint.....	??
Finding and using a 3D shape for a footprint.....	??
How to export and test Gerber files.....	??
Part 9: Project - Design a simple breadboard power supply PCB.....	96
1. Introduction.....	97
Schematic design editing.....	??
1 - Setup.....	??
2 - Symbols.....	??
2 - Edit Component values.....	??
3 - Arrange, Annotate.....	??
3 - Associate.....	??
4 - Wiring.....	??
5 & 6 - Nets and Electrical Rules Check.....	??
7 - Comments.....	??
Layout design editing.....	??
1 - Setup.....	??
2 - Outline and constraints.....	??
3 - Place footprints.....	??
2 - Refine the outline.....	??
4 - Route.....	??
5 - Copper fills.....	??
6 - Silkscreen.....	??
7 - Design Rules Check.....	??
8 - Export and Manufacture.....	??
Part 10: Project - A 4 x 8 x 8 LED matrix array.....	102
1. Introduction.....	103
Schematic design.....	??
1 - Setup.....	

Schematic design	
Schema 1 - Setup	
Schema 2 - Symbols	
Schema 3 - Arrange, Annotate	
Schema 3 - Associate	
Schema 4 - Wiring	
Schema 5 - Nets	
Schema 6 - Electrical Rules Check	
Schema 7 - Comments	
Schema - Last-minute edits	
Layout design	
Layout 1 - Setup	
Layout 2 - Outline and constraints	
Layout 3 - Place components	
Layout 2 supplemental - Refine outline	
Layout 3 supplemental - Move footprints to back layer	
Layout 4 - Route	
Layout 4 - Copper fills	
Layout 5 - Silkscreen	
Layout 6 - Design Rules Check	
Layout 7 - Manufacture	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
2 - Symbols.....	??
3 - Arrange, Annotate.....	??
3 - Associate.....	??
4 - Wiring.....	??
5 - Nets.....	??
6 - Electrical Rules Check.....	??
7 - Comments.....	??
Last-minute edits.....	??
Layout design editing.....	??
1 - Setup.....	??
2 - Outline and constraints.....	??
3 - Place components.....	??
2 - Refine outline.....	??
3 - Move footprints.....	??

4 - Route.....	??
4 - Copper fills.....	??
5 - Silkscreen.....	??
6 - Design Rules Check.....	??
7 - Manufacture.....	??
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
Bonus - Found a bug in the schematic! (and fix).....	??
The assembled and working PCB	
Schema design	
Schema 1 - Setup	
Schema 2 - Symbols	
Schema 3 - Arrange, Annotate	
Schema 3 - Associate	
Schema 4 - Wiring	
Schema 5 - Nets	
Schema 6 - Electrical Rules Check	
Schema 7 - Comments	
Schema - Last-minute edits	
Layout design	
Layout 1 - Setup	
Layout 2 - Outline and constraints	
Layout 3 - Place components	
Layout 2 supplemental - Refine outline	
Layout 3 supplemental - Move footprints to back layer	
Layout 4 - Route	
Layout 4 - Copper fills	
Layout 5 - Silkscreen	
Layout 6 - Design Rules Check	
Layout 7 - Manufacture	
Bonus - 3D shapes	
Bonus - Found a bug in the schematic! (and fix)	
The assembled and working PCB.....	??
Part 11 : Project - MCU datalogger.....	108
1. Project - Introduction.....	109
Create the new project and Git repository.....	??

Schematic design.....	??
Schema 1 - Setup.....	??
Schema 2 - Symbols.....	??
Schema 2 - Sheet two.....	??
Schema 3 - Arrange, Annotate.....	??
Edit component values.....	??
Schema 3 - Associate.....	??
Schema 4 - Wiring of sheet 1.....	??
Schema 4 - Wiring of sheet 2.....	??
Schema 5 - Nets.....	??
Schema 6 - Electrical Rules Check.....	??
Schema 7 - Comments.....	??
Create the 2-layer branch in Git.....	??
Layout design.....	??
Layout 1 - Setup.....	??
Layout 2 - Outline and constraints.....	??
Layout 3 - Place components.....	??
Layout 2 - Outline refinement.....	??
Layout 4 - Route.....	??
Layout 4 - Copper fills.....	??
Layout 4 - Routing improvements.....	??
Layout 5 - Silkscreen.....	??
Layout 4 - Routing violations and complete silkscreen.....	??
Layout 6 - Design Rules Check.....	??
Layout 7 - Manufacture.....	??
3D shapes.....	??
Merge 2-layer branch to main.....	??
Design 4 Layer PCB in new Git branch.....	??
Four-layer PCB routing.....	??
Four-layer PCB manufacturing.....	??
Updating layout from changes to the schematic with Git.....	??
Part 12 : Project - An ESP32 clone.....	??
1. Project - Introduction.....	113
Schematic design.....	??
Schema 1 - New KiCad project and Schematic Setup.....	??
Schema 2 - Symbols.....	??
Schema 3 - Annotate and set component values.....	??
Schema 3 - Arrange.....	??



Schema 3 - Associate.....	??
Schema 4 - Wiring.....	??
Schema 5 - Nets and Net Classes.....	??
Schema 6 - Electrical Rules Check.....	??
Schema 7 - Comments.....	??
Layout design.....	??
Layout 1 - Setup.....	??
Layout 2 - Outline and constraints.....	??
Layout 3 - Place components.....	??
Layout 2 supplemental - refine outline.....	??
Layout 4 - Route.....	??
Layout 4 - Copper fills and keep out areas.....	??
Layout 5 - Silkscreen.....	??
Layout 4 - Routing improvements.....	??
Layout 6 - Design Rules Check.....	??
Layout 7 - Manufacture.....	??
3D shapes.....	??
Part 13: Recipies.....	??
Create a custom silkscreen or copper graphic.....	??
Change a symbols and footprints in bulk.....	??
Change a symbol in bulk.....	??
Change a footprint in bulk.....	??
Interactive delete.....	??
Find and Replace (Eeschema).....	??
Edit Text & Graphics Properties.....	??
Edit Track & Via Properties (Pcbnew).....	??
Text variables.....	??
Board Setup - pre-defined sizes for tracks and vias.....	??
Board Setup - Design rules violation severity.....	??
Board Setup - Custom design rules.....	??
Schematic Setup - Electrical Rules and violation severity.....	??
Schematic Setup - Electrical Rules and Pin conflicts map.....	??
Field name templates.....	??
Bill of Materials.....	??
Build-in BOM in Pcbnew.....	??
Build-in BOM in Eeschema.....	??
A plug-in for BOM.....	??
Import components from Snapeda.....	??

The Freerouting autorouter.....	??
Install and start FreeRouting on MacOS.....	??
Install and start FreeRouting on Linux Kubuntu.....	??
Install and start FreeRouting on Windows.....	??
How to use the Freerouting autorouter 2-layer example.....	??
How to use the Freerouting autorouter 4-layer example.....	??
Pcbnew Inspection menu.....	??
Single track and differential pair routing.....	??
Track length tuning.....	??
Differential pair skew tuning.....	??
Interactive router modes.....	??
The footprint wizard.....	??
Pin and wire highlighter tool.....	??
Pcbnew Origins.....	??
KiCad project management with Git.....	??
Install Git.....	??
Git configuration.....	??
Create a new KiCad project Git repository.....	??
How to ignore files.....	??
Basic Git commands: add, commit.....	??
Basic Git commands: branch.....	??
Basic Git commands: merge.....	??
Sharing your KiCad project on GitHub.....	??
Customize the editor color scheme.....	??
Import an EAGLE, Altium, or Cadstar project.....	??
The circuit simulator.....	??
Prepare the circuit for simulation.....	??
Configure the simulator.....	??
Simulate.....	??
Import a KiCad 5 project.....	??
KiCad project templates.....	??
Using a system project template.....	??
Create a user project template.....	??
Archive/unarchive and share a project.....	??
Buses.....	??
Calculate the width of a trace.....	??
Design a custom schematic sheet.....	??

## 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](#), 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 meetups. 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](#) and [Octopart](#), 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 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](#) 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](#) looks just as exciting.

**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](#) 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.

Over the last seven years, 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!

# Introduction



# 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 wondrous things: resistors, transformers, integrated circuits, coils, and power supplies.

Engineers had attached those things on small green boards, like the one in Figure 1.1.1. This is an example of a printed circuit board, or PCB, for short.

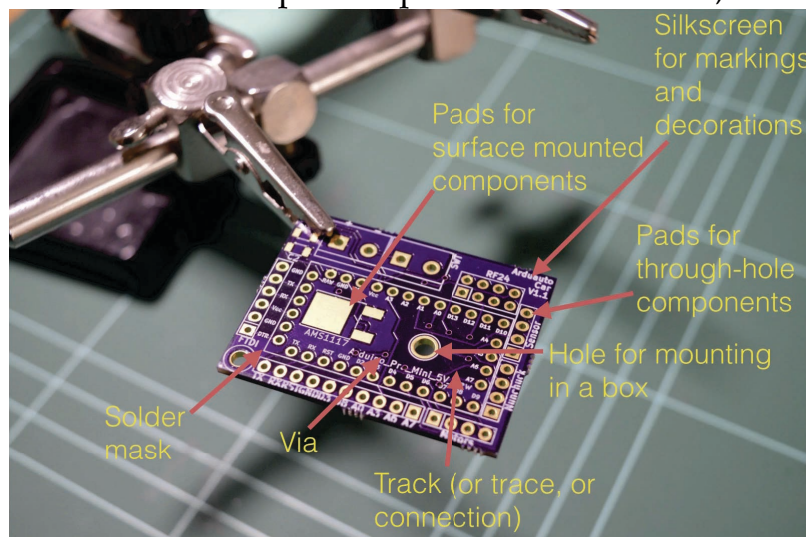


Figure 1.1.1: The top side of a printed circuit board.

Let's 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 1.1.1).

The top side of the PCB is the side where we place the components. We can place components on the bottom side, too.

In general, there are two kinds of components: through-hole or surface-mounted components. We can attach through-hole components on the PCB by inserting the leads or the pins through small holes and using hot solder to hold them in place. In the example pictured in Figure 1.1.1, you can see

several holes to insert the through-hole component pins. The holes extend from the top side to the bottom side of the PCB and are plated with a conductive material. This material is usually tin, or as in the case of the board in the image, gold. We use solder to attach and secure a component through its lead onto the pad surrounding the hole (Figure 1.1.2).

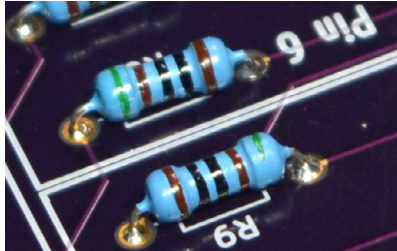


Figure 1.1.2: A through-hole component attached to a PCB.

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 1.1.3).



Figure 1.1.3: 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 helpful information about the board and its components. The graphics can include logos, other decorations, and useful markings.

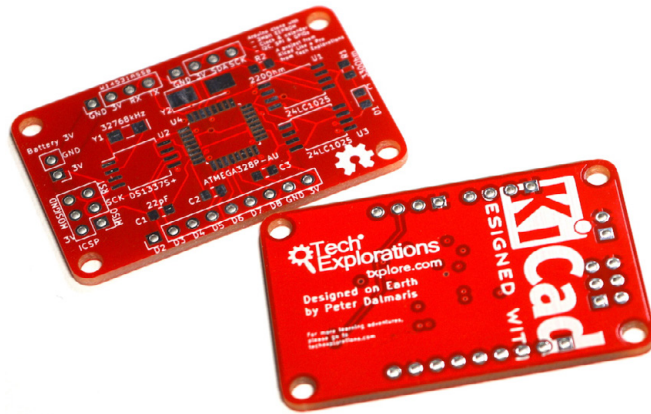


Figure 1.1.4: The white letters and lines is the silkscreen print on this PCB.

In Figure 1.1.4, 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 they don't need to be tinned. You can use a screw with a nut and bolt on the other side to secure the PCB inside a box.

Next are the tracks. In this example (Figure 1.1.5), they look red because of the color of the masking chemical used by the manufacturer.

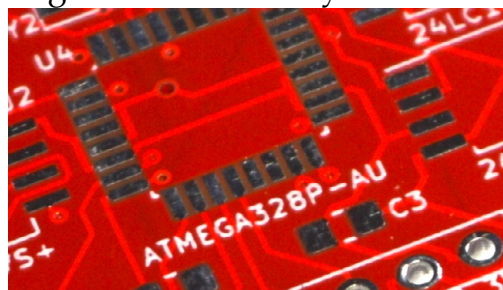


Figure 1.1.5: The bright red lines connecting the holes are tracks.

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. You can also refer to a "track" as a "trace."

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

The red 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 oxidized over time. The oxidization of the copper tracks negatively affects their conductivity. The solder mask prevents oxidization.

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 complicated 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 1.1.6). The solder mask prevents bridges because the solder cannot bond with it.

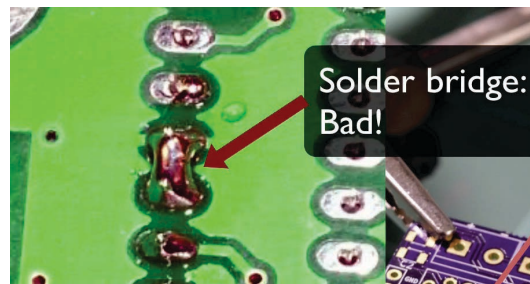


Figure 1.1.6: A solder bridge like this one is a defect that a solder mask can prevent.

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 a solder mask helps in preventing that from happening.

In Figure 1.1.7 you can see an example of the standard 1.6mm thick PCB.

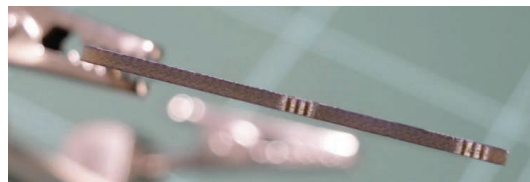


Figure 1.1.7: This PCB has a thickness of 1.6mm, and is made of fiberglass.

Typically, PCBs are made of fiberglass. The typical thickness of the PCB is 1.6 millimeters. In this closeup view of a PCB picture (Figure 1.1.8), 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 tinned on the inside, electrically connecting the front and back.



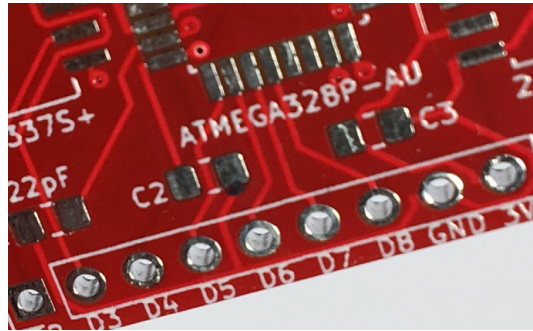


Figure 1.1.8: A closeup view of the top layer.

In Figure 1.1.8, you can see several vias (the small holes) and tracks, the red solder mask, and the solder mask between the pads. In this closeup, 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 1.1.9 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.

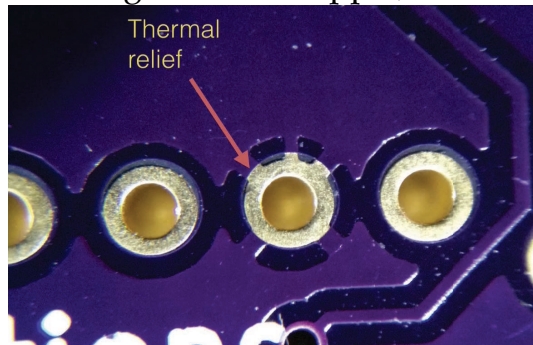


Figure 1.1.9: Thermal relief connects a pad to a copper region.

In Figure 1.1.9, 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.' Thermal reliefs make it easier to solder because the soldering heat won't dissipate into the large copper area.

Figure 1.1.10 gives a different perspective that allows us to appreciate the thickness of the tracks.

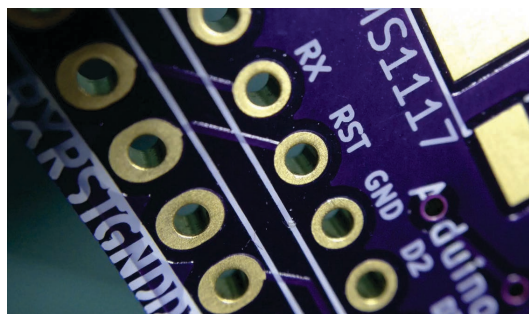


Figure 1.1.10: The plating of the holes covers the inside of the hole and connects that front end with the back end.

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 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 to reduce manufacturing costs.

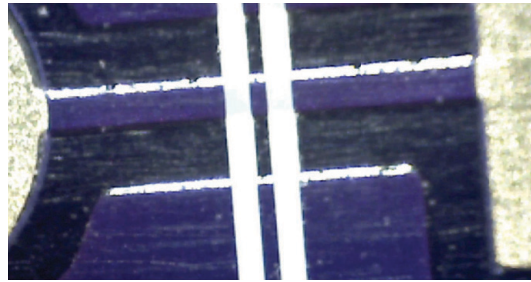


Figure 1.1.11: A detail of this example board at 200 times magnification.

The image in Figure 1.1.11 was taken at 200 times magnification. You can see a track that connects two pads and the light that reflects off one side of the track.

## 2. The PCB design process

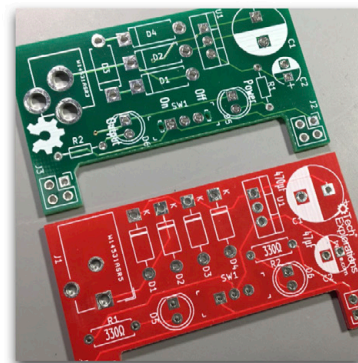
To design a printed circuit board, you have to complete several steps, make decisions, and iterate until you are satisfied with the result.

A printed circuit board is a physical device that takes time and money to manufacture. It must be fit to perform its intended purpose, and must be manufacturable. Therefore, your design must be of high quality, safe, and possible to manufacture by your chosen manufacturer.

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.

### The PCB design process

- Designing a PCB involves:
  - Several steps
  - Decisions
  - Iteration
- The result should be
  - Functional
  - High quality
  - Manufacturable
  - Beautiful



Kicad Like a Pro 3e  
txplo.re/kicadr

Tech  
Explorations

Figure 1.2.1: Some considerations of the PCB design process.

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.

## The Kicad design workflow

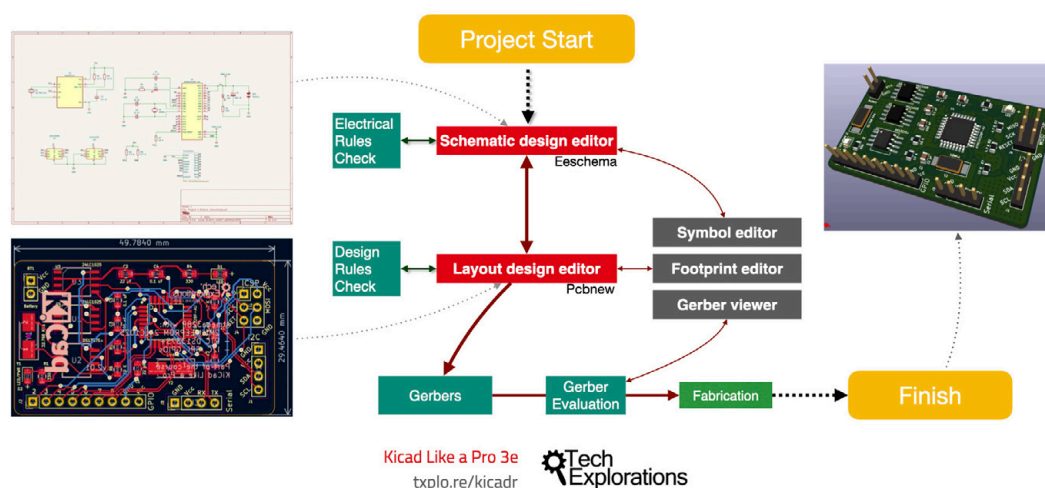


Figure 1.2.2: KiCad is a suite of applications.

KiCad is not a single application. It is a suite of apps that work together to help you create printed circuit boards. As a result, it is possible to customise the PCB design process to suit your particular style and habits. But when you are just starting up, I think it is helpful to provide a workflow that you can use as a model.

In Figure 1.2.2 you can see my KiCad PCB design workflow model. You can use it as it is, or you can modify as you see fit. I distilled this workflow 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.

In this book, I will be following this PCB design workflow in all of the projects.

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



1. Step 1 is the schematic design using the schematic design editor (Eeschema);
2. Step 2 is the layout design using the layout editor (Pcbnew).

Once you have the layout design, you can export it and the manufacture it.

The goal of the schematic design step is to capture information about the circuit that will be implemented in the final PCB. Once you have a schematic design, you can use the layout editor to create a version of the PCB. Remember, a schematic design can have many different layouts.

## The Kicad design workflow

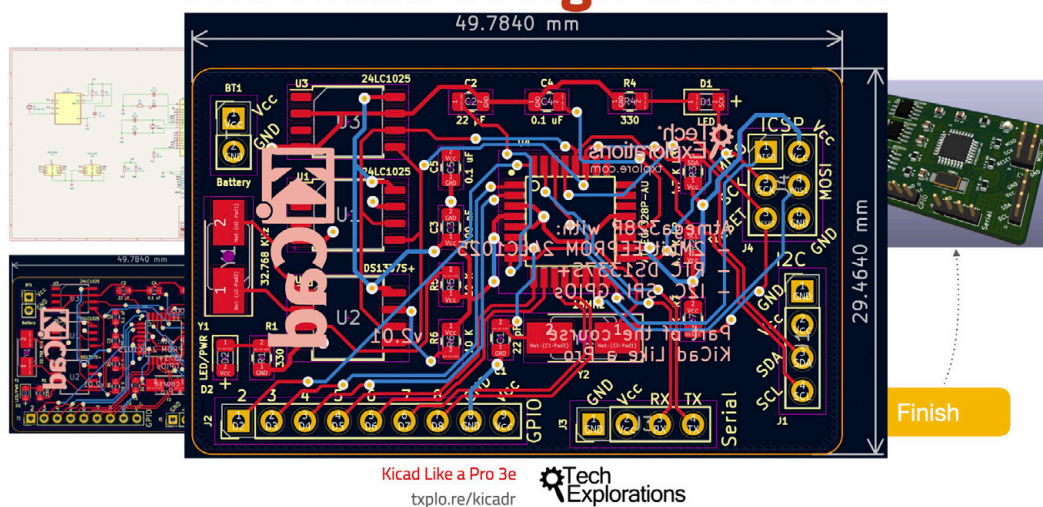


Figure 1.2.3: The KiCad layout file contains information about the physical PCB.

The KiCad layout file 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).

Let's walk through the workflow now using the diagram in Figure 1.2.3 as an aid.

For the discussion that follows, keep in mind these definitions:

- A symbol is a symbolic representation of a real component in the schematic; a symbol represents a component's function, not its physical appearance or location in the final PCB.

- A footprint is a graphical depiction of a real component in the layout. It relates directly to a real physical counterpart. It contains information about the real component's location and dimensions.

In this book, I will be using the terms “symbols” and footprints according to the definitions above.

In KiCad, the process begins with Eeschema, which is the schematic editor.

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 search for it on the Internet, or create yourself with the help of the schematic library editor.

Running regular electrical rules checks helps to detect defects early. Eeschema has a built-in checker utility for this purpose.

Pcbnew (KiCad's layout editor) has its own validator, the Design Rules Checker.

These two utilities help to produce PCBs that have a low risk to contain design or electrical defects.

Before you finish work in Eeschema and continue with the layout, you must first associate the schematic symbols with layout footprints.

In KiCad 6, many symbols come with preset symbol to footprint associations, but many don't, so you'll have to do this yourself. Also keep in mind that, as I said earlier, KiCad is very flexible. It is possible to assign many different footprints to the same schematic symbol (one at a time, of course).

Once you have completed the Electrical Rules Check and symbol to footprint associations, you can continue with layout using the KiCad Layout design editor, or Pcbnew.

You use Pcbnew to position the footprints on the sheet and connect the footprint pins using wires. You'll also add an outline that marks the outer limit of the PCB, and other design elements like mounting holes, logos, and instructional text.

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.

Let's look at some of the PCB terminology before we continue.

# The Kicad design workflow

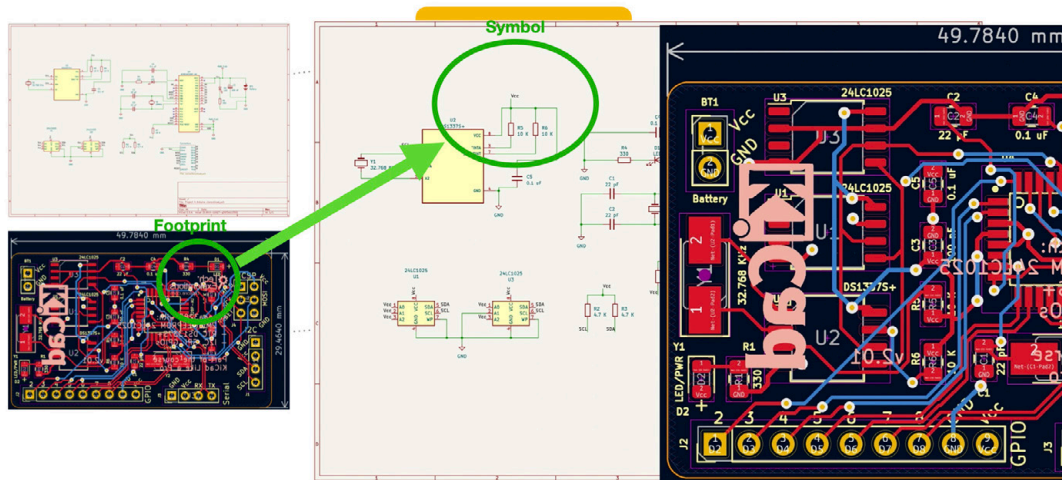


Figure 1.2.4: Symbols and footprints.

As you know, a symbol is a symbolic representation of a real component in the schematic. A footprint is a graphical depiction of a real component in the layout.

You, as the designer, must tell KiCad which footprint you want to use in your PCB by associating it to a particular symbol. Take the example of a resistor. A resistor uses a specific symbol in the schematic, but on the PCB it can be realized as a through-hole or SMD device of varying sizes.

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. Fabrication

Imagine that you have finished 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](#).

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 how 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](#), [NextPCB](#), and [OSHPark](#) are very good at what they offer.

You can get a professionally made PCB for 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 an extensive array of manufacturing options) extensively. I'm always happy with the result. Using an online manufacturer takes a little bit of planning because once you order your PCBs, it can take up to several weeks 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.

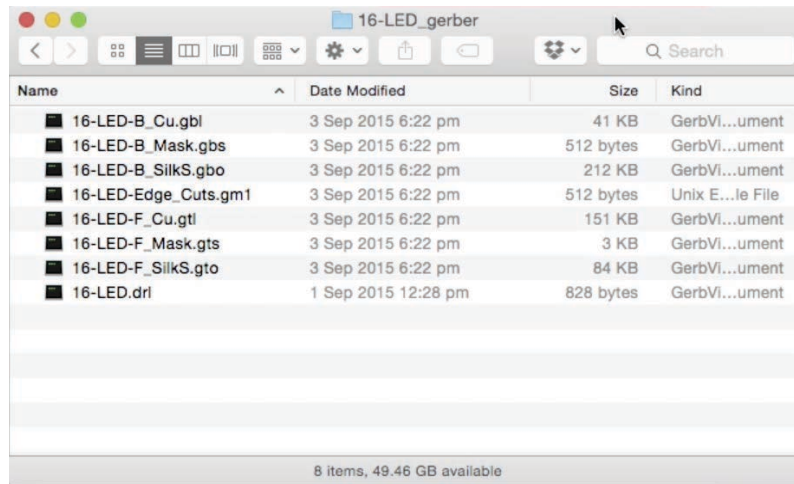


Figure 3.3.1: An example of the Gerber files that the manufacturer will need in order to make your PCB.

Now, let's turn our attention to the files 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 3.3.2 shows the contents of an example Gerber file.

```

1 G04 #0! TF.FileFunction,Soldermask,Bot*
2 %FSLAX46Y46*%
3 G04 Gerber Fmt 4.6, Leading zero omitted, Abs format (unit mm)*
4 G04 Created by KiCad (PCBNEW (2015-07-06 BZR 5891, Git 351914d)-product) date
5 03/09/2015 18:22:08*
6 %MOMM*%
7 G01*
8 G04 APERTURE LIST*
9 %ADD10C,0.100000*%
10 %ADD11R,1.727200X2.032000*%
11 G04 APERTURE END LIST*
12 D10*
13 D11*
14 X122555000Y-42545000D03*
15 D12*
16 X120015000Y-42545000D03*
17 X117475000Y-42545000D03*
18 X114935000Y-42545000D03*
19 X112395000Y-42545000D03*
20 M02*
21

```

Figure 3.3.3: 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 to maintain your project history and store and share via online repositories like [Github](#).

[Ucamco](#) has designed the Gerber files system and standard. They make equipment and write software for PCB manufacturers — things like PreCAM software, PCB CAM, laser photoplotters, and direct imaging systems. If you're curious about how to read these Gerber files, you can look up the Gerber format specification on [Ucamco's website](#).

## 4. Get KiCad for your operating system

It is now time to download your copy of KiCad and install it on your computer.

Kicad has support for a variety of operating systems. The major operating systems, Mac OS and Windows, are supported. Of course, there is support for Ubuntu and a lot of different flavors of Linux. I have tested, and I frequently use KiCad on Mac OS. Mac OS is my primary operating system, but I'm also working on Windows and [Kubuntu](#) instead of [Ubuntu](#).

Kubuntu is based on Ubuntu in its core but uses the KDE Desktop and related software. I find Kubuntu to offer a much better experience compared to Ubuntu. Of course, this is my personal preference, and opinions vary greatly.

I'm not going to show you how to install KiCad on each one of those operating systems. The KiCad developer team has refined the installer over the years. The KiCad installation process on the supported operating systems is just like that of any other refined application.

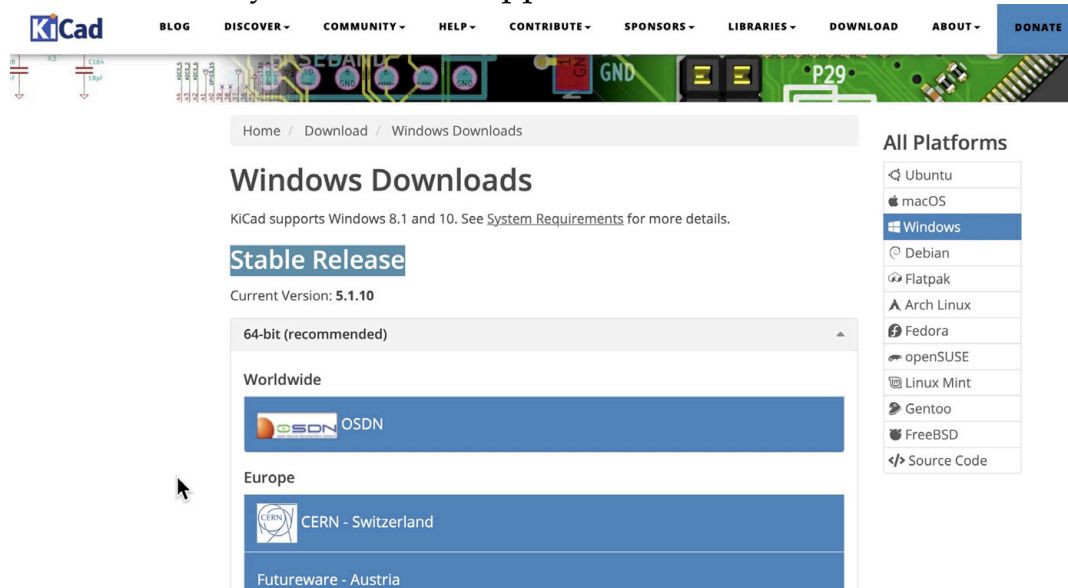



Figure 1.4.1: Windows stable release download page.

For example, to get the KiCad installer for Windows, go to the [KiCad Windows page](#) and download the stable version of KiCad from your preferred source. Double-click on the installer icon and follow the installation wizard instructions to complete the installation on your computer.





[BLOG](#)
[DISCOVER](#)
[COMMUNITY](#)
[HELP](#)
[CONTRIBUTE](#)
[SPONSORS](#)
[LIBRARIES](#)
[DOWNLOAD](#)

---

The nightly build KiCad signature is:

Signer Name	KiCad Services Corporation
Issuer	Sectigo RSA Code Signing CA
Serial Number	1f70b098b5c21a254a6fb427cdf8893e

## Previous Releases

Previous releases should be available for download on:

<https://kicad-downloads.s3.cern.ch/index.html?prefix=windows/stable/>


## Testing Builds

The *testing* builds are snapshots of the current stable release codebase at a specific time. These contain the most recent bugfixes that will be included in the next stable release.

<https://kicad-downloads.s3.cern.ch/index.html?prefix=windows/testing/5.1/>

## Nightly Development Builds

The *nightly development* builds are snapshots of the development (master branch) codebase at a specific time. This codebase is under active development, and while we try our best, may contain more bugs than usual. New features added to KiCad can be tested in these builds.



These builds may be unstable, and projects edited with these are not usable with the current stable release. **Use at your own risk**

<https://kicad-downloads.s3.cern.ch/index.html?prefix=windows/nightly/>

Figure 1.4.2: Nightly build download

You can download and install the latest available version of KiCad that is available as a nightly build. Nightly builds are work-in-progress. They contain the latest code committed by the KiCad developers but are considered "unstable." Therefore, you should not use it for work that you do not want to lose. The major operating systems have a nightly build generated (almost) every night. If you want to look at the cutting-edge version of KiCad and you are not afraid of weird behaviors and strange crashes, then go to the nightly releases page for your preferred operating system and download the installer. Example: [Windows](#).

<https://kicad-downloads.s3.cern.ch/osx/nightly/>

Last Modified	Size	Key
-----		
		../
2021-07-18T23:47:27.451Z	1.3 GB	<a href="#">kicad-unified-20210718-154855-4c457b5ed3.dmg</a>
2021-07-19T11:03:08.964Z	1.3 GB	<a href="#">kicad-unified-20210719-030141-1c1f7ac07e.dmg</a>
2021-07-20T11:09:17.743Z	1.4 GB	<a href="#">kicad-unified-20210720-030631-75190370dd.dmg</a>
2021-07-22T00:03:14.338Z	1.3 GB	<a href="#">kicad-unified-20210721-155825-0fb864d596.dmg</a>
2021-07-22T11:27:56.241Z	1.3 GB	<a href="#">kicad-unified-20210722-031619-1a301d8eea.dmg</a>
2021-07-23T00:08:47.211Z	1.3 GB	<a href="#">kicad-unified-20210722-154659-8d1dd1f8b0.dmg</a>
2021-07-23T23:46:13.759Z	1.3 GB	<a href="#">kicad-unified-20210723-154243-3c1af1af74.dmg</a>
2021-07-24T11:20:29.721Z	1.3 GB	<a href="#">kicad-unified-20210724-031709-3c1af1af74.dmg</a>
2021-07-24T23:57:44.486Z	1.3 GB	<a href="#">kicad-unified-20210724-155639-728b160719.dmg</a>
2021-07-25T11:03:46.312Z	1.4 GB	<a href="#">kicad-unified-20210725-030257-728b160719.dmg</a>
2021-07-25T23:57:21.176Z	1.3 GB	<a href="#">kicad-unified-20210725-155531-13a03f77d3.dmg</a>
2021-07-26T23:42:57.971Z	1.3 GB	<a href="#">kicad-unified-20210726-154229-8fd83cbb95.dmg</a>
2021-07-27T11:08:39.206Z	1.3 GB	<a href="#">kicad-unified-20210727-030638-c946070005.dmg</a>
2021-07-27T23:46:28.141Z	1.3 GB	<a href="#">kicad-unified-20210727-154317-43cb710297.dmg</a>
2021-07-28T11:08:28.471Z	1.3 GB	<a href="#">kicad-unified-20210728-030651-11becc5a68.dmg</a>
2021-07-29T00:10:00.570Z	1.3 GB	<a href="#">kicad-unified-20210728-155536-befd30a1a1.dmg</a>
2021-07-29T11:12:28.102Z	1.3 GB	<a href="#">kicad-unified-20210729-030932-46338403e7.dmg</a>
2021-07-29T23:50:41.678Z	1.4 GB	<a href="#">kicad-unified-20210729-154718-c716548b29.dmg</a>
2021-07-30T11:13:19.294Z	1.3 GB	<a href="#">kicad-unified-20210730-030602-baf6798695.dmg</a>
2021-07-31T11:21:37.894Z	1.4 GB	<a href="#">kicad-unified-20210731-030903-9a9a155d67.dmg</a>
2021-07-31T23:52:59.659Z	1.4 GB	<a href="#">kicad-unified-20210731-154912-878538abff.dmg</a>
2021-08-01T11:10:37.543Z	1.3 GB	<a href="#">kicad-unified-20210801-030923-878538abff.dmg</a>

Figure 1.4.3: Nightly build download

If you're working on Mac OS, go to the [Mac OS downloads page](#) and download the latest available stable release. You can also [download a nightly build](#) if you are comfortable with the inherent risk. Both stable and nightly builds come as a regular [DMG file](#). The download file contains the entire KiCad suite with all its applications, the documentation, and the libraries for the schematic symbols, footprints, and templates. It also includes several demos projects.

The installation process makes use of Ubuntu's apt-get system. For Ubuntu, you can find installations instructions on the [Ubuntu page](#). For using nightly development builds in Ubuntu, you will find instructions on the same page.

There is there are similar instructions for the various other operating systems like [Suse](#) and [Fedora](#).

You also have the option to [download the source code](#) and build from the source on your operating system. This is not something that I usually do unless I want to play around with it and experiment. Luckily, the operating systems I use or have excellent binary builds, so I never needed to build my KiCad instance from the source. But if you are someone who enjoys doing that, then go to the [source code page](#) and follow the detailed instructions.

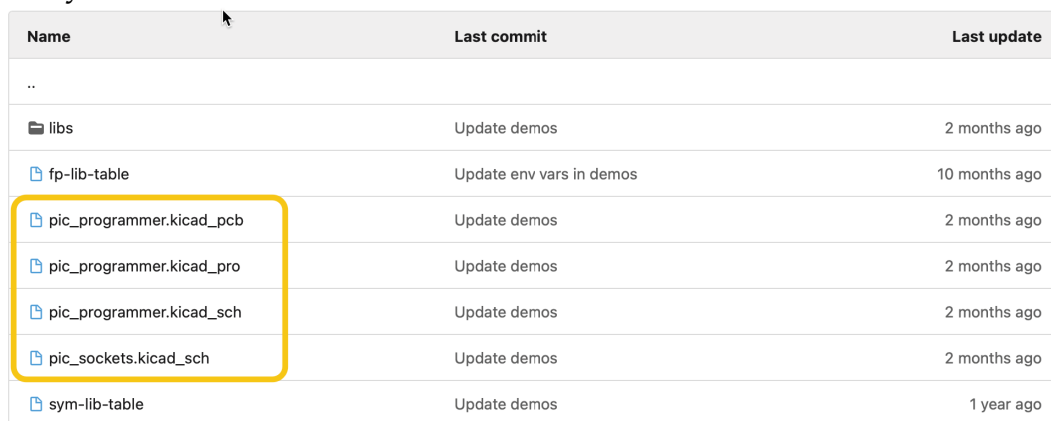
At this point, I invite you to download the version of KiCad that is suitable for your operating system and install KiCad on your computer. Once you finish installing KiCad, verify that it's up and running by starting KiCad.



In the next chapter, you will use your brand new instance of KiCad to look at some of the demo projects that ship with KiCad.

## 5. Example 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 download the one titled 'pic\_programmer' (Figure 1.5.1). You can also download the entire “demos” folder if you wish.



Name	Last commit	Last update
..		
libs	Update demos	2 months ago
fp-lib-table	Update env vars in demos	10 months ago
pic_programmer.kicad_pcb	Update demos	2 months ago
pic_programmer.kicad_pro	Update demos	2 months ago
pic_programmer.kicad_sch	Update demos	2 months ago
pic_sockets.kicad_sch	Update demos	2 months ago
sym-lib-table	Update demos	1 year ago

Figure 1.5.1: The contents of the 'pic\_programmer' demo project folder.

The demo project folder contains several files that make up the project. For now, the ones to focus on have the extensions 'kicad\_pro', 'kicad\_pcb' and 'kicad\_sch.' The file with the 'kicad\_pro' extension contains project information. The 'kicad\_pcb' file contains layout information. The files with the 'kicad\_sch' extension contain schematic information. There are two 'kicad\_sch' files because this project includes two schematics.

Double-click on the 'kicad\_pro' (project) file. The main KiCad window will appear. This window is the launchpad for the other KiCad apps, like Eeschema (the schematic editor) and Pcbnew (the layout editor). You can see the main KiCad window in Figure 1.5.2.

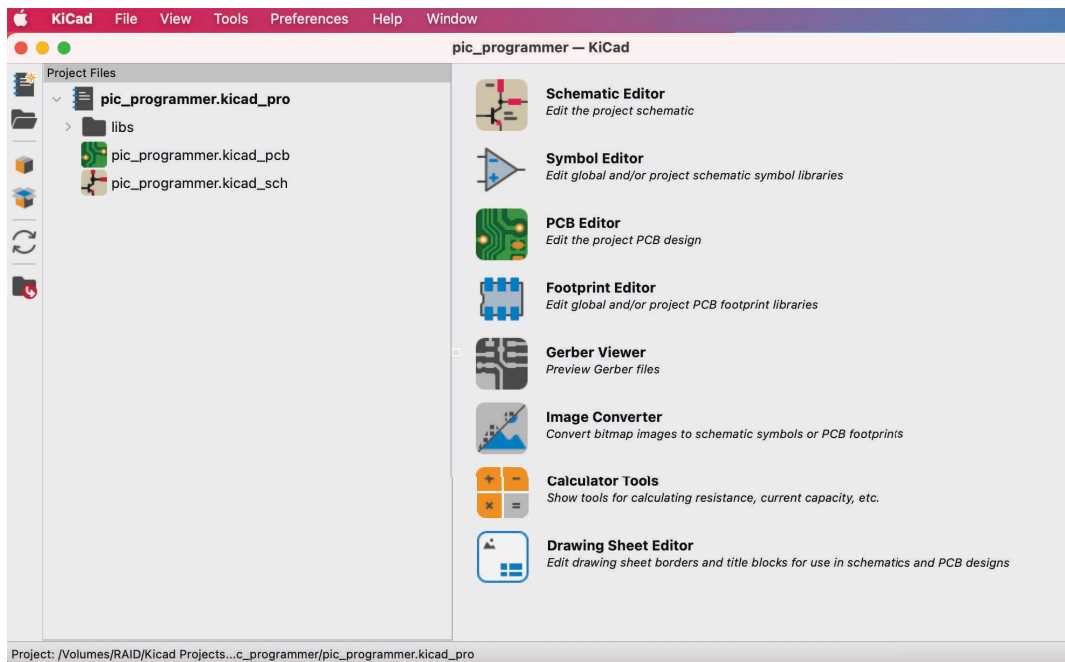


Figure 1.5.2: The main KiCad window.

Let's explore the schematic of this demo project. 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. In the right pane, click on the Schematic Editor button. This button will start the Eeschema application, the schematic layout editor. You should see the editor as in the example in Figure 1.5.3.

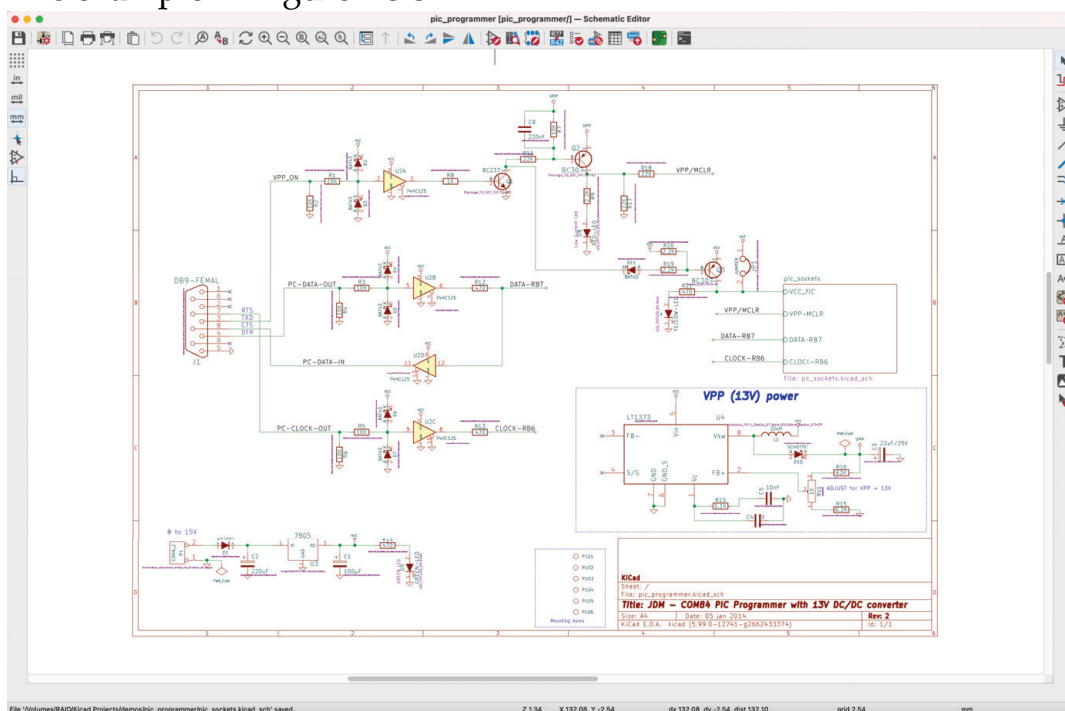


Figure 1.5.3: The schematic editor.

A few things are going on here. At first, this window might seem overwhelming. Don't worry about the various buttons and menus; 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 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's.

There is a rectangular symbol on the right side of the schematic with the title 'pic\_sockets' (Figure 1.5.4).

Double click on it. What happened?

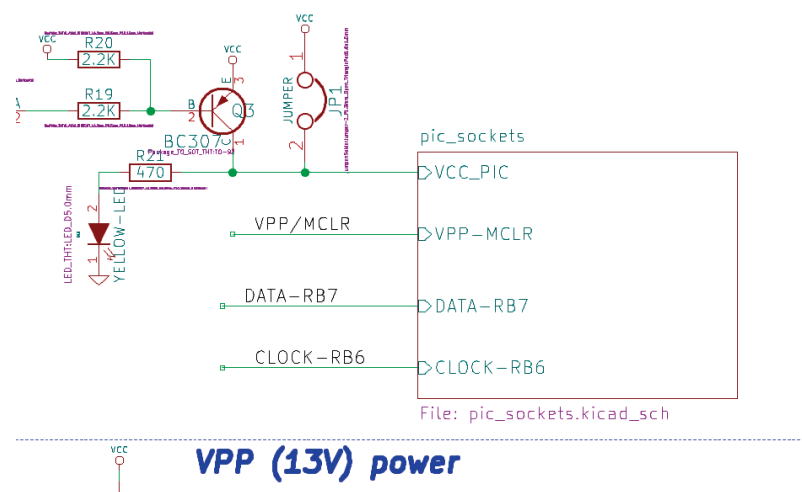


Figure 1.5.4: A link to another sheet.

This symbol links to another sheet, which contains additional symbols that are part of the same schematic. It looks like the example in Figure 1.5.5.

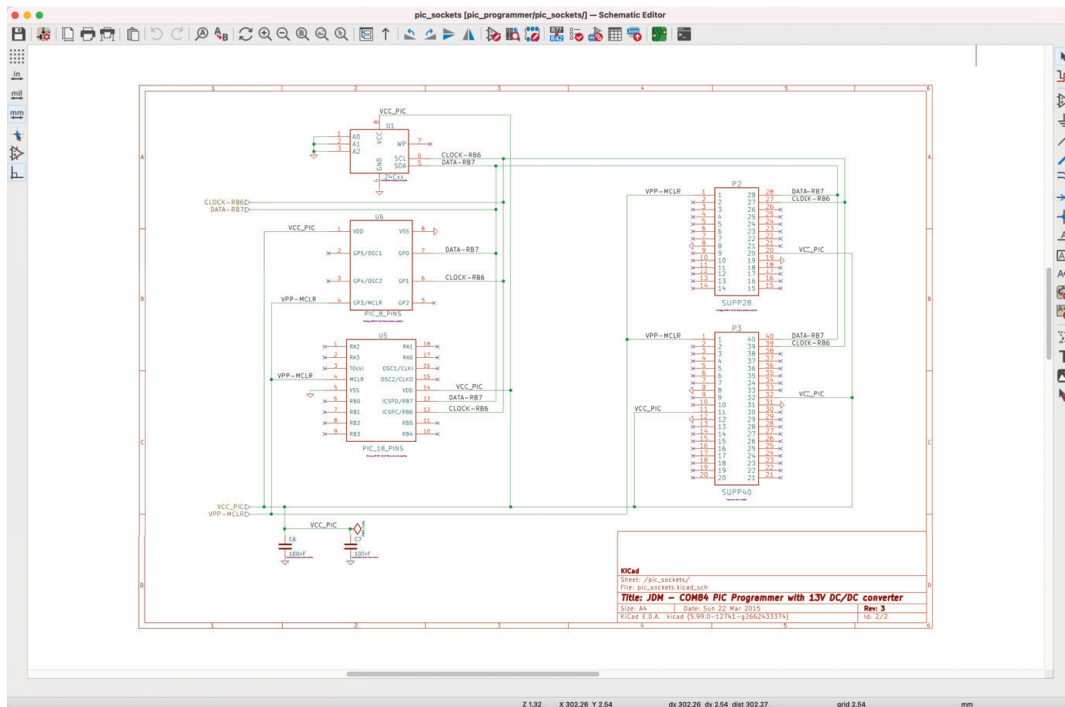


Figure 1.5.5: KiCad's schematics can span over multiple sheets.

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

I encourage you to spend a bit of time studying this schematic. You can learn a lot about drawing 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 button labeled "PCB Editor." This will launch Pcbnew, the layout editor. The window that appears will look like the example in Figure 1.5.6.

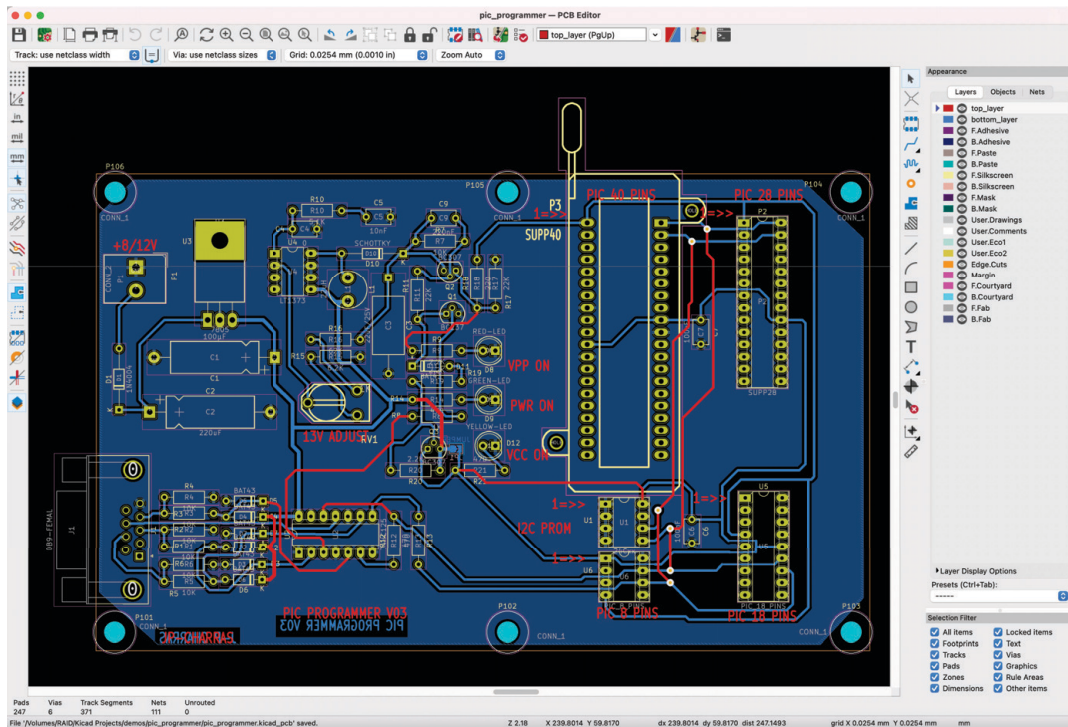


Figure 1.1.5.6: Pcbnew, the layout editor.

Again, don't worry about the various buttons and menus; concentrate on the layout inside the sheet. Use your mouse's scroll wheel to zoom in and out and the Alt+right mouse button to pan (you should also be able to pan by holding down the middle mouse button). 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 colors of the front copper and back copper layer traces. Note: in Linux, panning is done with the middle mouse button, and the alt key is not used.

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 1.5.7.

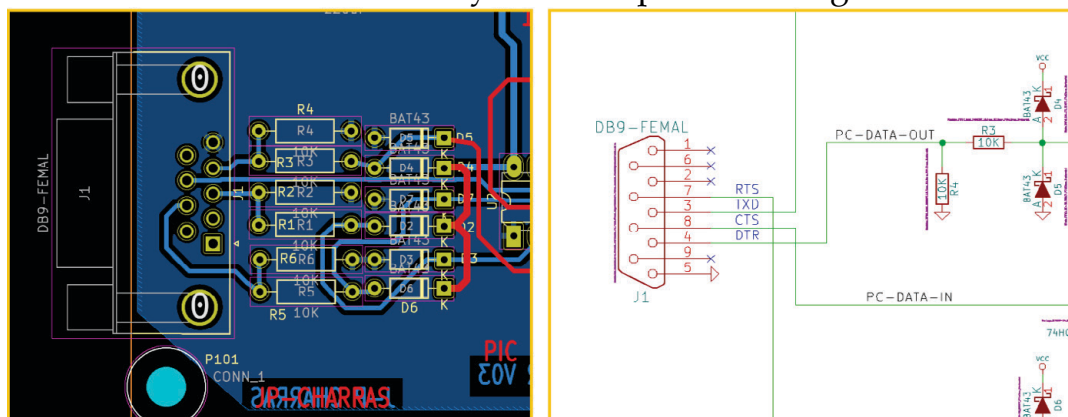


Figure 1.5.7: A side-by-side comparison of a footprint (left) and its schematic symbol (right).

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 on 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.

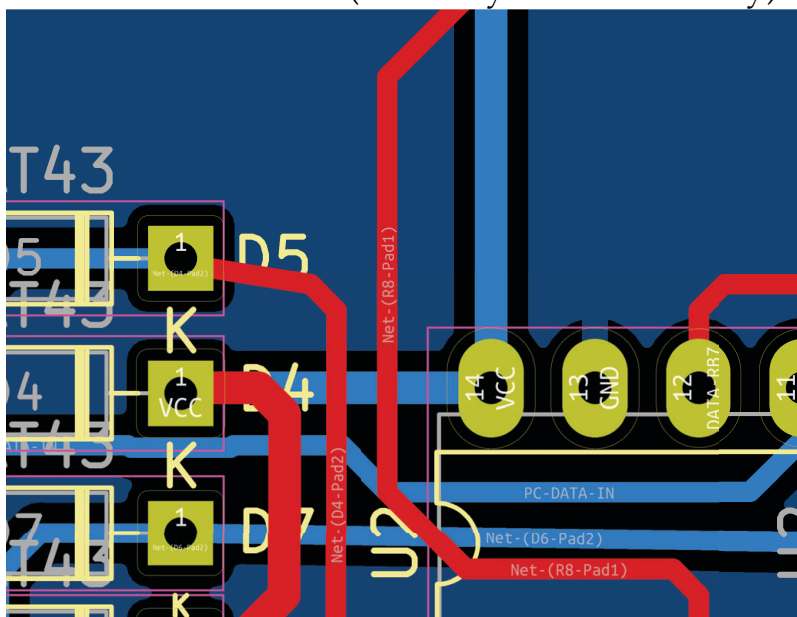


Figure 1.5.8: Traces have names.

Try one more thing: In Pcbnew, click on the View menu and choose the 3D Viewer. The 3D Viewer will show you a three-dimensional rendering of the PCB, with remarkable detail. You can zoom in and turn the board around to see it from any angle you want (Figure 1.5.9). Many 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.

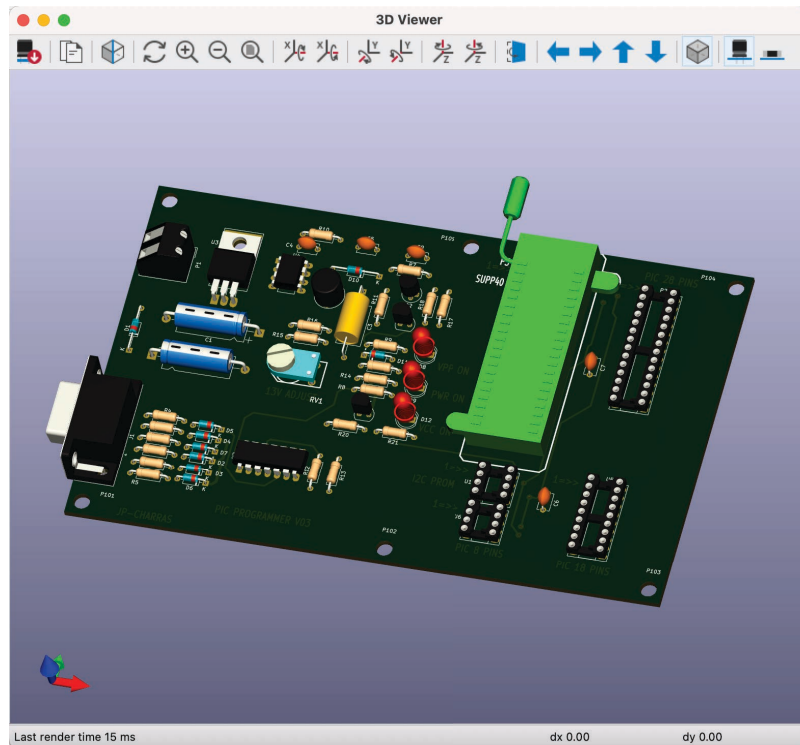


Figure 1.5.9: The 3D viewer will give you a realistic rendering of your board that you can examine in 3D.

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

Apart from the demo projects that KiCad ships with, you should also look at some of the very impressive showcased projects of boards “[made with 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. Go to [txplo.re/madewkicad](http://txplo.re/madewkicad) for more examples of projects made with KiCad.



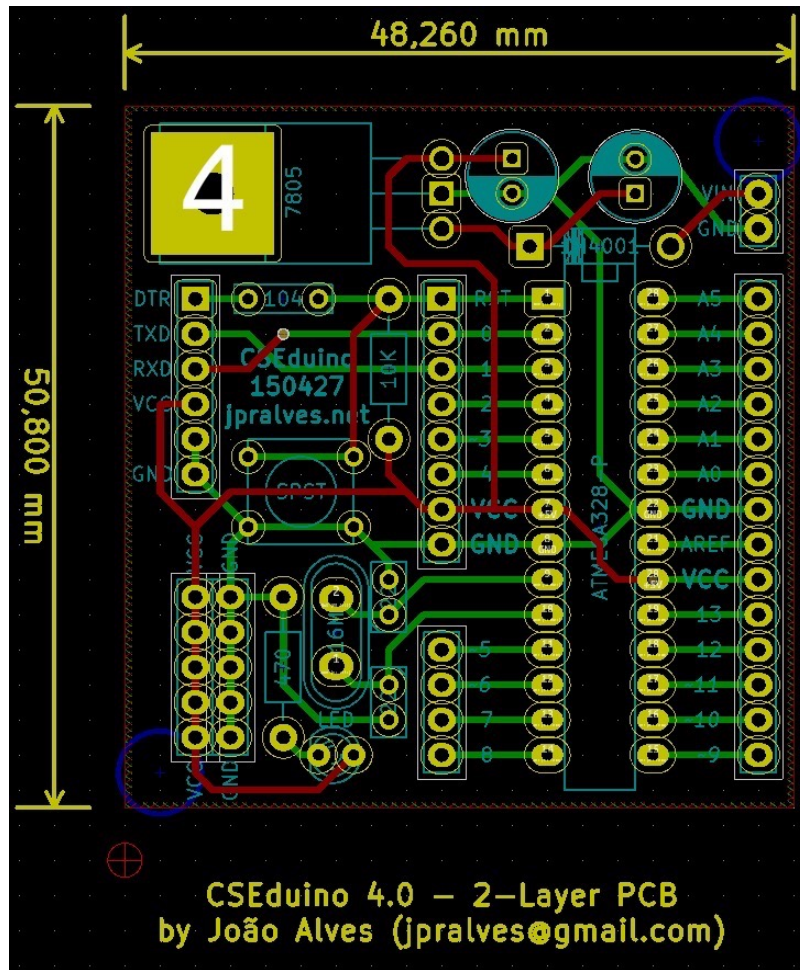


Figure 1.5.10: Featured board 'Made with KiCad': CSEduino.

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.

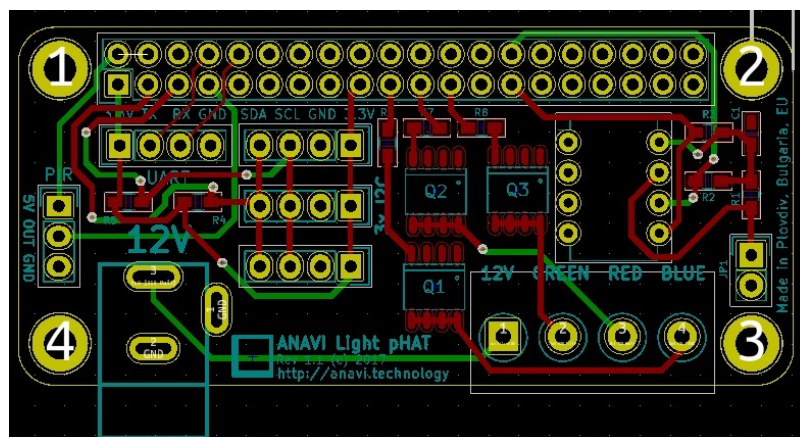


Figure 1.5.11: Featured board 'Made with KiCad': Anavi Light.

Finally, a truly impressive board made with KiCad is Crazyflie (Figure 1.5.12). Crazyflie is a dense 4-layer PCB with a rather elaborate shape. The board implements 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.

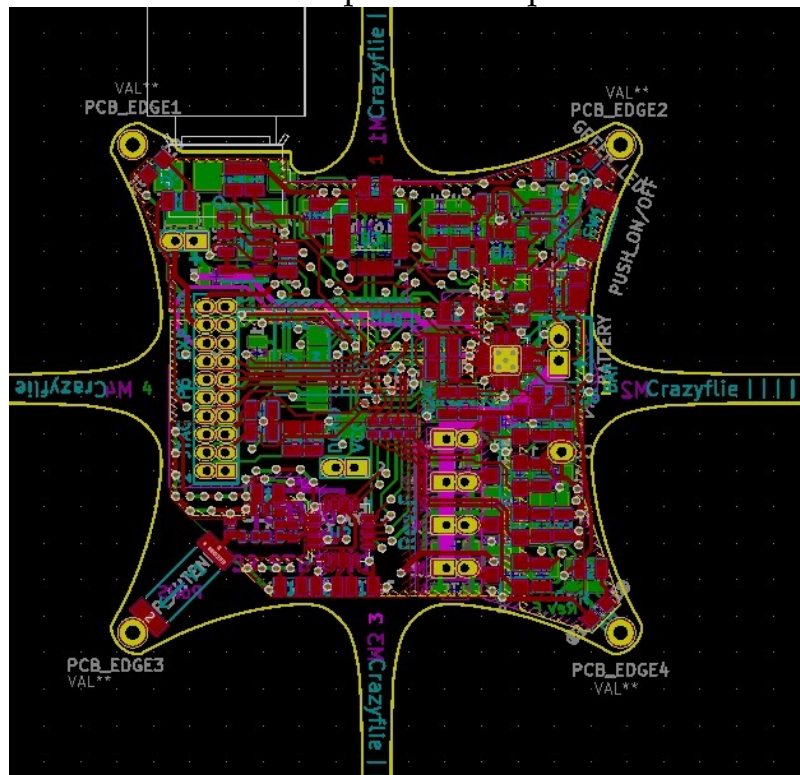


Figure 1.5.12: Featured board 'Made with KiCad': Crazyflie.

With this chapter complete, you should now understand the kinds of projects that people use KiCad. These are also the kinds of boards that you will design by the time you complete this book. Let's get straight into the first project so that you can start discovering this fantastic tool by doing.

## **Part 2: Getting started with KiCad 6**

# 1. Introduction

Welcome to Part 2 of this book.

In the chapters of this Part, I will give you a brief overview of KiCad 6. This overview will help you with the first hands-on activity of this course, in which you will create your first PCB in the chapters of the following two Parts.

Ensure that you have installed KiCad on your computer so that you can follow along. If you haven't done so yet, please go back to chapter "4. Get KiCad for your operating system," where I provide information on installing KiCad on Mac OS, Windows, and Linux.

In the following chapters, I will introduce the individual apps that make up the KiCad software suite. I will also explain the roles of paths to the symbol, footprint, 3D model, and template libraries, show you how to create a new project from scratch and a template.

I will also compare KiCad 6 as it runs on the three supported platforms. If you have experience with KiCad 5, read the relevant chapter at the end of this Part.

Let's continue with the following chapter, where I'll give you an overview of KiCad's core apps.

## 2. KiCad Project Manager (main window)

This chapter will give you an overview of the KiCad project manager, otherwise known as the "main" KiCad window.

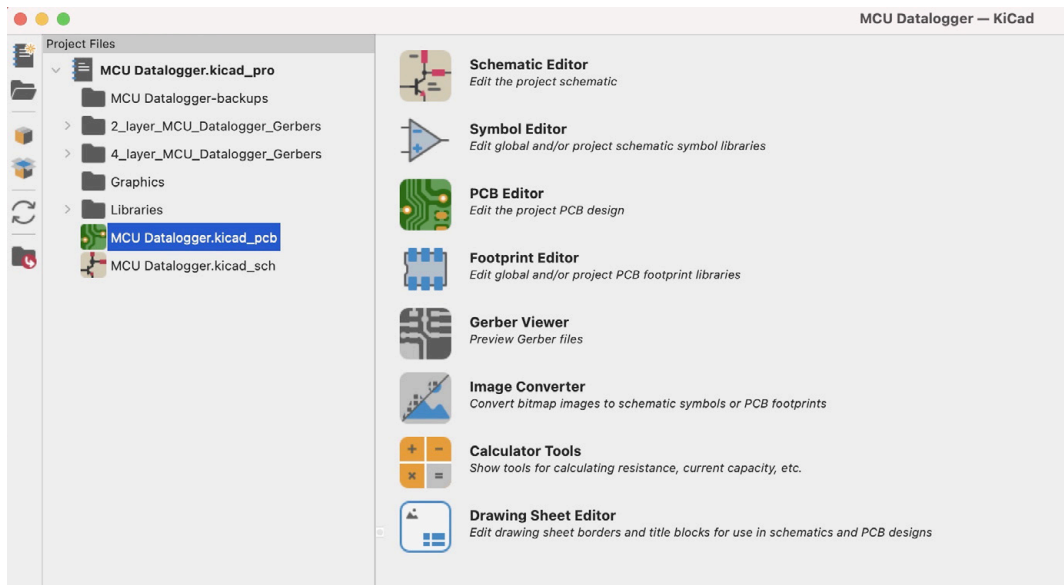


Figure 2.2.1: The KiCad Project Manager window.

This is the window that you will see first when you start KiCad. The project manager gives you access to the various KiCad applications, like the schematic and symbol editors, and shows you the project files.

The main window contains:

- A toolbar on the left.
- The project files are in the middle.
- The application buttons are on the right side.

The left toolbar has buttons to create a new project or open an existing project and archive/unarchive.

The middle pane shows the project files and folders. This is essentially a file browser that gives you access to the individual files and folders inside the main KiCad project directory.

The right pane contains buttons for the individual applications. Say that you want to start the schematic editor. You can do this in three ways:

1. Double-click on the file with the extension "kicad\_sch" in the middle pane (file browser).
2. Click on the Schematic Editor button in the right pane.

3. Click on "Schematic Editor" under Tools in the top menu (see below).

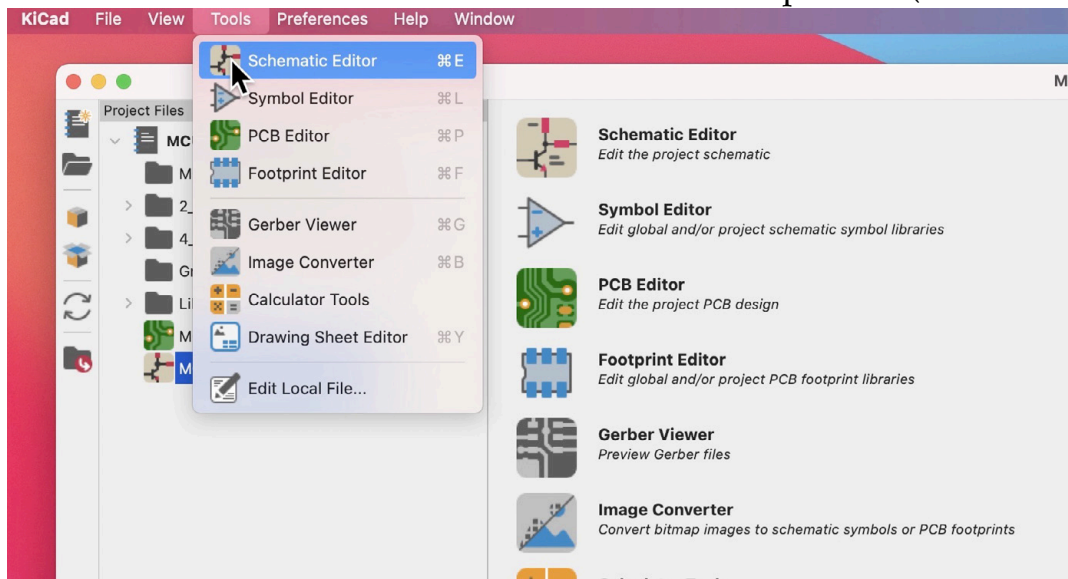


Figure 2.2.2: Starting the Schematic editor.

If you create a new directory via your operating system's file manager or create a new file, the middle pane will display those items. Remember that a KiCad project will contain files that KiCad creates and files created by other tools, like the Autorouting autorouter and Git. You will learn about the core files in KiCad later in this book.

You will learn about the buttons in the right pane in the next chapter. First, let's do a tour of the items in the top menu bar.

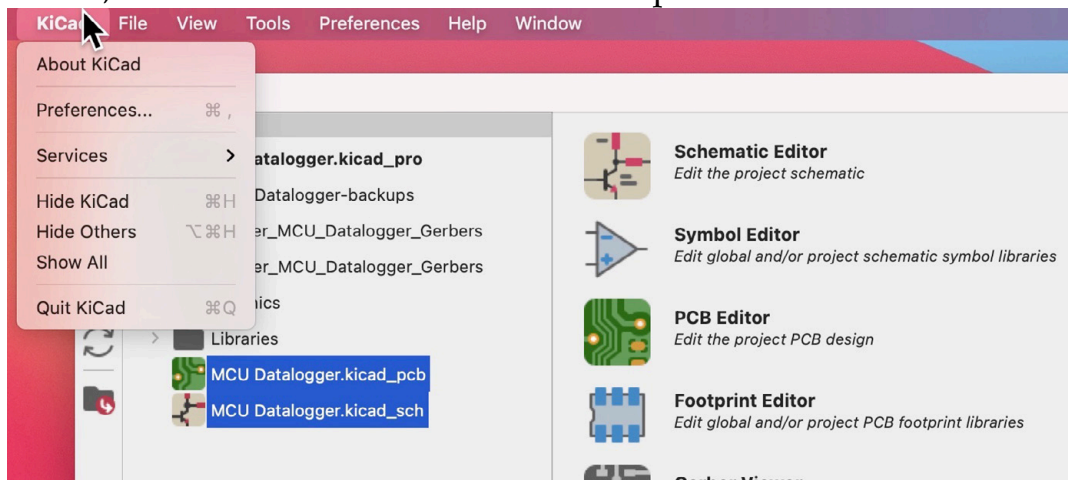


Figure 2.2.3: The top menu bar in KiCad.

To get information about your instance of KiCad, click on "About KiCad" under the KiCad menu item. You will need to use the information provided in this window if you have found a bug and wish to report it to the development team.



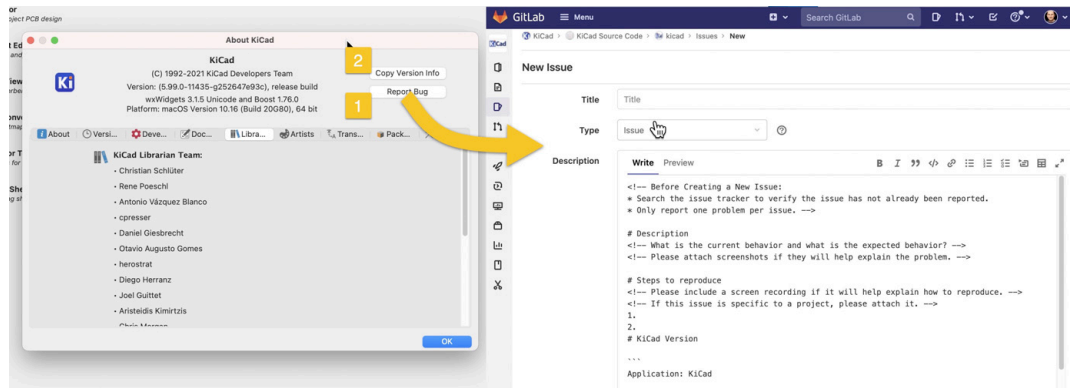


Figure 2.2.4: Report a bug.

To report a bug, open the About KiCad window, and click "Report Bug" (see "1" above). This will use your web browser to open the New Issue page in Gitlab. You will need to include your KiCad instance version information, which you can get from the About KiCad window ("2", above).

Also, from the KiCad menu item, you can bring up the Preferences window.

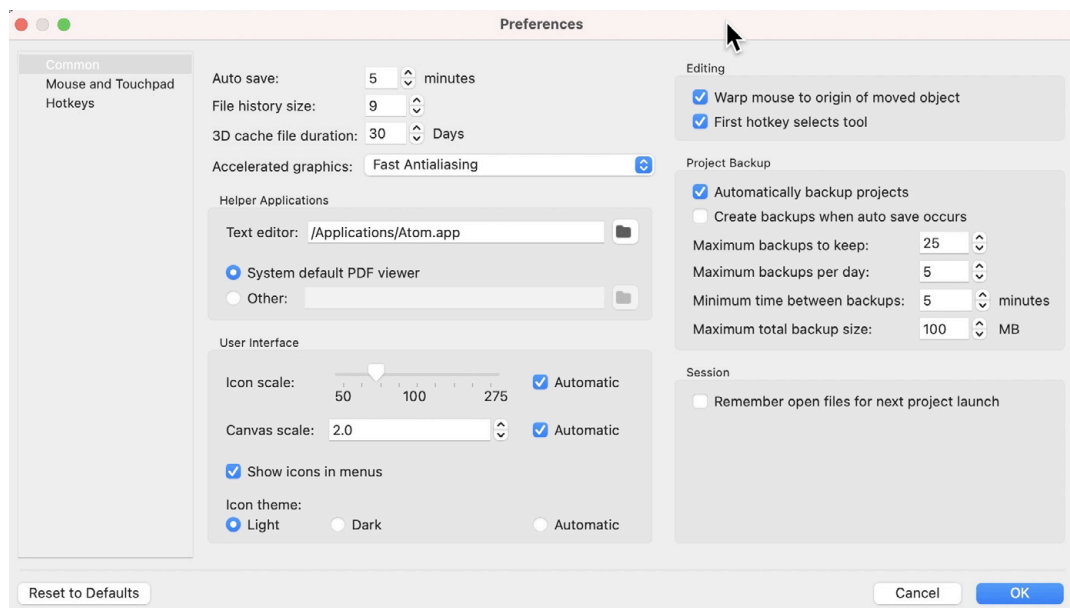


Figure 2.2.5: The KiCad Preferences window.

In the Preferences window contains several tabs with widgets that allow you to customize KiCad. Exactly what you see here depends on which applications are open. In the example above, only the main KiCad project window is open. The right pane would contain additional items if Eeschema or Pcbnew were also open. You can learn about the details in dedicated chapters later in this book (Eeschema, and Pcbnew).

Under the File menu, you see the standard options for file and project management. You can open/close a project, create a new project, archive/

unarchive a project, and import non-KiCad projects. You will find some of those options as buttons in the right toolbar of the main KiCad window.

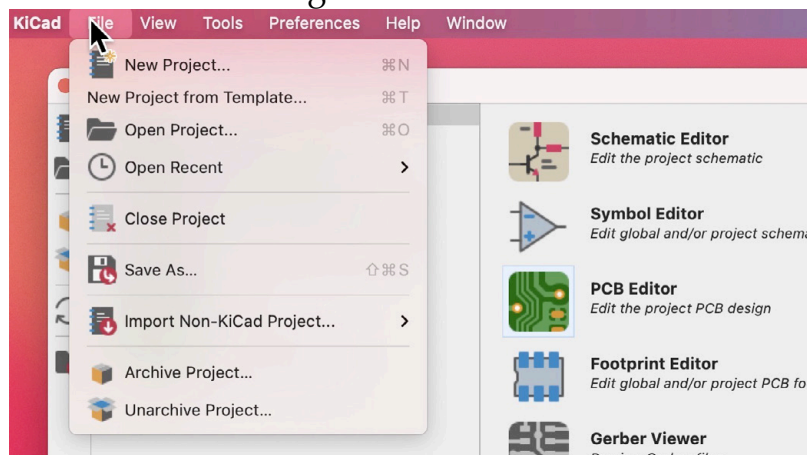


Figure 2.2.6: The KiCad File menu.

You will be using those options in the projects through this book. In the Recipes part of this book, you can learn how to import a non-KiCad project, and how to archive/unarchive.

Under View, you can use a text editor to view any of KiCad's project files. You can define your preferred text editor in the Preference window in the Common tab. Below you can see an example of a KiCad schematic file loaded in the Atom text editor.

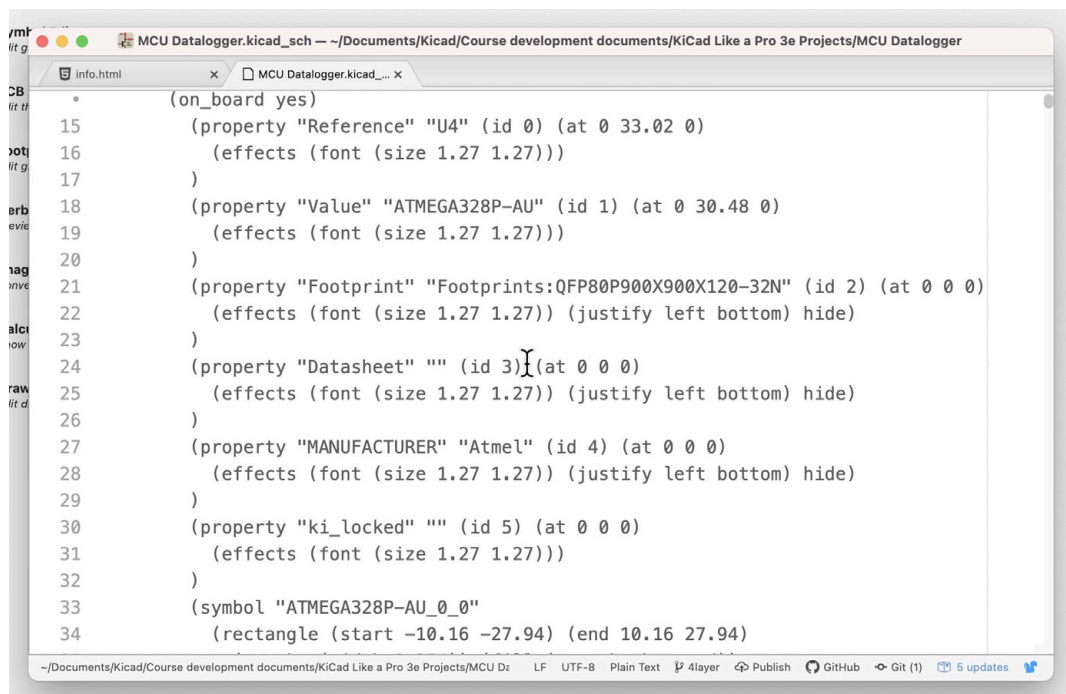


Figure 2.2.7: A schematic design file in a text editor.

All KiCad files are text files, and as such, you can open them in a text editor. It is also possible to programmatically edit those files using automation



implemented in a language like Python directly, without needing an API. Beware, though: modifying these files by hand or programmatically without knowing precisely what you are doing will most likely damage your KiCad project. Always back up your work before any such experimentation!

The Tools dropdown menu gives you access to the individual apps in the KiCad software suite. The items in this menu replicate the application buttons in the right pane of the KiCad main window.

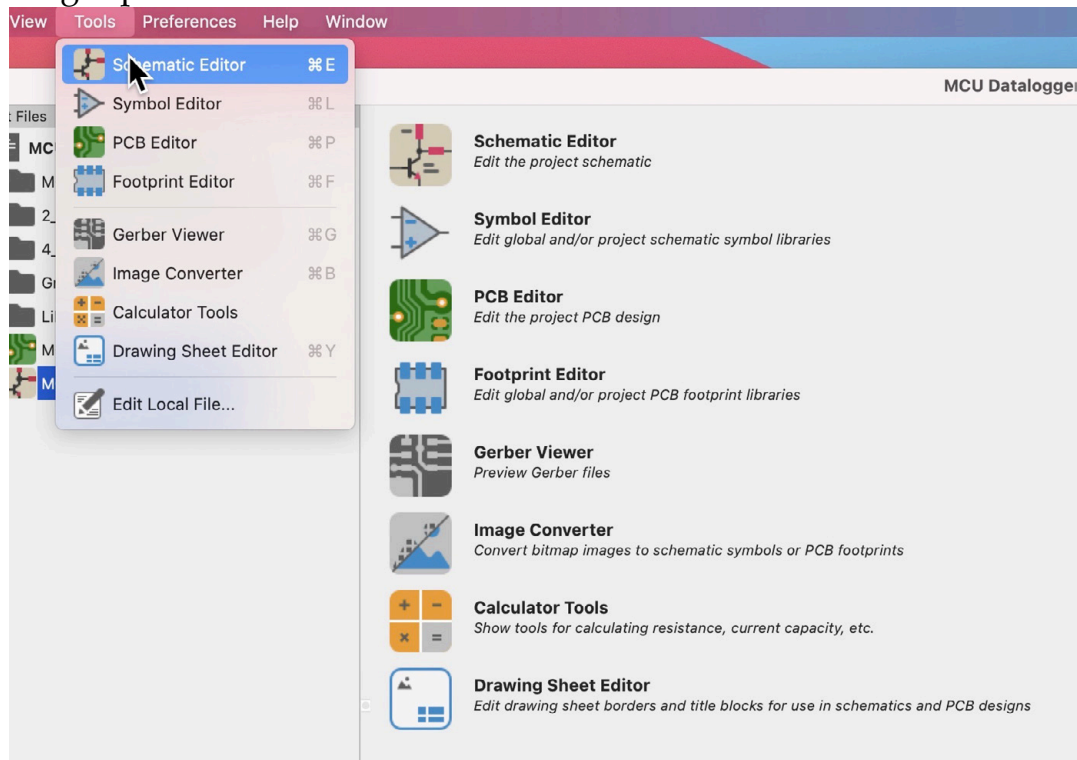


Figure 2.2.8: The Tools menu items.

I will describe these applications in the next chapter.

Under Preferences (not to be confused with the Preferences window under "KiCad"), you can access the Paths, Symbol Libraries, and Footprint Libraries manager windows.

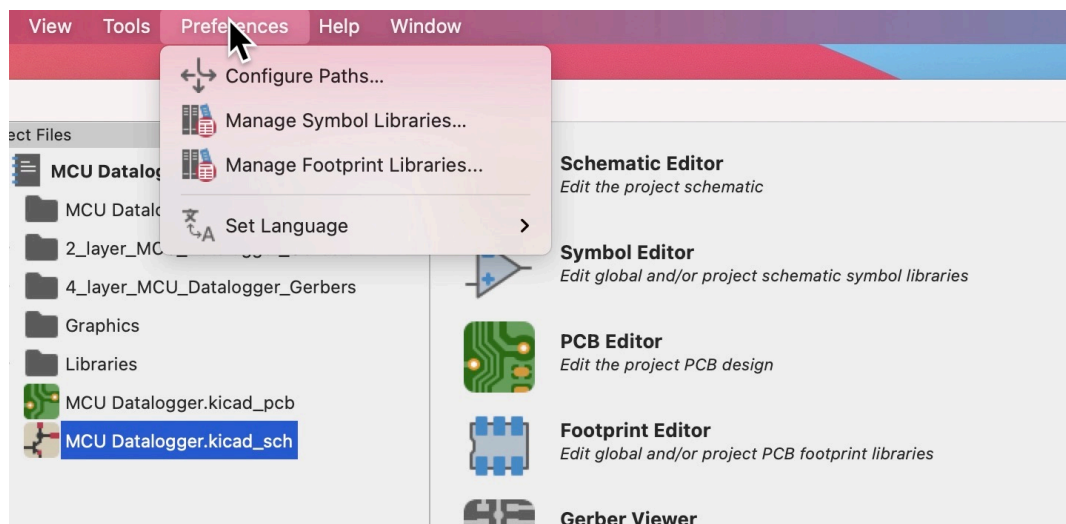


Figure 2.2.9: The Preferences menu.

I have written a dedicated chapter on these manager windows with details later in this Part of the book.

Finally, the help menu. It allows you to access a local copy of the official KiCad documentation, which opens in your browser, and a window that contains a list of hotkeys. Be mindful that this documentation may be old. When I am writing this, this documentation has not been updated since KiCad 5.0.0-rc2, and most of the links are not working.

The Hotkeys window, apart from listing current hotkeys, allows you to make changes. I prefer to keep the default hotkeys unless there is a conflict with other applications running on my computer.

This was an overview of the main KiCad window, the KiCad project manager. In the next chapter, you will learn about the individual applications that make up the KiCad software suite.

### 3. Overview of the individual KiCad apps

In the previous chapter, you learned about the KiCad Project Manager. This chapter will give you a tour of the individual applications that make up the KiCad software suite.

As you may recall from the previous chapter, you can access the KiCad applications via the project manager's right pane or the Tools menu. To open Eeschema or Pcbnew, you can also double-click on the schematic and layout files listed on the middle page of the project manager.

Let's take a closer look at each of the KiCad applications.

#### Schematic editor: Eeschema

Click on the Schematic Editor button to open the application. You can see the editor window below.

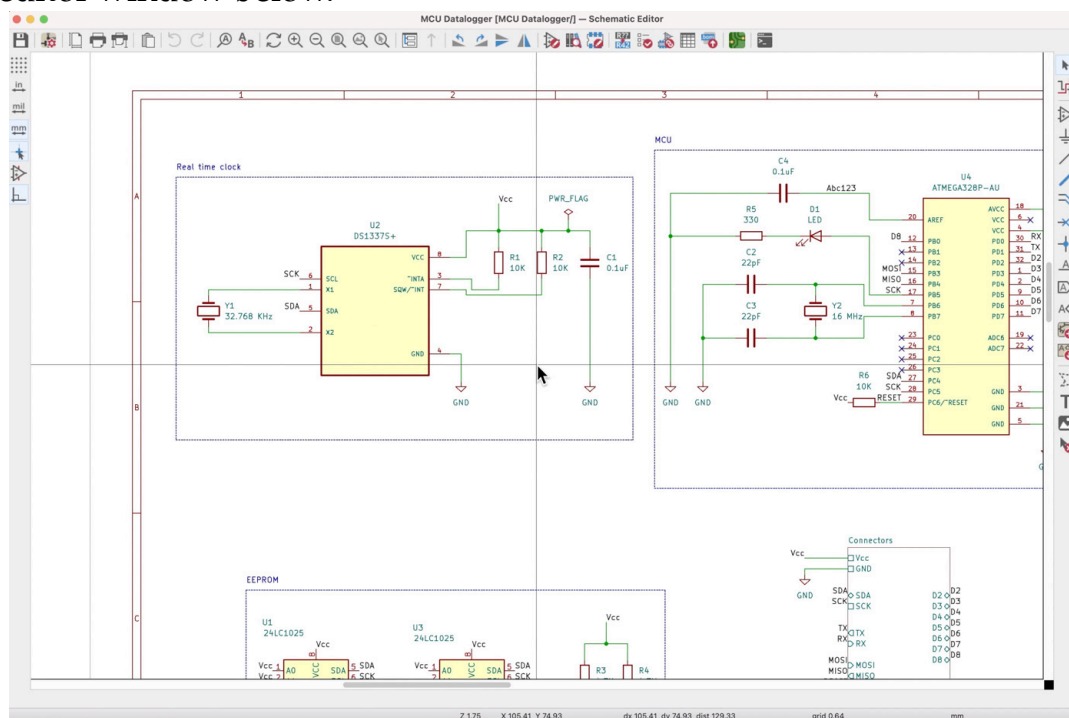


Figure 2.3.1: Eeschema, or the Schematic Editor.

You use Eeschema to draw the schematic of the PCB. Although KiCad is flexible enough and allows you to create PCBs without a schematic, this is rarely a good idea. The schematic diagram captures all necessary information that the layout editor uses: components (as symbols), wires that connect pins,

nets, and various kinds of netlabels, busses, power nets, and much more. Eeschema is the first KiCad application you will use when you start a new KiCad project.

In the example above, you can see a schematic from one of the projects in this book. You can see the symbols (such as U2, R1, and R2), green wires connecting pins, special symbols representing unconnected pins and power nets, and other elements like graphics and text labels.

You can learn how to use and configure the schematic editor in a dedicated Part of this book.

## Layout editor: Pcbnew

Once you have completed work in Eeschema, you will continue with the Layout editor, or “Pcbnew.” To open Pcbnew, you can click on the Pcbnew button in the KiCad project manager or the Pcbnew button in the top toolbar of Eeschema. Below you can see an example instance of Pcbnew.

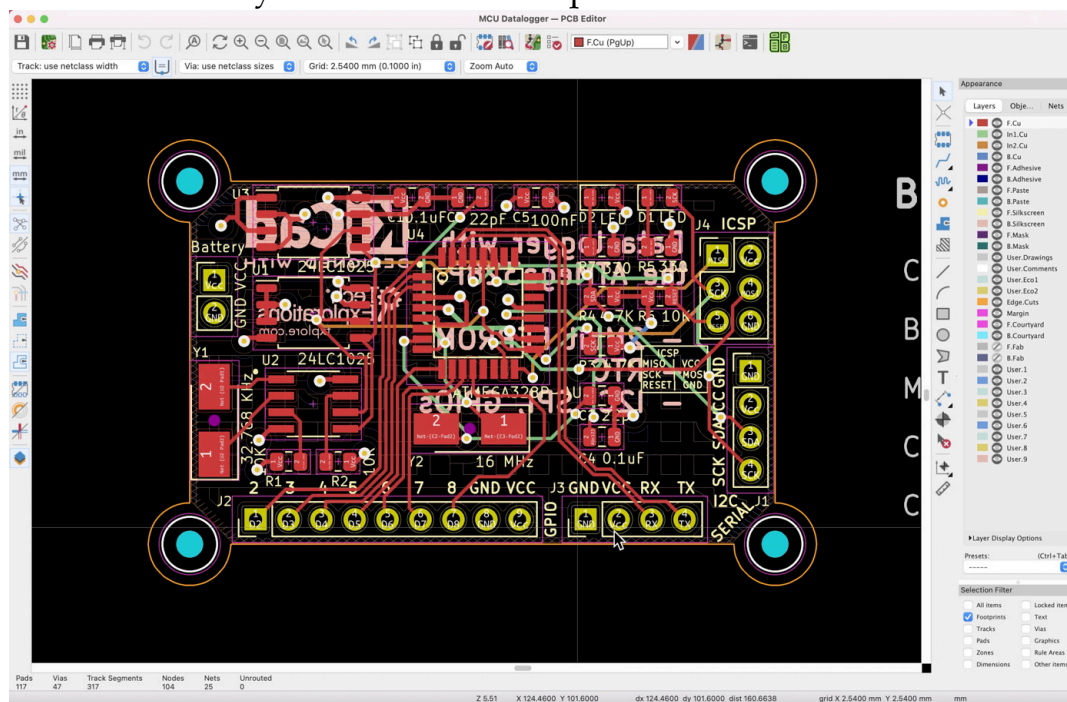


Figure 2.3.2: Pcbnew, or the Layout Editor.

In the example above, you can see the finished PCB design from one of the projects in this book. The layout editor allows you to select the layers and design elements you want to see. For example, you can enable or disable the visibility of layers, footprints, tracks, zones, and vias. In the example above, I have enabled the visibility of all layers and elements and an outline of the top and bottom copper zones.

The layout editor includes various sophisticated tools, such as an interactive router and a 3D viewer. You can see a 3D rendering of the PCB from Figure Figure 2.3.2 below:

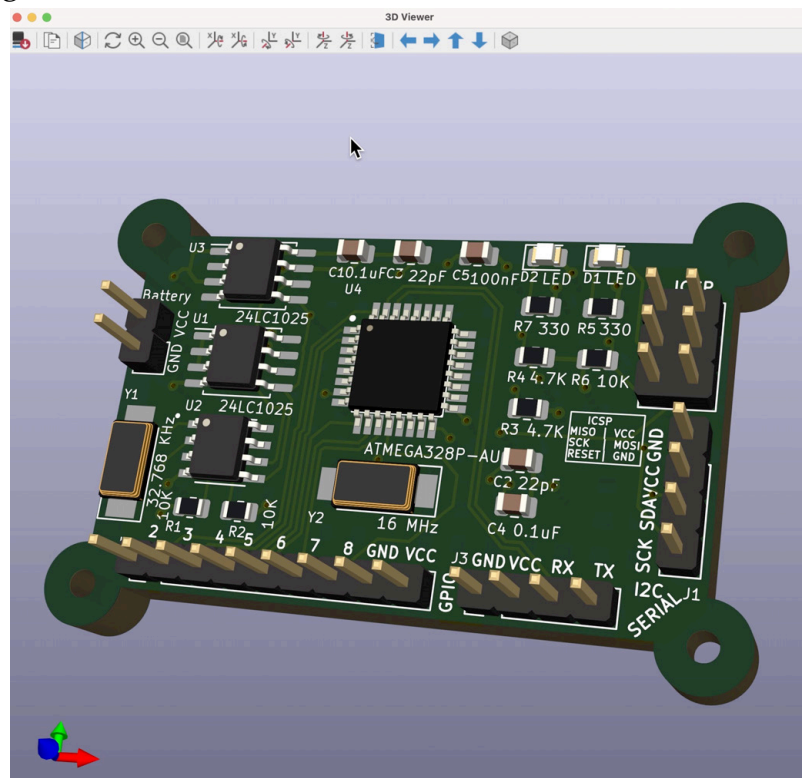


Figure 2.3.3: The 3D viewer in Pcbnew.

You can learn how to use and configure the layout editor in a dedicated Part of this book.

## Symbol Editor

Let's continue with the Symbol Editor. You can open this application from the KiCad Project Manager or the top toolbar of Eeschema.

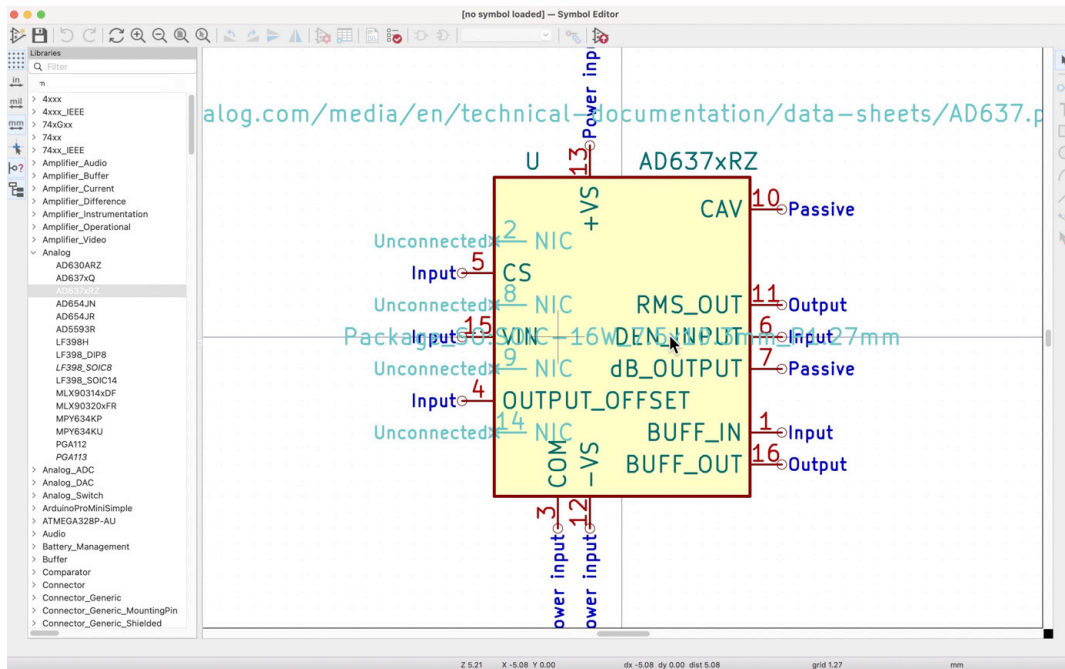


Figure 2.3.4: The Symbol Editor.

With the Symbol Editor, you can modify existing symbols or create new ones. You can think of the Symbol Editor as a simplified version of the schematic editor. In the Symbol Editor, you can work with a single symbol at a time.

KiCad 6 comes with an extensive set of symbol and footprint libraries. There are also thousands of third-party symbols and footprints that you can import. However, you will eventually need to create a symbol, and that's when the Symbol Editor comes in.

You can learn how to create new symbols from scratch later in this book.

## Footprint editor

Similar to the symbol editor, there is also the footprint editor. You can open the footprint editor from the KiCad project window or the Pcbnew top toolbar.



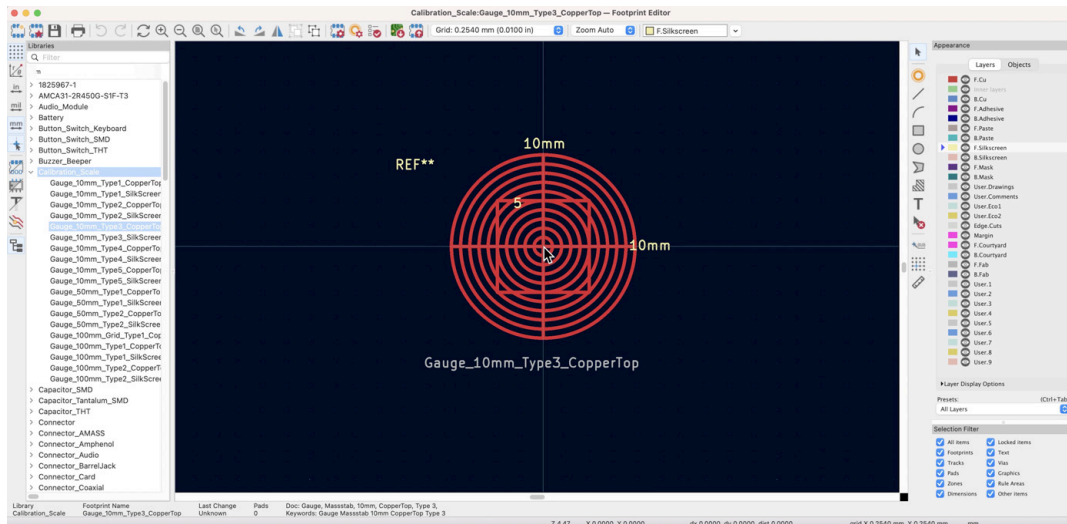


Figure 2.3.5: The Footprint Editor.

With the footprint editor, you can create a footprint from scratch or modify an existing footprint. The footprint editor also contains a wizard that allows you to quickly generate footprints that follow convention, such as those that use BGA, QFN, DIP and SOIC, packages.

You can learn how to use the footprint editor in a dedicated chapter.

## Gerber Viewer

When you have completed work on your PCB and wish to order it from an online manufacturer, the most common way is to export a set of Gerber files from Pcbnew. Before you upload those files to your preferred manufacturer, you should take the time to inspect them. KiCad has a tool for this: the Gerber Viewer.

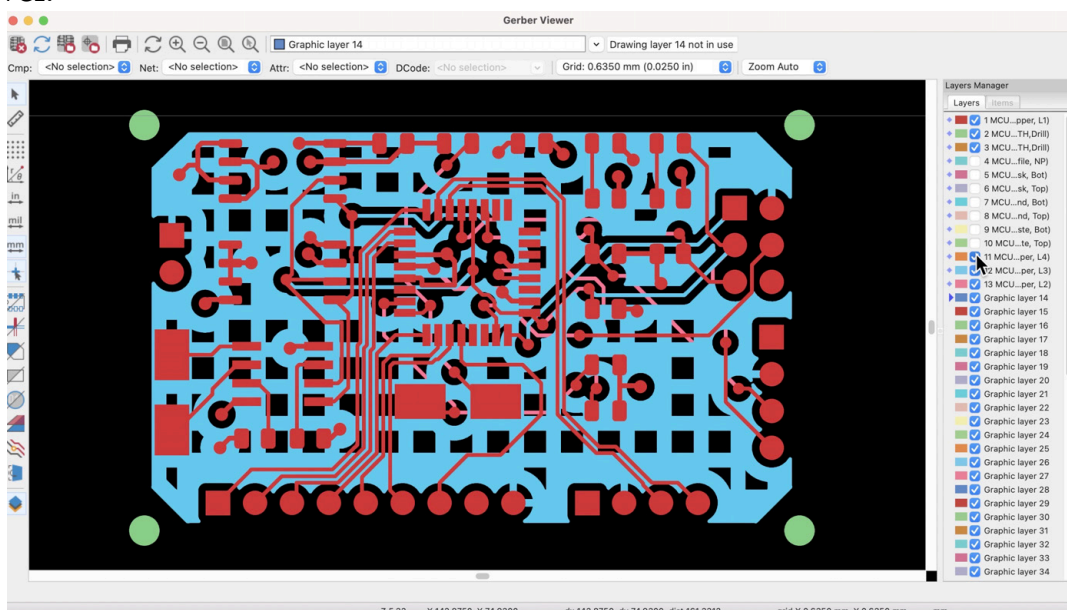


Figure 2.3.6: The Gerber Viewer.

With the Gerber Viewer, you can examine the project Gerber files visually, layer by layer. This way, you can ensure that all its elements are correct. Silkscreen text and graphics, drills, copper fills, the board outline, and cutouts, etc.

Think of the Gerber Viewer as a quality control tool. Use it to reduce or eliminate the risk of ordering a defective PCB.

You can learn how to export the Gerber files and use the Gerber Viewer (and online Gerber viewers) in dedicated chapters later in this book.

## Image Converter

You can open the image converter app from the KiCad Project Manager. With the Image converter, you can convert a bitmap image into a footprint. Typical uses of the converter are to create a graphics footprint (such as a company logo) or a footprint with an irregular shape that would be too tedious to design in the footprint editor.

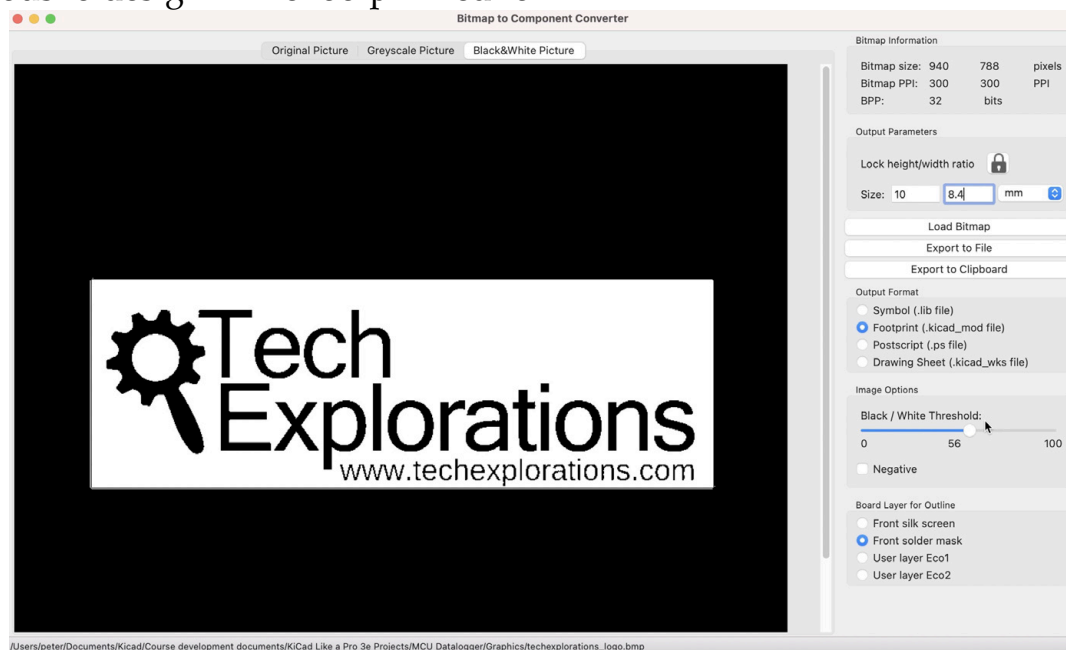


Figure 2.3.7: The Image Converter.

In the example above, I use the Image Converter to create a logo that I can include in my PCBs. You can learn how to use the Image converter in a dedicated chapter in Part 13 of this book.

## Calculator tools

The calculator tool contains multiple calculators. Here is a list of tools:

1. Voltage regulators.
2. RF Attenuators.



3. E-Series.
4. Resistor color codes.
5. Transmission lines.
6. Via size.
7. Track Width.
8. Electrical spacing.
9. Board classes.

In the example below, I am using the Track Width calculator to calculate the correct width given a set of parameters.

The screenshot shows the 'PCB Calculator' application with the 'Track Width' tab selected. The 'Parameters' section on the left includes: Current: 1.0 A, Temperature rise: 10.0 °C, Conductor length: 20 mm, and Copper resistivity: 1.72e-08 Ω·m. The central area displays the IPC 2221 formula:  $I = K \cdot \Delta T^{0.44} \cdot (W \cdot H)^{0.725}$ , with definitions for I, ΔT, W, H, and K. The right panel shows calculated values for 'External Layer Traces': Trace width: 0.300387 mm, Trace thickness: 0.035 mm, Cross-section area: 0.0105135 mm², Resistance: 0.0327197 Ω, Voltage drop: 0.0327197 V, and Power loss: 0.0327197 W. A similar section for 'Internal Layer Traces' shows a trace width of 0.781437 mm and other related values. A 'Reset to Defaults' button is at the bottom right.

Figure 2.3.8: The Calculator tool.

You can learn how to use the Track Width calculator by reading the relevant chapter in the Recipes part of this book. The mode of operation for the rest of the calculators is similar.

## Drawing Sheet editor

The last main application in the KiCad suite is the Drawing Sheet editor. You can use this editor to customize your schematic editor sheet. You can see the editor in the example below.

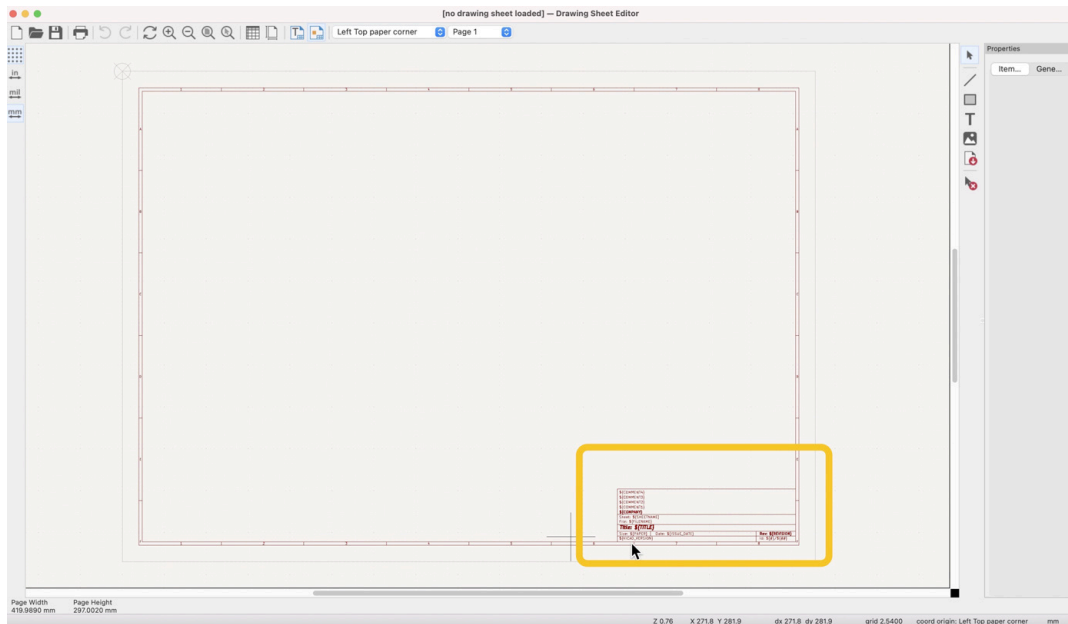


Figure 2.3.9: The Drawing Sheet editor.

With the Drawing Sheet Editor, you can change the size of the schematic sheet and everything within it. For example, you can remove or change the size and location of the information container. You can also change the setup of the text placeholders inside the information box.

To learn how to use the Drawing Sheet Editor, please read the relevant chapter in the Recipes part of this book.

## 4. Paths and Libraries

In the KiCad project window, you will find the paths and libraries configurations options under the preferences menu item.

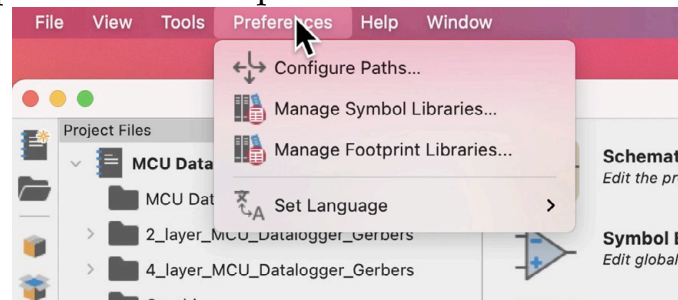


Figure 2.4.1: The Preferences menu.

Let's look at each one.

### Configure Paths

Bring up the “Configure Paths” window from the Preferences menu.

This window contains a table to environment variables that contain paths to important collections of files.

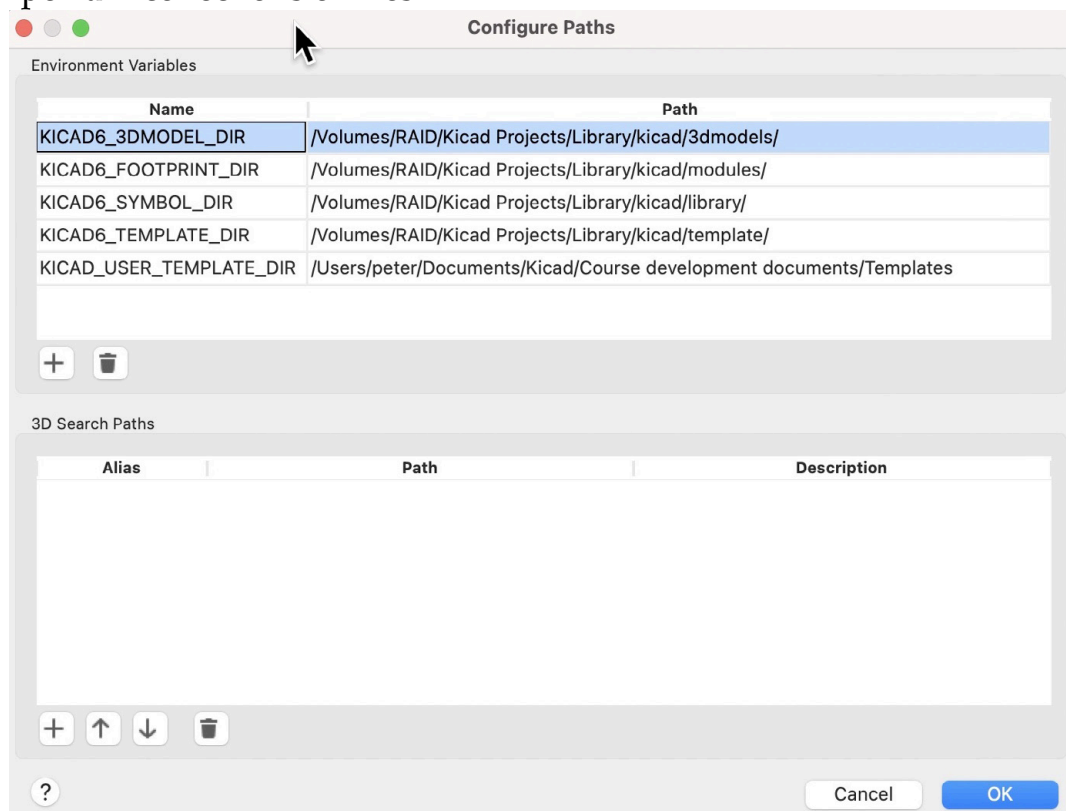


Figure 2.4.2: The “Configure Paths” window.

As you can see in the figure above, there are five path environment variables:

- KICAD6\_3DMODEL\_DIR: points to a directory that contains 3D models of components for use by the 3D viewer. Learn more about this in a dedicated chapter.
- KICAD6\_3RD\_PARTY: points to a directory that contains 3<sup>rd</sup> party plugins, libraries, and other downloadable content.
- KICAD6\_FOOTPRINT\_DIR: points to a directory that contains footprint files for use by Pcbnew. Learn more about this in a dedicated chapter.
- KICAD6\_SYMBOL\_DIR: points to a directory that contains symbol files for use by Eeschema. Learn more about this in a dedicated chapter.
- KICAD6\_TEMPLATE\_DIR: points to a directory that contains sheet template files for use by Eeschema. Learn more about this in a dedicated chapter.
- KICAD\_USER\_TEMPLATE\_DIR: points to a directory that contains project template files created by the user. You can use these template files to start a new project quickly. Learn more about this in a dedicated chapter.

When you install KiCad, these variables will inherit default values that point to the KiCad application installation folder. You can use the Configure Paths window to change these values.

For example, my computer has a solid-state drive with a limited amount of available space on it. Because the libraries (especially the 3D models) take several gigabytes of storage, I have opted to use my external RAID drive for those resources. As you can see in Figure 2.4.2 above, the footprint, symbol, and 3D model paths point to my external RAID drive, while the rest point to locations on the internal SSD.

## Manage Symbol Libraries

Use the symbol libraries manager to:

- Add new symbol libraries.
- Delete symbol libraries.
- Activate or deactivate symbol libraries.

The Symbol Libraries window contains a list of active or inactive libraries installed in your KiCad instance. Each library may contain one or more schematic symbols. When a library is installed and activated, you can use its symbols in your schematics in Eeschema.

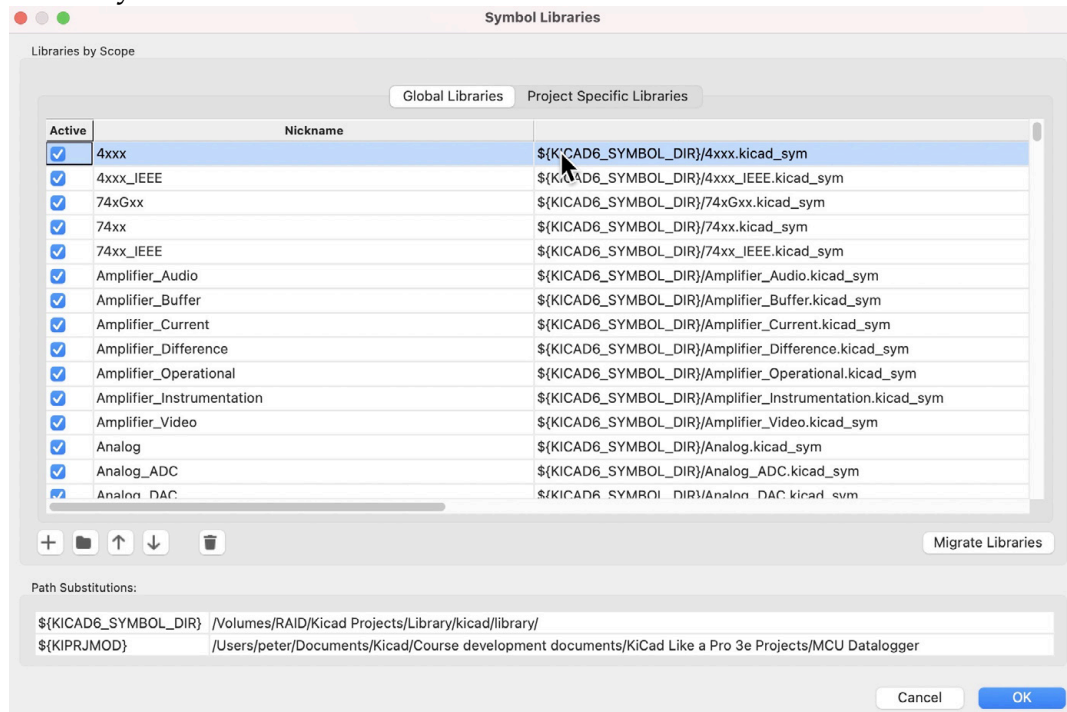


Figure 2.4.3: The “Symbol Libraries” window.

In the figure above, you can see the Symbol Libraries window with several of the libraries installed in my instance of KiCad.

Notice that:

- The table contains two tabs: “Global Libraries” and “Project Specific Libraries.” You can manage libraries under each tab to control the library visibility (global or project-specific).
- Each library has a name and a path. The path can use an environment variable, as in the example above. Alternatively, you can set an absolute path to a library; this is often a good option when you want to install a library stored outside the standard environment paths.
- If you forget the environment variable paths, look at the bottom of the window. In the table “Path Substitutions,” you can see the actual path stored in the environment variables.

Learn how to use the symbol libraries manager in a dedicated chapter later in this book.

## Manage Footprint libraries

Use the footprint libraries manager to:

- Add new footprint libraries.
- Delete footprint libraries.
- Activate or deactivate footprint libraries.

The footprint libraries manager window works similarly to the symbol libraries manager.

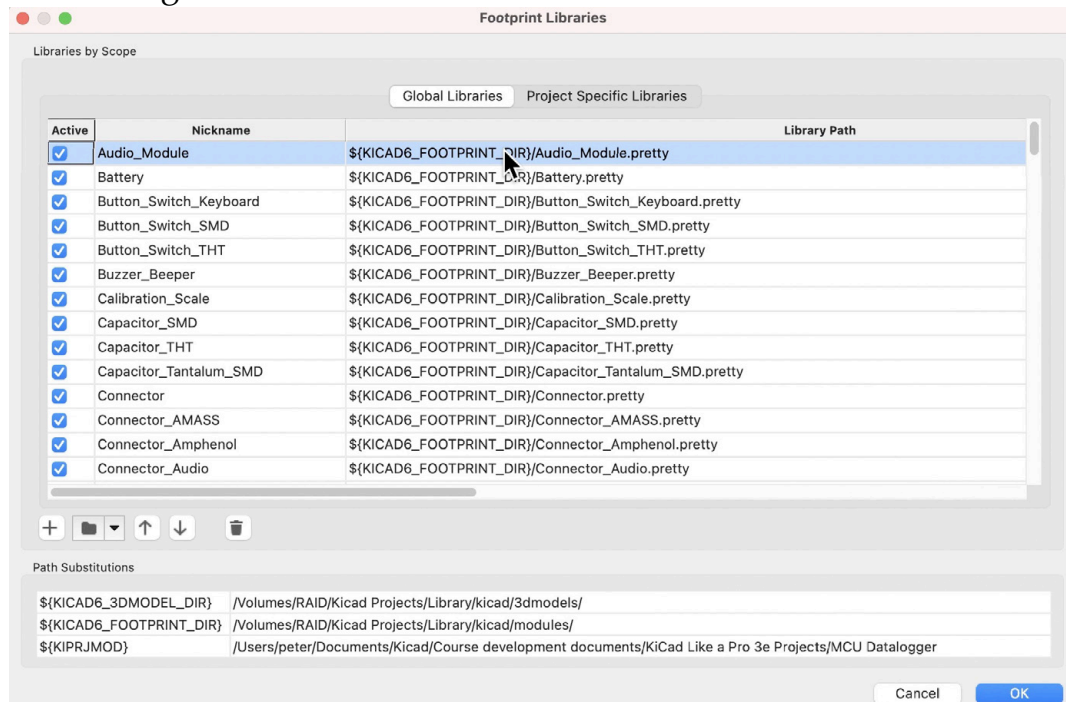


Figure 2.4.4: The “Footprint Libraries” window.

You can control the context of a library by listing them under the “Global Libraries” or “Project Specific Libraries” tab. Each library has a name and a path, and the path may contain an environment variable or an absolute path. Learn how to use the footprint libraries manager in a dedicated chapter later in this book.

## 5. Create a new project from scratch

In this chapter, you will learn how to create a new KiCad project.

Kicad offers you two ways to start a new project:

1. A new blank project.
2. A new project from a template.

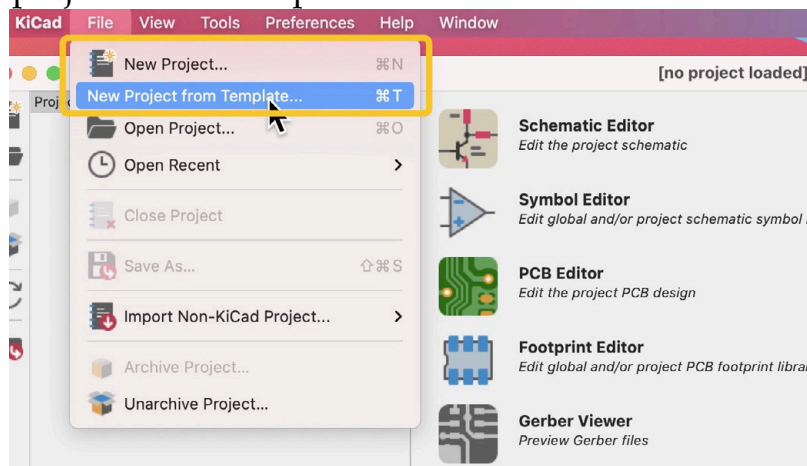


Figure 2.5.1: KiCad offers two ways to start a new project.

When you start a new project from a template, you can take advantage of work that you (or the original author of the template) have done in the past. Project templates offer an excellent way to speed up the initial time-consuming steps for projects that share a common base. For example, if you create Arduino shields, you can set up an Arduino shield base template and use it to create new Arduino shield projects. You can learn more about project templates in a dedicated chapter in the Recipes part.

In this chapter, you will create a new blank project. In the File menu, click on “New Project...”. In the window that appears, set a name (“1”, below), check the new folder box to have KiCad automatically create a new folder for your project (“2”), and click Save (“3”).

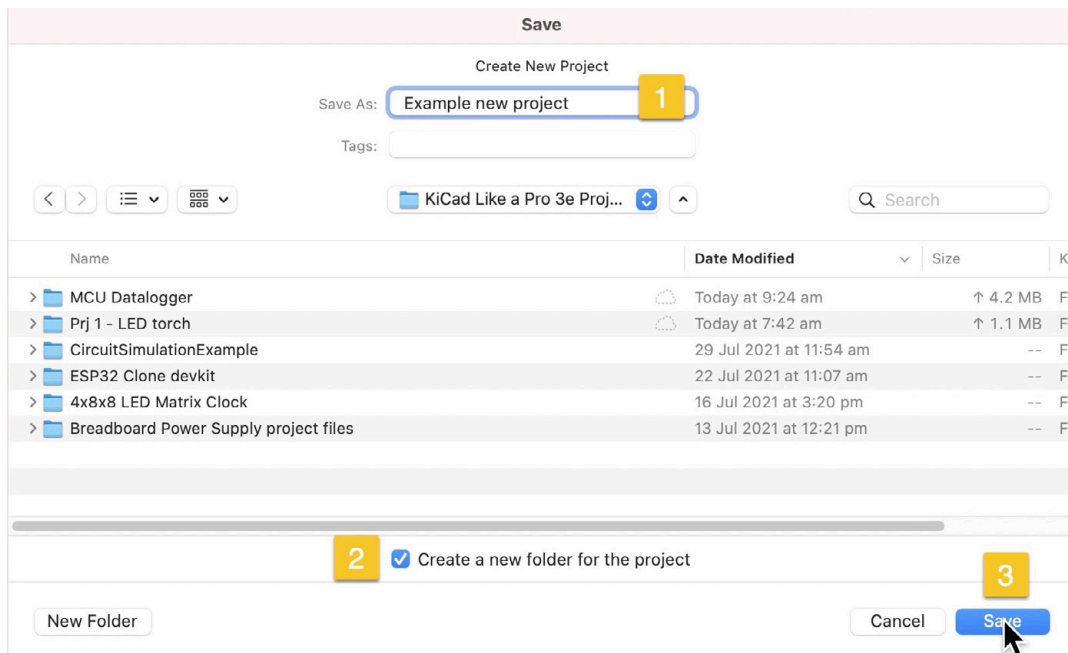


Figure 2.5.2: Set a name and directory for the new project.

KiCad will set up your new project. In the project folder, you will see three new files:

1. The main project file with extension “.kicad\_pro.”
2. The schematic design file with extension “.kicad\_sch.”
3. The layout design file with extension “.kicad\_pcb.”

The KiCad project window will show the project as a hierarchy tree. At the top of the hierarchy is the project file (“.kicad\_pro”), and inside of that are the schematic and layout files.

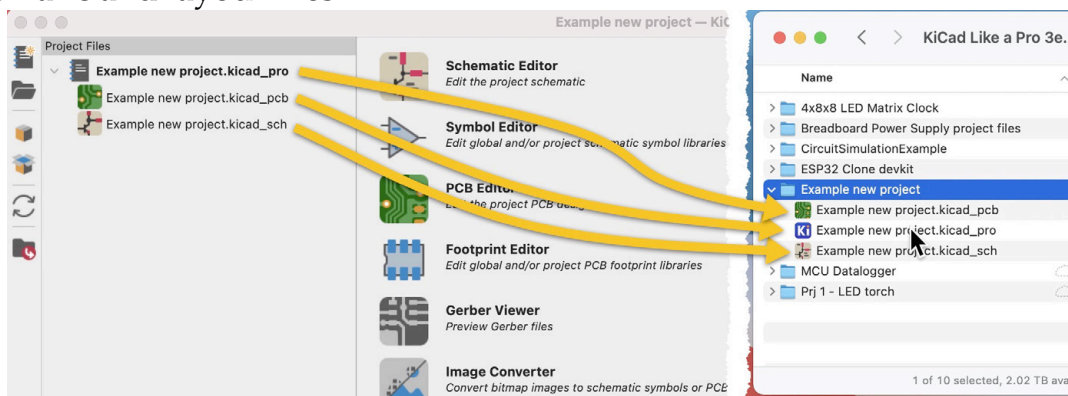


Figure 2.5.3: The new project is ready.

At this point, your new project is ready. You can open the schematic editor and begin work on the schematic. This is where you will begin work in the next part of this book, in which you will work on your first KiCad project. In the next chapter, you will learn how to create a new KiCad project from a template.



## 6. Create a new project from a template

In this chapter, you will learn how to create a new project from a template. KiCad comes with several project templates ready to use, but you can also create yours. You can read a dedicated chapter in the Recipes part if you are interested in creating custom project templates.

Click on “New Project from Template” in the File menu to create a new project from a template. The project templates window will appear (see below).

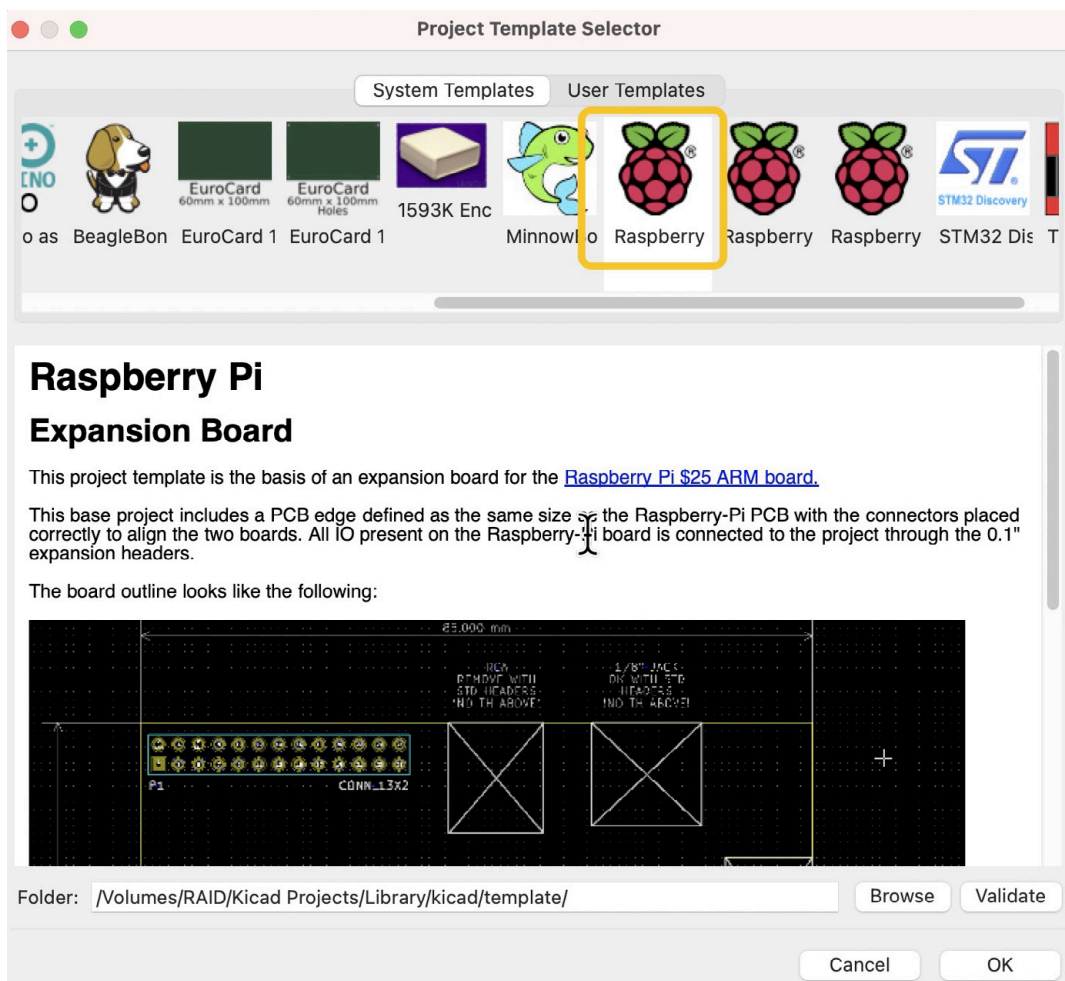


Figure 2.6.1: The project templates selector.

The selector window contains two tabs: System Templates and User Templates.

In a new KiCad installation, the User Templates tab will be empty until you create a new template and store it in the appropriate template directory (learn how to do this in the relevant chapter in the Recipes part).

The System Templates tab shows a collection of built-in templates. Click on a template icon to see information about it. For this example, I have selected one of the Raspberry Pi templates. The information box shows a description of the template. The description is composed of regular HTML so that you can include text, links, and images.

After selecting the template, you want to use, click OK. This will bring up the Save dialog box. This is identical to the dialog box that appears when you create a new blank project. Give the new project a name and location, and click Save.

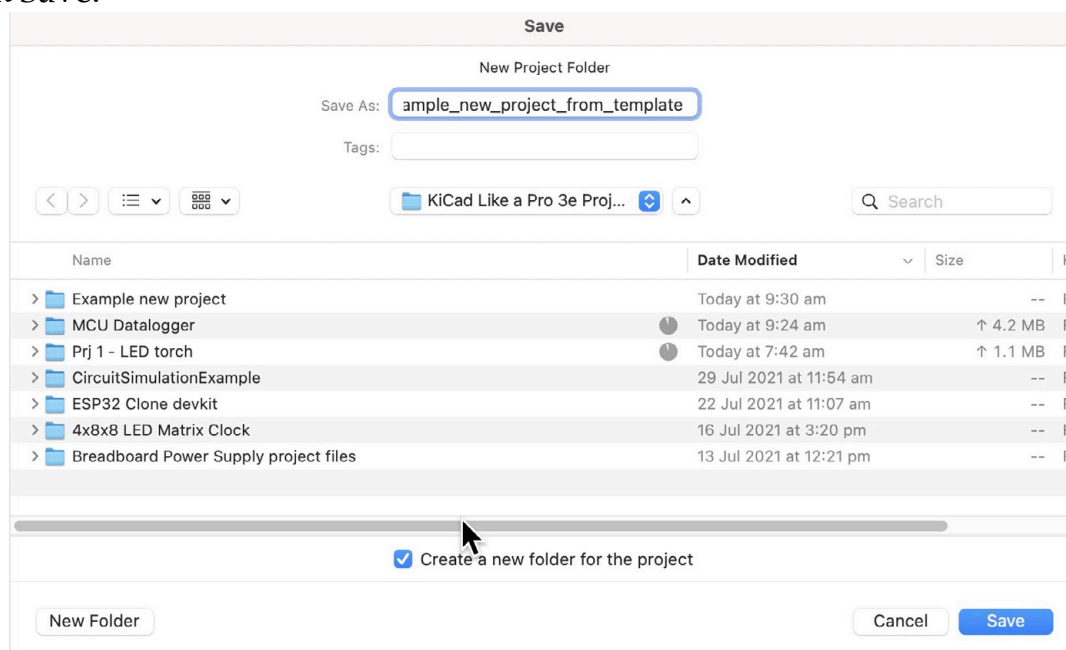


Figure 2.6.2: The name and location of the new project.

When KiCad finished creating the new project from the Raspberry Pi template, you will see several new files in the project folder (right, below) and the project hierarchy in the KiCad project window (left, below).



Figure 2.6.3: The new project created from a project template.

In the project folder (above, right), notice that several additional files also appear in addition to the project, schematic, and layout files. These additional files have been copied from the Raspberry Pi project template.

In the KiCad project window, click on the Schematic Editor button to open Eeschema. In a new blank project, the schematic editor is empty. But this is a new project from a template; the schematic and layout editors are already populated with seeding content.

Below is the schematic editor showing a header and mounting holes for a Raspberry Pi project:

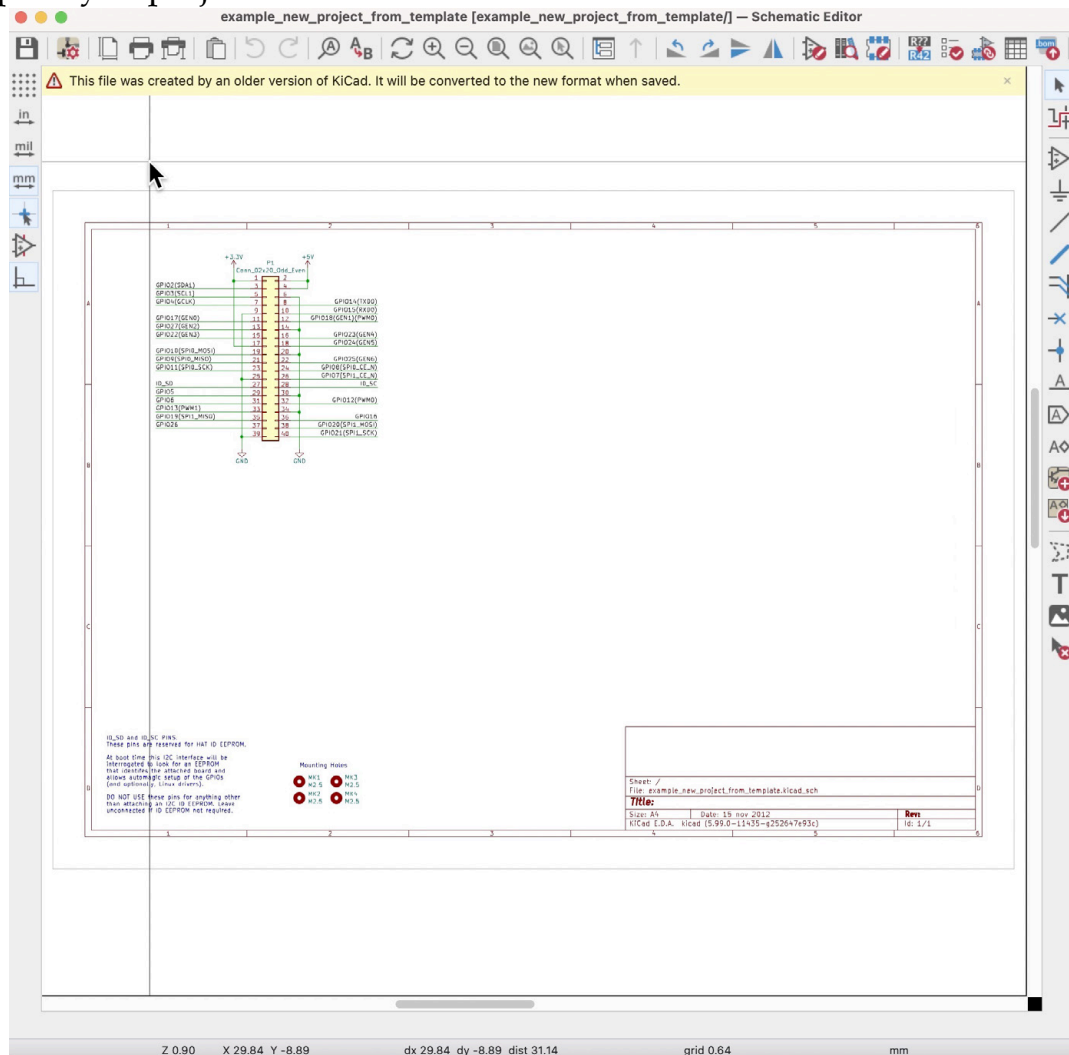


Figure 2.6.4: The new project schematic is already populated with content from the template.

Similarly, the layout editor is already populated with content from the template:

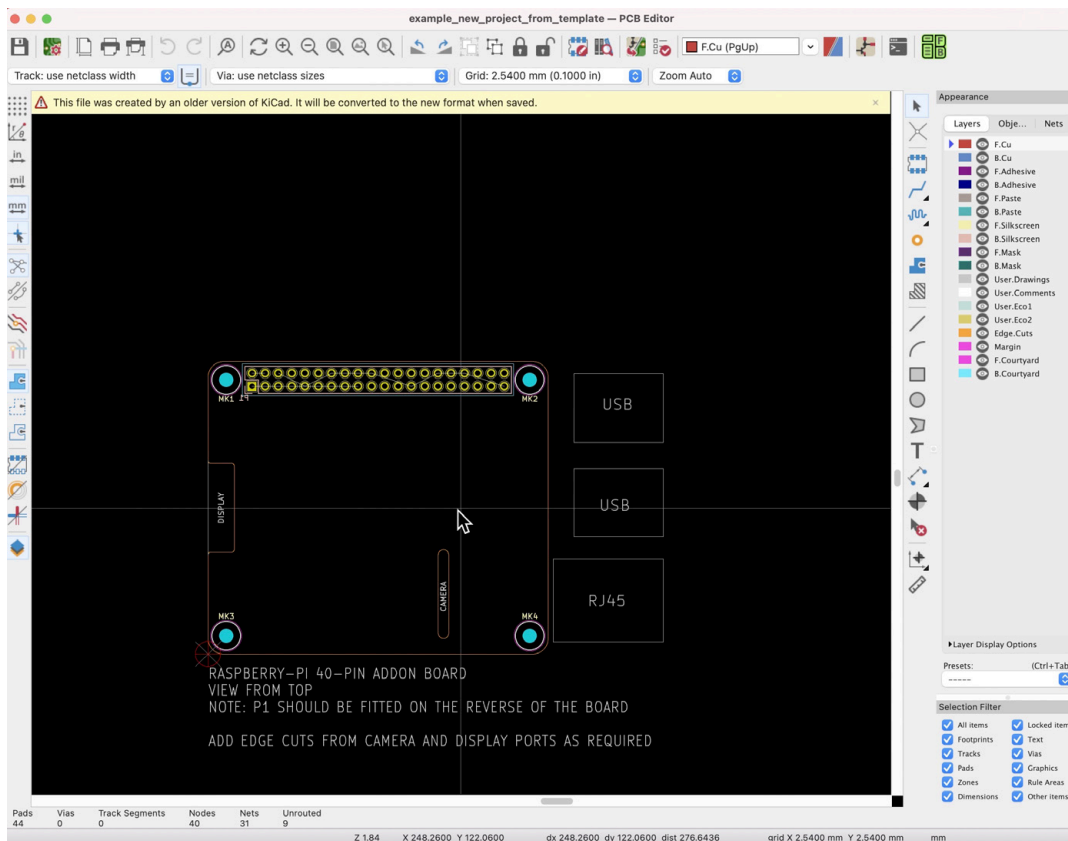


Figure 2.6.5: The new project layout is already populated with content from the template.

As you can see, much of the work has already been done. In the layout editor, the design of the board outline requires exact measurements, which are time-consuming. The placement of the mounting holes and connectors, likewise, must be exact and, as a result, very time-consuming. All this is work that you can avoid when you create a new project from a template.

Creating a new project from a template is an example of a productivity-boosting tool that KiCad provides. You will learn about many more in this book.

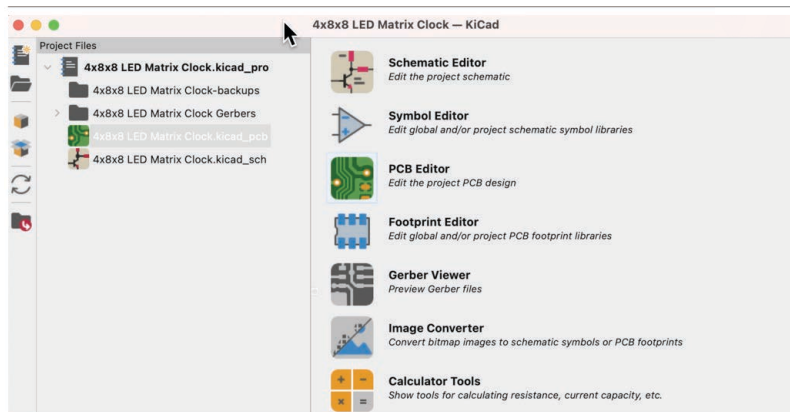
## 7. KiCad 6 on Mac OS, Linux, Windows

KiCad has supported multiple operating systems from its early days. When I started using KiCad in version four, I used it on Windows, Mac OS, and Linux (Ubuntu). However, there were differences between those platforms, both in terms of reliability (I found Windows, generally, worked better) and how the user interface looked and behaved.

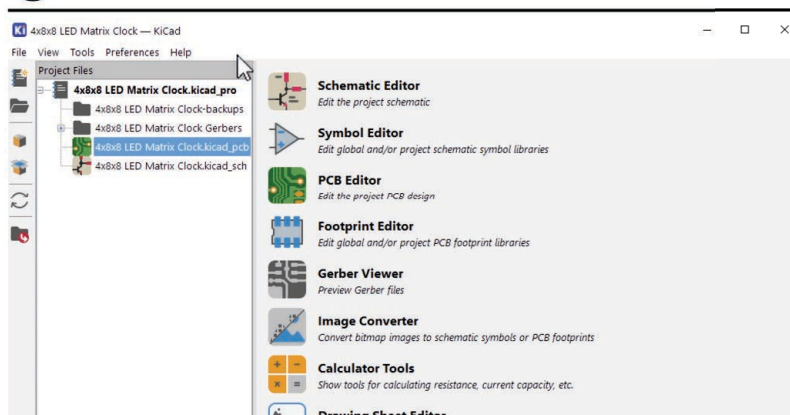
I have been using KiCad 6 almost daily for almost nine months now, and I feel that KiCad works seamlessly on the three operating systems I have used (Mac OS, Windows 10, and Linux).

I spent a lot of time comparing the two. My testing consisted of a single project that I opened and edited across the three operating systems. I used KiCad's "archive project" function, which you can find under "File" in the KiCad project window. Opening and working on a project that I previously edited on a different operating system were trouble-free.

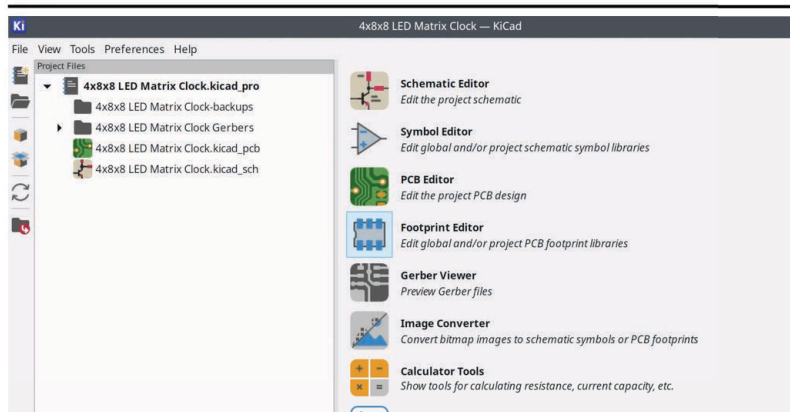
Below, you can see the same project's main KiCad project window in Mac OS, Windows, and Kubuntu. They look identical while following the UI conventions of their host operating system.



## 1 Mac OS



## 2 Windows

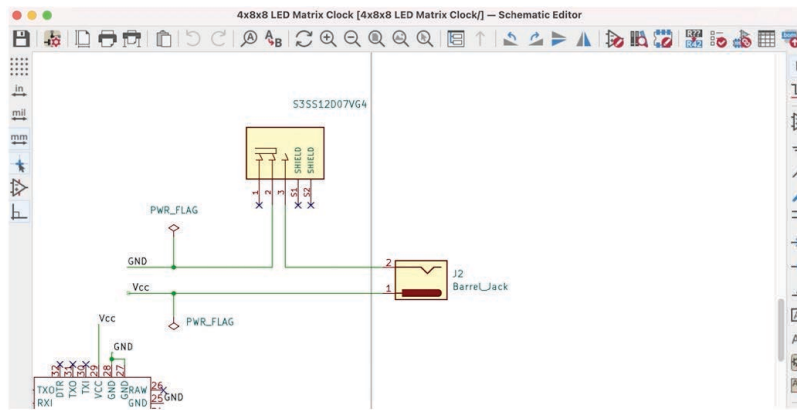


## 3 Linux Kubuntu

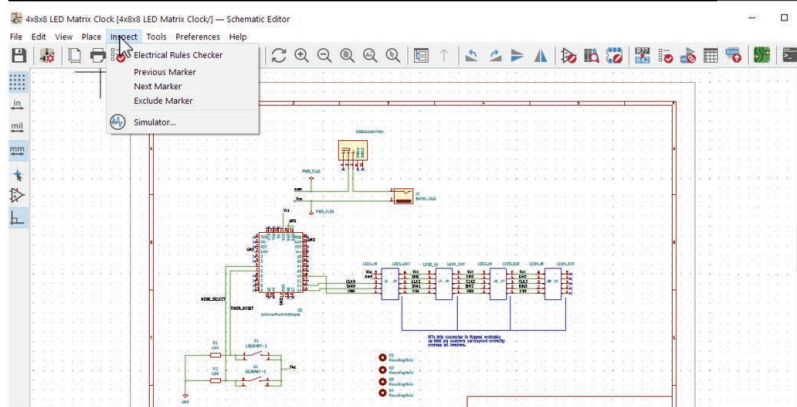
Figure 2.7.1: KiCad project window on three OSs.

There were no surprises in terms of KiCad's main applications, Eeschema and Pcbnew, and how those work. Shortcuts, mouse conventions, menus, buttons, colors; all work as expected in a truly cross-platform compatible application suite.

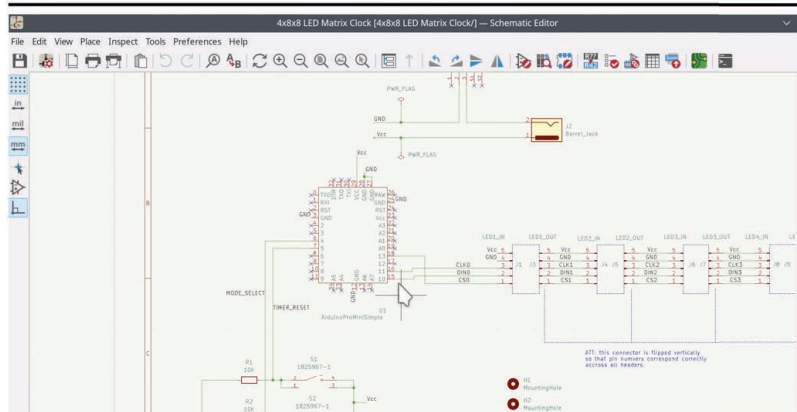
Below is an example of Eeschema in the three operating systems:



1 Mac OS



2 Windows

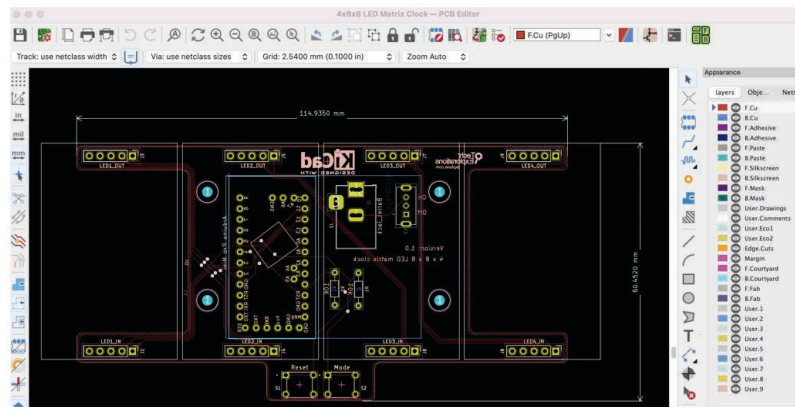


3 Kubuntu

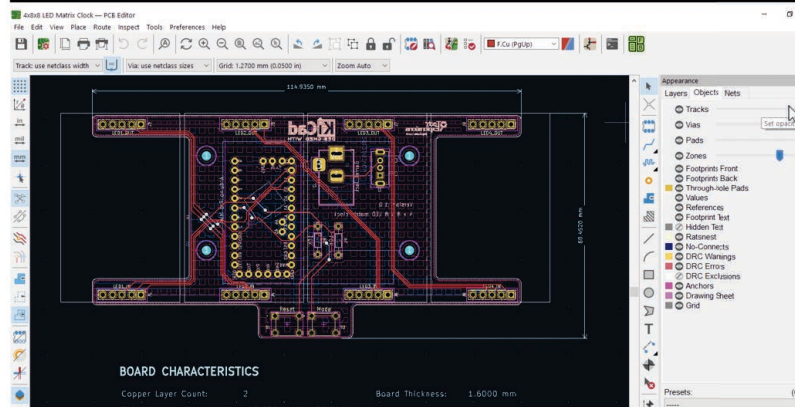
Figure 2.7.2: Eeschema in the three OSs.

And here is Pcbnew:

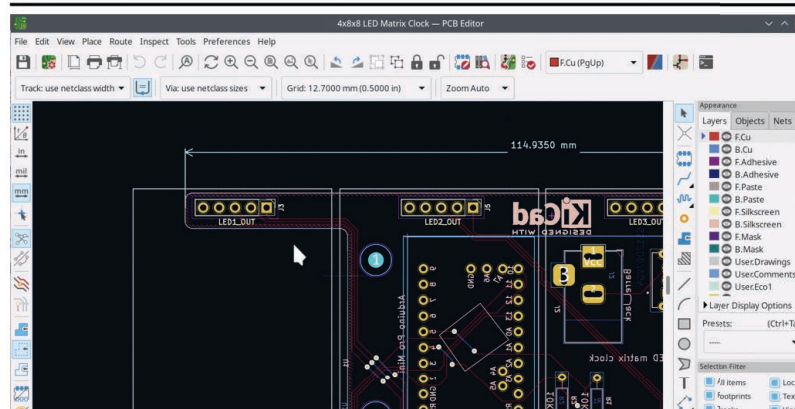




## 1 Mac OS



## 2 Window

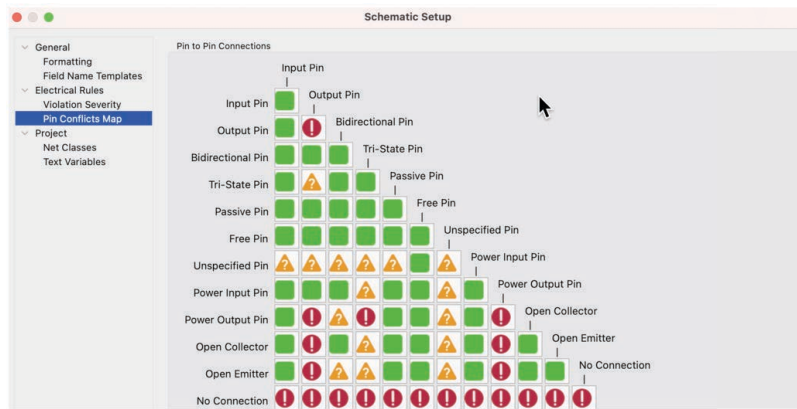


## 3 Kubuntu

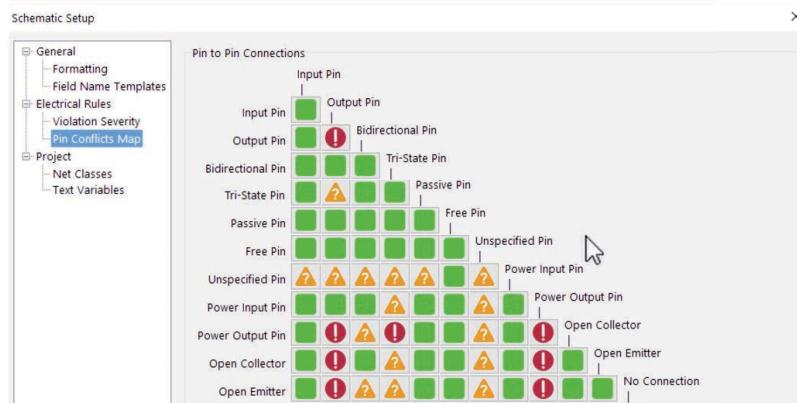
Figure 2.7.3: Pcbnew in the three OSs.

The same uniformity appears when testing other KiCad applications, such as the 3D viewer, the various preferences windows, and the interactive router. Even secondary widgets and features work well across the supported platforms.





1 Mac OS



2 Windows

Figure 2.7.4: Schematic Setup in Mac OS and Windows.

The quality of the implementation of Kicad in the three operating systems I have tested is excellent. The implication for solo users and teams is that you can use KiCad 6 with high confidence that you can edit the same projects across platforms. If you are in a team, your team members will work using their preferred operating system.

## 8. Differences between KiCad 6 and 5

KiCad 6 is a significant upgrade over KiCad 5. If you are new to KiCad, and KiCad 6 is the first KiCad you have ever used, you can safely ignore this chapter. Go ahead to Part 3, and begin work on your first KiCad project.

However, if you have used a previous version of KiCad and created one or more projects, you take some time to read a [blog post](#) that I wrote in early 2021. In that blog post, I go into detail to highlight and explain the differences between KiCad 6 and KiCad 5.

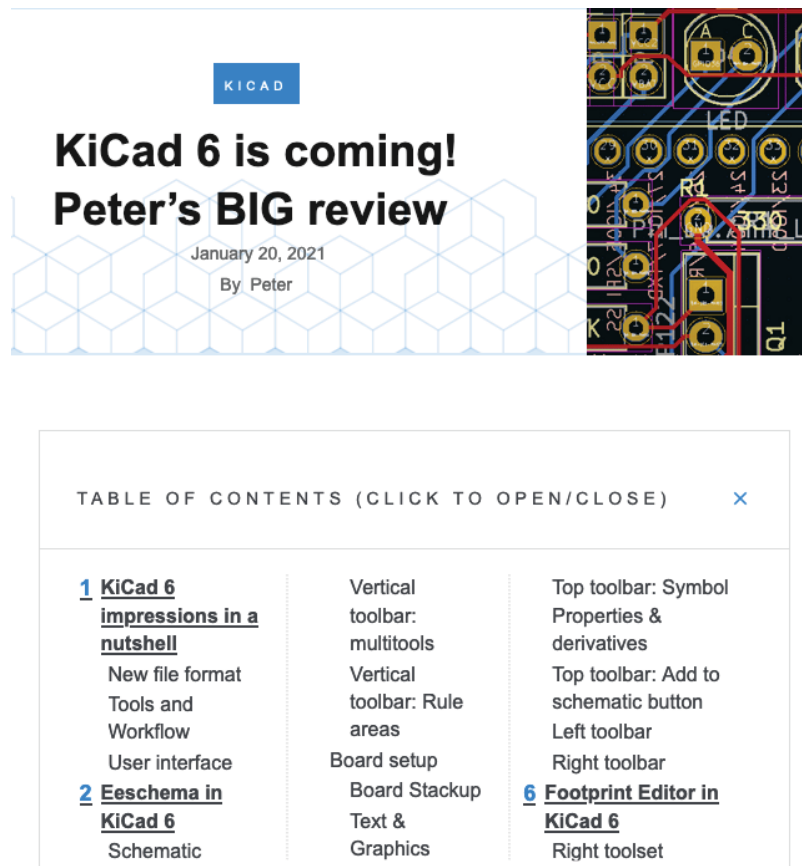


Figure 2.8.1: Peter's Big KiCad 6 review.

Here, I will list my top-three most significant changes in KiCad 6:

1. KiCad 6 has a new file format. The transition into this format, based on the S-Expressions standard, started in KiCad 5. With KiCad 6, the transition is complete.
2. The user interface is refreshed and modernized. While in KiCad 6, the user interface is still recognizable from the earlier versions, it follows

modern conventions on how the mouse and keyboard work. If you are coming from an earlier version of KiCad, you will use your existing KiCad knowledge. Icons have been redesigned. The menus and toolbars are better placed and organized. There is a single Preferences window.

3. The schematic editing paradigm is updated. Now, when you click on an element in the schematic editor, the element is selected. This was not the case in KiCad 5 and prior, causing much confusion and frustration.

Get the full details of what's new in KiCad 6 in my comprehensive [blog post](#).

## **Part 3: Project - A hands-on tour of KiCad - Schematic Design**

# 1. Introduction to schematic design and objective of this section

In Part 3 of the book (which you are reading now), you will learn about the basics of KiCad by working and completing a simple PCB project. In Part 3, the focus is on the schematic design, while in Part 4, the focus shifts to the layout design and the manufacturing. By the end of this project, you will have experienced the PCB design process using KiCad from start to finish.

While this first project is relatively simple, it will teach you the most important KiCad features and tools. You will develop skills that you will use in every future project regardless of its complexity.

As you work your way through this project, remember that you may need to reference the chapters in Part 13, Recipes, if you want to learn more details about specific features. To keep the size of the project concise, I have moved detailed descriptions of various features and tools to the end of the book.

The practical objective of this project is to design and manufacture a simple LED torch, like the one you see in 3.1.1 (below):

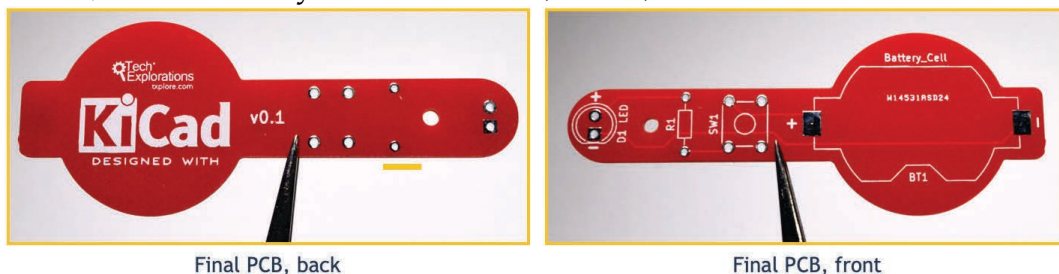


Figure 3.1.1: The manufactured project deliverable.

Most of the work will be in Eeschema (the schematic design editor) and Pcbnew (the layout design editor). At the end of this Part 3 of the book, the schematic design will look like this (Figure 3.1.2):

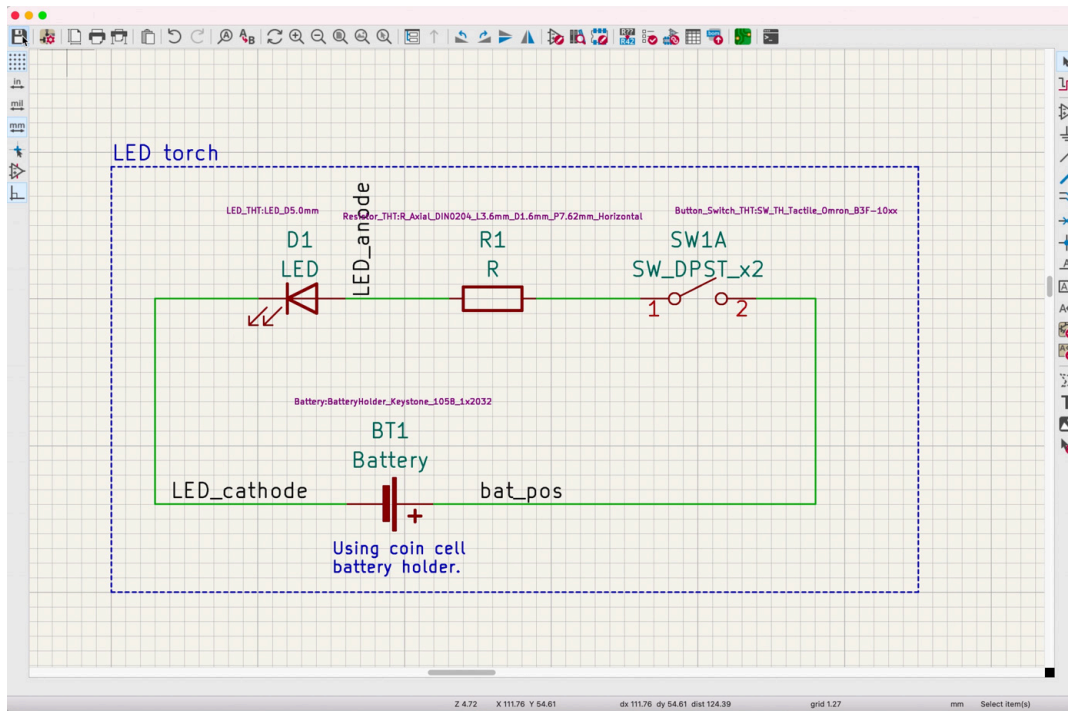


Figure 3.1.2: The final project schematic design.

The final layout will look like this (Figure 3.1.3):

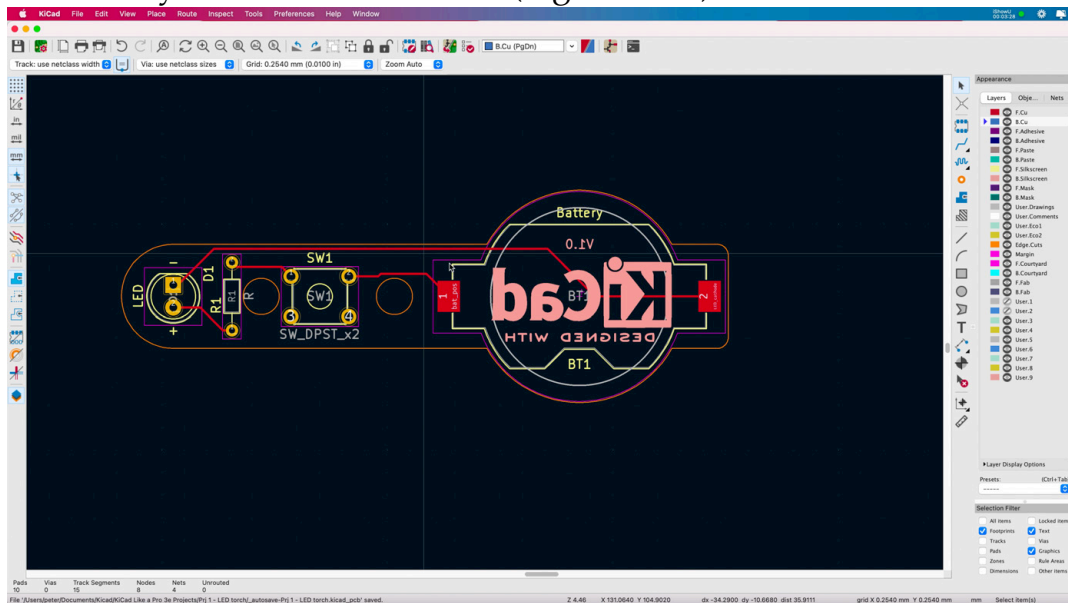


Figure 3.1.3: The final project layout design.

To guide the design of the PCB, I will be using the PCB design workflow that I outlined earlier in this book. I am also providing a summary in the next chapter.

The schematic (see Figure 3.1.2) contains only a few standard component symbols: an LED, a resistor, a button switch, and a battery holder. All these symbols are available in the KiCad libraries, so you will not need to get them

from external sources. Electrically, the circuit contains a single loop. When you press the button, the circuit closes, and the LED turns on.

Despite this being a simple project, you will learn how to find and add symbols to the editor, associate them with layout footprints, annotate them, wire them, create named nets, run the Electrical Rules Checker, and decorate the schematic with text and graphics.

In Part 4, you will learn how to import the schematic in Pcbnew and design the physical layout, complete with beautifully rounded corners, mounting holes, silkscreen graphics, and, of course, pass the design rules check before sending it to manufacturing.

## **Part 4: Project- A hands-on tour of KiCad - Layout**



# 1. Introduction to layout design and objective of this section

In the chapters of this part of the book, I will continue developing the LED torch project that I started in Part 3. At the end of the previous chapter, I completed the schematic design of the project PCB. I will now continue with the layout design.

To guide me with this work, I will follow the steps outlined in the layout design workflow that I outlined in Part 3.

To design the layout of the PCB, I'll be using Pcbnew. At the end of this part of the book, the PCB will look like this (Figure 4.1.1):

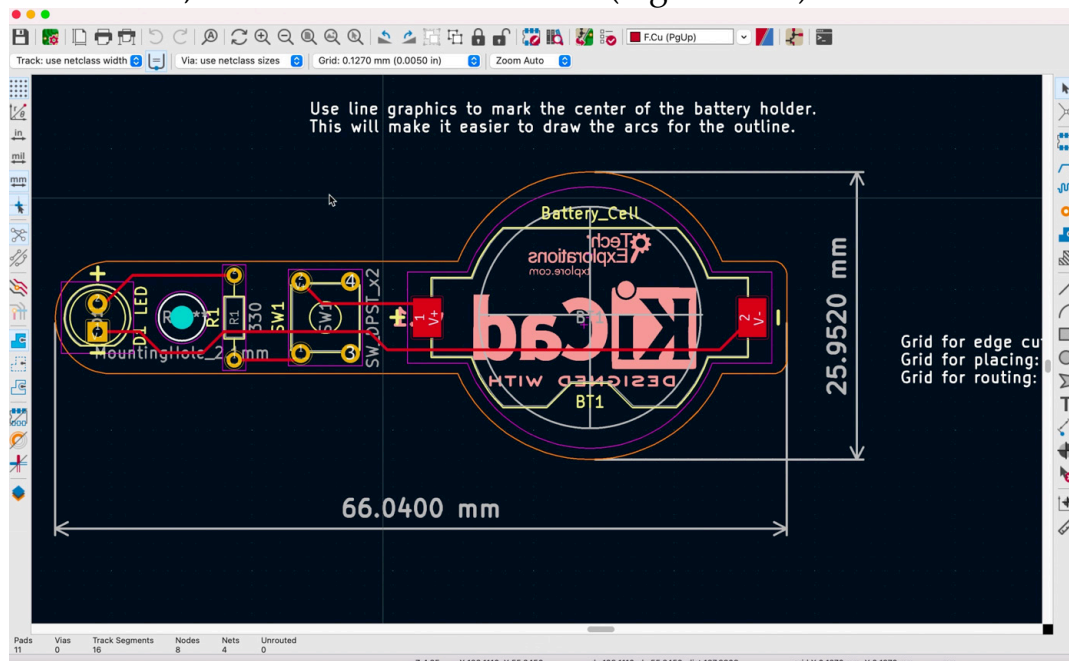


Figure 4.1.1: The final LED torch PCB layout.

Here is a 3D rendering, also made in KiCad (Figure 4.1.2):

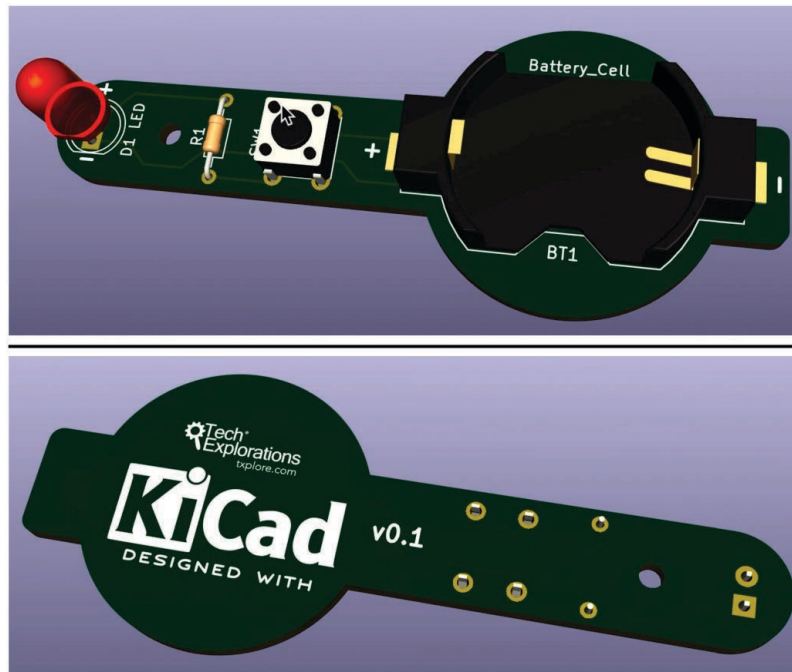


Figure 4.1.2: The final LED torch PCB layout in 3D.

The final PCB layout of this simple project contains several interesting elements:

- Both surface-mounted and through-hole components.
- Rounded edges moulded around the footprints of the PCB components.
- Silkscreen graphics (logos) and text.

Even though this is a simple project, it allows us to practice the essential skills for PCB design using KiCad.

Let's begin the layout design workflow with Pcbnew in the next chapter.

## **Part 9: Project - Design a simple breadboard power supply PCB**

# 1. Introduction

Welcome to Part 9 of this book! In the following chapters, you will learn how to design a simple yet practical PCB. This PCB is a component of a breadboard power supply. You can use this power supply to provide power to circuits implemented on a mini breadboard, which is a core part of electronics prototyping.

This project is an opportunity to use the knowledge you acquired in the last part of this book to create a non-trivial PCB. To design this PCB, you will be using the majority of the capabilities of KiCad's schematic and layout editors. You will also practice the PCB development workflow that you learned in Part 6 of the book.

The inspiration for the design of this PCB came from my work at creating small electronics circuits for my Arduino and ESP32 courses. When the circuit I was building on the breadboard needed more power than the MCU could provide, I would search through a range of possible options that usually included one of my bench-top power supplies and wires. The problem is that the bench-top power supplies are noisy (they have a large cooling fan), need some setup (select voltage, current), and their wires get in the way. In addition, I have drawers full of wall power supplies that I could be using. They are plug-and-play and silent.

For my breadboard power supply, I needed something that:

1. Plugs directly on the breadboard; therefore, there are no wires.
2. Have an on/off switch.
3. Can provide 5V and 3.3V power.
4. Can draw power from a range of wall power supplies, from 6V to

12V.

After some deliberation, I settled for a design like the one in the image below:

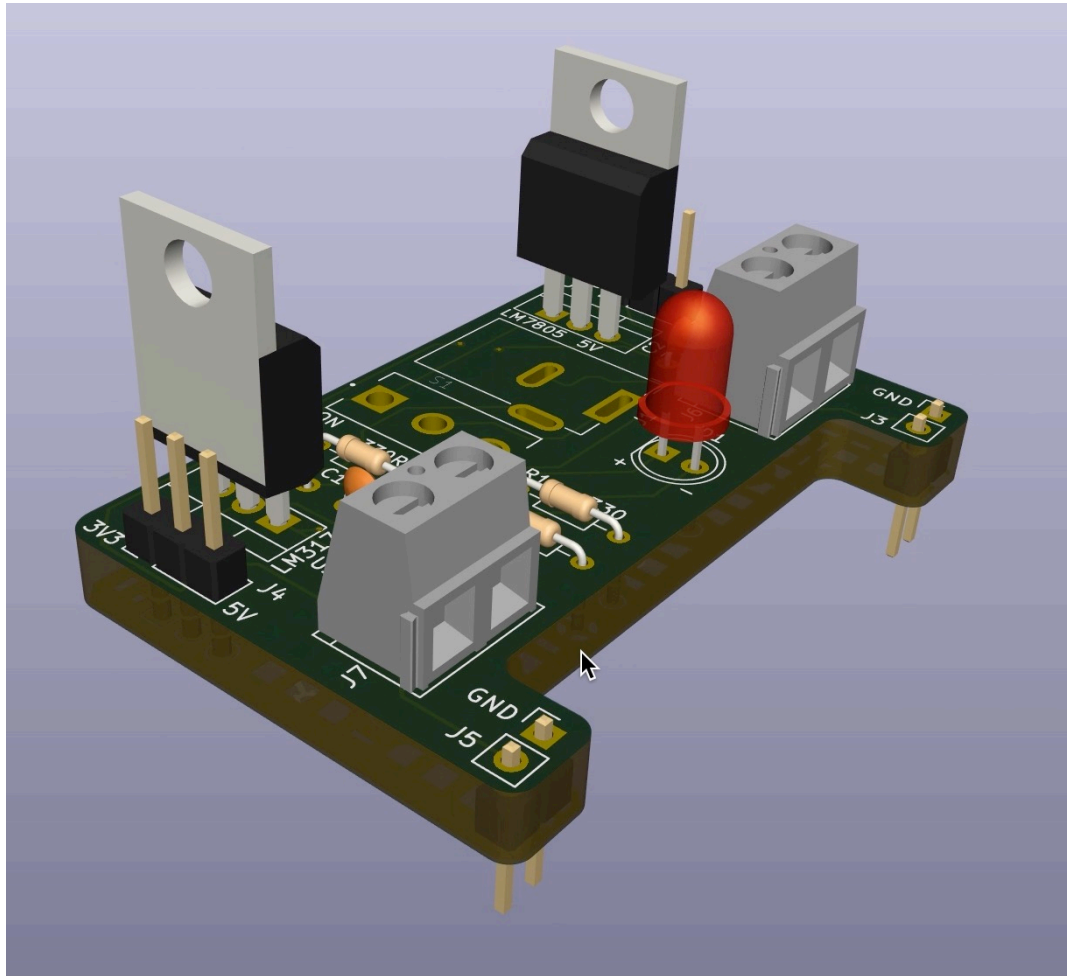


Figure 9.1.1: A 3D view of the breadboard power supply PCB.

The PCB's dimensions and shape are constrained by the dimensions of power row locations of the mini breadboard on which the PCB will connect. The connection between the breadboard and the PCB is made via two sets of pin headers. I have added two double screw terminals to provide an additional way to output power via jumper wires instead of the pins.

Below you can see a photo that shows the PCB against a mini breadboard:

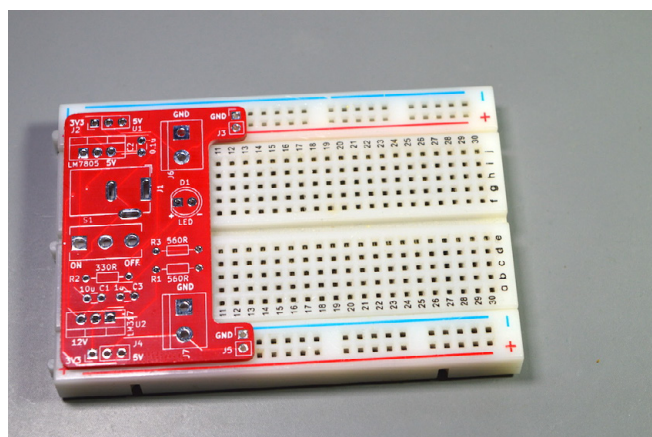


Figure 9.1.2: The power supply PCB over a mini breadboard.

When the PCB is attached to the left side of the breadboard, almost the entire right side is available for the prototyping circuit. The indentation between the pin headers also allows access to the first couple of columns in the breadboard that otherwise would have been covered.

To keep the power supply cable away from the prototyping area, I have placed the barrel connector on the left side of the PCB. The voltage selector switches are on the top and bottom of the board to make it easy to access.

Below you can see the final schematic design:

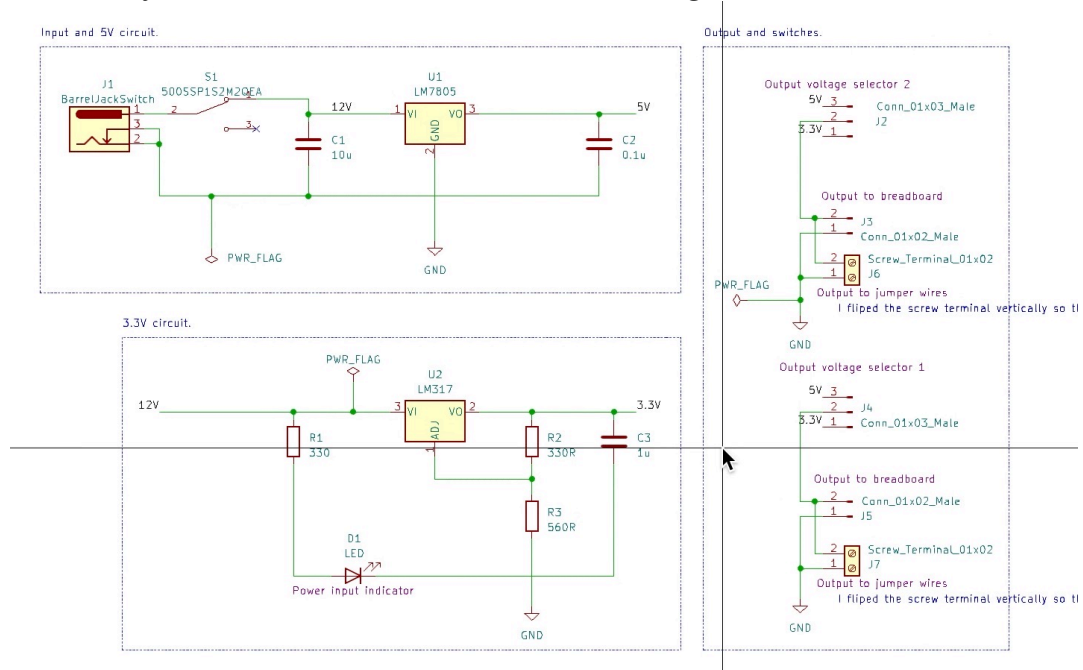


Figure 9.1.3: The project schematic design (final).

In the schematic above, you can see the power supply components arranged in three functional groups. You can see the two major components, the voltage regulators, inputs, outputs, and switches. You will work on the schematic design in the next chapter.

We'll do the schematic design in a single sheet. Most of the symbols needed come with KiCad's libraries, but one is available in the Digikey library.

Below, you can see the final layout design:

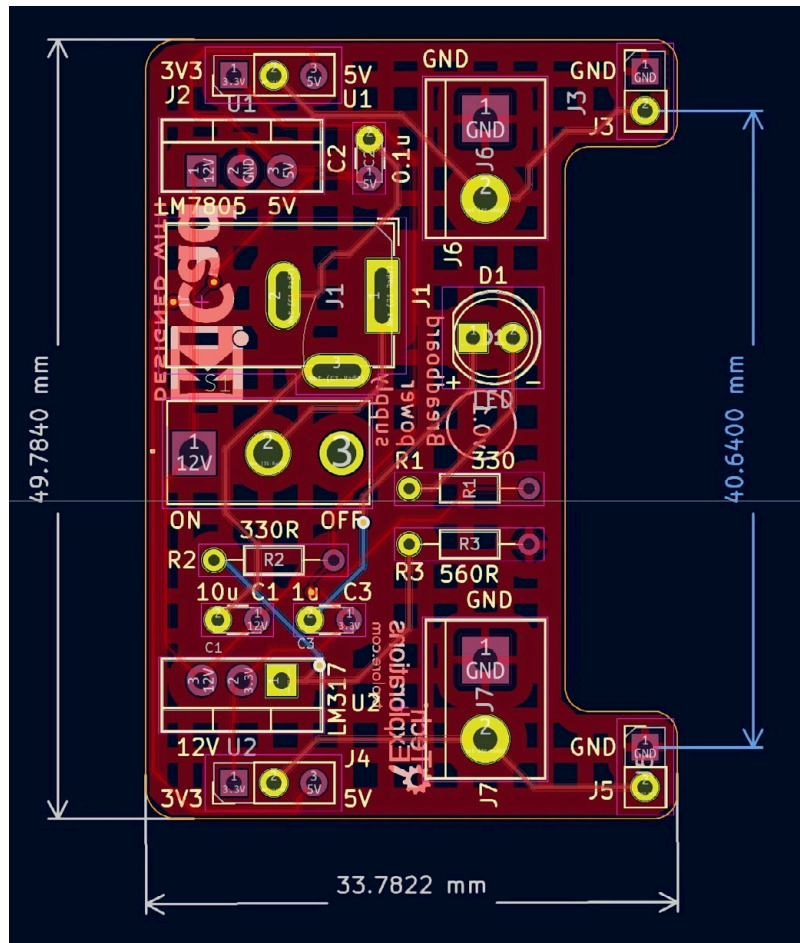


Figure 9.1.4: The project layout design (final).

The layout has several interesting features, including a composite shape with rounded corners, copper fills, all THT components to make it easy to assemble, a complete set of top and bottom silkscreen text and graphics, and is manually routed.

Perhaps the most challenging aspect of the layout design is its dimensions. The PCB's pin headers have to match precisely with the mini breadboard's power row pins. To achieve a good match, you will need to make accurate measurements on the breadboard and then use those measurements to precisely position the two double pin headers. Then you will design the board around those fixed footprints.

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
C1	10u	Capacitor_THT:C_Disc_D3.0mm_W1.6mm_P2.50mm

C2	1u	Capacitor_THT:C_Disc_D3.0mm_W1.6mm_P2.50mm
C3	0.1u	Capacitor_THT:C_Disc_D3.0mm_W1.6mm_P2.50mm
D1	LED	LED_THT:LED_D5.0mm
J1	Barrel_Jack_Switch	Connector_BarrelJack:BarrelJack_Horizontal
J2, J5	Screw_Terminal_01x02	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
J4, J6	Conn_01x02_Male	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
J3, J7	Conn_01x03_Male	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
R1, R2	330	Resistor_THT:R_Axial_DIN0204_L3.6mm_D1.6mm_P7.62mm_Horizontal
R3	560	Resistor_THT:R_Axial_DIN0204_L3.6mm_D1.6mm_P7.62mm_Horizontal
S1	EG1218	digkey-footprints:Switch_Slide_11.6x4mm_EG1218
U1	LM317_TO-220	Package_TO_SOT_THT:TO-220-3_Vertical
U2	LM7805_TO220	Package_TO_SOT_THT:TO-220-3_Vertical

Table 9.1.1: The Bill of Materials for this project.

Let's begin with the schematic design in the next chapter.



## **Part 10: Project - A 4 x 8 x 8 LED matrix array**

# 1. Introduction

Welcome to Part Ten! In the chapters that follow, you will design a PCB that can hold four 8x8 LED matrix displays controlled by an Arduino Pro mini. The board also includes two push buttons to which you can assign arbitrary functions. I plan to use mine as a [Pomodoro](#) timer. I will use one button to select the lap duration (say, 15, 20, or 25 minutes) and the other button to reset the timer. When the duration I have set expires, the display will blink to let me know.

You can see the two sides of the populated PCB in the photograph below:

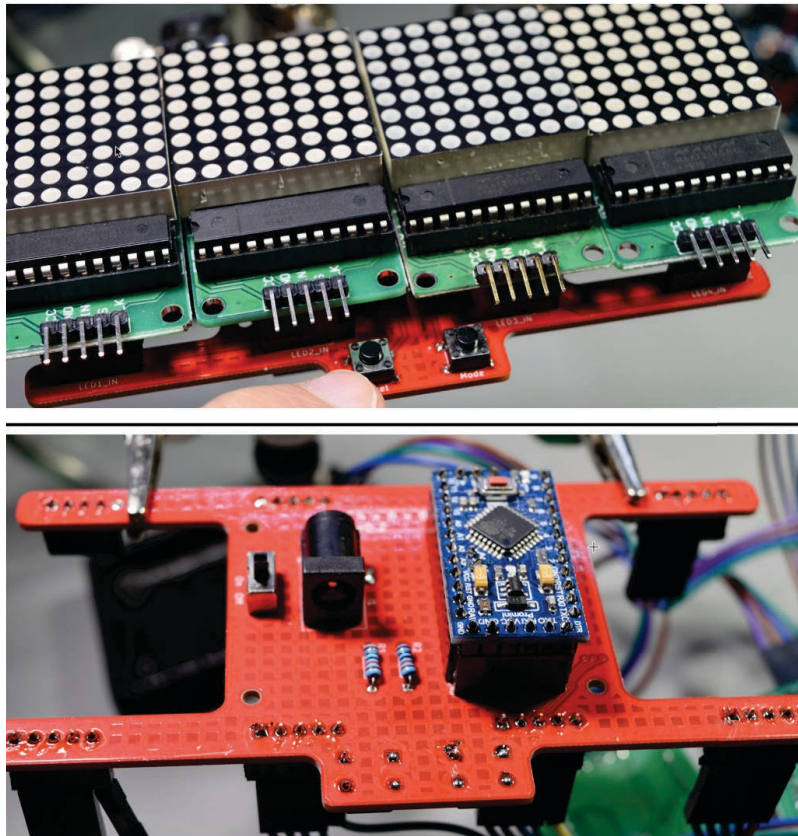


Figure 10.1.1: The PCB in this project, populated.

The inspiration for this project is that I forget to get up from my desk at regular intervals. I could use a desktop or phone Pomodoro app or even a classic mechanical Pomodoro timer, but why buy one when I can make one?

I decided to use an [Arduino Pro Mini](#) because:

- I have a lot of them.
- They are easy to find in the market

- They are very cheap.
- They are small.
- They have an onboard regulator.
- They are accurate enough to count short periods.
- I don't need a clock function, so the absence of a real-time clock is not an issue.

This board does not contain a UART to USB interface, so you must provide an external bridging device. This device is required for programming the microcontroller. I use a USD to serial adaptor from [Freetronics](#), but there are many other options in the market.

I decided to use the 8x8 LED matrix display module because I like its versatility. An 8x8 LED matrix display can show numbers, text, and simple graphics compared. I also like the way it looks from a distance and the ability to create simple animation. All this gives me scope to add features to my Pomodoro project in the future.

Below you can see the schematic for this PCB. This is the schematic you will create by the end of Chapter 10.2.

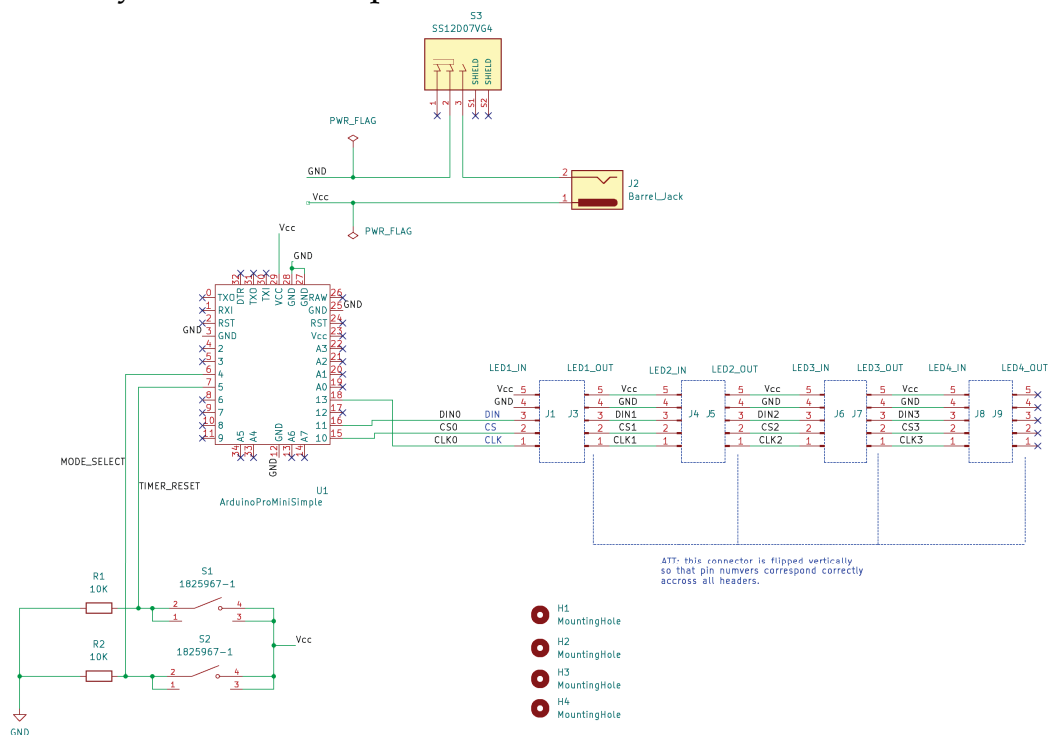


Figure 10.1.2: The project schematic.

To draw this schematic, I have opted to use line wires for all pins that are nearby. I have added net labels to all power, data, clock nets, and button signal nets. I was unable to find a schematic symbol for the Arduino Pro Mini board that I planned to use, so I created one, along with its matching

footprint. In the layout, I have included four mounting holes. To avoid getting error messages from the DRC, I have associated the mounting hole footprints with mounting hole symbols in the schematic.

Another consideration was how to deal with the LED matrix modules. I was not able to find a symbol and footprint for this device, so I had two options:

1. Create a custom symbol and footprint, as I did with the Arduino Pro Mini module.

2. Ignore the module, and concentrate on the headers.

Since I went with option 1 for the Arduino module, I opted for option 2 for the LED modules. Instead of treating each LED module as a single device, I treated it as a set of two-pin headers. My objective, then, was to wire the header symbols correctly and place their footprints precisely on the PCB (see the discussion on the PCB below).

Below is the layout of the PCB, as it will be at the end of Chapter 10.3:

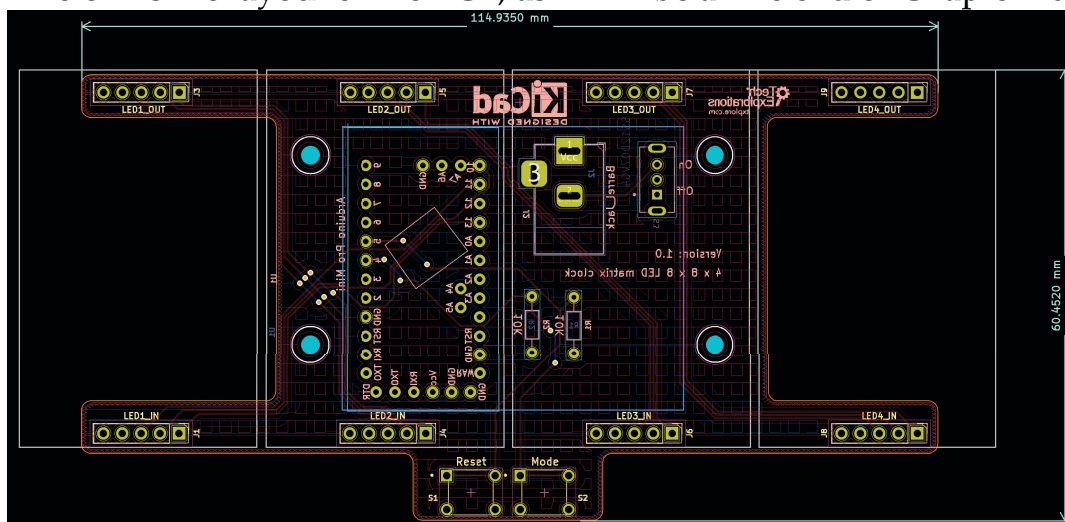


Figure 10.1.3: The project layout.

The dominating feature of this board is its shape. I wanted to experiment with a shape that uses “arms” that extend from its center to hold the LED modules rather than a conventional rectangular shape. I did not do this to reduce the manufacturing cost. Even though the shape you see above has two significant parts of the substrate material removed, the manufacturing cost relates to the all-inclusive height and width of the board (you can see those dimensions in the figure above). But I do think that the board with the four arms extending from the center looks great. Along with the rounded edges and the button notch at the bottom, I am satisfied with the physical design aspect of the board.

A significant challenge for the layout design of this board is the position of the pin headers for the LED modules. As I mentioned above, I decided to treat the LED modules as a set of two headers each (input and output). The positions of those headers must be very accurate. If the headers are too far from their neighbors, the four-module display will not look continuous but as four individual displays. If they are too close, the assembly will not be possible as there will not be enough space on the board to attach adjacent modules.

As a result, this project will give you the opportunity to use all of KiCad's measurement and alignment tools to make sure that the end product looks beautiful and works.

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
H1-H4	MountingHole	MountingHole:MountingHole_2.5mm
J1	LED1_IN	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J2	Barrel_Jack	Connector_BarrelJack:BarrelJack_Horizontal
J3	LED1_OUT	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J4	LED2_IN	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J5	LED2_OUT	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J6	LED3_IN	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J7	LED3_OUT	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J8	LED4_IN	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
J9	LED4_OUT	Connector_PinHeader_2.54mm:PinHeader_1x05_P2.54mm_Vertical
R1, R2	10K	Resistor_THT:R_Axial_DIN0204_L3.6mm_D1.6mm_P7.62mm_Horizontal

S1, S2	1825967-1	1825967-1:SW_1825967-1
S3	SS12D07VG4	SS12D07VG4:SW_SS12D07VG4
U1	ArduinoProMiniS imple	DesktopLibrary:ArduinoProMiniCustom

Table 10.1.1: The Bill of Materials for this project.

Apart from the custom symbol and footprint Arduino Pro Mini, I used Snapeda to find the symbol-footprint pairs for the two buttons (S1 and S2) and the power barrel connector (S3).

Let's begin with the schematic design in the next chapter.

## **Part 11 : Project - MCU datalogger**

# 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):

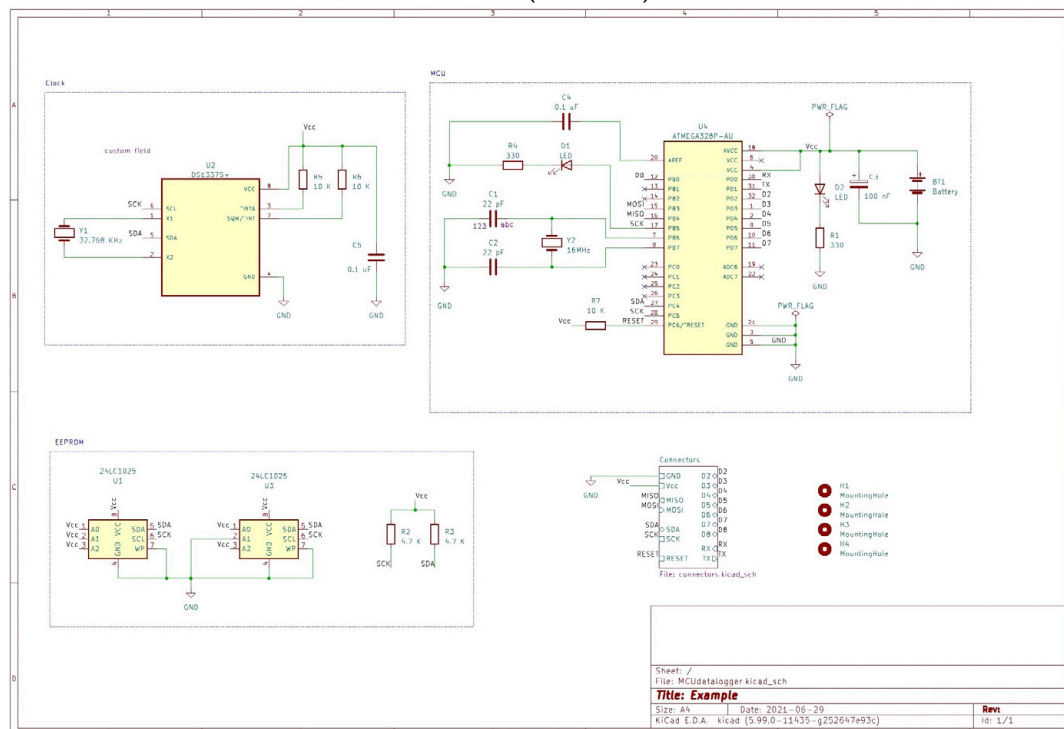


Figure 11.1.1: Sheet 1 of the project's final schematic design.



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

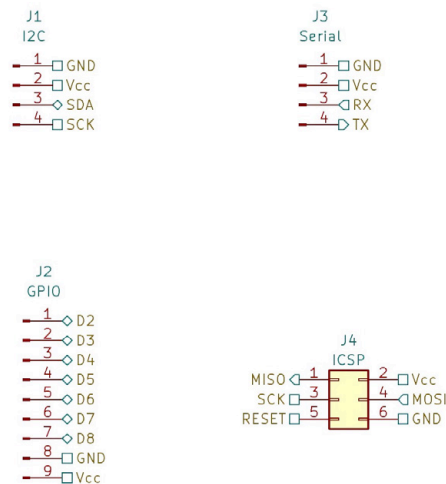


Figure 11.1.2: Sheet 2 of the project's final schematic design.

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:

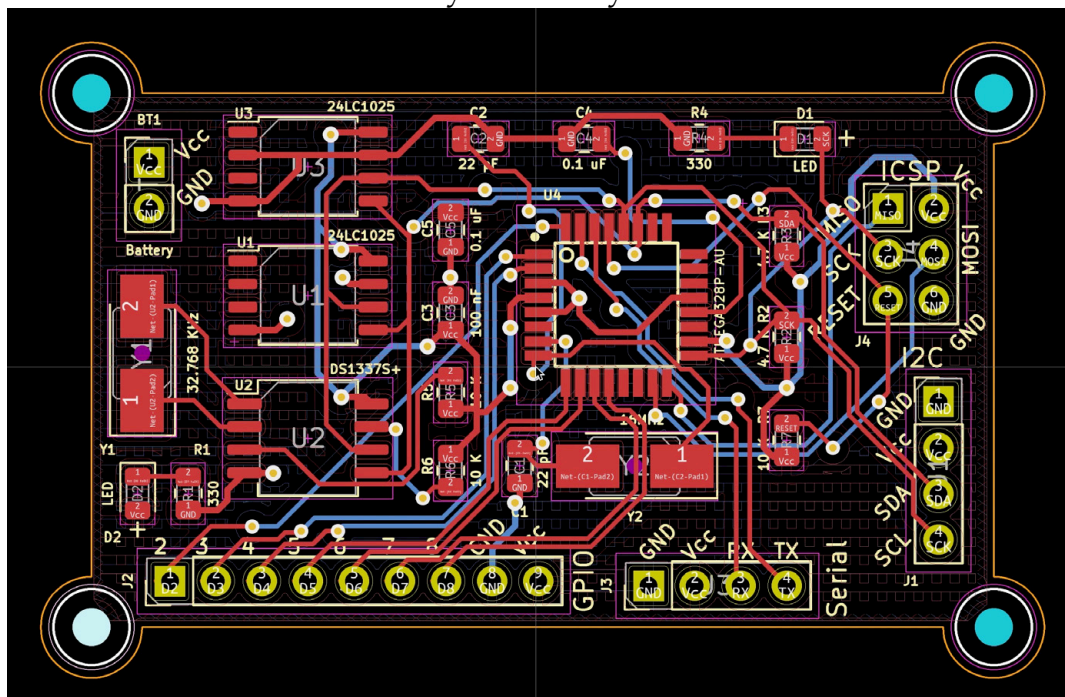


Figure 11.1.3: The project's final layout design (two layers).

You can see the final four-layer PCB below:

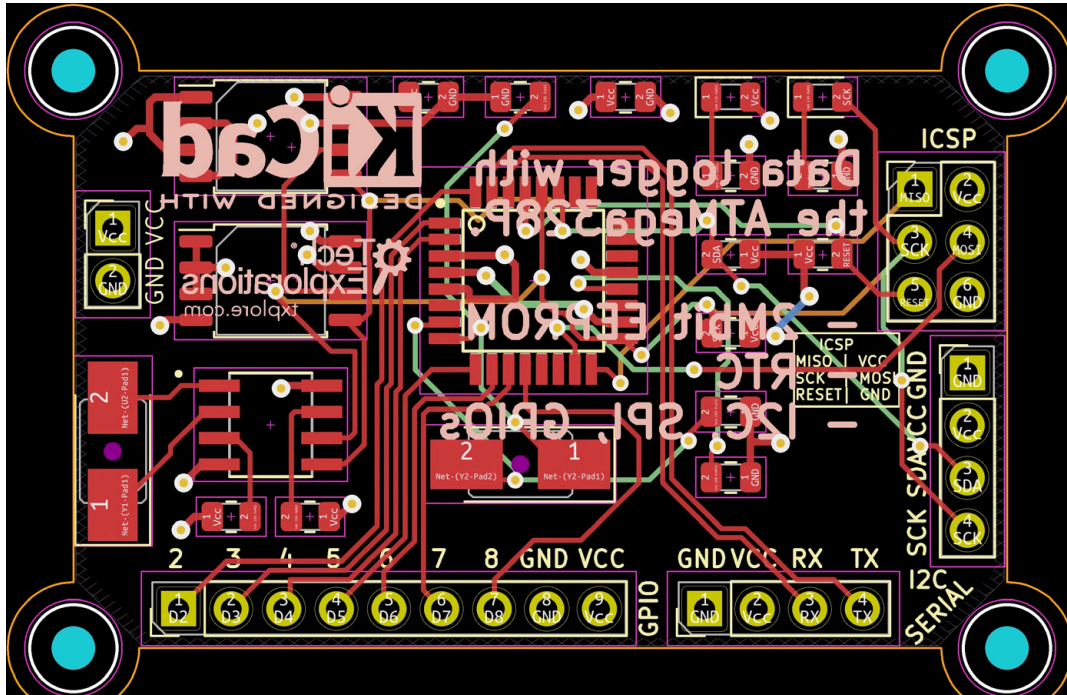


Figure 11.1.4: The project's final layout design (four layers).

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:S0IC127P600X175-8N
U1 , U3	24LC1025	Package_S0:S0IC-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.

## 1. Project - Introduction

Welcome to Part 12 of this book. In the chapters that follow, you will design an ESP32 development board. To assist you with the design process, you will use the reference schematic design from Espressif, the original designer and manufacturer of the original ESP32 dev kit module.

By working on this project, you will learn how to use and modify reference designs. The ESP32 is a widely used and understood board. It is compact, yet packs significant computing power and capabilities. It's circuit board is more complex compared to those in the previous projects of this book, but not too complex to make this project long and tedious.

Below you can see the layout design as it will look once the project is completed.

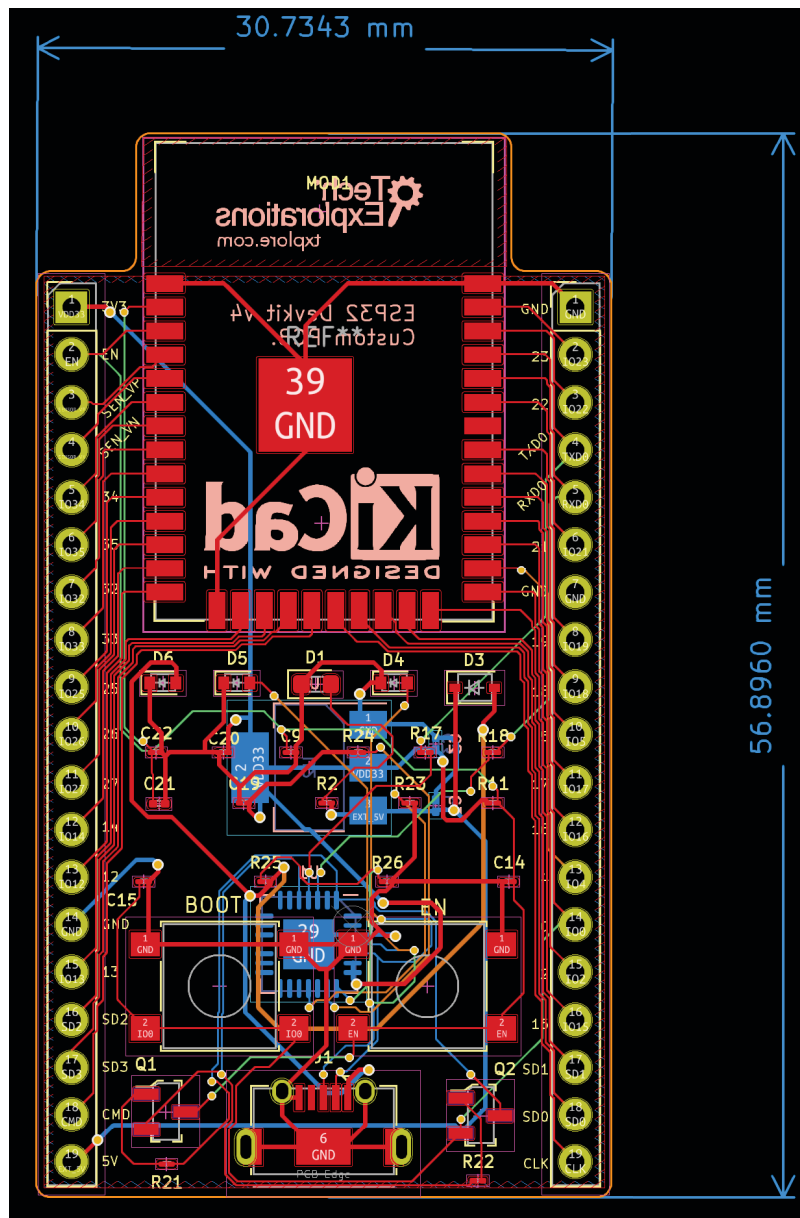


Figure 13.1.1: The ESP32 development kit clone project deliverable.

In this instance of the development kit, I have retained the form factor and shape of the original ESP32 development kit. This is a four-layer board, with the ESP32 module at the top with its integrated antenna extending outside of the board, the power and serial communications port at the bottom, and the pin headers along the sides. The board also features the two control buttons, “BOOT” and “EN”, and uses SMD components on the front and back. Here’s a view of the reference layout:

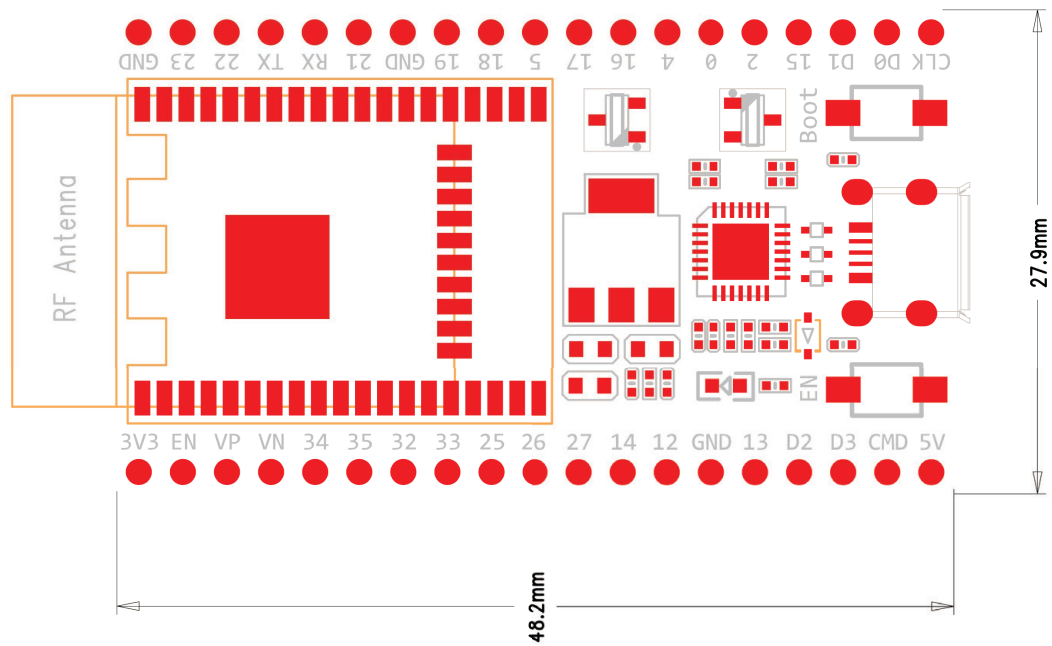


Figure 13.1.2: A view of the ESP32 development kit reference layout.

Once you understand the design elements of this board, you will be able to modify its shape and component placement to match the requirements of your project. For example, in my instance of the board, I have chosen to place the voltage regulator in the back of the PCB, even though in the reference design the regulator is in the front. I did this because I wanted to allow more room in the front for small components, like the resistors, to make assembly by hand easier.

Below you can see the final schematic.

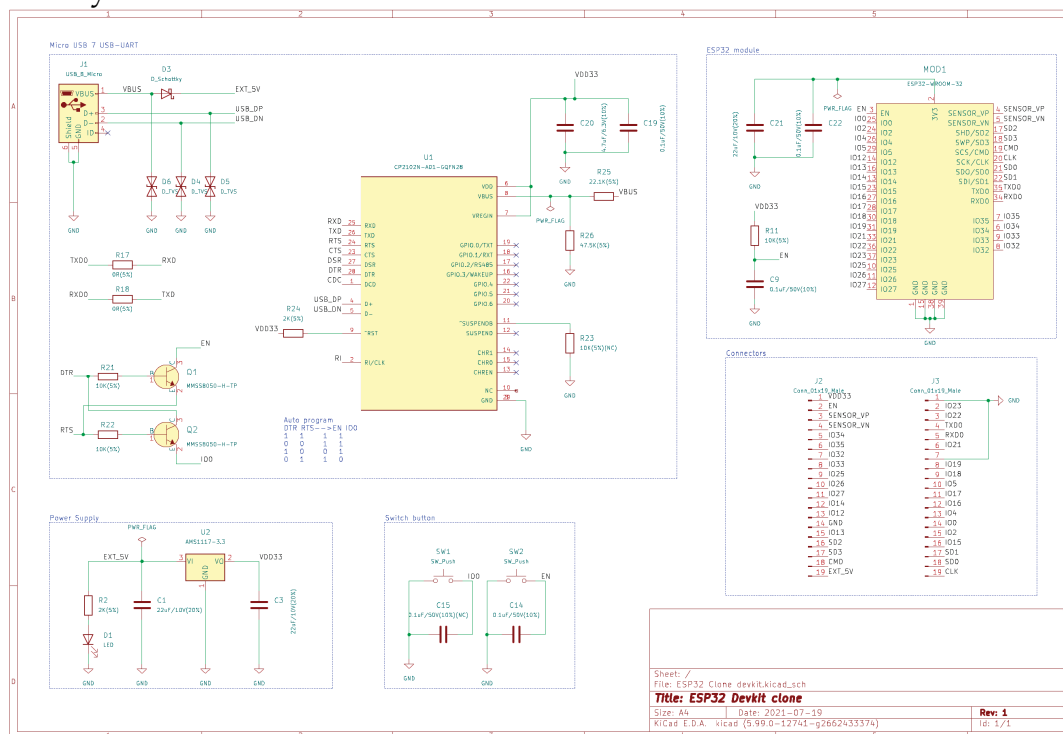




Figure 13.1.3: The ESP32 development kit clone project schematic.

Thanks to the ESP32 module's highly integrated design, the board that hosts the module (i.e. the development kit) is relatively simply. In the instance of the development kit design that you see above, I have arranged the components in five functional groups, and I have drawn all details from the original Espressif reference schematic design. You can see the reference schematic design below:

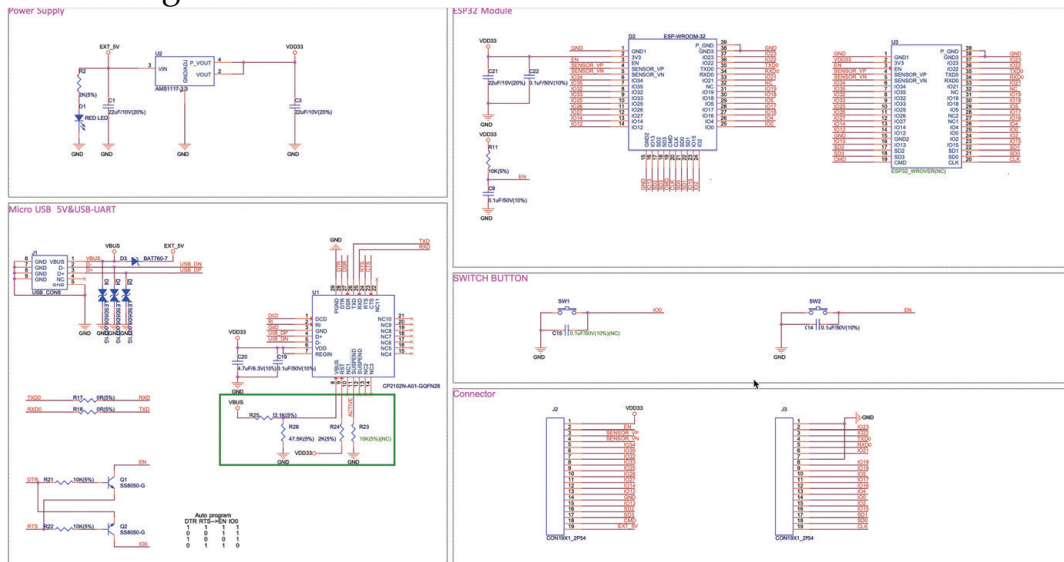


Figure 13.1.4: The ESP32 development kit reference schematic.

You will learn more about this in chapter 2.

Below you can see the 3D rendering of the development kit.

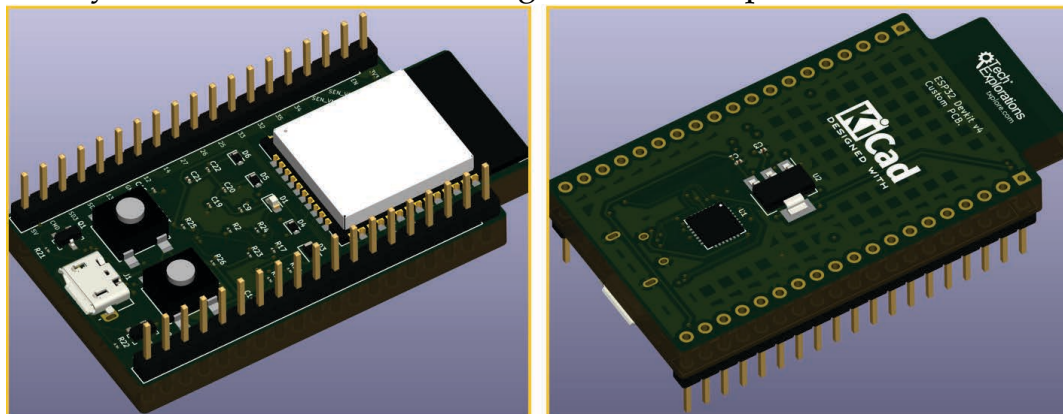


Figure 13.1.5: A 3D rendered view of the project deliverable.

The 3D rendering contains models for all components and is accurate in terms of what the final manufactured board would look like.

The process of customising an existing board begins by finding the original schematic and layout designs. In the case of the ESP32 development kit, you can find the official reference designs in the Espressif website ([https://](https://www.espressif.com/)

[www.espressif.com/en/support/documents/technical-documents](http://www.espressif.com/en/support/documents/technical-documents)). I have used the references design for the ESP32-DevKitC-v4 board. You can use the search filter to find this design (or try my search URL "<https://www.espressif.com/en/support/documents/technical-documents?keys=ESP32-Devkit>").

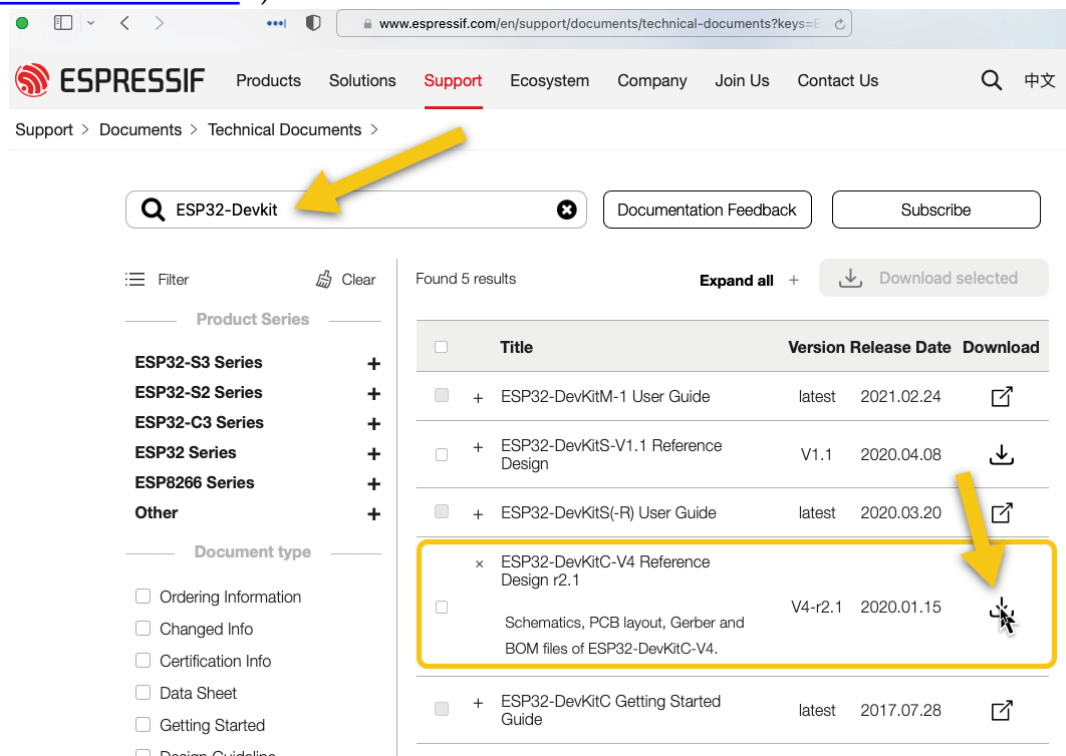


Figure 13.1.6: The source documentation and designs for this project.

Download the ZIP file that contains everything you will need for this project: the schematics, the PCB layout, and the BOM. The ZIP also contains a set of Gerber files which you do not need (you will generate a new set at the end of the project).

The reference BOM is very important because it gives you all the components and their values that you will need for your custom board. These components and values are tested by Espressif and used in millions of manufactured board, so you can be confident that they will work. I used the reference BOM to help me select symbols and footprints for my instance of the board. The project BOM that you see below contains information that I sourced from the reference BOM.

Refere nce	Value	Footprint
C1, C2, C21	22uF / 10V(20%)	Capacitor_SMD:C_0201_0603Metric



C9, C14, C19, C22	0.1uF / 50V(10%)	Capacitor_SMD:C_01005_0402Metric
C15	0.1uF / 50V(10%) (NC)	Capacitor_SMD:C_01005_0402Metric
C20	4.7uF / 6.3V(10%)	Capacitor_SMD:C_01005_0402Metric
D1	LED	LED_SMD:LED_0603_1608Metric
D3	D_Schottky	Diode_SMD:D_SOD-323
D4, D5, D6	D_TVS	Diode_SMD:D_SOD-523
J1	USB_B_Micro	Connector_USB:USB_Micro-AB_Molex_47590-0001
J2, J3	Conn_01x19_Male	Connector_PinHeader_2.54mm:PinHeader_1x19_P2.54mm_Vertical
MOD1	ESP32-WROOM-32	digikey-footprints:ESP32-WROOM-32D
Q1, Q2	MMSS8050-H-TP	digikey-footprints:SOT-23-3
R2, R24	2K(5%)	Resistor_SMD:R_01005_0402Metric
R7, R18	0R(5%)	Resistor_SMD:R_01005_0402Metric
R11, R21, R22	10K(5%)	Resistor_SMD:R_01005_0402Metric
R23	10K(5%)(NC)	Resistor_SMD:R_01005_0402Metric
R25	22.1K(5%)	Resistor_SMD:R_01005_0402Metric
R26	47.5K(5%)	Resistor_SMD:R_01005_0402Metric
SW1, SW2	SW_Push	Button_Switch_SMD:SW_SPST_B3S-1000
U1	CP2102N-A01-GQFN28	Package_DFN_QFN:TQFN-28-1EP_5x5mm_P0.5mm_EP2.7x2.7mm
U2	AMS1117-3.3	Package_TO_SOT_SMD:SOT-223-3_TabPin2

Table 13.1.1: The Bill of Materials for this project.

The BOM shows several SMD components that are very small, such as the 0402 resistors. If you plan to assemble your custom board by hand, you should consider the difficulty of working with such small components. But, since this is your custom board, you can choose to replace those small components with alternatives that you feel more comfortable to work with.

This is just an example of how you can modify a reference design to fit your specific requirements.

The main objectives of this project are:

1. To help you practice skills you acquired in previous projects.
2. To gain experience in the design of a PCB based on an existing reference design.
3. To gain experience in creating dense, four-layer PCBs.

I used Snapeda to find the symbol-footprint pairs for MOD1 (the ESP32 module). You should be able to find all other symbols and footprints in KiCad's libraries.

Let's begin.