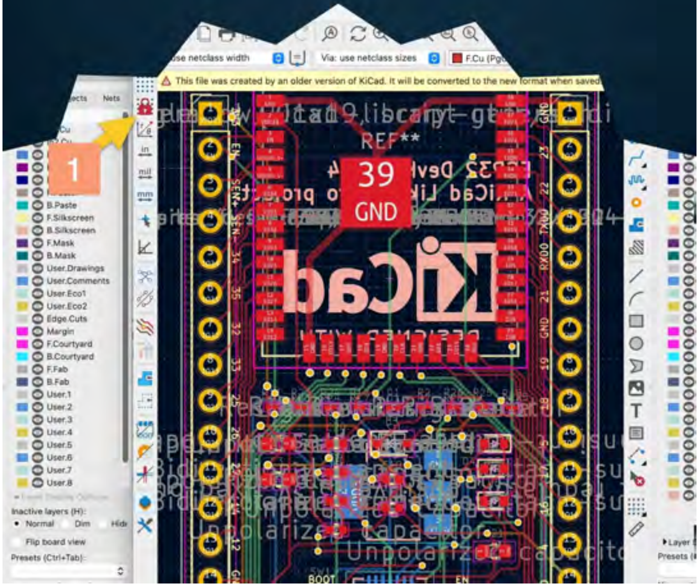
Tech Explorations
www.techexplorations.com

KICAD

A "GETTING STARTED" GUIDE



Peter Dalmaris, PhD

KiCad

Getting Started

Welcome to this special collection of articles, meticulously curated from the Tech Explorations blog and guides. As a token of appreciation for joining our email list, we offer these documents for you to download at no cost. Our aim is to provide you with valuable insights and knowledge in a convenient format. You can read these PDFs on your device, or print.

Please note that these PDFs are derived from our blog posts and articles with limited editing. We prioritize updating content and ensuring all links are functional, striving to enhance quality continually. However, the editing level does not match the comprehensive standards applied to our Tech Explorations books and courses.

We regularly update these documents to include the latest content from our website, ensuring you have access to fresh and relevant information.

What is a PCB?

An overview of the components of a PCB, how a PCB looks, and the terminology that we use.

My earliest experiences with PCBs

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.

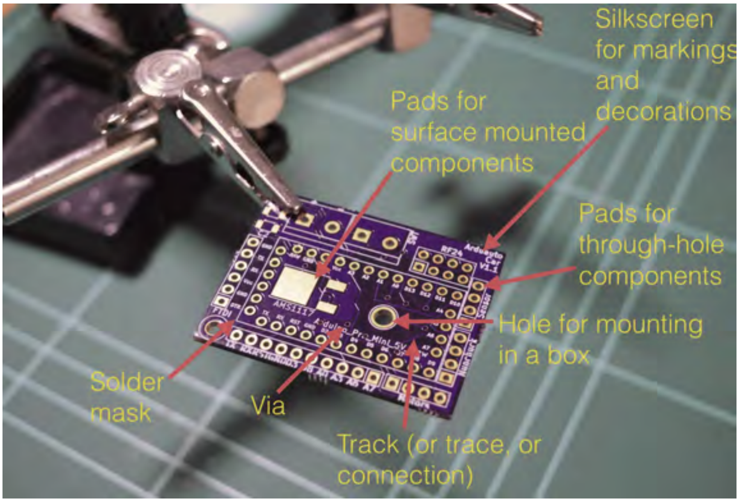

Silkscreen for markings and decorations
Pads for surface mounted components
Pads for through-hole components
Hole for mounting in a box
Solder mask
Via
Track (or trace, or connection)

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

Components of a PCB

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).

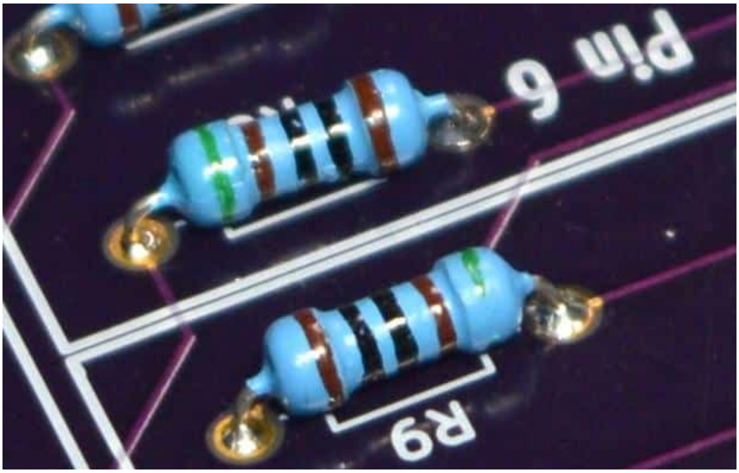


Figure 1.1.2: A through-hole component attached to a PCB.

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).

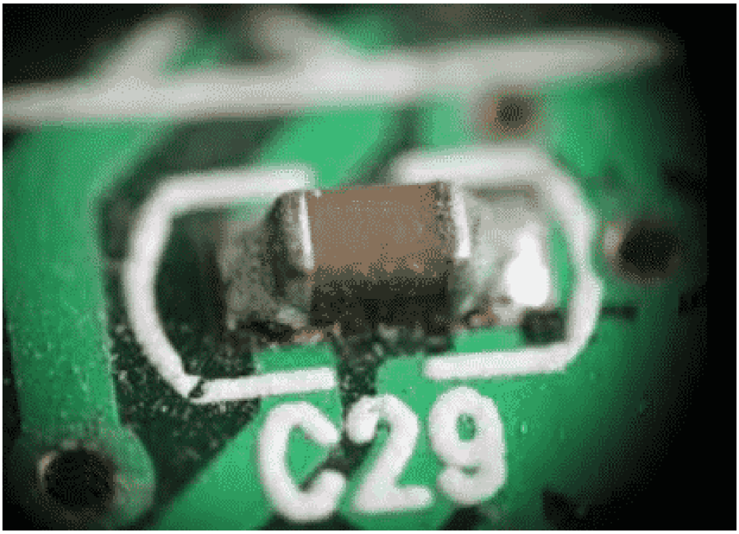


Figure 1.1.3: A surface-mounted component attached to a PCB.

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.

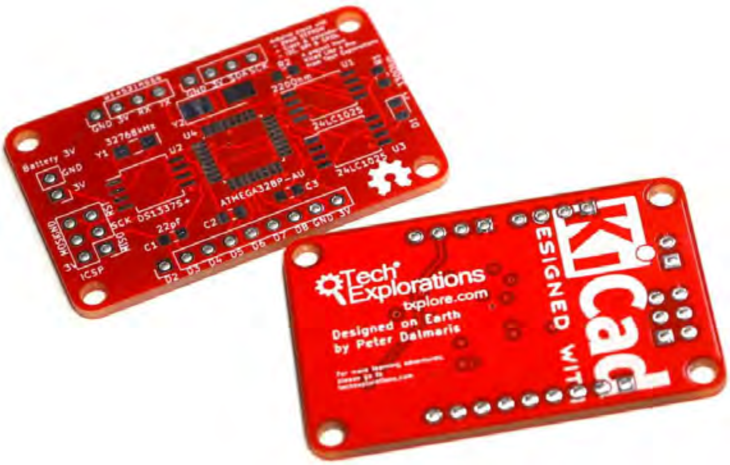


Figure 1.1.4: The white letters and lines is the silkscreen print on this PCB.

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.

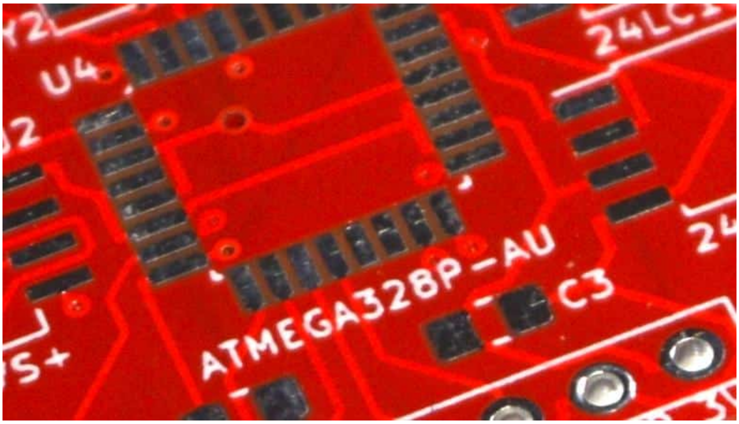


Figure 1.1.5: The bright red lines connecting the holes are tracks.

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.

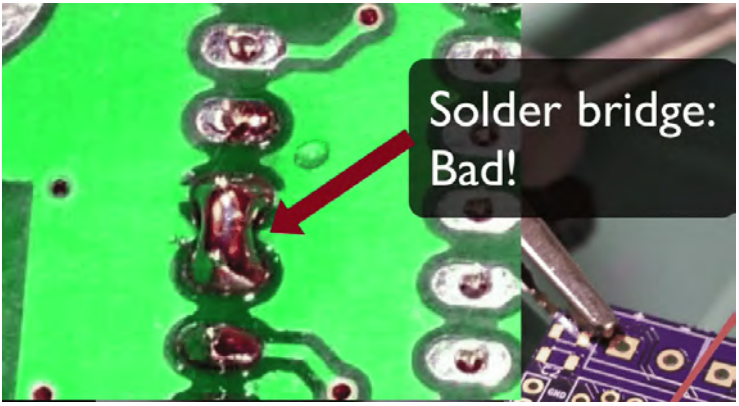


Figure 1.1.6: A solder bridge like this one is a defect that a solder mask can prevent.

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.

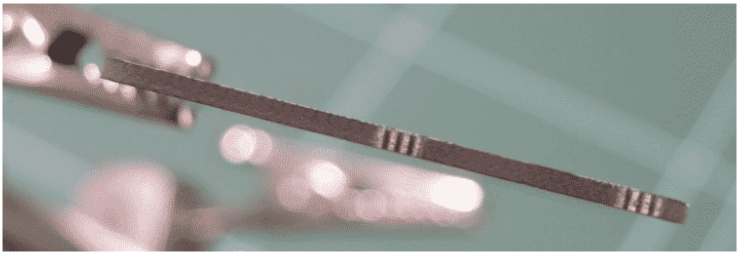


Figure 1.1.7: This PCB has a thickness of 1.6mm, and is made of fiberglass.

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 tined on the inside, electrically connecting the front and back.

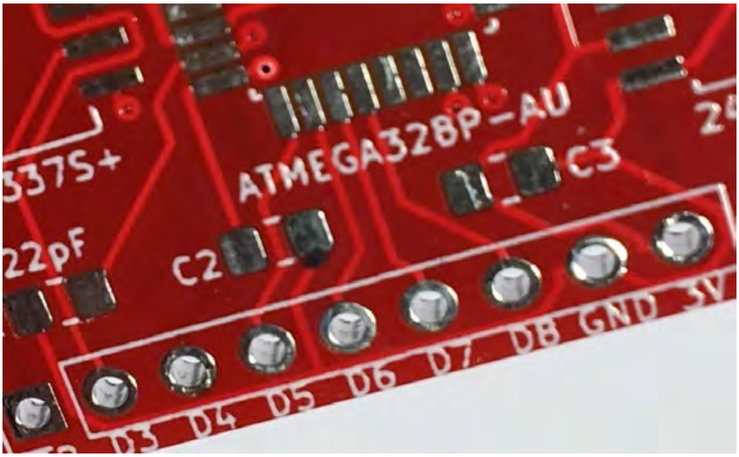


Figure 1.1.8: A closeup view of the top layer.

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.

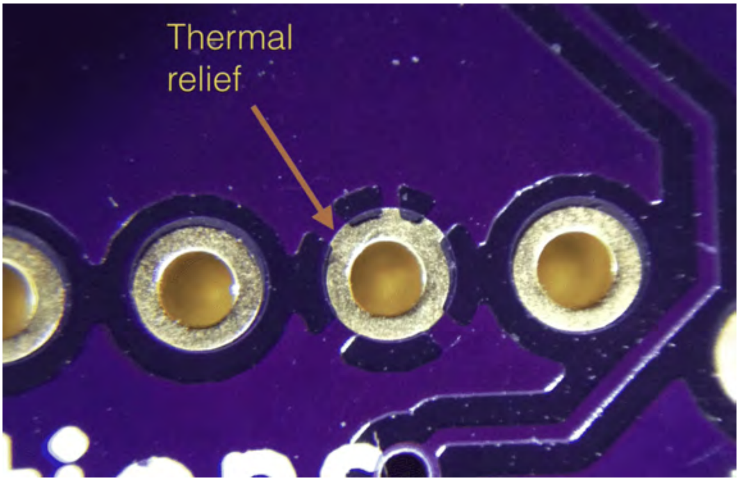


Figure 1.1.9: Thermal relief connects a pad to a copper region.

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.

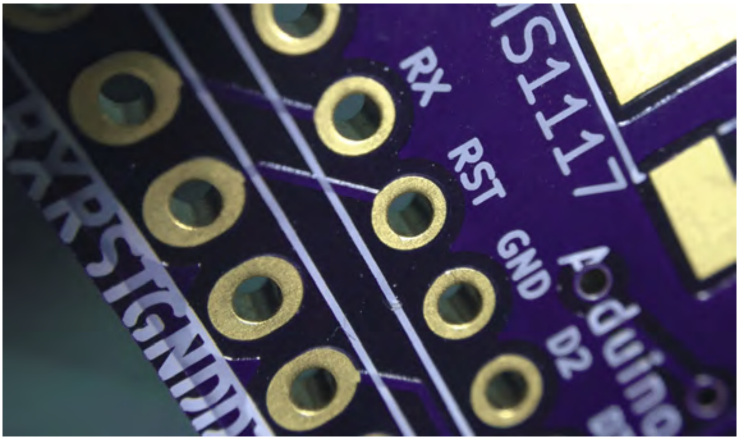


Figure 1.1.10: The plating of the holes covers the inside of the hole and connects that front end with the back end.

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.

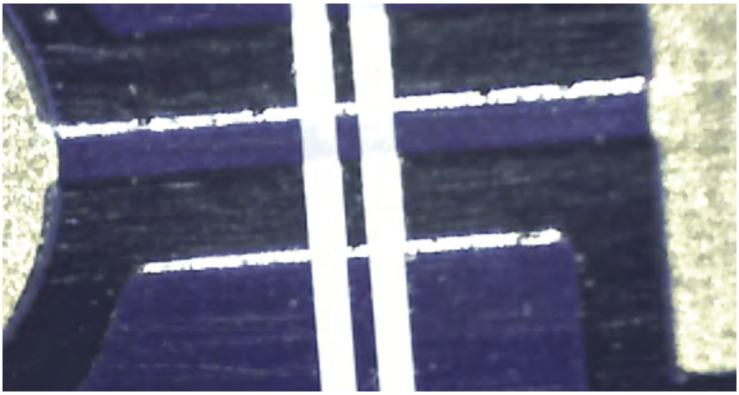


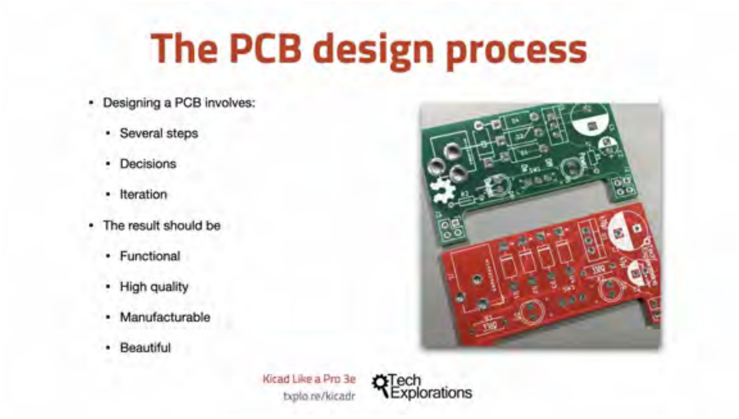
Figure 1.1.11: A detail of this example board at 200 times magnification.

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.

The PCB design process

In this article I will demonstrate the technical aspects of designing a PCB in KiCad.

To design a printed circuit board, you must go through a series of steps, make decisions, and iterate until you are satisfied with the outcome. The process of creating the plans for a printed circuit board is referred to as PCB design.



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

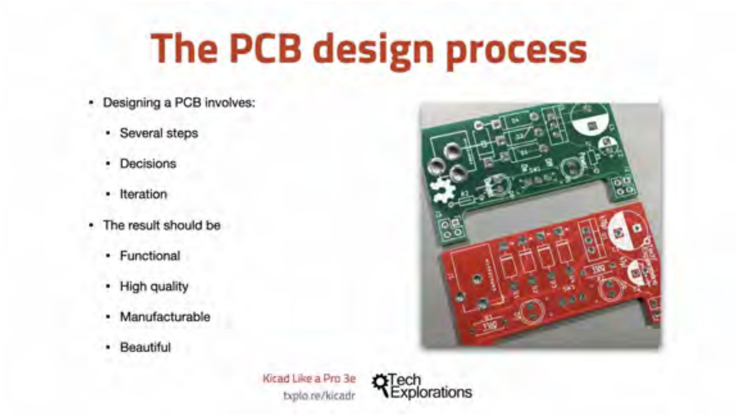
Introduction

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



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 the **KiCad Like a Pro 3e** course and eBook, 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

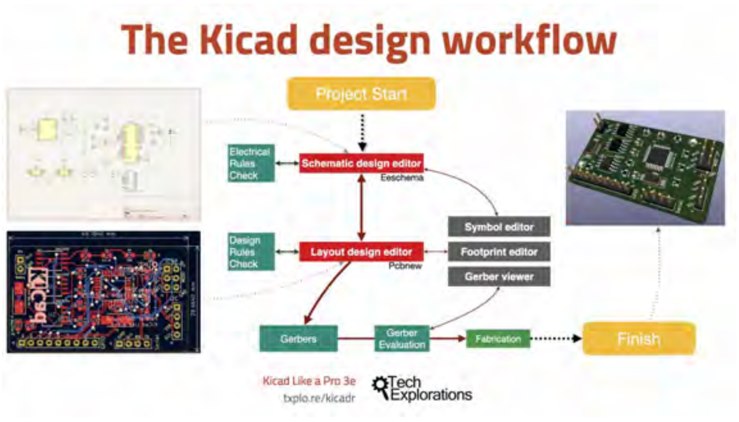


Figure 1.2.2: KiCad is a suite of applications.

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 customize 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 the **KiCad Like a Pro 3e** course and eBook, 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:

- Step 1 is the schematic design using the **schematic design editor (Eeschema)**;
- 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.

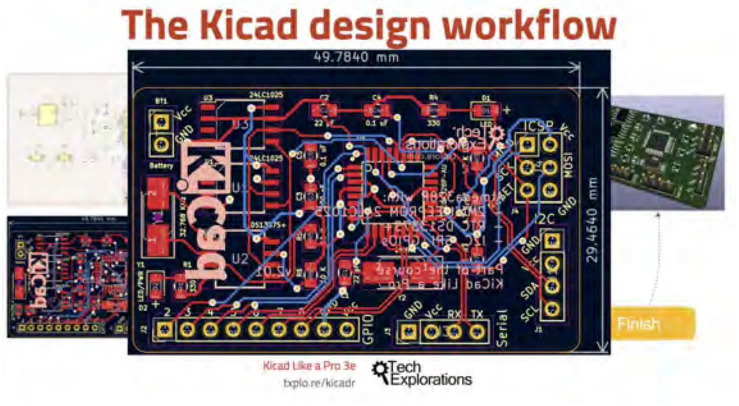


Figure 1.2.3: The KiCad layout file contains information about the physical PCB.

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 with the course or eBook).

Let's walk through 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 guide, 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.

PCB terminology

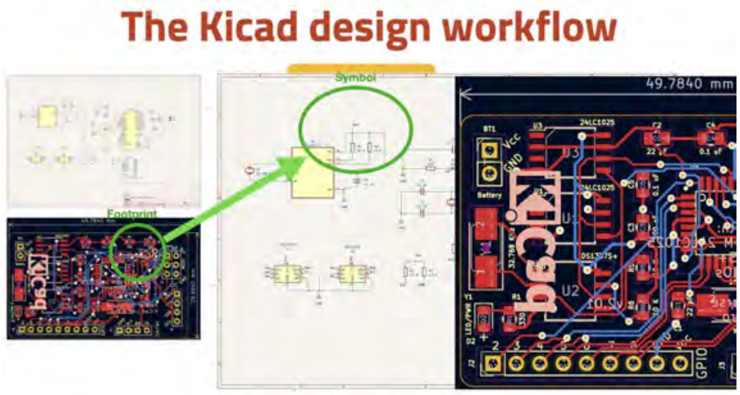


Figure 1.2.4: Symbols and footprints.

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.

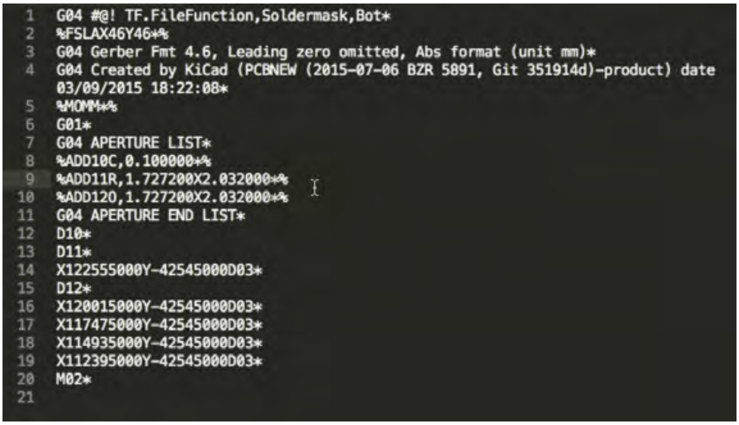
The industry standard for this is a format called 'Gerber.' Gerber files contain several related files, with one Gerber file per layer on your PCB, and contain instructions that the fabrication house needs to manufacture your PCB.

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

PCB Fabrication

Assume you've finished laying out your board in KiCad and are ready to build it. What are your alternatives?

In this article, we will compare the benefits of making a PCB at home versus using professional PCB manufacturing services.



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

Make a PCB at home

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.

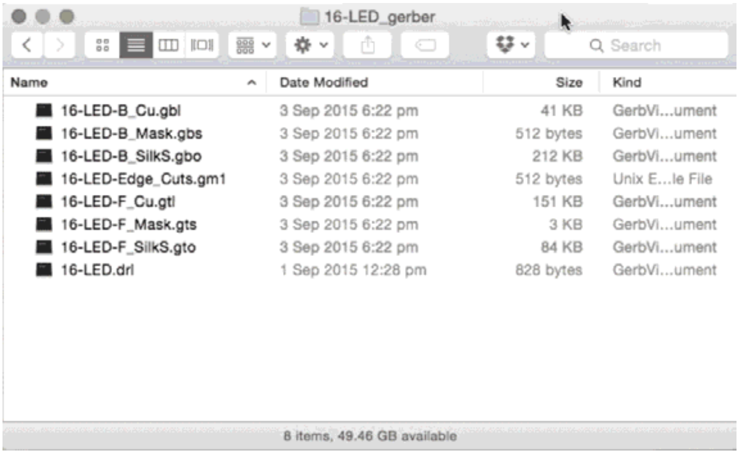
Professional PCB manufacturing services

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 OSH Park 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.

What is a Gerber file

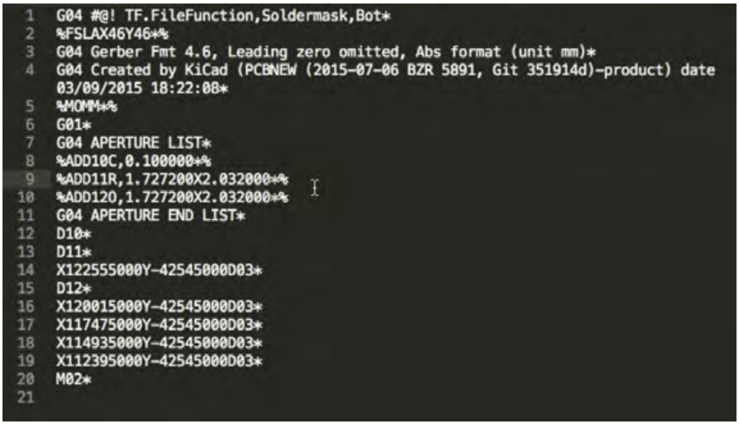


Name	Date Modified	Size	Kind
16-LED-B_Cu.gbl	3 Sep 2015 6:22 pm	41 KB	GerbVi...ument
16-LED-B_Mask.gbs	3 Sep 2015 6:22 pm	512 bytes	GerbVi...ument
16-LED-B_SilkS.gbo	3 Sep 2015 6:22 pm	212 KB	GerbVi...ument
16-LED-Edge_Cuts.gm1	3 Sep 2015 6:22 pm	512 bytes	Unix E...le File
16-LED-F_Cu.gtl	3 Sep 2015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

8 items, 49.46 GB available

Figure 3.3.1: 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.



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

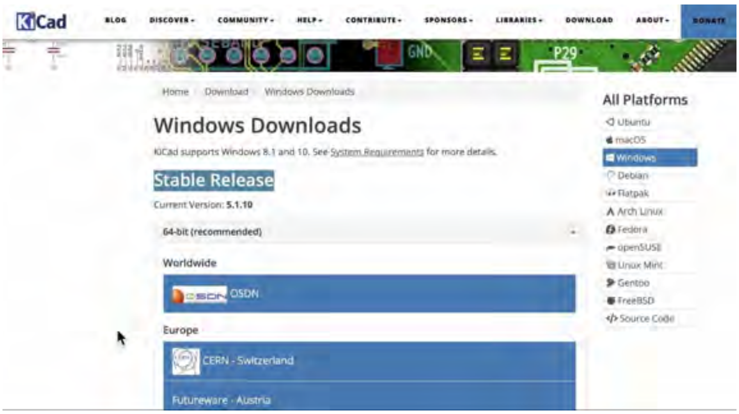
Figure 3.3.3: Gerber files contain text

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.

Get KiCad for your operating system

This article outlines the installation method for KiCad on the various supported operating systems, as well as KiCad's nightly development builds and source code.



Windows Downloads

KiCad supports Windows 8.1 and 10. See System Requirements for more details.

Stable Release

Current Version: 5.1.10

Worldwide

Europe

The KiCad installation process

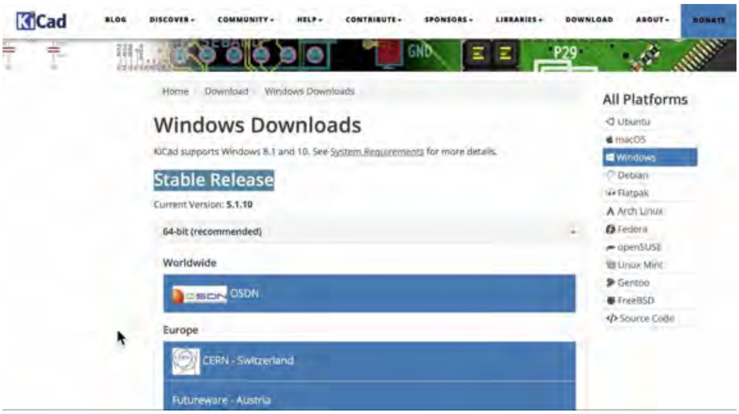
It is now time to download your copy of KiCad and install it on your computer.

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.



Windows Downloads

KiCad supports Windows 8.1 and 10. See System Requirements for more details.

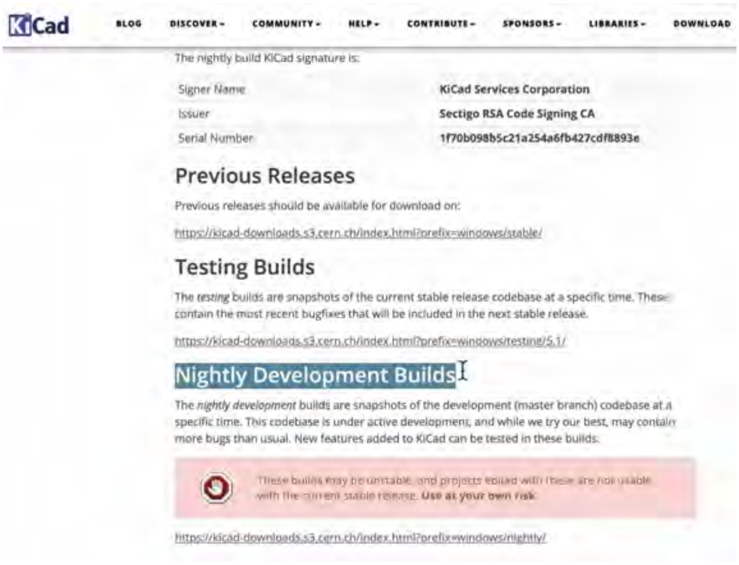
Stable Release

Current Version: **5.1.10**

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.

KiCad’s nightly development builds



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

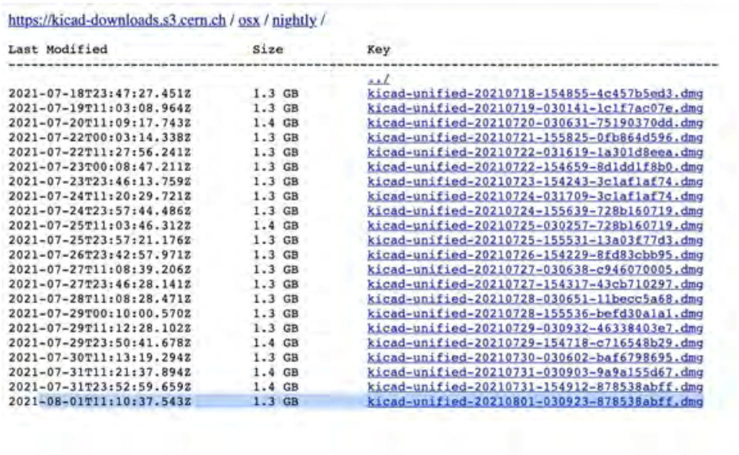
The *nightly development* builds are snapshots of the development (master branch) codebase at a specific time. This codebase is under active development, and while we try our best, may contain more bugs than usual. New features added to KiCad can be tested in these builds.

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

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.

KiCad's source code

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

Ready to learn KiCad?

Learn the world's favourite open-source PCB design tool with the world's most comprehensive course

KiCad Like a Pro, 3rd edition is available as a video course or as an eBook.

Choose the version that fits best with your style of learning, or get both to get the full benefit of the video demos plus the details of the eBook.

When you complete KiCad Like a Pro 3e, you’ll be able to use KiCad to design and manufacture multi-layer PCBs with highly integrated components and a professional-looking finish.

Work through five projects that give many opportunities to learn and practice all of KiCad’s important features.

KiCad Like a Pro 3e contains full sections dedicated to PCB and design principles and concepts. These ensure that you will master the fundamentals so that your PCB project are awesome.

If you are someone who is interested in designing PCBs using KiCad, or moving to KiCad from another CAD application, then KiCad Like a Pro, the video course and eBook, is for you.

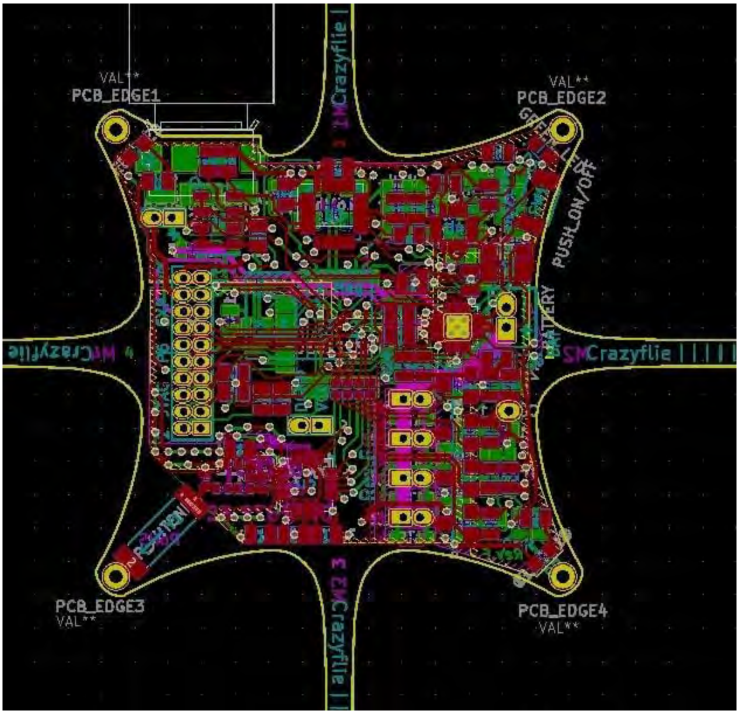
Learn more

An example KiCad project

Now that you’ve installed KiCad, you can begin familiarizing yourself with it by looking at one of the examples included with it, as described in this article.

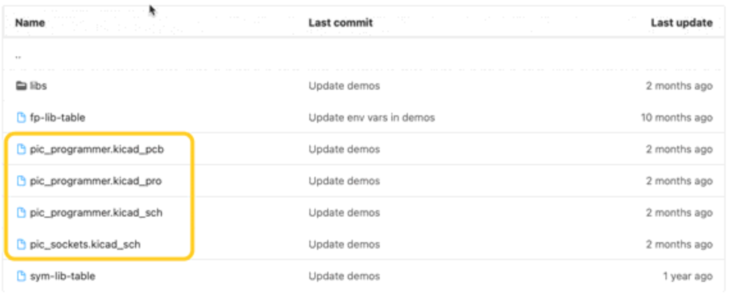
Examining the demo project will teach you about KiCad apps like the schematic editor and the layout editor.

Updated for KiCad 7.0.8.



Introduction to a KiCad demo project

Now that you have installed your instance of KiCad let's start familiarising yourself with it by looking at one example that comes 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, such as the schematic editor and the layout editor. You can see the main KiCad window in Figure 1.5.2.

KiCad’s schematic editor

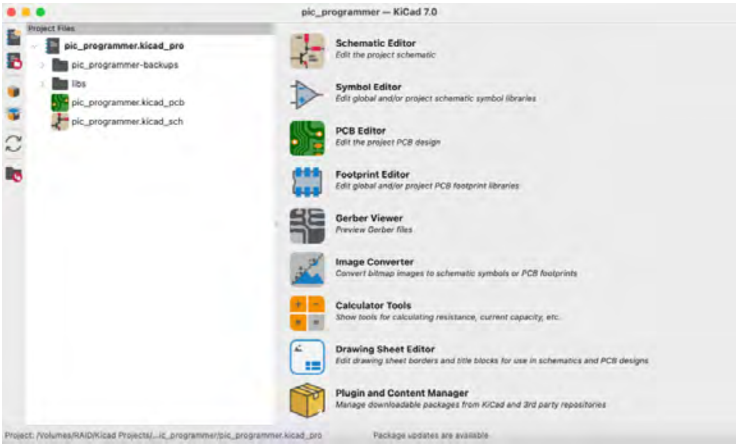

pic_programmer — KiCad 7.0
Project Files
pic_programmer.kicad_pro
pic_programmer-backups
libs
pic_programmer.kicad_pcb
pic_programmer.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs
Plugin and Content Manager
Manage downloadable packages from KiCad and 3rd party repositories
Project: /Volumes/RAID/Kicad Projects/...ic_programmer/pic_programmer.kicad_pro
Package updates are available

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 **schematic editor** application, the schematic layout editor. You should see the editor as in the example in Figure 1.5.3.

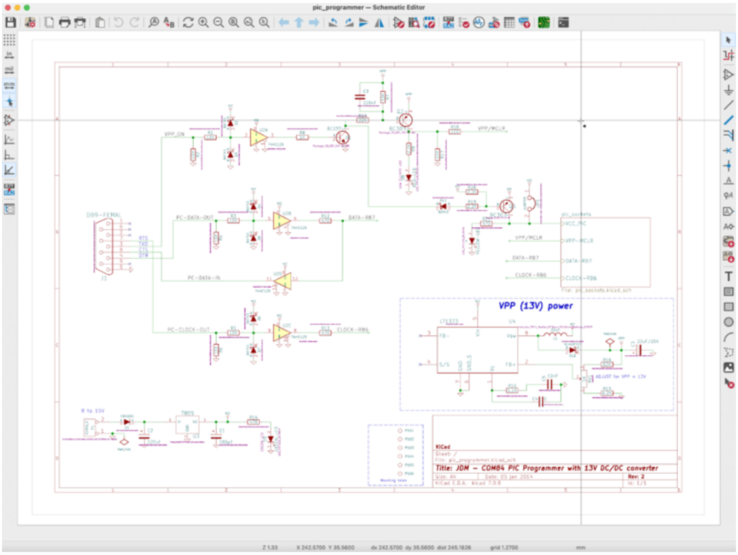

pic_programmer — Schematic Editor
VPP (13V) power
Title: JDM — COM84 PIC Programmer with 13V DC/DC converter

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 and 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.

A rectangular symbol is on the schematic's right side with the title 'pic_sockets' (Figure 1.5.4).

Double-click on it. What happened?

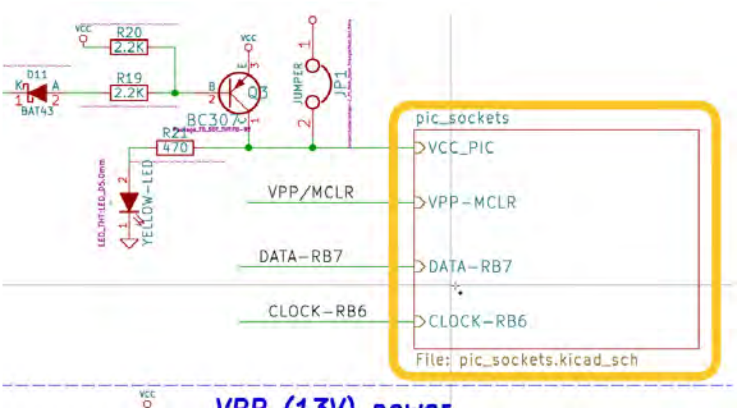


Figure 1.5.4: A link to another sheet.

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

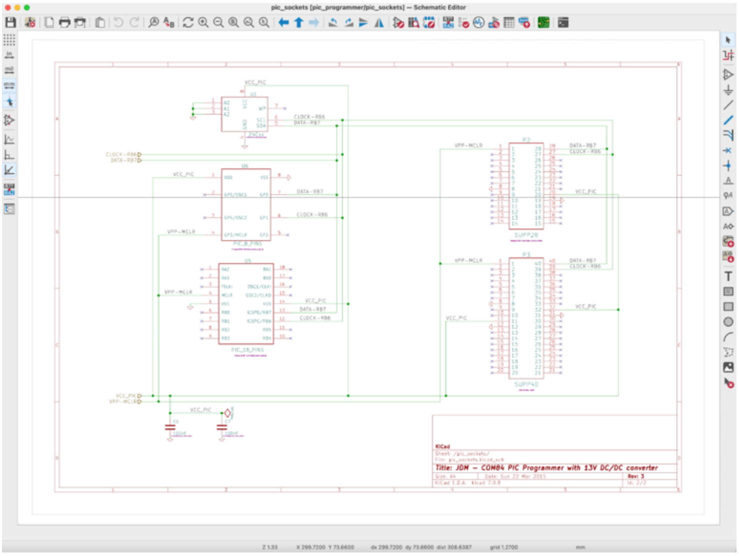

pic_sockets [pic_programmer/pic_sockets] — Schematic Editor
Title: JDM - COM84 PIC Programmer with 13V DC/DC converter
SUPP28
SUPP40
PIC_8_PINS
PIC_18_PINS

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 the **KiCad Like a Pro 3e** course or eBook).

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.

KiCad's layout editor

Go back to the main KiCad window. Click on the **button labeled "PCB Editor."** This will launch the layout editor. The window that appears will look like the example in Figure 1.5.6.

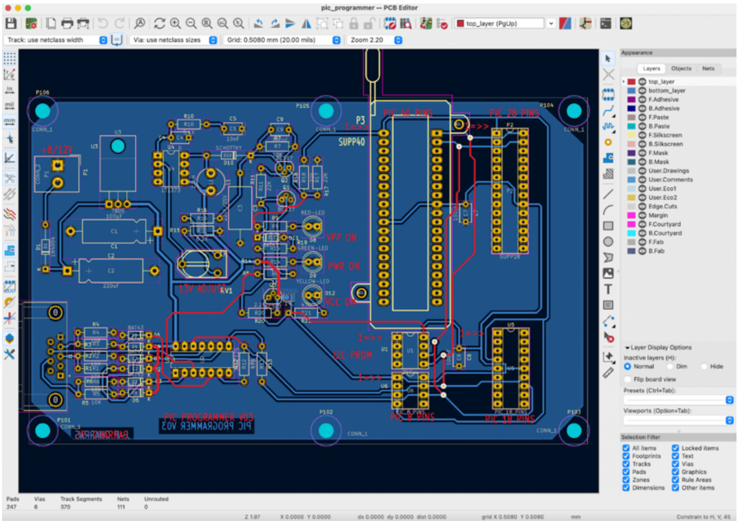


Figure 1.1.5.6: The layout editor.

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 colours 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.

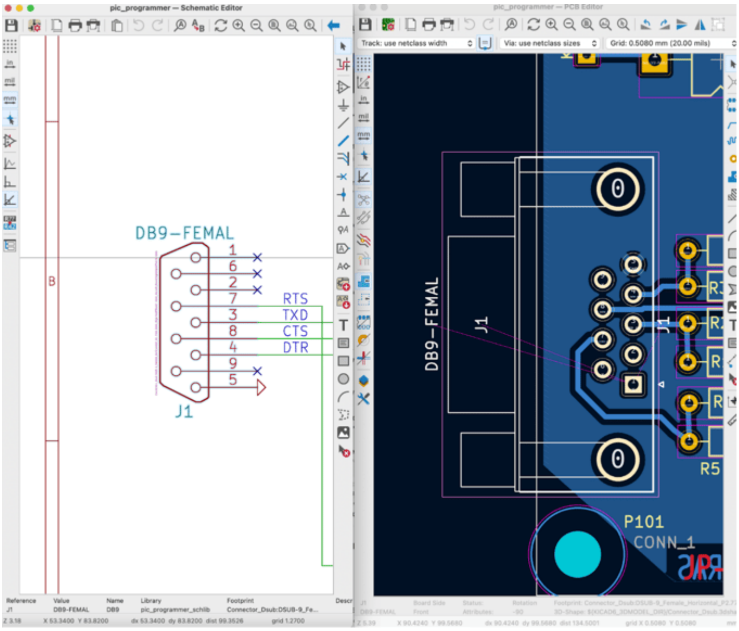
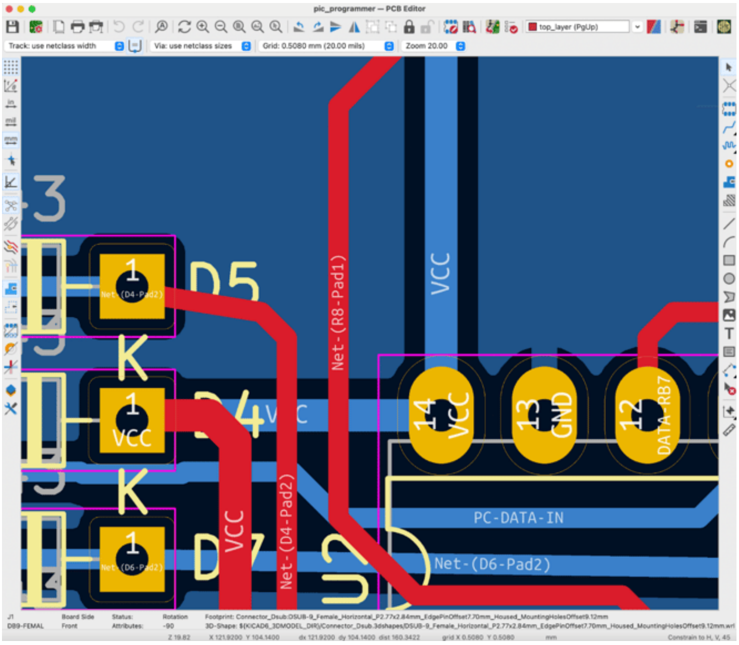

pic_programmer — Schematic Editor
pic_programmer — PCB Editor
DB9–FEMAL
J1
RTS
TXD
CTS
DTR
B
Track: use netclass width
Via: use netclass sizes
Grid: 0.5080 mm (20.00 mils)
P101
CONN_1
R5

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 this in the **KiCad Like a Pro 3e** eBook and course. In the layout, zoom in on the J1 connector to see one of its details: the trace name 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 the layout editor are defined (manually or automatically) in the schematic editor.



Track: use netclass width
Via: use netclass sizes
Grid: 0.5080 mm (20.00 mils)
Zoom 20.00

J1 DB9-FEMAL | Board Side Front | Status: Attributes: | Rotation -90 | Footprint: Connector_Dsub:DSUB-9_Female_Horizontal_P2.77x2.84mm_EdgePinOffset7.70mm_Housed_MountingHolesOffset9.12mm
3D-Shape: ${KICAD6_3DMODEL_DIR}/Connector_Dsub.3dshapes/DSUB-9_Female_Horizontal_P2.77x2.84mm_EdgePinOffset7.70mm_Housed_MountingHolesOffset9.12mm.wrl
Z 19.82 X 121.9200 Y 104.1400 dx 121.9200 dy 104.1400 dist 160.3422 grid X 0.5080 Y 0.5080 mm Constrain to H, V, 45

Figure 1.5.8: Traces have names.

Try another thing: In the layout editor, 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.

You can still see their pads and outlines on the board for the rest.

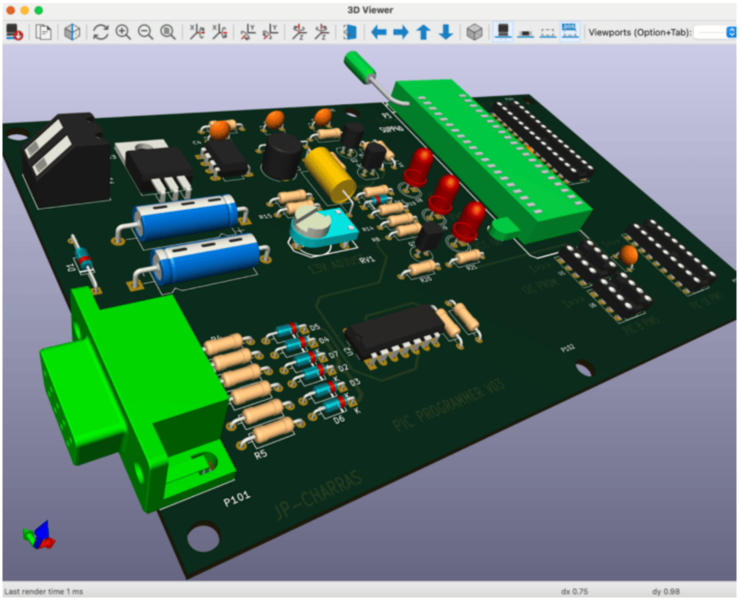


Figure 1.5.9: The 3D viewer will give you a realistic rendering of your board that you can examine in 3D.

As with the schematic editor, I encourage you to spend some time studying this demo project’s layout. Later in the course or eBook, you will learn about the most important layout guidelines to help you design well-functioning and elegant PCBs.

Showcased KiCad projects

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 the **KiCad Like a Pro 3e** course or eBook. Go to txplo.re/madewkicad for more examples of projects made with KiCad.

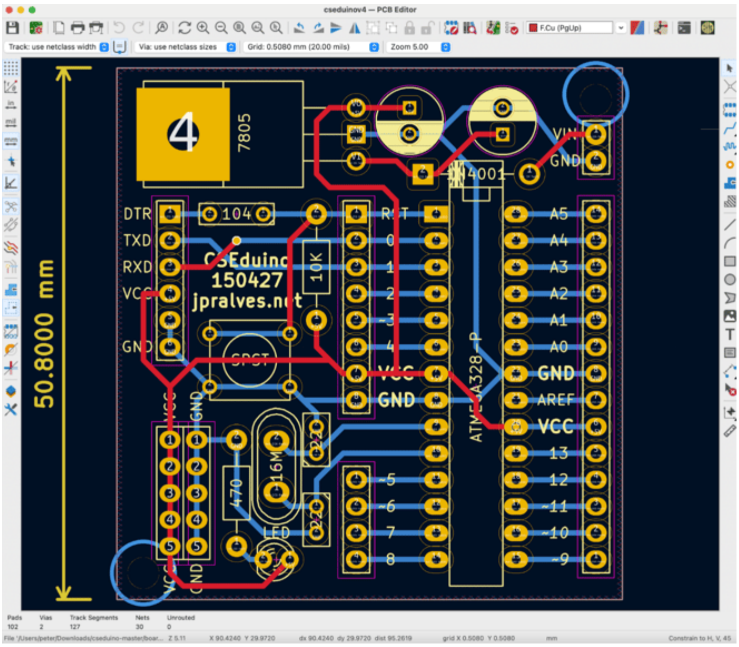


Figure 1.5.10: Featured board ‘Made with KiCad’: CSEduino.

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.

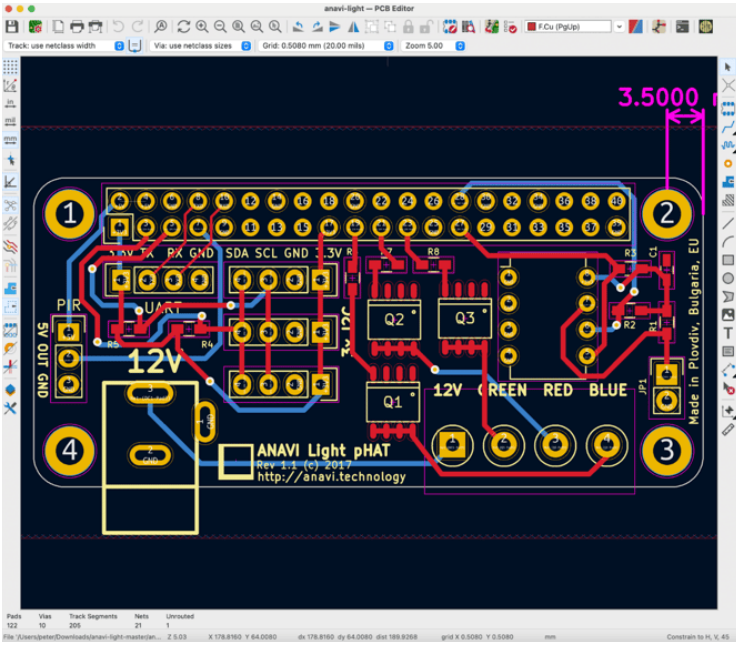


Figure 1.5.11: Featured board ‘Made with KiCad’: Anavi Light.

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. This course and eBook will also teach you how to create complicated PCBs with complicated shapes.

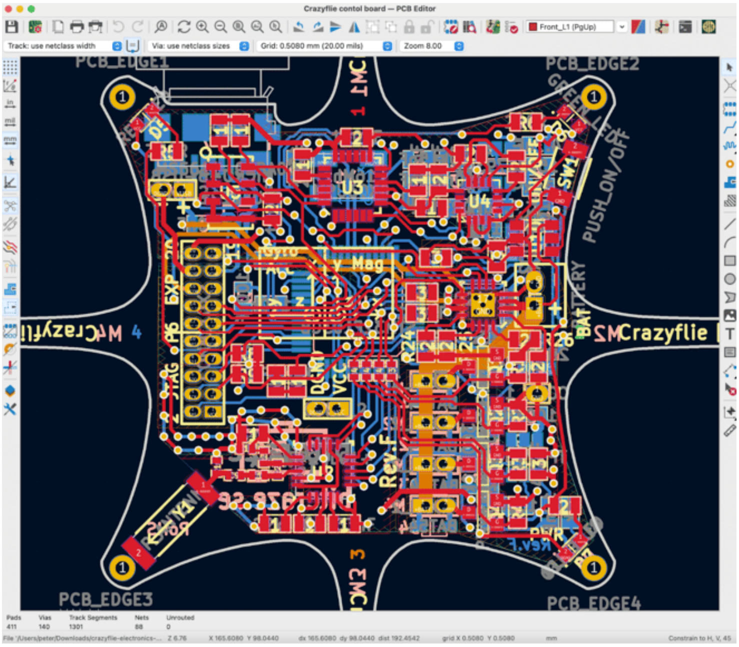


Figure 1.5.12: Featured board 'Made with KiCad': Crazyflie.

With these examples, 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 the **KiCad Like a Pro 3e** course or eBook. Let's get straight into the **first project** so that you can start discovering this fantastic tool by doing.

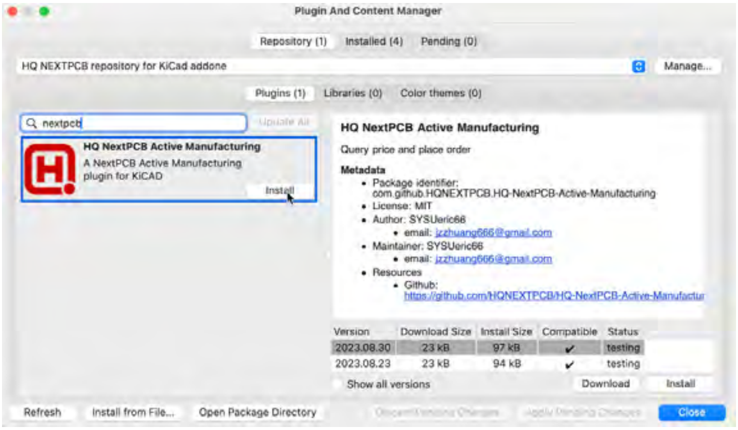
NextPCB's KiCad plugin

One of the best features of KiCad is the Plugin and Content Manager (PCM). The PCM in KiCad is a feature that allows users to discover, install, and manage plugins and libraries directly within the KiCad software. It serves as a centralized repository for extending the functionality of KiCad through add-ons and content packages.

One of the most useful new plugins is the “HQ NextPCB Active Manufacturing”. You can find details on the plugin repository, and you can install it easily with the help of the PCM. Remember to add this plugin repository URL to the PCM repositories list in Tools Plugin and Content Manager Manage:

https://raw.githubusercontent.com/HQNEXTPCB/HQNEXTPCB-kicad-addone-repository/main/repository.json

Then, install the plugin by searching for “nextpcb” (“1”, in the screenshot below) and then click “Install” (“2” in the screenshot below):



HQ NextPCB Active Manufacturing
A NextPCB Active Manufacturing plugin for KiCAD

HQ NextPCB Active Manufacturing

Query price and place order

Metadata
- Package identifier: com.github.HQNEXTPCB.HQ-NextPCB-Active-Manufacturing
- License: MIT
- Author: SYSUeric66
  - email: jzzhuang666@gmail.com
- Maintainer: SYSUeric66
  - email: jzzhuang666@gmail.com
- Resources
  - Github: https://github.com/HQNEXTPCB/HQ-NextPCB-Active-Manufactur

Version	Download Size	Install Size	Compatible	Status
2023.08.30	23 kB	97 kB	✔	testing
2023.08.23	23 kB	94 kB	✔	testing

Figure 2: Installing the NextPCB plugin to KiCad.

The HQ NextPCB Active Manufacturing plugin integrates directly with KiCad, enabling you to access PCB manufacturing services from NextPCB within the KiCad environment. This plugin allows for an efficient transition from PCB design to production by providing tools to check PCB design rules, quote PCB fabrication costs, and place orders directly through NextPCB’s manufacturing services.

The plugin’s key features are:

- **Design Rule Check (DRC):** Users can

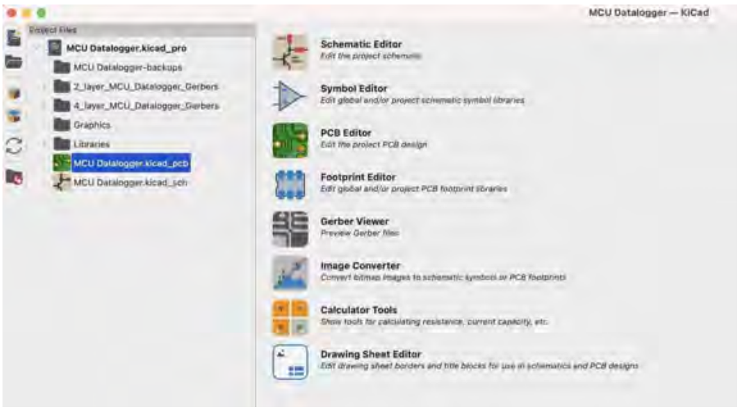
verify their PCB designs against NextPCB's manufacturing capabilities to ensure compliance with the manufacturer's specifications.
- **Quotation and Ordering:** The plugin provides real-time pricing for various PCB attributes such as size, layers, material, and quantity. Users can get quotes and place orders without leaving the KiCad interface.
- **Seamless Integration:** Designed to work within the KiCad PCB design software, streamlining the workflow from design to manufacturing.

I used the plugin with one of my KiCad Like a Pro 3e projects, and I was able to get pricing and manufacturing options without ever leaving the KiCad environment. I didn't have to export Gerbers, upload to the NextPCB website, and repeat the process each time I wanted to make a change to the PCB. This plugin is a real time saver.

Here is how easy it is to get a quote and place an order for a PCB all within the KiCad environment using the NextPCB plugin:

KiCad Project Manager (main window)

This article provides an overview of the KiCad project manager, also known as the "main" KiCad window, which provides access to all KiCad applications.


MCU Datalogger — KiCad
Project Files
MCU Datalogger.kicad_pro
MCU Datalogger-backups
2_layer_MCU_Datalogger_Gerbers
4_layer_MCU_Datalogger_Gerbers
Graphics
Libraries
MCU Datalogger.kicad_pcb
MCU Datalogger.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs

Getting started with KiCad 6: Introduction

In this and the following articles, I will give you a brief overview of **KiCad 6**. This overview will help you with the first hands-on activity of the **KiCad like a Pro 3e** course or eBook, in which you will create your first PCB.

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 article "4. Get KiCad for your operating system," where

I provide information on installing KiCad on Mac OS, Windows, and Linux.

In the following articles, 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 article on the major differences between the two versions.

KiCad Project Manager (main window)

Below, I will give you an overview of the **KiCad project manager**, otherwise known as the “main” KiCad window.

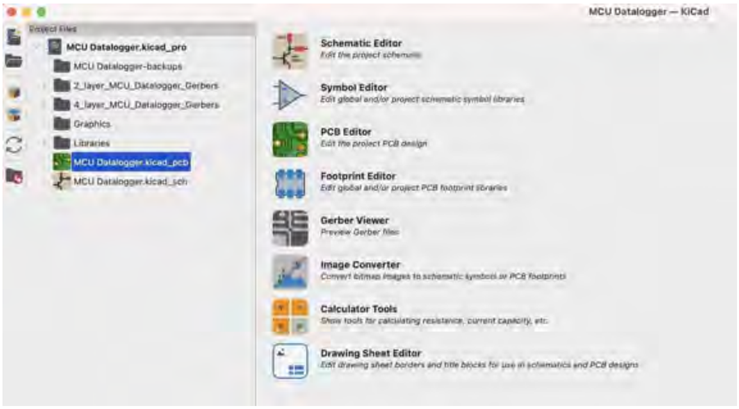

MCU Datalogger — KiCad
Project Files
MCU Datalogger.kicad_pro
MCU Datalogger-backups
2_layer_MCU_Datalogger_Gerbers
4_layer_MCU_Datalogger_Gerbers
Graphics
Libraries
MCU Datalogger.kicad_pcb
MCU Datalogger.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs

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 image below).

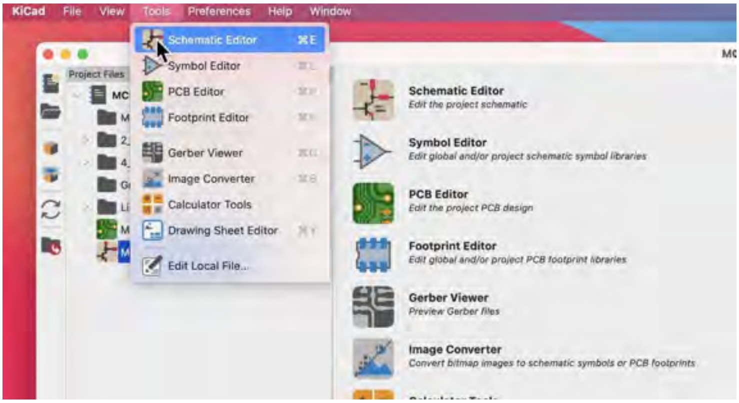


Figure 2.2.2: Starting the Schematic editor.

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 guide.

You will learn about the buttons in the right pane in the next article.

First, let's do a tour of the items in the top menu bar. The top menu bar appears at the top of the screen on Mac OS, and the top of the KiCad window in Microsoft Windows.

Below you can see the KiCad main app in Mac OS with its menu bar in the top of the screen. I have opened the KiCad menu to reveal the "**About KiCad"** option.

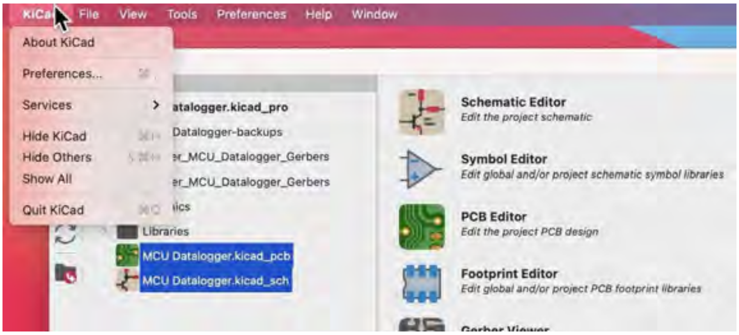


Figure 2.2.3: The top menu bar in KiCad (Mac OS).

Below you can see the KiCad main app in Microsoft Windows with its menu bar in the top of the KiCad window. I have opened the Help menu to reveal the “About KiCad” option.

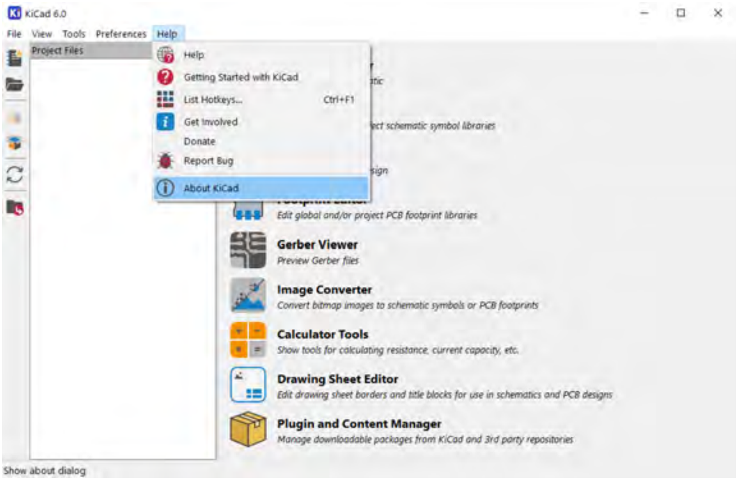

KiCad 6.0
File
View
Tools
Preferences
Help
Project Files
Help
Getting Started with KiCad
List Hotkeys... Ctrl+F1
Get Involved
Donate
Report Bug
About KiCad
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs
Plugin and Content Manager
Manage downloadable packages from KiCad and 3rd party repositories
Show about dialog

Figure 2.2.4: The top menu bar in KiCad (Windows).

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.

Below you can see the “About KiCad” window in Mac OS, next to the New Issue page in GitLab.

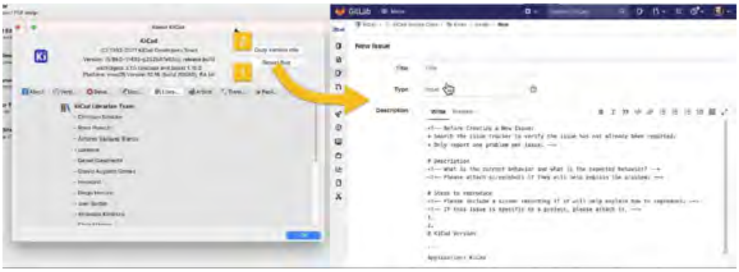
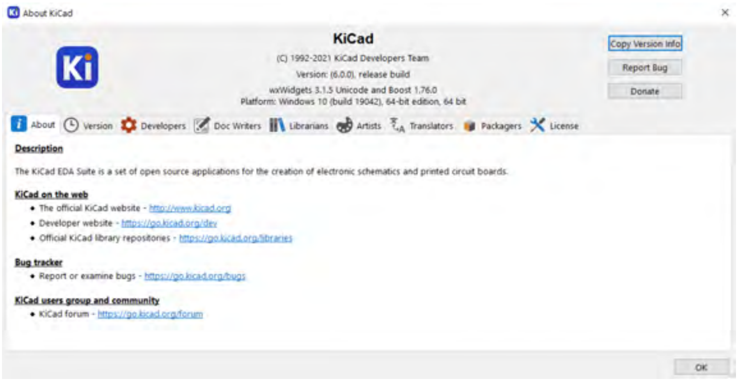


Figure 2.2.5: Report a bug.

Below you can see the “About KiCad” window in Microsoft Windows. The Linux version looks very similar.



KiCad

(C) 1992-2021 KiCad Developers Team

Version: (6.0.0), release build

wxWidgets 3.1.5 Unicode and Boost 1.76.0

Platform: Windows 10 (build 19042), 64-bit edition, 64 bit

Description

The KiCad EDA Suite is a set of open source applications for the creation of electronic schematics and printed circuit boards.

KiCad on the web

- The official KiCad website - http://www.kicad.org
- Developer website - https://go.kicad.org/dev
- Official KiCad library repositories - https://go.kicad.org/libraries

Bug tracker

- Report or examine bugs - https://go.kicad.org/bugs

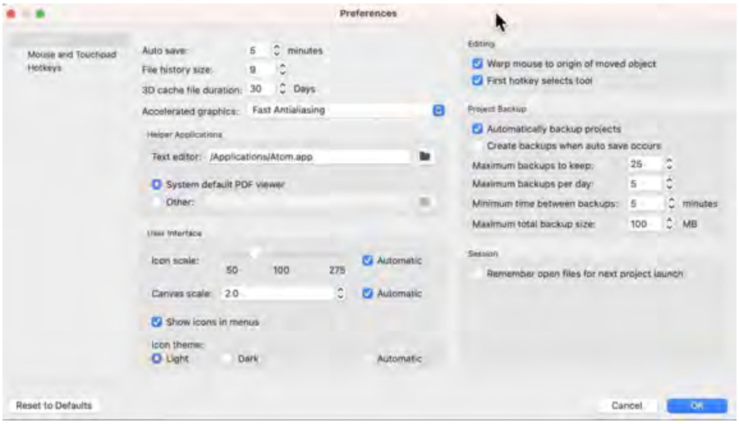
KiCad users group and community

- KiCad forum - https://go.kicad.org/forum

Figure 2.2.6: The KiCad “About” window in Microsoft Windows.

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.


Preferences

Mouse and Touchpad
Hotkeys

Auto save: 5 minutes
File history size: 9
3D cache file duration: 30 Days
Accelerated graphics: Fast Antialiasing

Helper Applications
Text editor: /Applications/Atom.app
[✓ System default PDF viewer]
Other:

User Interface
Icon scale: 50 100 275 [✓ Automatic]
Canvas scale: 2.0 [✓ Automatic]
[✓ Show icons in menus]
Icon theme: [✓ Light] Dark Automatic

Editing
[✓ Warp mouse to origin of moved object]
[✓ First hotkey selects tool]

Project Backup
[✓ Automatically backup projects]
Create backups when auto save occurs
Maximum backups to keep: 25
Maximum backups per day: 5
Minimum time between backups: 5 minutes
Maximum total backup size: 100 MB

Session
Remember open files for next project launch

Reset to Defaults
Cancel
OK

Figure 2.2.7: 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 articles later in this guide (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.

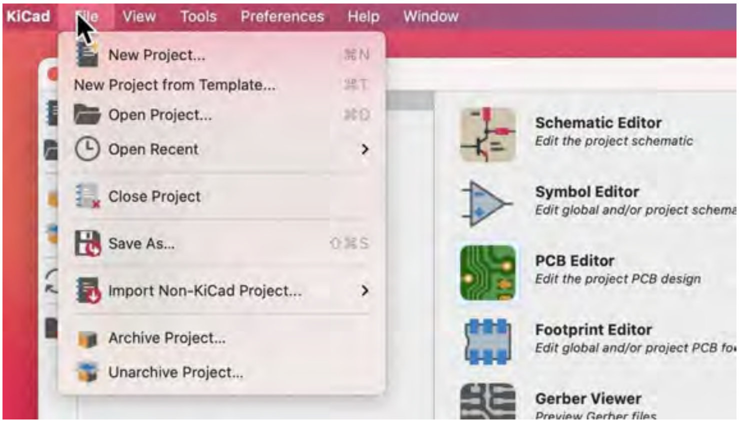
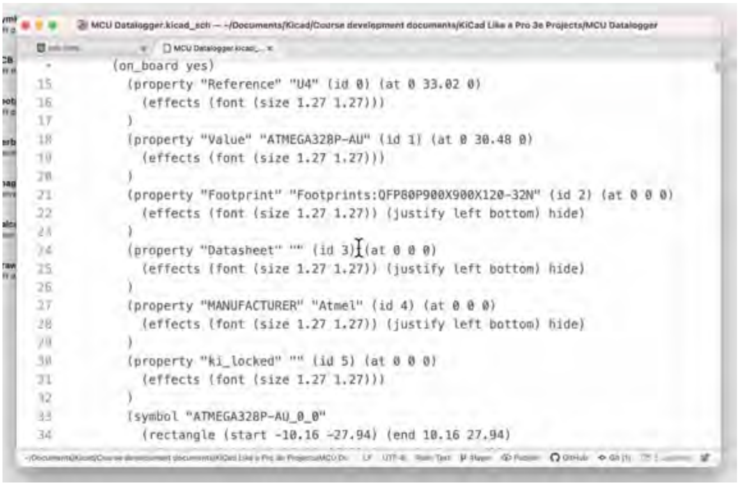


Figure 2.2.8: The KiCad File menu.

You will be using those options in the projects through the **KiCad like a Pro 3e** course and eBook. In the Recipes part of the **KiCad like a Pro 3e** course and eBook, 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.


MCU Datalogger.kicad_sch — ~/Documents/Kicad/Course development documents/KiCad Like a Pro 3e Projects/MCU Datalogger

```
        (on_board yes)
          (property "Reference" "U4" (id 0) (at 0 33.02 0)
            (effects (font (size 1.27 1.27)))
          )
          (property "Value" "ATMEGA328P-AU" (id 1) (at 0 30.48 0)
            (effects (font (size 1.27 1.27)))
          )
          (property "Footprint" "Footprints:QFP80P900X900X120-32N" (id 2) (at 0 0 0)
            (effects (font (size 1.27 1.27)) (justify left bottom) hide)
          )
          (property "Datasheet" "" (id 3) (at 0 0 0)
            (effects (font (size 1.27 1.27)) (justify left bottom) hide)
          )
          (property "MANUFACTURER" "Atmel" (id 4) (at 0 0 0)
            (effects (font (size 1.27 1.27)) (justify left bottom) hide)
          )
          (property "ki_locked" "" (id 5) (at 0 0 0)
            (effects (font (size 1.27 1.27)))
          )
          (symbol "ATMEGA328P-AU_0_0"
            (rectangle (start -10.16 -27.94) (end 10.16 27.94)
```

Figure 2.2.9: 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.

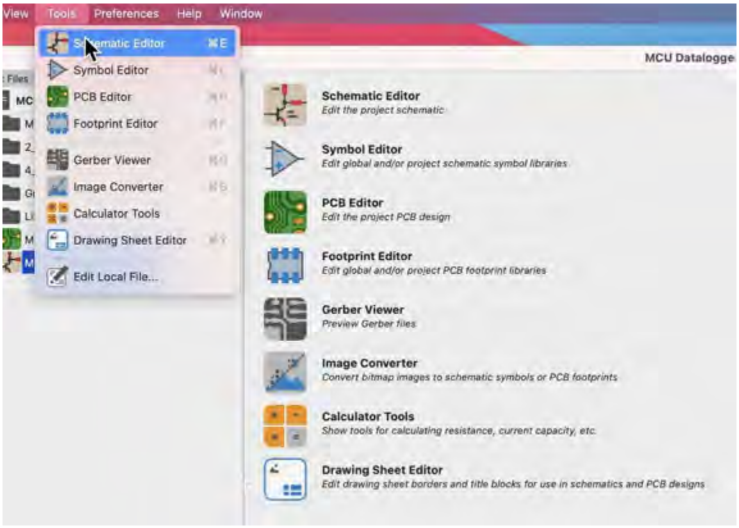

View
Tools
Preferences
Help
Window
Schematic Editor ⌘E
Symbol Editor
PCB Editor
Footprint Editor
Gerber Viewer
Image Converter
Calculator Tools
Drawing Sheet Editor
Edit Local File...
MCU Datalogge
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs

Figure 2.2.10: The Tools menu items.

I will describe these applications in the next article.

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

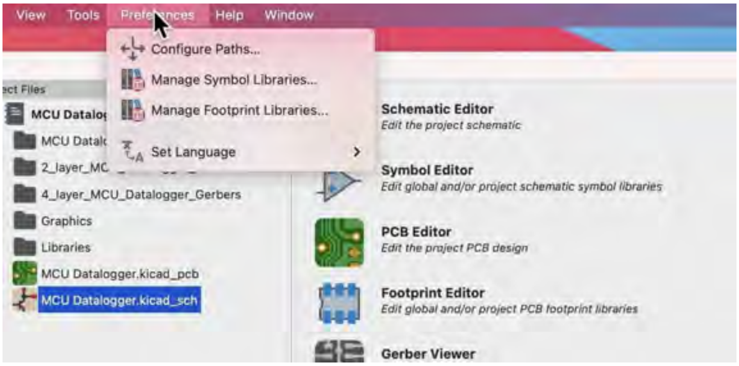

View
Tools
Preferences
Help
Window
Configure Paths...
Manage Symbol Libraries...
Manage Footprint Libraries...
Set Language
MCU Datalog
MCU Datal
2_layer_MC
4_layer_MCU_Datalogger_Gerbers
Graphics
Libraries
MCU Datalogger.kicad_pcb
MCU Datalogger.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer

Figure 2.2.11: The Preferences menu.

I have written a dedicated article on these manager windows with details later in this guide series.

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.

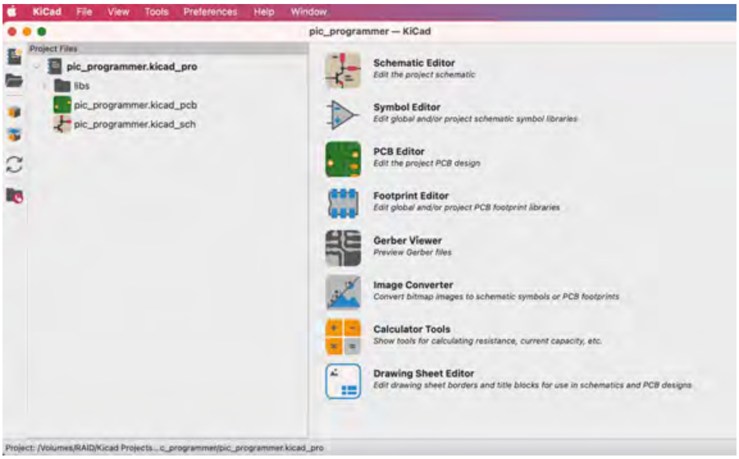
The Hotkeys window, apart from listing current hotkeys, allows you to make changes. I prefer to keep the default hotkeys unless there is a conflict with other applications running on my computer.

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

Overview of the individual KiCad apps

This article will take you on a tour of the individual applications that make up the KiCad software suite.

These are the Schematic, Symbol, Layout (PCB), Footprint, and Drawing Sheet Editors, as well as the Gerber Viewer, the Image Converter, and the Calculator Tools.


KiCad
File
View
Tools
Preferences
Help
Window
pic_programmer — KiCad
Project Files
pic_programmer.kicad_pro
libs
pic_programmer.kicad_pcb
pic_programmer.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Edit the project PCB design
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB footprints
Calculator Tools
Show tools for calculating resistance, current capacity, etc.
Drawing Sheet Editor
Edit drawing sheet borders and title blocks for use in schematics and PCB designs
Project: /Volumes/RAID/Kicad Projects...c_programmer/pic_programmer.kicad_pro

Introduction

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

As you may recall from the previous article, 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.

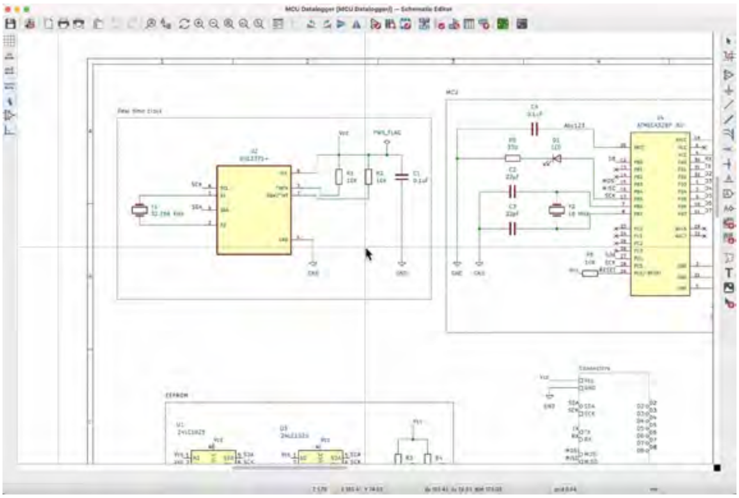


Figure 2.3.1: Eeschema, or the Schematic Editor.

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 the **KiCad Like a Pro 3e** course and eBook. 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 the course and eBook.

Layout (PCB) 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.

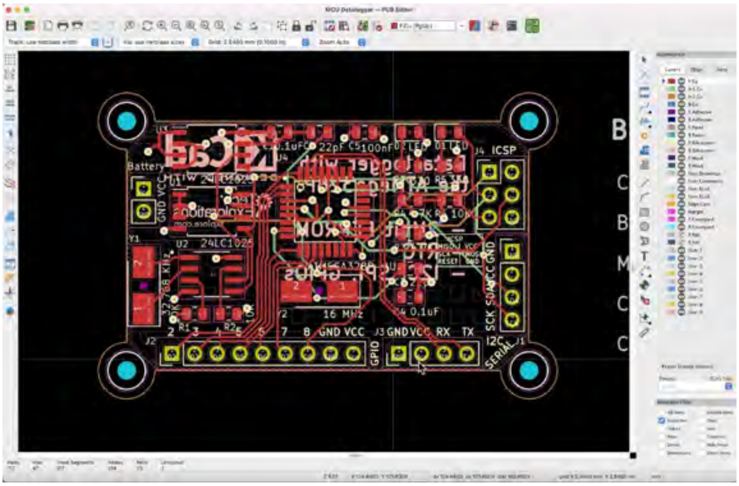


Figure 2.3.2: Pcbnew, or the Layout Editor.

In the example above, you can see the finished PCB design from one of the projects in this guide. 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:

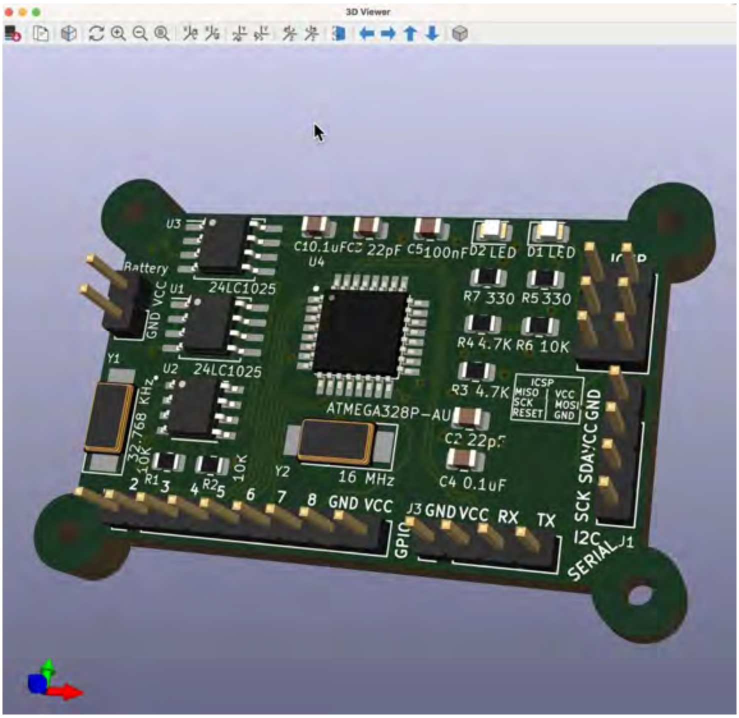


Figure 2.3.3: The 3D viewer in Pcbnew.

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

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.

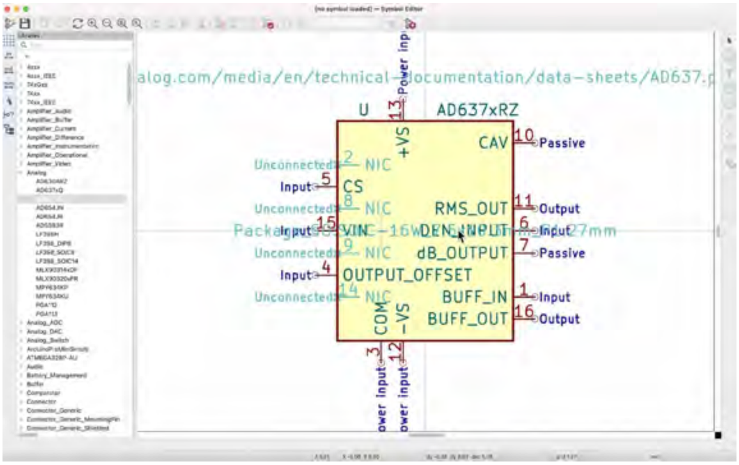


Figure 2.3.4: The Symbol Editor.

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 with the **KiCad like a Pro 3e** course or eBook.

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.

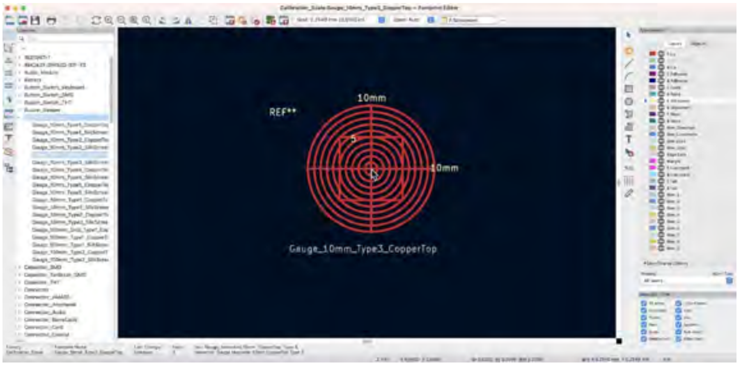


Figure 2.3.5: The Footprint Editor.

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 part of the **KiCad like a Pro 3e** course or eBook.

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**.

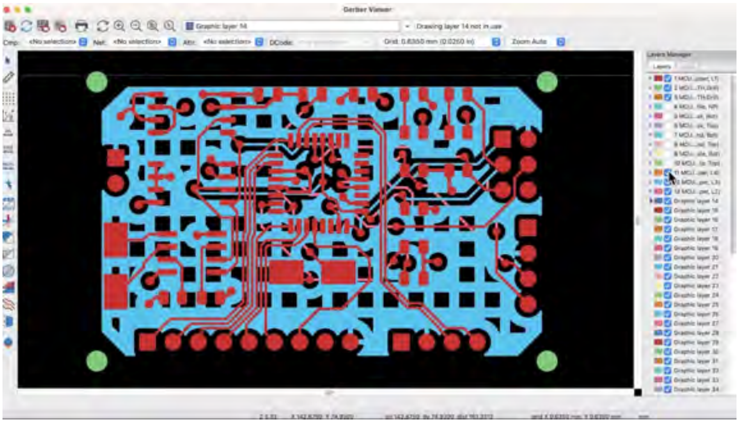


Figure 2.3.6: The Gerber Viewer.

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 parts of the **KiCad like a Pro 3e** course and eBook.

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.

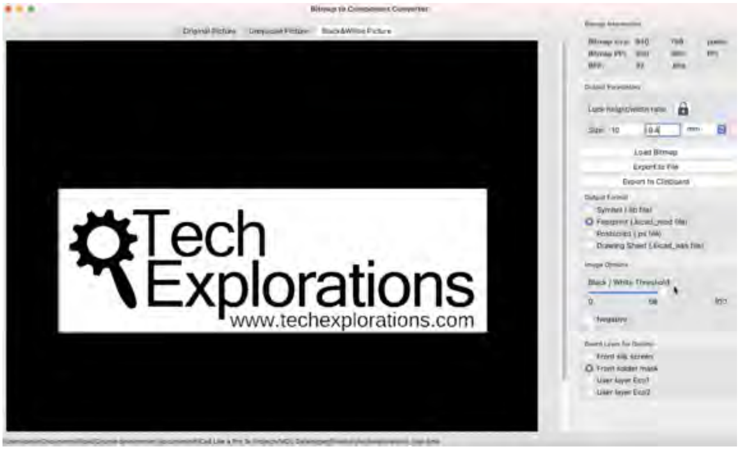


Figure 2.3.7: The Image Converter.

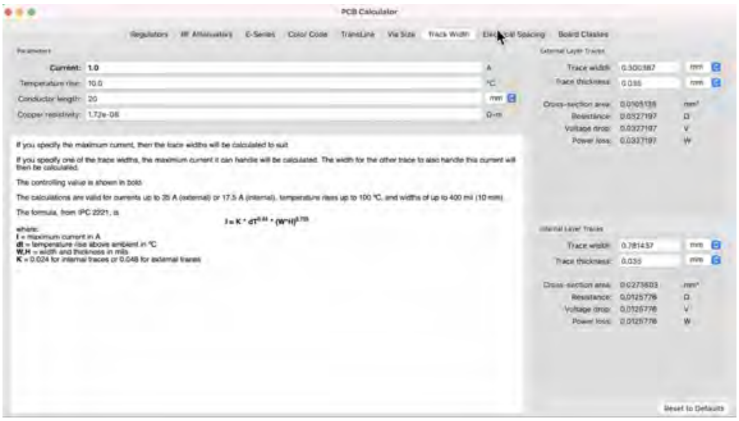
In the example above, I use the Image Converter to create a logo that I can include in my PCBs. You can learn how to use the Image Converter in a dedicated part of the **KiCad like a Pro 3e** course and eBook.

Calculator Tools

The Calculator Tools contain 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.



Current: **1.0** A
Temperature rise: 10.0 °C
Conductor length: 20 mm
Copper resistivity: 1.72e-08 Ω-m

If you specify the maximum current, then the trace widths will be calculated to suit.

If you specify one of the trace widths, the maximum current it can handle will be calculated. The width for the other trace to also handle this current will then be calculated.

The controlling value is shown in bold.

The calculations are valid for currents up to 35 A (external) or 17.5 A (internal), temperature rises up to 100 °C, and widths of up to 400 mil (10 mm).

The formula, from IPC 2221, is

$$I = K * dT^{0.44} * (W*H)^{0.725}$$

where:
I = maximum current in A
dT = temperature rise above ambient in °C
W,H = width and thickness in mils
K = 0.024 for internal traces or 0.048 for external traces

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
Power loss:	0.0327197	W

Internal Layer Traces

Trace width:	0.781437	mm
Trace thickness:	0.035	mm
Cross-section area:	0.0273503	mm²
Resistance:	0.0125776	Ω
Voltage drop:	0.0125776	V
Power loss:	0.0125776	W

Figure 2.3.8: The Calculator Tools.

You can learn how to use the Track Width calculator by going through the relevant lectures or chapters in the Recipes part of the **KiCad like a Pro 3e** course or eBook. 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.

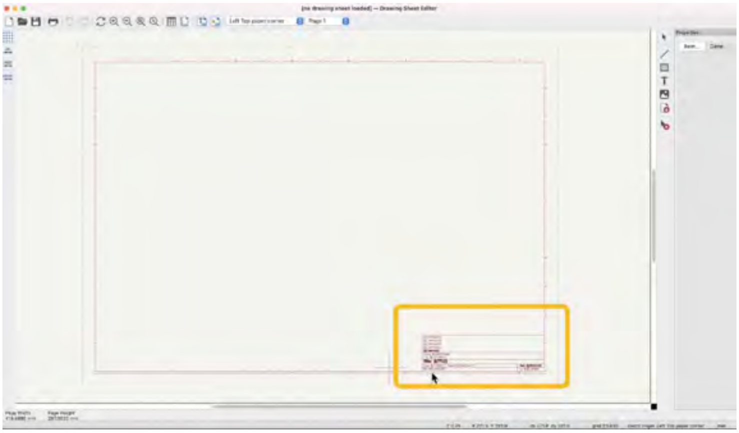


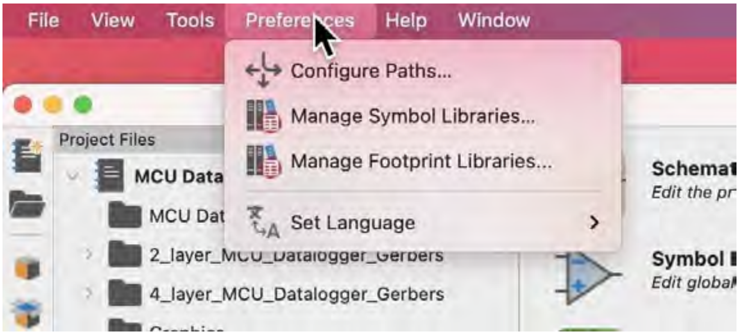
Figure 2.3.9: The Drawing Sheet editor.

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 go through the relevant material of the **KiCad like a Pro 3e** course or eBook.

KiCad Paths and Libraries

In this article, you'll learn about the paths and libraries configuration options in the KiCad project window's Preferences menu.



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

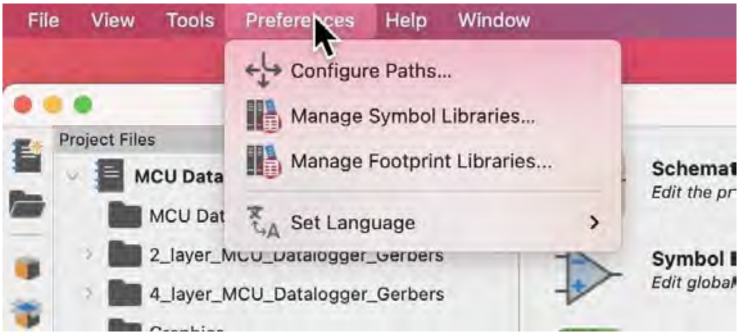


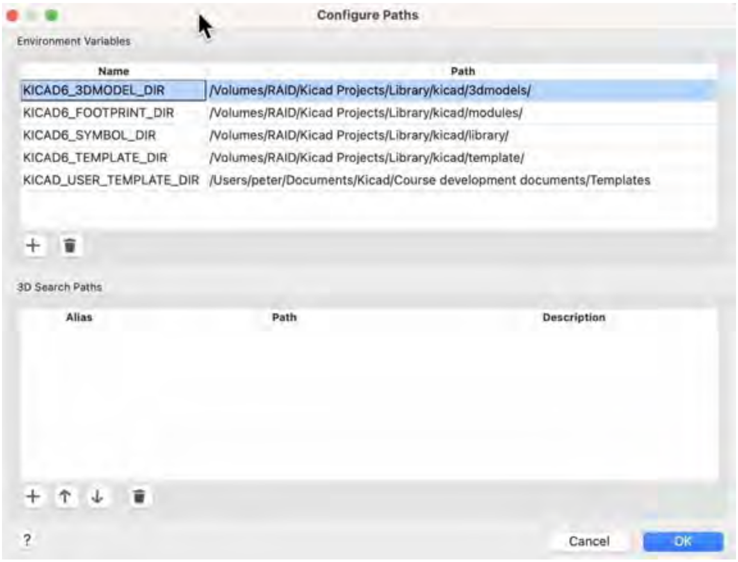
Figure 2.4.1: The Preferences menu.

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.


Configure Paths

Environment Variables

Name	Path
KICAD6_3DMODEL_DIR	/Volumes/RAID/Kicad Projects/Library/kicad/3dmodels/
KICAD6_FOOTPRINT_DIR	/Volumes/RAID/Kicad Projects/Library/kicad/modules/
KICAD6_SYMBOL_DIR	/Volumes/RAID/Kicad Projects/Library/kicad/library/
KICAD6_TEMPLATE_DIR	/Volumes/RAID/Kicad Projects/Library/kicad/template/
KICAD_USER_TEMPLATE_DIR	/Users/peter/Documents/Kicad/Course development documents/Templates

3D Search Paths

Alias	Path	Description

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 section of the course or eBook.
- KICAD6_3RD_PARTY: points to a directory that contains 3rd 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 section of the course or eBook.
- KICAD6_SYMBOL_DIR: points to a directory

that contains symbol files for use by Eeschema. Learn more about this in a dedicated section of the course or eBook.
- KICAD6_TEMPLATE_DIR: points to a directory that contains sheet template files for use by Eeschema. Learn more about this in a dedicated section of the course or eBook.
- 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 section of the course or eBook.

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.

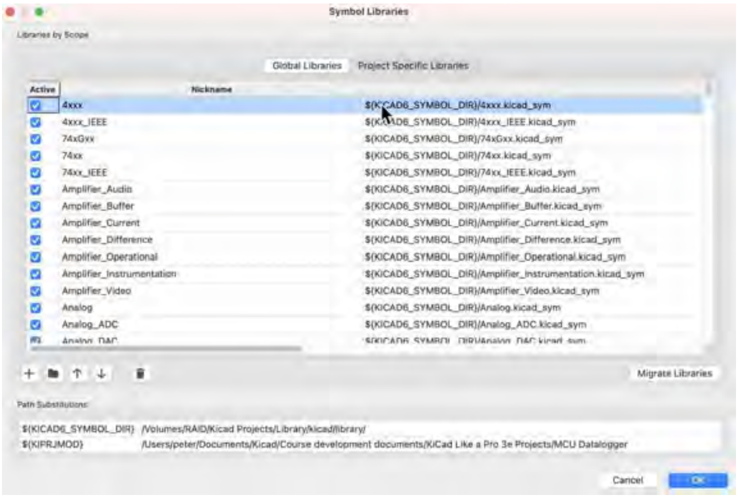
For example, my computer has a solid-state drive with a limited amount of available space on it. Because the libraries (especially the 3D models) take several gigabytes of storage, I have opted to use my external RAID drive for those resources. As you can see in Figure 2.4.2 above, the footprint, symbol,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



Active	Nickname	
✓	4xxx	${KICAD6_SYMBOL_DIR}/4xxx.kicad_sym
✓	4xxx_IEEE	${KICAD6_SYMBOL_DIR}/4xxx_IEEE.kicad_sym
✓	74xGxx	${KICAD6_SYMBOL_DIR}/74xGxx.kicad_sym
✓	74xx	${KICAD6_SYMBOL_DIR}/74xx.kicad_sym
✓	74xx_IEEE	${KICAD6_SYMBOL_DIR}/74xx_IEEE.kicad_sym
✓	Amplifier_Audio	${KICAD6_SYMBOL_DIR}/Amplifier_Audio.kicad_sym
✓	Amplifier_Buffer	${KICAD6_SYMBOL_DIR}/Amplifier_Buffer.kicad_sym
✓	Amplifier_Current	${KICAD6_SYMBOL_DIR}/Amplifier_Current.kicad_sym
✓	Amplifier_Difference	${KICAD6_SYMBOL_DIR}/Amplifier_Difference.kicad_sym
✓	Amplifier_Operational	${KICAD6_SYMBOL_DIR}/Amplifier_Operational.kicad_sym
✓	Amplifier_Instrumentation	${KICAD6_SYMBOL_DIR}/Amplifier_Instrumentation.kicad_sym
✓	Amplifier_Video	${KICAD6_SYMBOL_DIR}/Amplifier_Video.kicad_sym
✓	Analog	${KICAD6_SYMBOL_DIR}/Analog.kicad_sym
✓	Analog_ADC	${KICAD6_SYMBOL_DIR}/Analog_ADC.kicad_sym
✓	Analog_DAC	${KICAD6_SYMBOL_DIR}/Analog_DAC.kicad_sym

${KICAD6_SYMBOL_DIR}	/Volumes/RAID/Kicad Projects/Library/kicad/library/
${KIPRJMOD}	/Users/peter/Documents/Kicad/Course development documents/KiCad Like a Pro 3e Projects/MCU Datalogger

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.

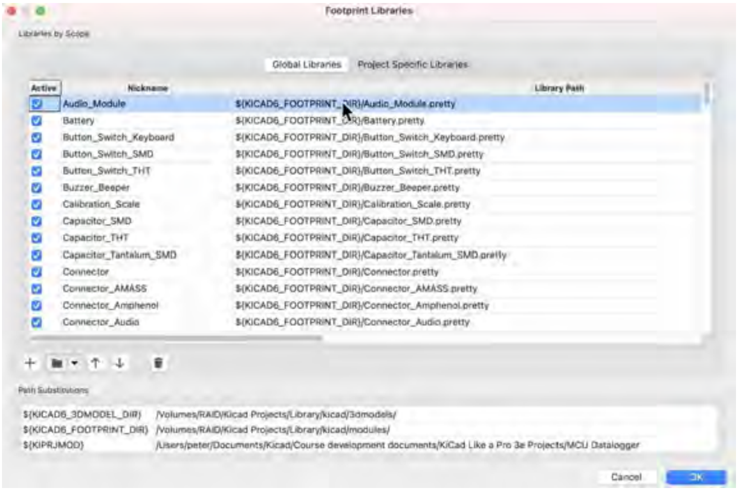
Learn how to use the symbol libraries manager in a dedicated section in the **KiCad Like a Pro 3e** course or eBook.

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.



Active	Nickname	Library Path
✓	Audio_Module	${KICAD6_FOOTPRINT_DIR}/Audio_Module.pretty
✓	Battery	${KICAD6_FOOTPRINT_DIR}/Battery.pretty
✓	Button_Switch_Keyboard	${KICAD6_FOOTPRINT_DIR}/Button_Switch_Keyboard.pretty
✓	Button_Switch_SMD	${KICAD6_FOOTPRINT_DIR}/Button_Switch_SMD.pretty
✓	Button_Switch_THT	${KICAD6_FOOTPRINT_DIR}/Button_Switch_THT.pretty
✓	Buzzer_Beeper	${KICAD6_FOOTPRINT_DIR}/Buzzer_Beeper.pretty
✓	Calibration_Scale	${KICAD6_FOOTPRINT_DIR}/Calibration_Scale.pretty
✓	Capacitor_SMD	${KICAD6_FOOTPRINT_DIR}/Capacitor_SMD.pretty
✓	Capacitor_THT	${KICAD6_FOOTPRINT_DIR}/Capacitor_THT.pretty
✓	Capacitor_Tantalum_SMD	${KICAD6_FOOTPRINT_DIR}/Capacitor_Tantalum_SMD.pretty
✓	Connector	${KICAD6_FOOTPRINT_DIR}/Connector.pretty
✓	Connector_AMASS	${KICAD6_FOOTPRINT_DIR}/Connector_AMASS.pretty
✓	Connector_Amphenol	${KICAD6_FOOTPRINT_DIR}/Connector_Amphenol.pretty
✓	Connector_Audio	${KICAD6_FOOTPRINT_DIR}/Connector_Audio.pretty

Path Substitutions

${KICAD6_3DMODEL_DIR}	/Volumes/RAID/Kicad Projects/Library/kicad/3dmodels/
${KICAD6_FOOTPRINT_DIR}	/Volumes/RAID/Kicad Projects/Library/kicad/modules/
${KIPRJMOD}	/Users/peter/Documents/Kicad/Course development documents/KiCad Like a Pro 3e Projects/MCU Datalogger

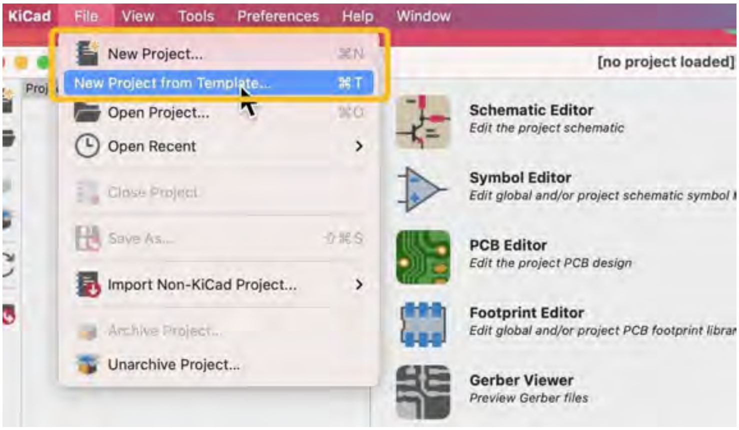
Figure 2.4.4: The “Footprint Libraries” window.

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 section in the **KiCad Like a Pro 3e** course or eBook.

Create A New KiCad Project From Scratch

There are two ways to create a new KiCad project. In this article, you will learn how to create a new KiCad project from scratch.



2 ways of creating 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**.

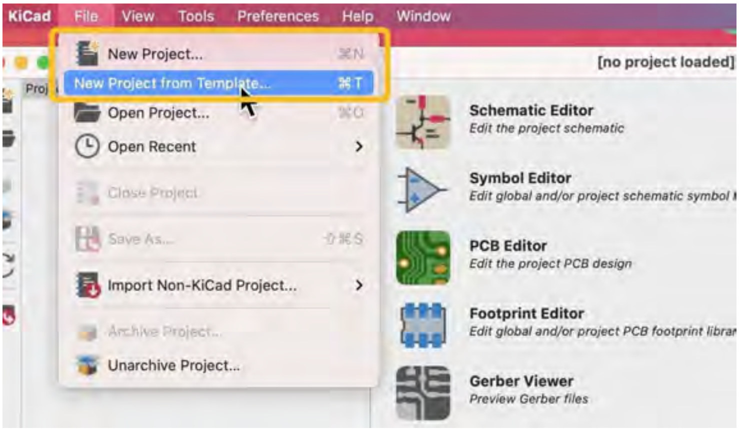
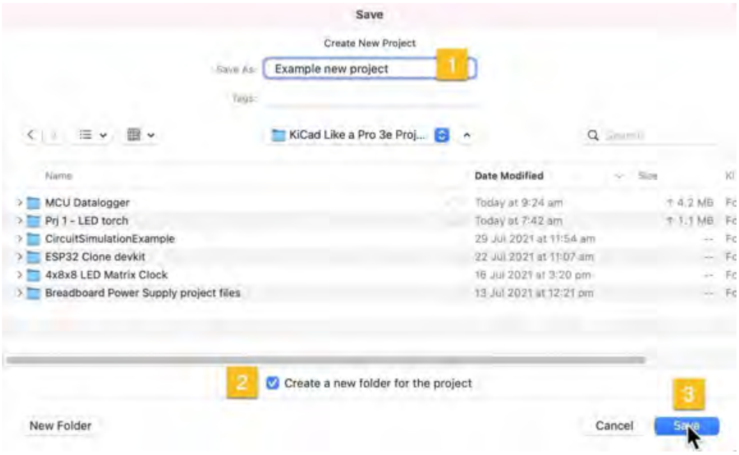


Figure 2.5.1: KiCad offers two ways to start a new project.

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 or lecture in the Recipes part of the **KiCad Like a Pro 3e** eBook or course.

Creating a new, blank KiCad project

In this article, 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").



Save As: Example new project

Tags:

KiCad Like a Pro 3e Proj...

Name	Date Modified	Size	Ki
MCU Datalogger	Today at 9:24 am	4.2 MB	Fo
Prj 1 - LED torch	Today at 7:42 am	1.1 MB	Fo
CircuitSimulationExample	29 Jul 2021 at 11:54 am	--	Fo
ESP32 Clone devkit	22 Jul 2021 at 11:07 am	--	Fo
4x8x8 LED Matrix Clock	16 Jul 2021 at 3:20 pm	--	Fo
Breadboard Power Supply project files	13 Jul 2021 at 12:21 pm	--	Fo

[✓] Create a new folder for the project

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.

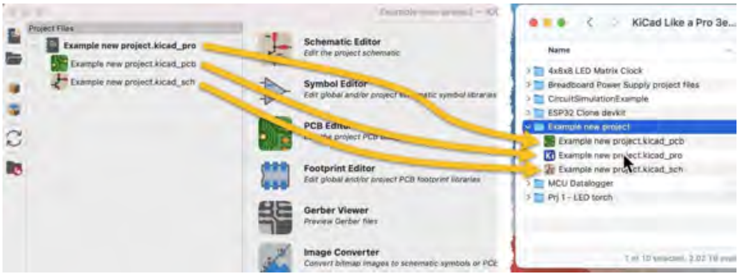
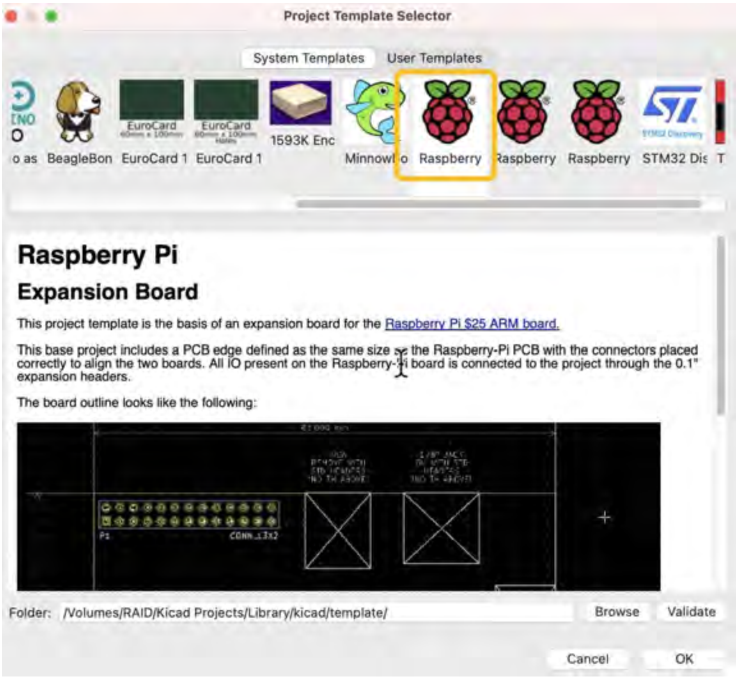

Project Files
Example new project.kicad_pro
Example new project.kicad_pcb
Example new project.kicad_sch
Schematic Editor
Edit the project schematic
Symbol Editor
Edit global and/or project schematic symbol libraries
PCB Editor
Footprint Editor
Edit global and/or project PCB footprint libraries
Gerber Viewer
Preview Gerber files
Image Converter
Convert bitmap images to schematic symbols or PCB
KiCad Like a Pro 3e...
Name
4x8x8 LED Matrix Clock
Breadboard Power Supply project files
CircuitSimulationExample
ESP32 Clone devkit
Example new project
Example new project.kicad_pcb
Example new project.kicad_pro
Example new project.kicad_sch
MCU Datalogger
Prj 1 - LED torch

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 article, you will learn how to create a new KiCad project from a **template**.

Create A New KiCad Project From A Template

There are two ways to create a new KiCad project. In this article, you will learn how to create a new KiCad project from a template.



Raspberry Pi

Expansion Board

This project template is the basis of an expansion board for the Raspberry Pi $25 ARM board.

This base project includes a PCB edge defined as the same size as the Raspberry-Pi PCB with the connectors placed correctly to align the two boards. All IO present on the Raspberry-Pi board is connected to the project through the 0.1" expansion headers.

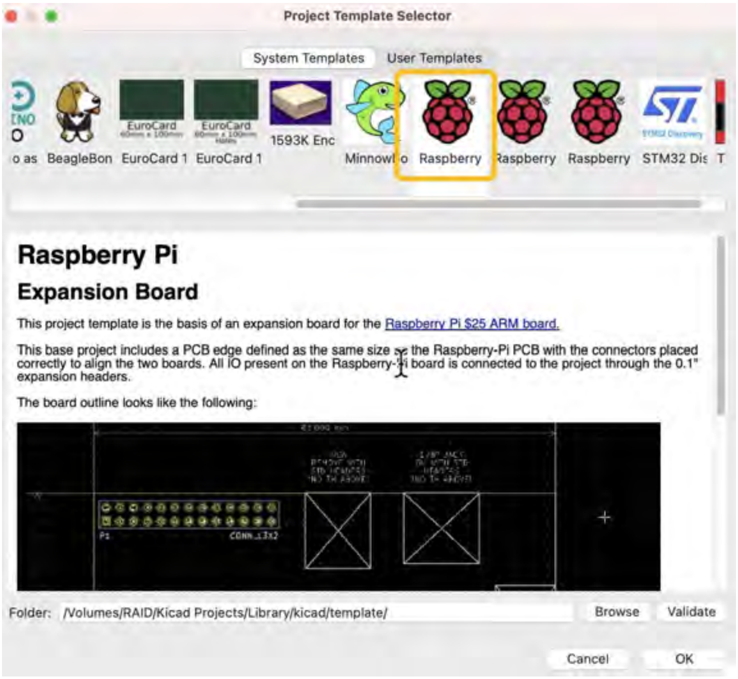
The board outline looks like the following:

Folder: /Volumes/RAID/Kicad Projects/Library/kicad/template/

KiCad comes with several project templates ready to use, but you can also create yours. You can go through a dedicated section in the Recipes part of the **KiCad Like a Pro 3e** course or eBook if you are interested in creating custom project templates.

Creating a KiCad project from a template

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).



Raspberry Pi

Expansion Board

This project template is the basis of an expansion board for the Raspberry Pi $25 ARM board.

This base project includes a PCB edge defined as the same size as the Raspberry-Pi PCB with the connectors placed correctly to align the two boards. All IO present on the Raspberry-Pi board is connected to the project through the 0.1" expansion headers.

The board outline looks like the following:

/Volumes/RAID/Kicad Projects/Library/kicad/template/

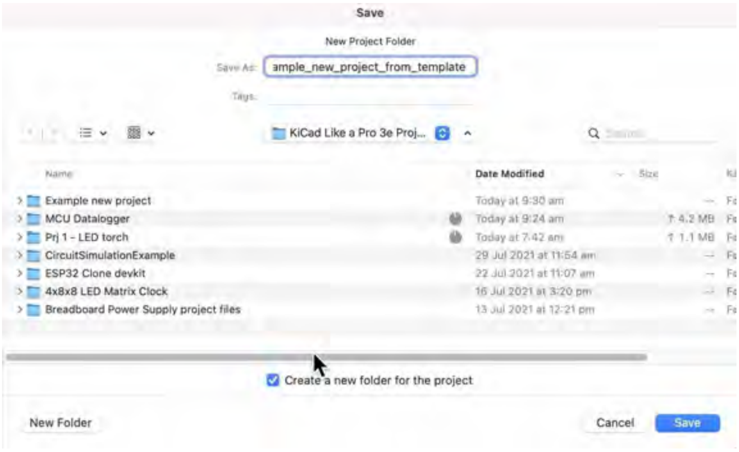
Figure 2.6.1: The project templates selector.

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 section in the Recipes part of the **KiCad Like a Pro 3e** course and eBook).

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.


Save

New Project Folder

Save As: ample_new_project_from_template

Tags:

KiCad Like a Pro 3e Proj...

Name	Date Modified	Size	Ki
Example new project	Today at 9:30 am	--	Fo
MCU Datalogger	Today at 9:24 am	4.2 MB	Fo
Prj 1 - LED torch	Today at 7:42 am	1.1 MB	Fo
CircuitSimulationExample	29 Jul 2021 at 11:54 am	--	Fo
ESP32 Clone devkit	22 Jul 2021 at 11:07 am	--	Fo
4x8x8 LED Matrix Clock	16 Jul 2021 at 3:20 pm	--	Fo
Breadboard Power Supply project files	13 Jul 2021 at 12:21 pm	--	Fo

☑ Create a new folder for the project

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

Working with a KiCad project created from a template

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).

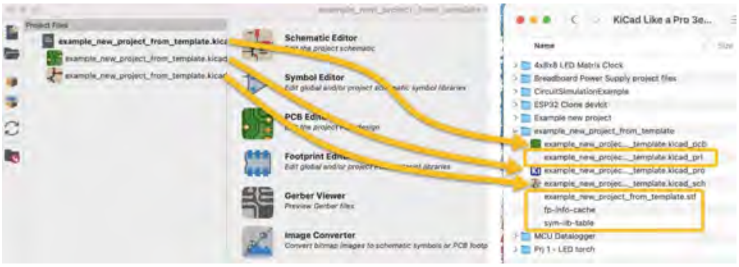

Project Files
example_new_project_from_template.kica
example_new_project_from_template.kicad
example_new_project_from_template.kicad
Schematic Editor
Symbol Editor
PCB Editor
Footprint Editor
Gerber Viewer
Preview Gerber files
Image Converter
KiCad Like a Pro 3e...
Name
4x8x8 LED Matrix Clock
Breadboard Power Supply project files
CircuitSimulationExample
ESP32 Clone devkit
Example new project
example_new_project_from_template
example_new_projec..._template.kicad_pcb
example_new_projec..._template.kicad_prl
example_new_projec..._template.kicad_pro
example_new_projec..._template.kicad_sch
example_new_project_from_template.stl
fp-info-cache
sym-lib-table
MCU Datalogger
Prj 1 - LED torch

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:

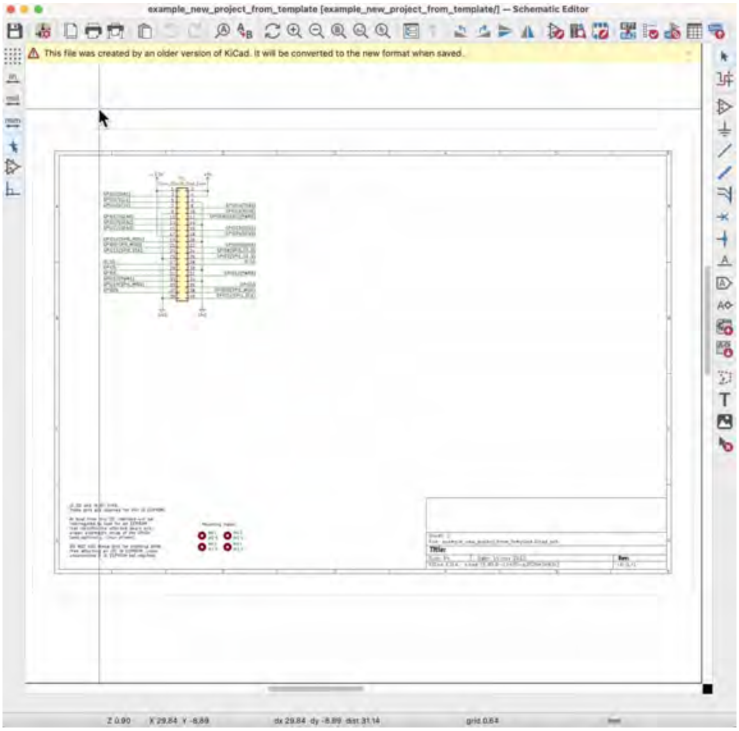
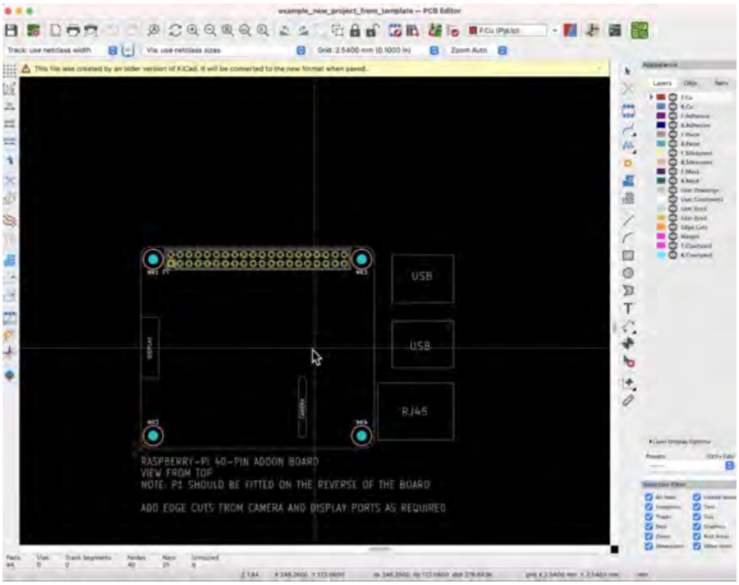

example_new_project_from_template [example_new_project_from_template/] — Schematic Editor
This file was created by an older version of KiCad. It will be converted to the new format when saved.
Z 0.90
X 29.84 Y -8.89
dx 29.84 dy -8.89 dist 31.14
grid 0.64

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:


example_new_project_from_template — PCB Editor

This file was created by an older version of KiCad. It will be converted to the new format when saved.

USB
USB
RJ45
DISPLAY
CAMERA
RASPBERRY-PI 40-PIN ADDON BOARD
VIEW FROM TOP
NOTE: P1 SHOULD BE FITTED ON THE REVERSE OF THE BOARD
ADD EDGE CUTS FROM CAMERA AND DISPLAY PORTS AS REQUIRED

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

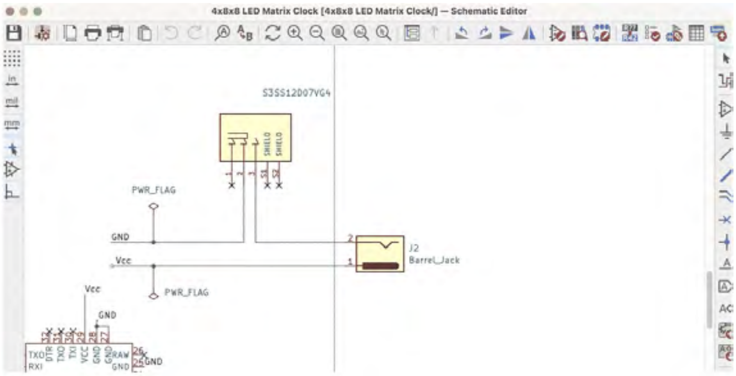
As you can see, much of the work has already been done. In the layout editor, the design of the board outline requires exact measurements, which are time-consuming. The placement of the mounting holes and connectors, likewise, must be exact and, as a result, very time-consuming. All this is work that 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 the **KiCad Like a Pro 3e** course and eBook.

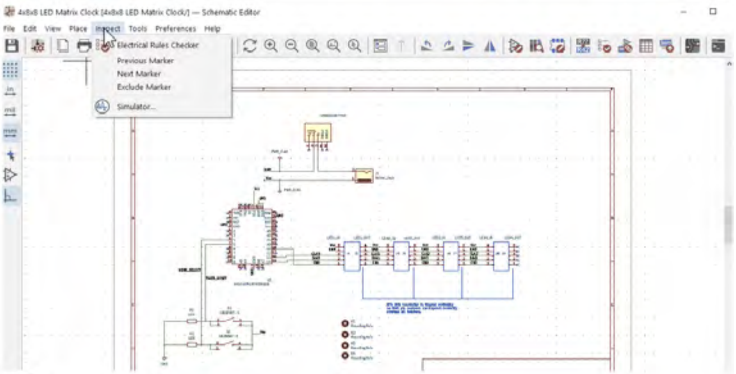
KiCad 6 On Mac OS, Linux, And Windows

Since its inception, KiCad has supported a wide range of operating systems with varying degrees of reliability.

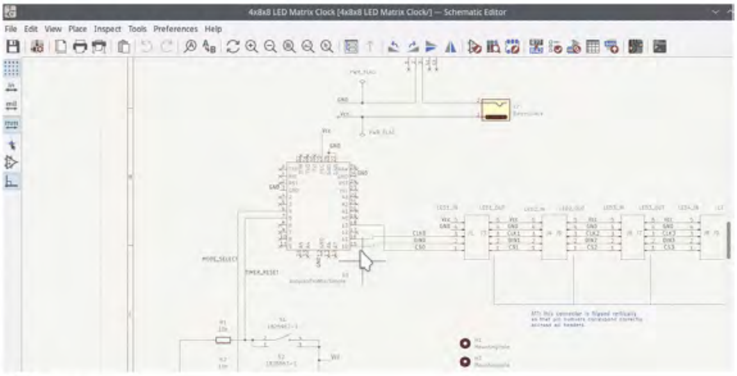
In this article, learn how KiCad 6 looks and behaves on the various operating systems that it supports.


4x8x8 LED Matrix Clock [4x8x8 LED Matrix Clock/] — Schematic Editor
S3SS12D07VG4
PWR_FLAG
GND
Vcc
J2
Barrel_Jack
RAW
GND

1 Mac OS



2 Windows


4x8x8 LED Matrix Clock [4x8x8 LED Matrix Clock/] — Schematic Editor
File
Edit
View
Place
Inspect
Tools
Preferences
Help

3 Kubuntu

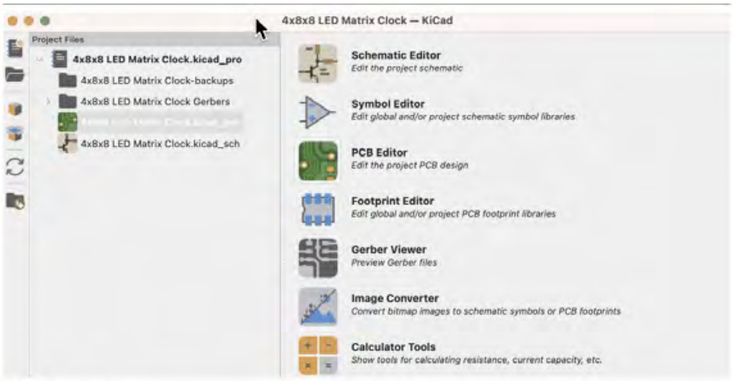
A consistent look and behavior between operating systems

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.


4x8x8 LED Matrix Clock — KiCad

Project Files

- 4x8x8 LED Matrix Clock.kicad_pro
  - 4x8x8 LED Matrix Clock-backups
  - 4x8x8 LED Matrix Clock Gerbers
  - 4x8x8 LED Matrix Clock.kicad_pcb
  - 4x8x8 LED Matrix Clock.kicad_sch

Schematic Editor *Edit the project schematic*

Symbol Editor *Edit global and/or project schematic symbol libraries*

PCB Editor *Edit the project PCB design*

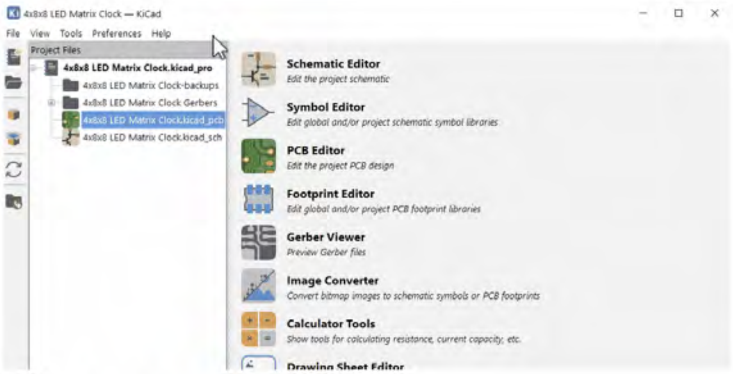
Footprint Editor *Edit global and/or project PCB footprint libraries*

Gerber Viewer *Preview Gerber files*

Image Converter *Convert bitmap images to schematic symbols or PCB footprints*

Calculator Tools *Show tools for calculating resistance, current capacity, etc.*

1 Mac OS


4x8x8 LED Matrix Clock — KiCad

File View Tools Preferences Help

Project Files

- 4x8x8 LED Matrix Clock.kicad_pro
  - 4x8x8 LED Matrix Clock-backups
  - 4x8x8 LED Matrix Clock Gerbers
  - 4x8x8 LED Matrix Clock.kicad_pcb
  - 4x8x8 LED Matrix Clock.kicad_sch

Schematic Editor *Edit the project schematic*

Symbol Editor *Edit global and/or project schematic symbol libraries*

PCB Editor *Edit the project PCB design*

Footprint Editor *Edit global and/or project PCB footprint libraries*

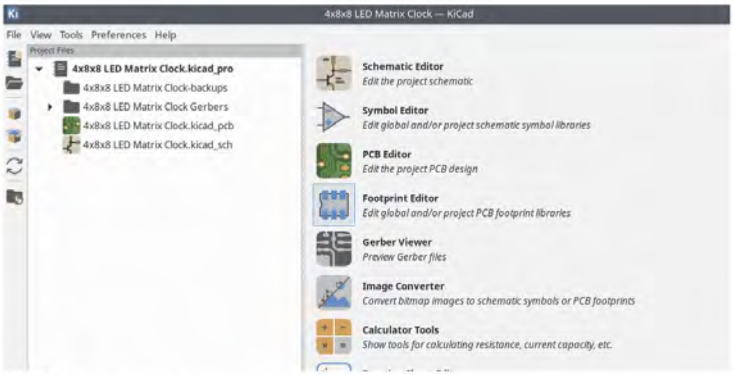
Gerber Viewer *Preview Gerber files*

Image Converter *Convert bitmap images to schematic symbols or PCB footprints*

Calculator Tools *Show tools for calculating resistance, current capacity, etc.*

Drawing Sheet Editor

2 Windows


4x8x8 LED Matrix Clock — KiCad

File View Tools Preferences Help

Project Files

- 4x8x8 LED Matrix Clock.kicad_pro
  - 4x8x8 LED Matrix Clock-backups
  - 4x8x8 LED Matrix Clock Gerbers
  - 4x8x8 LED Matrix Clock.kicad_pcb
  - 4x8x8 LED Matrix Clock.kicad_sch

Schematic Editor *Edit the project schematic*

Symbol Editor *Edit global and/or project schematic symbol libraries*

PCB Editor *Edit the project PCB design*

Footprint Editor *Edit global and/or project PCB footprint libraries*

Gerber Viewer *Preview Gerber files*

Image Converter *Convert bitmap images to schematic symbols or PCB footprints*

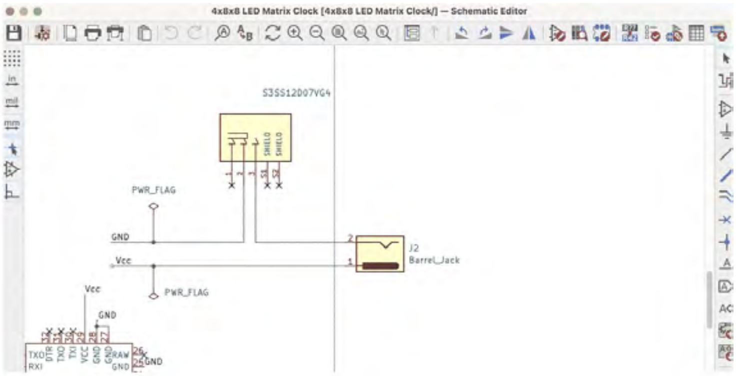
Calculator Tools *Show tools for calculating resistance, current capacity, etc.*

3 Linux Kubuntu

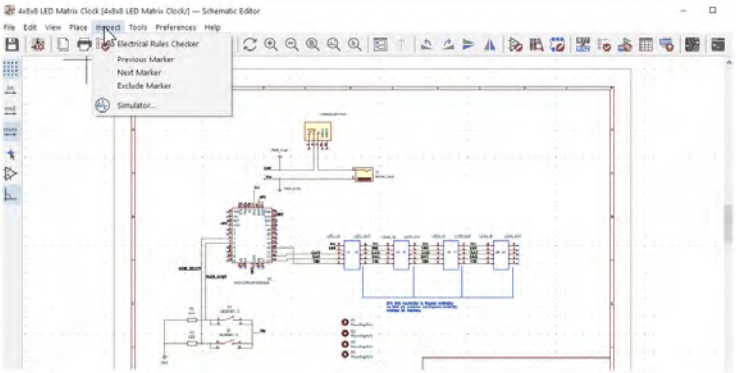
Figure 2.7.1: KiCad project window on three OSs.

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.

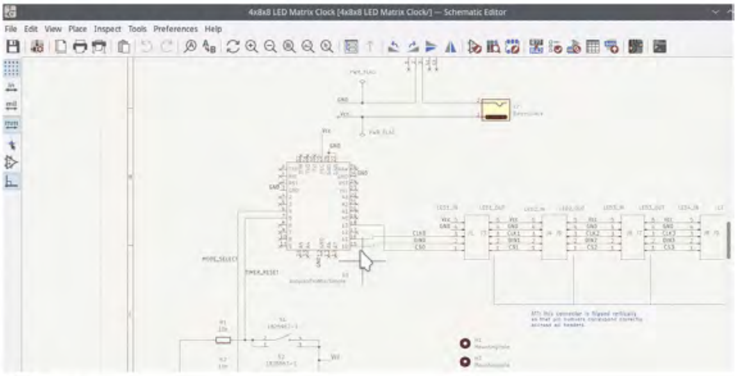
Below is an example of **Eeschema** in the three operating systems:


4x8x8 LED Matrix Clock [4x8x8 LED Matrix Clock/] — Schematic Editor
S3SS12D07VG4
PWR_FLAG
GND
Vcc
J2
Barrel_Jack

1 Mac OS


4x8x8 LED Matrix Clock [4x8x8 LED Matrix Clock/] — Schematic Editor
File Edit View Place Inspect Tools Preferences Help
Electrical Rules Checker
Previous Marker
Next Marker
Exclude Marker
Simulator...

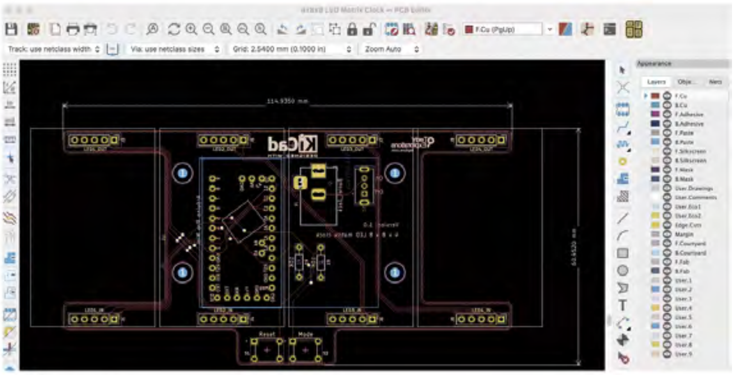
2 Windows


4x8x8 LED Matrix Clock [4x8x8 LED Matrix Clock/] — Schematic Editor
File Edit View Place Inspect Tools Preferences Help

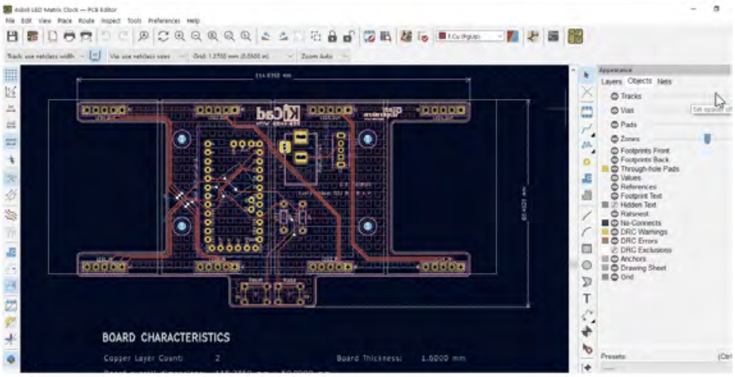
3 Kubuntu

Figure 2.7.2: Eeschema in the three OSs.

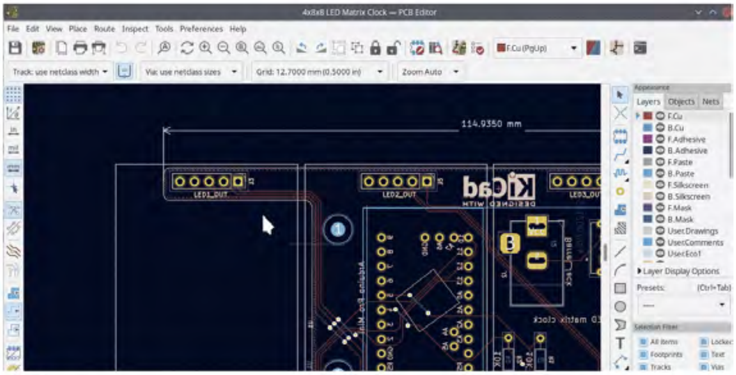
And here is **Pcbnew**:


4x8x8 LED Matrix Clock — PCB Editor
Track: use netclass width
Via: use netclass sizes
Grid: 2.5400 mm (0.1000 in)
Zoom Auto
F.Cu (PgUp)
114.9350 mm
Appearance
Layers
Nets
Reset
Mode

1 Mac OS



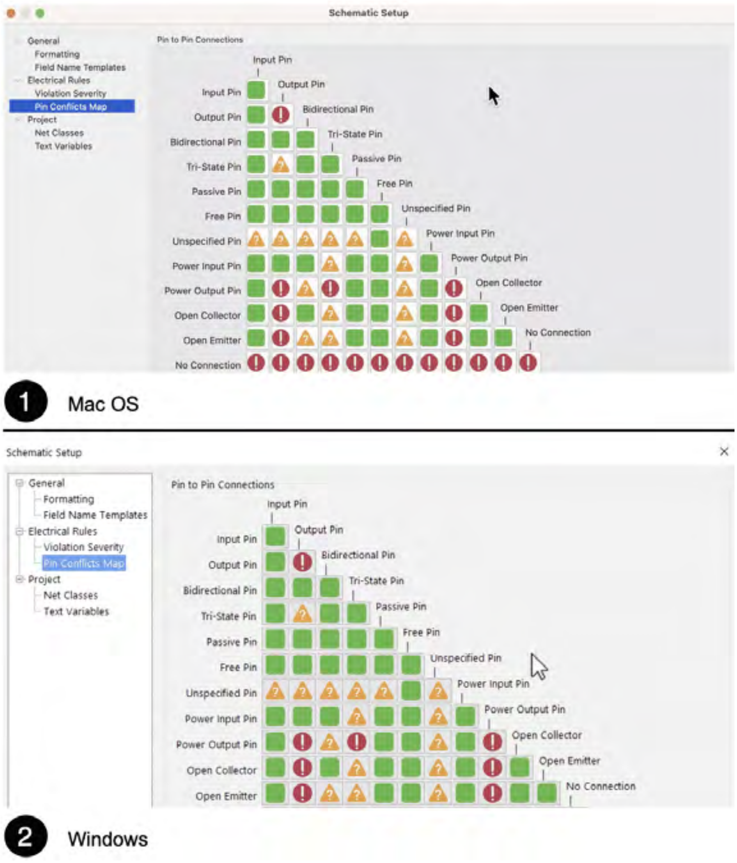
2 Window



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.



Schematic Setup

- General
  - Formatting
  - Field Name Templates
- Electrical Rules
  - Violation Severity
  - Pin Conflicts Map
- Project
  - Net Classes
  - Text Variables

Pin to Pin Connections

	Input Pin	Output Pin	Bidirectional Pin	Tri-State Pin	Passive Pin	Free Pin	Unspecified Pin	Power Input Pin	Power Output Pin	Open Collector	Open Emitter	No Connection
Input Pin	🟩											
Output Pin	🟩	❗										
Bidirectional Pin	🟩	🟩	🟩									
Tri-State Pin	🟩	⚠	🟩	🟩								
Passive Pin	🟩	🟩	🟩	🟩	🟩							
Free Pin	🟩	🟩	🟩	🟩	🟩	🟩						
Unspecified Pin	⚠	⚠	⚠	⚠	⚠	🟩	⚠					
Power Input Pin	🟩	🟩	🟩	⚠	🟩	🟩	⚠	🟩				
Power Output Pin	🟩	❗	⚠	❗	🟩	🟩	⚠	🟩	❗			
Open Collector	🟩	❗	🟩	⚠	🟩	🟩	⚠	🟩	❗	🟩		
Open Emitter	🟩	❗	⚠	⚠	🟩	🟩	⚠	🟩	❗	🟩	🟩	
No Connection	❗	❗	❗	❗	❗	❗	❗	❗	❗	❗	❗	❗

Schematic Setup

- General
  - Formatting
  - Field Name Templates
- Electrical Rules
  - Violation Severity
  - Pin Conflicts Map
- Project
  - Net Classes
  - Text Variables

Pin to Pin Connections

	Input Pin	Output Pin	Bidirectional Pin	Tri-State Pin	Passive Pin	Free Pin	Unspecified Pin	Power Input Pin	Power Output Pin	Open Collector	Open Emitter	No Connection
Input Pin	🟩											
Output Pin	🟩	❗										
Bidirectional Pin	🟩	🟩	🟩									
Tri-State Pin	🟩	⚠	🟩	🟩								
Passive Pin	🟩	🟩	🟩	🟩	🟩							
Free Pin	🟩	🟩	🟩	🟩	🟩	🟩						
Unspecified Pin	⚠	⚠	⚠	⚠	⚠	🟩	⚠					
Power Input Pin	🟩	🟩	🟩	⚠	🟩	🟩	⚠	🟩				
Power Output Pin	🟩	❗	⚠	❗	🟩	🟩	⚠	🟩	❗			
Open Collector	🟩	❗	🟩	⚠	🟩	🟩	⚠	🟩	❗	🟩		
Open Emitter	🟩	❗	⚠	⚠	🟩	🟩	⚠	🟩	❗	🟩	🟩	

Figure 2.7.4: Schematic Setup in Mac OS and Windows.

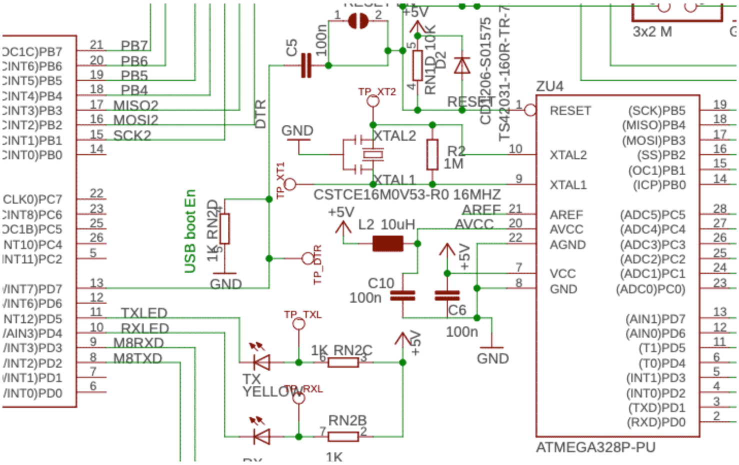
The quality of the implementation of KiCad in the three operating systems I have tested is excellent. The implication for solo users and teams is that you can use KiCad 6 with high confidence that you can edit the same projects across platforms. If you are in a team, your team members will work using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

KiCad 6 Guide series

KiCad Schematic Symbols

Before moving on to more exciting PCB projects, it’s a good idea to brush up on the concepts, conventions, and design patterns that will help you work more efficiently and produce better-performing boards.

In this article, you will learn about the symbols and units that appear in schematic and layout diagrams. You’ll also discover the terms used to describe the various components and characteristics of a printed circuit board.



Design principles and PCB terms: Introduction

In the previous articles of this guide series, you learned about some of KiCad’s most commonly used features. You used **Eeschema** and **Pcbnew** to design the schematic diagram and the layout of a simple board. You also had your first experience with some of the decisions a PCB designer must make during a printed circuit board design process.

Before you continue your learning journey with more interesting PCB projects, it is appropriate to become familiar with concepts, conventions, and design patterns that will help you work and produce better-performing boards.

You will learn about the **symbols** and **units** that appear in the schematic diagrams and the layout diagrams.

You will also learn the terminology used to describe a printed circuit board’s various components and characteristics.

In Part six of the **KiCad Like a Pro 3e** course and eBook, you can learn about the general processes of the schematic and layout steps of designing a PCB. These are processes that every designer will go through to one degree or another, regardless of which CAD application they are using.

Schematic symbols

Electronics and PCB design has their own symbolic language. We use this language to create schematic diagrams. In Figure 5.2.1 you can see an example of a schematic that contains several symbols of this language.

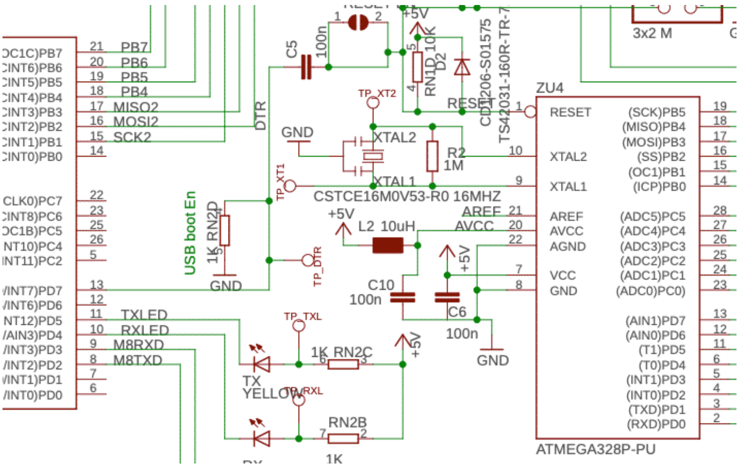


Figure 5.2.1: A segment of the Arduino Uno Rev3 schematic diagram. (Source of this schematic.)

This example shows the schematic symbols of several components that make up the Arduino Uno Rev3. The large rectangular symbol with the designator ‘ZU4’ is the ATMEGA328P-PU microcontroller chip. The symbol contains several pins that provide inputs and outputs, and each of them is named.

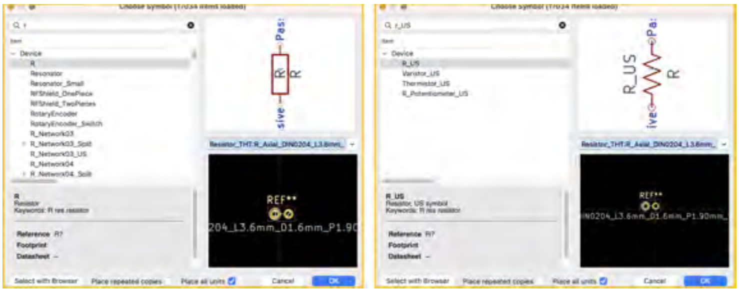
You can see a few capacitors, designated C5, C10, and C6, none of them polarized. There is one resistor, R2, and a few resistor networks (like RN2 and RN1, which are simple resistors in a network configuration). The capacitors and the resistor also have their values marked in the schematic. There are also symbols for an LED, a jumper connector, and power (Vcc, RAW, and GND).

All of these symbols follow a particular standard. . Several standards are available, but most notably, engineers worldwide tend to work with the American style (‘IEEE’) or the European (‘IEC’) style. The symbols in Figure 5.2.1 follow the IEC style.

In KiCad, you can choose to use either the American or the

European style symbols. Whichever one you choose, be consistent. Do not mix American-style resistor symbols with European-style capacitor symbols in the same schematic.

The KiCad standard symbols library contains symbols of both styles, though the European symbols seem more plentiful. In most cases, American-style symbols will have the postfix ‘US’ in their name. In Figure 5.2.2, you can see an example of a resistor symbol, with the European style on the left and the American on the right. The name of the American-style symbol ends in ‘US,’ and you can take that into account when you are searching for a symbol in the Symbol Chooser (Figure 5.2.2).



5.2.2: A resistor, European-style (left) and American-style (right).

Ready to learn KiCad?

Learn the world’s favourite open-source PCB design tool with the world’s most comprehensive course

KiCad Like a Pro, 3rd edition is available as a video course or as an eBook.

Choose the version that fits best with your style of learning, or get both to get the full benefit of the video demos plus the details of the eBook.

When you complete KiCad Like a Pro 3e, you’ll be able to use KiCad to design and manufacture multi-layer PCBs with highly integrated components and a professional-looking finish.

Work through five projects that give many opportunities to learn and practice all of KiCad’s important features.

KiCad Like a Pro 3e contains full sections dedicated to PCB and design principles and concepts. These ensure that you will master the fundamentals so that your PCB project are awesome.

If you are someone who is interested in designing PCBs using KiCad, or moving to KiCad from another CAD application, then KiCad Like a Pro, the video course and eBook, is for you.

PCB key terms

KiCad 6 Guide series

PCB Key Terms

This article will teach you the most commonly used terms for understanding information found on PCB fabrication websites and CAD tool documentation.

Creating printed circuit boards is an engineering discipline. As such, it has its own 'language.' In this article, you will learn the most commonly used terms to understand the information found in places such as PCB fabrication websites and CAD tool documentation.

FR4

The most common material used to make printed circuit boards is **FR4** (or FR-4). It is a glass-reinforced epoxy laminate composite material, or in simpler terms, fiberglass cloth bound using an epoxy resin.

Go over to the Wikipedia article to read more about this material.

The ‘FR’ part of the name stands for ‘Flame Retardant,’ a desirable quality for a board that will hold together components that can potentially ignite when they fail.

Other valuable attributes of the FR4 substrate are:

- Very light and strong
- Does not absorb water
- It is an excellent isolator
- Maintains its quality in dry and humid environments

Other materials can be used in rigid or flexible printed circuit boards, apart from the standard FR4 and variants (like FR4 tracking resistant and halogen-free). Examples include G-11 for applications that must operate in high temperatures, FR-3 (cotton-paper impregnated with epoxy), and Polyimide (high-performance yet expensive, appropriate for cryogenic applications).

Traces

Traces (also called ‘tracks’) are conductive paths. Most often, the material used to make traces is copper. Electrical signals and power use traces to travel throughout a circuit.

In Figure 5.3.2.1, you can see the traces in the front side of this PCB as thin purple lines that provide the connections between the golden pads where the component terminals will eventually be. You will learn more about pads later.

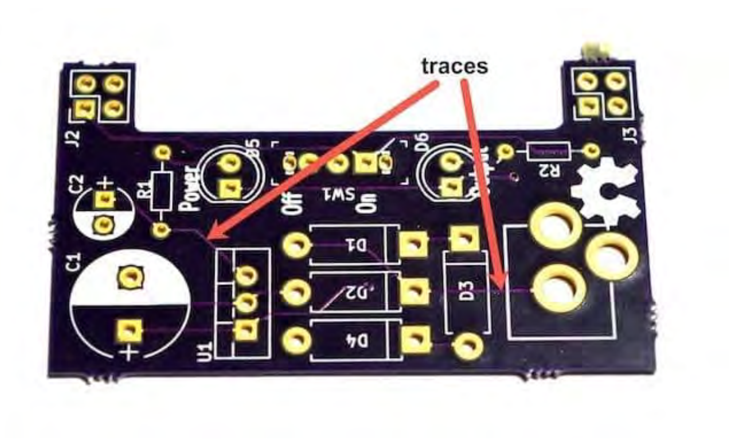


Figure 5.3.2.1: An example of traces.

As the designer of a PCB, you have total control over the characteristics of traces. You can control their width, height, and route, including the angles by which a trace changes direction. If you want a trace to accommodate a large current flowing through it with little resistance and temperature rise, you can design it wider and thicker. This is useful when a trace must feed power to the components of your board. Traces that convey low power-current signals (less than 20 mA) can be narrower using less copper.

Keeping the width of traces to around 0.3 mm (or even less, depending on your manufacturer's guidelines) makes it possible to draw traces closer together and reduce the final size of your PCB.

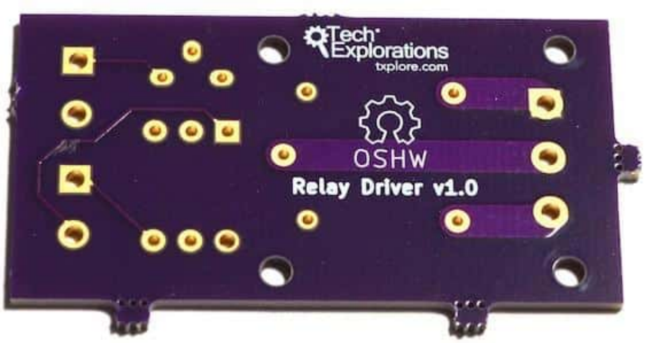


Figure 5.3.2.2: An example of wide traces.

In Figure 5.3.2.2, you can see an example of much wider traces than regular signal traces. These traces connect the terminals of a 240 Volt relay.

The traces in these examples are purple because of the solder mask chemical used to finish the manufacturing process. You will learn about the solder mask further down in this article.

Pads and holes

Pads and **holes** are the most prominent feature of a printed circuit board. Pads come in two varieties: TH (through-hole) and SMD (surface-mounted device). For each, there are several shapes.

In Figure 5.3.3.1, you can see an example of a board that contains TH pads exclusively, and in Figure 5.3.3.2, you can see a board with TH and SMD pads.

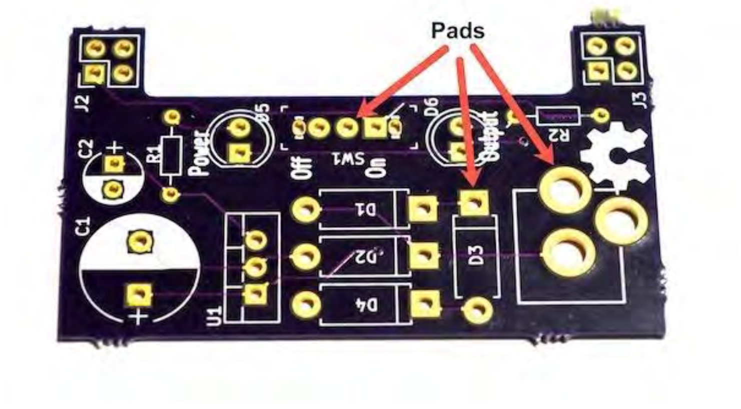


Figure 5.3.3.1: Through-hole pads.

Through-hole pads, unlike SMD pads, connect the front for the PCB with the back electrically. In the examples, you can see that the gold plating of the pad fills the inside of the hole. If you turn the PCB around, you will see that a matching pad exists in the back.

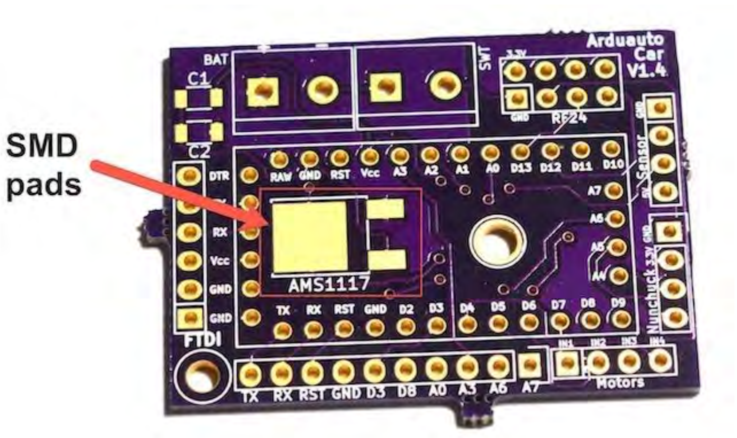


Figure 5.3.3.2: An example of SMD pads.

Boards with mostly TH pads are popular among hobbyists because through-hole components are easier to work with, at least initially. SMD components are smaller; hobbyists tend not to use them until they are more comfortable with their soldering skills. With a bit of practice, SMD components are as easy to work with as their TH counterparts.

In the industry, on the other hand, the vast majority of PCBs are designed to contain SMD components. This is because SMD components can be populated automatically using pick and place machines and because their small size results in smaller PCBs.

Apart from the two varieties I described above, pads also come in several shapes. Most often, you will see round pads, but rectangular and oval shapes are also possible. Using KiCad, you can create such pads and control their geometry to the extent that your PCB manufacturer allows.

In Figure 5.3.3.3, you can see an illustration of a cross-section of a PCB showing the configuration of pads, and two types of holes, Plated-Through Hole (PTH) and Non-Plated Through Hole (NPTH).

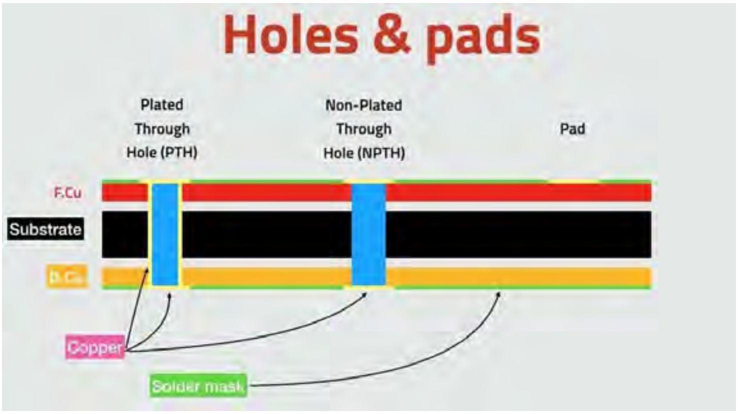


Figure 5.3.3.2: An example of SMD pads.

Plated-Through Holes is the more common variety and the default type of hole in more cases. We use a drill to create the hole and then copper to cover the hole's sides so that its two ends (at the front and back copper layers) are electrically connected. Vias are constructed the same way, except they have a smaller diameter, so it is impossible to accommodate component pins.

On the other hand, in a Non-Plated Through Hole, we use the same drill to create the hole, but no copper is used to cover the sides of the hole, so there is no electrical connection between its two ends.

Finally, pads without holes are useful for attaching surface-mounted components, as you learned earlier.

Via

You can create a **via** when you want to move a signal that travels across a trace from one side of a PCB to another (say, from front to back). A via is a hole with its sides covered with copper or gold (or other conductive material) that allows a trace to continue its route across layers.

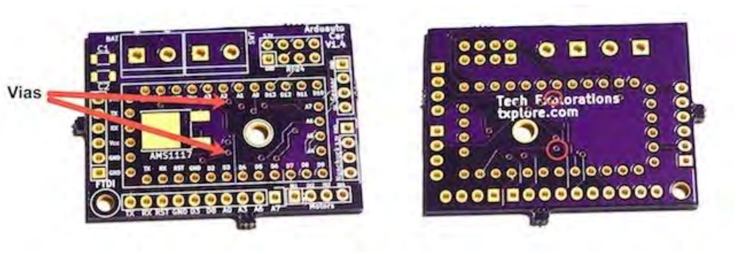


Figure 5.3.4.1: Vias allow a trace to continue between layers.

In Figure 5.3.4.1, you can see the two sides of the same PCB. On the left, the arrows point to two vias in the front of the PCB, and on the right, the circles indicate the same vias on the back of the PCB. Vias are similar to through-hole pads, except they don't have any exposed copper (the solder mask covers them), and they don't have a pad (so you can't solder a component).

In simple circuits with only a few components, it is possible to create all traces on one layer of the PCB. When a PCB gets busy with more components, it quickly becomes impossible to do the routing on a single layer. When multiple layers are needed, vias provide the simplest method of allowing a trace to use the available board real estate.

In Figure 5.3.4.2, you can see the types of interconnections between layers that are possible.

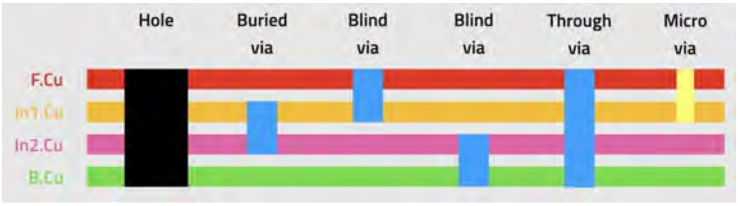


Figure 5.3.4.2: Types of interconnections between layers.

For through-hole components, you would design a hole that connects the top and bottom copper layers. We use a drill to create this hole. It is wide enough to allow for the pin of the component to go through it.

In vias, the diameter is smaller when compared to a regular platted hole. They are not wide enough for pins to go through them, but they are plated, like holes, and allow for electrical connection between layers.

A 'through via' is like a hole but narrower. It connects the top and bottom layers. A buried via is a via that connects any two internal layers. In the four-layer example of Figure 5.3.4.2, the buried via connects the In1.Cu and In2.Cu. A 'blind via' also connects two layers but has one end exposed to the outside of the board, either top or bottom.

Another option for interconnecting layers in high-density boards is to use a 'micro via' ('uvia'). A micro via is made using a high-powered laser instead of a mechanical drill; the use of lasers makes it possible to reduce the diameter of the via dramatically .

Annular ring

The **annular ring** is a term that describes the area on a pad that surrounds a via. A primary metric of an annular ring is its width, defined as the minimum distance between the edge of the pad and the edge of the via or pad hole.

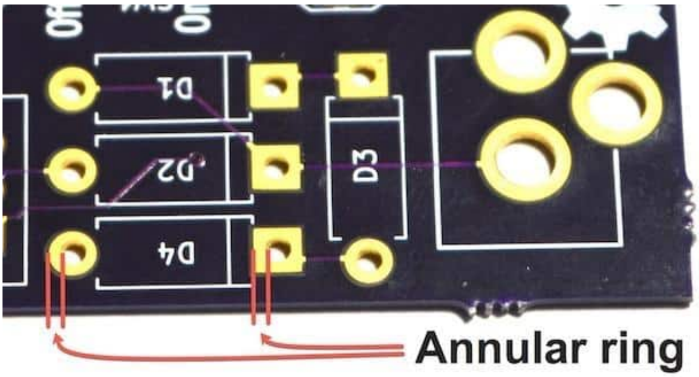


Figure 5.3.5.1: Annular rings and width.

In Figure 5.3.5.1, the width of two annular rings is marked with two red lines. Ideally, the drill hit (the location on the board where the drill lands and creates a hole) is in the middle of the pad. If the drill bit is not aligned correctly, the hole can be closer to one edge of the pad (a ‘tangency’), or it could even miss the pad completely (a ‘breakout’).

Soldermask

As you know, traces are made of copper. Copper slowly reacts with oxygen in the air, resulting in oxidization. Oxidized copper produces a pale green outer layer. PCB manufacturers cover the exposed copper with a **solder mask**, a thin layer of polymer that insulates it from oxygen to prevent this from happening. As an additional benefit, the solder mask also prevents solder bridges from forming between pads.

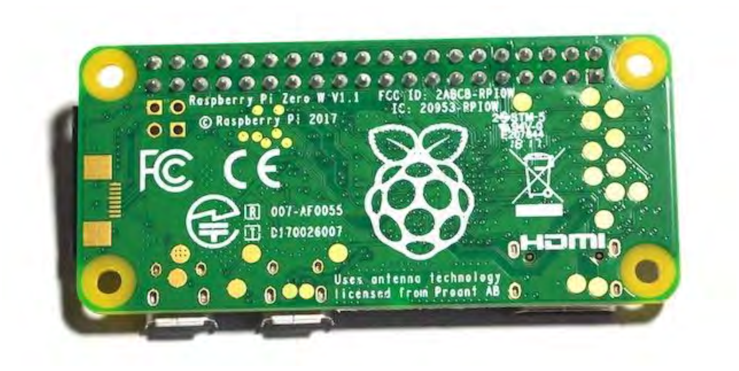


Figure 5.3.6.1: The rear of a Raspberry Pi Zero is protected by a thin layer of solder mask.

In Figure 5.3.6.1, you can see the back of a Raspberry Pi Zero. In this example, the copper is protected by a thin layer of green solder mask. Only the pads and the mounting holes are not covered by the solder mask.

Solder mask polymers are available in different colors, with green being the most common and cheaper. You can create fancy-looking PCBs with black, blue, red, purple, and many other colors.

Silkscreen

Printed circuit boards are not complete without text and artwork. The purpose of those elements is to convey useful information and add a touch of elegance. In Figure 5.3.7.1, you can see an example of such text and artwork on the back of a Raspberry Pi Zero. You can see the Raspberry Pi logo, logos of various certifications, and different text items that inform us about the model, etc. All this consists of the **silkscreen**.

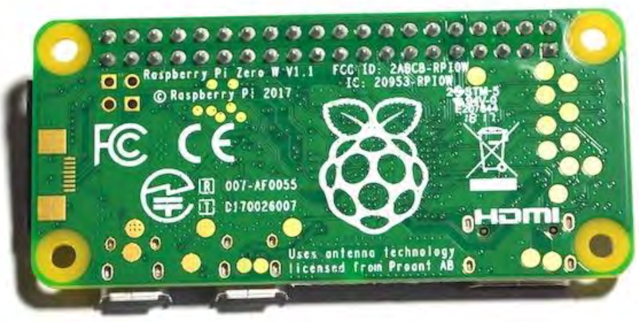


Figure 5.3.7.1: The

The name 'silkscreen' is somewhat misleading. Of course, no natural silk is used to produce the white elements on the PCB. The method used to print the silkscreen in large numbers is a relative of the traditional screen printing process that you can use to print a graphic on a T-shirt. The silkscreen text and graphics are printed on the boards while they are still in their panels.

White is the most common color for the silkscreen, but black and yellow are also available.

In the projects that you will work through during the **KiCad Like a Pro 3e** course or eBook, you will spend a considerable amount of time creating the informational and decorative text and graphics in the silkscreen layer of the PCB.

Drill bit and drill hit

Drill bits are used to create holes and vias, but also cutouts. Drill bits are typically made of solid coated tungsten carbide material and come in many sizes, like 0.3 mm, 0.6 mm, and 1.2 mm. They look like the one in Figure 5.3.8.1. These drills are attached to computer-controlled drilling machines and are guided by a file that contains information about the coordinates and the drill size for each hole on the PCB.

Figure 5.3.8.1: A drill bit.

It is interesting to note that drill bits are replaced with lasers for tiny holes, like vias. These vias are often called ‘micro vias. With laser drilling, it is also possible to create vias that connect in-between layers of the PCB.

The term **‘drill hit’** describes the location on the PCB where the drill bit comes in contact with the PCB and creates a hole.

Surface mounted devices

If your objective is to create a PCB that is easy to manufacture in large numbers, with a minimum size, you should design it to contain surface-mounted components instead of through-hole components.

In Figure 5.3.9.1, you can see an example of what is possible to do with SMD on PCB. A computer, on a tiny board, for a few dollars.

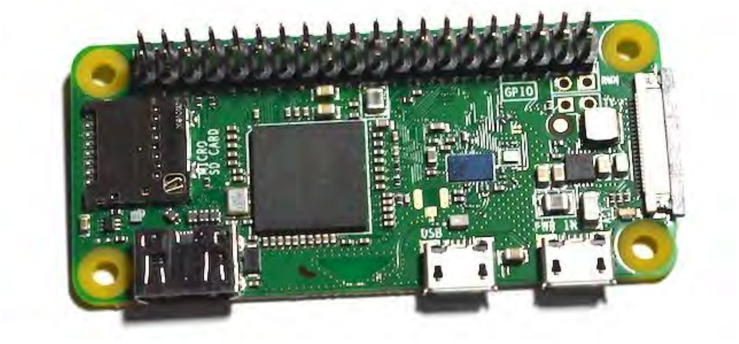


Figure 5.3.9.1: The Raspberry Pi Zero contains almost exclusively SMD components.

On this board are a highly integrated microprocessor, memory, communications, and connectors. Even the connectors are SMD. The only through-hole component is the pin header.

Creating something like this using through-hole components, if at all possible, would result in a board that was many times the size of the Raspberry Pi Zero and would cost many times more because most of the assembly would have to be done by hand.

While hobbyists prefer to work with TH components because they are easier to solder and repair, learning to work with SMD, at least the larger ones, is certainly possible.

In the **KiCad Like a Pro 3e** course and eBook, you will learn how to create an SMD version of a PCB, in addition to the TH version.

Gold Fingers

Appropriately called **'Gold fingers'** are gold-plated connectors placed on the edge of a PCB. Gold fingers are useful for interconnecting one board to another. You can see an example in Figure 5.3.10.1; it shows the micro:bit educational single board computer.

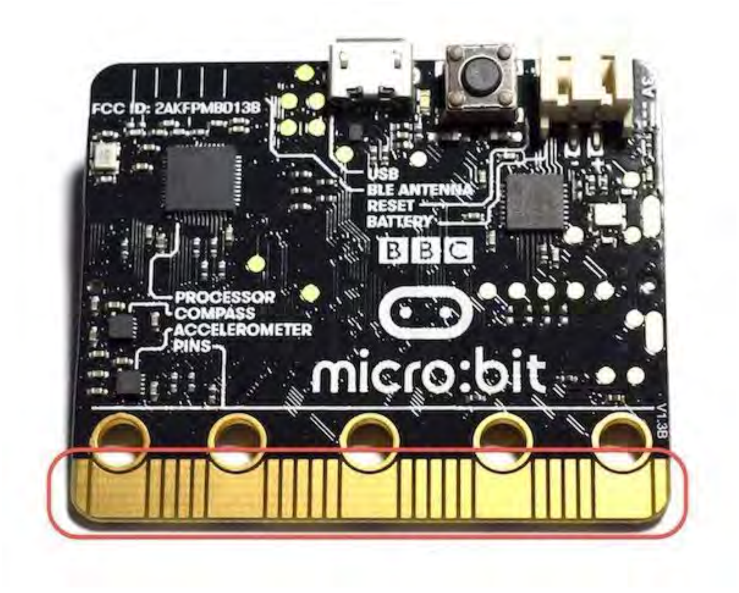


Figure 5.3.10.1: Gold fingers on a Micro:bit.

The micro:bit uses gold fingers to connect to other devices via a slot, like motor controllers and sensors. Gold fingers make it possible to attach and detach the PCB to a slot at least 1,000 times before they start to wear out.

Keep-out areas

A **"keep out area"** is what it sounds like: an area on the PCB that must be clear of components and perhaps even traces.

In Figure 5.3.11.1 I show three examples of devices that contain keep out areas.

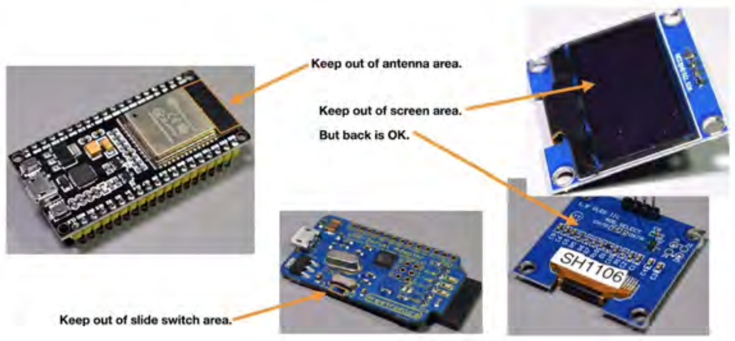


Figure 5.3.11.1: Examples of devices that contain keep out areas

CAD software, including KiCad, allows you to mark an area on the PCB as “keep-out.” In KiCad, you can also configure the keep-out area to prevent the user from adding specific or all types of elements, including footprints, tracks, vias, and cutouts. You can also tell KiCad to apply the keep-out rules to specific (or all) copper layers.

On the top left is an ESP32 development kit. The kit is based on an ESP32 module that contains an integrated antenna that requires a patch of PCB clear of components and other traces. You don’t want to add any other components in that area not to affect the antenna’s performance. Note that the keep-out area must include all copper layers, not just the front one where the antenna is.

On the right side of Figure 5.3.11.1, I show the back and front sides of a TFT screen that I use in my Arduino projects. The front side is where the TFT screen is placed. You can see that the screen’s ribbon connector is attached to a row of pads in the back of the PCB. We can mark a keep-out area in the front of the PCB only and allow for footprints and traces to be placed on the back of the PCB.

The last example, in the middle of Figure 5.3.11.1, is a UART interface with a voltage slide switch. The slide switch is oriented to its side. Even though the switch notch is tiny, it is still a good idea to mark the area below it as a keep-out area for footprints but allow traces. This way, there is no risk of placing a footprint by mistake and obstructing the travel path of the notch. However, we can still use that patch of PCB for tracks or vias.

With KiCad, you can create keep-out areas of any shape and configure them in almost any way needed.

Panel

To manufacture PCBs economically, manufacturers use machines that can work on large panels. Each **panel** can contain many copies of the same PCB. It is also possible to use clever algorithms that place different PCBs on the same panel so that the panel's capacity is fully utilized and that the individual cost of each PCB is reduced. This is how it is possible to have a single 'hobby' PCB manufactured for a few dollars. This panelization process is key to this reduction in costs.

In the example of Figure 5.3.12.1, a single panel contains four individual PCBs. The four PCBs are populated while still part of the panel using an automated pick and place machine. A pick and place machine is a robot that uses an arm to pick each component from a container and places it precisely on the pads. Once the components are on the board, the panel moves into the next step of the process, in which they are 'baked' and secured in place.

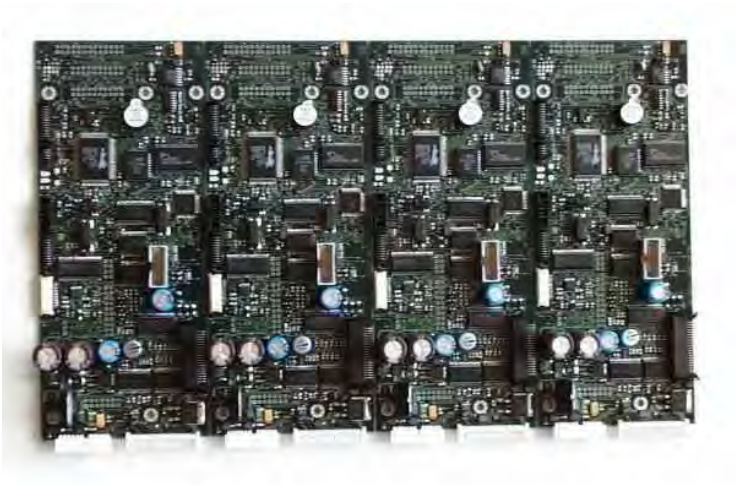


Figure 5.3.12.1: A panel with four individual PCBs.

Manufacturers utilize defined breakaway routes and points on the board to remove the PCBs from the panel to snap them off. In Figure 5.3.12.1, you can see the breakpoints along the edges of this PCB.

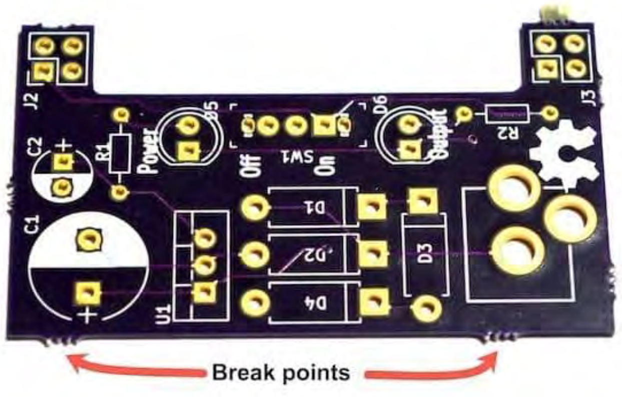


Figure 5.3.12.1: This PCB was part of a panel.

Using a drill, the manufacturer removed the substrate material in between the breakpoints. With a small amount of force, you can remove individual PCBs from the panel without damage.

Solder paste and paste stencil

Solder paste (or solder cream) is a soft and sticky material (at room temperature) applied on pads. The purpose of solder paste is to help attach an SMD component to the pad. Think of solder paste as ordinary solder. With solder, you will need a soldering iron to heat it, melt it, and apply it on a component pin already in place. With solder paste, you will first use a syringe (or one of the other application methods) to cover the pad, then place the component on the pad, and provide heat in the form of an oven to heat the paste and bond it with the pad and the component's plated area.

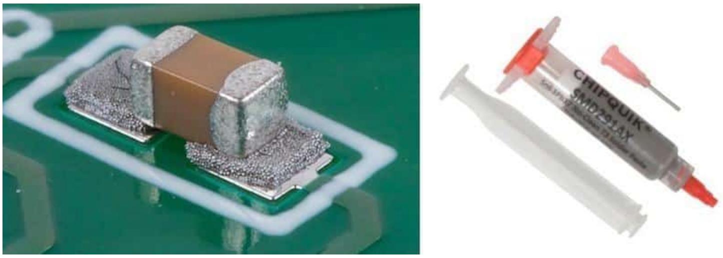


Figure 5.3.13.2: Solder paste in a syringe dispenser and an SMD component.

In Figure 5.3.13.2, you can see an example of a solder paste in a syringe dispenser that you can purchase from retailers like RS Components. Using the syringe equipped with a thin nozzle, you can manually deposit a small amount of solder paste on the pads. Using tweezers, you can place the component you want to attach on the solder paste. Because solder paste is sticky (before it's baked), the component will adhere to it. After you have all the components you want on the board, you place the board in an oven to bake it. After the baking process is complete, the solder paste becomes solid. The SMD components will be mechanically secure and electrically connected to the pads.

Solder paste also comes in a tub, which is more appropriate for application to a board using a stencil (Figure 5.3.13.3). Stencils are helpful in large-scale productions.

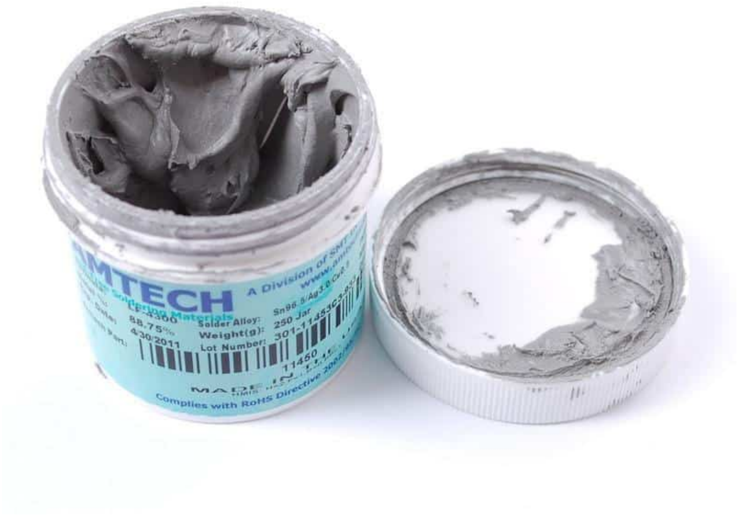


Figure 5.3.13.3: Solder paste is a tub container.

A **stencil**, typically made of stainless steel, is cut to have openings of the exact size and the precise location of the board's pads. The technician will place the stencil over the board and then apply the paste to the openings. When the technician removes the stencil, the paste remains on the pads only.

Then, manually or using an automated pick and place machine, the components are placed on the pads and stick on them because of the paste. The last step is to bake the board in a reflow oven to solidify the paste.

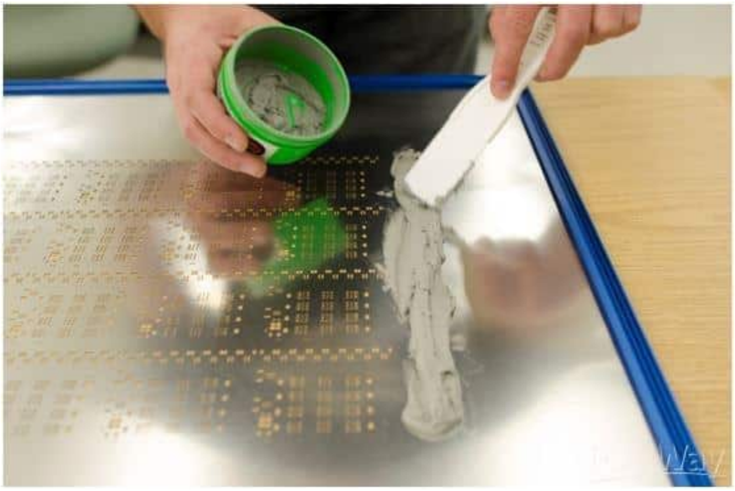


Figure 5.3.13.4: A stencil, with solder paste being applied using a squeegee (photo courtesy of Pcbnew.com).

A reflow oven is an industrial-sized machine used to complete attaching SMD components on a PCB. You can also purchase or make a reflow oven for use at home. People have even made reflow ovens for their projects using discarded toaster ovens. In either case, a reflow oven is designed to operate under a specific program that controls the amount of heat a board receives over time. This is important because the heat must be appropriate for converting the solder paste into good-quality electrical connections without causing damage to the board or the components on it.

Pick-and-place

Pick and place machines are robots that assemble the various components on the surface of a circuit board. When you contract a manufacturer to make your boards and populate them, they will be using a pick and place machine. You can see an example of a pick and place machine in Figure 5.3.14.1.

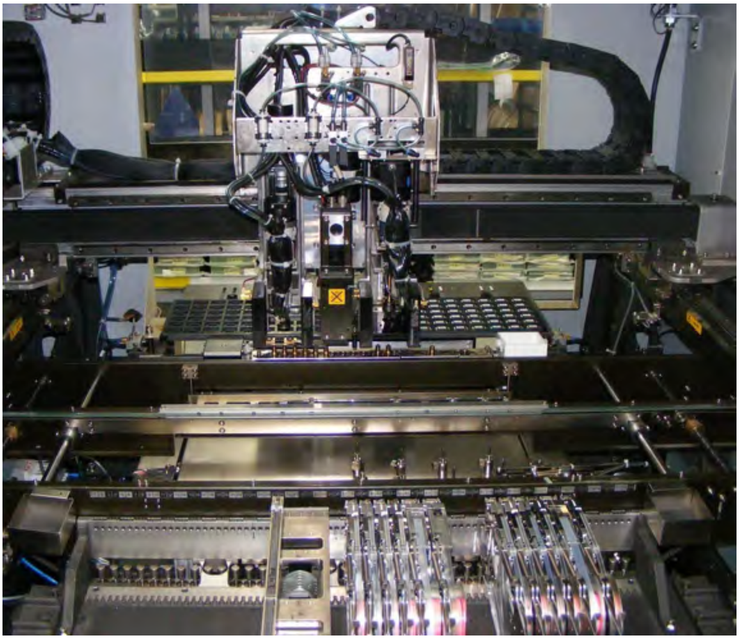


Figure 5.3.14.1: Figure 12.18: A large pick and place machine. By Peripitus [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], from Wikimedia Commons

A typical pick and place machine, like the one in Figure 12.18, includes:

1. A repository of the various components that are to be placed on the board
2. A conveyor belt that brings in the boards.
3. An inspection system composed of cameras that can optically recognize the board, components, and other guidance markings on the board.
4. A robotic arm that can pick a component from the repository and place it on the board

(these arms are usually fitted with suction cups so they can pick and manipulate components).

Modern high-end machines are very versatile, optimized for short runs of complicated boards that employ artificial intelligence. These machines are designed to assemble and test boards autonomously, ensuring high levels of reliability.