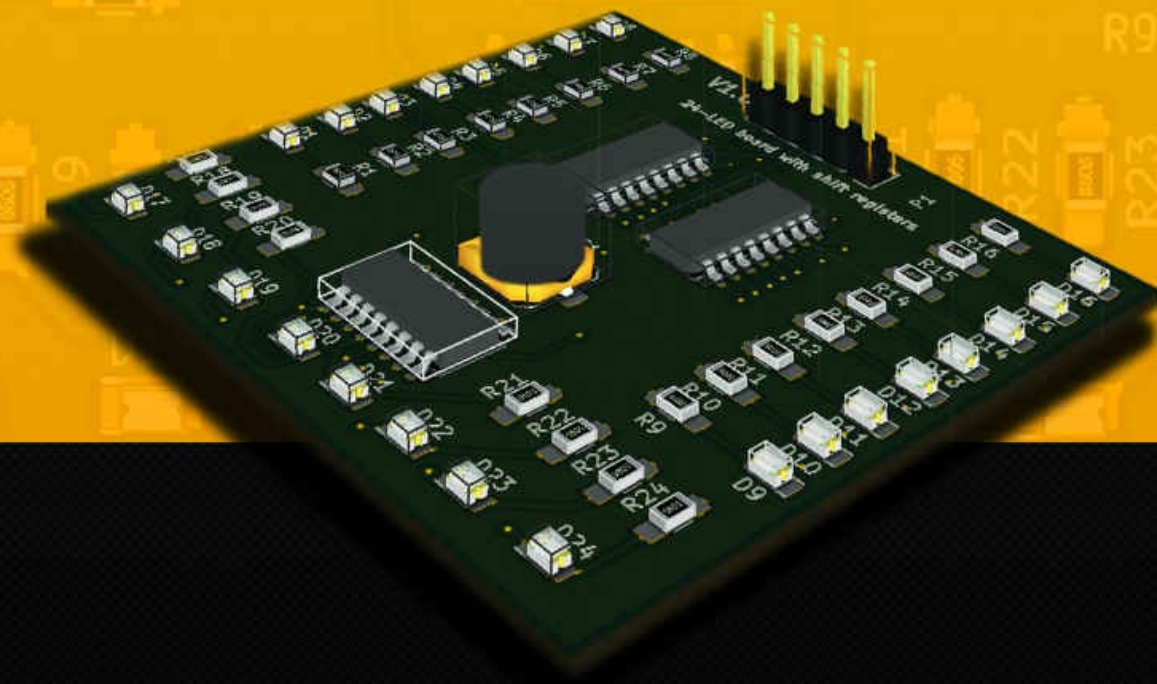




KiCad Like a Pro

Learn the World's Favourite
Open Source PCB Electronic
Design Automation tool!



Peter Dalmaris, PhD



CONTENTS

[About this book](#)

[Video course companion](#)

[Discussion forum and email list](#)

[Dedication](#)

[Copyright](#)

[Part One - About this course](#)

[Chapter 1 - What is Kicad?](#)

[Chapter 2 - The structure of this course](#)

[Part Two - Kicad basics](#)

[Chapter 3 - What is this part](#)

[Chapter 4 - Installation on Windows](#)

[Chapter 5 - Installation on Mac OS X](#)

[Chapter 6 - Kicad main components](#)

[Chapter 7 - Finding documentation](#)

[Chapter 8 - What is a Printed Circuit Board?](#)

[Chapter 9 - The Kicad design process](#)

[Chapter 10 - Fabrication](#)

[Part Three - Project 1: create an nRF24 breakout board](#)

[Chapter 11 - What is this part](#)

[Chapter 12 - Creating a new project](#)

[Chapter 13 - Starting the schematic for the nRF24](#)

[Chapter 14 - How to create a schematic component](#)

[Chapter 15 - Wiring](#)

[Chapter 16 - Annotating the schematic](#)

[Chapter 17 - Electrical Rules Check](#)

[Chapter 18 - Associate components to footprints](#)

[Chapter 19 - Create a custom footprint](#)

[Chapter 20 - Saving the new footprint](#)

[Chapter 21 - Associate the new footprint and component](#)

[Chapter 22 - Create a netlist](#)

[Chapter 23 - Footprints placement](#)

[Chapter 24 - Edge cuts](#)

[Chapter 25 - Wiring](#)

[Chapter 26 - Add text labels](#)

[Part Four - Project 1: Enhancing the design](#)

[Chapter 27 - What is this part](#)

[Chapter 28 - Add a capacitor to the schematic using Eeschema](#)

[Chapter 29 - Add a capacitor to the layout in Pcbnew](#)

[Chapter 30 - Controlling the track width](#)

[Chapter 31 - Calculate the appropriate track width](#)

[Chapter 32 - Adding copper fills](#)

[Part Five - Project 1: Fabrication](#)

[Chapter 33 - What is this part](#)

[Chapter 34 - Creating the Gerber files and uploading to fabricator](#)

[Chapter 35 - Adding a decorative graphic](#)

[Part Six - Project 2: a 7-segment display board](#)

[Chapter 36 - What is this part](#)

[Chapter 37 - Create the schematic with Eeschema](#)

[Chapter 38 - Create nets and labels](#)

[Chapter 39 - Hidden pins and the power flag](#)

[Chapter 40 - The data bus](#)

[Chapter 41 - The unconnected component](#)

[Chapter 42 - Component - footprint associations](#)

[Chapter 43 - Create a 2 layer PCB in Pcbnew](#)

[Chapter 44 - Control track widths with nets](#)

[Chapter 45 - Add GND and Vcc copper fills](#)

[Chapter 46 - Add text labels](#)

[Chapter 47 - Add a decorative graphic](#)

[Chapter 48 - Exporting Gerber files](#)

[Part Seven - Project 3: a full-SMD 16-LED board](#)

[Chapter 49 - What is this part](#)

[Chapter 50 - The circuit](#)

[Chapter 51 - Create the schematic in Eeschema](#)

[Chapter 52 - Schematic wiring, Part 1](#)

[Chapter 53 - Schematic wiring, Part 2](#)

[Chapter 54 - Associate components with footprints](#)

[Chapter 55 - Create the PCB in Pcbnew](#)

[Chapter 56 - Wiring in Pcbnew](#)

[Chapter 57 - Adding copper fills](#)

[Chapter 58 - Adding text labels and decorative graphics](#)

[Chapter 59 - Export the Gerber files](#)

[Part Eight - Conclusion](#)

[Chapter 60 - What's next?](#)

About this book

I wrote this book to help makers learn how to make their own printed circuit boards. No prior experience is assumed or needed. I only expect that you have an interest in electronics and you are at a stage in your learning journey where you wish to go beyond the breadboard.

The software that I use to demonstrate the various PCB design concepts and techniques is Kicad. Why Kicad? In short, because it is awesome! It is a PCB design software that is used by thousands of professional and hobbyist makers around the world. It does everything that a hobbyist would ever want to do with PCBs. It is open source, supported by an active dedicated community of programmers, and is sponsored by some of the world's most respected science and engineering institutions. It is also free!

Its technical features, its price, and the benefits of open source are the reasons that I believe that Kicad is the perfect PCB design software choice for any hobbyist maker.

Why write this book? I am glad you asked!

Kicad is also easy to learn, despite a reputation for the opposite. However, even though Kicad's own documentation is rich, it can be hard to follow and learn from, especially for beginners. By writing this book, I wanted to give people a resource that makes learning Kicad as easy as it is to actually use it. Based on three projects, you will progress through concepts and techniques of increasing complexity, but without the stress.

The progress is gradual and gentle, and designed to minimise the frustration often associated with learning complicated software. If you also enrol to the Kicad Like a Pro video course (more information about this is available further down), you will also have access to a discussion forum. You can use this forum to communicate with me and with other students of the course, and ask questions. Learning in a community is so much better!

Wishing you learning success!

Peter



Video course companion

This book is also available as a video course. The video course contains everything you see in this book, in 72 lectures and over 8 hours of high-definition video.

Watch me design the PCB of the 3 projects from the start to completion, including the PCB ordering process. Don't miss any details, replay, fast forward and reverse.

The screenshot shows a video player interface. On the left is a sidebar with a list of 15 items, including 'Hidden pins and Power Flags', 'Using busses to create readable schematics', 'The Unconnected component', 'Associate components with footprints', 'Create a 2-layer PCB in Pcbnew' (highlighted), 'Controlling wire width with nets', 'Another look at OpenGL', 'Copper fills for VCC and GND', 'Add text labels', 'Creating a graphic to decorate your PCB', 'Export the Gerber files (4-52)', 'A few weeks later, the PCB arrives!', and 'End of section quiz'. Below this is 'Project 3: a 16-led display with shift registers' with 'Section introduction' and 'Sneak preview: what will this board look like in the end?'. The main area shows a video titled 'Create a 2-layer PCB in Pcbnew' with a thumbnail of a PCB layout. Below the video, text says 'In this lecture, we will import the netlist file into Pcbnew and layout the board.' There are '0 Comments' and a 'Tech Explorations' tag. A user 'Peter Dalmaris' is listed. There are 'Recommend' and 'Share' buttons, and a 'Sort by Best' dropdown. A comment input field says 'Start the discussion...' and a note says 'Be the first to comment.'

Learn Kicad Like a Pro with my detailed video course.

By enrolling to the video course, you also gain access to the discussion forum so that you can interact with me and other students. Discuss Kicad topics, ask questions, offer ideas. Access to the video course and its discussion forum is permanent.

As a reader of this book, you can enrol to the video course on txplore.com for the special price of just \$20 by using coupon code ebook. This represents a 20% discount over the regular price. Or, simply click on [this link](#) to preview and purchase.

Discussion forum and email list

If you have enrolled to the video course, you also receive access to the course discussion forum. The discussion forum is a great resource to have when you are getting started. You can use the discussion forum to ask question, or exchange ideas with other students.

Here's a couple of examples of conversations from the Kicad Pro discussion forum:

Student asked a question in Lecture 29

Problem with redrawing board around capacitor

I have re-drawn the board several times and I get the following error with different endpoint values depending on the accuracy of my drawing in the 3D view:

IO_ERROR: Unable to find the next boundary segment with an endpoint of (160.02 mm, 127 mm).

Edit Edge.Cuts perimeter graphics, making them contiguous polygons each.

from /build/kicad-TRMME0/kicad-4.0.1/kicad/pcbnew/specctra.cpp : ThrowIOError()
: line 144

Unable to calculate the board outlines.

Therefore use the board boundary box.

Can I ignore the error and proceed?

Thanks in advance

Answer by Peter Dalmaris

Hi Jeff, yes, you can ignore this, and confirm that your design is valid with gerblook.org.

I know that you have tried to redraw the boundary a few times, so I'll just say that I also have this issue often when I try to make a change to an existing boundary. In most cases, I have to completely remove the old boundary, zoom in to make the lines thick, then redraw. It is annoying, but apart from the warning message in the 3-d view, it doesn't

cause problems in making the board.

Student asked a question

What to do when i don't have all devices library like a microcontrollers, capacitors and so on.

Hi peter,

what could i do when i don't have some libraries like a specific capacitor, microcontroller 8051 and so on. How can i download or to load new libraries devices to used.

regards.

RT

7 replies · Like · Following (3 followers) ·

Answer by Peter Dalmaris

Hi RT, there are two basic options:

1. Make your own library parts, schematic or footprints

2. Google for parts that other people have made and import them to your library. This is easy to do, but I will be adding a lecture on how to do this ASAP. I will try to do that this week.

Student reply

Thank you peter.

Peter Dalmaris

Hi, just an update, I'm uploading the lectures on third party libraries now to help you with option #2.

Student reply

Thank you so much !!! peter.....

is a great section.

regards.

RT

Student reply

Hi, I saw your post. Try this: <http://www.snapeda.com/parts/>

Peter Dalmaris

Hi, wow! Got to spend some time browsing snapeda!

Student reply

Excellent!!!! thank you.

Dedication

I wish to express a big THANK YOU to my wife Michelle and our children Leo and Ari!

I know that living with a geek writing a book is not easy. Correction: living with a geek in general is not easy!

Thank you, Michelle, for your understanding and for helping me complete yet one more project!

Living with the three of you is such a pleasure, and a day-in-day-out geek fest!

Copyright

Copyright © 2016 by Peter Dalmaris

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

First published in 2016.

Futureshock Enterprises, PO Box 22, Berowra, 2081, NSW, Australia

txplore.com

PART ONE
About this course

Chapter 1: *What is Kicad?*

KiCad is a tool that makes it possible to design high-quality printed circuit boards. It is not the only one, in fact there are many many other tools out there. Some are free, some are very expensive, and each tool has its advantages over others.

Software tools that help people make printed circuit boards are often referred to as “Electronic Automation Design” tools, or “EDA” for short.

If you want to make a printed circuit board, you don’t even have to use an EDA tool. You can get by using just use a special conductive ink pen and draw a circuit on a sheet of paper.



Or, you can use an Etch Resist Pen, something like a Sharpie pen or anything permanent and water resistant will work, and hand-draw a circuit on a copper board.

These manual methods are good for simple circuits, but for most practical applications they are far from ideal. So good design software and professional fabrication labs are the essential friend of the serious maker.

There are many EDA tools. Popular tools include free and open source ones like Fritzing, gEDA, KiCad, and FreePCB, and proprietary ones like Altium Designer, Design Spark PCB, and Eagle.

All of these tools can be used to create at least 2-layer PCBs. The proprietary ones typically include free versions that offer a limited set of features compared to their fully licensed versions. In some cases, EDA tools offer circuit simulation on top of design.

Let's focus on Kicad now.

KiCad is an open-source EDA tool that has been in development since 1992. By some accounts, it is the most popular EDA tool in the world. Many well known and a lot more unknown projects have been build with KiCad, which is a testament to the tool's professional-level capabilities.

KiCad is supported by a few important international organisations which contribute towards its future in development resources. CERN, the European Organization for Nuclear Research, is a core contributor to the project. The Raspberry Pi Foundation and Arduino LLC have also made contributions.

Why learn it?

The ability to create a custom PCB is a core skill of an electronics enthusiast. Without the ability to create a custom PCB, your breadboard-based projects are doomed to oblivion. Learning how to use KiCad will make you a better maker because you will be able to create custom PCBs for your best designs, and as a result you will ensure that they realise their full potential.

Learning KiCad over alternatives in particular makes good sense. KiCad is a well tested tool that has performed with excellence in many demanding projects. It's development is ongoing, with the support of many individuals and large organisations. KiCad is here to stay, so your the effort you expend in learning it will not be wasted.

And did I mention that KiCad is free? It is perhaps the only fully-featured and

unrestricted EDA tool that is free. This must be worth something!

Ok, let's spend a few minutes in the next lecture to explain how this course is structured, and then we'll get started with KiCad!

Chapter 2: *The structure of this course*

This course is designed to teach you how to use KiCad assuming no prior knowledge in PCB design. My objective is to help you reach a high level of competency. Even though you will not be able to design 16-layer super-dense and sophisticated PCBs without several years of experience and study of advanced topics in electronics and physics in addition to this course, you will be able to create any PCB that an Arduino or general electronics enthusiast can dream of.

To achieve this, I have organized this course around three projects. With each project, I introduce several new KiCad features and extend your skill set.

Project one is the simplest one, from a circuit point of view. Assuming that you have no prior experience with PCB design, this is also the hardest project because I use it to demonstrate the bulk of Kicad's features. The circuit it self is simple so that it does not get in your way of learning KiCad. It is a breakout for the nRF24 wireless module, familiar to many Arduino enthusiasts, but often painful to work with because it is available as a breadboard-unfriendly breakout. We will fix this problem by designing a breadboard-friendly single-layer board.

In the second project, we will look at a slightly more complicated circuit. The circuit is a seven-segment display driven by a shift register integrated circuit. This circuit will give us the opportunity to explore additional KiCad capabilities. We will create a two-layer PCB on which through-hole components will be mounted, and interconnect the layers with vias.

In the third project, we will reuse all of the skills and features from the first two project to create a more complicated PCB. The circuit you will build will contain 16 LEDs driven by two shift register ICs. All of the components on this PCB will be surface-mounted,

something that gives designers an advantage that helps in reducing the total space required for the board, improving the layout of the component.

If you are thinking of launching a Kickstarter campaign, then surface mounted components will also reduce the mass-manufacturing costs of your amazing gadget!

Towards the end of the book of the course, I have added chapters to show you how to import third-party libraries to your Kicad installation, and how to use hierarchical sheets to deal with large and complicated schematics.

PART TWO

Kicad basics

Chapter 3: *What is this part*

In this introductory section of the book, first I will help you setup KiCad on your computer and show you where to find help for when you need it.

Then, I will discuss the basic components and terminology of PCBs.

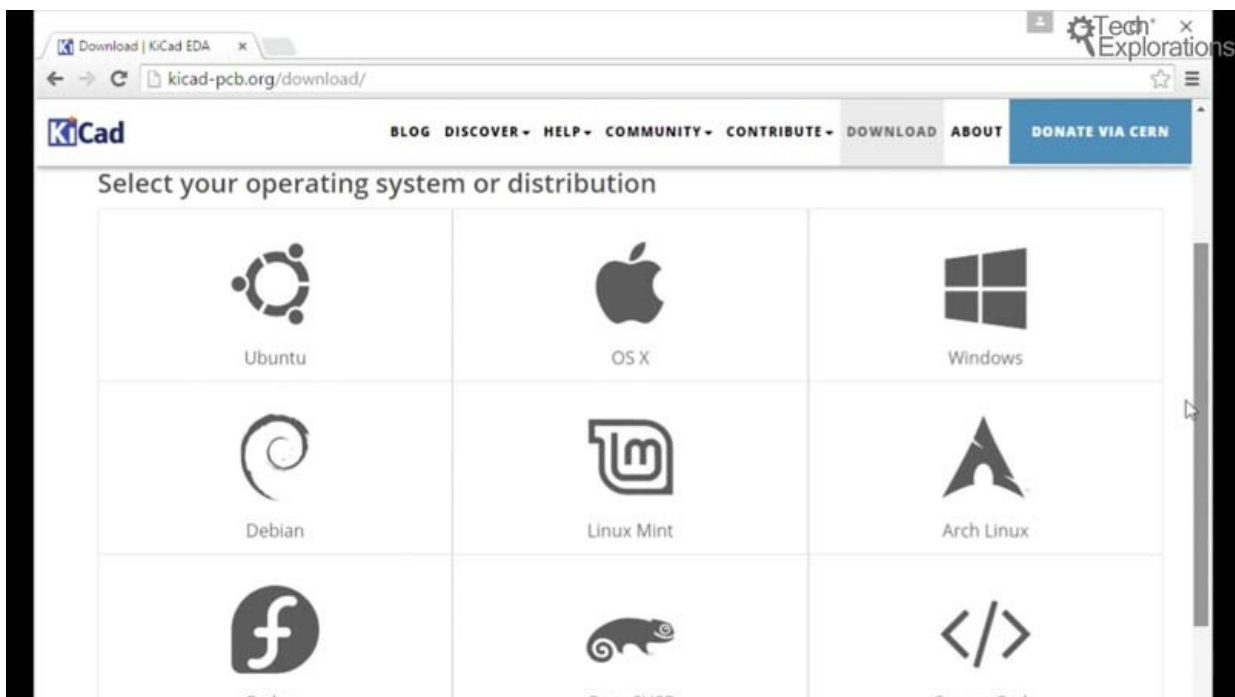
I will spend a few minutes talking about Kicad's PCB design process, which is worth watching even if you have experience in designing PCBs on other CAD systems because KiCad has a reputation of being different.

Finally, I will talk about fabrication and give you some examples of how and where you can have your PCBs manufactured.

Chapter 4: *Installation on Windows*

In this chapter, we'll install KiCad on Windows.

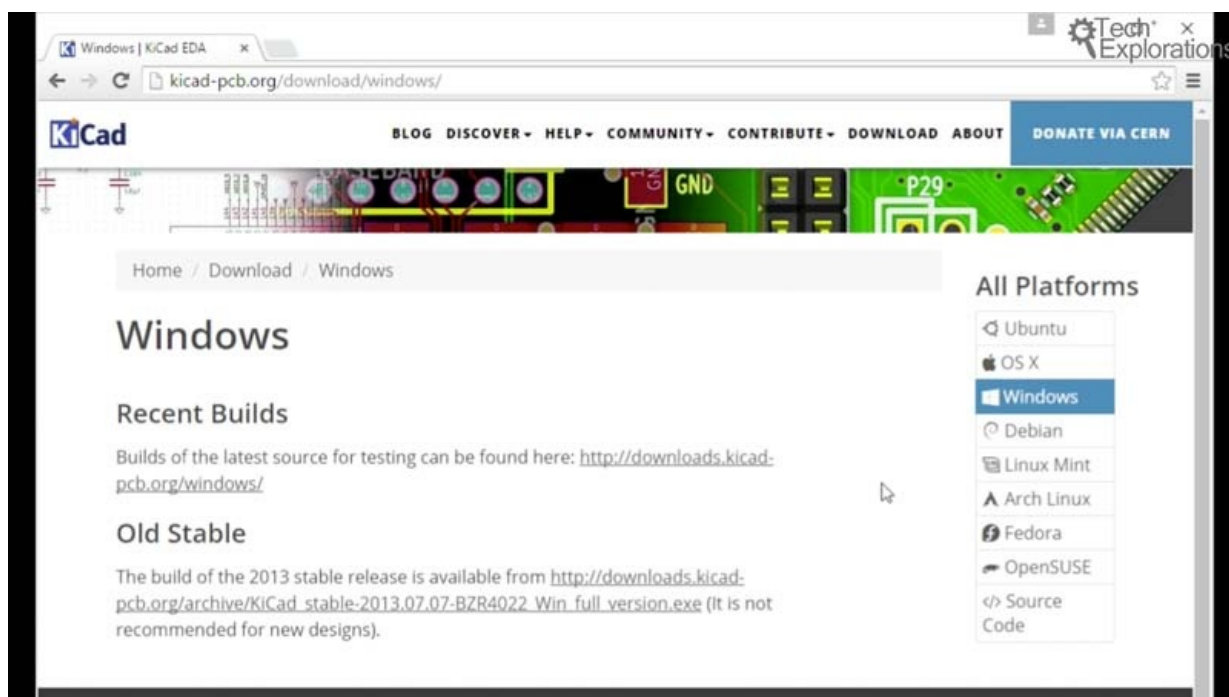
To do this, you will need to use your web browser and navigate to <http://kicad-pcb.org/download/>.



The main Kicad download page, at <http://kicad-pcb.org/download>

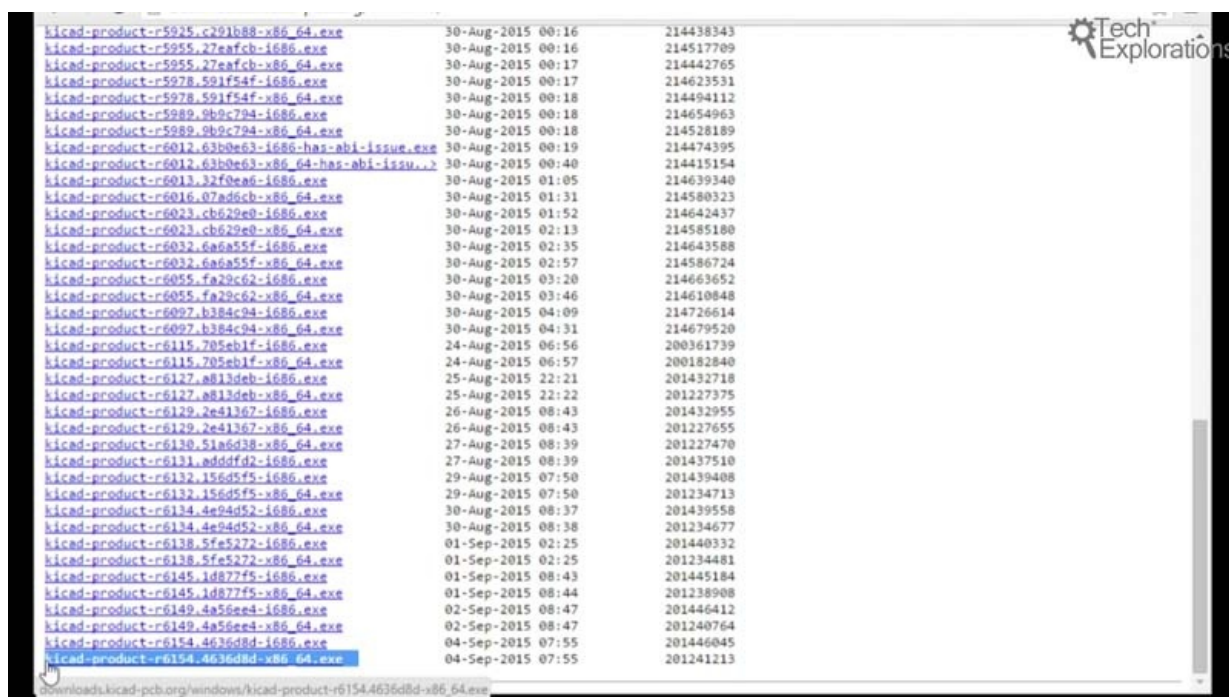
KiCad comes precompiled for a variety of operating systems, so all the major operating systems are supported. But if you prefer, you can even download the source code and compile it yourself. In this chapter, I'll be showing you how to install KiCad on Windows.

Navigate to the Windows branch by clicking on the Windows logo in the download page, at <http://kicad-pcb.org/download/windows>.



Download the Windows installation file from, at <http://kicad-pcb.org/download/windows>

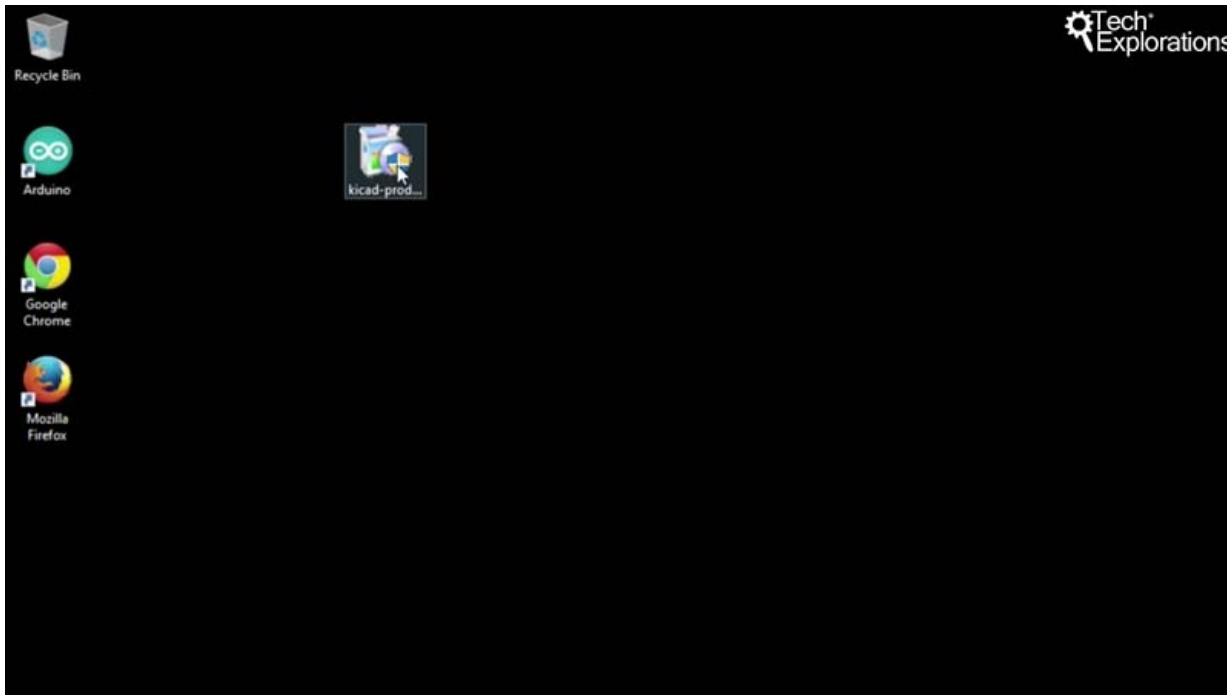
You have the option to download the most recent build which is preferable for any new designs that you make. Let's download the most recent build. The latest one is at the bottom of this list. I am using a 64bit version of Windows 10, so I'm going to go for the very last option to download. Choose yours based on your version of Windows.



Download the latest Windows installation file from, at <http://downloads.kicad-pcb.org/windows>

This download is 192 megabytes in size. I'm going to let this download complete, come back in a few minutes, and continue...

A few minutes later the download is complete. Find your installation file in your downloads folder. In this screenshot, you can see mine on the Desktop.



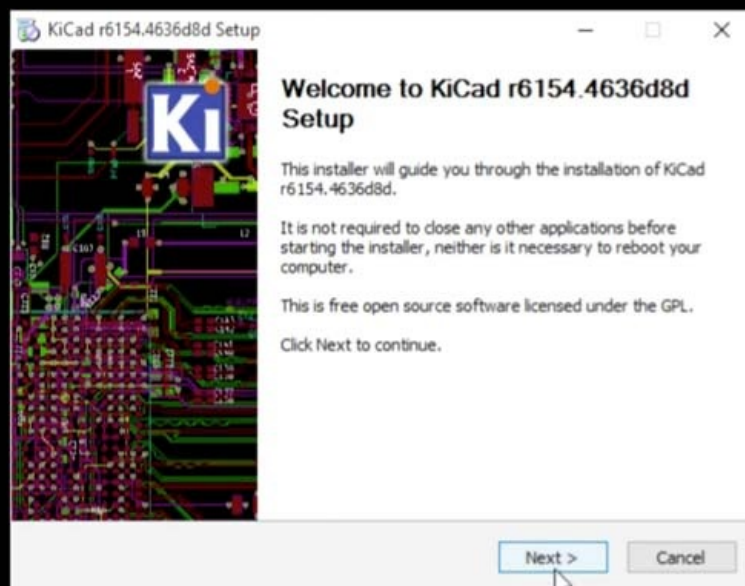
The Kicad Windows installation file on my desktop

Let's start the installation process. Double click on the file, and accept the prompt. Click "Yes" to proceed.



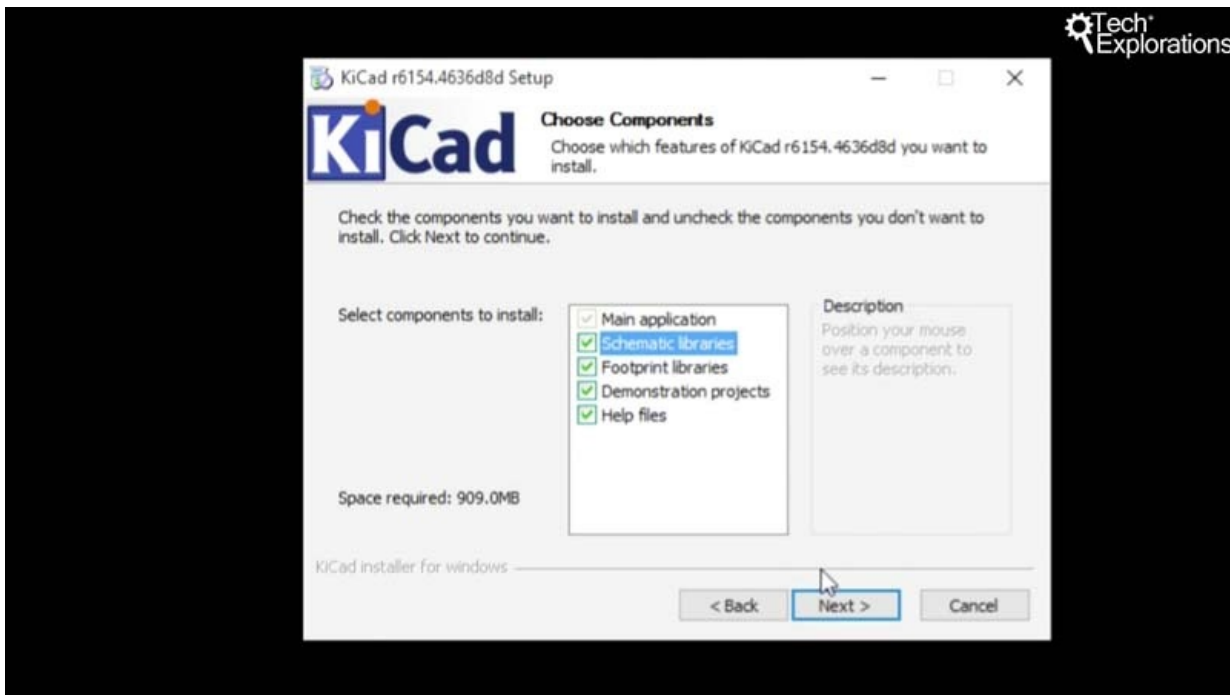
Accept the security prompt

Also accept the setup Welcome dialog box. Click on Next to continue.



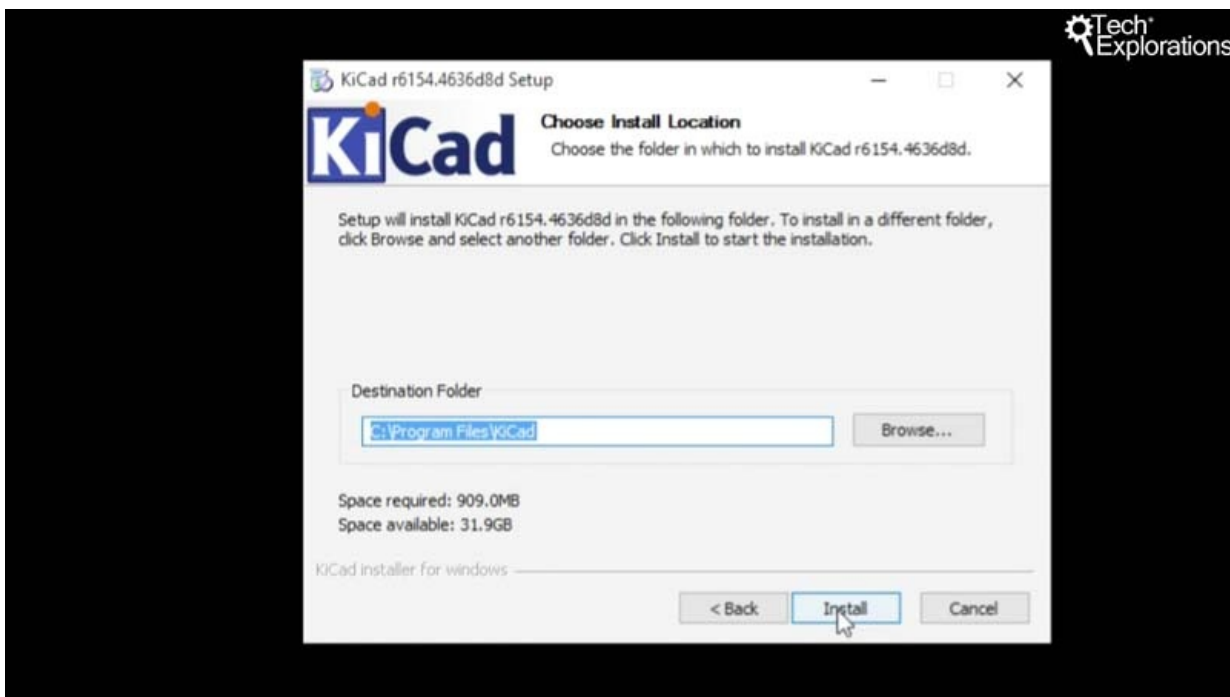
Accept the Setup Welcome dialog box.

In the component chooser, I recommend including all of the available components. Click on Next to continue.



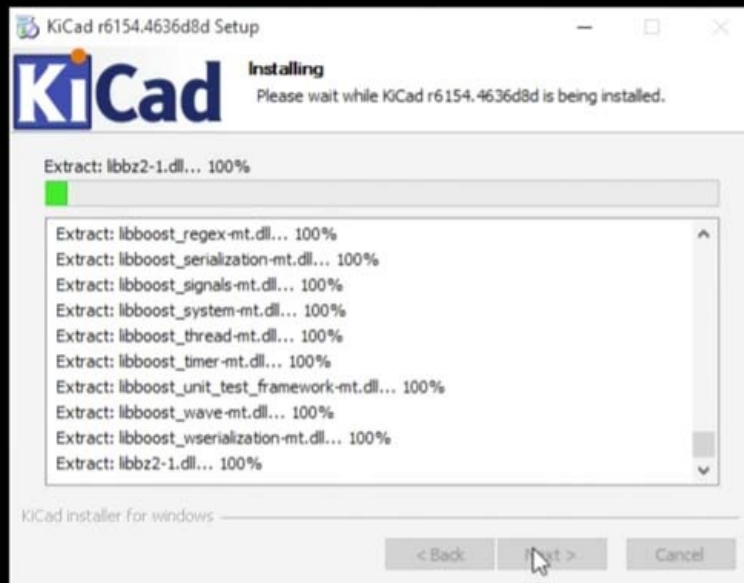
I recommend that you include all available components to your Kicad setup.

Next, the Setup Wizard will ask you for the installation location. You can simply accept the suggested location and click on Next.



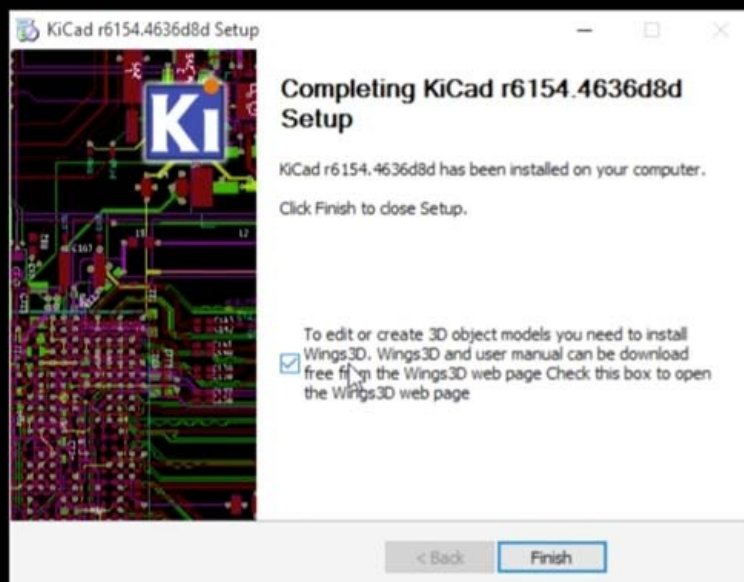
Specify the installation location. The default location is fine in most cases.

The installation process will begin, and the status information will be shown in the Installing window. Let it finish, and then click on Next to continue.



The installation process is shown in the Installing window

Even though I am not covering Wings3D in this book, I think you should still install this feature. With Wings3D you can create 3D object models for your designs. It is a nice way to visualize your design will look like eventually before you have manufactured it and installed all the components on it. Check the box and click Finish.



Wings3D allows you to visualise your designs in 3D.

All right, so KiCad installation is complete.



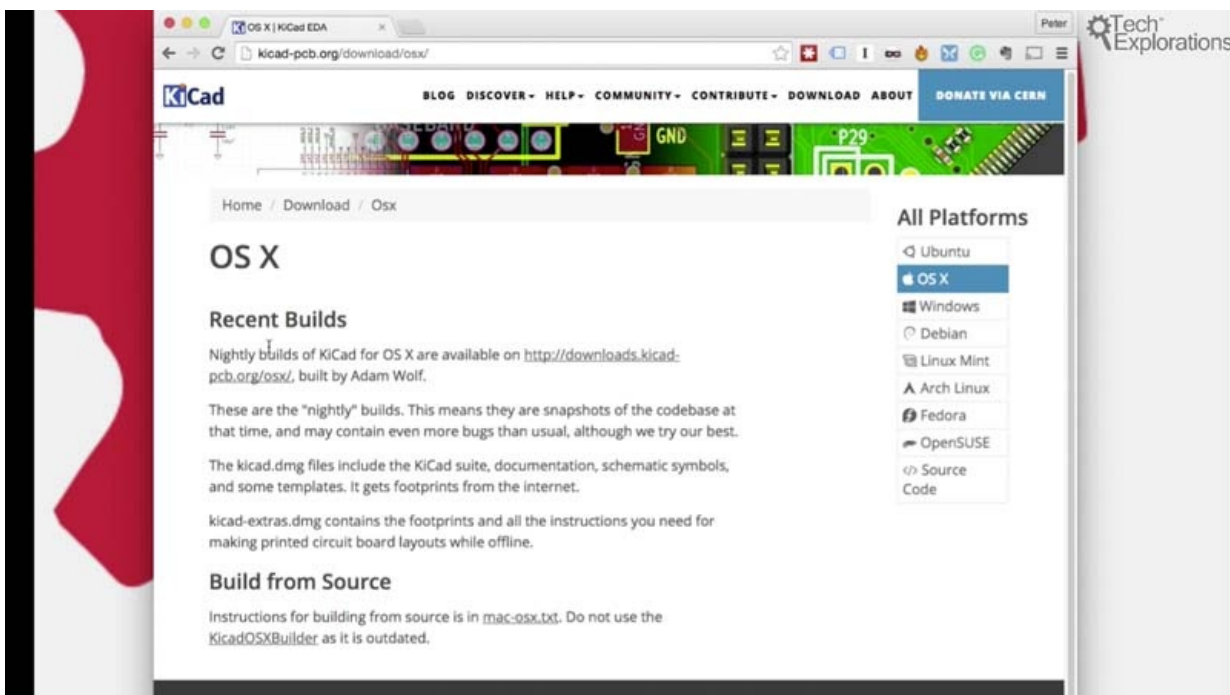
The Kicad shortcut on my desktop

In the next chapter I'll explain how to install Kicad on Mac OS X. If this is not something that interests you, feel free to skip it and go straight to the chapter titled "Kicad main components", where I'll give you an overview of the most important components that come with KiCad.

Chapter 5: *Installation on Mac OS X*

In this chapter, we'll install KiCad on the Mac. There is KiCad support for many other operating systems as well, including downloading the source code and compiling from the source code if you know how to do that. Here, we'll keep it simple and install the already compiled version.

Let's go into the Mac OS X branch, and there are two types of downloads that you can do; the recent builds and you can build from source.



Download the Kicad installation files for Mac OS X from <http://kicad-pcb.org/download/osx>

As far as the recent build is concerned, there's a slight difference to how this works for

Windows. There are two big files you can download. The first one is *KiCad.dmg* and the second one is *KiCad-extras.dmg*. The first file contains KiCad itself, documentation, the schematic symbols, and some templates, but it doesn't contain any footprints. So you'll be getting the footprints from the Kicad GitHub repository. To do that, you obviously need a live Internet connection. If you find yourself working offline, like traveling on a train or a bus, like I find myself sometimes doing, then you will also need to download *KiCad-extras.dmg* and this contains the footprints that you can use offline.



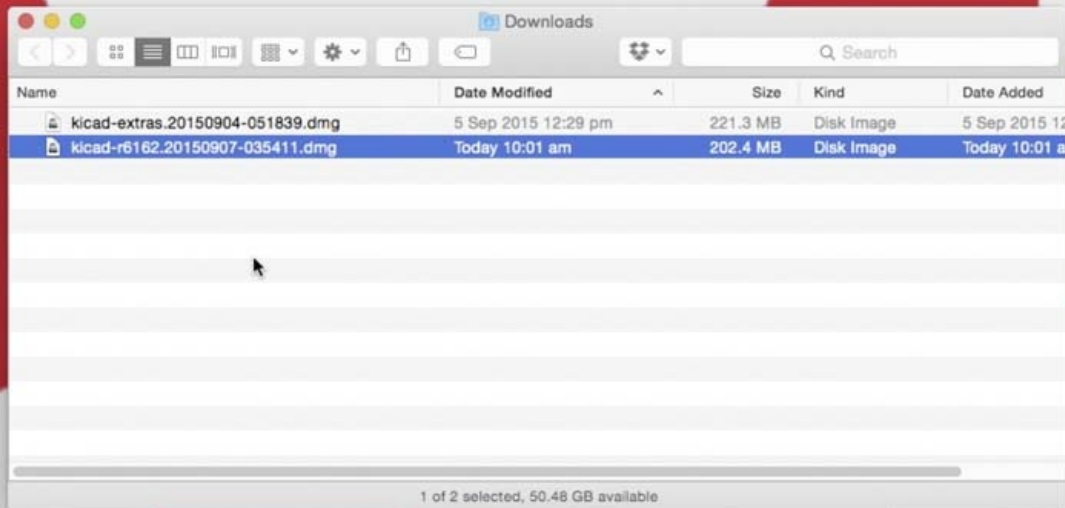
..../	21-Jul-2015 19:00	-
DEBUG/	21-Feb-2015 11:00	494
README	20-Aug-2015 19:50	221152663
kicad-extras.20150821-035254.dmg	21-Aug-2015 19:50	221071291
kicad-extras.20150822-035324.dmg	22-Aug-2015 21:16	221090771
kicad-extras.20150823-051623.dmg	23-Aug-2015 19:51	221076992
kicad-extras.20150824-035351.dmg	25-Aug-2015 19:51	221077759
kicad-extras.20150826-035404.dmg	26-Aug-2015 21:14	221075591
kicad-extras.20150827-051716.dmg	27-Aug-2015 19:51	221077450
kicad-extras.20150828-035410.dmg	28-Aug-2015 21:15	220972466
kicad-extras.20150829-051722.dmg	29-Aug-2015 21:14	221069054
kicad-extras.20150830-051703.dmg	30-Aug-2015 19:51	221190765
kicad-extras.20150831-035416.dmg	31-Aug-2015 19:51	221308540
kicad-extras.20150901-035411.dmg	01-Sep-2015 19:51	221213423
kicad-extras.20150902-035438.dmg	02-Sep-2015 19:52	221282002
kicad-extras.20150903-035428.dmg	03-Sep-2015 21:15	221284738
kicad-extras.20150904-051839.dmg	20-Aug-2015 19:58	202313032
kicad-r6109.20150821-035147.dmg	21-Aug-2015 19:58	202296965
kicad-r6112.20150822-035219.dmg	22-Aug-2015 21:25	202297328
kicad-r6112.20150823-051519.dmg	23-Aug-2015 20:00	202299346
kicad-r6117.20150824-035244.dmg	25-Aug-2015 20:00	202376915
kicad-r6127.20150826-035253.dmg	26-Aug-2015 21:22	202301320
kicad-r6130.20150827-051609.dmg	27-Aug-2015 19:59	202318391
kicad-r6131.20150828-035306.dmg	28-Aug-2015 21:23	202281524
kicad-r6132.20150829-051616.dmg	29-Aug-2015 21:22	202271547
kicad-r6132.20150830-051556.dmg	30-Aug-2015 19:59	202358074
kicad-r6135.20150831-035312.dmg	31-Aug-2015 19:59	202357172
kicad-r6138.20150901-035306.dmg	01-Sep-2015 20:00	202367567
kicad-r6146.20150902-035330.dmg	02-Sep-2015 20:01	202356684
kicad-r6149.20150903-035321.dmg	03-Sep-2015 21:24	202404442
kicad-r6154.20150904-051736.dmg		

Download the files for Max OS X from <http://downloads.kicad-pcb.org/osx/>

Let's download both files. Go to <http://downloads.kicad-pcb.org/osx/> and scroll to the bottom. This is where you will find the latest version of the installers. At the time I wrote this book, the latest version was KiCad r6154, so I downloaded the file titled "*kicad-r6154.20150904-051736.dmg*". You should download the latest version available at the time you read this.

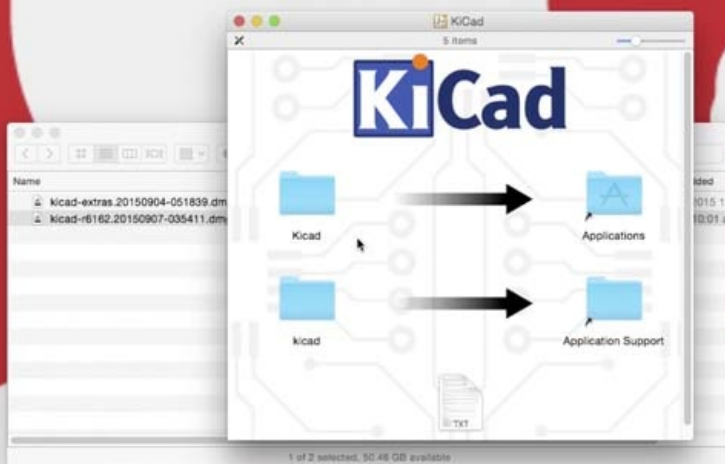
Next, download the extras installer. At the time of writing this, the latest extras installer was the file titled "*kicad-extras.20150904-051839.dmg*".

When the download is finished, you will have two files.



The installation files for Kicad on Mac OS X

You should have the main KiCad installation file, and the extras file. I will not install the extras now, prefer to access the GitHub repository for all the library files and just be able that way to get the latest. If at some point, I'll be doing a lot of design work offline, then I will go ahead and install this file as well.

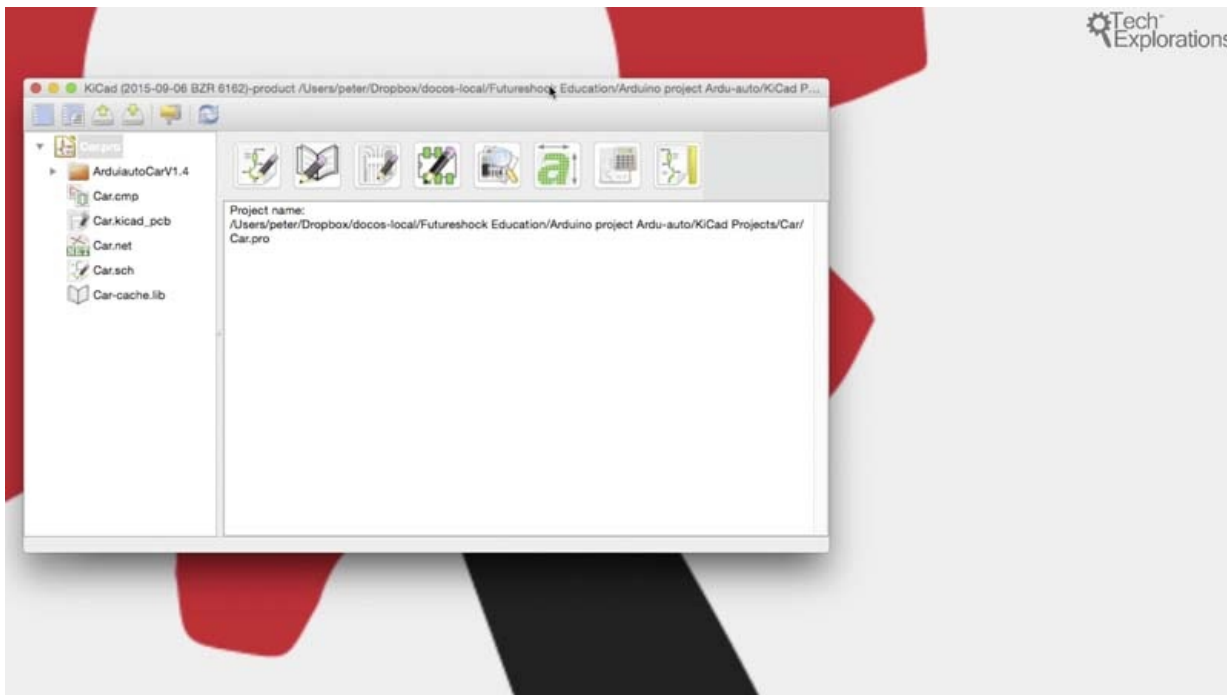


The installation archive contains two folders that must be copied to two locations

To install Kicad, we follow a process familiar to Mac users. Double click on the “dmg” archive. In the archive, there are two things that you need to copy. First, the KiCad folder goes into the applications folder, and then the other KiCad folder into Applications Support. Drag and drop as the arrows show, one at a time. Copying into the Application

Support directory will require authentication. That's all you have to do, Kicad is now installed on your Mac!

Let's start KiCad. You can use Spotlight Search to find Kicad, or just look for the Kicad icon in the Applications folder. Double click on the icon to launch the application. It looks exactly same as its Windows and Linux counterpart.



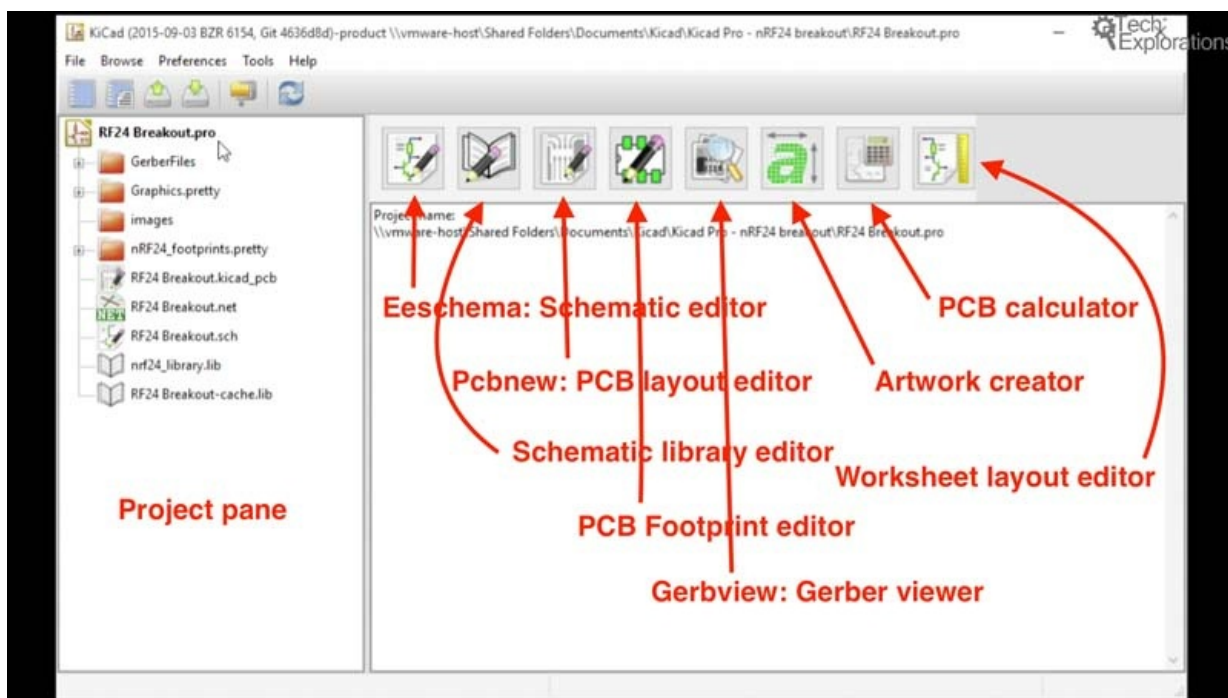
Kicad on the Mac looks identical to Kicad on Windows or Linux

For the remainder of this book, I will be using the Windows version of KiCad, since I'm more used to it. But you'll be able to follow along regardless of whether using a Mac, Windows, or Linux version of KiCad. All the features are the same across platforms. Just remember that you will get the best experience with a two-button mouse instead of the Apple Magic Mouse, and to use the Command key when I make a reference to the Control key on Windows.

Chapter 6: *Kicad main components*

Now that you have KiCad installed, lets go inside and have a look at its main components.

Double click on the KiCad icon to start the program.



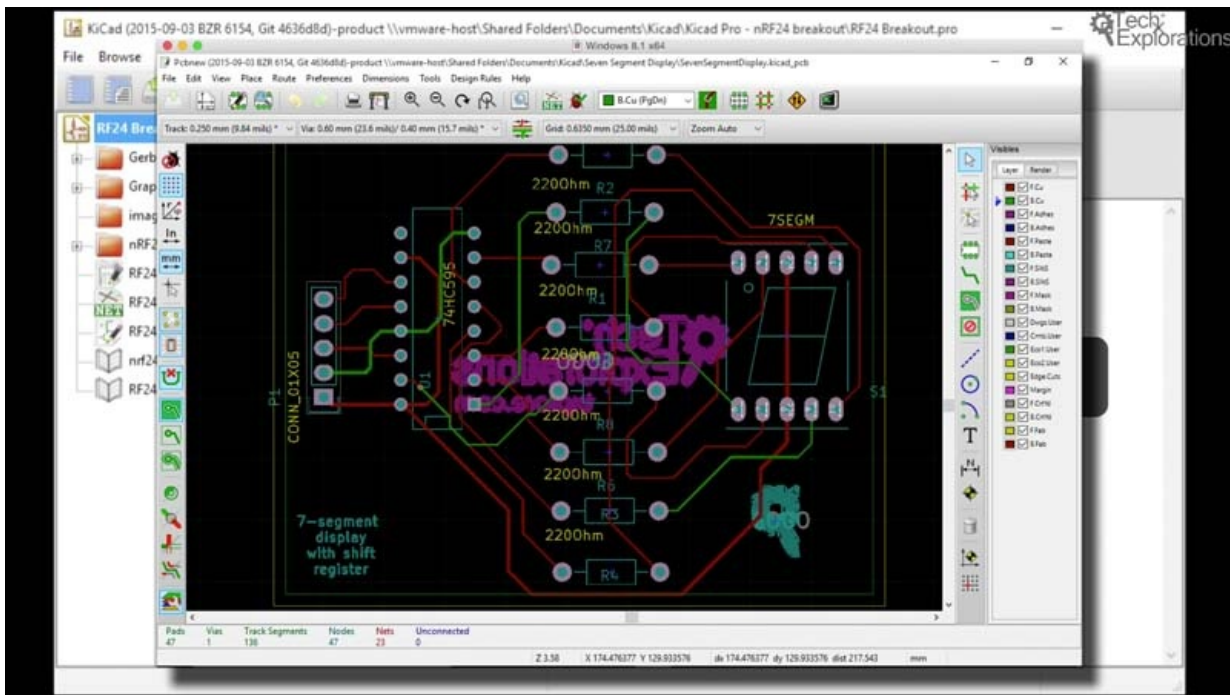
Kicad and its components

This is the main window in KiCad. It gives you access to the various applications that come with it. On the left side, the left pane, you will find the project pane. The project pan contains the files and directories that make up your project. If you have just installed KiCad, then this area will be empty.

Now, let's concentrate on the eight large buttons on the right side of the project pane. These are the buttons that give you access to the applications that come with KiCad.

The first one is the schematic editor, EESchema. This allows you to create schematics

The next most important application that is part of KiCad is PcbNew. PcbNew is the

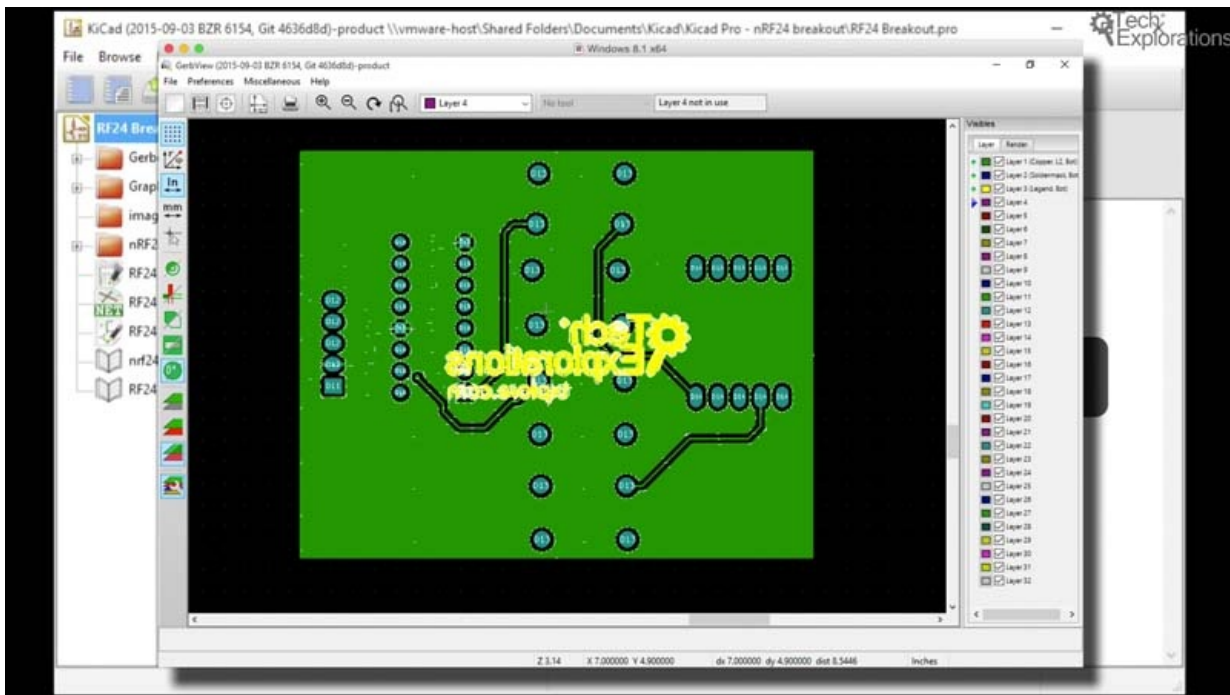


A view of Pcbnew, the Kicad layout editor

There's a few others that are supporting, for example this one here, it's the schematic Library Editor. You can use the Library Editor to manage component libraries. You can create your own components, or import libraries created by other people. Components are the parts representing real life components that you insert in a schematic design.

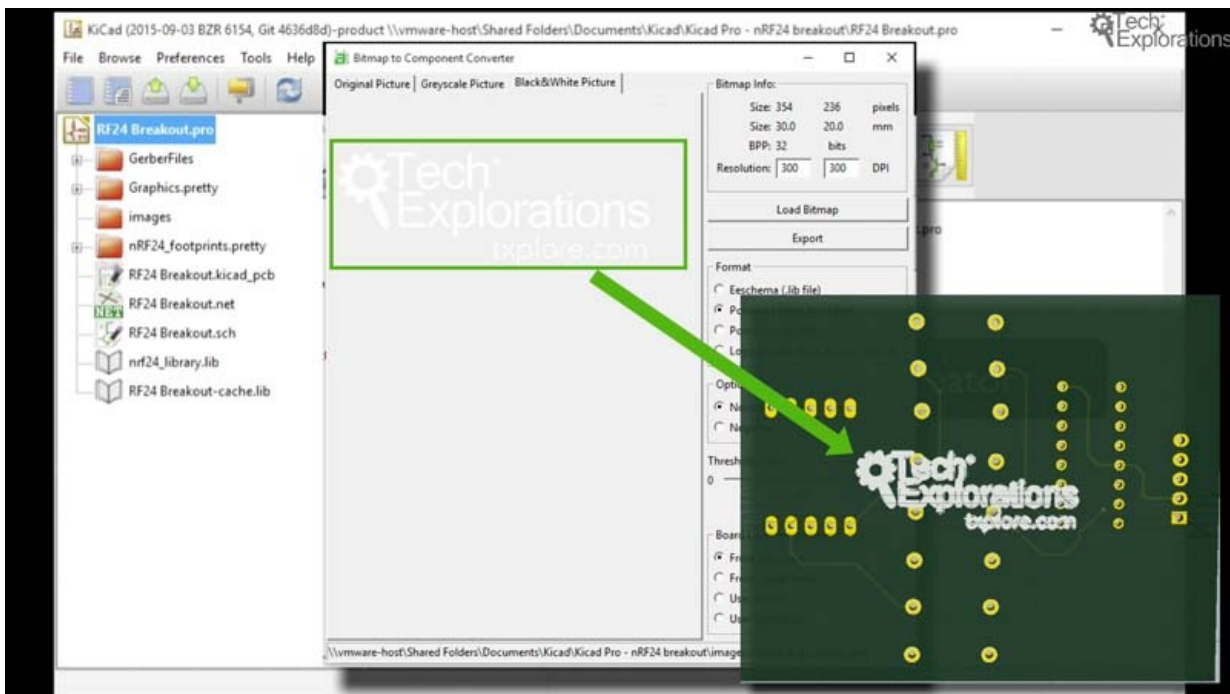
The PCB Footprint Editor is another helpful application that allows you to create footprints that you can't find in the footprint library. Or, perhaps you have a component with a footprint that exists in the Footprint Editor and you'd like to make a small change to it, so again you'll be using the PCB Footprint Editor application to do that.

There is this application called GerbView. GerbView is an application that gives you a view of a set of files that are exported from Pcbnew. These are the files that you will give to the PCB manufacturer to make the actual PCB. Before you do that, you'd like to inspect these exported files visually. To do that you'll GerbView. I'll show you how that works later plus we'll also going to be using a third party Gerber file viewer that I find very useful because it also checks for errors in your Gerber files.



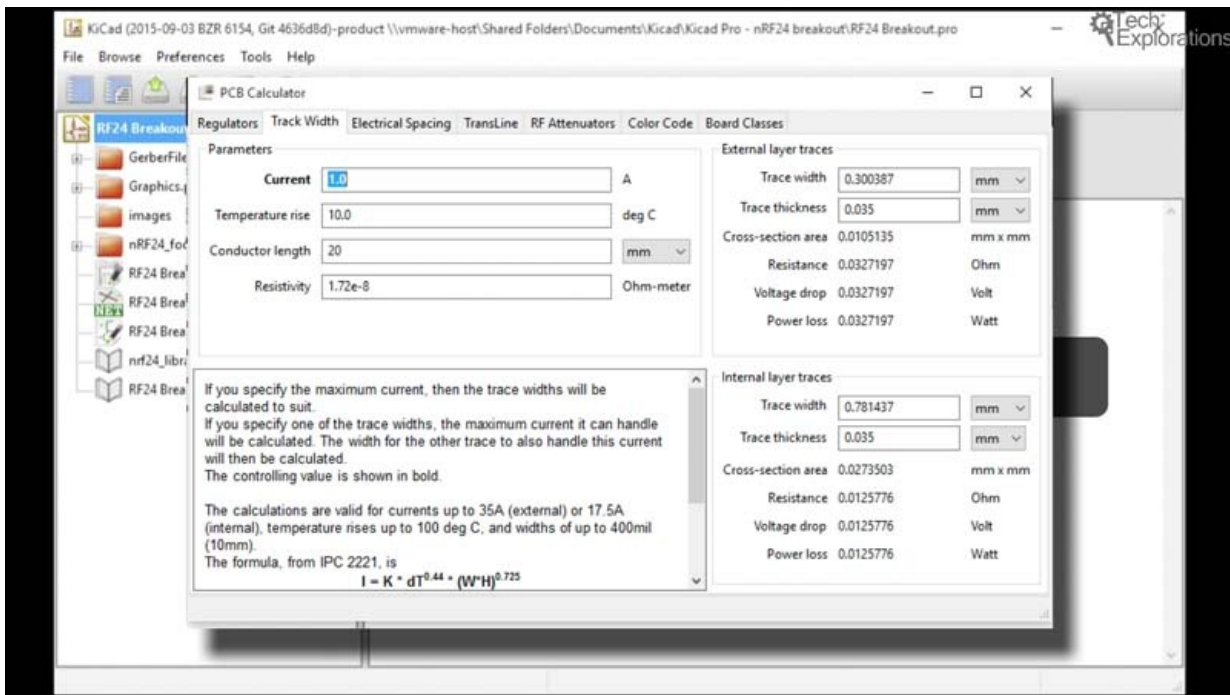
A view of GerbView, the build-in Gerber file viewer

Next, there is a little application called Bitmap 2 Component. This allows you to convert an image into a footprint so that you can put it on your PCB. Let's say for example, that you have a nice image like a logo for your company perhaps, or even a graphic that you like to put on your PCBs. To do this, you need to convert the graphic into a footprint using Bitmap 2 Component.



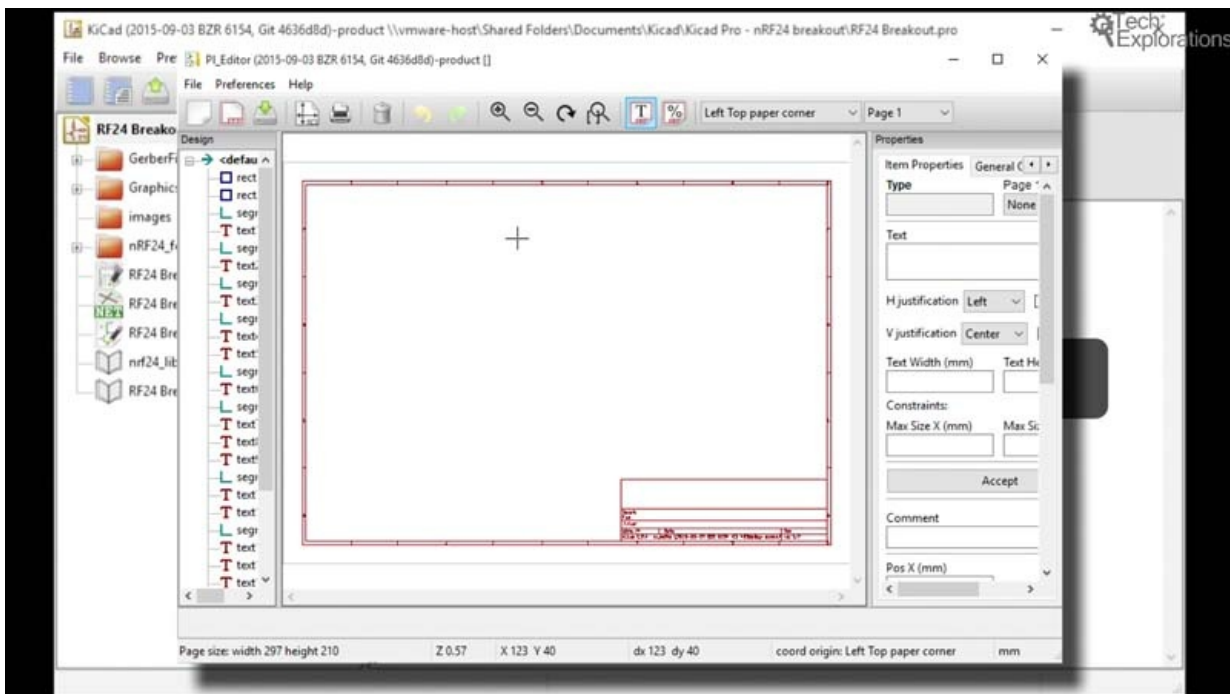
Bitmap2Component allows you to convert a graphic into a decorative footprint

Next, we've got a bunch of calculators in a small application called PCB Calculator. We will use the PCB Calculator to calculate the width of a track for power tracks and it can also be used to do a bunch of other calculations. It's a very handy tool.



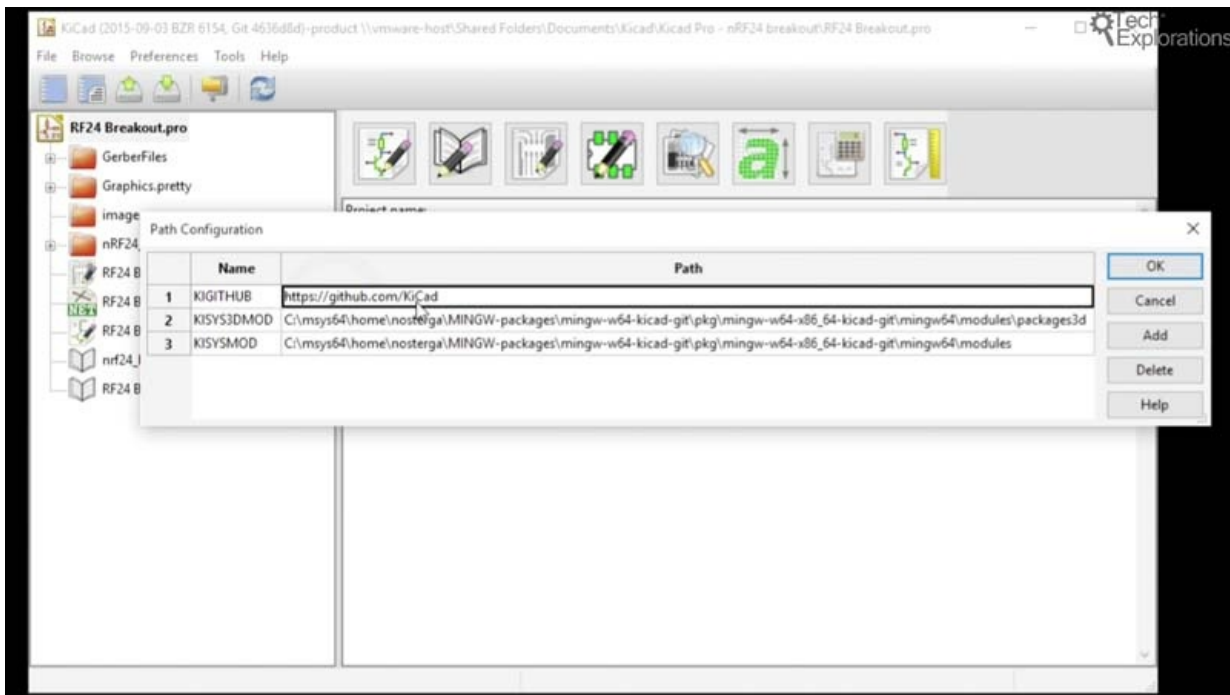
The PCB calculator contains various calculators, including one for calculating optimal track widths

Finally, we've got the Pi Editor, which is the worksheet layout editor. This allows you to make changes to the way that the schematic worksheet looks like. I'm not going to be using this tool at all because I like the schematic worksheet the way it is anyway but it is here if you need to use it.



With the PI Editor, you can change the information section of the Schematic sheet

These are the main components. There's a few other things that are good to know. For example, from the Preferences menu you can access the Kicad Paths editor, from where you can configure paths.



In the Kicad Path Configuration window you can specify the path to the online repository for footprints and schematics, among other things.

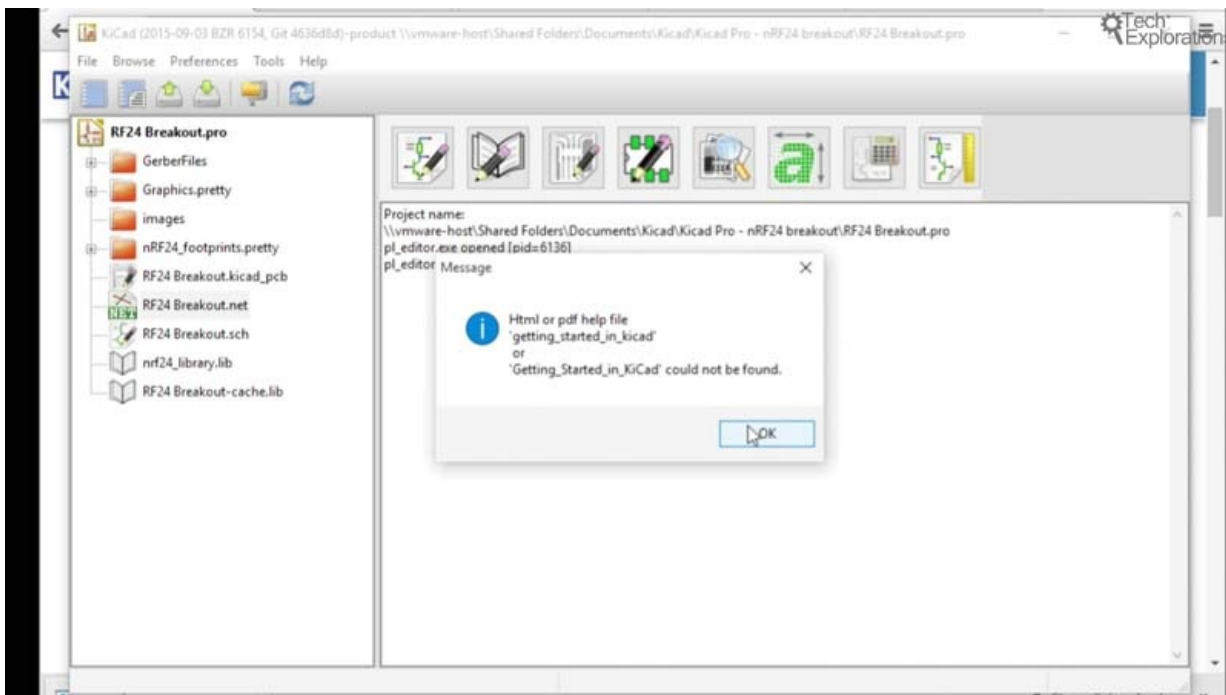
You can see that we've got access to [GitHub.com/KiCad](https://github.com/KiCad). This is the online location where footprints and schematic components are stored and where your Kicad installation will retrieve them from. You can, of course, choose to have these libraries installed on your local machine, however you will then need to update them manually as they will eventually be out of date.

In the next chapter we will talk a little bit about documentation and where you can go for help if you need it.

Chapter 7: *Finding documentation*

By some measures Kicad is the most popular PCB computer-aided design (CAD) open source software in the world. As a result it is no accident that it's got some of the best documentation out there.

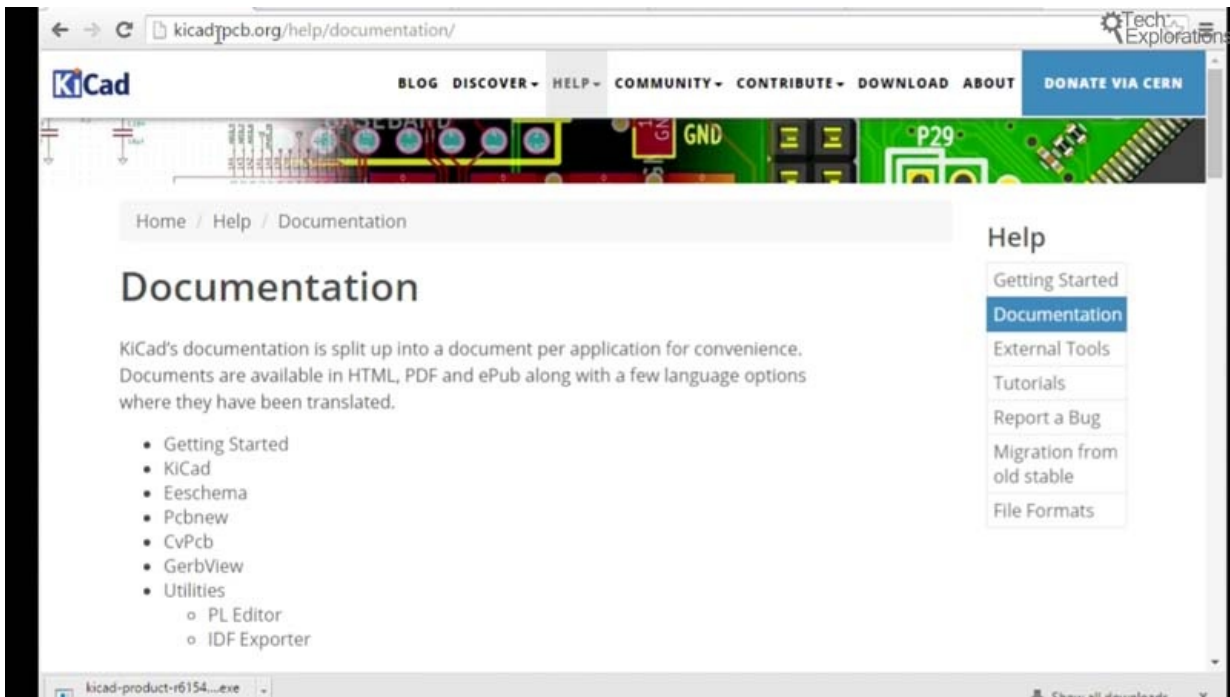
If you look for the documentation in the help menu in the main Kicad window, will receive a message that the help file was not found.



The documentation is not installed by default

The documentation wasn't installed. And same thing happens when getting help. These resources don't come by default with Kicad and I actually rather go to the web to find help instead of relying to the built-in help system that comes with the application.

To get help go to main Kicad website or <http://kicad-pcb.org/help/documentation>.

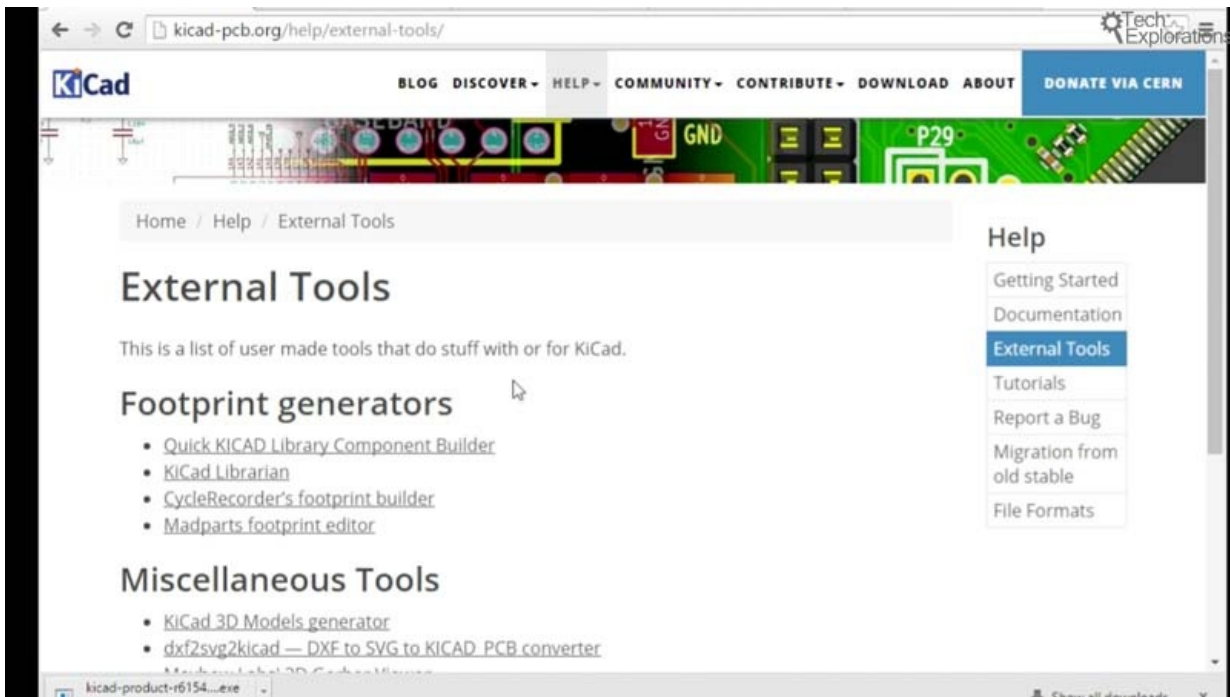


The on-line documentation is at <http://kicad-pcb.org/help/documentation>

In this locations, there are several resources that you can access. There is a getting started guide, which comes as a PDF document that you can download. There is also documentation that can help you with the individual applications that come with Kicad, and is available in several languages. There's documentation for EEschema and PCB new etc.

Some of the documentation in these resources may be a little bit old since the Kicad developers release new versions and updates frequently. Therefore, keep in mind that some of the documentation maybe lagging behind the software, especially for newly released features. Having said that, the main features in Kicad don't change much between releases, so the documentation for these features is mostly valid.

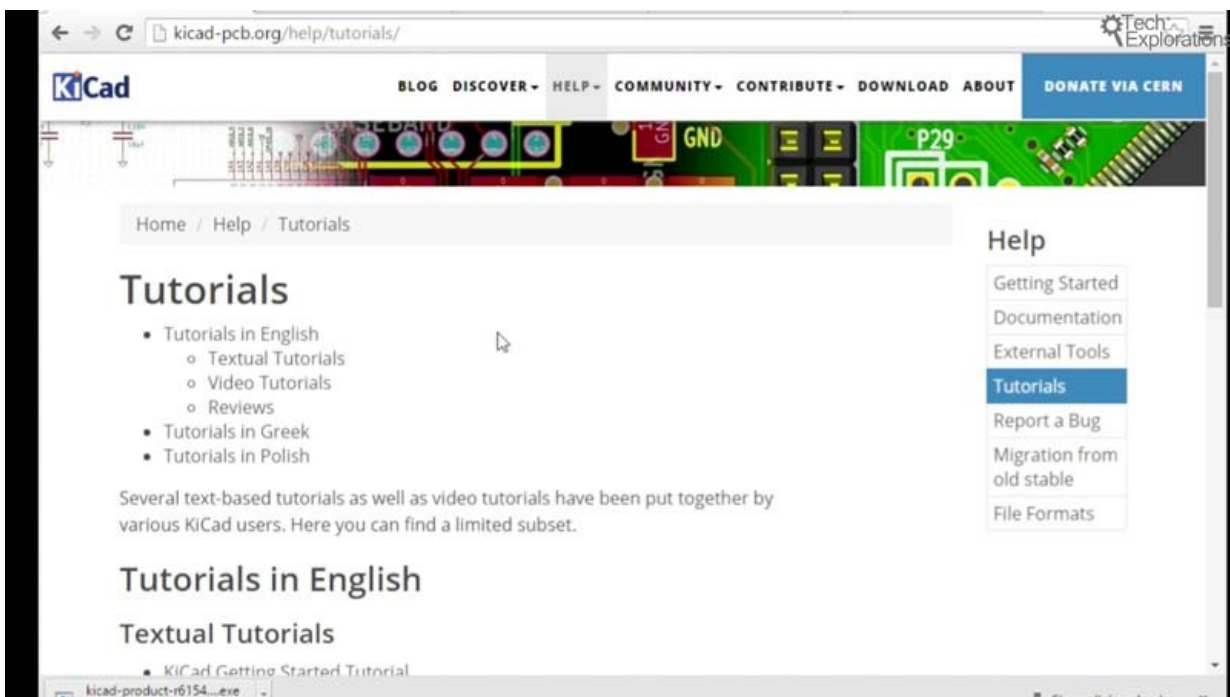
We also have external tools – so going into that – and you can see various editors and libraries and footprint, builders and 3D model generators and so on.



External tools documentation at <http://kicad-pcb.org/help/external-tools/>

These are things that you will become interested in due course, as you become used to the work flow in Kicad and familiarise yourself with the features of Kicad. Once you start looking for more advanced capabilities remember to visit this page.

You can also look for tutorials. Kicad users have documented their experiences in the form of tutorials that you can learn from.

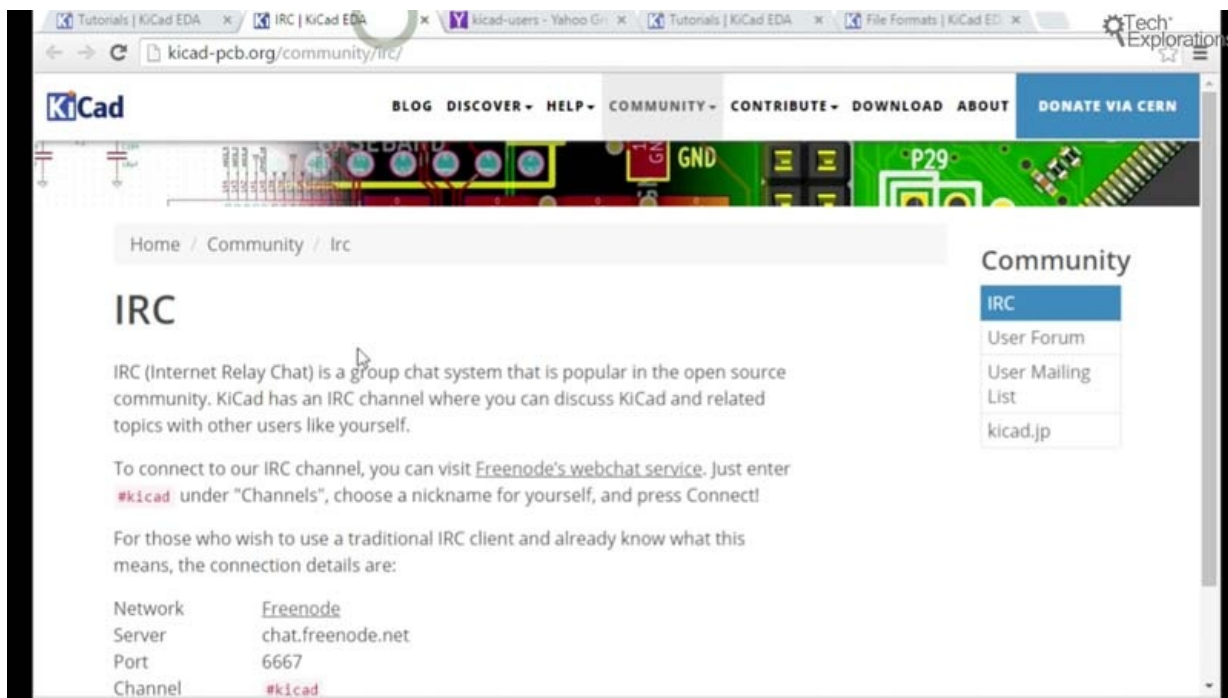


Tutorials, available at <http://kicad-pcb.org/help/tutorials>

So, apart from this book, these are great resources for you to keep in mind.

If you are looking for almost real-time help, it's good to know that there is an Internet

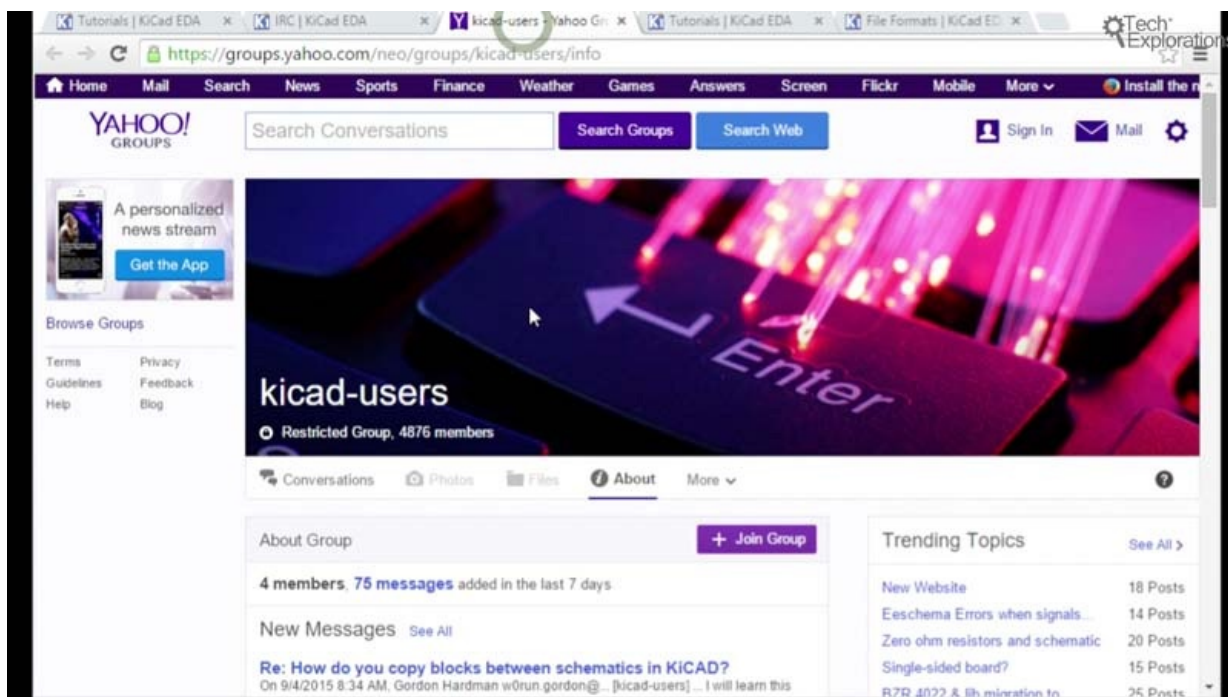
Relay Chat (IRC) channel, frequented by very experienced Kicad users.



An IRC channel is available at <http://kicad-pcb.org/community/irc/>

This IRC channel is very useful in case you have a question and need a quick answer. There's always somebody to talk to in this channel, who can help you out.

There is a fairly large Kicad users group on Yahoo, counting almost 5,000 members. To join it you must make an application, and you will be allowed access.

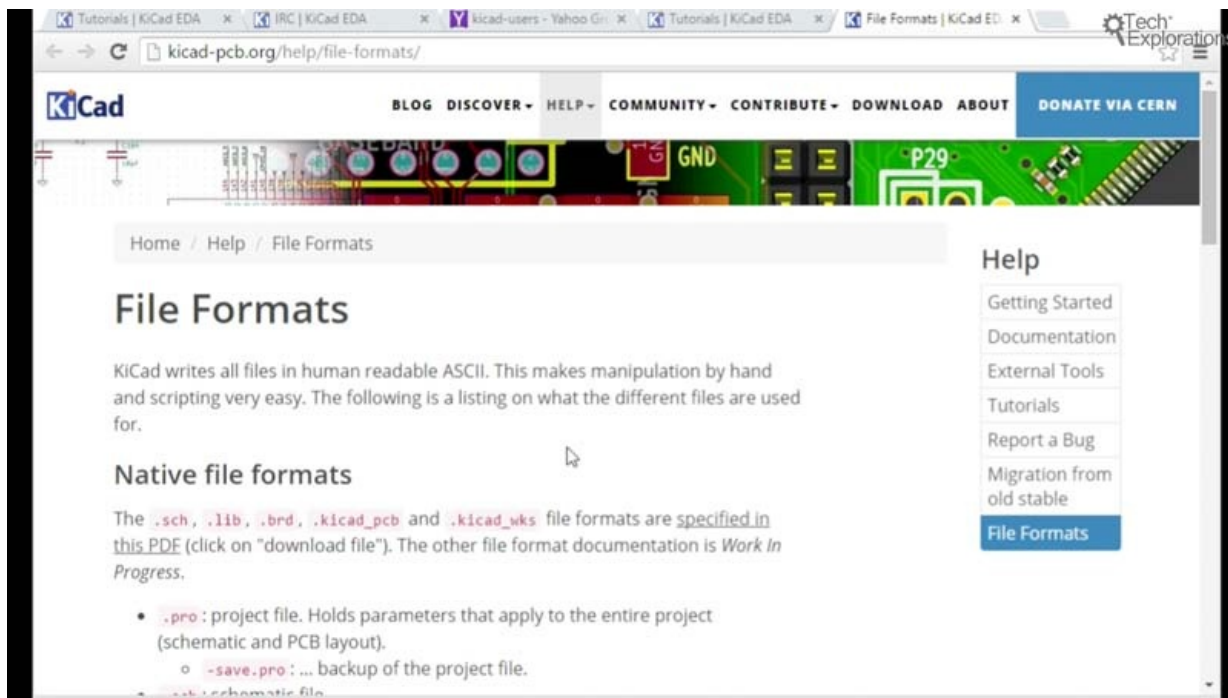


The Kicad Yahoo group is available at <http://groups.yahoo.com/neo/groups/kicad-users/>

This group contains a lot of very interesting conversations that can help you with a lot

of problems.

Another very nice resource is the documentation for the various file formats that Kicad uses. As you had a glimpse of earlier back in the main window of Kicad, you see that there are a lot of files that are exported and created by Kicad and you will need to figure out what each one of those files is eventually.



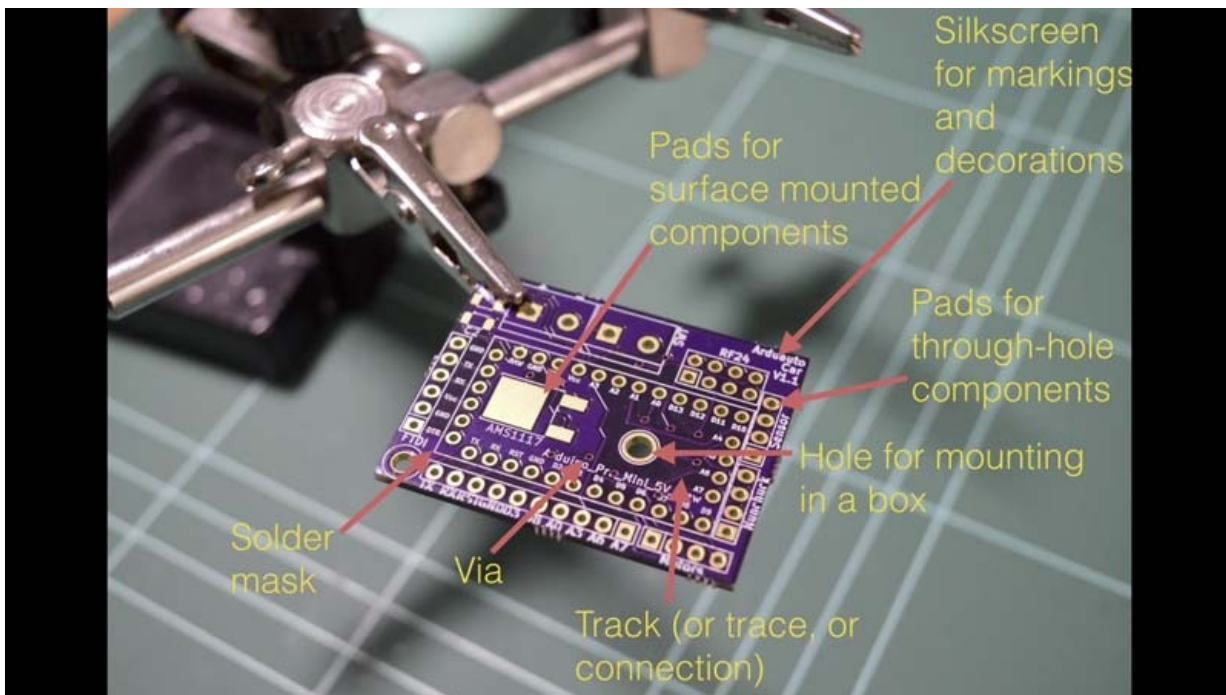
Documentation for Kicad's file formats is available at <http://kicad-pcb.org/help/file-formats/>

Here you will learn, for example that a “.pro” file is the project file that contains overall project information. There are schematic “.sch” files, “.lib” library files, “.mod” for footprints and so on.

In the next chapter we will have a look at the printed circuit board, and discuss what a PCB is, it's features and components.

Chapter 8: *What is a Printed Circuit Board?*

Let's have a look at the components of a PCB and what a PCB looks like and the terminology that we use. So let's all have a look at a PCB I made earlier. Here it is:

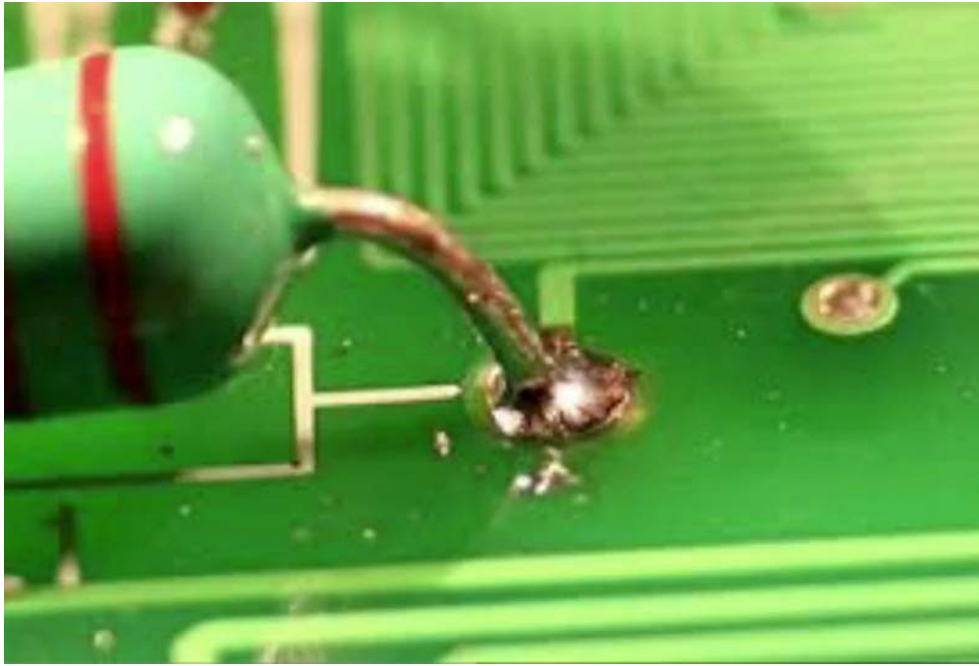


The top side of a PCB

I made this one at oshpark.com. The top side of the PCB is the side where we place the components. You can place components on the bottom side too, however this is unusual.

In general, we've got two kinds of components: through hole or surface mounted components. Through hole components, are attached on the PCB via inserting the leads or the pins through small holes. In the example pictured, we've got lots of holes into which the through-hole component pins are inserted. The holes extends from the topside to the bottom side of the PCB with, and are plated with a conductive material, like tin, or in this

case, gold. We use solder to attach and secure a component through its lead onto the pad that is surrounding the hole.



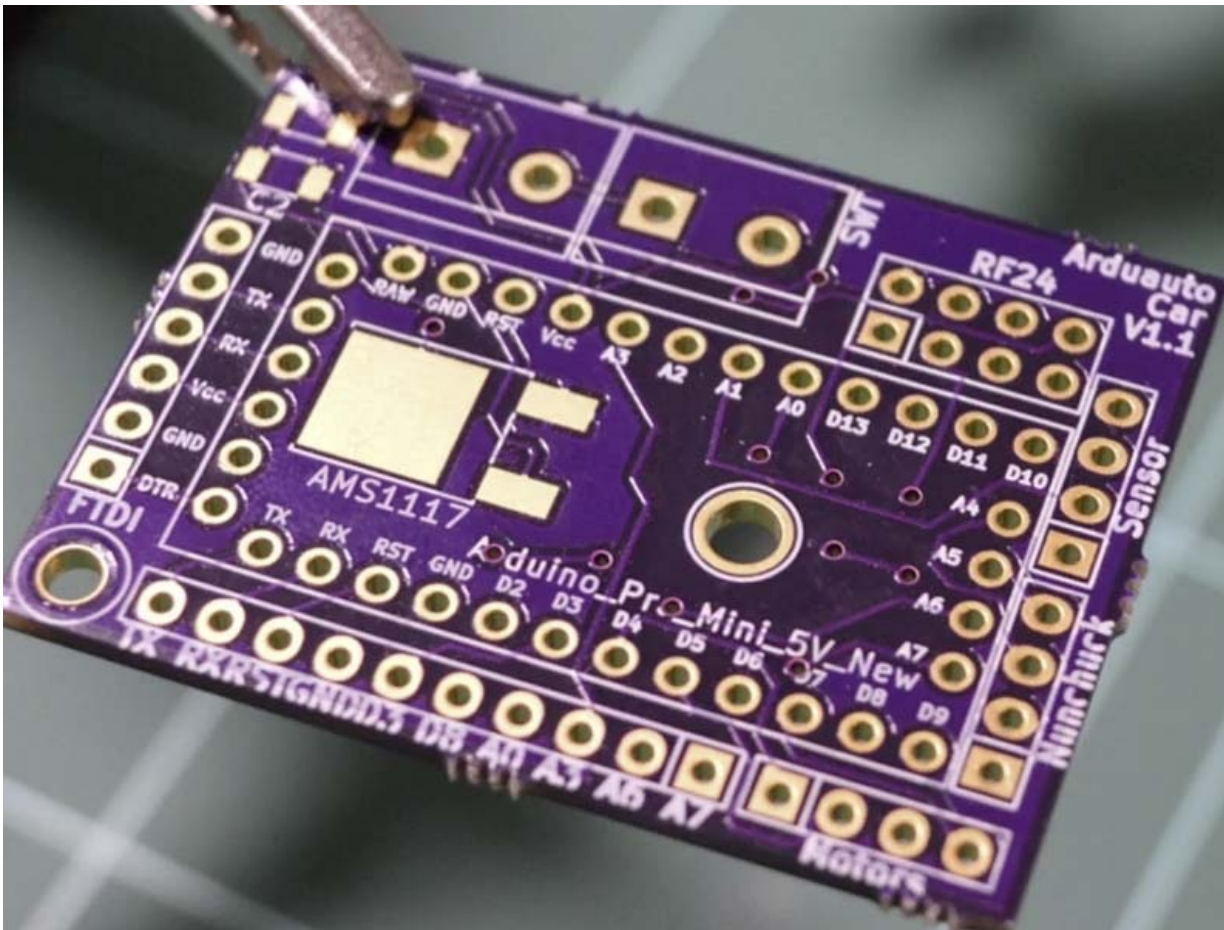
A through-hole component attached to a PCB

If you wish to attach a surface mounted component, then instead of holes, we just attach the component onto the surface of the PCB on pads. We use just enough solder to make the connection solid between the flat connector of the component and the flat pad on the PCB.



A surface-mounted component attached to a PCB

Next, we've got the silkscreen. So we use silkscreen to add writings and graphics or draw boxes and things like that onto the PCB just to mark the different areas where either different things go or to provide some information for the user.

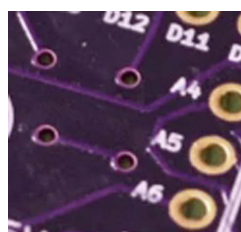


The white letters and lines is the silkscreen print on this PCB

You can see here that I've used boxes to delineate the location of various components. I've used letters and numbers 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 attach your PCB onto a surface just to secure it. To do that, you simply add more holes. The same ones that you used for through hole component is just that, in this case, you don't attach them electrically to any of the other pins. You can use a screw and a nut and bolt to the other side so that the PCB is secured inside, for example, a box.

Next are the tracks. In this example, they look purple because of the mask used by Oshpark.



The purple lines connecting the holes are tracks.

I'll discuss the mask soon. The tracks are made of copper and they just connect pins together, or different parts of the board. You can control how thick or thin a trace is. In terms of terminology, some people called traces tracks or connections or traces, so it's a few different names around meaning the same thing. I will try to standardize and just use the word track from now on.

Notice the small holes that have no pad around them? These are called "vias". A via looks like a hole, but are not meant to be used to mount a component on it. It's meant to switch a track from using one surface onto the other surface. If you're using PCBs that have more than two layers, like three, four, five, etc, then you can use vias to connect a track from any one of the layers to any of the other layers. Via are very, very useful for routing your tracks around the PCB.

The purple substance that you see on the PCB is called the solder mask. It's meant to do a couple of things. Firstly, it prevents the copper on the PCB from being oxidized over time. Oxidisation of the copper tracks affects their conductivity in a negative way. So, the solder mask prevents oxidization.

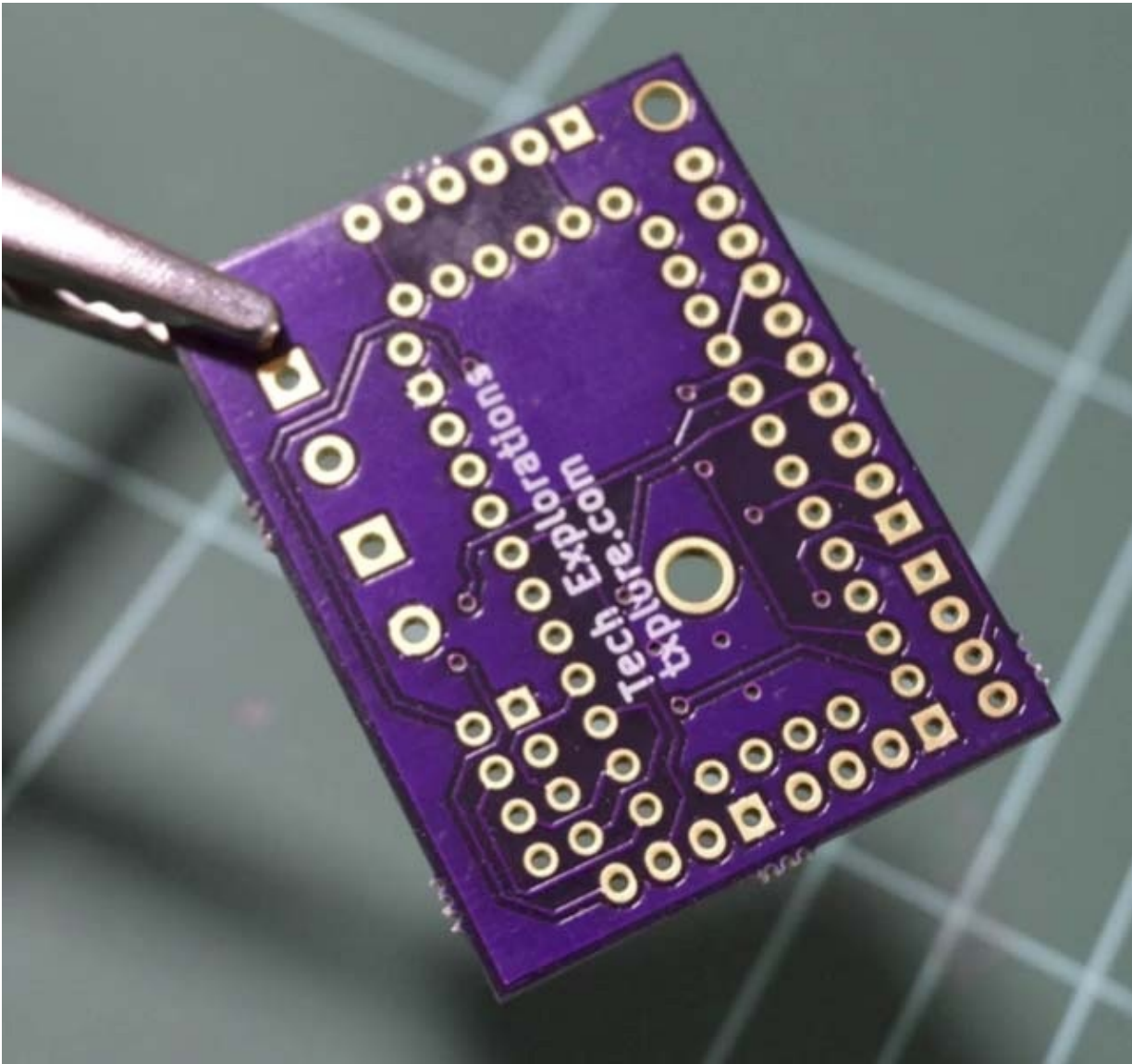
Another thing that the solder mask does is to make it easier to solder. On the pads, for example, you can see you can see that the pads contain the solder mask can be very close to each other. Sometimes, the distance between pads can be less than a millimetre. Without the solder mask, it is easy to create bridges between those pads. The solder mask prevents bridges because solder cannot bond with it.



A solder bridge like this one is a defect that solder mask helps in preventing

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

And here is the back side of the same PCB:

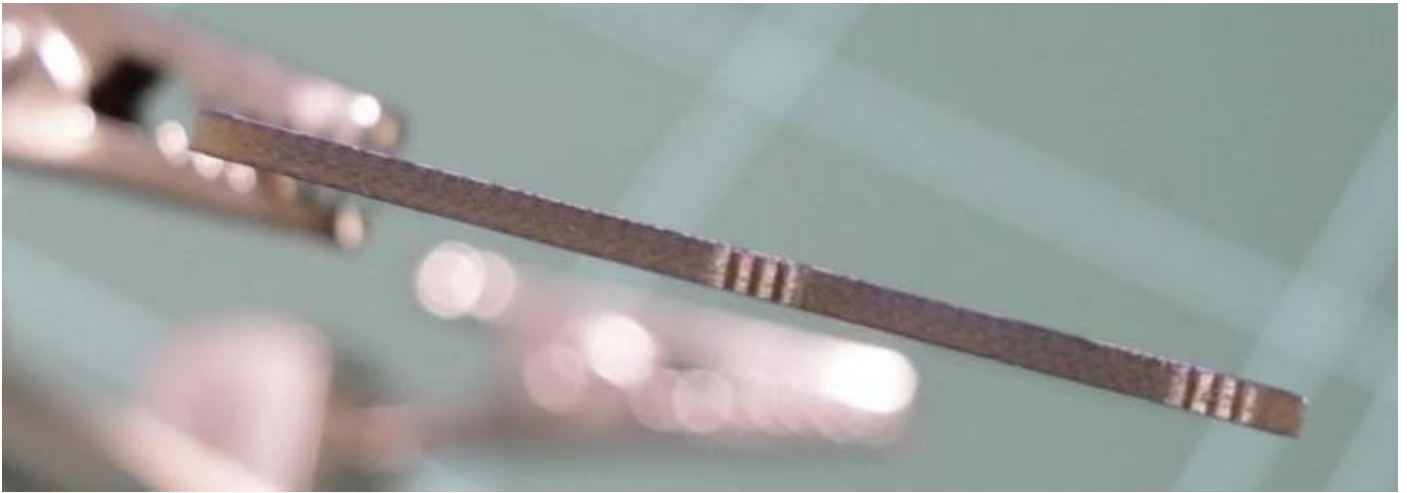


The bottom copper layer of the example PCB.

You can see that the holes from the through hole components extend to the back of the PCB, and there's also a pad there and that's where the solder goes, that's where the solder attaches.

You can't see any evidence here of the pads for the surface mounted components that exist on the other side. Those pads don't extend to the bottom side of the PCB.

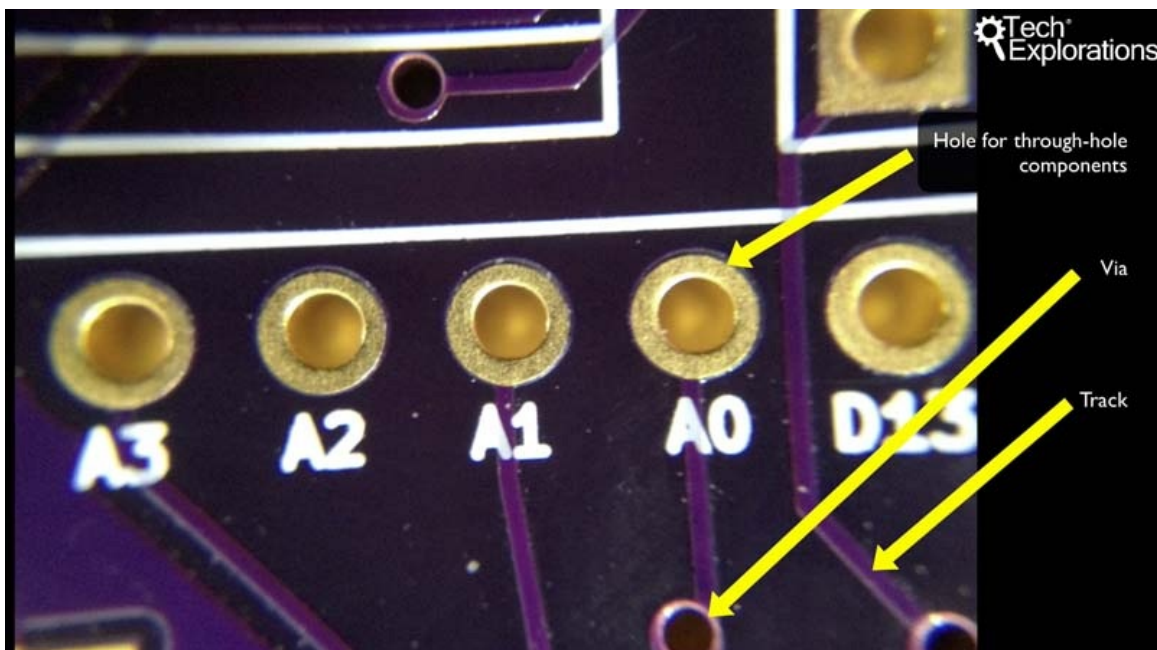
If you turn the PCB to the side, you can see how thick it is:



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

Typically, PCBs, at least the ones that you get from fabrication houses, are made of fiber glass. In the case of OSH Park and that's actually the case for most lower cost fabrication houses, the width of the PCB is around 1.6 millimeters. I can do a zoom-in and you can see the fiberglass in there.

In this microscope picture, you can see the holes for the through hole components:



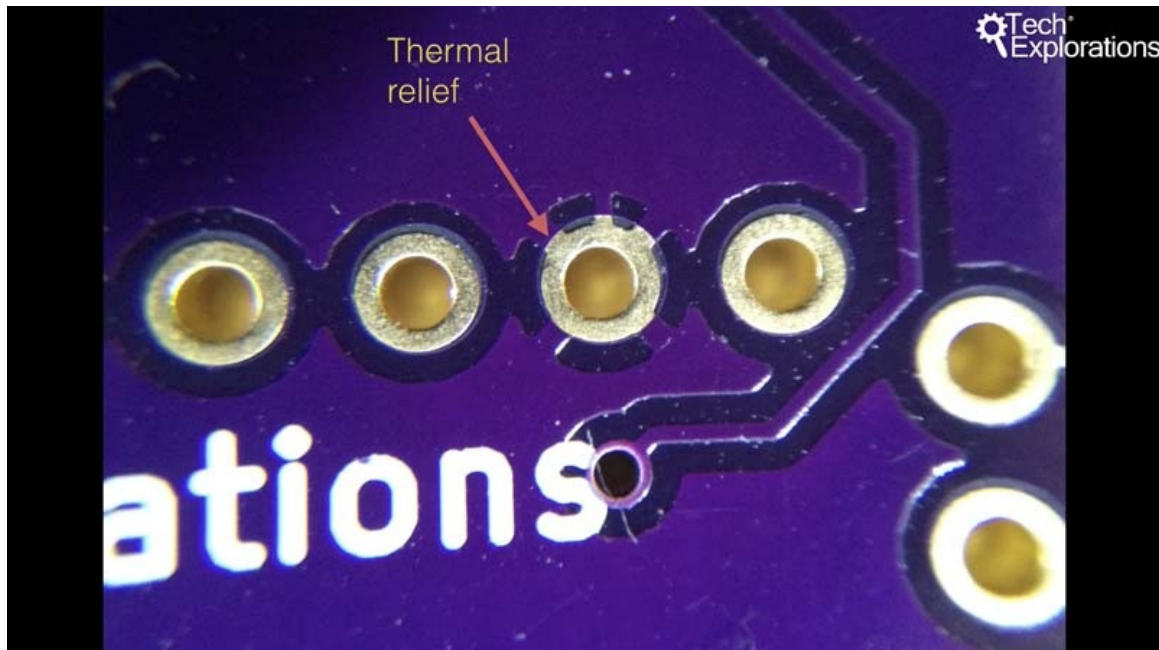
A microscope view of the top-layer

You can see a couple of vias and tracks, and of course you can also see solder mask and you can see how in between the pads, there is solder mask.

In this close up, you can also see the detail of the silkscreens. The wide ink is what you use in the silkscreen to do all your markings and your writings and it's around—depending on the manufacturer – it's around 150 points per inch. It's not as good as a good

laser printer, which goes to about 300 points per inch but it's good enough. And for more expensive manufacturers, you can actually go to very high definition and have very elaborate and highly detailed markings and drawings done on your PCB.

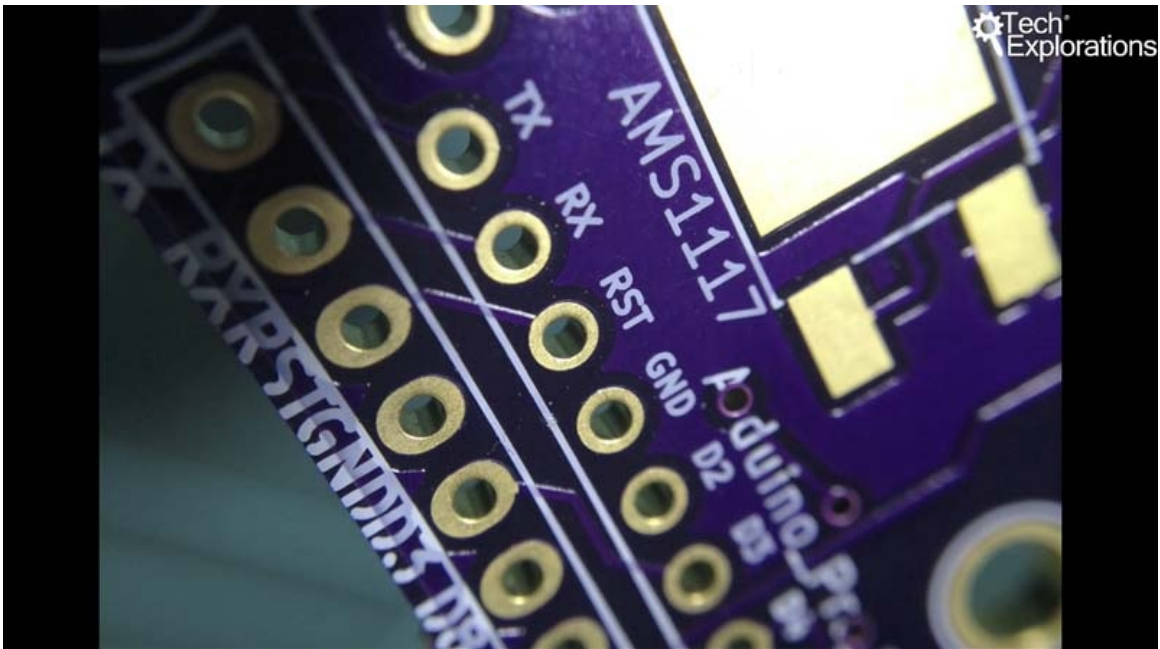
This picture here is interesting because it shows you a way to connect grounds and VCC pads to large areas of copper which is called the copper fill:



Thermal relief connects a pad to a copper region

The arrow here shows you how this particular pad is connected to a large area of copper around it via special connections called thermal relief. Through thermal reliefs, you can both have an electrical connectivity between the pad and its surrounding copper that can carry current to other parts of the board.

Here's another nice view of the front of the PCB:



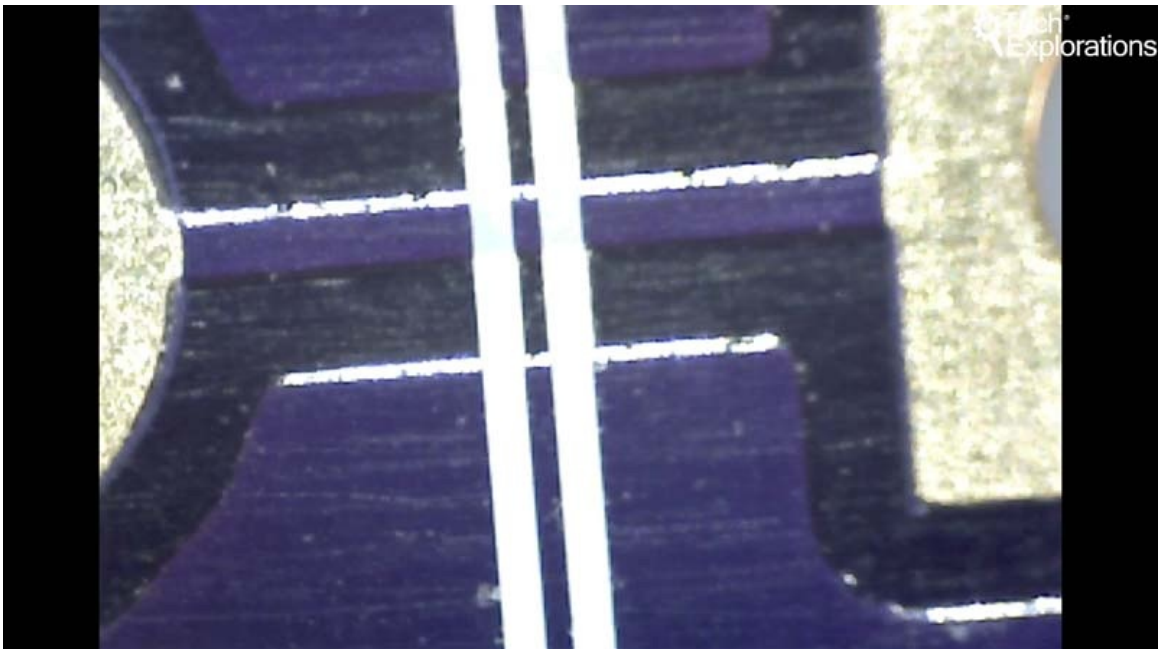
Notice how the plating of the holes covers the inside of the hole, and connects that front end with the back end.

This one, gives you a way to also appreciate the thickness of the tracks. Notice, for example, the short track that connects the two reset holes (RST). The light that is reflected off the side of the track gives you an idea of the thickness of that copper which is covered by the purple solder mask.

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

Instead of gold plating, you can also use tin plating and that's a cheaper way of creating pads and several manufacturers also support, actually, most manufacturers seem to support tin plating.

Finally, this is the last close up image:



Notice the light reflecting of the side of a short track. This gives you an idea of the track width.

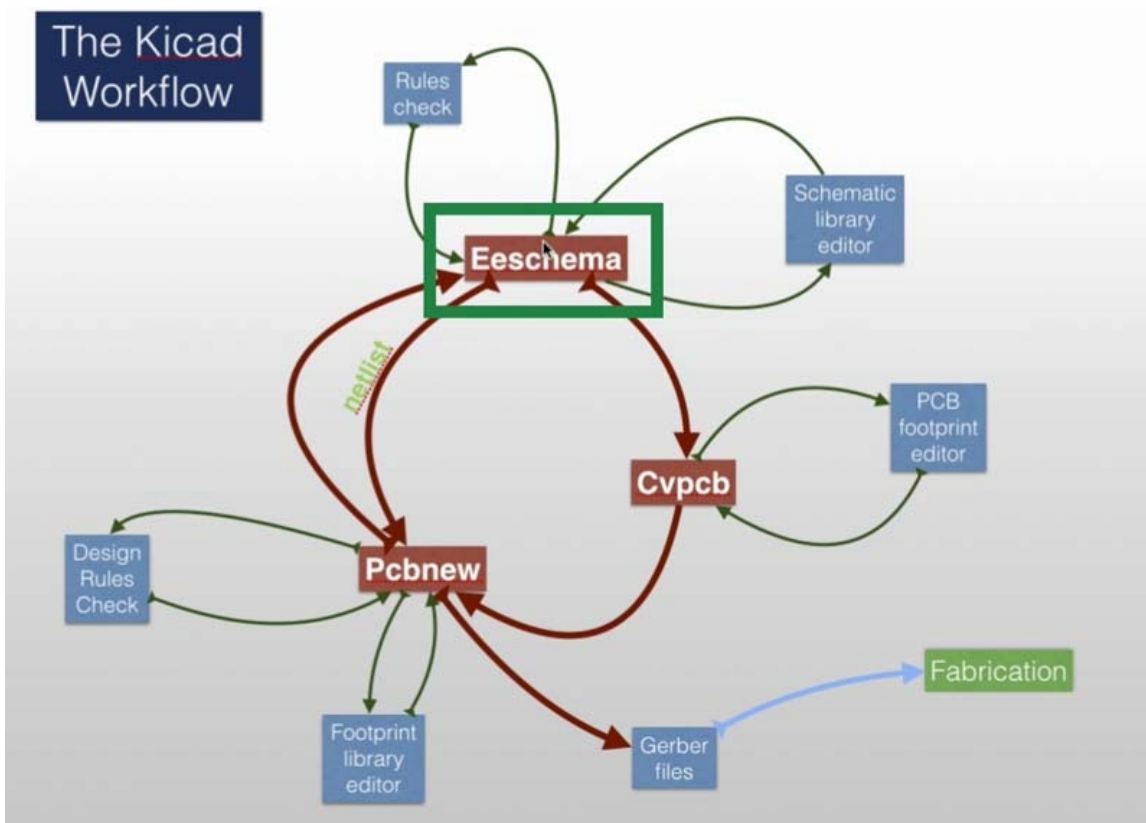
This image is at 200 times magnification. so you can see here that you've got a track connecting two pads, light reflecting off one side. Gives you an idea of how thin the track is.

So with this, I hope you have a better idea of what a PCB is like close up and its various components and the terminology used.

In the next chapter, we'll have a look at the PCB design process using Kicad.

Chapter 9: *The Kicad design process*

In this chapter you will learn about Kicad's PCB design workflow.



The process of designing a PCB using Kicad begins with Eeschema.

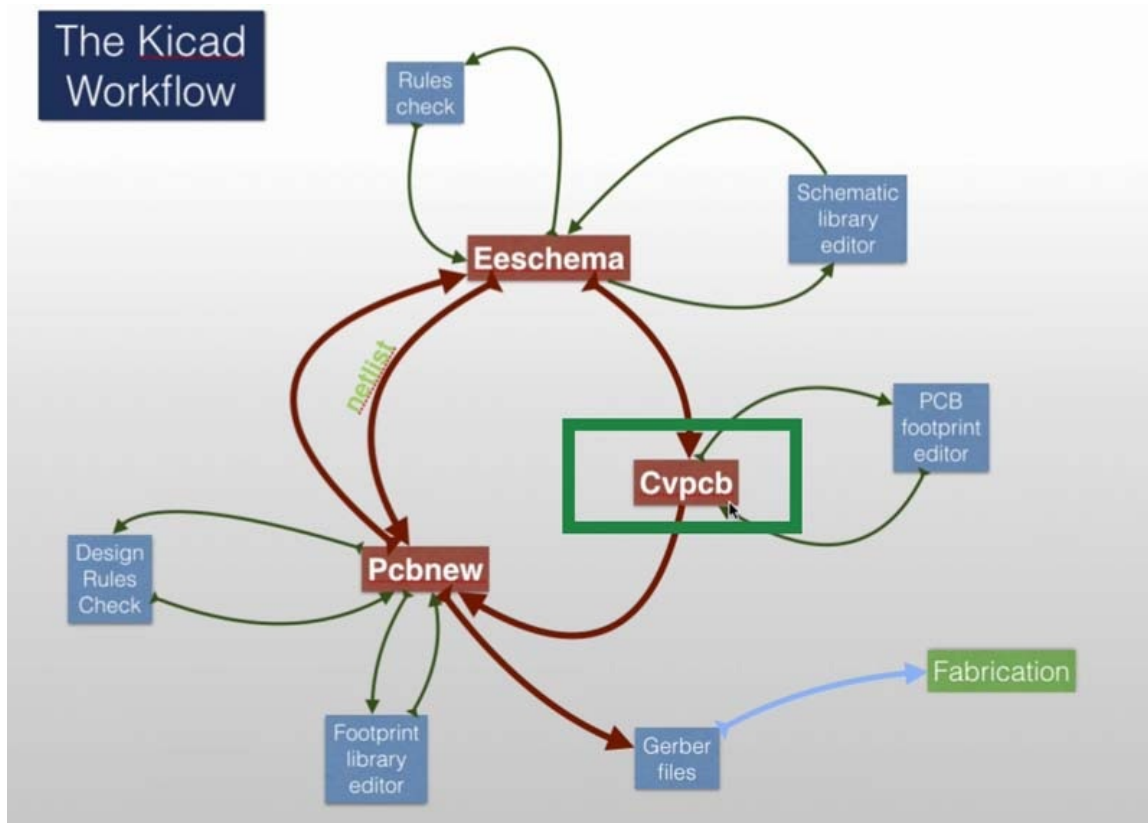
The process begins with Eeschema. In Eeschema we create the electrical schematic that describes the circuit that eventually will be printed onto the PCB board. We draw the schematic by picking components from the library and if a component that we need doesn't exist in the library, we can create it using the schematic library editor.

At some point in our design of this schematic we can run the electrical rules check and that is a utility that ensures that we don't have any obvious mistakes or any electrical mistakes in our design. The electrical rules check will give us a defect report and we'll use

that report correct any problems in Eeschema.

Once we have everything set out correctly and there are no more defects in our schematic, we are almost ready to move ahead and go into the Pcbnew application in order to start doing the layout for our PCB.

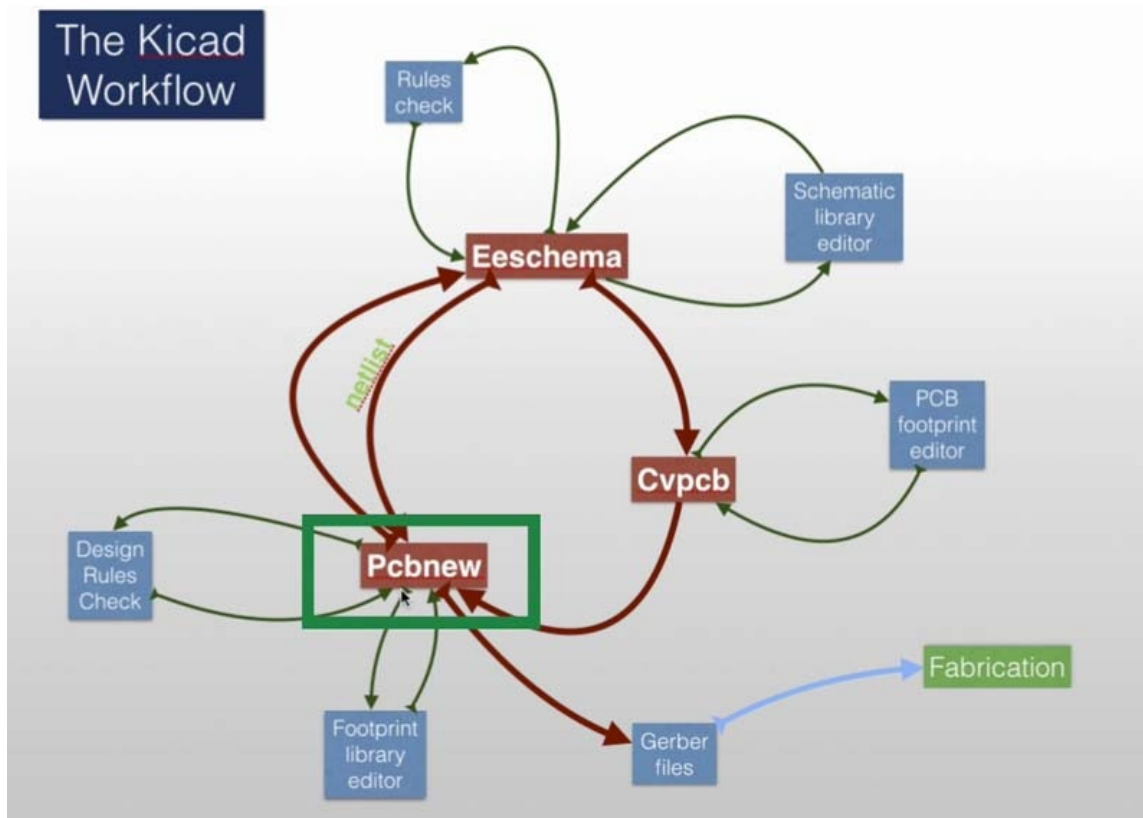
There are two things we need to do before we go to Pcbew. The first one is to associate the components in Eeschema with footprints. To do that we use another component of Kicad called Cvpcb.



With Cvpcb, we associate schematic components with footprints.

This allows us to match schematic components with footprint modules. One thing to remember in terms of terminology is that in Kicad a schematic has components and a PCB has footprints. If there is a footprint that doesn't exist in the footprint library, then again we can create it. There is a small utility called the "PCB footprint editor" that allow us to create our own custom footprints just like you can create your custom schematic components using the schematic library editor with Eeschema.

Once you have the matching between components and modules completed then we export a file called Netlist through Eeschema. The netlist file will contain in it all the information about components and their connections to other components, as well as their associations with footprints and other meta-information (like component names and values). Then we'll import the netlist into Pcbnew and all the components that we created in Eeschema will appear in a new blank canvas that makes up our PCB.



Once the schematic is complete, we work on the layout of the board using Pcbnew

We use Pcbnew to place the footprints on the blank canvas, and then do the wiring. We can move the components around so that they are nice and tidy and we'll start doing all the connections between the various pins using. This process is very time consuming, especially for large projects. The placing and routing is mostly a manual process. Although Kicad does have an auto router, most people prefer not to use it, in favour of the manual process which give the designer absolute control.

Once you have your PCB laid out and connection's completed you can go ahead and do a design rules check that's going to make sure that, for example, a track's not too close to a pad and things like that. Look for any pads that are not connected and things of that sort. We can always fix footprints using the footprint editor.

Assuming that you are finished with the current iteration, you will want to have it fabricated. To do that you must export the PCB data as a collection of files, called "Gerber files". I will describe Gerber files in more detail in the next chapter. Gerber files contain several related files, with one Gerber file per layer on your PCB and contains instructions that the fabrication house will need in order to make your PCB.

One thing to notice about the Kicad design process is that it has a lot of closed loops; there is a lot of iteration happening. As we'll see in the first example project, It is often the case where you go into your PCB design and then you realize that there is a component you should have included in your schematic design in Eeschema but then you forgot about it or you just didn't think it would be necessary until later in the process. In this case you can go back to Eeschema, adjust the schematic with whatever was missing or make your additions or changes and then go and create a netlist and go back to Pcbnew to continue your layout work there. There is a lot of iteration that is happening here. This is ok! It's

not that hard to be iterative in Kicad; the workflow allows you for that.

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

Chapter 10: *Fabrication*

Let's imagine that you have finished with laying out your board in Kicad and you're ready to make it. What are your options? Well, you can do it yourself at home. There's a nice guide here from Fritzing, at <http://fritzing.org/learning/tutorials/pcb-production-tutorials/diy-pcb-etching/>.



Etching is a "subtractive" method used for the production of printed circuit boards: acid is used to remove unwanted copper from a prefabricated laminate. This is done by applying a temporary mask that protects parts of the laminate from the acid and leaves the desired copper layer untouched.

You can etch a PCB by yourself, in a lab or even at home, through a simple and inexpensive production process. It makes sense when you wish to produce a single or a very small number of boards and want to avoid manufacturing costs. The etching process is therefore effective for a small workshop. There are however some issues to consider:

1. There is a risk of injuries due to the chemicals involved.
2. The quality of the results depends on several factors which you won't be able to master completely the first time. This can be somewhat compensated by using good machinery.

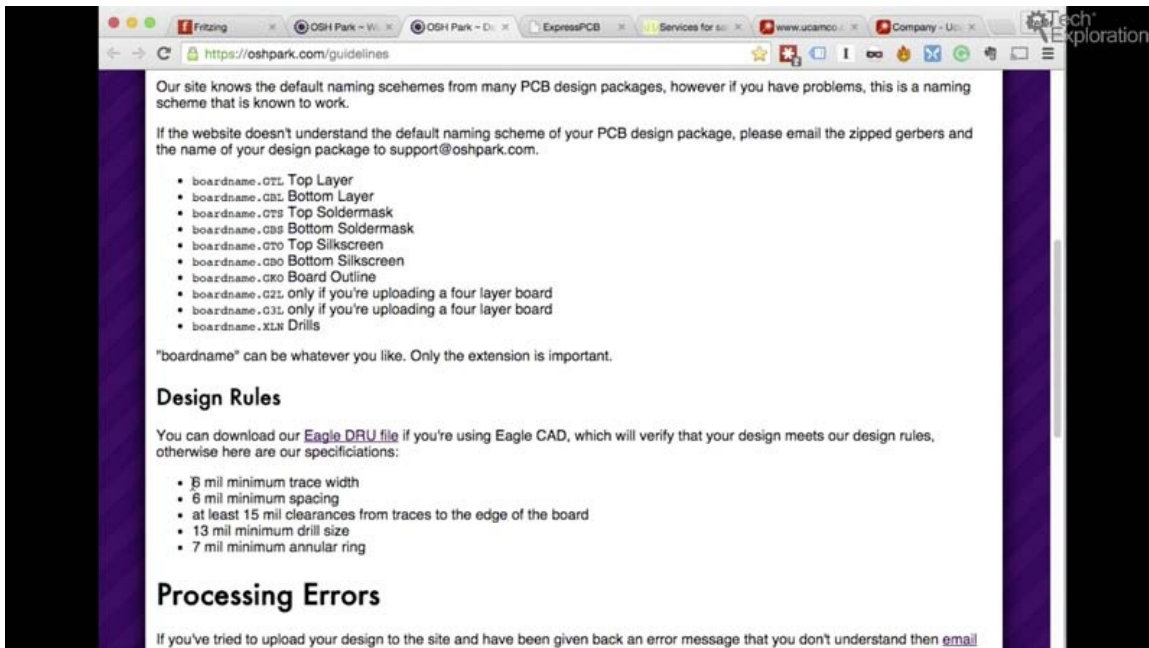
Etching, a process by which you can make your PCBs at home.

The process described in the Fritzing article 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, hopefully keep your children away if you have any, deal with smelly and potentially dangerous fumes. Once you have a board etched, you still need to use a drill to create holes and vias, and then figure out a way to connect your top and bottom layers.

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

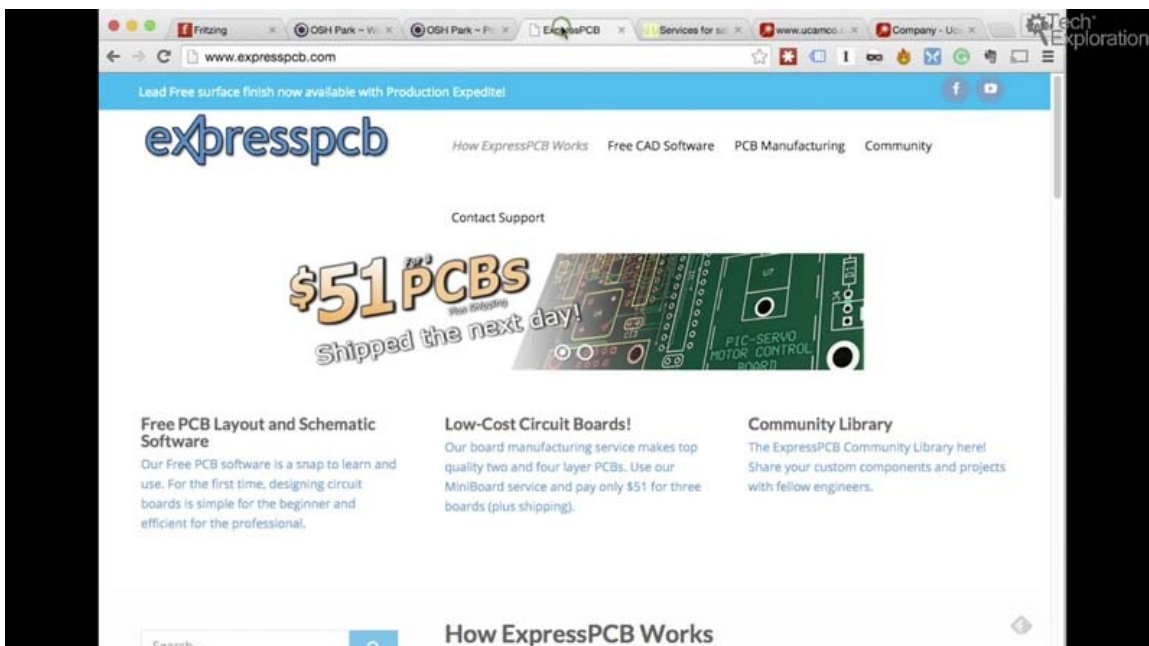
You can get a professionally made PCB for not a lot of money and without danger to yourself as well. I've used OSH Park and some of these others a lot of times. I'm always happy with the result and it does take a little bit of planning because once you order your PCBs it could take a three or four weeks, at least back here in Australia to get your PCB shipped back to you. If you're in a hurry there are options to expedite the process, it gets a

bit more expensive though. In this book, I'll be using OSH Park just because I've used them before, they're very convenient and I like the outcome. Their design rules which are fairly standard in the industry, so everything I show you in the book will work with the vast majority of public PCB manufacturers.



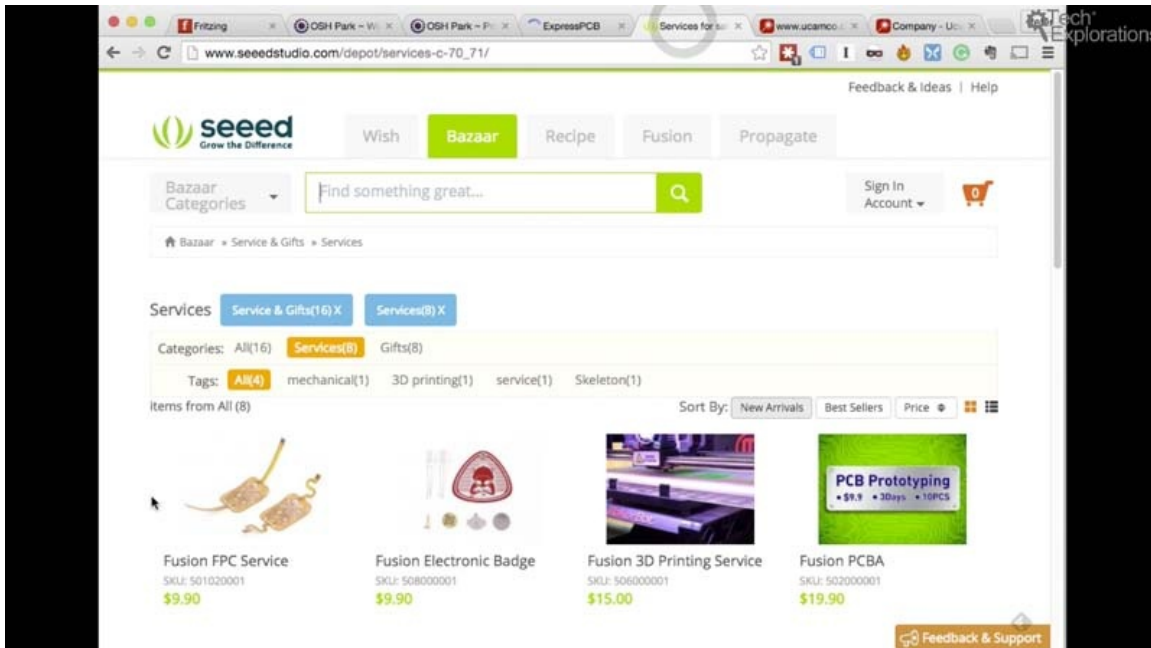
OSH Park's design rules are standard in the industry.

The typical standard two-layer order will cost \$10 for a two square inch board and you get three copies of that. This works out to around \$5 per square inch. The pricing is consistent in the industry. You pay based on how big your PCB is. So there's a good incentive to try and make your PCBs as small as possible. Be aware of this when you do your layout, try to keep the components as closely packed as you can. There is a give and take between the proximity of the components to each other and the amount of space there is left for the routing. The less space there is available, the harder it will be to do the wiring.



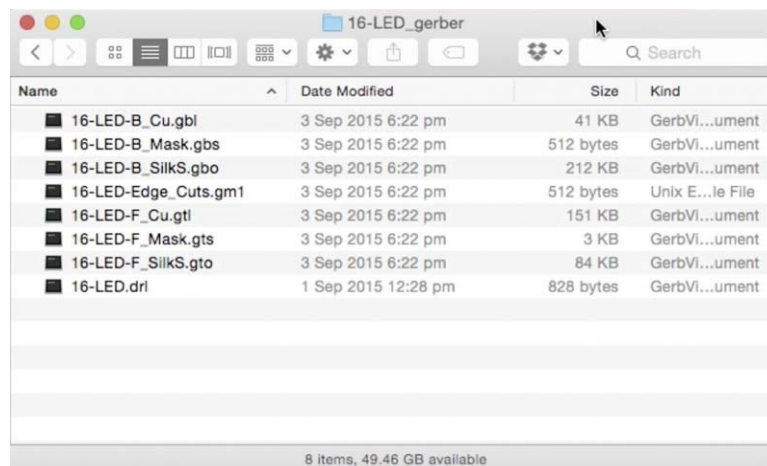
Expresspcb.com, another public PCB manufacturer

There's also ExpressPCB, I've never used ExpressPCB myself so I can't comment about them. But it seems like also a layer to download for free their own ExpressPCB software and use that to build your PCBs.



Seed's advantage is their impressive range of PCB customisation they offer to retail customers

Another example is Seed. I've used Seed in the past, really nice, just as good as OSH Park in my opinion and they do give you a much wider option of how you'd like your PCB to be made. For example, if we go to Fusion PCB that's one of the types of products that you can get from them. You start at \$9,90 but then you can choose things such as how many layers it'd like to have, the dimensions of your PCB. This is a nice one, the thickness, if you pressed for space in your application, if you building a wearable gadget perhaps, then you can reduce the thickness of the PCB. You can choose the quantity, the PCB colour. Nice touch, you can make your PCB red, for example. These are just some of the customisations offered by Seed.



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

Now, let's turn our attention to the files that you need to upload for these services and the files are Gerber files. Each layer on your PCB will have its own Gerber file which is simply a text file.

Let's have a look inside:

```
1 G04 #@! TF.FileFunction,Soldermask,Bot*
2 %FSLAX46Y46*%
3 G04 Gerber Fmt 4.6, Leading zero omitted, Abs format (unit mm)*
4 G04 Created by KiCad (PCBNEW (2015-07-06 BZR 5891, Git 351914d)-product) date
  03/09/2015 18:22:08*
5 %MOMM*%
6 G01*
7 G04 APERTURE LIST*
8 %ADD10C,0.100000*%
9 %ADD11R,1.727200X2.032000*%
10 %ADD120,1.727200X2.032000*%
11 G04 APERTURE END LIST*
12 D10*
13 D11*
14 X122555000Y-42545000D03*
15 D12*
16 X120015000Y-42545000D03*
17 X117475000Y-42545000D03*
18 X114935000Y-42545000D03*
19 X112395000Y-42545000D03*
20 M02*
21
```

Gerber files contain text.

You can see that this is just a text based file contains instructions. Although you could edit such files with your text editor, they are meant for the manufacturer's equipment. An advantage of this text format is that you can use a version control system like Git and keep your projects stored in repositories like [Github.com](https://github.com).

The Gerber files system and standard has been designed by this company here Ucamco. They make equipment and write software for PCB manufacturers – things like a PreCAM software, PCB CAM, there's a laser photo plotters and direct imaging systems. This is the company that makes the hardware that companies like OSH Park and Seeedstudio will use to make your PCBs. If you're curious about how to read these Gerber files then you can lookup the specification of the Gerber format specification in Ucamco's web site. Beware, it's a huge file.

That's enough about fabrication and Gerbers for now. In the next chapter we will get started with the first project for this course and we will build one sided-NRF24 breakout.

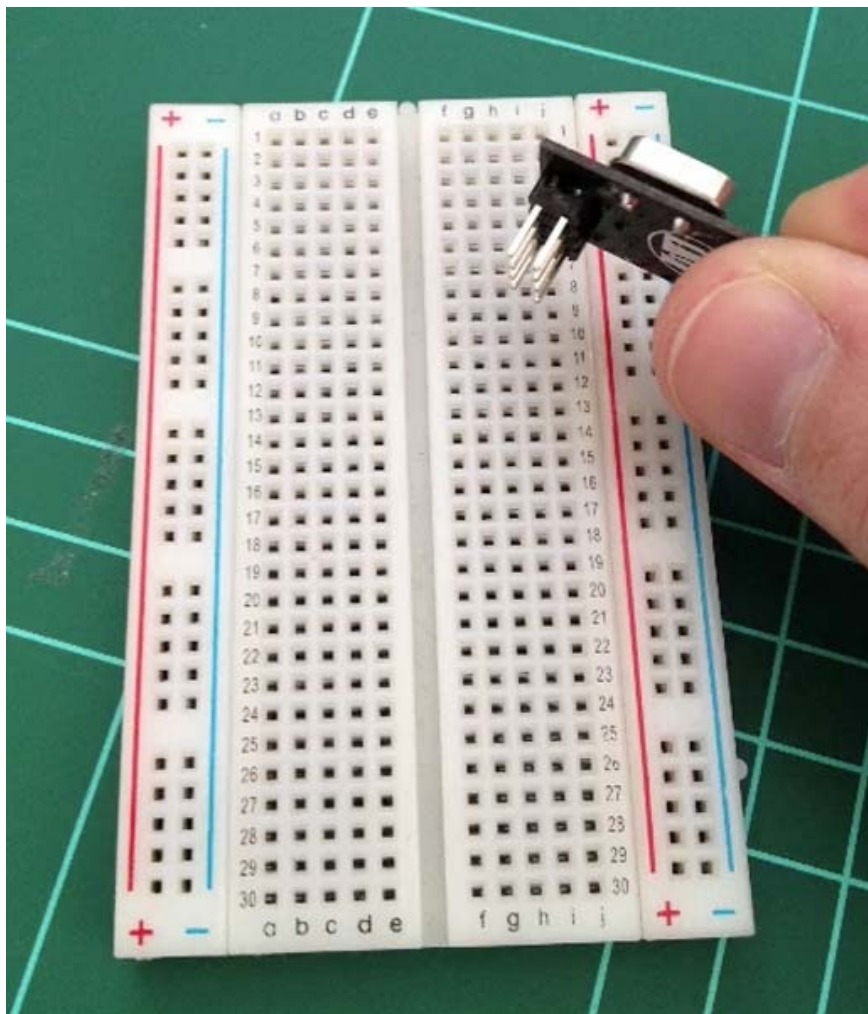
PART THREE

Project 1: create an nRF24 breakout board

Chapter 11: *What is this part*

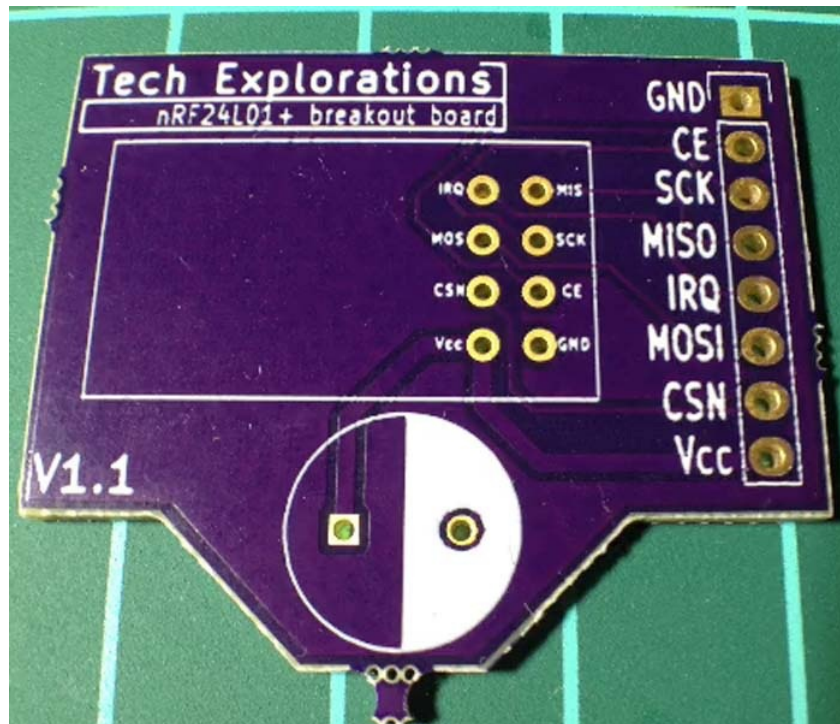
In this first project for this course, we'll build a simple breakout PCB for the nRF24 board.

If you have worked with the common nRF24 board, you know that its two rows of pins are not compatible with a breadboard.



The pins of the common nRF24 breakout are not compatible with a breadboard.

As a result, you have to use ribbon cable and stick jumper wires in its connector, and then stick these wires in the breadboard. I have wasted so much time dealing with a mess of wires, debugging with my multimeter that I decided it is time to deal with this problem once and for all.



We will create a PCB that will make it easy to use the nRF24 breakout with a breadboard.

The breakout we will create in this project will allow us to escape the two rows by four pins that the nRF24 comes with into a single row of 8 pins. We will design the connector on the PCB so that it is compatible with a breadboard. This first PCB will be single-sided. In the next two projects, we will work on 2-sided PCBs.

In this project, you will learn about:

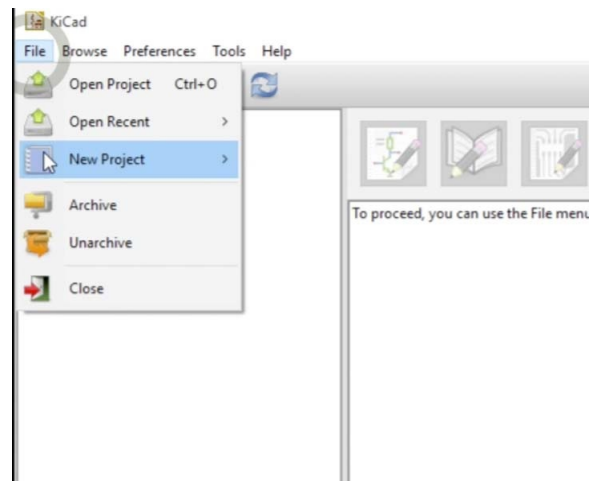
- * How to create a new Kicad project and set up it's parameters
- * How to create a simple schematic using Eeschema
- * How to find schematic components in the library
- * How to create a custom schematic component
- * How to annotate parts in a schematic
- * How to make sure that your design is ok by doing an electrical rules check
- * How to associate schematic components with footprints
- * How to create custom footprints

- * Footprint features, like pins, pads, silkscreen borders, and labels.
- * How to design a PCB using the Pcbnew tool.
- * Creating and importing a netlist
- * Several of the Pcbnew features, like edge cuts, 3D views, making wirings, copper fills, tracks, thermal reliefs, adding labels and versioning.
- * How to modifying the schematic and update the PCB design based on the updated schematic.
- * How to create the Gerber files
- * How to upload your Gerbers to a fabricator and order your PCB.

There's so much to do, so let's get into it!

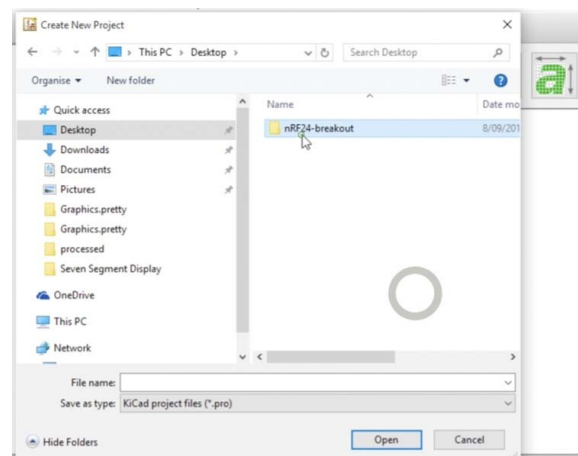
Chapter 12: *Creating a new project*

It is time to get started with the first project. Start KiCad, then click on the File menu item and select New Project.



Start Kicad and create a new project

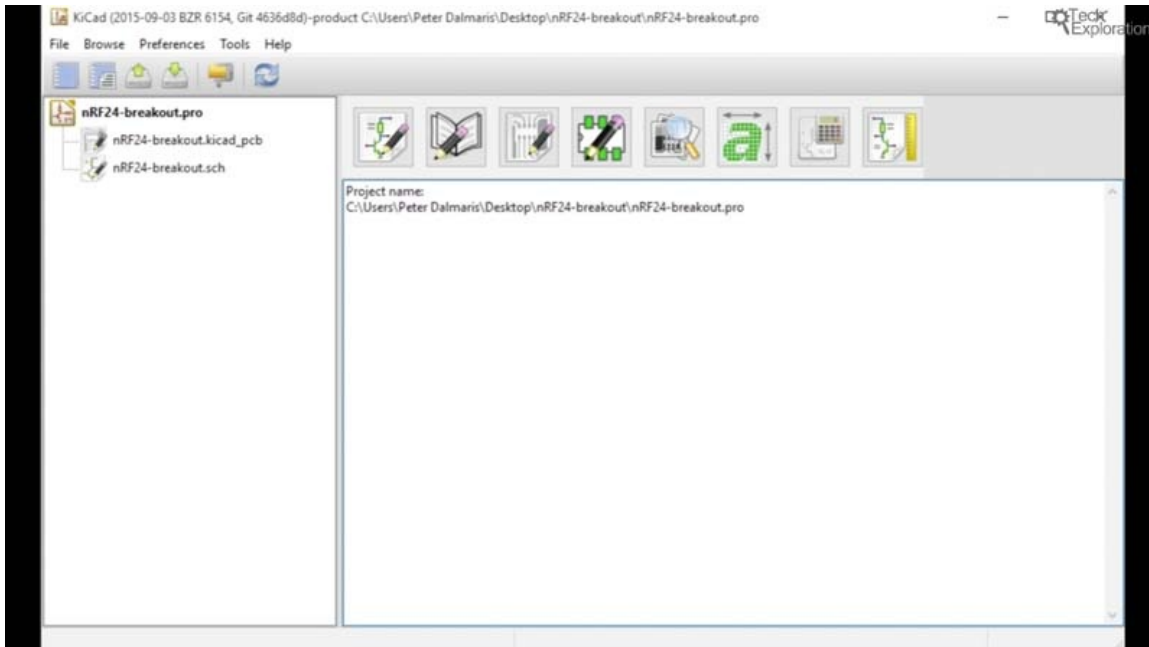
It is a good practice to store project files inside a project directory. Create a new directory named nRF24-breakout.



Create a new directory to hold the project files

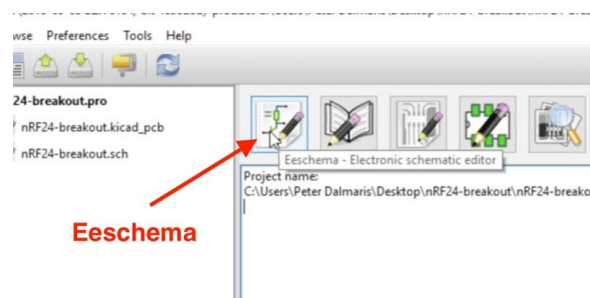
Then, go inside this new directory and create the project file, call it nRF24-breakout. Click on the Save button to finish the process.

Your new project should now look like this:



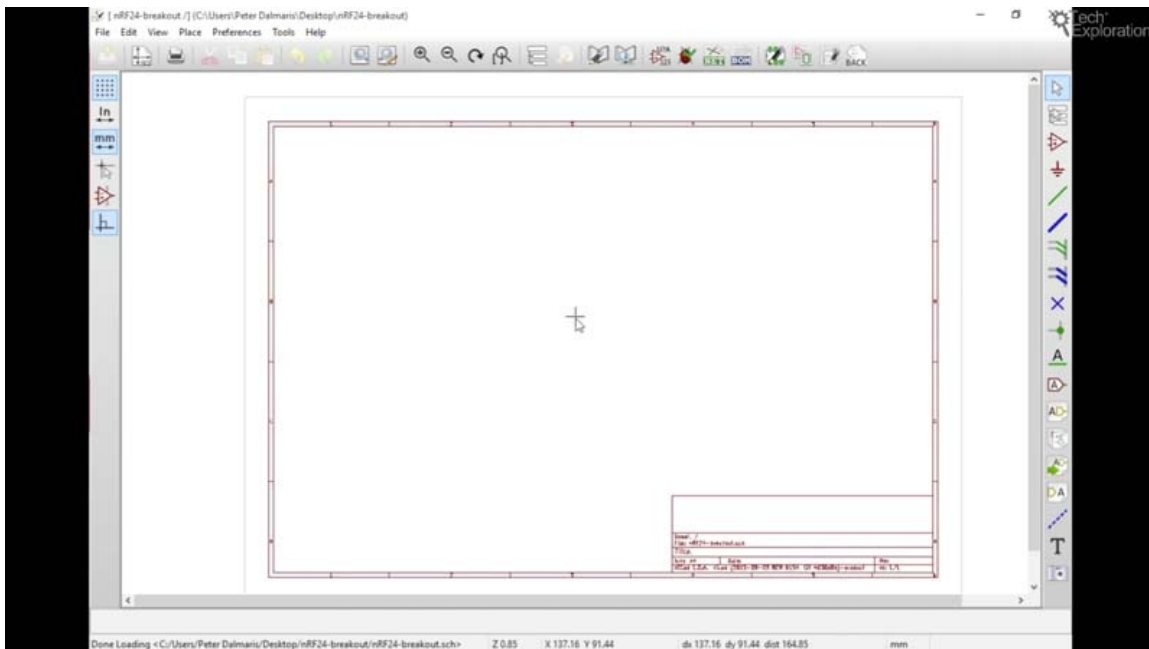
Your new project.

You may remember from the workflow that the first thing that we do when we create a new project in KiCad is to create the schematic with Eeschema. To start Eeschema, the Electronic Schematic Editor, we click on the first button from the left:



Start Eeschema

Once the Eeschema window appears, maximise it to gain as much screen real estate as possible. Components will be going into this canvas, which is the white area inside the red border, in the middle of the screen. Much of the work that you'll be doing will be done via shortcuts and through the mouse.

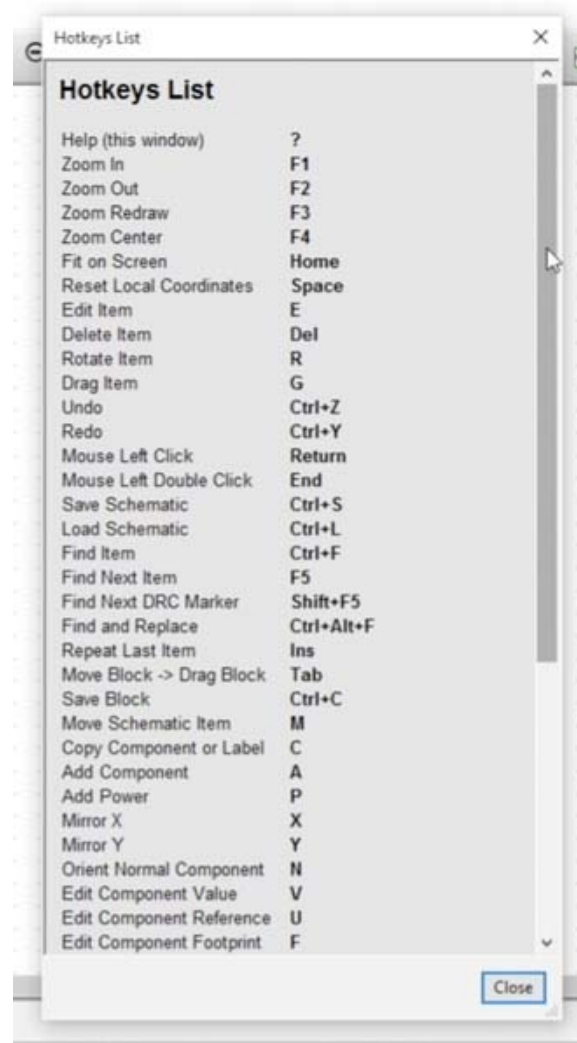


The blank canvas in Eeschema

You can zoom in and out using the scroll wheel of the mouse. This is a basic function that you will be using constantly. If you've got a mouse without a scroll wheel, I strongly suggest you get one with a scroll wheel. I use a Logitech Bluetooth mouse, and it is very convenient.

Another very useful feature is panning. Panning allows you to move around the canvas by clicking it's left button while holding the command key on my keyboard (I am using a Macintosh keyboard with a Windows virtual machine, so the exact key combination may be different for you). Depending on the keyboard that you have, you may need to use a control or the shift key and that, again, it depends on whether you are on Windows, Mac or Linux.

If you type Shift and the question mark then you'll get the hotkeys list which contains all the most important and commonly used shortcuts.



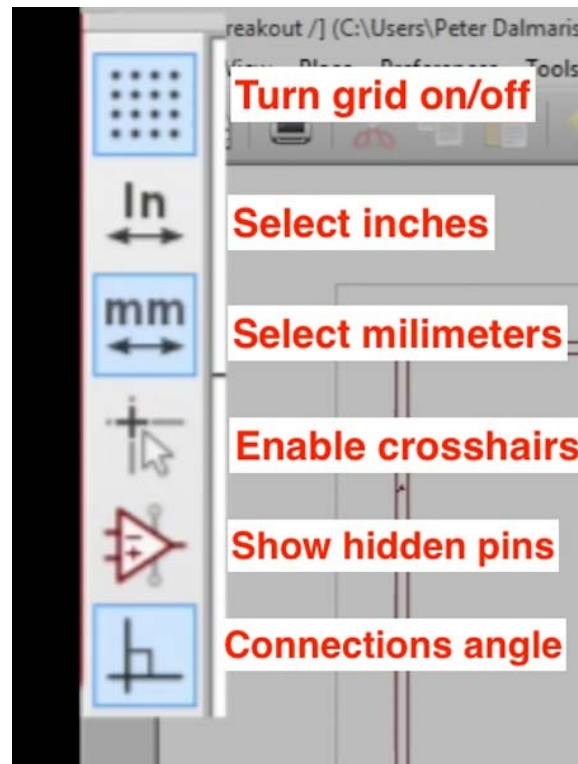
The hotkeys list.

For example, by pressing the A key and you can add a new component. By pressing the P key you can add a particular kind of component: a power component. You can use the V key to edit a component value so it can set for example a resistor to its particular value and so on.

There are a lot of hotkeys. We will not going to be using all of these in this project, but you can speed up your work by a lot if you can memorise only 4-5 of them. If you forget a particular shortcut, remember to type Shift and the question mark to bring up the hotkeys list.

If you look carefully at the canvas you will notice small dots spread out throughout. These dots mark the grid. The grid allows you to align the schematic components in tidy rows and columns. You'll be using the grid to make sure that everything aligns well.

On the left side, you've got a few buttons so you can switch your distance units to inches or millimeters.



Useful buttons in the left tool bar

I'll be using millimeters in this book. You can change the cursor shape to cross hairs by clicking on the crosshairs button. I think that's a little bit distracting so I prefer to have that off and just have a small cross in the middle. The "Show hidden pins" button allows you to show hidden pins, usually found in integrated circuit components. We're not going to use this feature in this project but our next project will have integrated circuits with hidden pins, so we'll be using this to turn them on and off. The last button allows you to draw wires and busses in any direction.

The components, the wires, the busses, the labels and so on are available via the right tool toolbar.



The right tool bar.

I'll be showing you what these are as we go through.

Finally, we have the main navigation and tools bar above the canvas.



The top navigation and tools bar

Through this bar you are able to do things such as zoom in and out, rotate a component, and redraw the canvas. If you have somehow moved too far away from the part of the schematic that you'd like to be working on, you can click on the "Fit schematic" button (look for the button with a magnifying glass over two square brackets, towards the middle of the bar) to take you straight back to the centre of the view and the canvas, so then from here you can go back to your component.

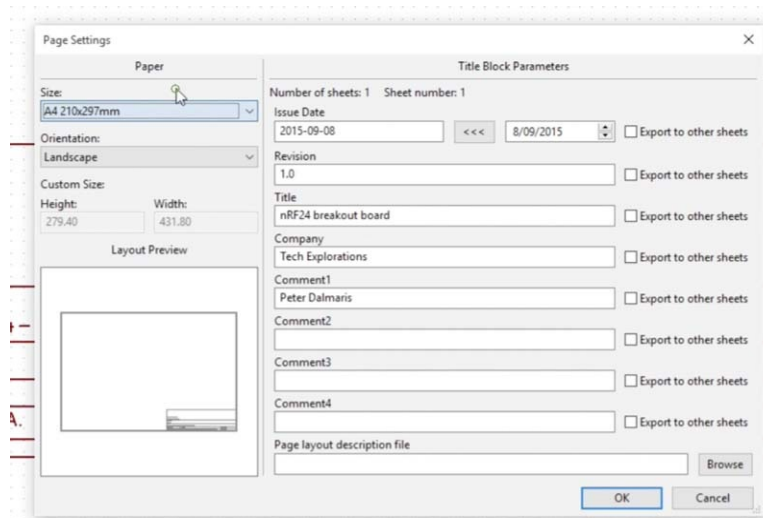
The next thing that I try to remember to always do is to set the page settings. So what this does is to populate the text in the label down the bottom right corner of the schematic. So you can see that right now it's empty and I'd like to put in the attributes of my schematic in here. And usually I forget to do this if I leave it for later so it's the first thing that I do whenever I start a new project.

To do that, click on File and then Page Settings:



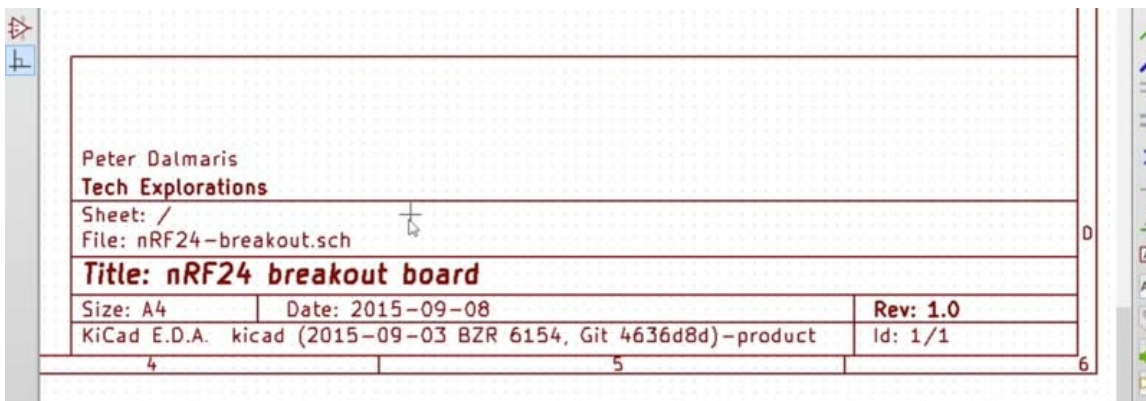
Getting to the Page Settings window.

Then, fill in the attributes as you see fit. To fill the issue date click on the button with the three arrows; this will copy the date from the calendar into the issue date field.



The Page Settings window

For revision, since this is going to be our first revision, I'm just going to say 1.0, I give it a name, so this is an NRF24 Breakout Board, the name of your company. Fill in the rest as per the example in the image above.



The updated page settings

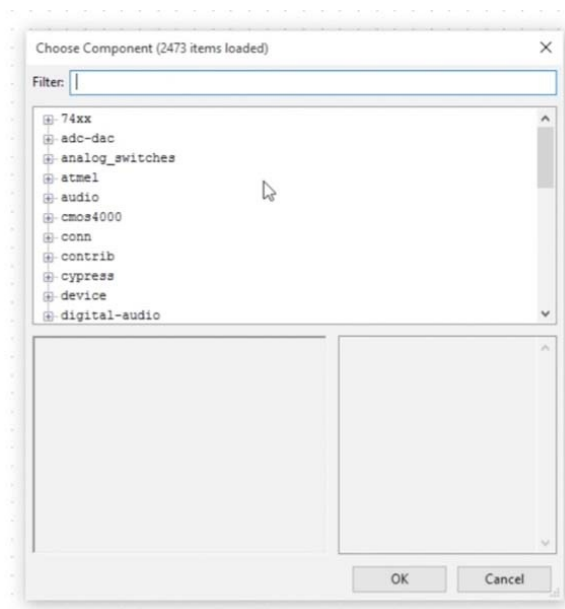
And you can see that all the details that I've just entered are now part of my schematic, the bottom right corner in the label.

Chapter 13: *Starting the schematic for the nRF24*

In this chapter we will begin the process of creating our first schematic in Kicad.

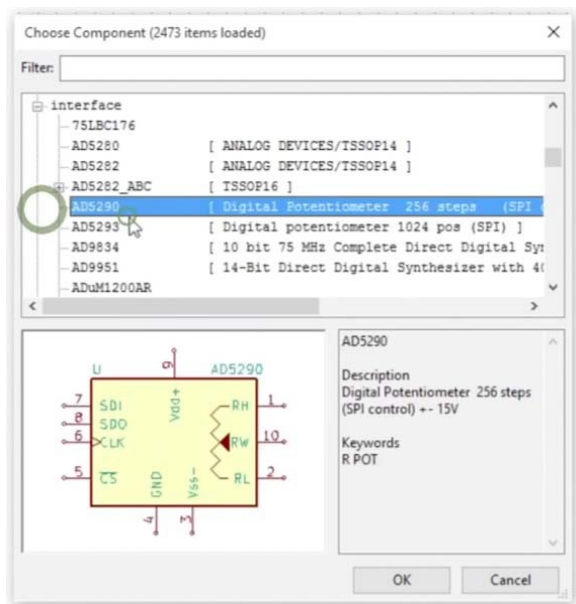
Start by centering the schematic in the window so you can see the whole canvas. The board that we are building will only contain two components. The first one is a straight 8-pin connector and the second one is an RF24 component that we will plug into the board. The RF24 component does not exist yet in the component library, so we will have to create it.

But the straight connector does exist, so we will start with this. Zoom in the canvas a little and then hit the A key on the keyboard. This brings up the component chooser.



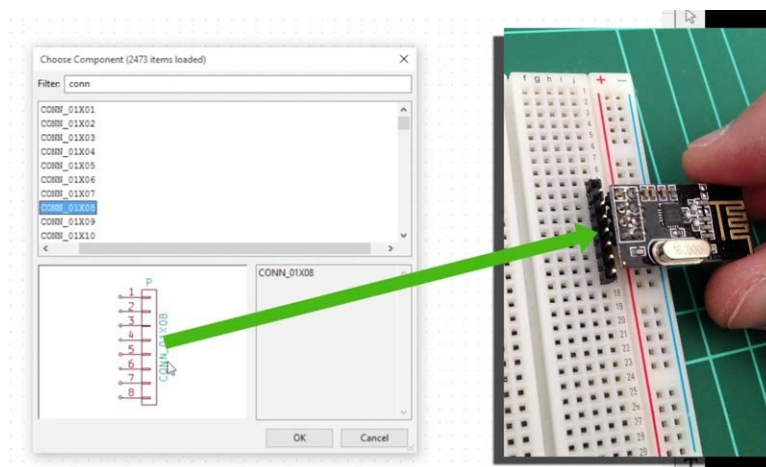
The component chooser

You can manually drill down these libraries and look at the components and figure out which one is the one that you're looking for. As you're clicking on a component, you're going to get a view of its schematic and some description of the component. Sometimes you will get a detailed description, some other times you will get almost nothing.



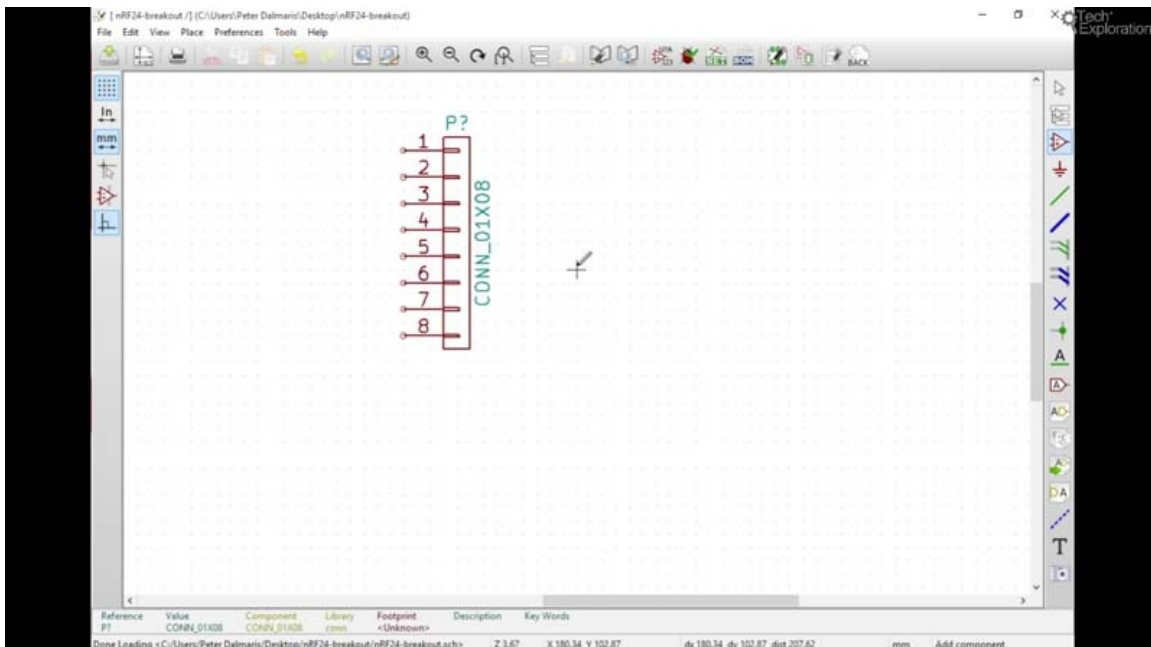
The component chooser often (but not always) contains details and a schematic preview.

Probably, a faster way to do find a component is by using the filter. You can type a keyword, like “connector”, in the filter text box, and components that contain this keyword will show up. In this example, the keyword “connector” returns nothing, but “conn” returns several components.



Use a keyword to quickly find a component. In this case, the 01x08 connector is what we need.

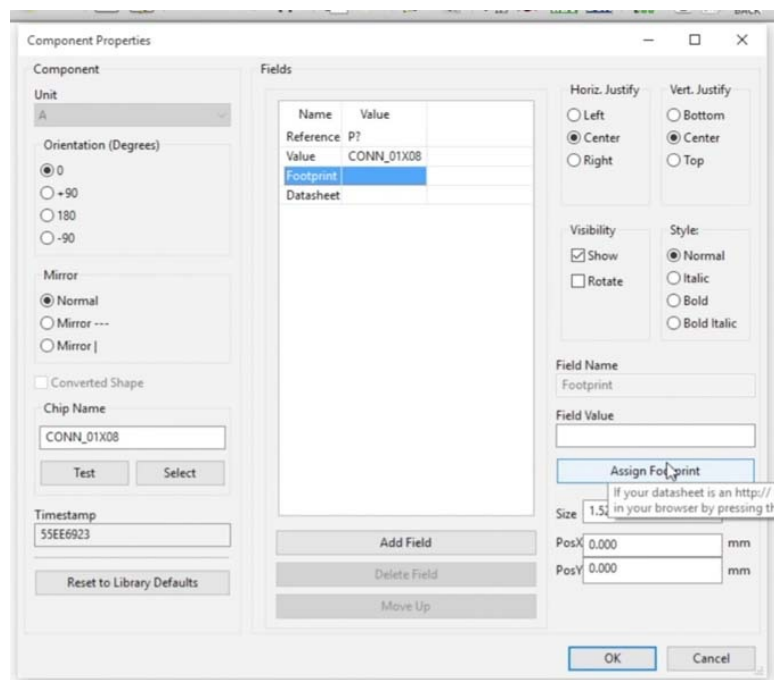
There are several components that are returned. We are looking for a connector that is straight, so it's got a single row of eight pins. The 01x08 connector is what I'm looking for. Click on the connector row in the list and then click “OK”. Drop it somewhere on my canvas. Your canvas now looks like this:



You have just dropped your first component on the canvas!

This connector, just like most components that have pins, has pins that are numbered one to eight. The component also has a unique designator, that starts with a P followed by a question mark “P?”. This indicates that the final unique designator for the component has not been decided yet. Fixing the designators is something we will do later on, automatically. There is also the name of the component, “CONN_01x08”.

You can edit the properties of a component by putting your mouse over it and hitting the E key. “E” stands for edit. You can change its name, and several other values.



Place the mouse pointer over a component and hit the “E” key to show the component properties window.

The parameters for components are set. It’s fine by me. I rarely have to make any

changes to these components if they come out of the library. So I just hit on OK.

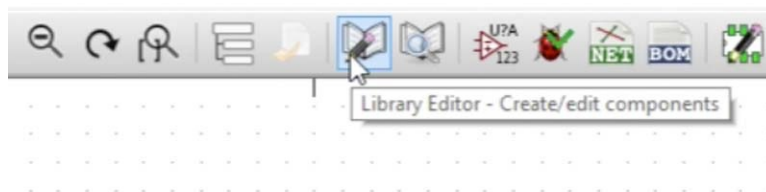
You can move your part around by placing your mouse over the component and hitting the M key. The component is now tied to the mouse pointer and you can move it around the canvas. Click the mouse right button to release the component and drop it back on the canvas. While the component is tied to the mouse pointer, you can also rotate it. Use the “R” key for this. Continue typing “R” until you’re happy with the orientation of your component.

Lets look for the RF24 part round here. Hit the A key to go back to the component chooser and let’s see if something like that exists. Type “RF24” in the filter. Unfortunately, nothing of that description exists in the library. It seems like we have to make a custom built RF24 component for the schematic. We’ll do this in the next chapter.

Chapter 14: *How to create a schematic component*

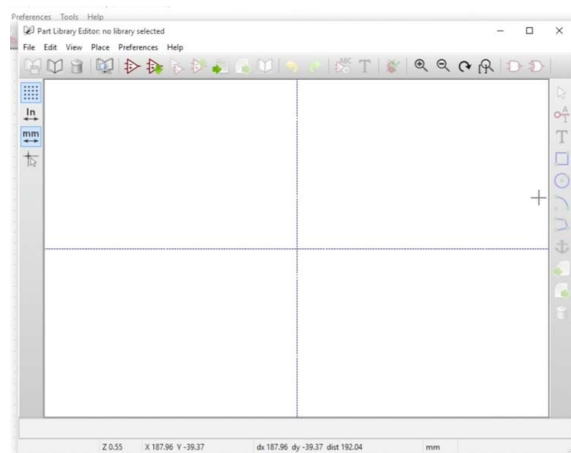
In the previous chapter we searched for an nRF24 part in the schematic library but didn't find anything relevant. In this chapter, I will show you how to create this part from scratch.

To do that we'll use the library editor, this button here brings up the library editor.



To start the library editor, click on the button that looks like a book with a pencil over it.

Click on it and here is the library editor window.

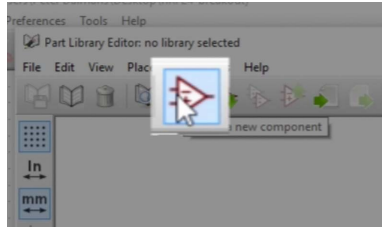


The empty library editor window.

In the library editor window, we will make symbol to represent the RF24 component.

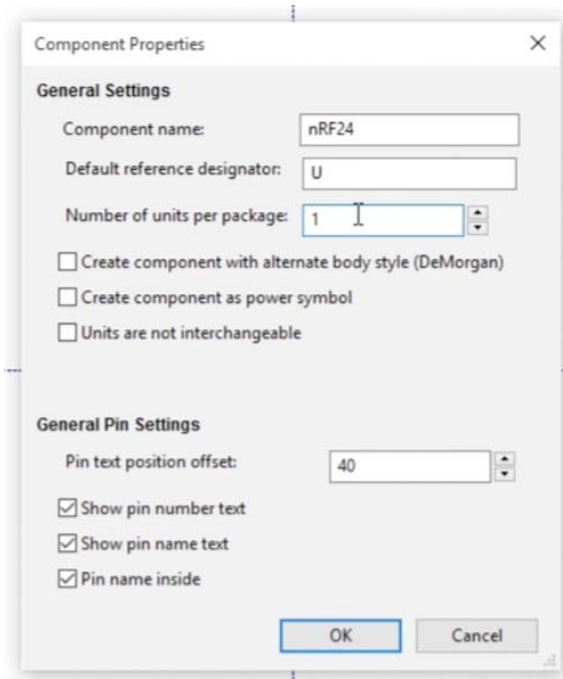
Remember that this schematic only so it does not need to look like the real thing. The objective is just to have a symbol that represents the component and especially its electrical connections. As far as Eeschema is concerned, the symbol is the component.

Components like the nRF24 are typically represented as a box. So we'll create a new part through the part library editor and will start by clicking on the new component button to create a new component.



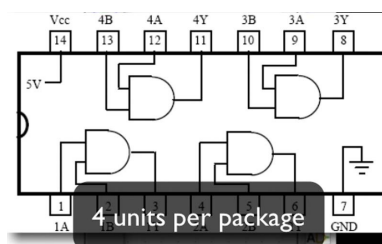
The new component button.

We'll give this new component a name. We'll call it nRF24. We leave the designator like that as a "U" and this package is only going to have a single unit.



The new component properties window.

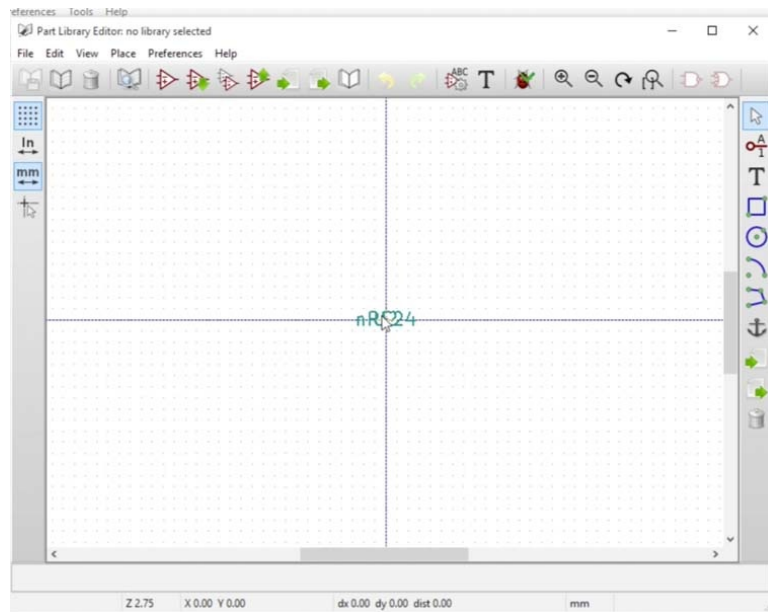
Imagine that other cases of—for example, integrated circuits that contain logical components like gates, for example, and/or gates and you could have multiple of those gates inside a single integrated circuit.



This component contains 4 units per package.

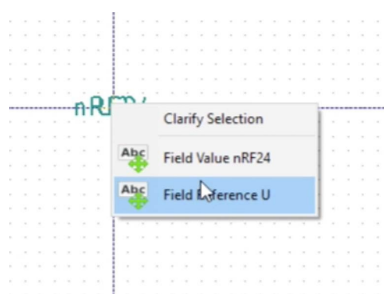
If you have an integrated circuit with two “AND” gates in them, then you may want to indicate that to the user, so you would say “2” in the “units per package” box.

Click “OK” to close the properties box. In the Part library editor window you can see the name of the new component, as we entered it in the component properties window.



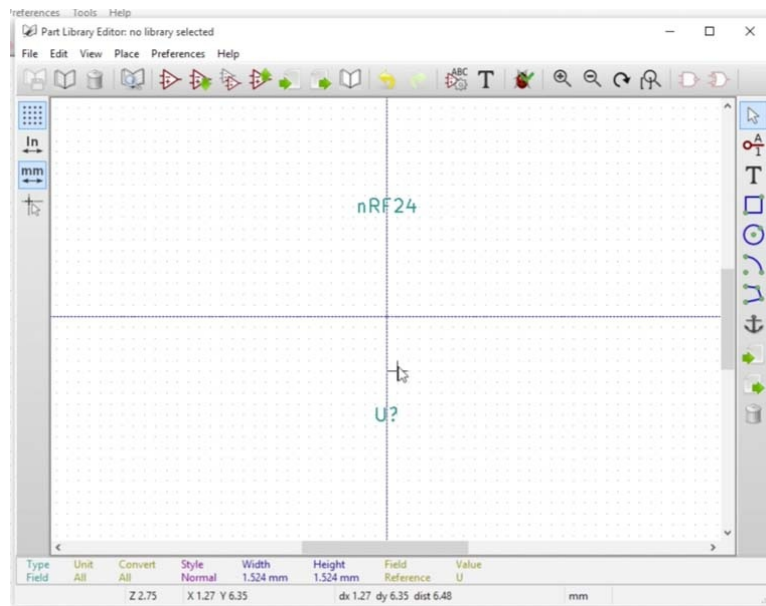
The name of the new component appears in the centre of the part editor window.

We can make use of the same shortcuts as in the main schematic editor window. Because we have two text labels, one on top of the other, we should separate them by moving them. We can move a component by selecting it with the M key. Put your mouse cursor over the text and type “M”.



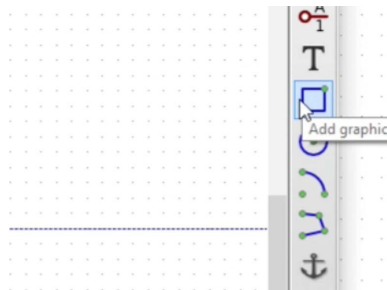
Kicad does not know which label I would like to move, so it asks for clarification.

Because there are two labels under the mouse cursor, Kicad is asking me to select which field or which label is it that I want to move. Choose the first one and move that up the top and I’ll take the second label and just move it down here. Center the component where the two lines are intersecting, just to make it a bit more symmetrical.



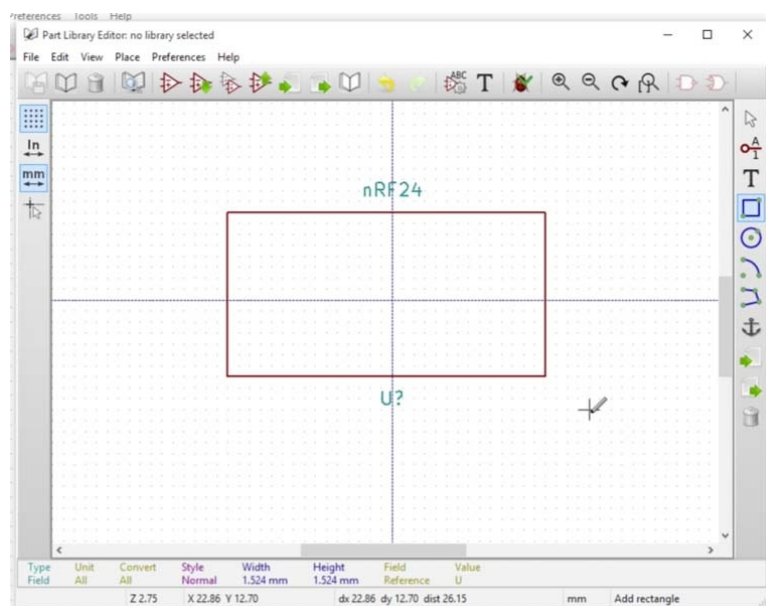
The two labels are separated.

Next, we will create a frame for the RF24 custom component.



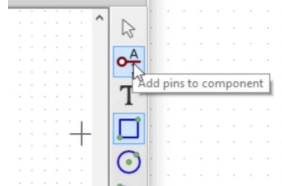
Choose the rectangle button to draw a rectangle.

Choose the rectangle button and then draw the rectangle by adding a line starting from the top left corner of where I'd like the rectangle to be. Right-click there and then drag a line to the bottom right corner where I'd like the rectangle to end.



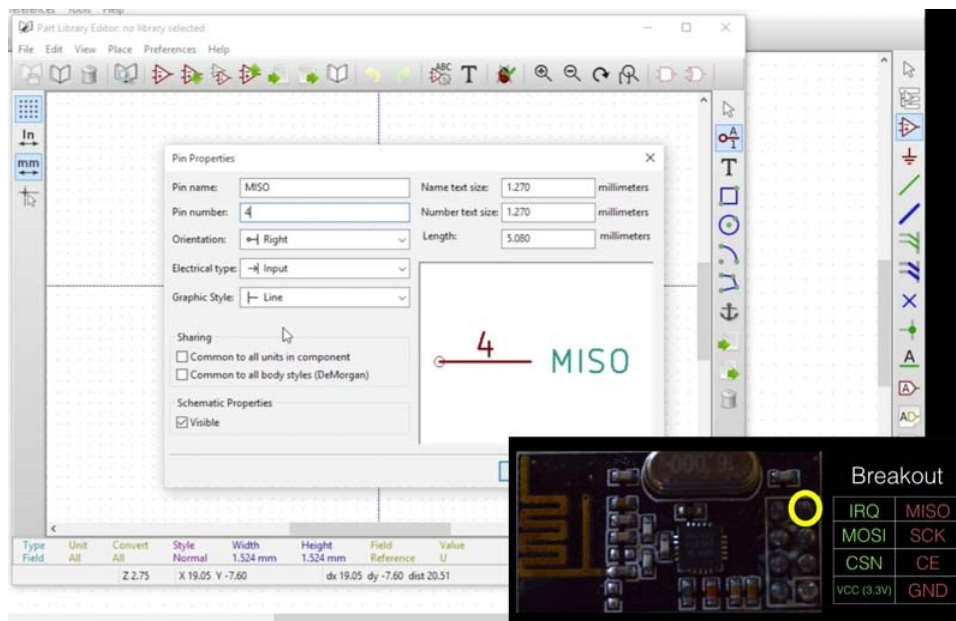
This rectangle represents the nRF24 component, but without the pins added.

Next, we must add the pins. As you know, the actual RF24 part has its pins arranged in two rows of four pins each. To make this schematic more readable, we will add connectors on one of the four sides of the box we just created, in a single row. There is no need for a one-to-one match with the real actual component – the real life component. Later, we'll connect these pins to the part footprint which we'll also have to create from scratch and that will be modelled after the real, physical RF-24.



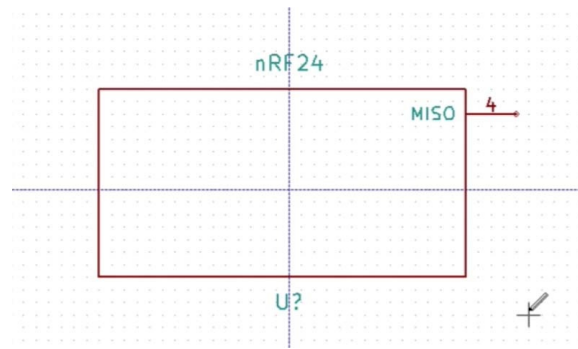
You can click on the “Add pins” button to add a new pin.

To add pins, you can either click on the “Add pins” button or hit the P key. The pin properties window will come where you can type in the pin name.



The pin properties window.

You should look at the original part pin descriptions so that you can properly name the pins in your custom component. And on the original nRF24 part, “MISO” is pin number four. The orientation is going to be to the right. The pin type should be input and the graphic style should be a line. When you are finished with these edits, click on OK to exit the editor.



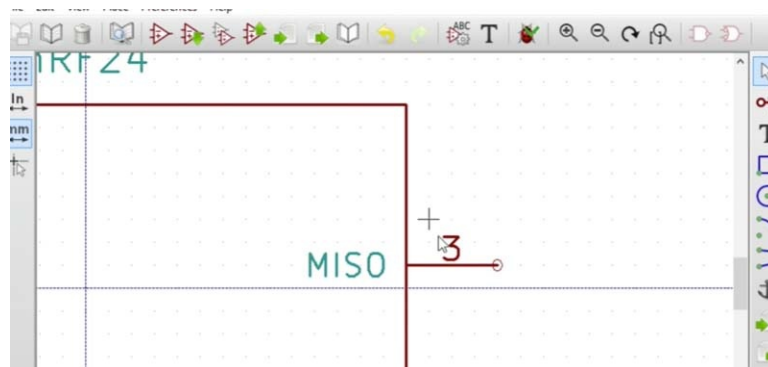
The new custom part, not complete yet

In the editor window, you can see the new MISO pin #4 added to the schematic of the custom part. Notice the little dot on the edge of the line? this is where wires will connect eventually to/from other components. This of the dot as the terminal for the pin. The line should be placed so that the dot is away from the border of the box. Also, the number of the pin, in this example “4”, corresponds to the number of the MISO pin on the real part. Consistent numbering makes it easier to figure out which pin is which eventually when you got to connect everything together.

If you want to make a change to this pin, you can just edit like any other component. Place your mouse over it and then hit the E key. This will bring up the pin properties so you can make any changes here.

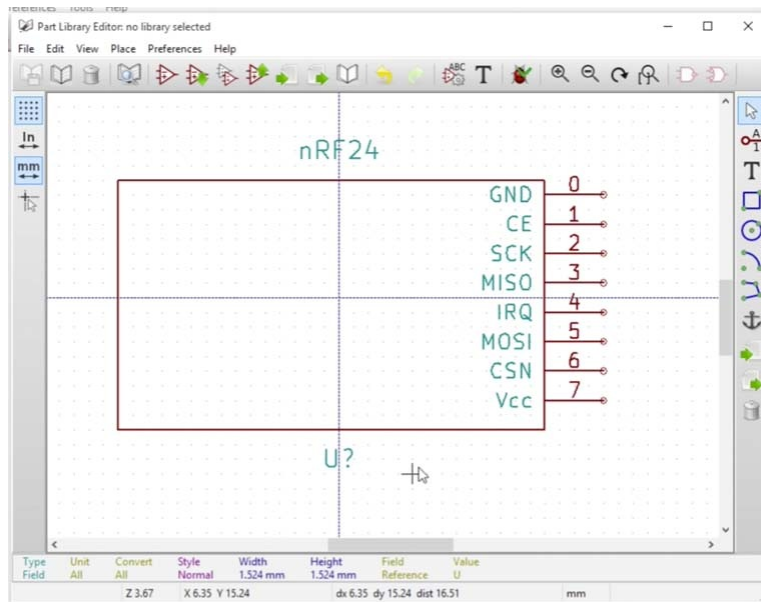
I have actually just spotted an error: in the real part, MISO is actually pin number three. I am also going to move this pin a bit lower so that there is enough room above it for the other pins.

Here is the current state of the custom component:



The new custom part, current state.

Let's create the rest of the pins. Repeat the process described above 7 times, so that in the end, you have a schematic like this:

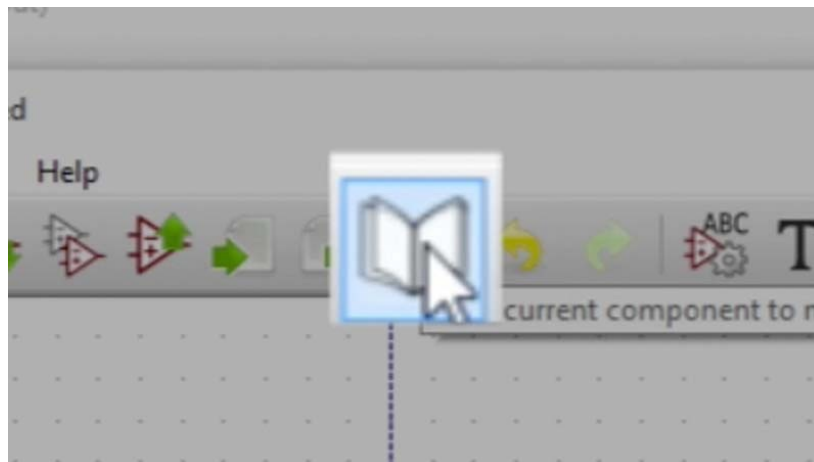


The new custom part, completed

Remember that you can use the M key to move things around and arrange everything so that the component looks symmetrical

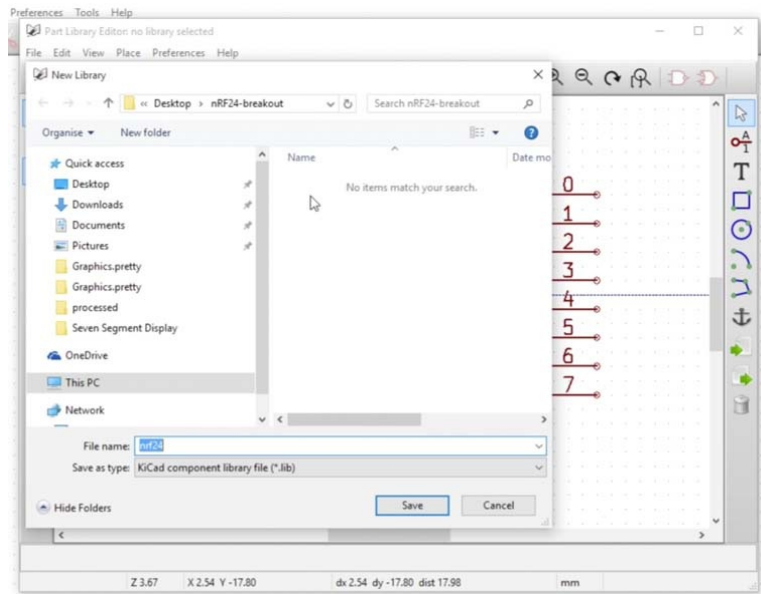
Now we need to save this part, but before we can save it, we must choose or create a working library. In KiCad, you cannot have a schematic component on its own. A component must be a member of a library, even if the library contains only a single schematic component.

Let's create a new library. To save the component to a new library, click on this icon here, looks like a book.



Click on this button to save the component to a new library.

Next, select the location on your computer for the library. I recommend that you place this library in the same working directory as my project, so NRF-24 breakout, and the naming of the library—let's call it "nRF24_schematic_library".



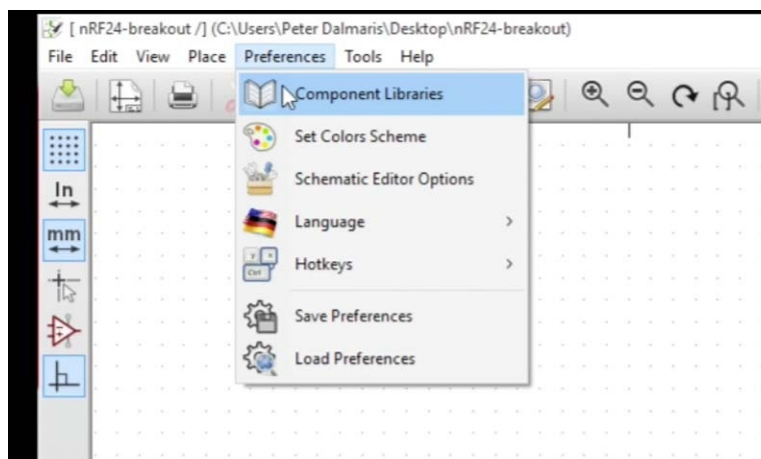
Store the new library within the project folder.

Click on the Save button to finish the process.

The new library and its content will not be available until it is loaded by Eeschema. Creating a new library will not automatically load it in Eeschema. You have to explicitly go into Eeschema and add this new library to the list of libraries that it has access to. This is something that people get confused about and the way that the libraries work in Kicad is not very intuitive, to say the least.

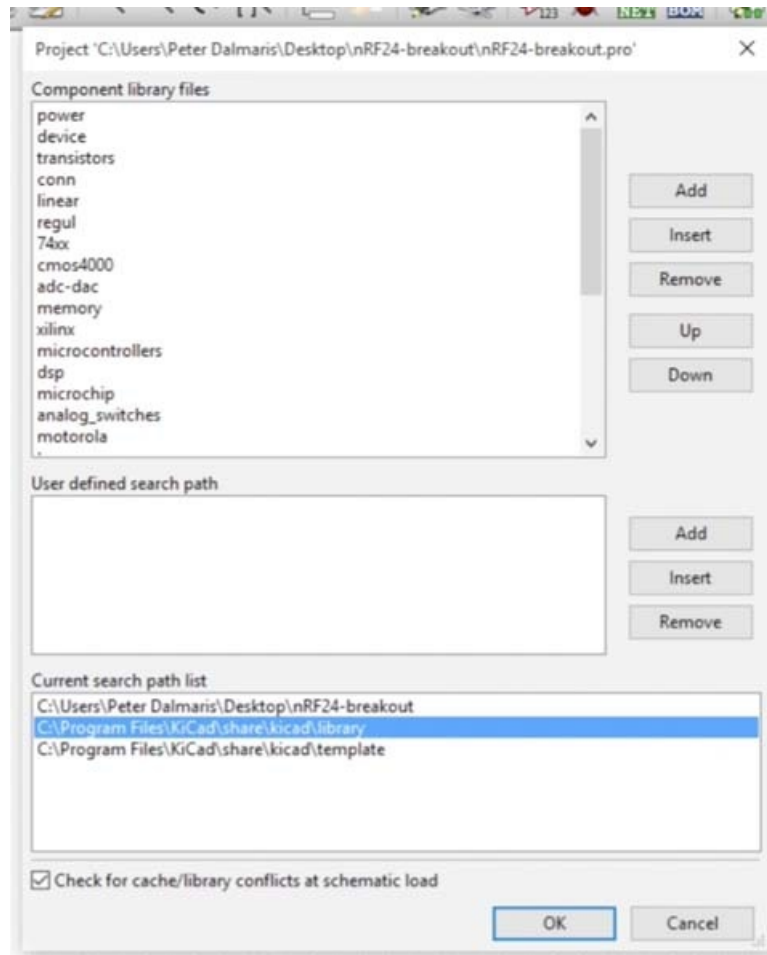
And at this point you've got a new library and in it you have a new component. You've got the NRF24 schematic component. We can get out of the library editor and now we'll have to add that library that we've just created to the project so that we can access the parts in it.

In Eeschema, go to preferences and click on component libraries.

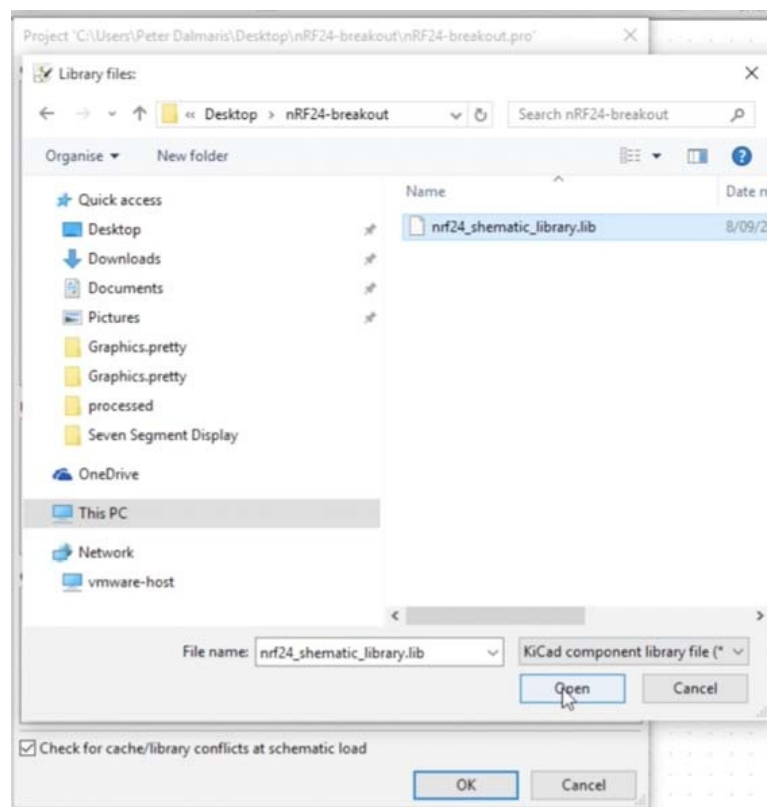


To add a new library to Eeschema, go to Preferences and the Component Libraries

Here, we will add the library that we just created. Click on the Add button, and browse to the location where you saved the new library.



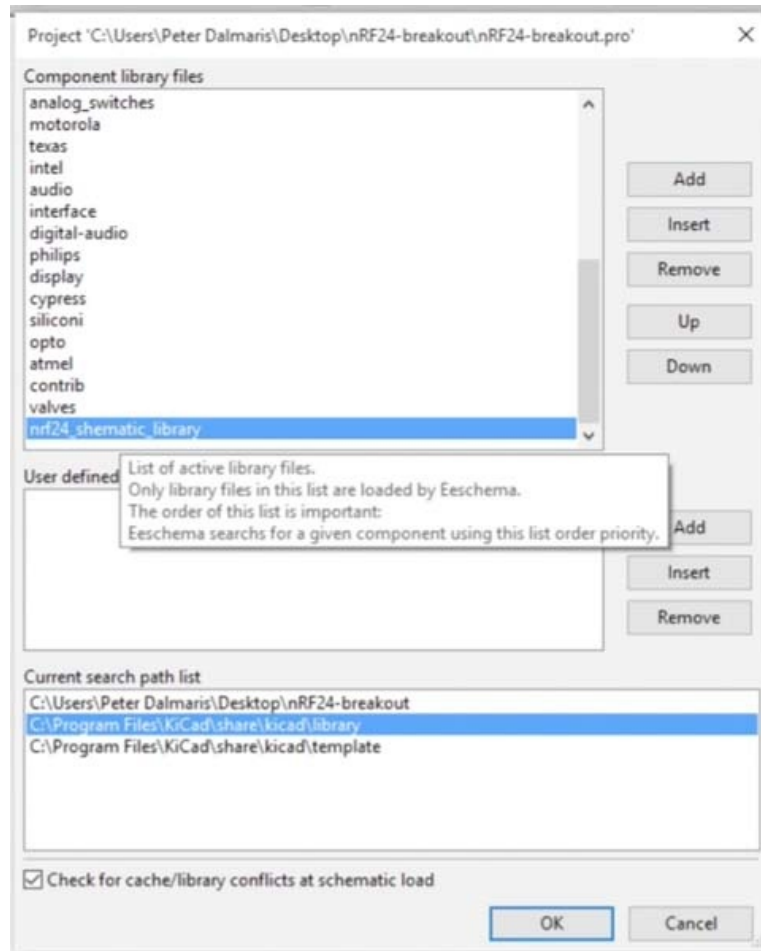
Click on Add to add the new library.



Browse to the location of the new library, and click Open.

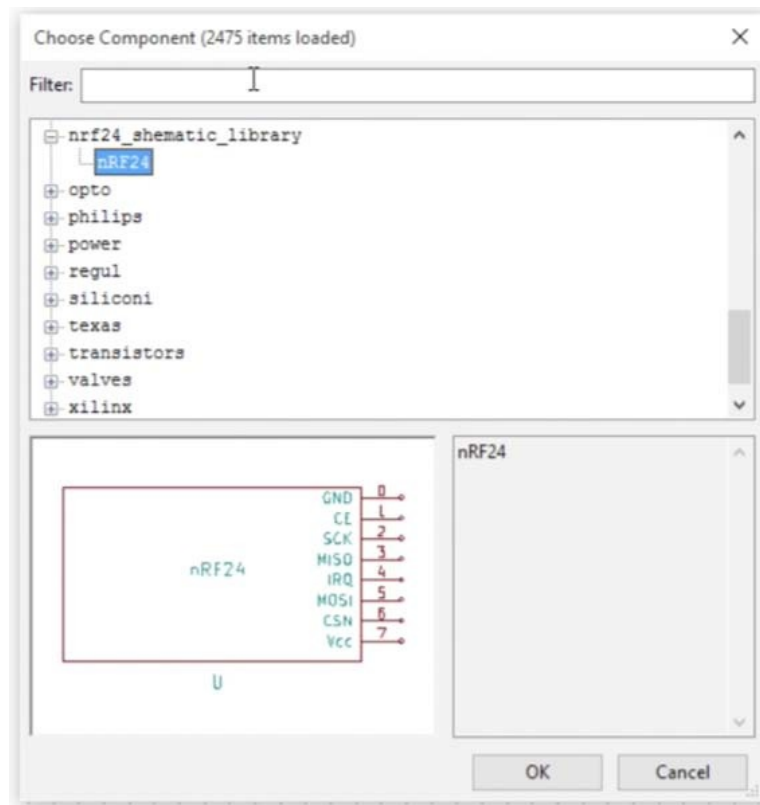
Scroll towards the bottom of the libraries list in the libraries editor, and you should see

the newly added library.



The new library appears in the bottom of the libraries list.

Let's drop the new component to the canvas. Press the A key, and search for the library that we just created and here it is. You can drill in it and you can find the part that we just built. Or you can just look for it using the filter. Select the component and click OK to add it to our schematic.



The new component is in the component chooser, from where you can add it to the schematic.

In the next chapter, we will do annotation of the parts in the schematic.

Chapter 15: *Wiring*

In this chapter we will connect the pins from the nRF24 component to the connector. There are two ways to do this. The first one is by using individual wires and just wiring each pin with its counterpart. The second method is to use labels. We can label each pin pair with a unique name, and Kicad will automatically connect those pins with the same name.

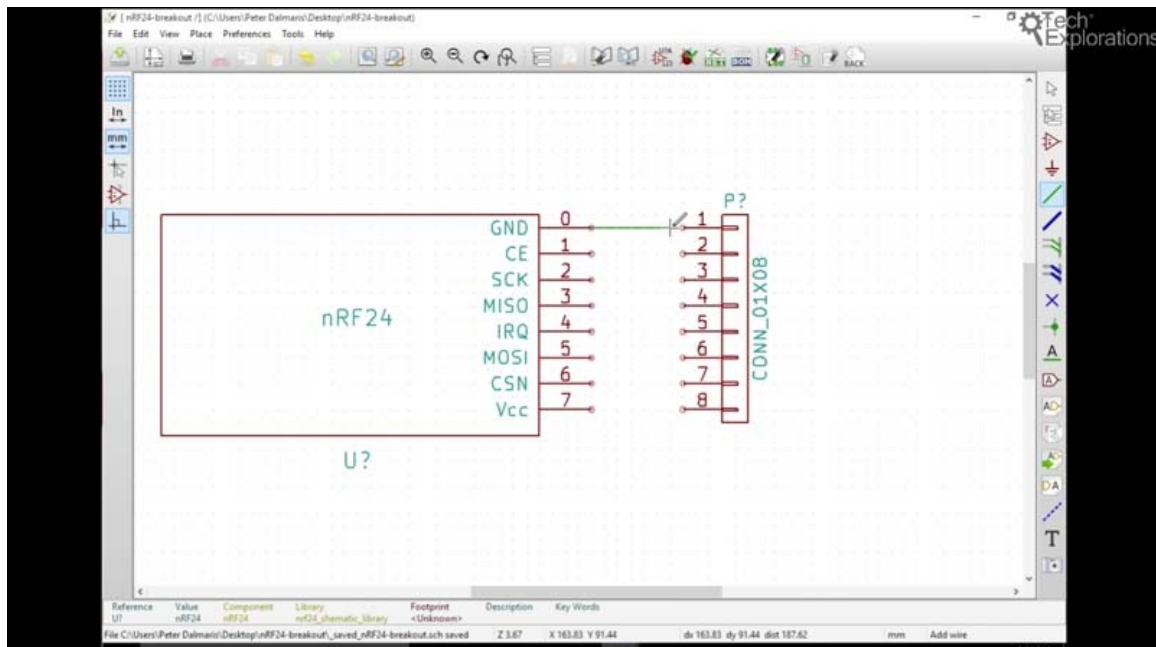
In this chapter, we will use the first method. In the next project, we will use the second method.

To create a wire you can use press the “W” key (“W” for “Wire”), or you can click on the wire button.



Click on the Wire button to go into the wiring mode

I prefer to use the ‘W’ key instead of clicking on the wiring button. So, I hit the ‘W’ key, you can see that a wire now is being drawn.

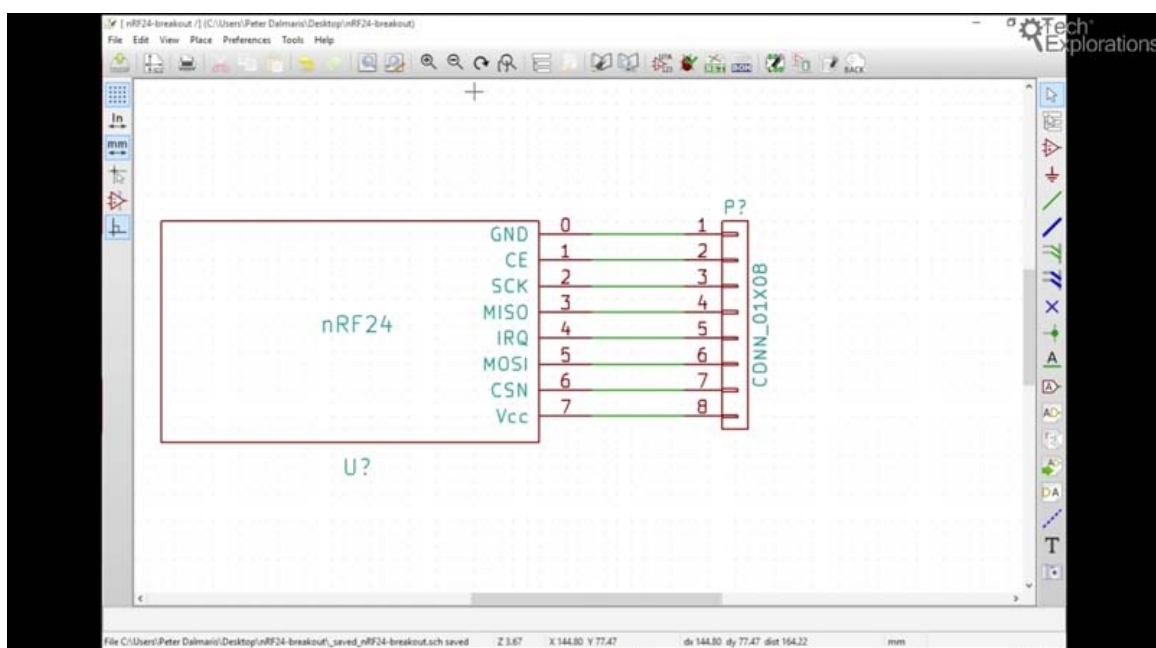


After typing “W”, click on a pin circle to start a wire, move the cursor to the pair pin’s circle terminal and click again to finish the wire.

Click again to end the wire. Here is the process again:

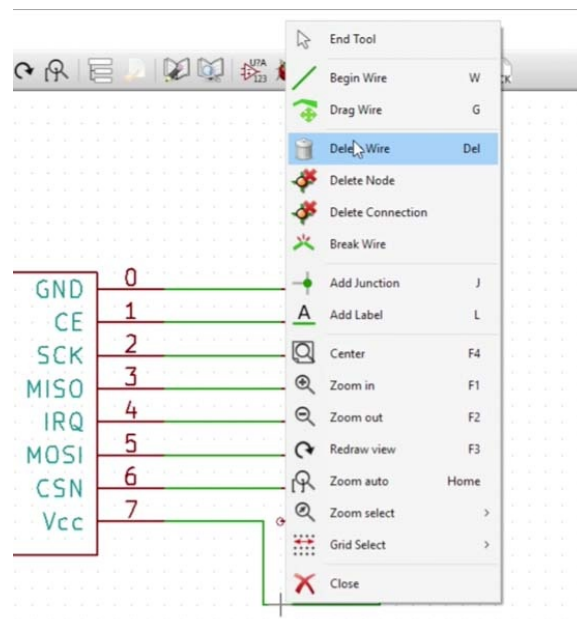
First you put your cursor over the pin, the circle of the pin that you’d like to start a connection from, then you hit the ‘W’ key, then a wire has already been drawn and is just following the cursor of your mouse. You just drag that along without having any buttons down, or any mouse buttons down, until you reach the pin where you’d like the wire to end. And you click again to finish the wiring.

Do the same thing with the second pair of pins. Repeat the process again for each pair of pins. At the end of the process, your schematic should look like this:



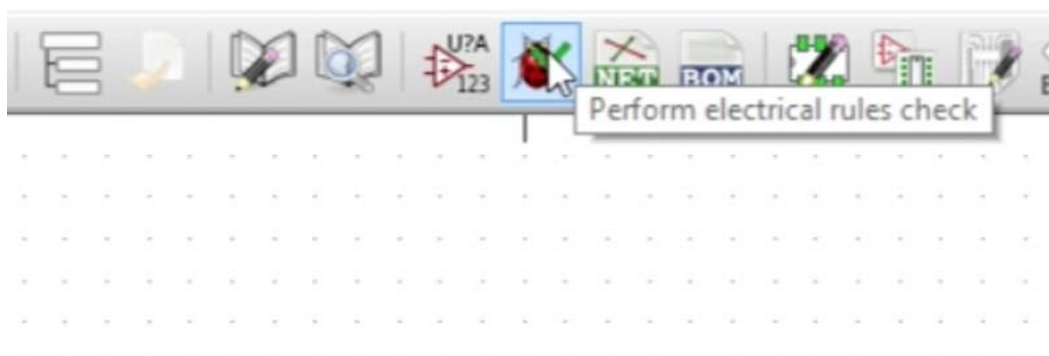
After making all the wirings, your schematic should look like this.

Another thing to notice as this point is that once you hit the ‘W’ key once, you’ve already selected the wiring tool. From that point onwards, you don’t really need to continue hitting the ‘W’ key. It’s enough to just click and start the wiring, then click to finish the wiring. So, one click to start, one click to finish. If you make a mistake, it’s not a big deal. You can just delete or undo the error and try again. To delete anything on the canvas, right click on the object to reveal the context menu, and then choose the Delete option.



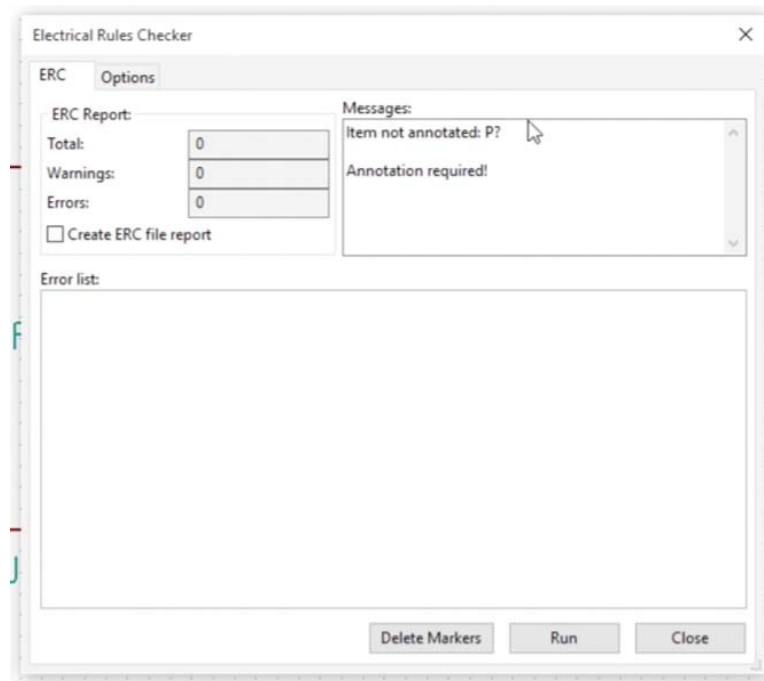
Right click on an object to reveal the context menu. You can then choose to delete that object.

If you make an error but don't realise it, you can get Kicad to find it. Let's pretend that you have made an error in your wiring, and have left a pair of pins un-wired. Kicad has a function called the "electrical rules check", or ERC.



To do an ERC, click on the bumblebee icon in the top bar.

Click on the ERC button. The ERC dialog box will come up. Click on 'run' and do the test.



The ERC has revealed that I forgot to do the annotations.

The ERC, in this example, is telling us that we forgot to do the annotations. It's ok, we will do this in the next chapter.

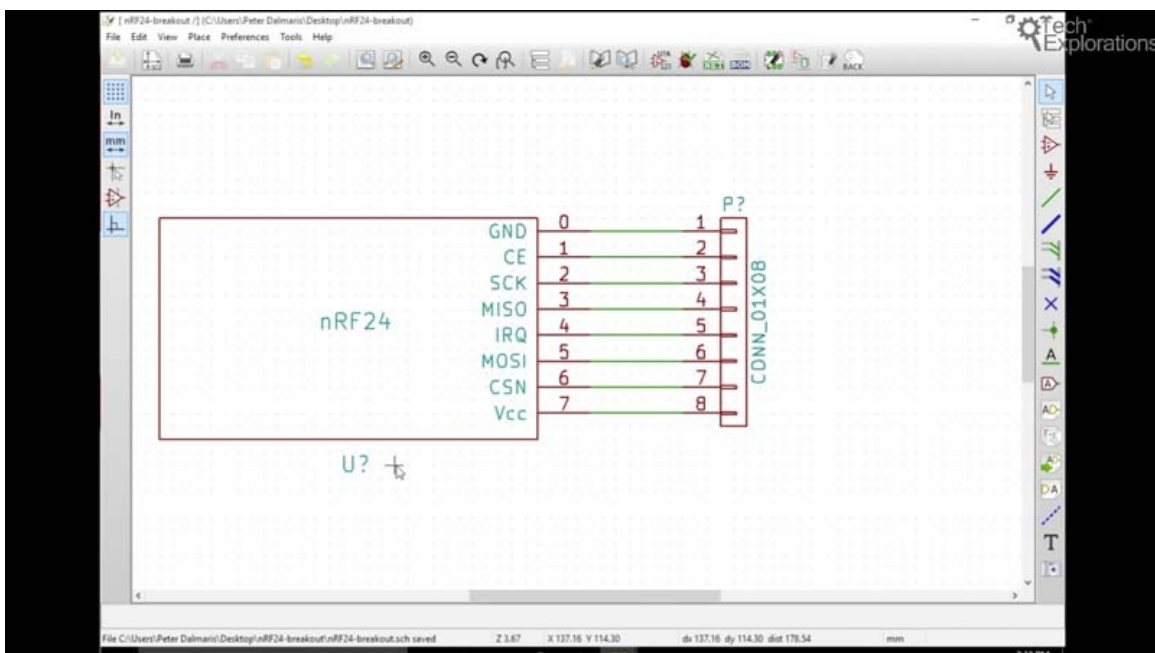
Let's save the project and the schematic at this point. Hit the 'save schematic' project.

In the next chapter, we will annotate the schematic.

Chapter 16: *Annotating the schematic*

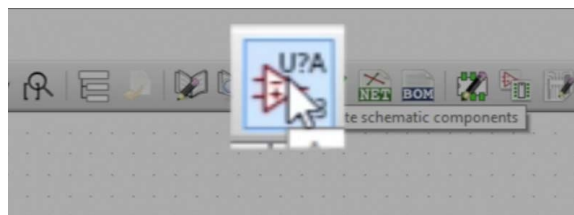
In this chapter I will show you how to use Kicad's automatic annotation tool.

We've got two components that are both are not annotated yet.



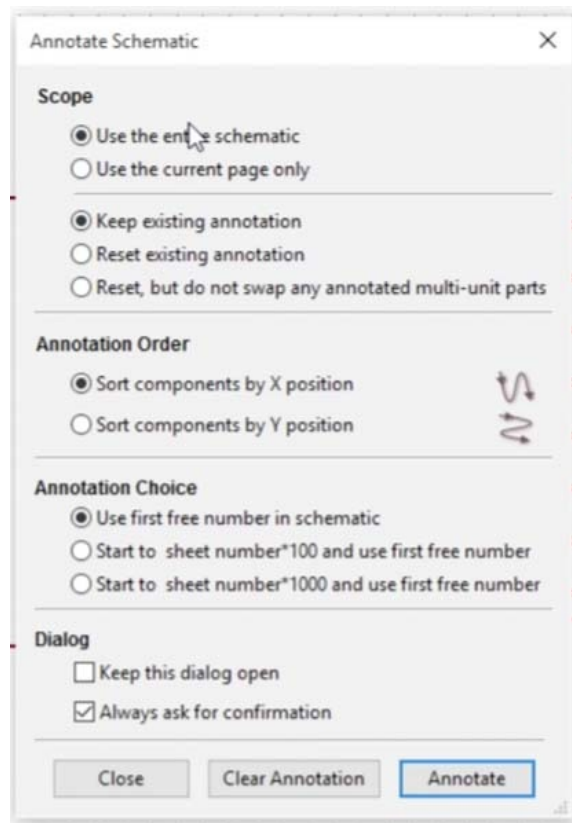
The two components are not annotated. Notice the “?” in their designators.

Notice the question marks in the component designators, “U?” and “P?”. To do the annotation, we will use the annotator.



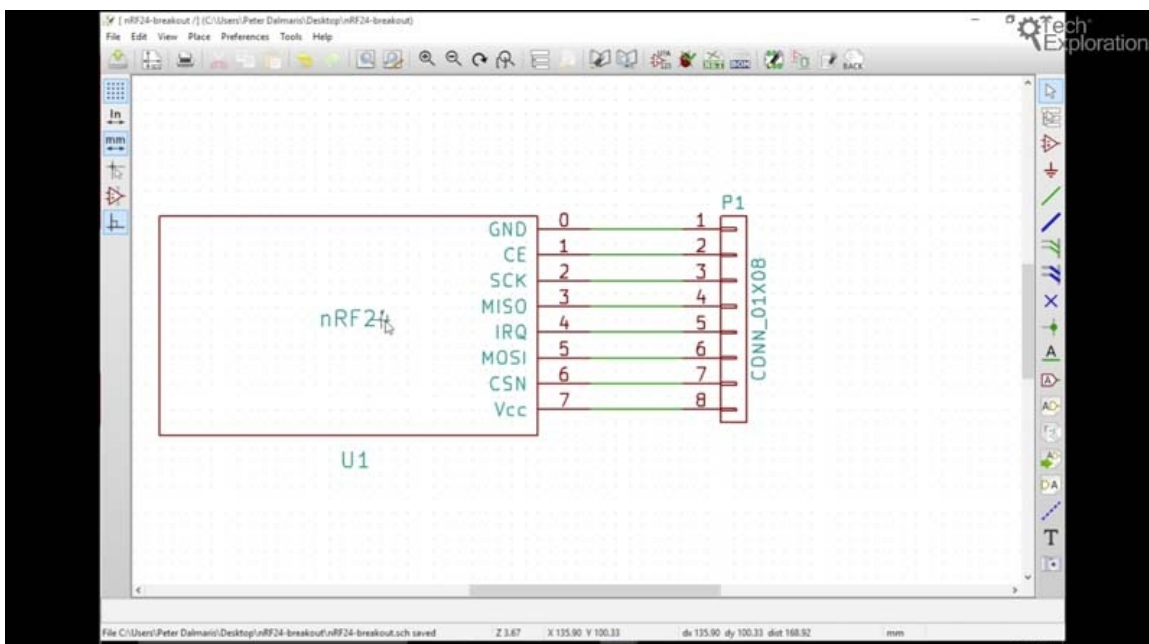
The annotator will automatically annotate components that are not already annotated.

Click on the annotator button. There are various parameters that control the way by which the components are to be annotated, however the default settings work fine.



The default annotator settings work well.

Click on the Annotate button to do the annotation. Here is the result:



The annotator automatically replaced the question marks with numbers so that each component has a unique designator.

Okay, and now you can see that the question marks have been substituted with actual

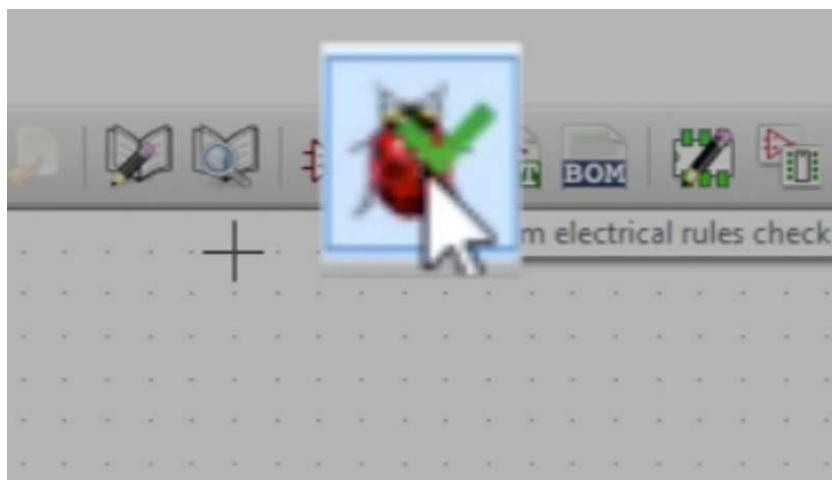
numbers. The NRF 24 component is U1 and the connector is designated P1.

In the next chapter, we will do an electrical rules check. This will complete the process of creating the schematic for our new board. Once we confirm that there are no faults in the schematic, we will proceed with the board layout and routing.

Chapter 17: *Electrical Rules Check*

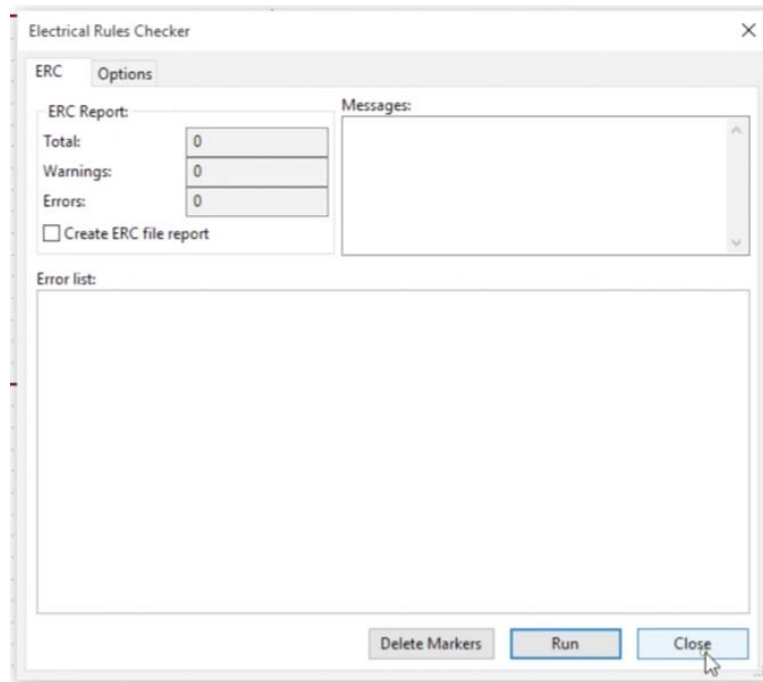
The only thing remaining to be done before we can complete the process of designing the schematic for our new board, is to do an electrical rules check.

To do the ERC click on the bumblebee button, in the top menu bar.



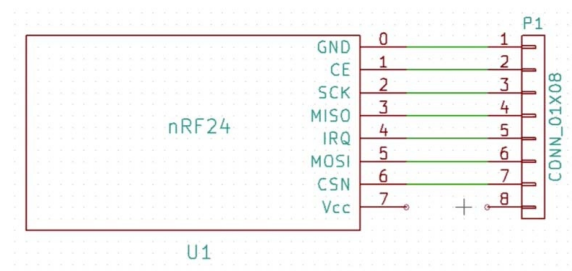
Click on the ERC icon to start the test.

Then, click on the Run button to execute the check. There is no output, which means there are no faults.



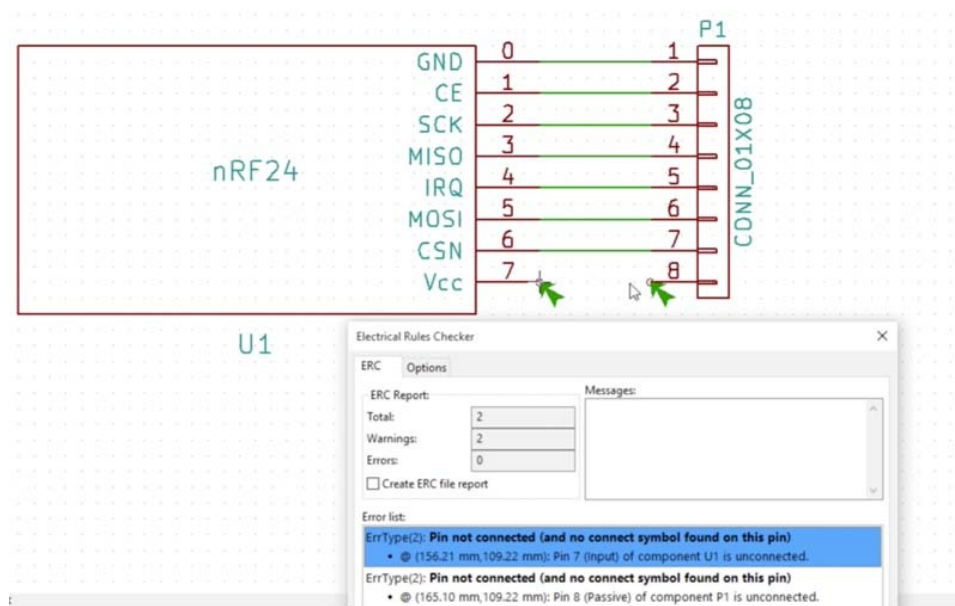
No news is good news!

What if we actually had an error? Let's try this out. Let's delete one of the wires, so that the schematic is like this:



Pin 7 on the nRF24 is NOT connected to pin 8 of the connector.

Do the ERC again. Here is the result:



The ERC has found two unconnected pins, and it indicates them with the green

arrows.

The checker is telling us that we forgot to do the connections for two pins, which were left unconnected. It uses arrows to help us find the problem and fix it. Let's fix it. Use the W key to go into wiring mode, and connect the VCC pins. If you try the rules check again you will be able to confirm that the problem is now fixed. No output is a good thing.

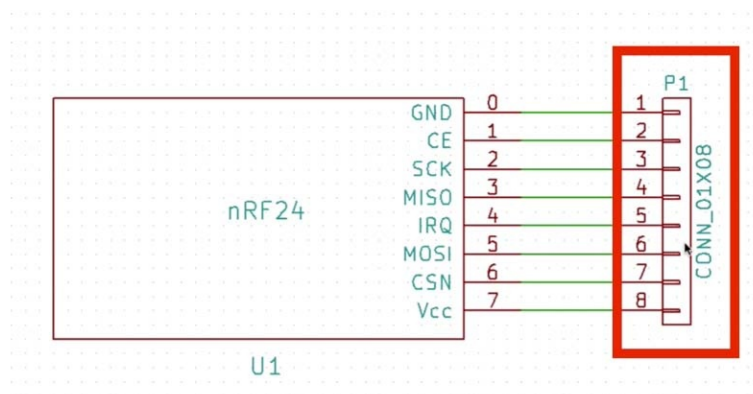
Close the checker and save the project.

Let's recap what we have so far. We've started a new project in Kicad. We have created a schematic. Our schematic contains a connector which is a one row by eight pins and we have created a custom part because it didn't exist in the schematic library. This part represents the NRF24 component. We have completed the wirings between the two parts and ran the electrical rules check. The check confirmed that the schematic has no faults.

We can now move ahead and start work on the PCB itself. We've saved our project and in the next lecture, we'll start work on the PCB.

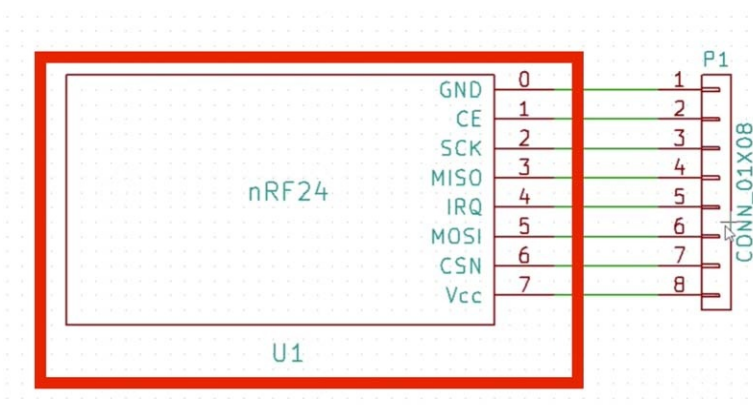
Chapter 18: Associate components to footprints

Unlike other PCB design tools, in Kicad, schematic components are not automatically linked to a footprint. This is of a footprint as the outline of a component that is mounted on a circuit board. It contains the outline of the pins, the device, and often text markings with the name of the device or its values. In Kicad, we must associate schematic components to footprints using a tool called Cvpcb.



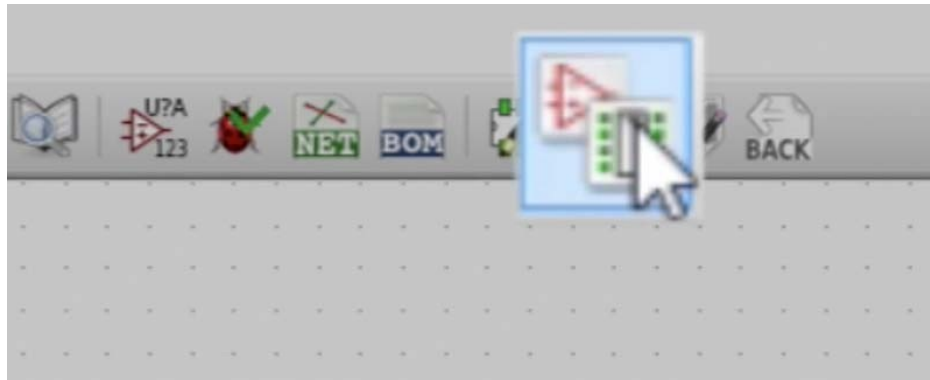
The straight 8-pin connector has a footprint to which we can associate the schematic component

The straight pin connector that we've got on the schematic already has an associated footprint in the library and we'll simply select it for that part.



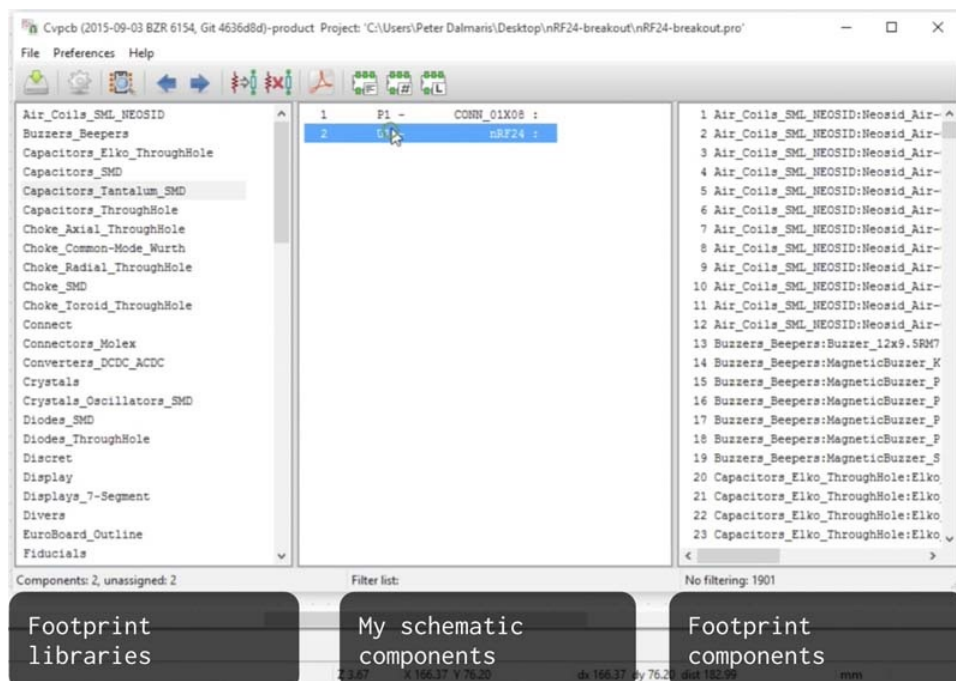
There is no footprint to associate with the nRF24 component, so we will need to create one.

However, the nRF24 part is a custom component and it doesn't have an existing footprint in the library. We will have to create a new one for that.



Click this button to start Cvcpcb

To start Cvcpcb, click on the Cvcpcb button. It takes a few seconds sometimes for all the panes to be populated with data and records because, in the background, Cvcpcb is accessing the Kicad repository on GitHub.com for schematic components and footprints.



The Cvcpcb tool allows you to find footprints via browsing or through filters.

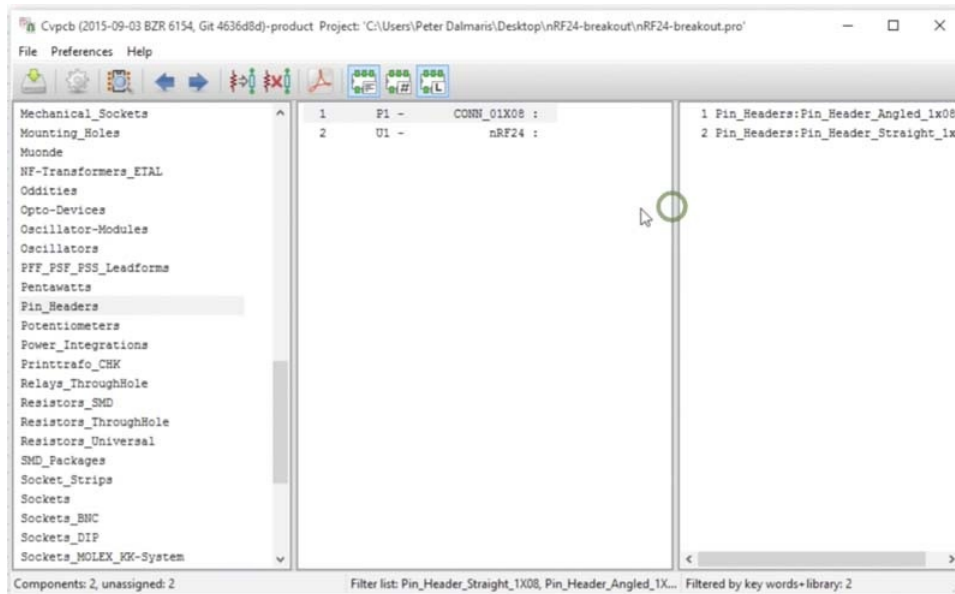
On the left pane are the footprint libraries. On the right side, are the contents of each library that is selected at a given time. And in the middle pane are the components from the current schematic. You can see here that in the middle pane we have two components: the connector component (P1), and the custom part, nRF24.

For the connector component, look through the contents of the “connect” library. Once you click on the Connect library in the left pane, you will see in the right pane all of the components from all of the libraries. This is not very useful. I wish to narrow that down to just the parts that are a member of the connect library.

To do this we will use the three filter buttons, found in the right side of the top menu bar. Click on the filter marked “L”. In the right pane we now have only the footprints that are members of the Connect library. We can also narrow down the hit list further. Select the straight connector component in the middle pane, and then click on the “#” filter. This

will return only those components that match my selected component by pin count.

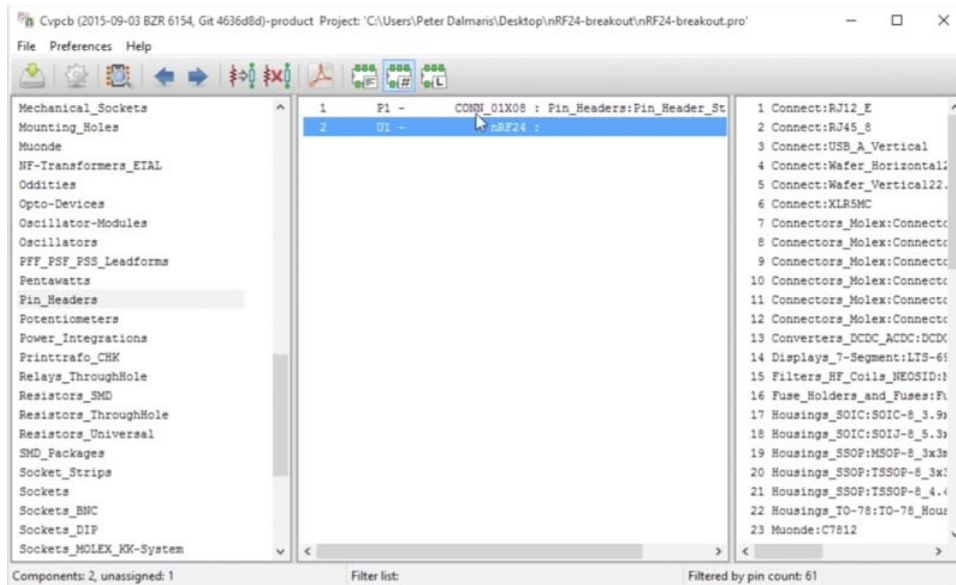
All of the members of the connect library that are displayed are those with eight pins because we have enabled the pin count (“#”) filter. Finally, we can also use filter with the text document in its button icon (third from the right), and narrow down the hit list by keyword. Once you click on that, notice that the right pane is now empty. This means that there is no component in the connect library that has the name con_01x08. Therefore, we were incorrect in looking in the Connect library for this particular footprint.



The straight connector footprint is in the Pin_Headers library. Notice the enabled filters, at the right side of the toolbar.

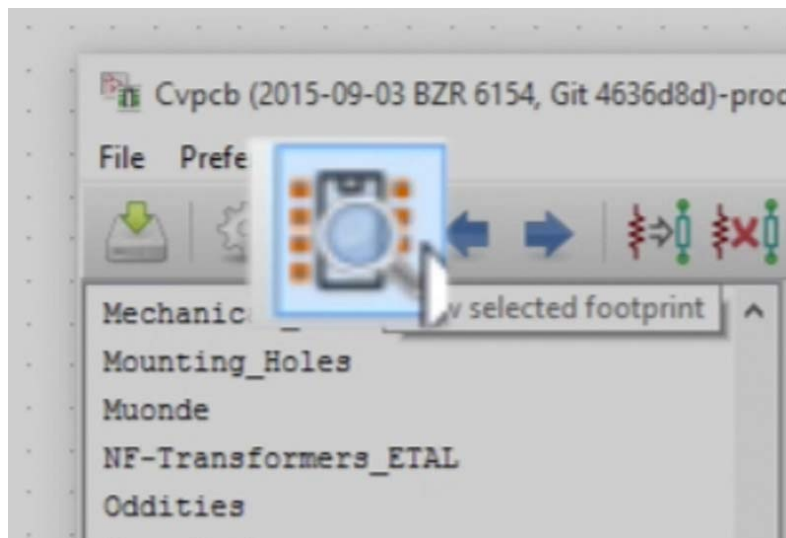
Let's look at something else. Where could the needed footprint be? We can check inside Sockets. Or in Pin Headers? Yes! Once I select Pin_Headers library, and with the schematic component selected in the middle pane, CvPcb will give us only those items in the Pin_Headers library that match the number of pins and the name of the component that we are looking for

The one that I would like to go with is the header straight 1x08. So, double-click on this record in the right pane to select it.

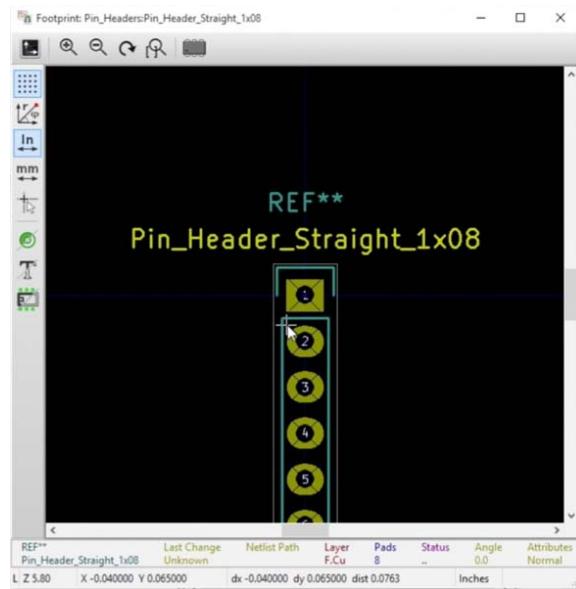


Once you double-click on the required footprint on the right pane, and with a component selected in the middle pane, the footprint and the component will become associated

You can also preview a footprint, and double check that this is what you need. To preview a footprint, click on the preview button.



Click on the Preview button to see a preview of the selected footprint.

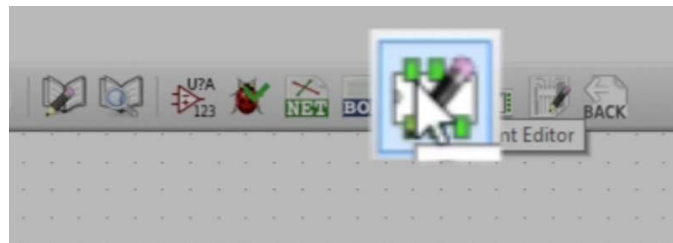


A preview of the 8 pin straight connector

As for the nRF24, there is no component available in any of the libraries. We will have to create one. Save the current associations and let's work on a custom footprint for the nRF24 component in the next chapter.

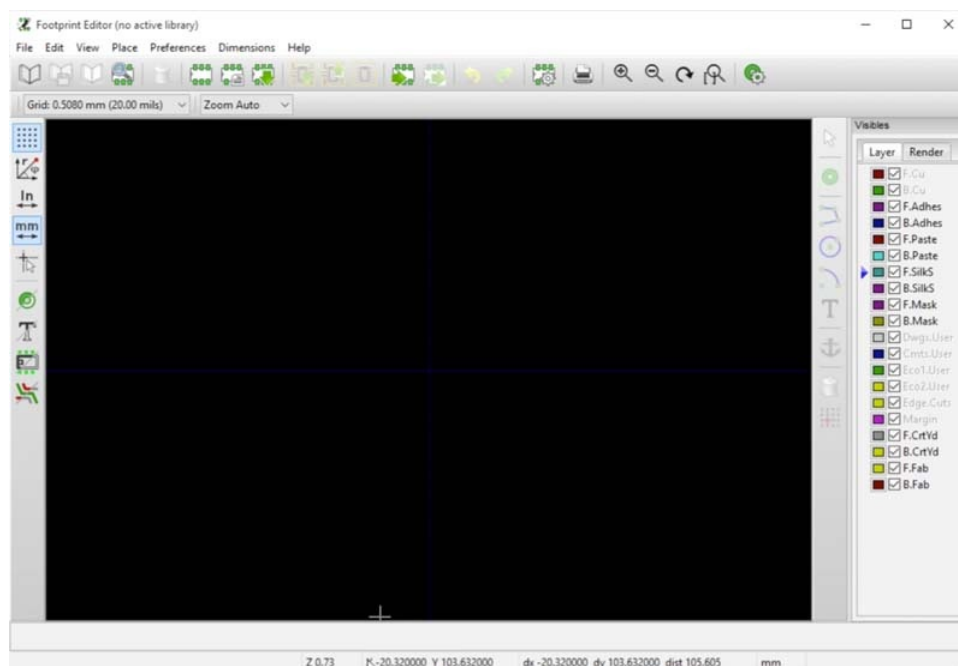
Chapter 19: *Create a custom footprint*

In this chapter we will create a custom footprint for the nRF24 schematic component. If you haven't done so already, close Cypcb. Next, start the footprint editor.



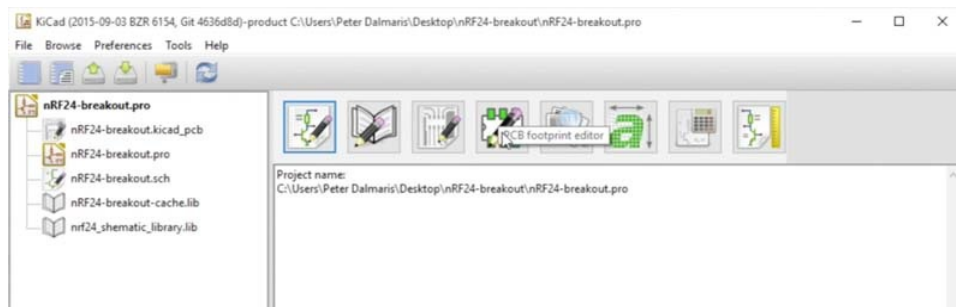
From Eeschema, start the Footprint editor by clicking on the button with the IC and the pencil icon.

So click on the footprint editor button, and the blank canvas of the editor will come up.



The blank canvas of the new footprint editor.

You can also launch the footprint editor from the main Kicad window. To do this, go to the main Kicad window and click on the fourth button from the right. Notice that it contains the same icon as the one in Eeschema, with the IC and a pencil over it.

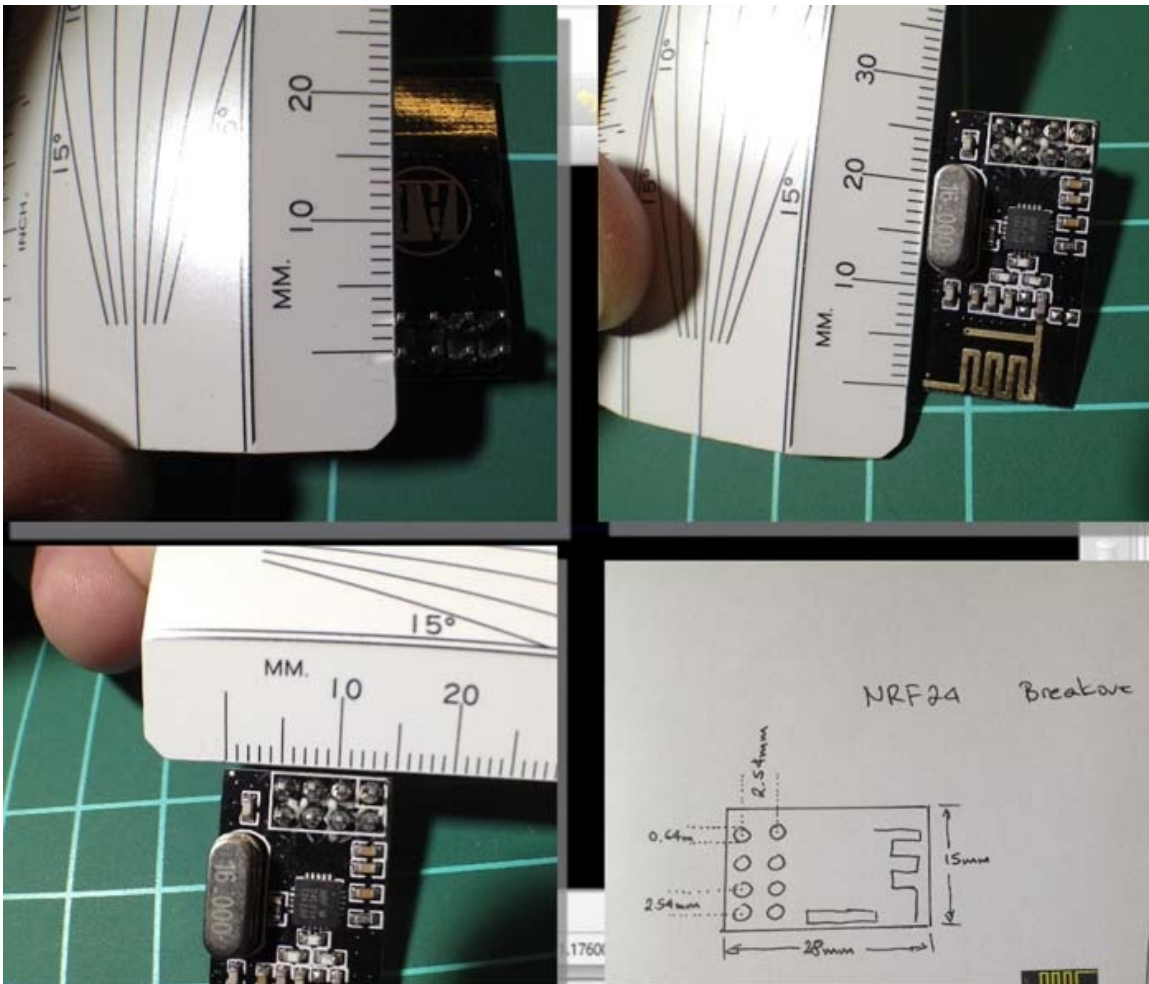


You can also start the Footprint editor from the main Kicad window.

Either way, you can reach the exact same footprint editor tool

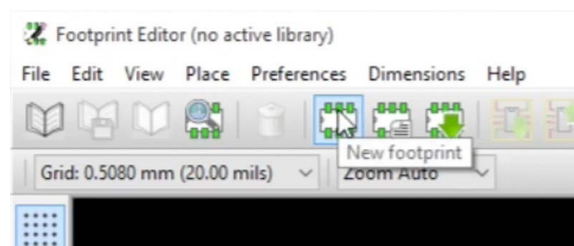
We need to know as accurately as possible the dimensions of the pins of the NRF24. The circumference of each pin, the distance to the neighboring pins, clearances from the edges of the board, etcetera.

You can do this by looking at online documentation for the particular part or if you have the part, by using your own ruler. I have measured a 2.54 millimeter pitch, which is the distance between the pins. I measured the distance on my NRF24 with my ruler, and it looks exactly correct. The NRF24 has two rows of pins. Again, using my ruler, the distance between the rows looks to be exactly 2.54 millimeters.



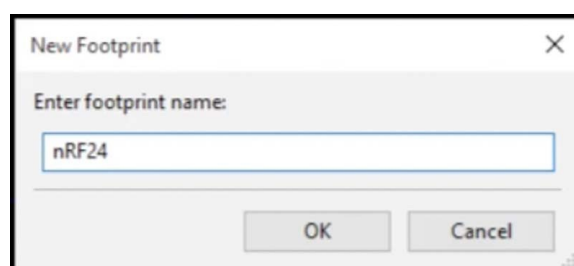
If you can't find the exact measurements of the pins and the part that you are modelling in the editor, use your ruler to find out.

The circumference of each pin is also measured at 0.64 millimeters. I have created a hand-drawn version of the schematic with the measurements as I worked them out. We will create something like this now in the footprint editor.



The new footprint button

In the footprint editor, we would like to create a new footprint so click on the new footprint button and give a new footprint name, like NRF24.



Name the new footprint

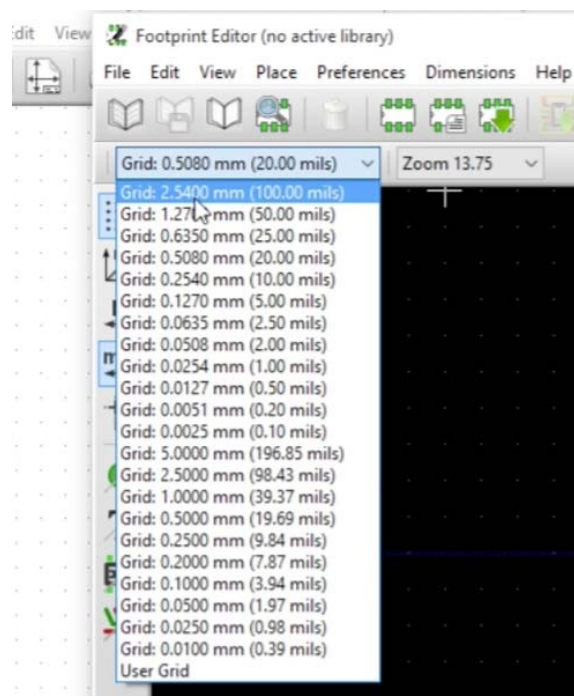
Click ok, and see how the canvas now contains the name of the footprint and a reference designator.



Creating the new footprint. We need to add boundaries and pins.

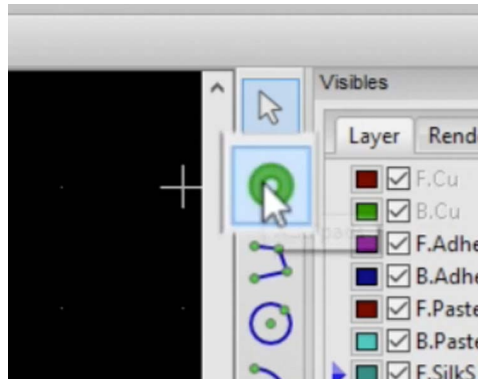
Move the two text labels so that there is enough space between them for the boundary. Use the M key for this, just like you moved components in Eeschema.

Let's continue with the holes for the pins. We need eight holes. We need to select holes that are wide enough so that the pins of the actual component will fit through them and that the exact distance from the adjoining pins as we measured earlier. So, let's start with a pitch. We know that the pitch (distance) between the pins is 2.54 millimeters, so to make it easy to space the pins at this exact distance, we'll set the grid to be this size.



You can change the size of the grid by selecting a value from the grip drop-down menu.

The setting for the grid can be changed via the grid drop-down menu, in the left side of the tool bar. Click on the drop-down menu to open it, and select the grid to be exactly 2.54 millimeters. Now, we can start adding pins right on top of the grid dots and they will be spaced at exactly 2.54 millimeters apart.



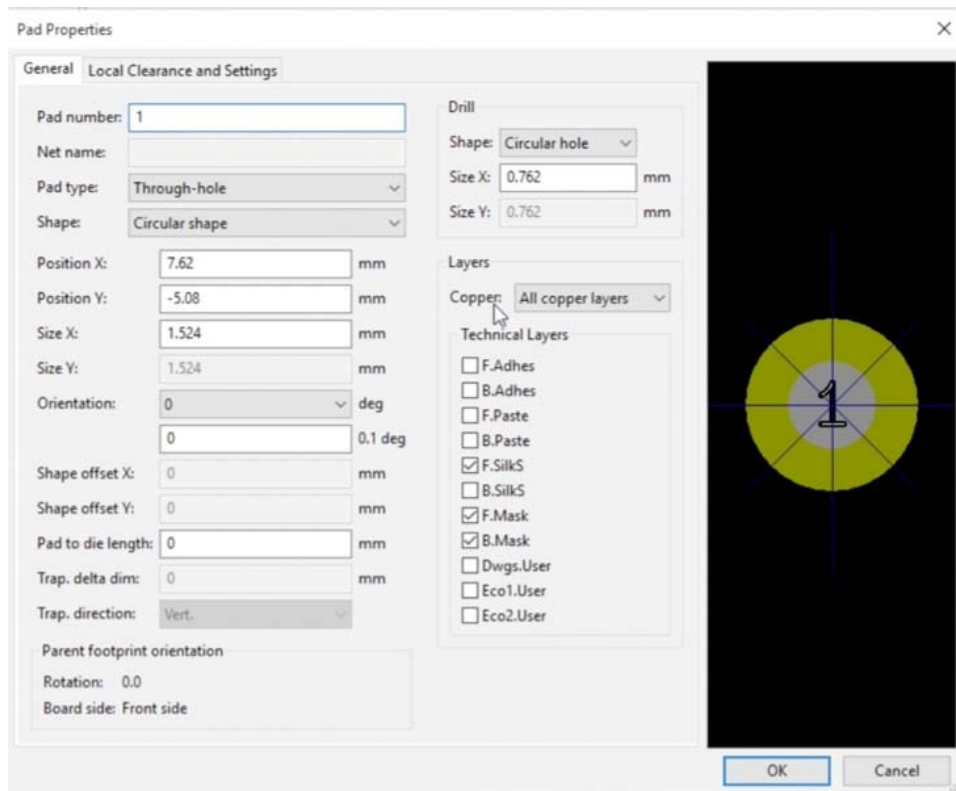
The pad button allows you to add pads on the canvas.

Click on the pad button from the top of the right vertical tool bar. This allows you to drop pads on the canvas. We need two rows of four pins each. So, select the pad tool and then click on the canvas to drop eight pads in two rows of four each. In the end of the process, you should have something like this:



Our footprint now has eight pads.

Each one of these pads has properties that you can access by putting a mouse over the pad and hitting the E key (for edit).

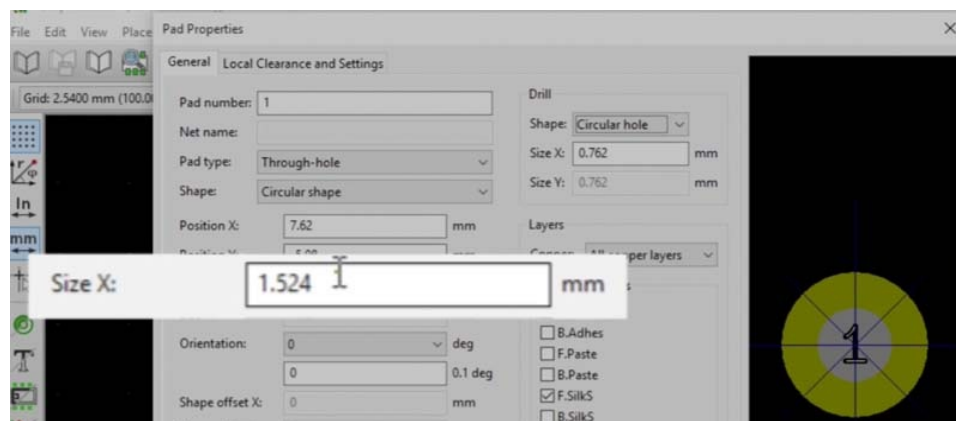


To access the properties for a pad, put the mouse cursor over it and type “E”

You can also see the pad number, marked in the center of the pad. KiCad will automatically set a unique number for each pad, as you create it. This number is important because we’ll be using it later to connect the pins on the footprint with the pins as they appear in the schematic.

In the drill group of properties in the pad properties window, you can see the shape and size, x comma y boxes, and circular shape is good for most pads, for pins at the 0.64 millimeter region. For finer pads, you can go for oval drills. The oval shape in a smaller pad allows for enough solder to make a good connection between the pad and the pin, while allowing for smaller pitch (i.e. smaller space between the pads).

In our example, a circular drill hole is fine so we’re going to set the “Size X” setting to 0.762, slightly larger then the pin’s 0.64 millimetre circumference. This ensures that the pin will fit in the hole.

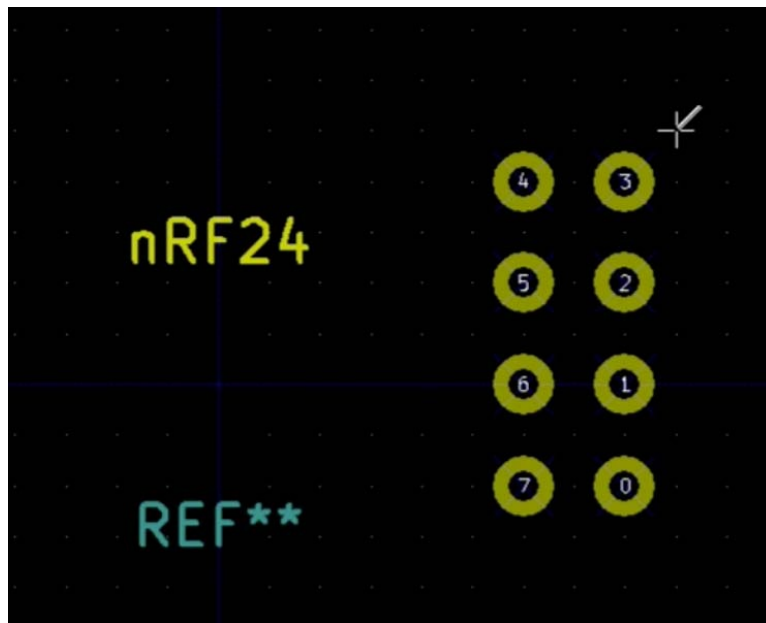


The Size X setting controls the outer size of the pad.

In the left column on the pad properties window, look at the size x textbox. This controls the total size of the pad. So for example, if you enter a value larger than the current value, you'll see that the pad size increases. If you have a pad that has larger pins like maybe a barrel power connector, then you want to have more area for the solder to attach onto, in order to support that larger part, and this is where you can make this change.

Lets close the properties window. I prefer to have consistent numbers on the pads between the KiCad footprint and my hand-drawn schematic. To do this, you can either move the pads around in order to put them in the correct order or edit the properties to update the numbers.

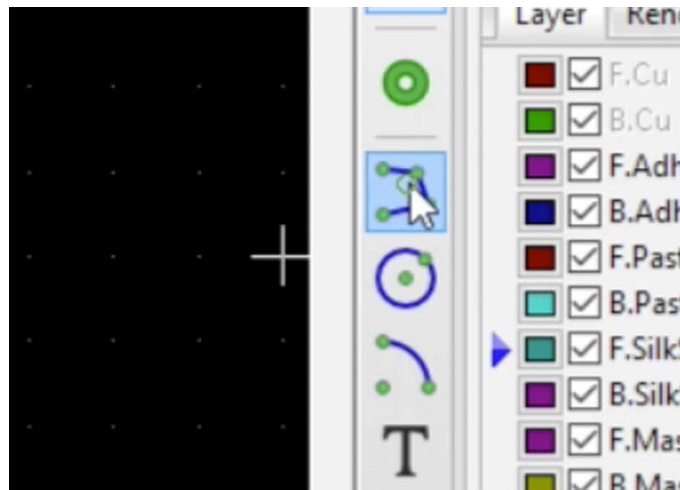
In this case, it is easier to shuffle the pins around. Use the M key to move the pads around, so that at the end of the process they are arranged like this:



The pins, rearranged.

If there are any artefacts in your canvas, like stray lines, you can refresh the page to make them go away. To refresh the page, hit the F3 key on your keyboard.

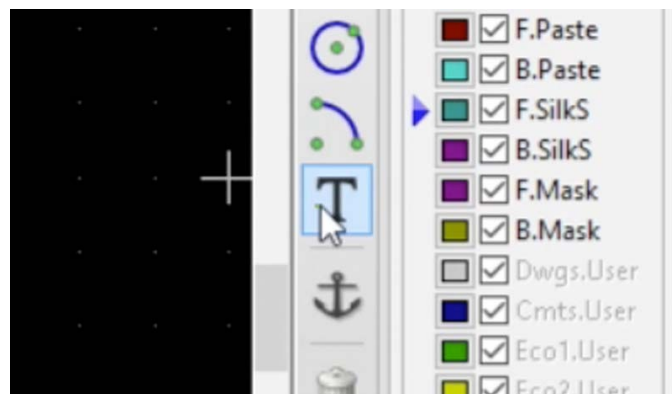
The pins are now arranged as we want them to be. The next thing to do is to draw the silkscreen border so that we have the layout for the component and the footprint as it will appear on the PCB. To do that, first reduce the grid size to 1.27 so that we have finer control about where the markings for the silkscreen will go.



Use the polygon tool to draw the silkscreen shape that will indicate the border of the custom footprint.

Next, select the polygon tool from the right vertical tool bar. With the polygon selected, draw a box around the pins and try to match the layout of the real life component. The box can be approximately the size of the real part, but there is no need to be accurate about this. Double click to close the polygon.

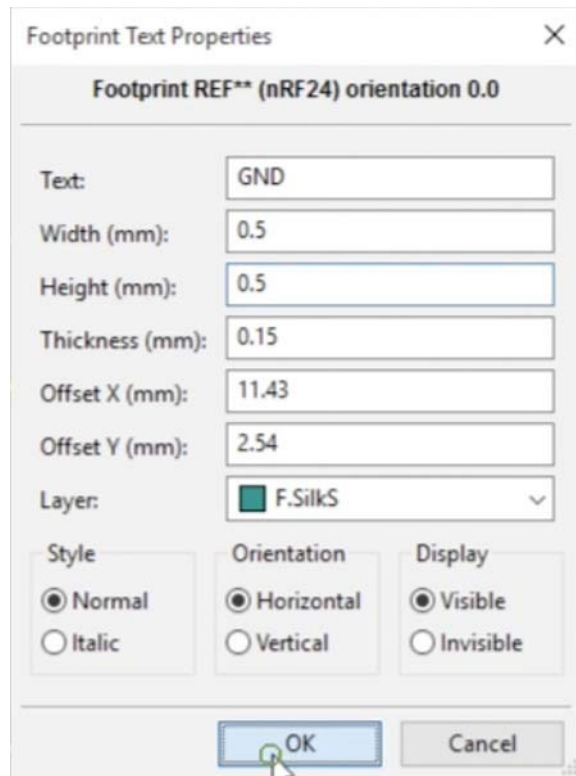
If you want to have some more interesting shapes then you can choose a circle. You can have circular footprints, will be useful for something like a button battery for example or it can have arcs like that and it can then have multiple arcs and so on. The arc, circle and polygon tools allows you to create nice, interesting shapes for your footprints.



Use the text label tool to label the pins.

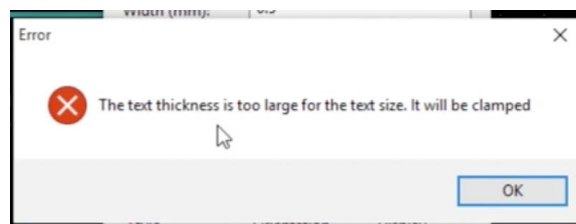
Next, we need to label the pads. To do this, we'll use the text label tool. So I'll add labels to each pad so that I can read the role just by looking at PCB. You may need to adjust the silkscreen border line if you need more space for the labels. Just delete the line than need adjusting and redraw it with the polygon tool.

With the text tool selected, click on the right side of the pin numbered "0". This will be the Ground pin. When you click, the footprint text properties window will come up.



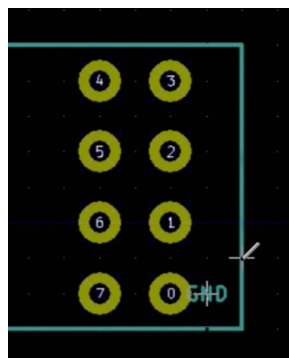
Type “GND” in the Text text field, and click “OK”.

Type “GND” in the Text text field. You can also adjust the size of the text by entering values in the Width, Height and Thickness text fields. Let’s make the text smaller than the default, so make the width a 0.5 millimeters and the height 0.5 millimeters as well. The thickness will be updated automatically by KiCad. Click OK. You should get a warning box telling you that the text thickness is too large and it will be clamped and that’s a good thing, so click “OK” again.



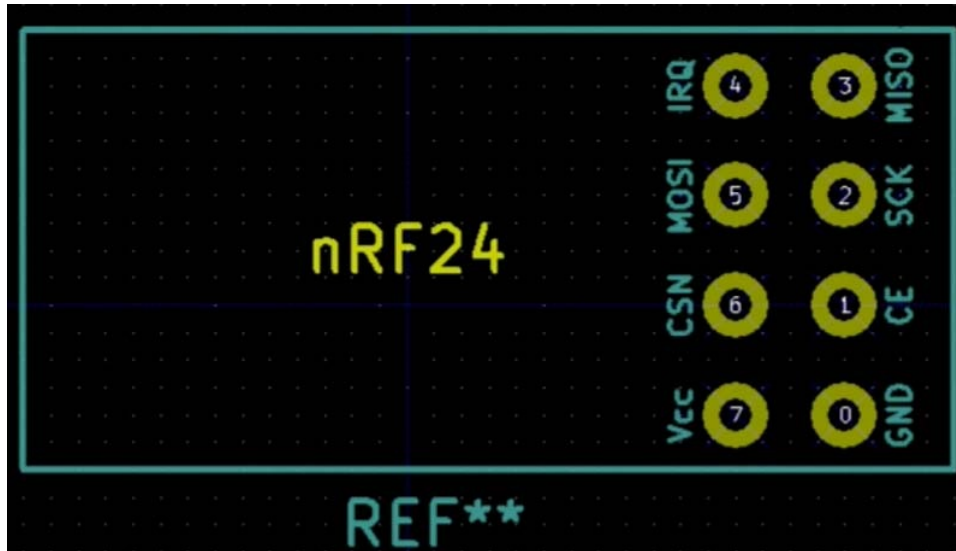
Kicad will adjust the thickness of the text based on the dimensions you chose.

This is what your footprint should look like now:



The current state of the footprint. The first text label is there.

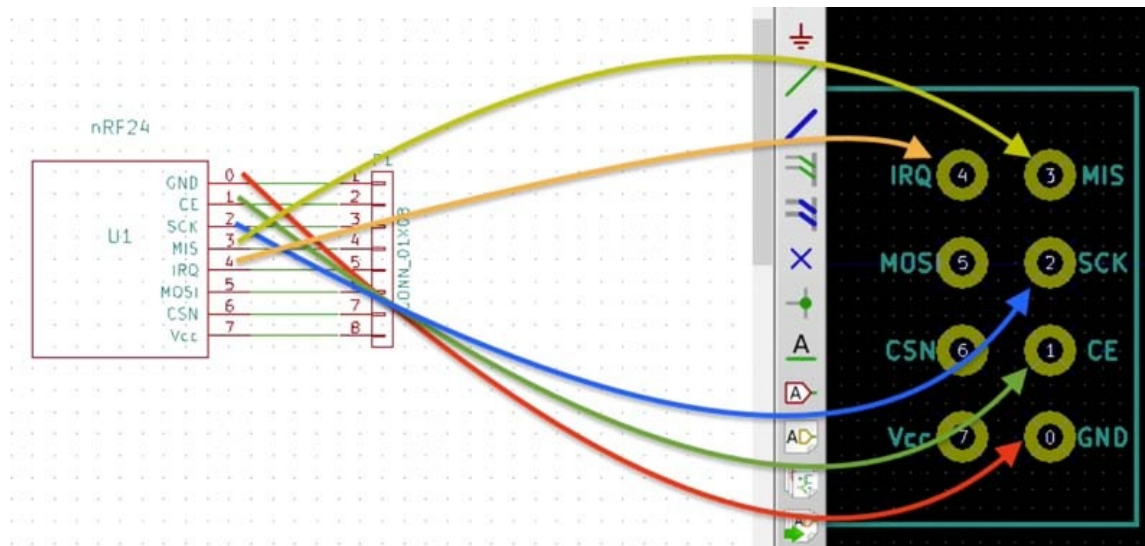
You can see the GND text label next to pin 0. You may need to move it further to the right. To do this, change the grid size to a smaller value, and use the M key to move the label. You can also rotate if you wish, by using the R key. Add the rest of the labels in the same way. A quicker way to add the rest of the labels once you have the first one configured the way you want it, is to duplicate the first one seven times. Type “D” with your mouse cursor over the GND label, and this will create a copy. Repeat the process to create as many copies as you like, with the same height, width, thickness and orientation as the original. Then, place them on the canvas and edit their name. At the end of the process, the footprint should look like this:



The current state of the custom footprint. All the labels are in place.

You'll may have noticed that there's a few more things here that you can also control. For example, you can configure a text block to be invisible if you don't think that it's necessary to show it, especially if you've got a part with a lot of markings and you don't want to have all the clutter showing. We'll use its feature later again. You can also choose between different styles, like normal and italic, and choose the orientation.

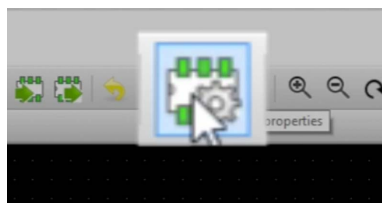
We are finished with designing this custom footprint. KiCad will use the pin numbers in the schematic and footprint to figure out how the two are supposed to be connected. To make sure that the pins match, you should compare the pin numbers across the two versions of the same thing, so the schematic version and the footprint version against the real part.



This image show how the pin numbers in the schematic match the pin numbers in the footprint.

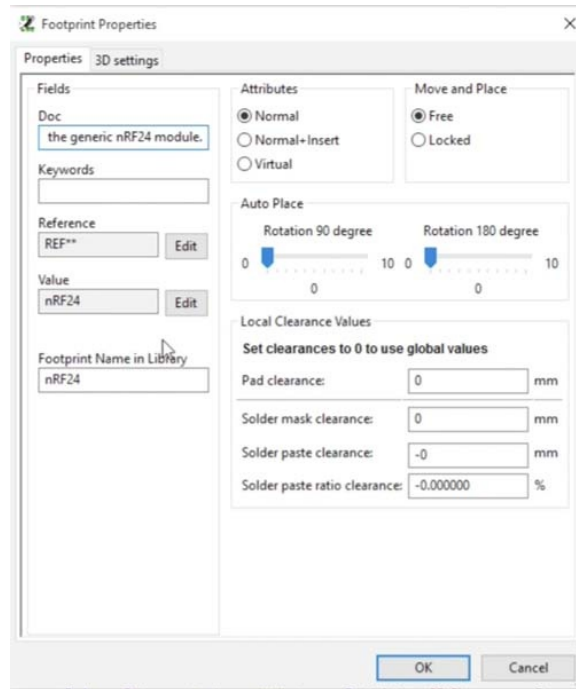
In this image, I use an arrow to show how pin 0, labeled GND in the schematic, corresponds to pin 0, labeled also GND in the footprint. Similarly, you should double check that all pairs have the same numbers and labels. I didn't use arrows for all the pairs in order not to clutter the image here.

You should also triple check that the markings on the footprint match exactly with the pin out on the real part. So, having a double and triple check of all of these details, I think that the part that we've designed here is correct.



Click to reveal the footprint properties.

Another thing to do before we save the part and use it is to update the footprint properties. So, to do that, we'll click on the footprint properties button, and this is going to give us a way to add some properties for the footprint.



The footprint properties window.

We should fill in the Doc field with something like “A footprint for the generic NRF24 module”. This text will appear in the library as you are searching around for parts.

Leave everything else with their default values, and click OK. We are not quite finished yet. We still need to save the new footprint, also referred to as a “module”, in a new library. We will do this in the next chapter.

Chapter 20: *Saving the new footprint*

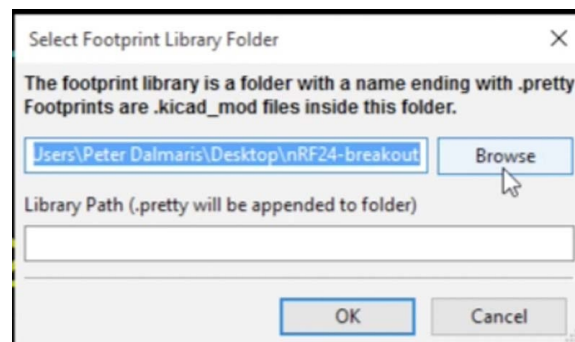
In the previous chapter, we created our first custom footprint. We haven't saved it yet, which is what we will do now.

You could go ahead and save the new part in an existing library but since this is our first custom part, it is better if we create a new library for it, and then save the footprint and save it in it. In general, this is a good practice. When you create a new part, it's usually tied to particular project. So you want to have a library for footprints specific to that project.



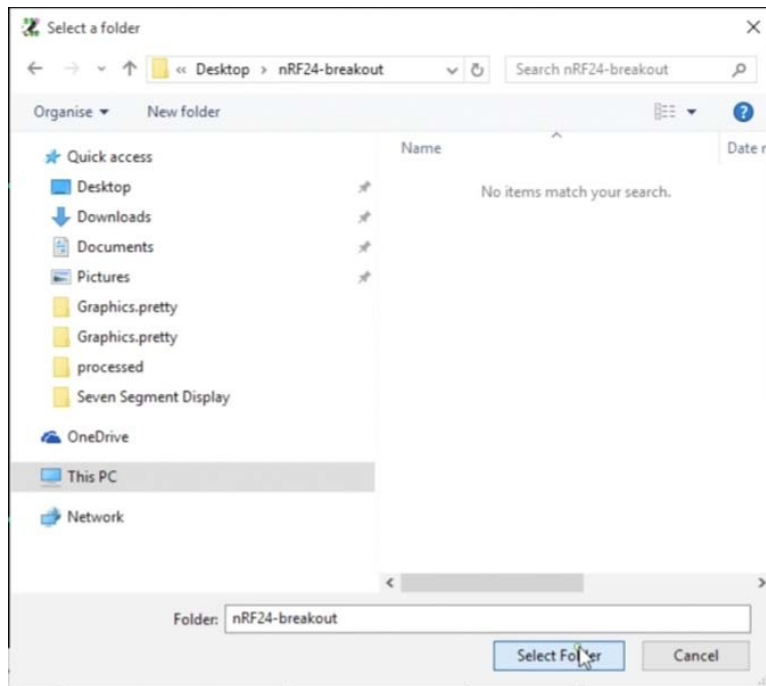
The New Library And Save Current Footprint button.

To create a new library, click on the New Library And Save Current Footprint button.



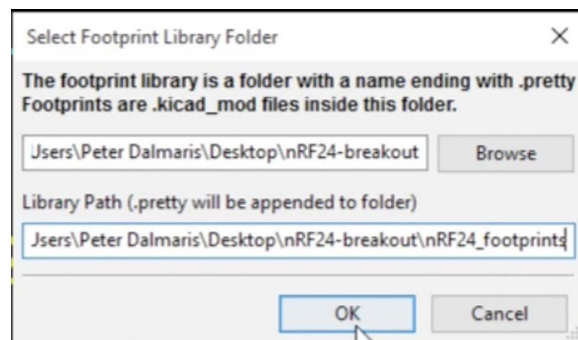
The Select Footprint Library Folder window. Browse to your project folder and save the new library there.

Click on the Browse button to browse to the location of our project.



Give the new library a suitable name.

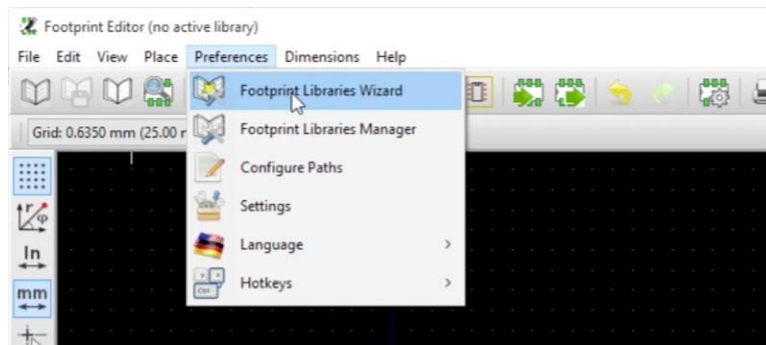
Give this library a name, something like “NRF24 footprints”.



The Footprints Library Folder is updated with the library path and name.

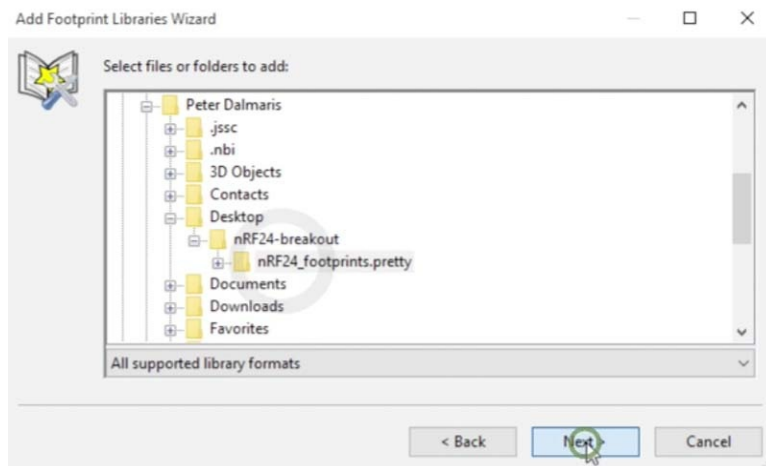
Click OK. Kicad will save this new footprints library. Go to your project folder to verify. In the project folder, notice a new folder named “NRF24_footprints.pretty”. Go inside this directory, and notice a file with the Kicad_mod extension to it. This file contains the new footprint.

Now we’ve created a new library, but to save the footprint in it, we must make it active. Even though we just created it, Kicad doesn’t do this automatically so we’ve got to do it manually. Here is how to do this:



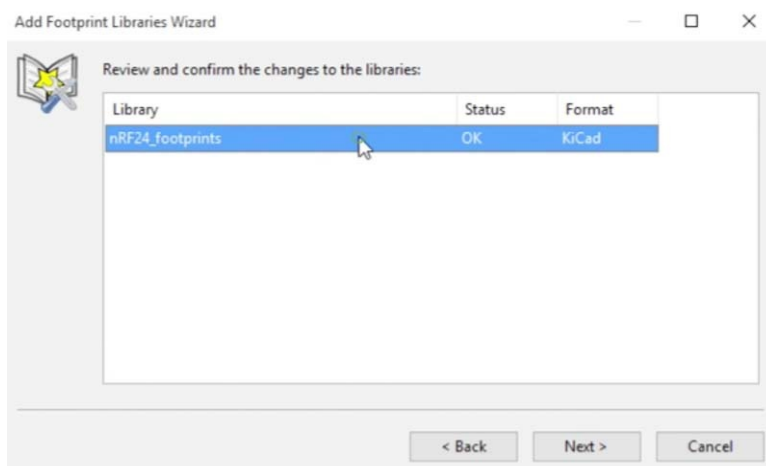
To make a library active, start by starting the Footprint Libraries Wizard.

We start by importing the library to Kicad. In the preferences menu, start the Footprint Library Wizard.

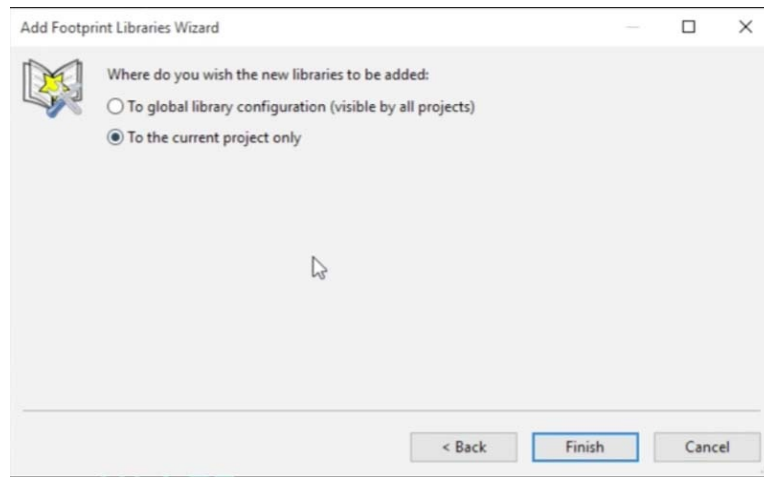


Browse to the location of the new library, in the project folder.

Look for the new library inside the project folder.



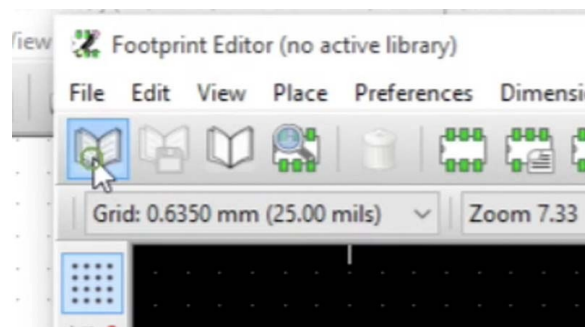
The wizard confirms that what we selected is a valid footprint library.



You can choose to make this library global or active for the current project only.

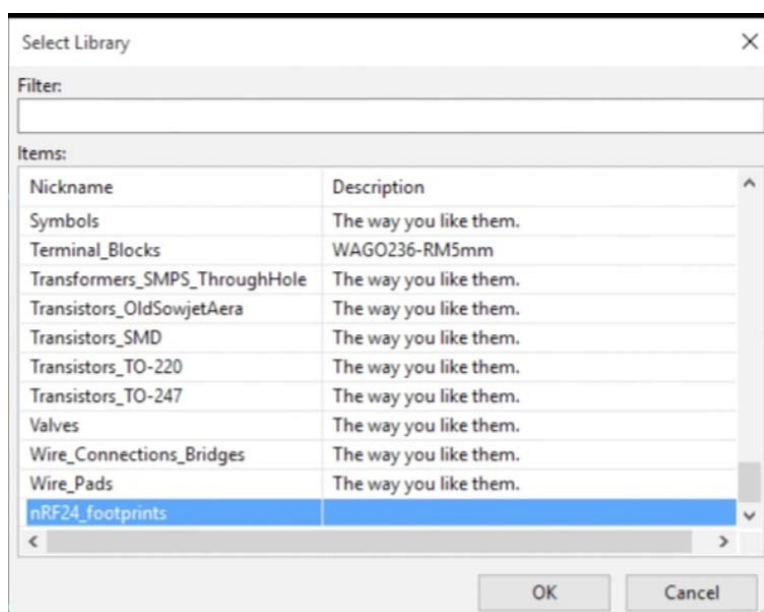
We will only use this library in the current project only so select the appropriate option in the last step of the wizard, and click on Finish.

The last step before we can finally save the footprint is to make the library active. I know this doesn't make much sense. Why do you need to make the library active, and then save the new footprint in it? I don't know but that's how it is with Kicad. Unfortunately, some things don't make intuitive sense.



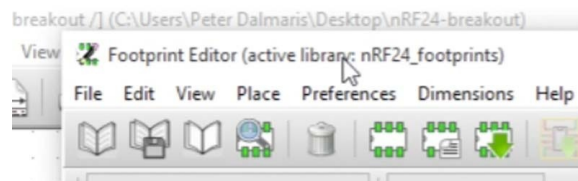
The Select Active Library button

To make the library active, click on the select active library button.



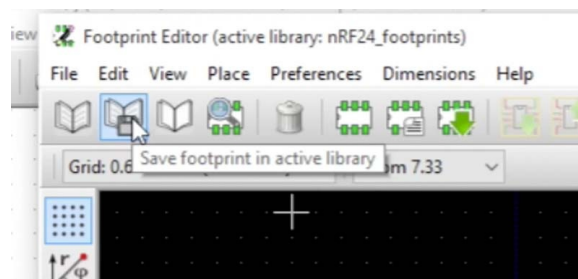
Scroll down the list of libraries that Kicad knows about, until you find our new library. Select it and click OK.

Scroll down to the bottom to find the one we added. The name is NRF24_footprints. Click OK. This is now the active library.



The name of the active library is shown at the top of the Footprint Editor window.

You can see that the name of the active library appears at the top of the Footprint Editor window.



The Save Footprint button

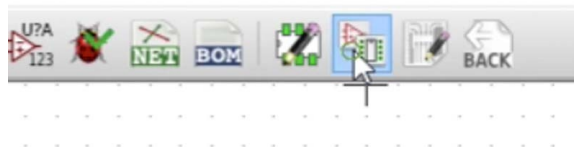
Finally now, click on the Save Footprint In Active Library button, and give a name for the footprint, or just accept the default. Click OK, and finally, we now have a new footprint stored safely inside a new library that is accessible only by this project.

In the next chapter, we will associate the new footprint with its schematic component.

Chapter 21: Associate the new footprint and component

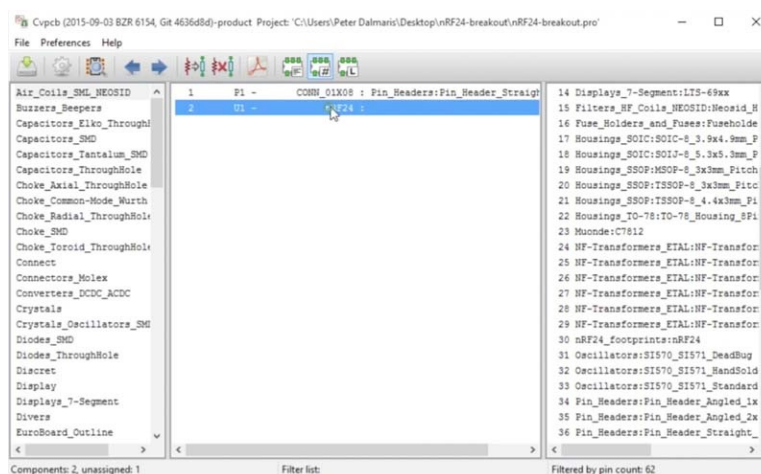
In this chapter we will associate the new footprint with its schematic component.

Go back to Eeschema and start Cypcb.



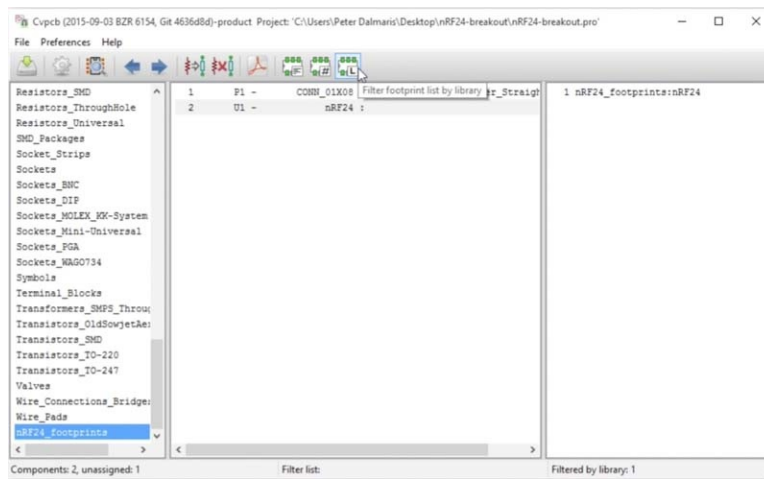
From the Eeschema application, start Cypcb.

From the Cypbc window, you can see that the straight connector already has a footprint associated with it. The nRF24 component doesn't have on.



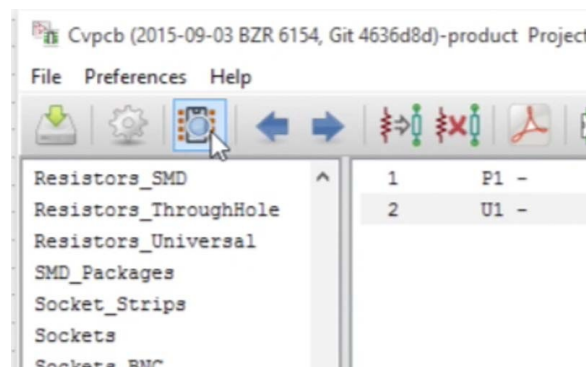
The straight connector component has an association, but the nRF24 does not.

To make an association for the nRF24 component, scroll down the list in the left pane. At the bottom of the list, you will see the library we created in the previous chapter.



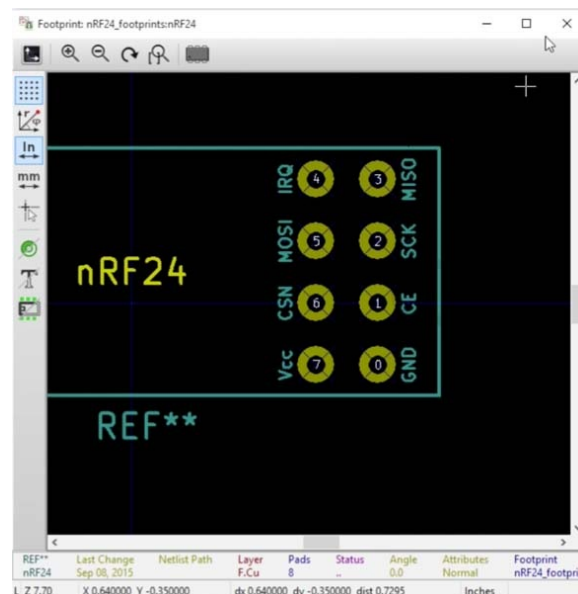
The new library is at the bottom of the list in the left pane.

Click on it to select it. With the custom component selected in the middle pane, click on the “#” filter. Notice that in the right pane, the custom footprint for the nRF24 component appears. Double click on the footprint to select it.



The preview button allows you to get a preview of a selected footprint

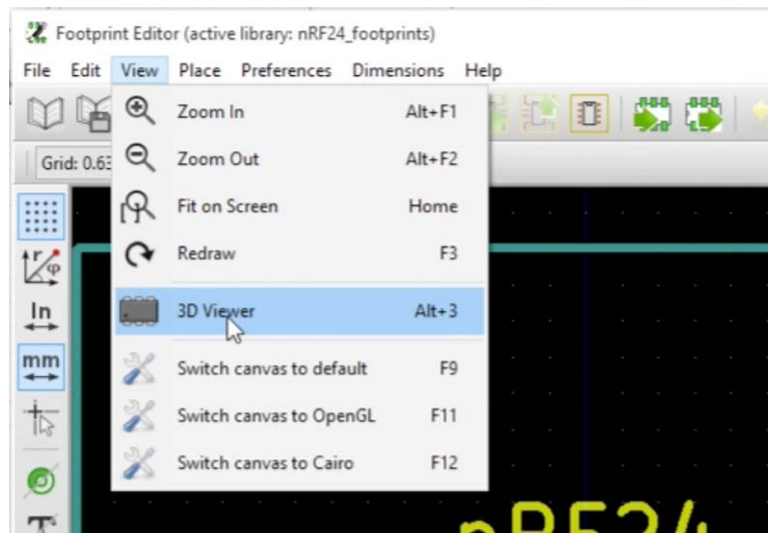
We know of course what the custom nRF24 footprint looks like, but let’s have another look to confirm. Click on the preview button from the top bar.



The preview of the nRF24 footprint.

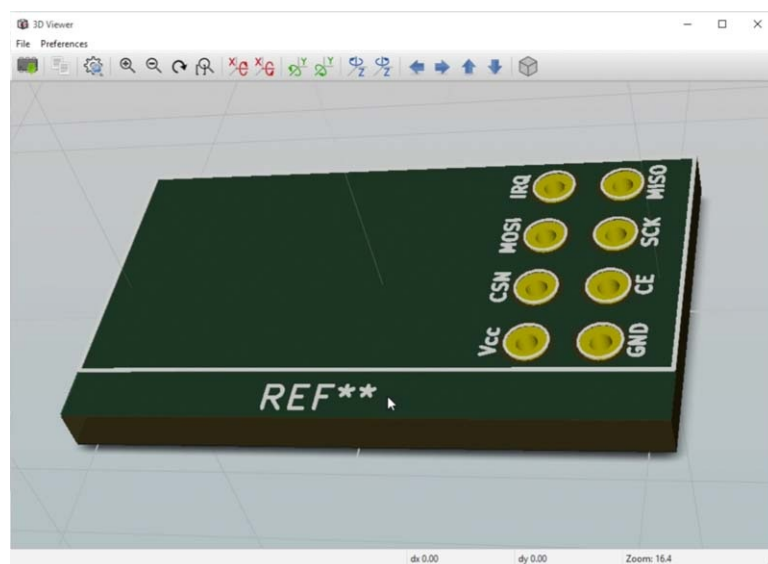
Let’s save the associations and close Cypcb and again save the schematics file.

Before we move on, I would like to show you one more useful feature. Start the footprint editor, then from the View menu item select the 3D Viewer.



Launch the 3D viewer from the View menu.

The 3D Viewer will give us a 3D representation of the footprint. We can see how the component will look like once it goes onto a PCB.



The 3D viewer give a “real life” representation of the footprint.

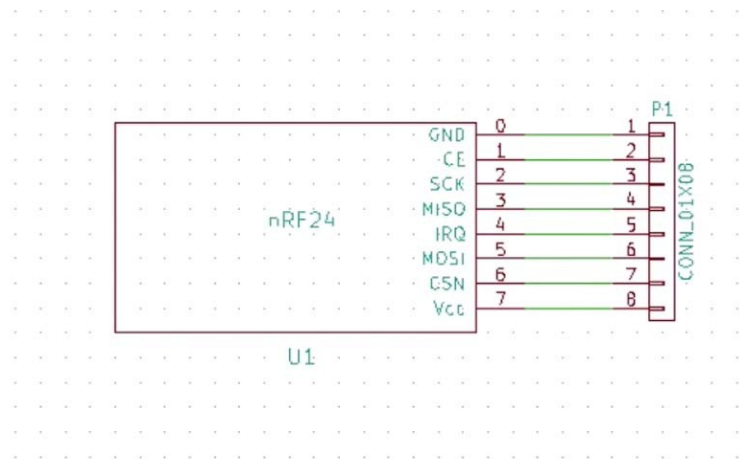
You can choose if you’d like to have this identifier inside the white silk screen books or not, it doesn’t really make any difference and you can have a look on the other side, the bottom side as well. Use your mouse buttons and scroll wheel to rotate, zoom and pan the viewer.

We are almost finished with Eeschema. In the next chapter, we will create the Netlist file. The Netlist file is a file that contains information about the circuit, it’s components, associated footprints, labels and pin numbers and many other things. Our PCBnew, which is the PCB editor, would read this file and load the appropriate footprints from the library and that will do the layout and wiring.

Chapter 22: *Create a netlist*

We have progressed with our project to the point that the bulk of the work in Eeschema is complete, and the only thing left to do is to export the netlist file. In this chapter I will show you how to export the netlist file from Eeschema, and then import it into Pcbnew.

Start Kicad and then launch Eeschema if it's not already started. This is the current version of the project.



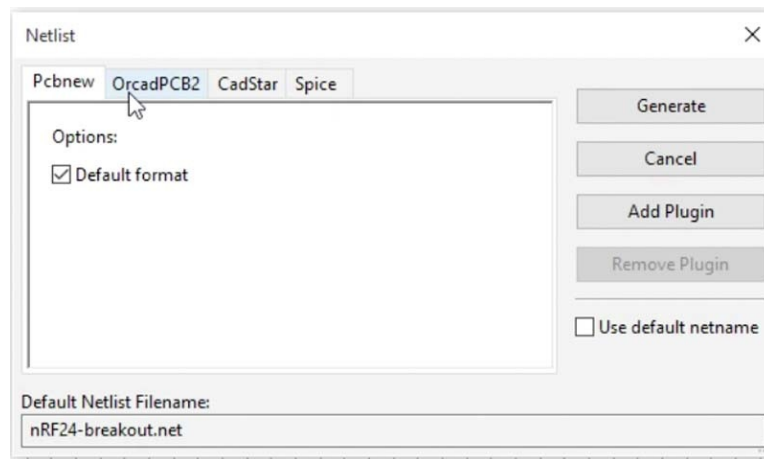
The current state of the project. We will export the netlist for this schematic.

To export the Netlist file we will click on the Generate Netlist button.



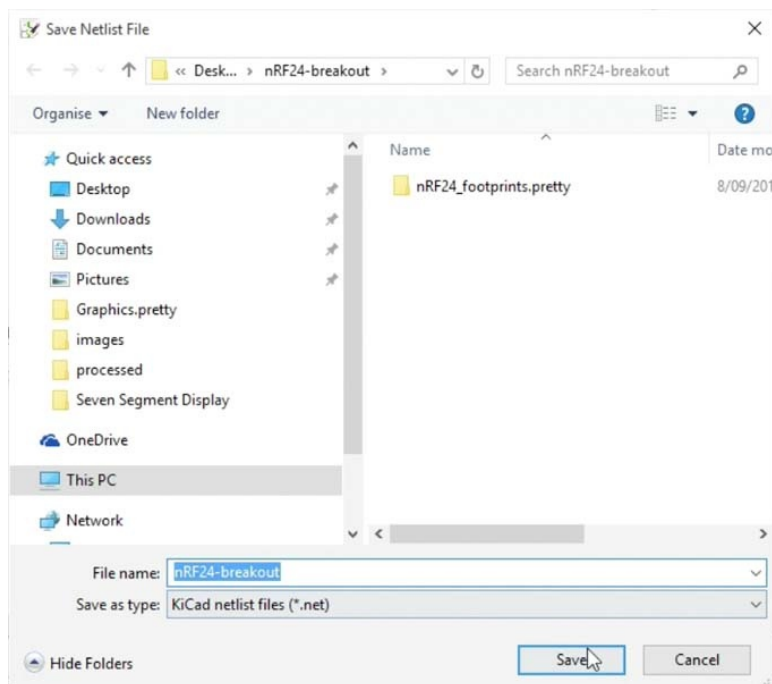
This is the Generate Netlist button.

The options in the export window are straight forward.



The netlist export options.

We don't have to worry about OrcadPCB, CadStar or Spice. We are not working with these applications. We are staying completely within KiCad. Simply select the Default format option in the Pcbnew tab, and click on Generate to generate the Netlist file. Next, choose the location of the exported file to be somewhere inside the project directory.

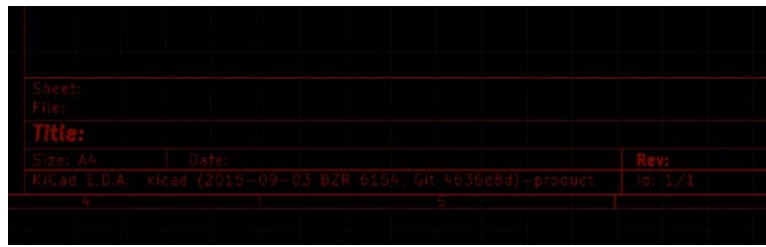


Save the netlist file in the project directory.

If you go into your project directory, you will see the new netlist file. It is a simple text file and you can inspect it with a text editor. Open it with Notepad++ (or any text editor), and notice that this file contains all the information that Pcbnew, the PCB layout program, will need in order to know which components are supposed to be included and what are the connections between those components, information about nets and so on, is all in here.

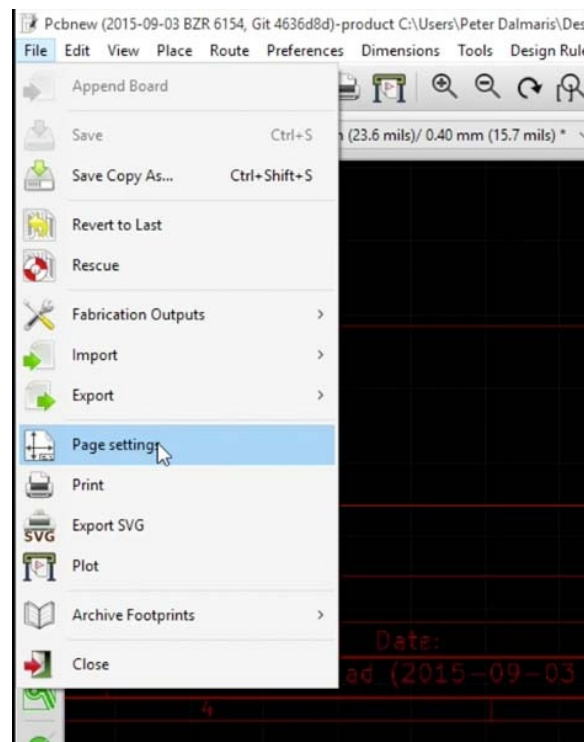
The blank Pcbnew canvas.

At the bottom right corner of the canvas you can see the the information for the project. Let's setup the contents of the label in the bottom right corner of the canvas.



The project information is in the bottom right corner of the canvas.

To set up the contents of the project information box, click on File and then we go to page settings.



Edit the contents of the information box by selecting “Page settings” in the File menu.

Fill the fields with the information you would like to show in the information box. as many or as little as you like.

Page Settings

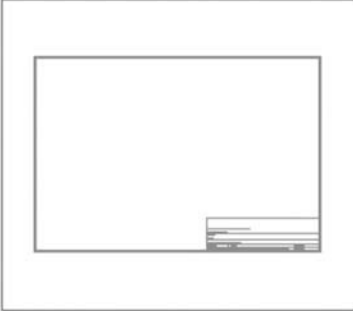
Paper

Size:
A4 210x297mm

Orientation:
Landscape

Custom Size:
Height: 279.40 Width: 431.80

Layout Preview



Title Block Parameters

Issue Date
2015-09-22 <<< 22/09/2015

Revision
1.0

Title
nRF24 breakout

Company
Tech Explorations

Comment1
Part of Kicad make PCBs like a Pro

Comment2

Comment3

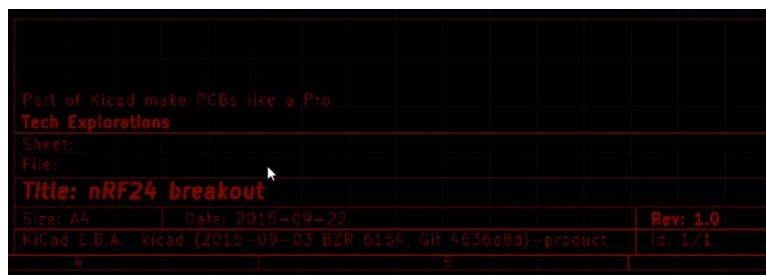
Comment4

Page layout description file
Browse

OK Cancel

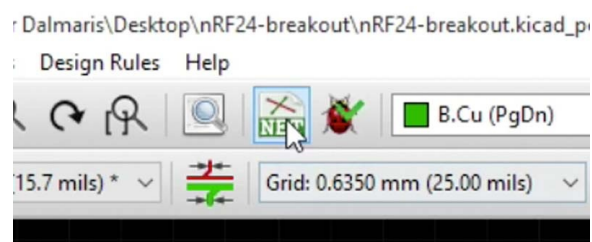
Enter the information you would like to include in the information box.

Click OK when you are finished typing in the information. Notice that the information now appears in the legend of the canvas.



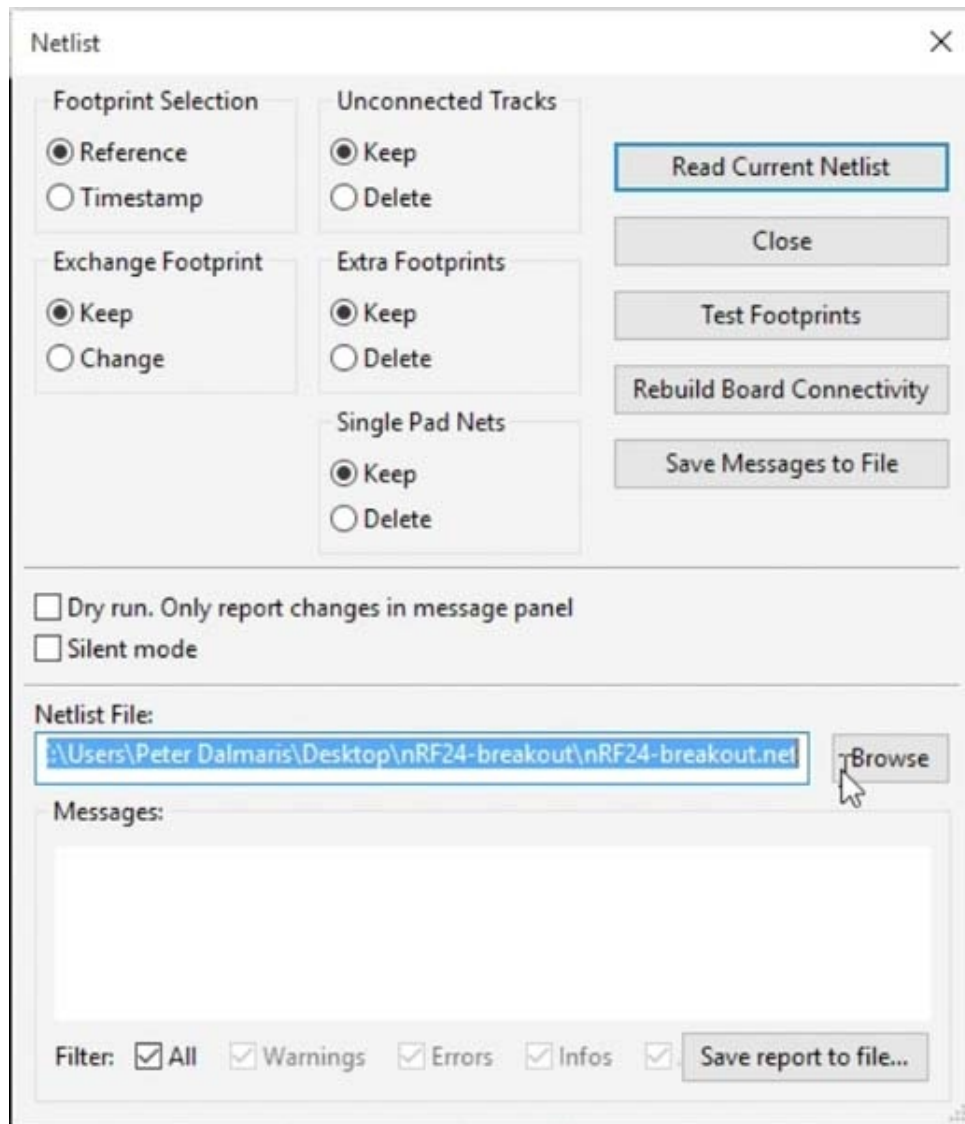
The updated information in the Pcbnew legend.

Next, let's read the Netlist file and import all the footprints that make up the project.



The netlist button

To read the Netlist file you click on the button marked "NET".



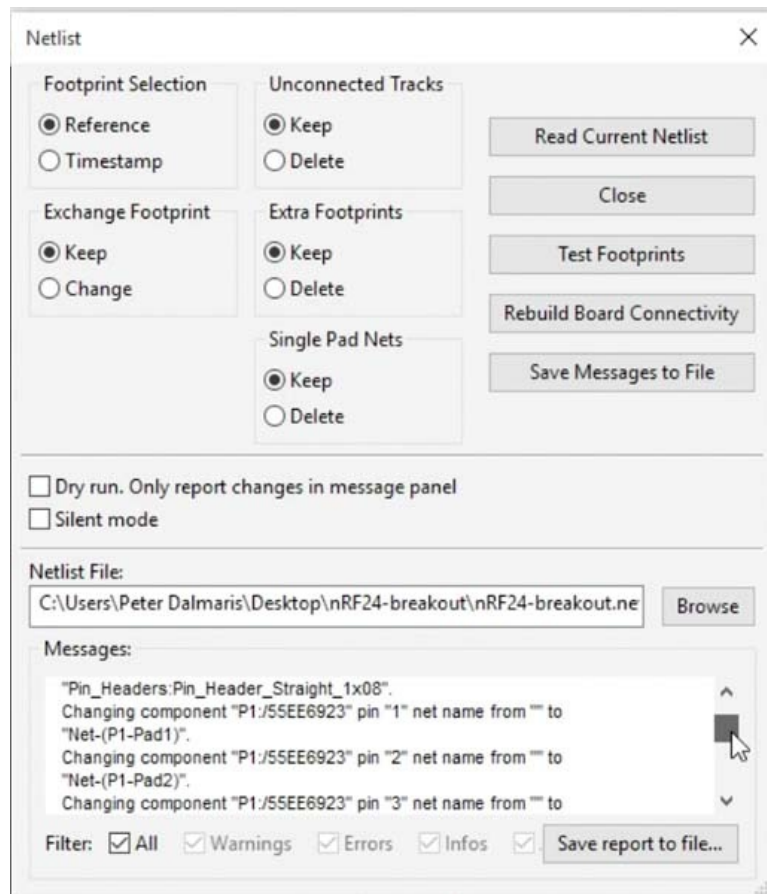
The netlist import dialog.

The default settings in the netlist import dialog are good as they are. Browse for the location of the netlist file (although most likely the path to the file will already be in file field).

Kicad, through the Netlist, file can track new components and old components. We will see later, for example, that a change that I will introduce to my original schematic will be reflected in Pcbnew via updating the netlist file.

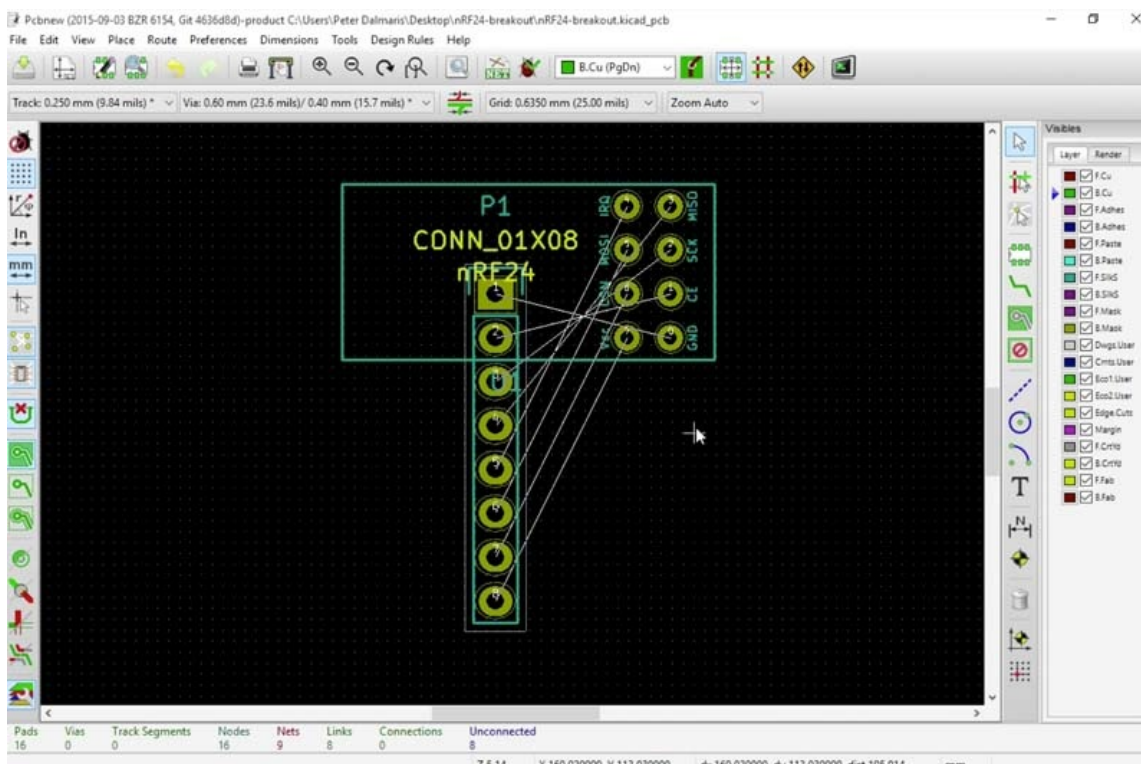
Kicad is able to detect new components in a netlist file and only import the new components from Eeschema into Pcbnew. This allows you to make changes to your schematic and then import those changes into your PCB layout without losing any work. I'm going to demonstrate this in practice in one of the upcoming chapters.

We not going to change any of the parameters for the import here, so go ahead and click on the Read Current Netlist button.



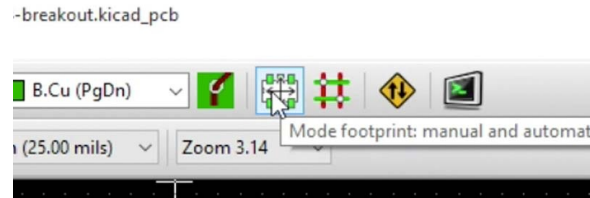
A report of the netlist import is provided in the Messages text box, once the import is complete.

At the bottom part of the Netlist window you can see the import report. There are no error messages. Close the window. As the Netlist file was read by the application, it placed all of the components right in the middle of my canvas.



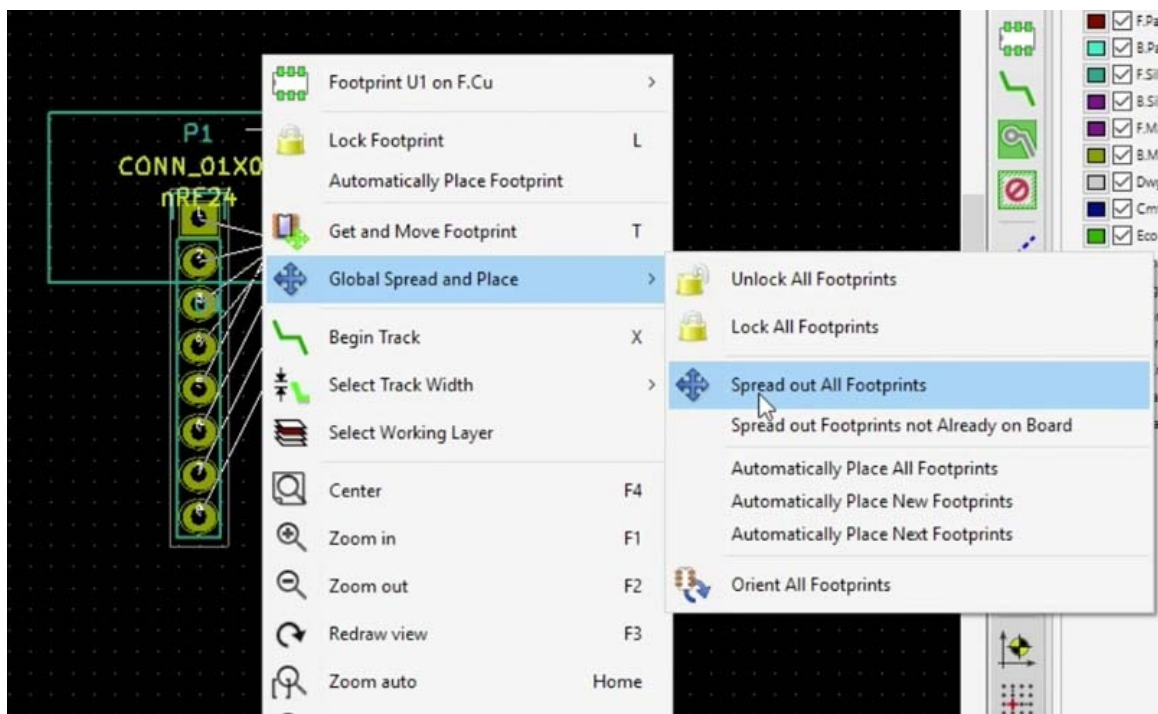
The footprints, after the import from the netlist file.

Imagine the situation where you had 10 or 20 or 100 components, all of those would appear right one on top of the other. It would be a big hassle to manually move them out. Kicad provides a nice feature to allow you to automatically separate those components before you manually set them and place them where you like them on the PCB.



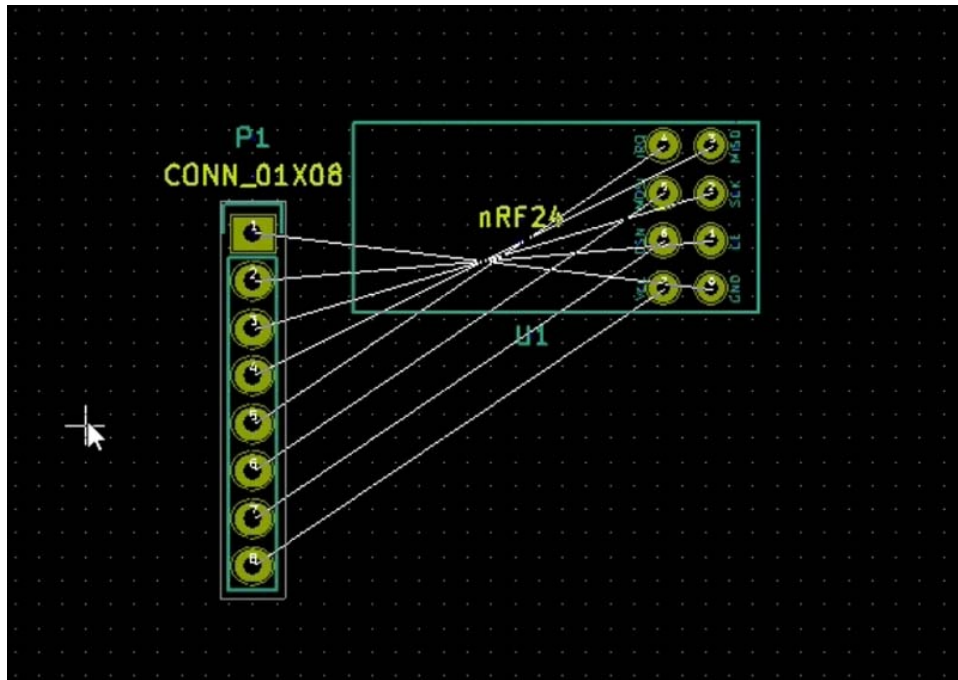
The Mode footprint button.

To enable that feature you click on Mode Footprint button. Once you enable Mode Footprint, you can right click anywhere on the bundle and you can see that there is an option called “global spread and place”. Then, select “spread out all footprints”.



Through the right click menu, you can access the “Spread out all footprints” function.

You will get a warning that locked footprints will not be moved. This is ok, as we don’t have any such footprints. Click “Yes,” .



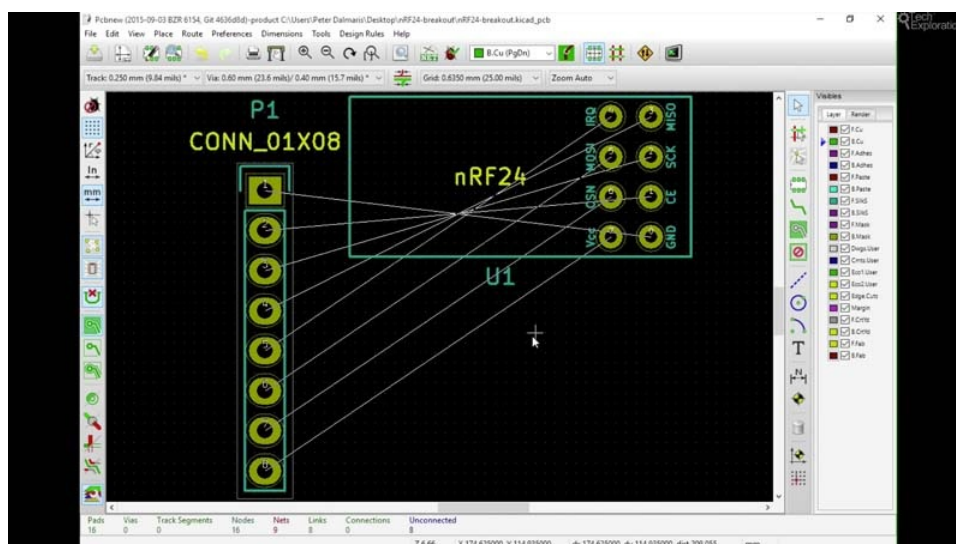
The footprints are now spread on the canvas.

Now it's cleaner. If you had 10 or 20 or a lot more components then those components would be nicely spread out the canvas, so you can then start doing your wiring or placement.

In the next chapter we will work on the footprint placement.

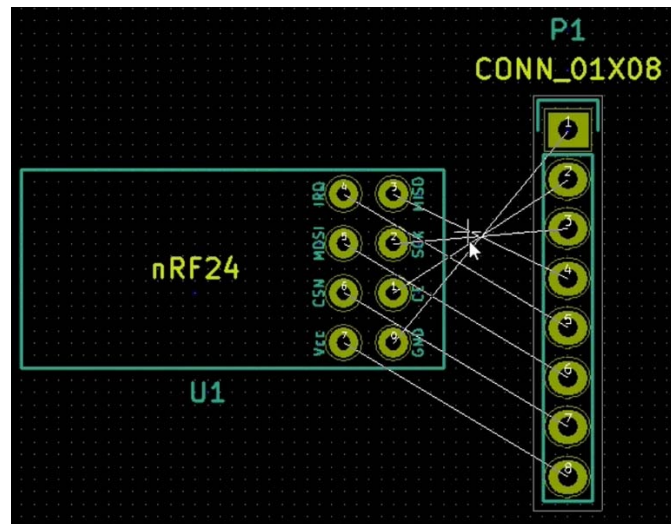
Chapter 23: *Footprints placement*

Our project is now at the stage where the two footprints that compose our PCB are spread out in the Pcbnew canvas. In this chapter we will do the footprint placement so that we can start giving shape to the final PCB.



The current state of the PCB design.

I would like to place the connector on the right side of the breakout and the nRF24 component on the left side. To do this, position the cursor over the nRF24 footprint and hit the 'M' key. This will allow you to move this footprint. Move it so that it is on the left side of the straight connector.



The nRF24 footprint is on the left side of the connector. Notice the rattiest lines. They indicate the pads that should be wired together.

The thin white lines that connect the pads together are called “ratsnests” . They are routing guides. As we wire each pair of pads together, the corresponding ratsnest will disappear.

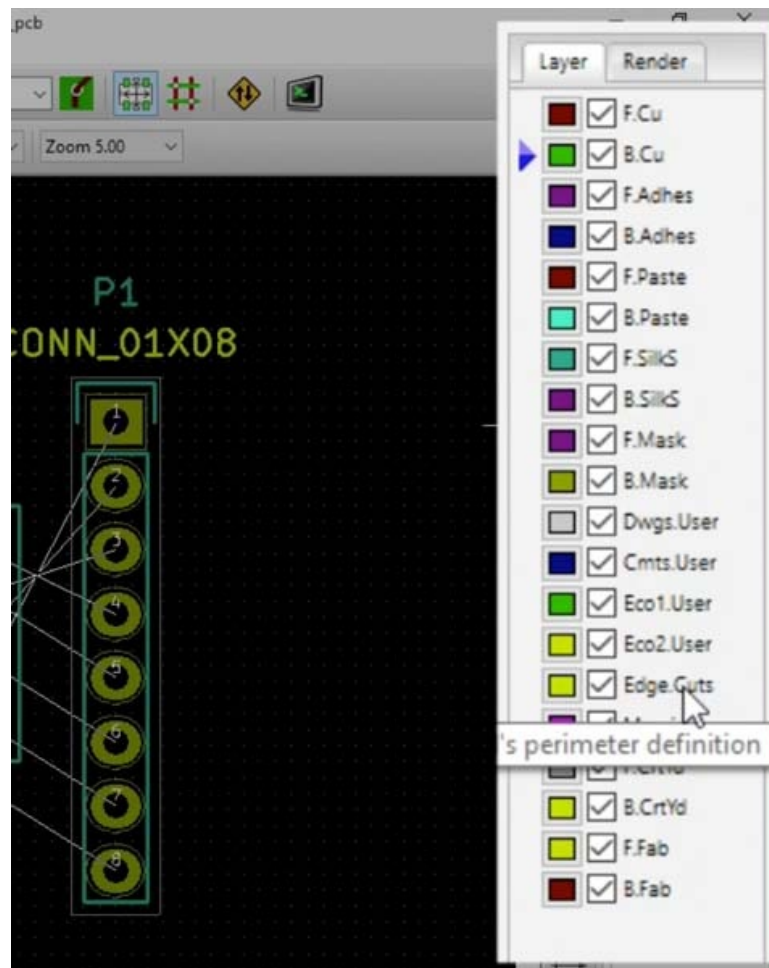
One thing to consider when you are placing your components onto your PCB is space. How much space is your final PCB going to take up? Remember that PCB manufacturers, will charge you not based on have many holes and tracks your PCB has but based on its dimensions. Therefore, the smaller your PCB is, the cheaper it will be to make.

However, the smaller the PCB is, the harder can be to route it. With less space in between footprints, the routing of tracks will be more difficult. This is not a problem for the simple PCB of this example, however if you had more footprints, then placing them too close to each other would make routing and then soldering harder. You must think about this and find a dimension that works both from a cost point of view and from these other technical considerations like the soldering and the routing point of view.

In the next chapter we will look into the PCB edges, known as edge cuts, that define the border of the PCB. After that, we will be able to go ahead to complete the wiring.

Chapter 24: *Edge cuts*

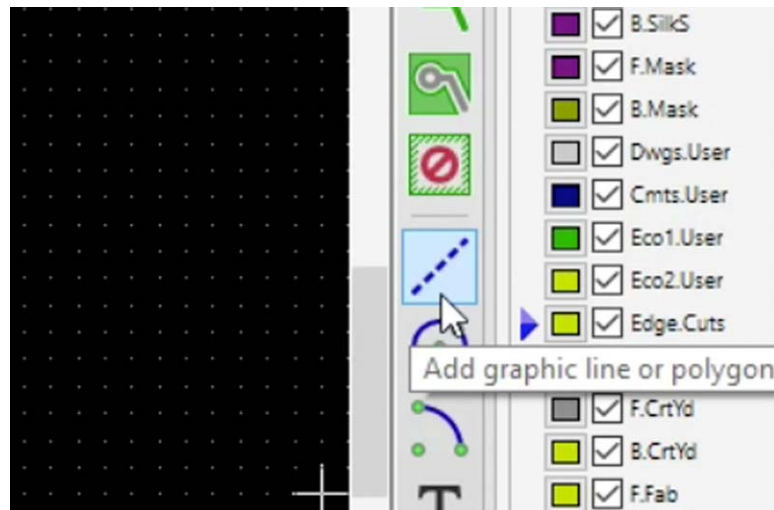
Before we start working on the wiring, we must define the boundary of the PCB. This boundary is created by drawing a box in a special layer of the PCB, the edge cuts layer.



The Edge Cuts layer is one of several PCB layers available in Kicad. It is the one where the boundary of the PCB is defined.

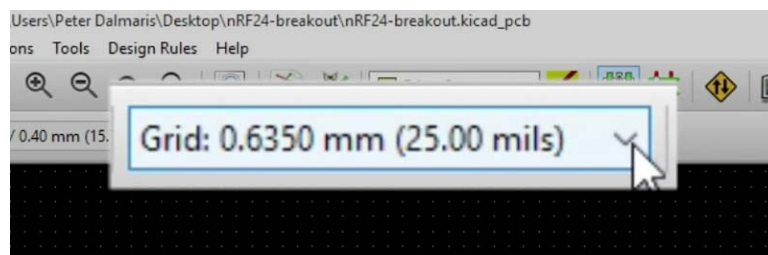
To do that, start by selecting the Edge.Cuts layer from the Layer chooser on the right side of the Pcbnew window. There are many other layers available, some of which we will

use later.



We will use the Polygon tool to draw the boundary of the PCB.

Once you have selected the Edge.Cuts layer, click on the add graphic line or polygon button. You can choose other types of graphics or graphic elements but for the task at hand we will use the polygon to create the boundary box around our PCB. To make the drawing easier and more precise, just like in EESchema, you can choose to change the size of the grid.

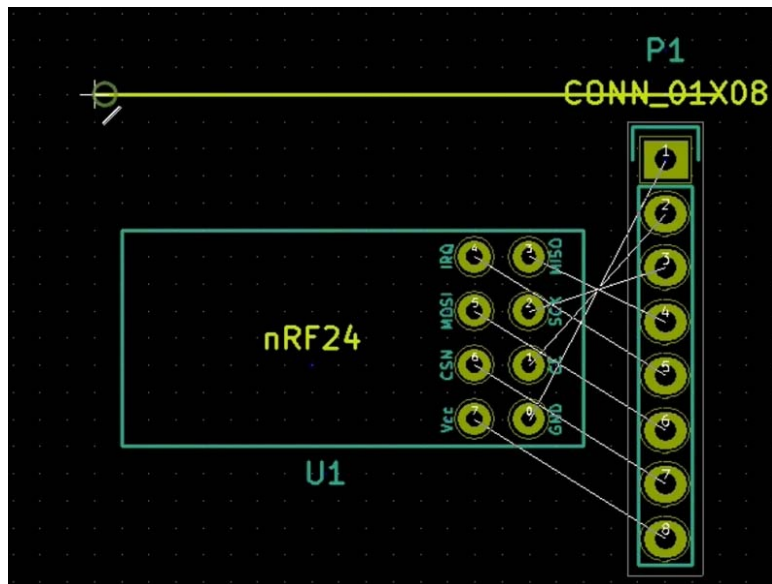


The grid setting in Pcbnew. Decrease it to make drawing more accurate.

At the moment, the grid is set to 0.635 millimeters. For this drawing, I suggest a setting of 0.127.

To start drawing, with the polygon tool selected, click just outside at the top right corner of the straight connector. This will start drawing a line. Move the cursor horizontally towards the left, until it is over the top left corner of the nRF24 footprint, and click again to mark the left top edge of the box.

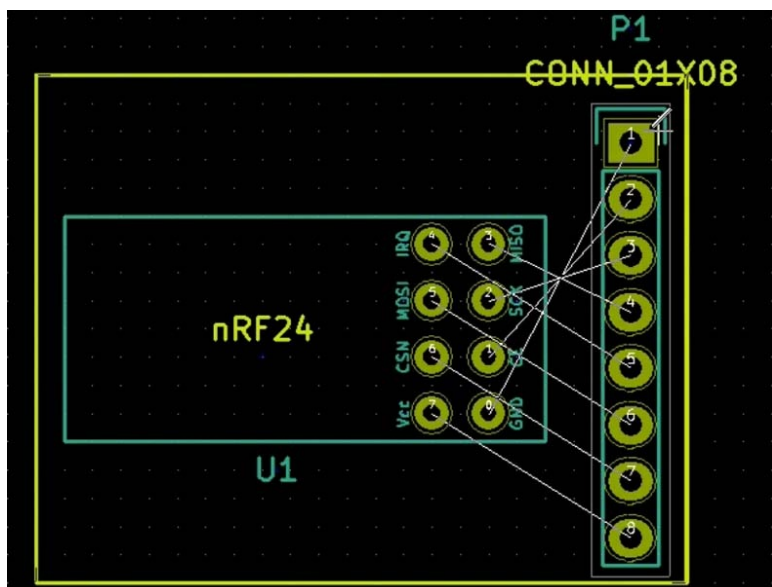
The line should look like this now:



The first line of the boundary box.

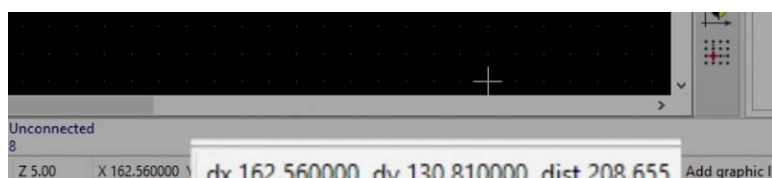
Continue in a similar way to close the box. Move the mouse down towards the bottom left corner of the nRF24 footprint, and click to define the third edge of the box. Then move towards the left, horizontally, until you reach just outside the right bottom corner of the straight connector, and click to define the third edge. Finally, move the mouse cursor over the first edge and double-click to close the polygon.

The boundary should look like this now:



The completed boundary box.

This box defines how big your PCB is going to be. One thing to notice is that you can measure lengths in KiCad very easily.

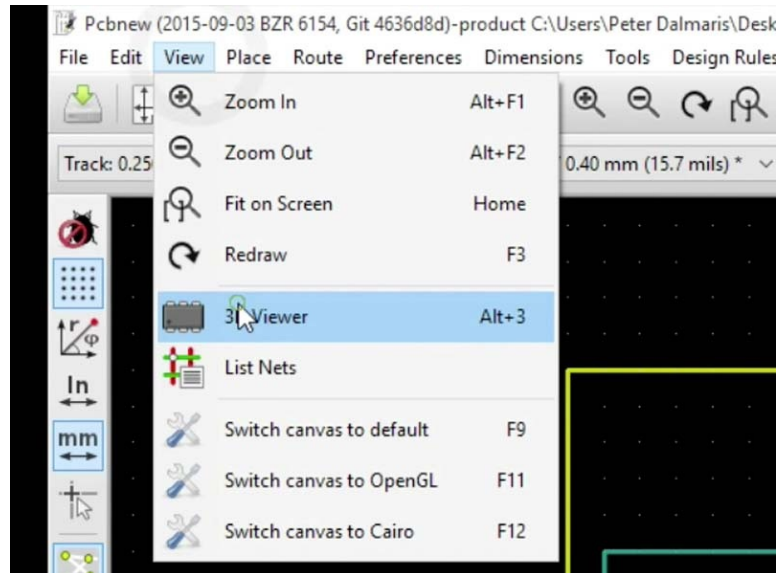


Distances between points on the PCB or the top-right corner of the canvas are

shown in the status bar.

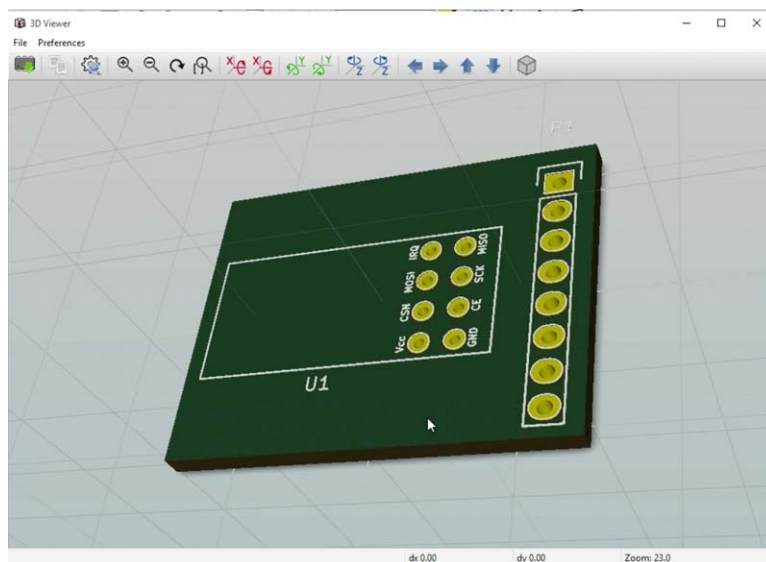
You can see, in the status bar of the Pcbnew window, the distance measurements. These are dimensions for X and Y axis. If you want to know for example, how long this side of my PCB is, move the cursor to one end, press the space bar to zero the measurement and then move across to the other side. The distance will be shown in the status bar.

How does our PCB look like now? Let's have a look. We will use the 3D Viewer for this.



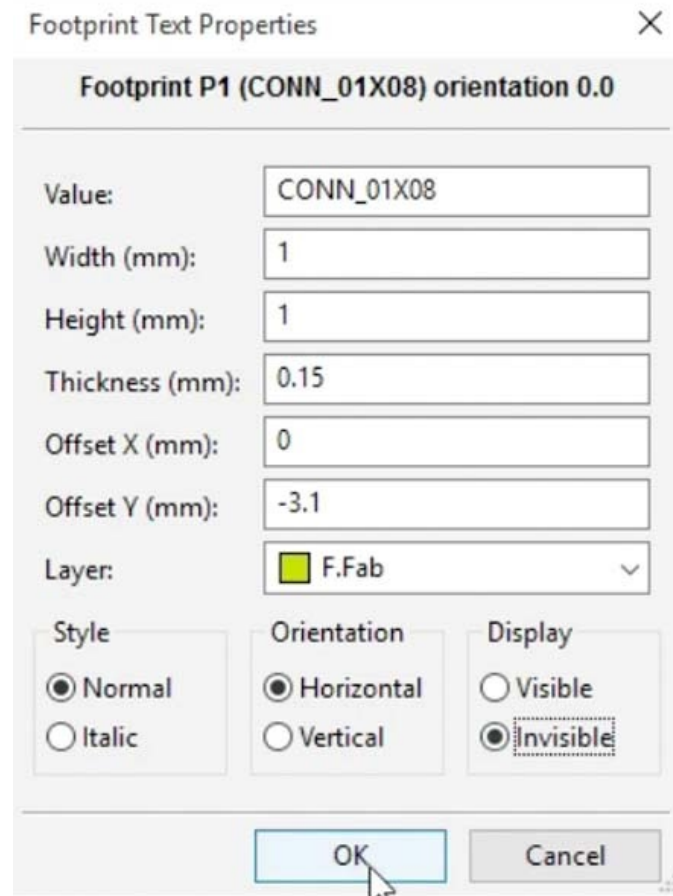
To access the 3D Viewer, go to the View menu and click on 3D Viewer.

This is what the PCB is going to look like this, without the routing done yet:



A 3D view of the PCB. The wiring is not done at the moment.

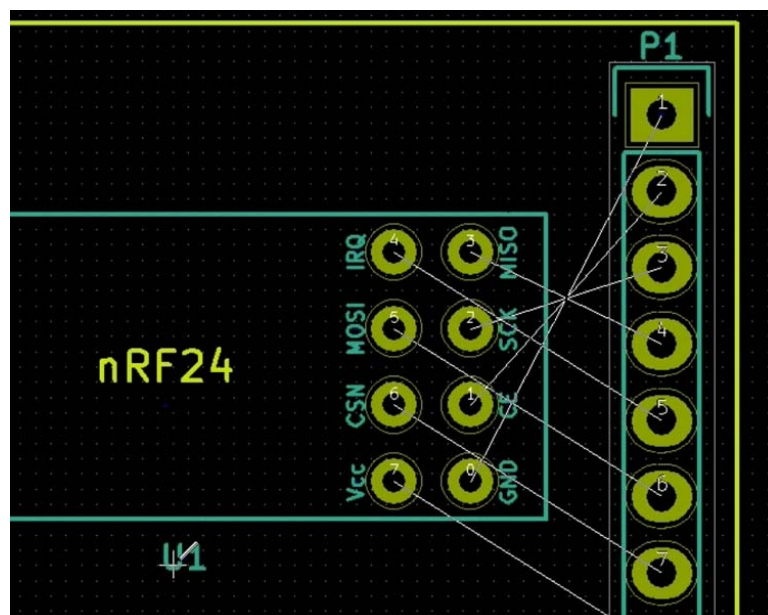
Close the 3D Viewer, and let's work on the text labels. If there is a text label that you prefer not to show in the final printed PCB, you can mark it as "invisible". For example, the "CONN_01x08" label can be made invisible. To mark it as such, put your mouse over it and hit the E key to edit it.



To make a label invisible, go to its properties and click on the “Invisible” radio button.

Click on the “Invisible” radio button, hit OK, and this label will now not show in the final printed PCB.

With the P1 label, I think it is useful to keep it. Try to move it inside the boundary, so that the end result is this:



I have moved the P1 label inside the boundary

Change the grid into something smaller, and edit the properties of the label to make its

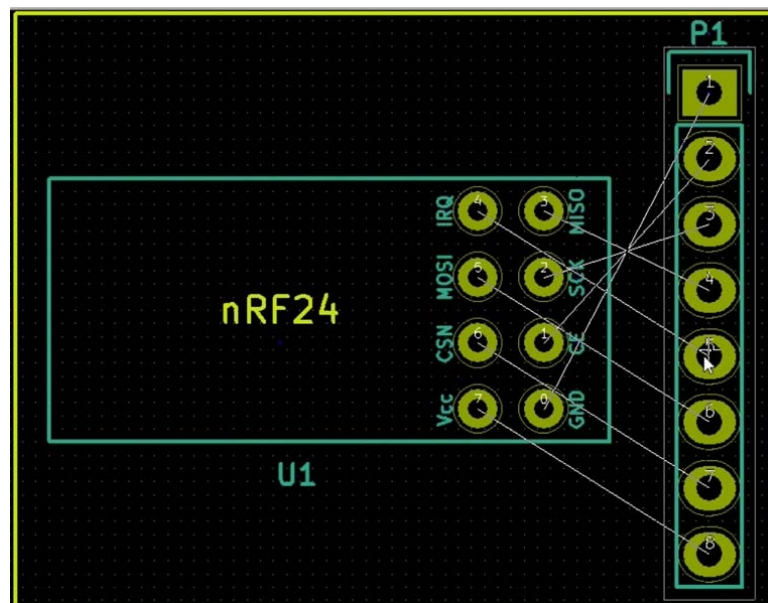
dimensions smaller if needed, so that it can fit in the space available. Do the same for same thing for U1.

In the next chapter, we will do the wiring.

Chapter 25: Wiring

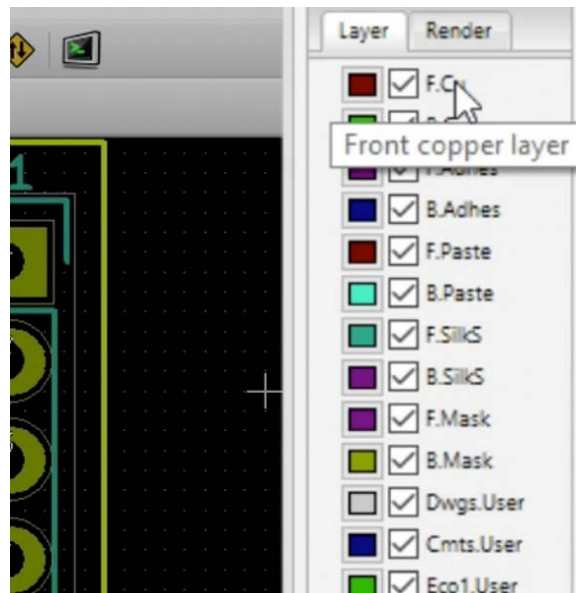
In this chapter we will do the wiring between the pins of the two footprints. This work is sometimes called “routing”, because it involves finding a good path for the connecting wires as they traverse the surface of the PCB. As you work on the routing, you have to be careful to avoid crossing two tracks (another word for “wire”), or stumbling into holes.

Let’s start the routing process with the nRF24 footprint. Our objective is to connect the nRF24 pins to the straight connector pins.



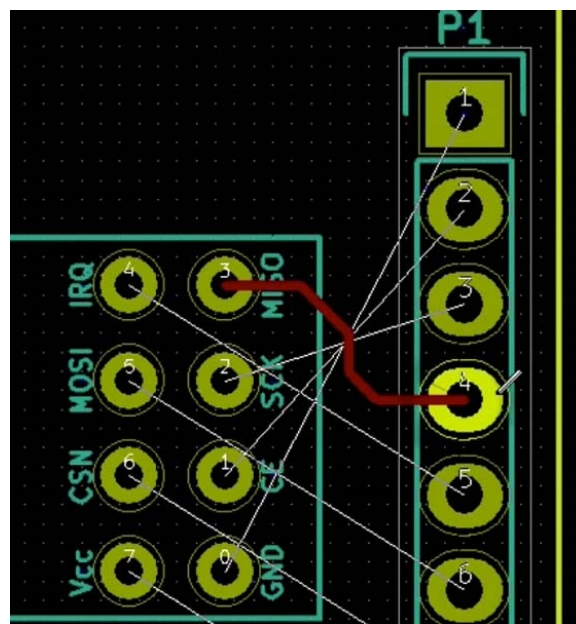
We connect the nRF24 pads with the corresponding pads of the straight connector. The ratsnests will help us distinguish the pad pairs.

To do the wiring, you first choose the layer on which you would like your wires to be on. For this first project, we will place all of the wires on the front copper layer, named “F.Cu”. In the other two projects we will use both the front and the back copper layer.



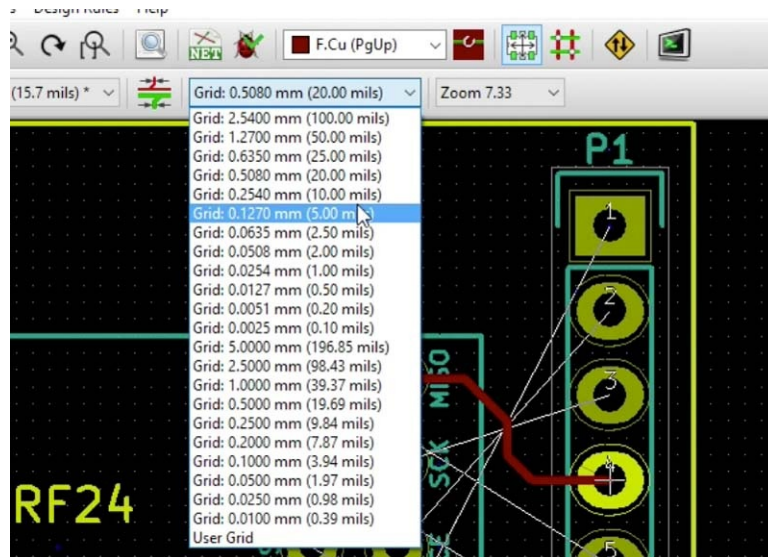
Select F.Cu to place the tracks in the front copper layer.

Select the front copper layer. Then hit the 'X' key to enable the wiring mode and simply click to start a wire, then click to change its orientation and its trace and then double-click once you reach the end.



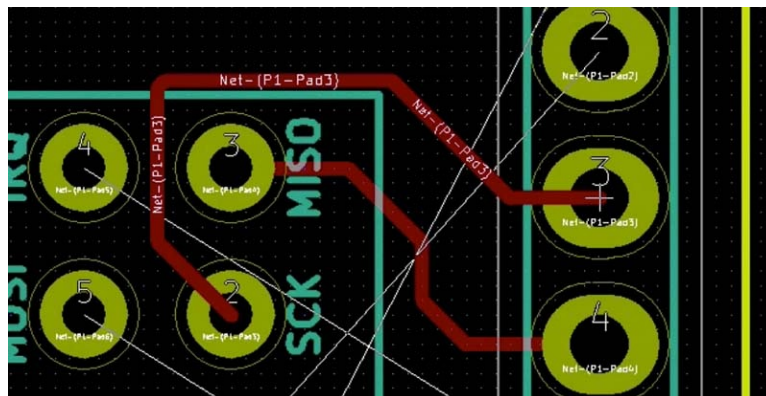
The first track.

Remember, you can change the grid for the tracks as well. So, if you want to have more control over how your track is moving throughout your PCB, you can change the grid to something smaller.



You can always adjust the grid to make it easier to draw a trace.

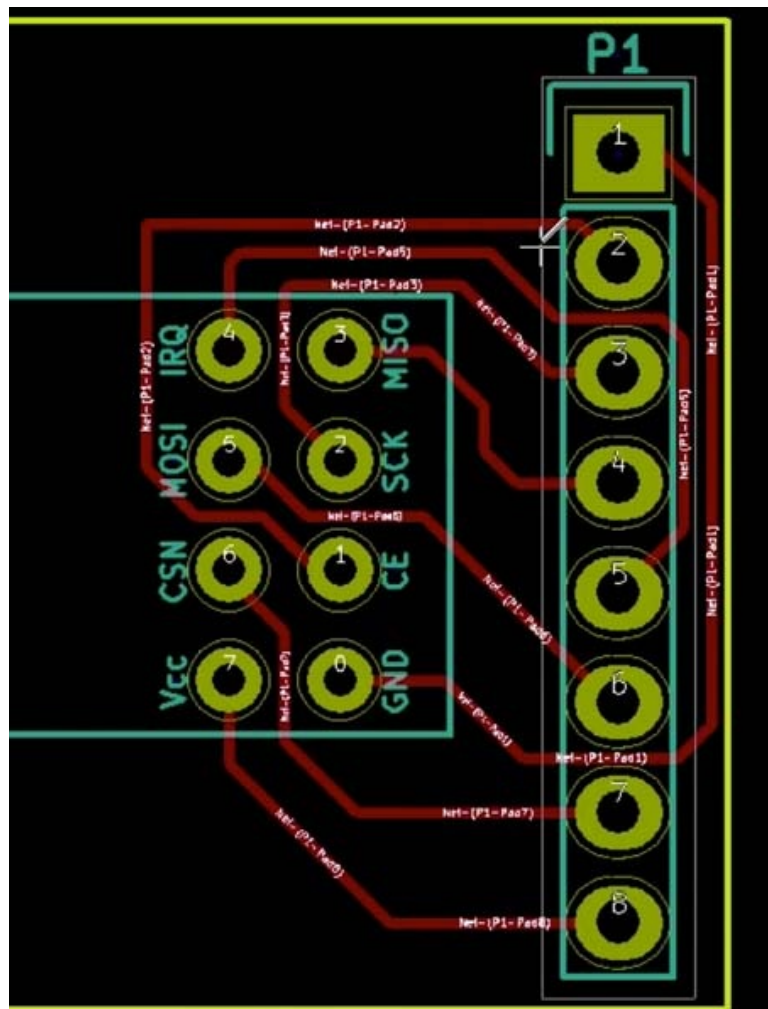
For example, if I go for something small as this, then I'll be able to have a bit finer control.



The second trace, created at a finer grid for more precise control.

Notice that if you try to cross a trace over an existing trace, Kicad will not allow you to close the it. Kicad is able to detect violation of electrical rules and enforce these rules.

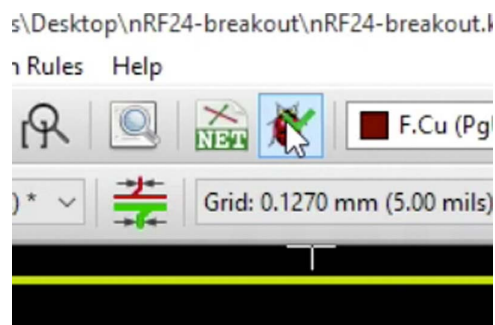
Continue with the wiring process, so that eventually your PCB will look something like this:



The completed wiring.

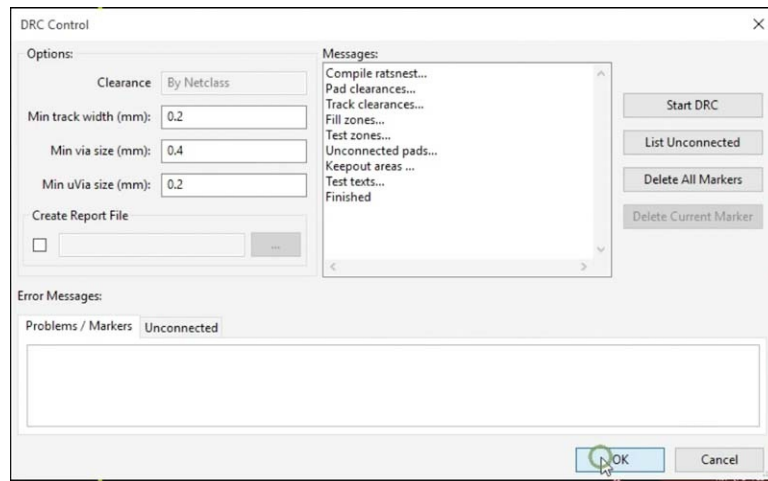
You can drag a single segment of a wire in order to adjust its position by typing 'G'. This is good for fine adjustments. If you make an error, you can also delete a segment (part of a wire) or a complete wire and try to route it again.

Before we do anything else, let's make sure that we haven't forgotten anything or that we haven't broken any design rules. Let's do a DRC check.



Electrical Rules Checking will save you from a lot of headaches. Do them often.

So, I click on the 'perform DRC check' button and start DRC.



This ERC returns an empty report, which means that there are no faults.

There is no message down here, which means that everything is fine. If we had forgotten to do a connection, the ERC would tell us that we have unconnected pads.

In the next chapter, I will show you how to add text labels in the top silkscreen layer of your PCB. These labels provide useful descriptions for the pins, and information about your PCB.

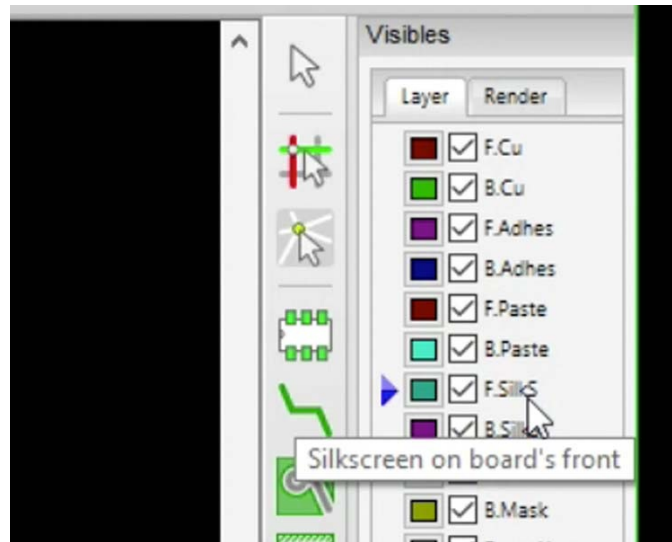
Chapter 26: *Add text labels*

In the last chapter, we completed the routing of the tracks. We are very close to completing this iteration of the board. What is left to do is to add text labels. With text labels, we can mark the purpose of pins, the names of the components on the board, as well as give our board a name and a version number. Text labels are placed on the silkscreen layer. In a later chapter, I will show you how to also add decorative graphics to this layer.



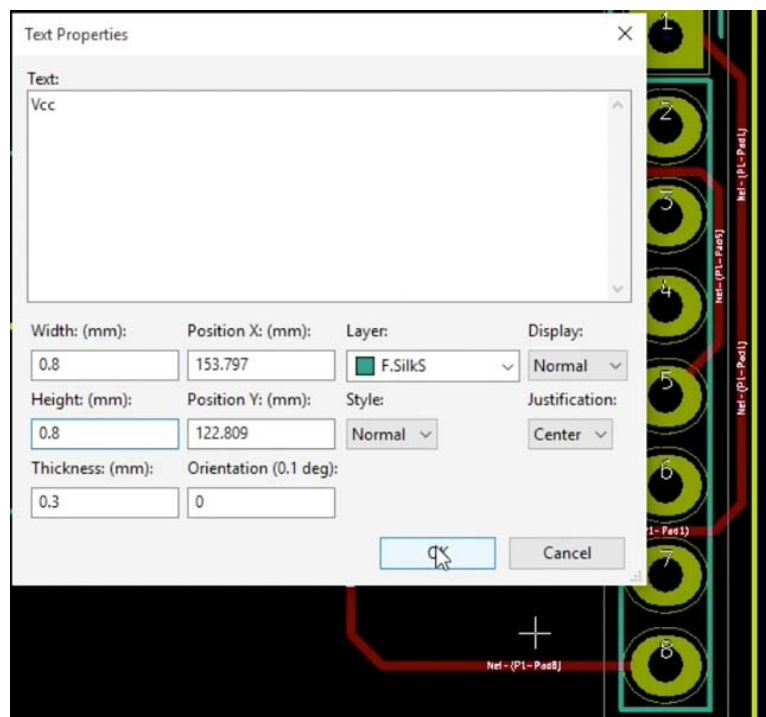
The Text tool allows you to add text labels on your board.

Next, ensure that you have selected the front silkscreen layer (“F.SilkS”) from the list of Layers on the right side of the Pcbnew window. There should be a small blue triangle marking the selected layer



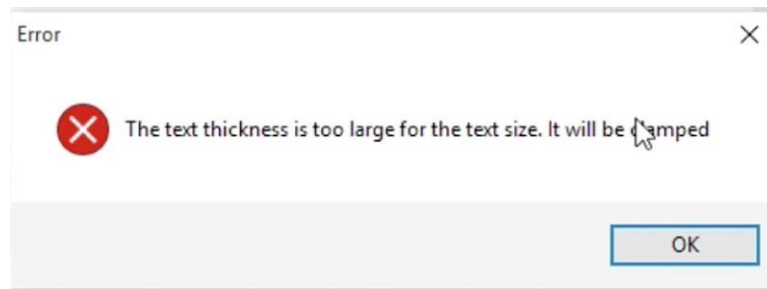
Ensure that a small blue triangle indicates the F.SilkS is the selected layer.

Let's start adding the text. Start with VCC. Click somewhere on the left side of the bottom pad of the straight connector. The properties window will come up. Adjust the text properties so that the width and height are set to 0.8 millimeters. The thickness will be adjusted by Kicad.



The text label properties window.

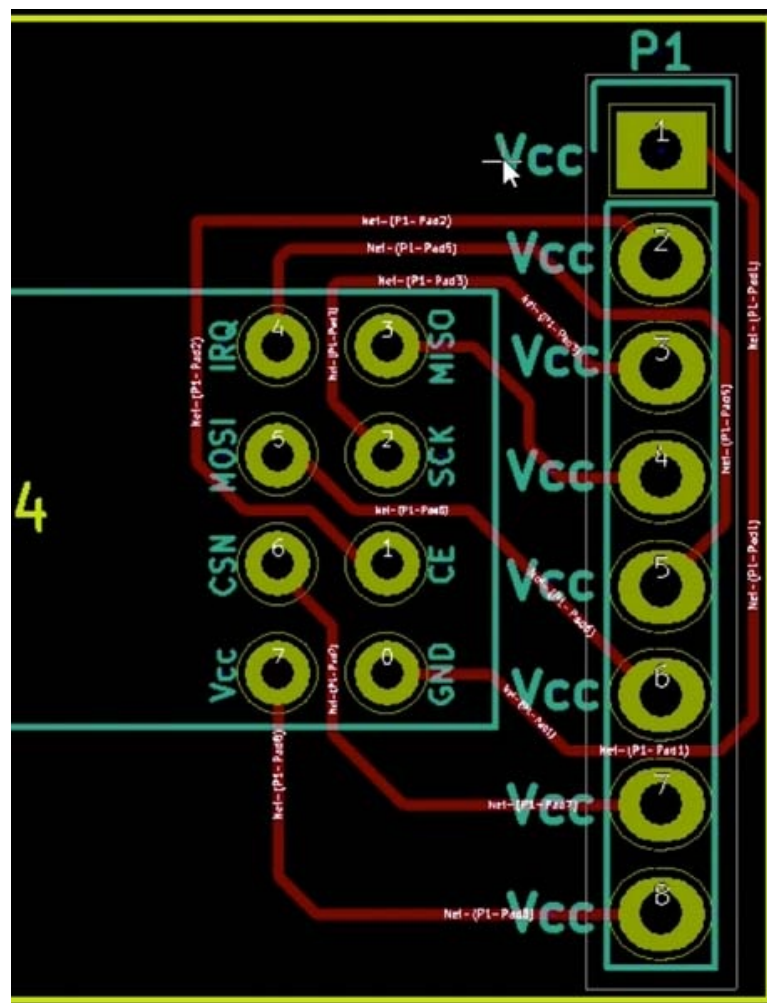
Click OK. You will see a warning message saying that the thickness is too large for the text size so Kicad will adjust it.



You can allow Kicad to adjust the text thickness based on the dimensions you chose in the properties window.

Click OK to accept Kicad's offer to adjust the thickness. You will now be able to fine-tune the position of the text next to the label. Click to fix the position. If you change your mind and you want to move the text label again, hit the "M" key.

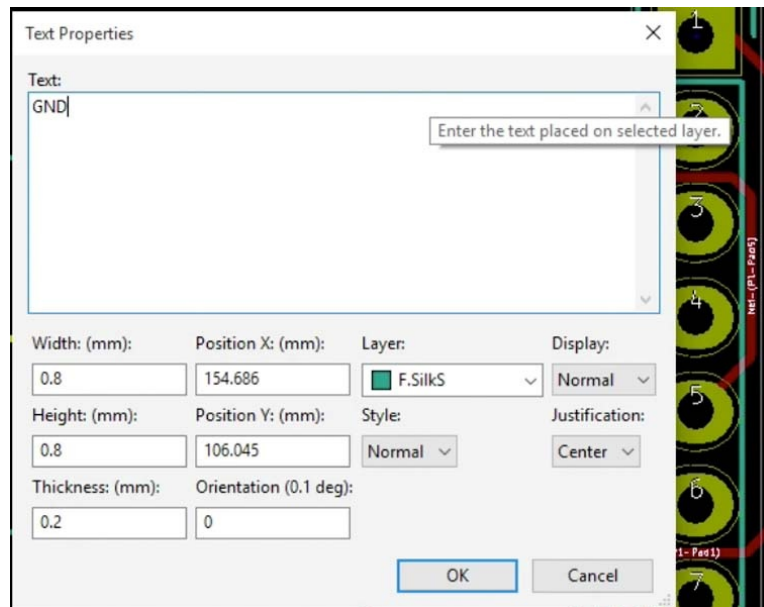
We should create the labels on this board so that they are also uniform in the way they look. Now that we have created the first label, we can duplicate it several time, and then simply change the text, but not their other properties. To do that, I'm going to put your mouse over the text and hit control D and this will create duplicates. Do that for all of the connectors. You can also do this by using the C key, as this creates a copy.



The board now contains duplicates of the original Vcc label. We'll adjust the text for the duplicates.

Now that the duplicate labels are in place, lets start editing their text. Start with the top

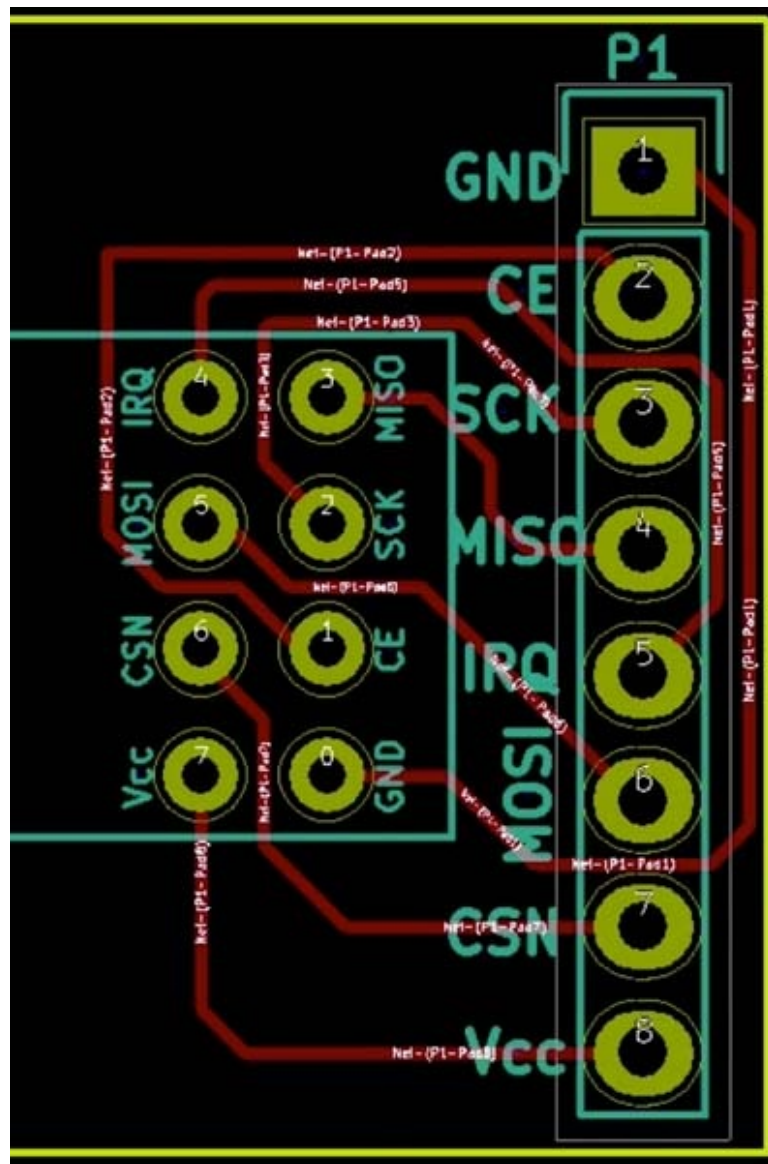
one, put your mouse cursor over it and type “E” to edit.



Change the original Vcc to GND.

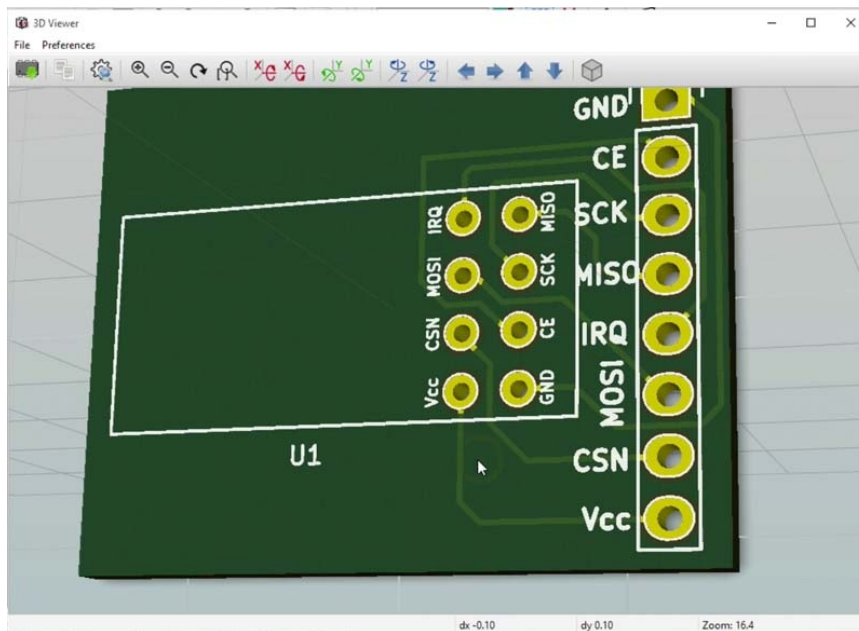
Change the original Vcc to GND, and hit OK.

Do the same for the rest of the duplicate labels, so that eventually you have something like this:

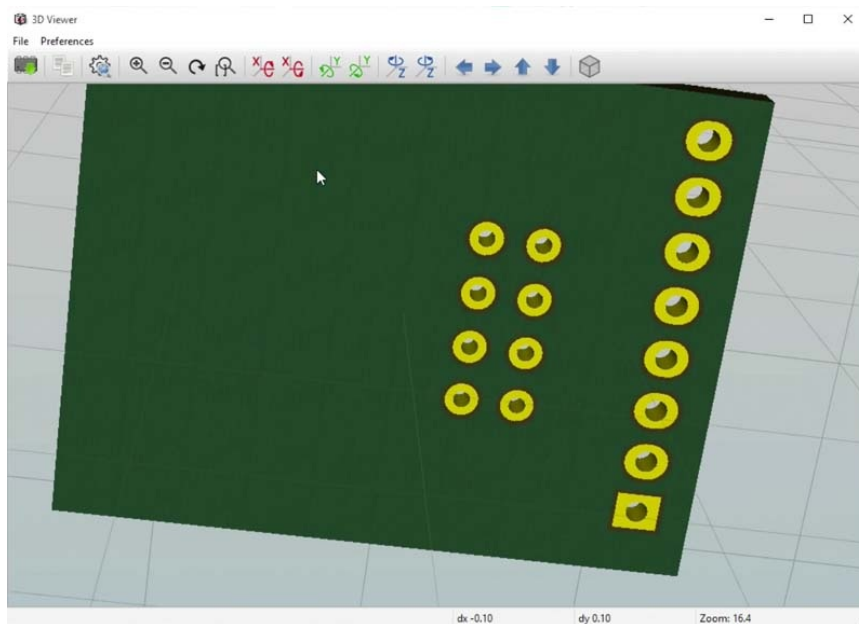


The straight connector labels, with their text edited.

Bring up the 3D view of the board to get an idea of how it looks like at the moment. Let's have a look at the 3D view off the breakout. Click on the View top menu item, and select 3D Viewer.



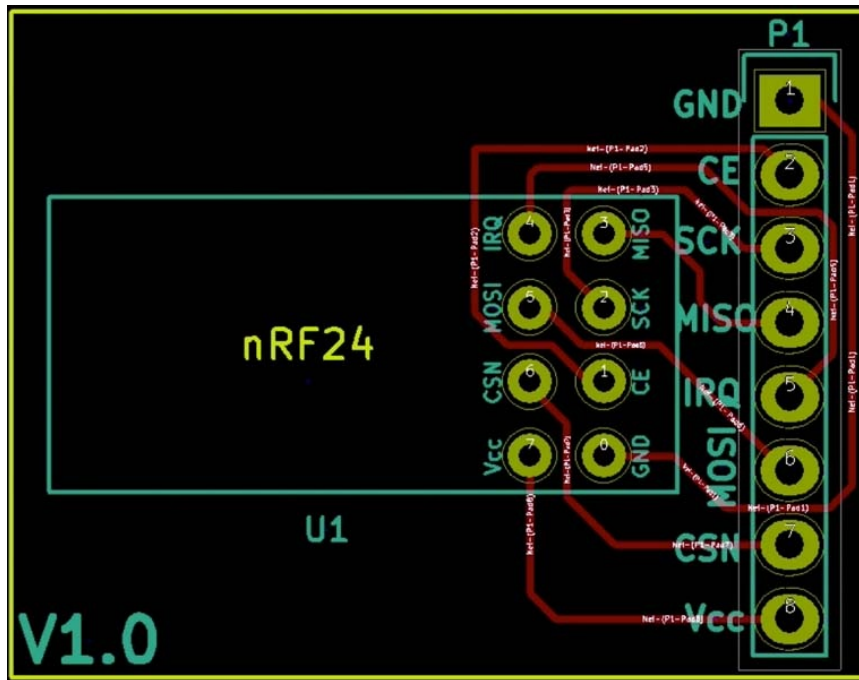
3D view, top side.



3D view, back side.

Notice that in the front side view, you can see the copper tracks connecting the pads. The back side has no copper tracks, since we are making a single-sided board.

One more piece of text I'd like to put on the board is the version number. Since designing a board is an iterative process, it is a good practice to always version them before you send them for manufacturing. This way, you will be able to tell them apart, especially if the difference between subsequent versions are small. Create a new label and place it on the board, so that you have something like this:



This is your board, V1.0!

This is your nRF24 breakout board, version 1!

After finishing with the process of creating version 1 of the board, I just realized that we should include a capacitor to improve the reliability of the NRF-24. I also want to have a look into the issue of power tracks and learn about copper fills. Let's explore these topics in the next section.

PART FOUR

Project 1: Enhancing the design

Chapter 27: *What is this part*

In this section, we will enhance the design of the nRF24 breakout by going through a second iteration.

We will add a bypass capacitor to the design, increase the width of the power tracks, and add a ground copper plane.

To add the capacitor, we will need to update the schematic diagram in Eeschema, and then update the layout in Pcbnew to include the new component. For the other two improvements, we will work only with Pcbnew.

Chapter 28: *Add a capacitor to the schematic using Eeschema*

The first version of the breakout is a good start, but considering how the nRF24 module is normally used, I realized that it can be improved. An obvious improvement is the inclusion of a capacitor on the breakout PCB, so that we don't have to plug one on the breadboard. The capacitor is used to help smooth out any ripple effects from the power supply that feeds the module. It also stores energy which is used for when the power supply can't provide enough. That's used for when you connect the module to a small Arduino or battery powered applications.

Two more things I would like to do in order to improve our design, is to add a copper fill for ground. The copper fill is simply an area on the board that is covered by copper, and connected to the ground pads. Ground copper fills (also known as "ground planes") have certain benefits, including reducing the amount of chemicals used in etching the copper from the board during manufacturing. Having a ground plane can make routing easy since a return path for the current can always be found nearby.

I would also like to adjust the width of the Vcc track, so that it's a little bit larger than the rest, again so, that it provides a bigger path with a lower impedance for current to flow through.

Both improvements are not strictly needed for this small project. They actually are not going to make any difference to the way that the circuit will behave, but they are a good common practice for any PCB design, regardless, and it takes very little effort to do, anyway.

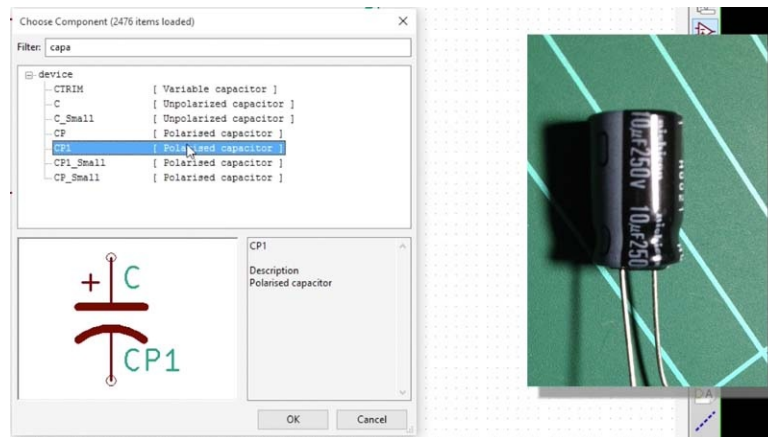
Let's go ahead and add the capacitor first.

First, we will update our schematic with the new component in Eeschema, then create a new netlist. After that, we will start Pcbnew and import the new netlist. This will, in effect, insert the new component to the layout editor, place it on the canvas, and do the wiring.

Start Eeschema and hit the 'A' key to add a new component.

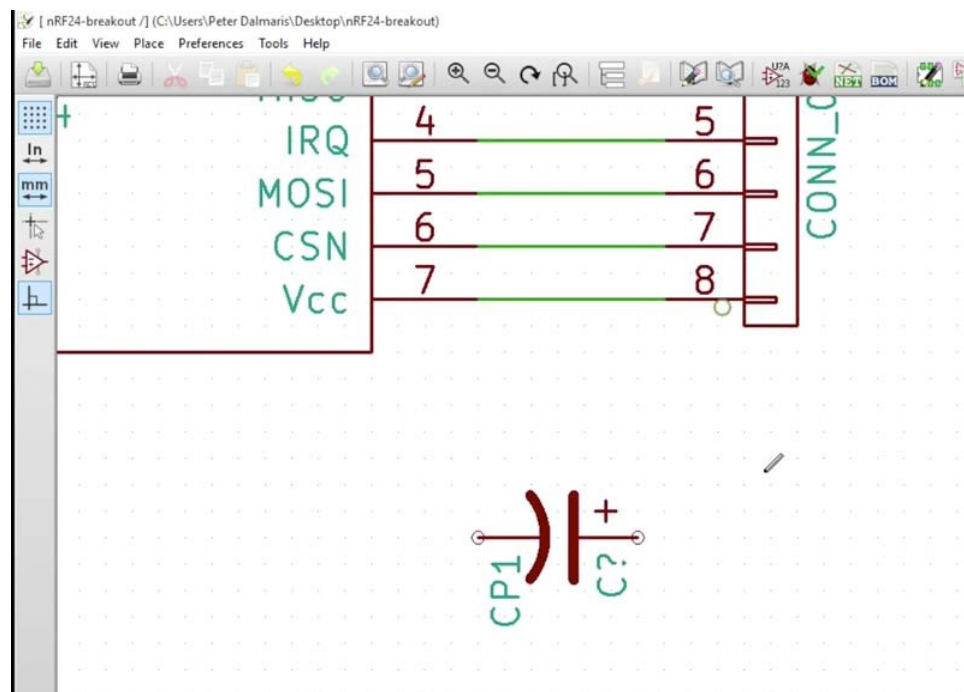
Let's search for a capacitor. I would like to use an electrolytic capacitor. The component chooser gives us a few options. Some of them are non-polarised capacitors,

some others are polarised. Our electrolytic capacitor is polarised, so I will choose an option with the appropriate symbol.



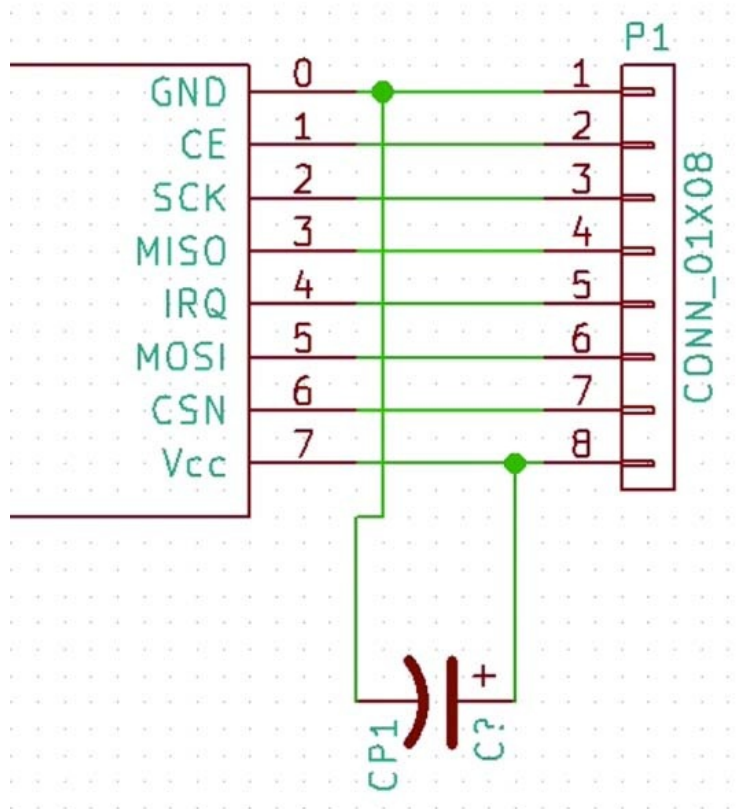
We will add a polarised capacitor; be careful to select the appropriate component.

Double-click on this option to insert this component to the canvas.



The new component is on the canvas now. Notice the “+” sign indicating the polarity of the capacitor, and the designator that contains a question mark.

You can see that the designator is got a question mark still, so it hasn't been given a number yet.

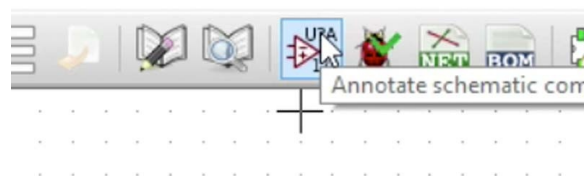


The capacitor is connected to the rest of the circuit.

Let's do the connections. Use the 'W' key to create a wire. Connect the negative pin of the capacitor to pin 0 of the nRF24 component. Connect the positive pin of the capacitor to pin 7 of the nRF24 component. To confirm that two wires are properly connected, look for a solid green dot at the junction of the two wires. If there is no dot, then the two wires are simply crossing, but not connected.

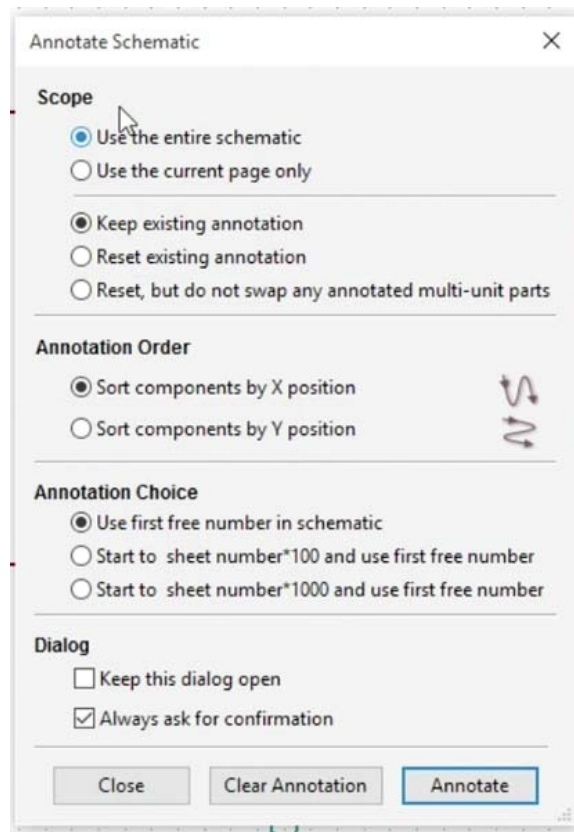
Don't forget to save your new schematic.

Let's continue with the the annotation.



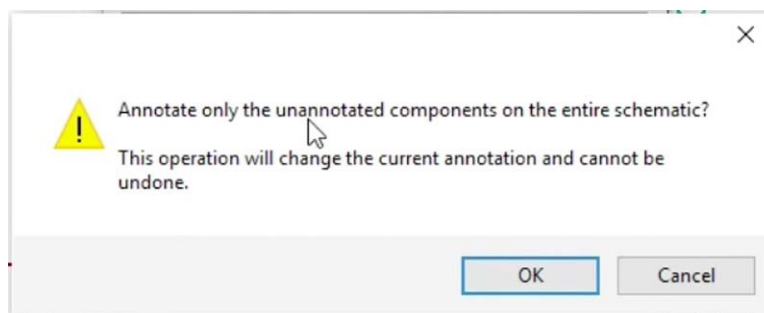
The annotation tool button.

Click on the 'Annotate Tool'.



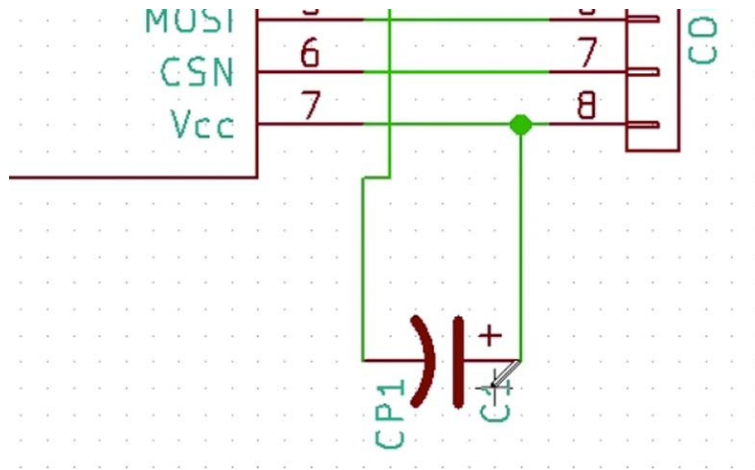
The Annotate Schematic window. The defaults are usually good to use.

The default options in the Annotate Schematic window are good as they are, so click 'Annotate'.



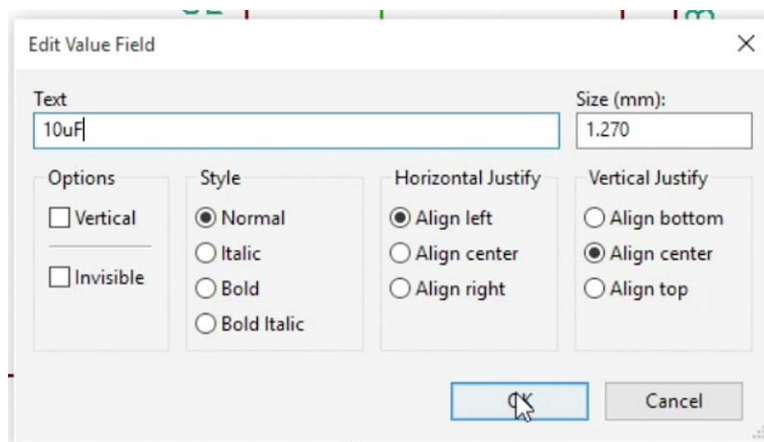
We only want to annotate unannotated components anyway.

You will see an information box explaining that only unannotated components will be annotated. This is exactly what we want, so click on OK to complete the process.



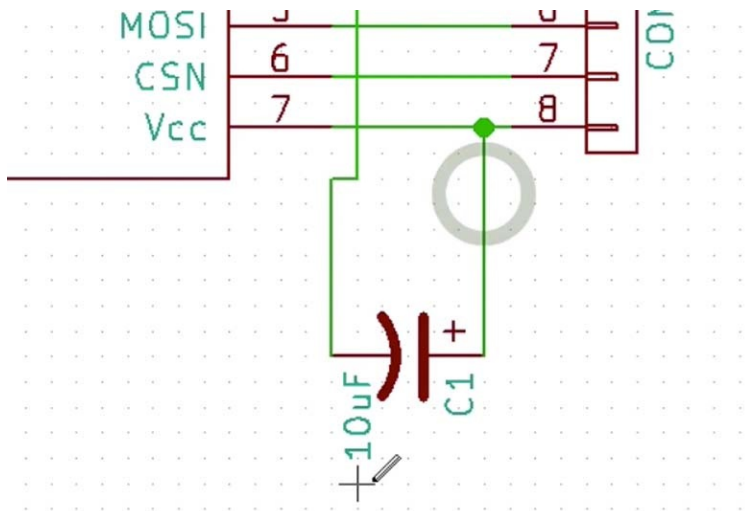
The capacitor now has a unique designator.

So, there's the designator for the capacitor, it's now C1. I would also like to assign the capacitor with a value, so that will show up on the schematic.



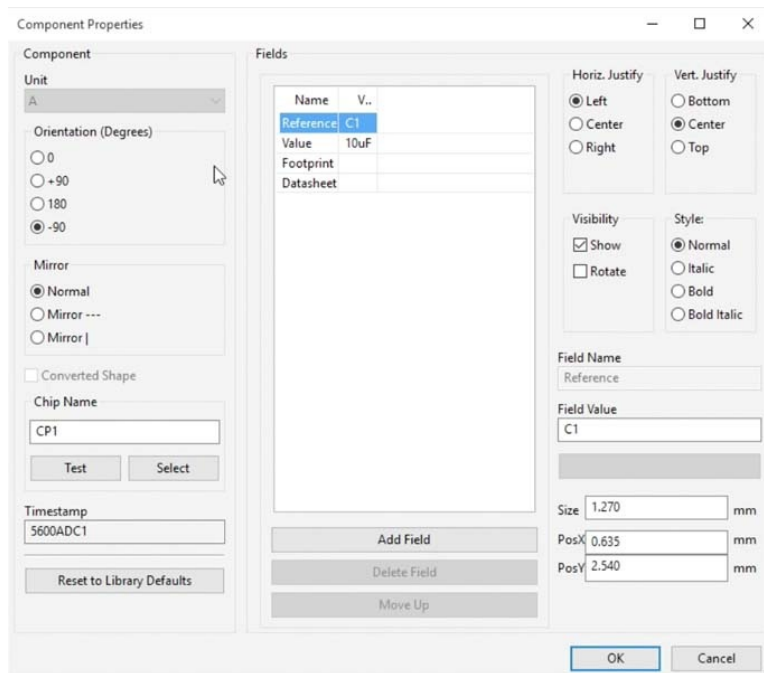
You can edit the value of a component by accessing it's Edit Value Field window.

Hit the 'V' key, as your cursor is over the capacitor. The Edit Value window will come up. Type the value "10 μ F" in the Text box, and click OK.



The capacitor value is showing in the schematic.

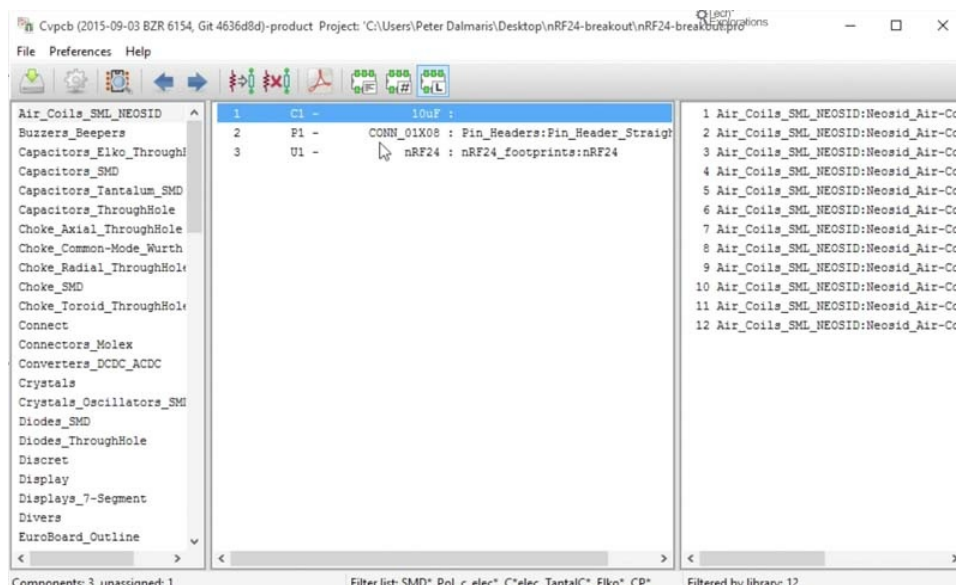
You could also have done the same thing by hitting the 'E' key.



Typing “E” will bring up the properties window for the component. You can edit the component value, as well as various other properties.

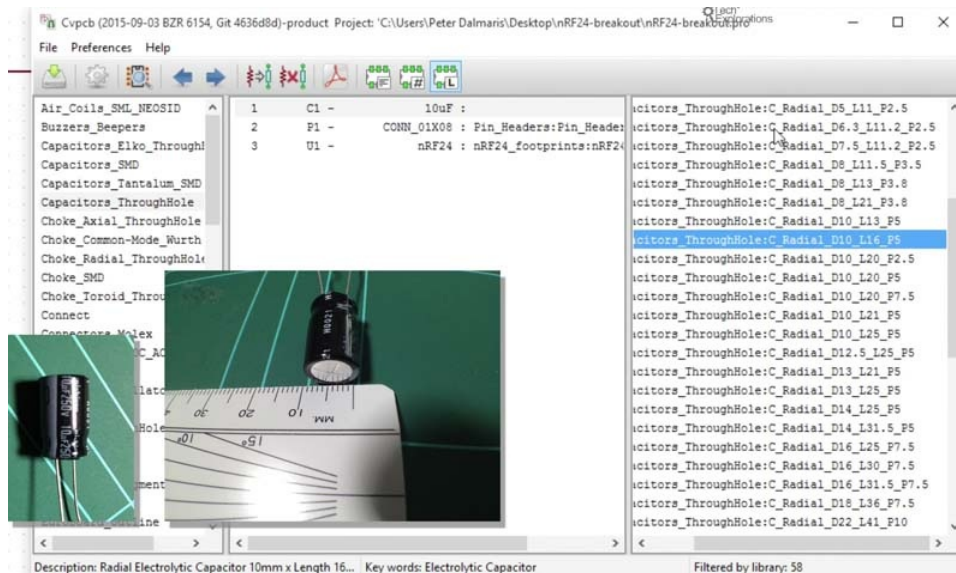
This would take you to the component’s properties window.

Next, let’s do the associations of this schematic component with a footprint. Start the Cypcb tool to do this.



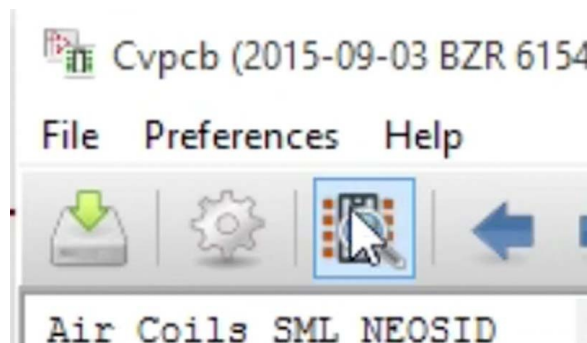
In Cypcb, you can see that the capacitor does not have an association yet.

You can see here that the new component does not have an association. Let’s look for a footprint for the capacitor. We need to find something that will have the appropriate size on the board.

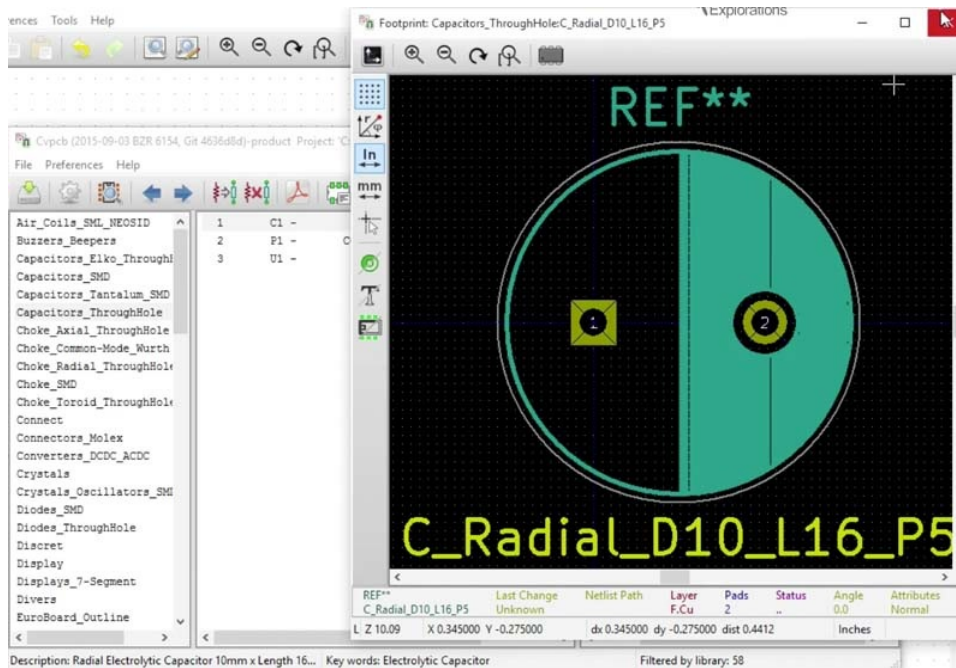


Find a footprint that has the right dimensions for the component.

Our capacitor is a through-hole component. You can use a ruler to measure the distance between the pins and the diameter at its base. You can see this in the photograph, that this capacitor is a radial capacitor with a diameter of 10 mm. You can look for a footprint in the Capacitors_Throughhole library. The contents of this library have footprints with the sizes included in the filename. This is very convenient! Look for one with “D10” in its name (for “Diameter”), and there is a good chance that this is the footprint you need.



Click on the preview button to see the selected footprint.

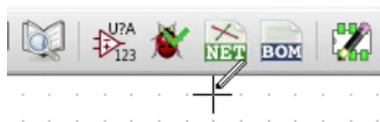


Inspect the footprint to ensure it is the right one for the component.

You can also look at the footprint preview in order to ensure that it is the right one for your component. You can have a look at the way that it looks like.

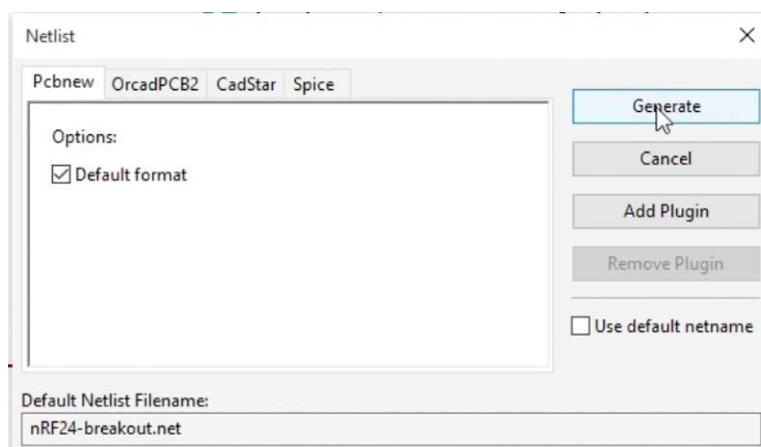
Double-click on the footprint titled “Capacitors_Throughhole:C_Radial_D10_L16_P5” to select it, and the association is complete.

Save the footprint associations and go back to Eeschema. Let’s generate the new netlist.



Click to generate a new netlist.

The Netlist dialogue, click on Generate. It is ok to overwrite the previous netlist.

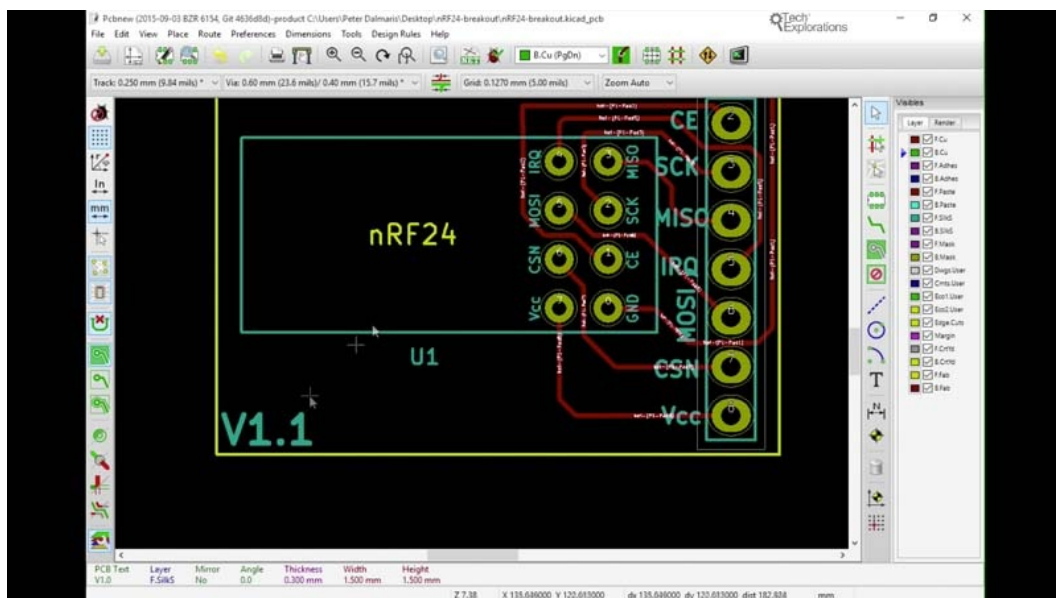


Click “Generate”. It is ok to overwrite the old netlist file since now it is out of date.

We now have a new netlist, so we can exit Eeschema and get into Pcbnew to do the layout and the wiring. We will do this in the next chapter.

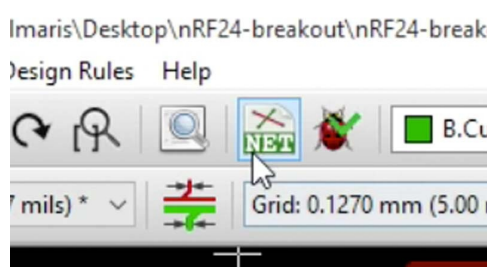
Chapter 29: Add a capacitor to the layout in Pcbnew

In this chapter, we will update the layout and wiring so that we can include the new capacitor in our design. Before we forget, let's change the version designator of the PCB to the new version designator which. Let's make it version 1.1. Remember, you can edit the text label by placing the mouse cursor over it and hating the “E” key, which will bring up the properties window for the label.

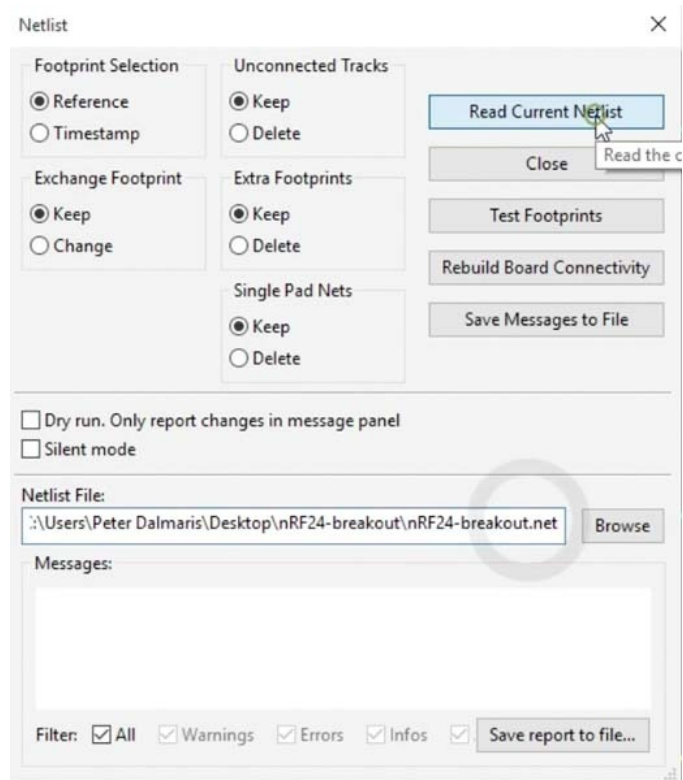


The board revision number is updated to 1.1

Next, let's import the new netlist. Click on the netlist button, and browse to the location of the netlist file.

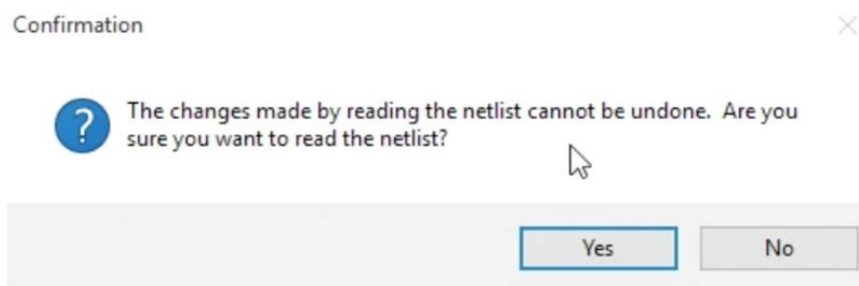


The netlist button.



The default settings are usually fine as they are. The Setlist file location should also be correct.

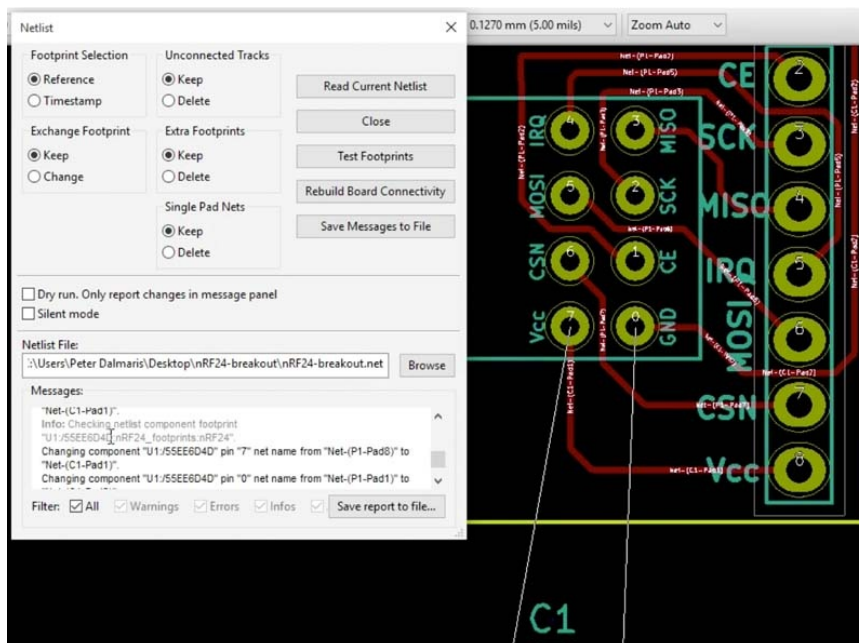
Click on Read Current Setlist to import the file. You will get a warning message:



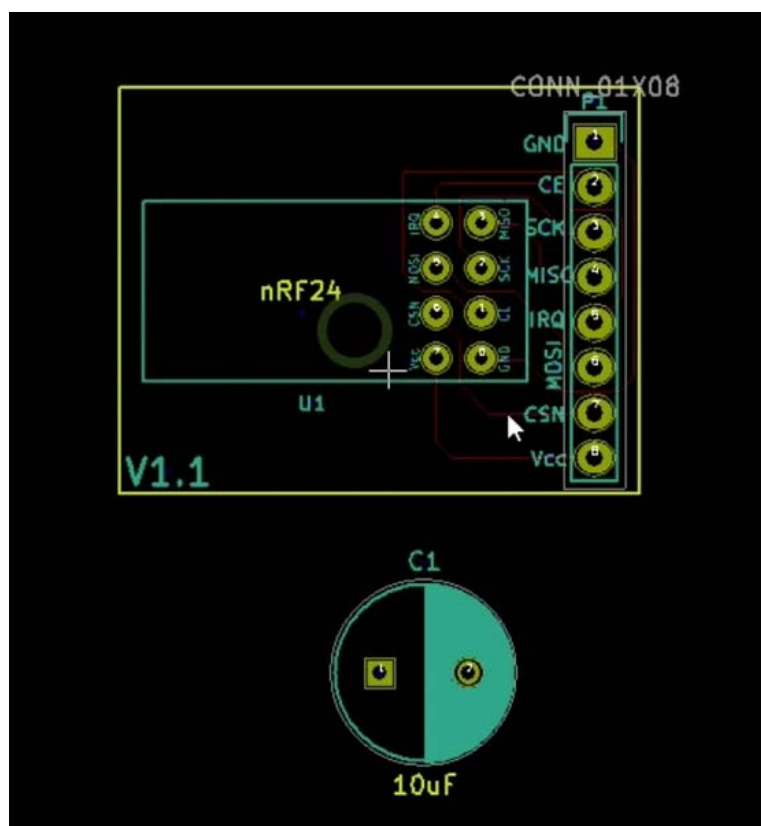
Yes, I am sure!

This message seems a little strong worded, but it is nothing to worry about. Reading the netlist file is exactly what we want to do, so click on Yes.

You can see in the messages text box confirmation that the new component has been added:

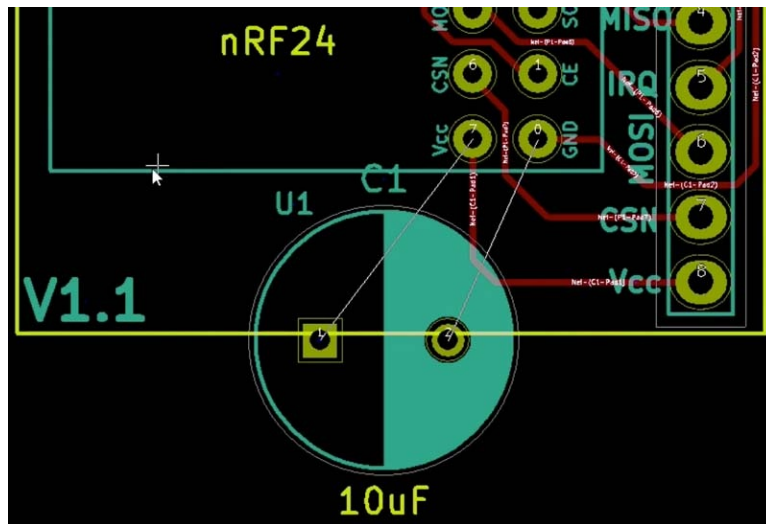


The Messages text box of the Netlist window contains confirmation that one new component was added to the canvas.



The capacitor footprint is automatically placed below the existing board.

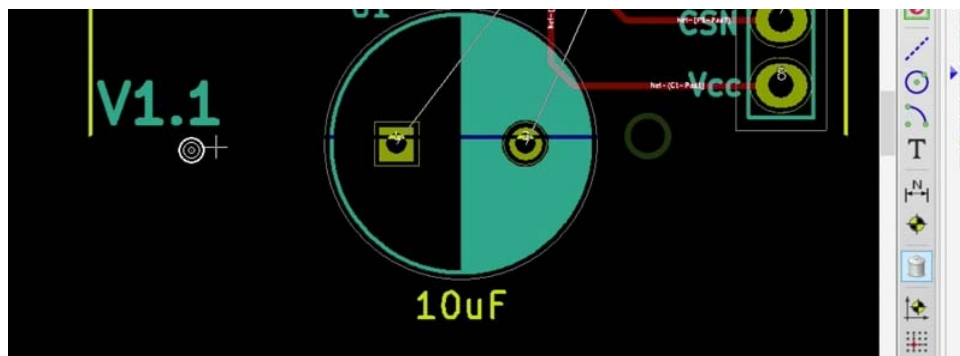
We can now position the new footprint. As a rule of thumb, we want to tracks from the capacitor pads to the VCC and ground pass to be as short as possible. Based on that, we should position the capacitor just below the bottom of the nRF24 component.



This position of the capacitor will allow us to draw short tracks to GND and Vcc. Notice the ratsnests indicating the pads that have to be connected.

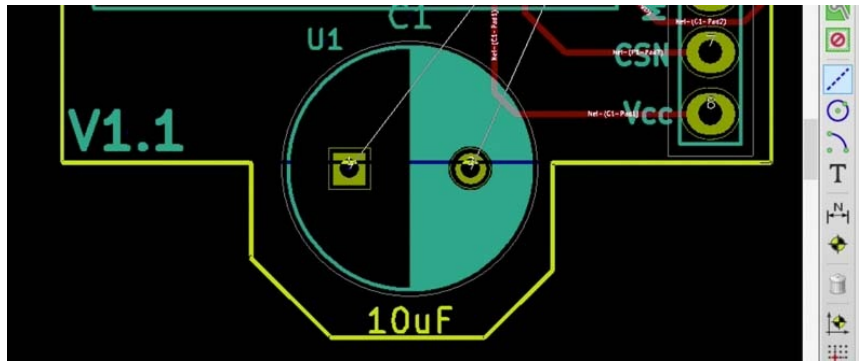
We should not try to place it any closer because we risk casing of the capacitor to be too close to the edge of the NRF module. This might make it too hard to actually mount the components on the board.

You can see that the capacitor, especially its pads, are outside the edges of the existing board border. We will have to change the bottom border so that there is more room for the capacitor. Let's update the edge cut next before we do the wiring. Select the edge cuts layer and delete the bottom part of the edge cut. To delete, click on the rubbish bin button.



The bottom edge of the border has been deleted.

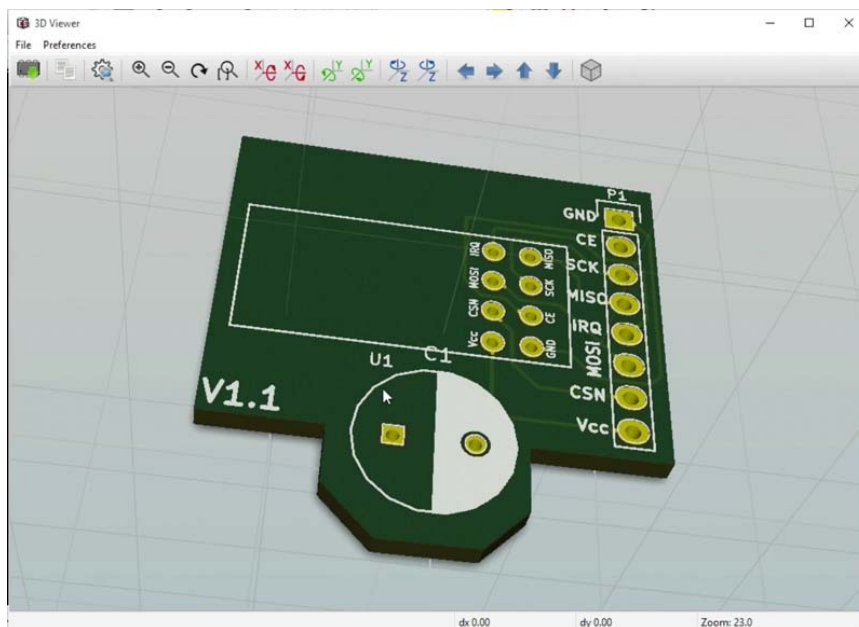
You can be a bit creating with the border edges. Click on the polygon tool so that you can draw a new border, and create something like this:



The new border, going around the capacitor.

I think it looks nice at least. You can improve on that of course, if you have a bit of patience you can make this look nice and rounded. In that case, I think this is good enough, so let's check it out.

You can use the 3D viewer to inspect the board at its current state:

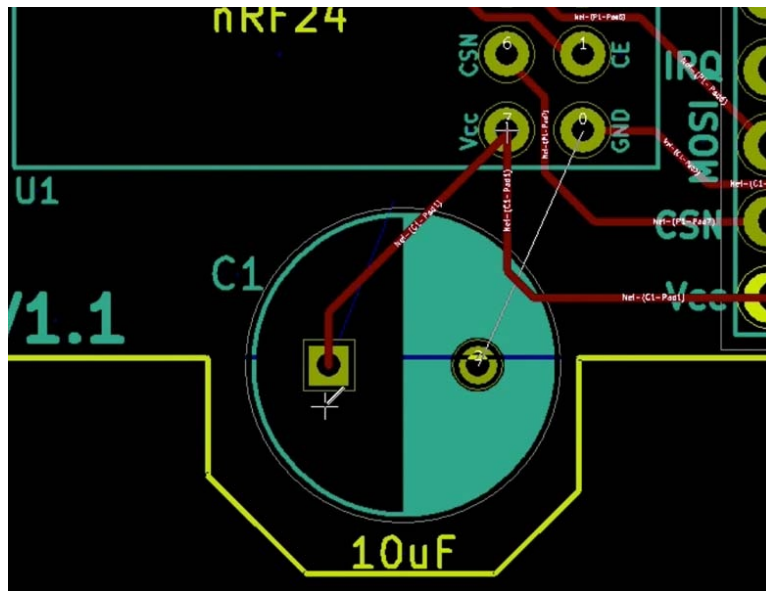


A 3D view of the board.

What does it look like now? It looks like this. A bit weird, but you can do interesting shapes in this way.

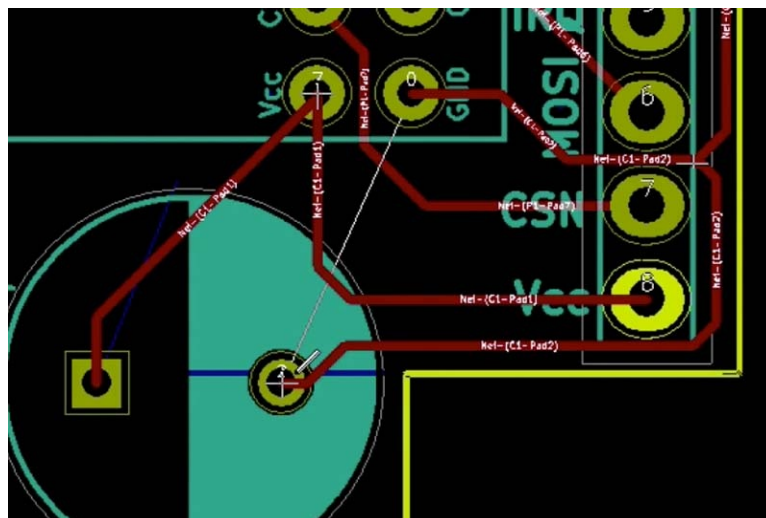
You can also move the labels to locations that look more appropriate to you. Feel free to experiment with this.

Let's now work on the wiring. Switch back to the front copper layer and hit the X key. Connect the VCC to the positive pin of the capacitor.



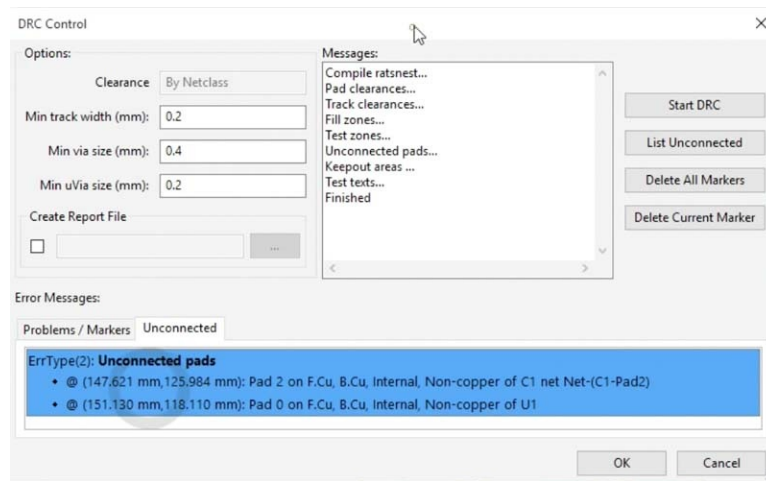
Connect the positive pad of the capacitor to the Vcc pad of the nRF24.

With the ground pin we need to be a bit more creative. You can see we can really route a wire directly from the negative pin of the capacitor to the GND pin of the nRF24 because there are two other wires in the way. But, we can do the routing by going around the side of the board and connect to the existing Ground wire, like this:



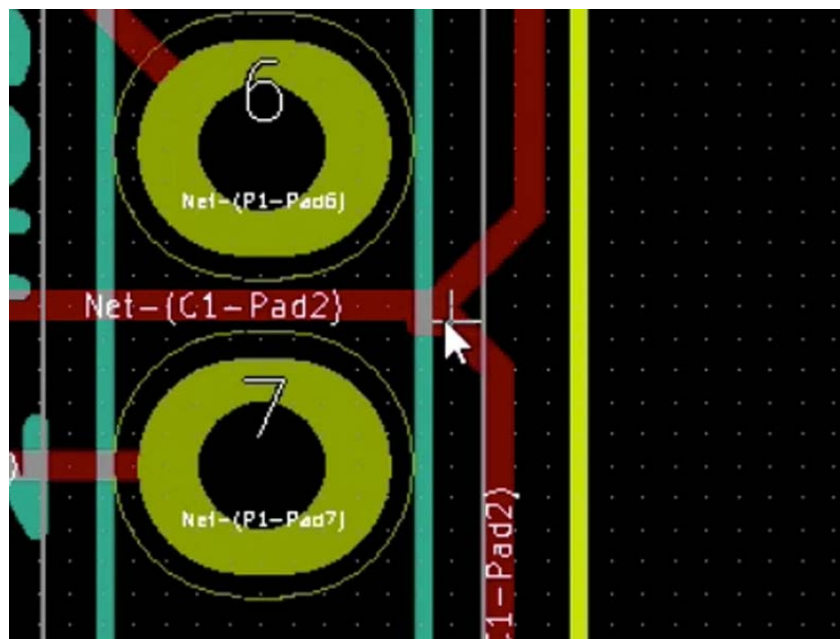
Attempt to connect the negative pin of the capacitor with an existing ground wire.

Notice that the ratnest line for the negative pin of the capacitor has not disappeared. This is even though we have done a connection between that pin and the existing GND wire. Why is that? Let's do an ERC.



The ERC is telling us that the capacitor negative pad and GND are not actually connected, even though they look connected.

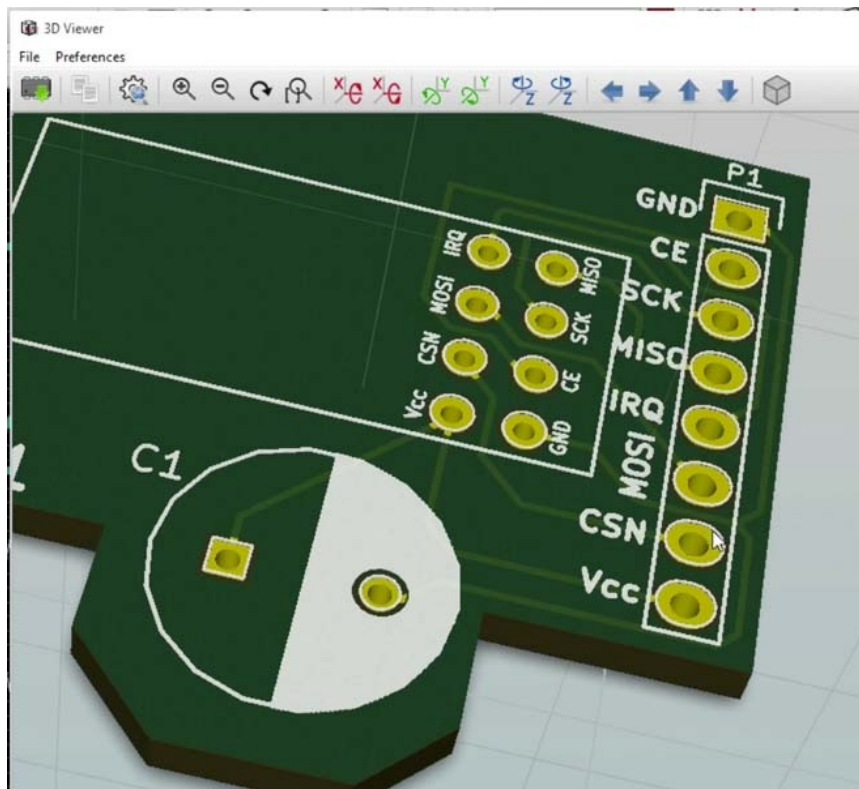
Even though the new wire **seems** to be properly connected to the negative pad of the capacitor and to the GND wire, it may actually not be properly connected. So, in situations like this, it helps to double check connections, and often to redo them in order to fix such annoying problems.



In my first attempt to connect the two wires, the connection was not successful. I had to try again to make it right.

The process of wiring can be fiddly. Remember that you can always use the ERC check to make sure you haven't forgotten anything even if it looks like it's connected, sometimes it's not really connected.

So, let's have another look at the PCB and what it looks like now in 3D.



The current 3D view of our board. You can see the capacitor and the new tracks.
You can see the wires for the capacitor connected to the rest of the board.

In the next lecture will look into improving some of the electrical characteristics of this PCB, and in particular, we'll look at the issue of the track width and copper plates.

Chapter 30: *Controlling the track width*

In this chapter we will improve the electrical characteristics of the PCB by increasing the width of the Ground and Vcc tracks. As far the features of this PCB are concerned, I think what we have now is enough. We could stop here and send it to the manufacturer but in general there are two more things that you can do in any design. First, you can take into account the current requirements of particular pads, especially those that convey power like in this case, VCC and ground. Also you can take into account the heat dissipation requirements.

For this board the main component is the NRF2401L+ module. In its documentation we can learn that at peak receive in transmitter operation, it will draw up to 14 milliamps of current. We should confirm that the tracks that convey power can accommodate this amount of current going through them. From my own experience, I know that this module can get a little hot when it is operating at peak. It doesn't get as hot as a CPU or RAM does but I would still like to take that into account and build my breakout board in a way that it manages the observed heat profile.

What I will show you next is not scientifically robust, but for our practical purposes it will be good enough. Let's start with the track width. You can control the width of each track manually or automatically. The automatic method works with nets, which is something I haven't discussed yet, but I will in the next project. The manual method involves creating custom track's widths and then applying one of those to a new or existing track.



To create custom track widths, select Design Rules.

Let's go ahead and add a few custom track widths that should cover 99% of your

requirements. To do that, let's go to design rules and check on the first option design rules.

Design Rules Editor

Net Classes Editor | **Global Design Rules**

Via Options:

Blind/buried Vias:

- ☒ Do not allow blind/buried vias
- ☐ Allow blind/buried vias

Micro Vias:

- ☒ Do not allow micro vias
- ☐ Allow micro vias

Minimum Allowed Values:

Min track width (mm): 0.2

Min via diameter (mm): 0.4

Min via drill dia (mm): 0.3

Min uvia diameter (mm): 0.2

Min uvia drill dia (mm): 0.1

Specific via diameters and track widths, which can be used to replace default Netclass values on demand, for arbitrary vias or track segments.

Custom Via Sizes:

Drill value: a blank or 0 => default Netclass value

	Diameter	Drill
Via 1		
Via 2		
Via 3		
Via 4		
Via 5		
Via 6		
Via 7		
Via 8		
Via 9		
Via 10		
Via 11		
Via 12		

Custom Track Widths:

	Width
Track 1	
Track 2	
Track 3	
Track 4	
Track 5	
Track 6	
Track 7	
Track 8	
Track 9	
Track 10	
Track 11	
Track 12	

Messages:

Current general settings:
Minimum value for tracks width: 0.2 mm
Minimum value for vias diameter: 0.4 mm
Minimum value for microvias diameter: 0.2 mm

OK Cancel

Once in the Design Rules Editor, click on the Global Design Rules tab.

Switch to the global design rules tab. Notice, in the middle area of the window, two empty grids. On the left side you can see the Custom Via Sizes grid, and on the right side is the Custom Track Width. We will populate these grids with our own values. Let's work on the widths first. I looked up the manufacturing specifications for OSH Park and I found the typical widths that they support. I will use these numbers here. These numbers tend to be used across the industry, so chances are they will be right for many other fabricators.

Please look at the screenshot below and copy the custom widths to your Custom Track Widths grid:

Custom Track Widths:

	Width	
Track 1	0.1524	
Track 2	0.254	
Track 3	0.381	
Track 4	0.508	
Track 5	0.8128	
Track 6		
Track 7		
Track 8		
Track 9		
Track 10		
Track 11		
Track 12		

The custom track width sizes that work with OSHPark (and many other fabricators).

Custom tracks 3, 4 and 5 are very large and unlikely I will ever use them, but we can include them for the sake of completeness.

Custom Via Sizes:

Drill value: a blank or 0 => default Netclass value

	Diameter	Drill
Via 1	0.4826	0.3302
Via 2	0.5	0.4
Via 3	1.905	0.254
Via 4		
Via 5		
Via 6		
Via 7		
Via 8		
Via 9		
Via 10		
Via 11		
Via 12		

The custom via sizes that work with OSHPark (and many other fabricators).

Even though we are not going to be using any vias in this project, we will in the next

project. We might as well put them in here to have them ready to use later. Copy the values from the screenshot above to your Custom Via Sizes grid.

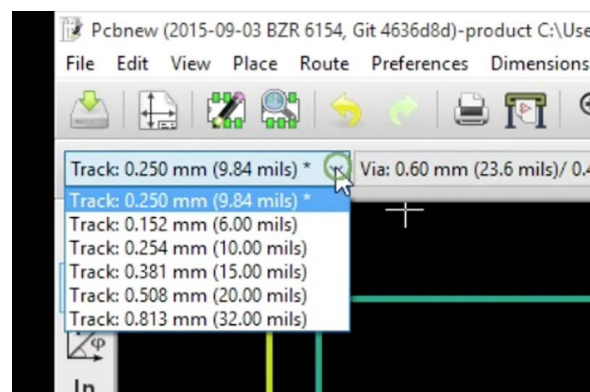
Minimum Allowed Values:

Min track width (mm):	0.1524
Min via diameter (mm):	0.4
Min via drill dia (mm):	0.3
Min uvvia diameter (mm):	0.2
Min uvvia drill dia (mm):	0.1

The minimum allowed values must be at least equal to the smallest values in the two grids.

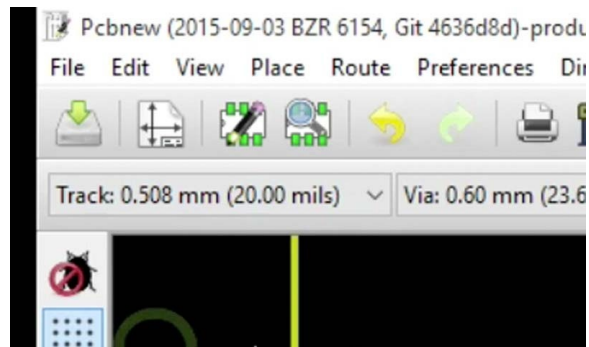
At the top right side of the window you will see the minimum allowed values group. None of the values in the two grids of this window can be smaller than the corresponding values in this group. I have entered the correct values to satisfy the minimum value rules in the Minimum Allowed Values group, so feel free to copy these values across to your Design Rules Editor.

When you are finished, click on OK to commit the changes.

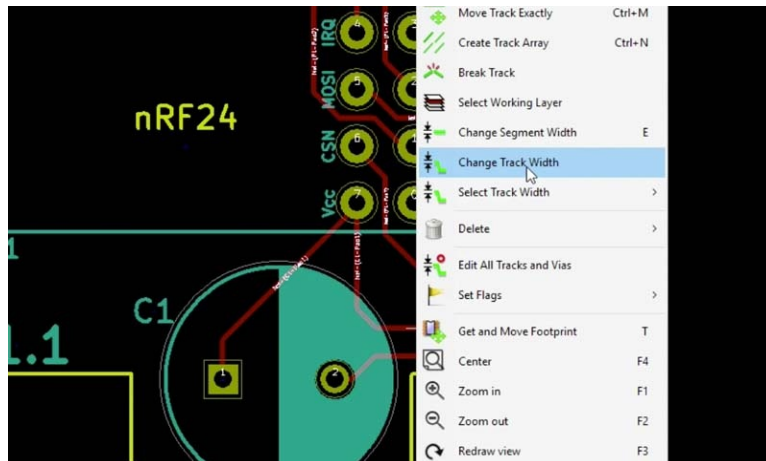


The custom track widths now appear in the width drop down menu.

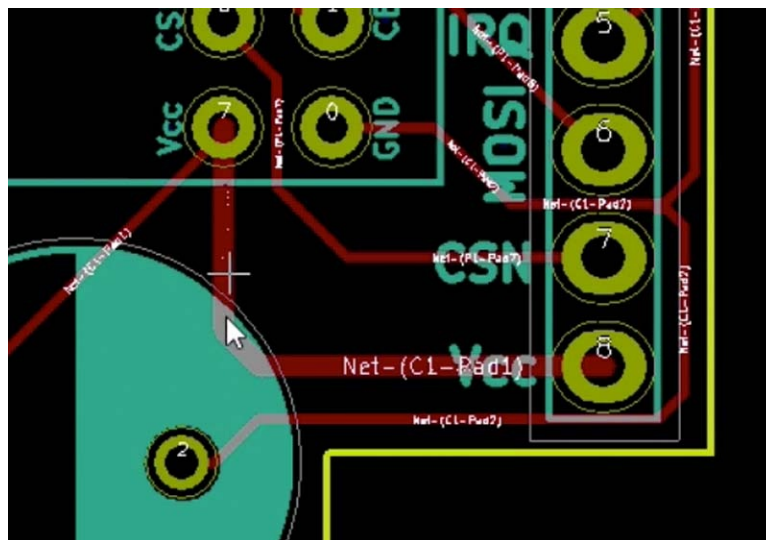
You can now see that the values that we have just entered are available from the Track Width drop down menu. Let's change the width of one of the existing tracks. Let's change the width of the Vcc track to something bigger. First, choose the new dimension, for example, 0.508 millimeters.



First, choose the desired track width.



Second, right click on the track you want to change, and select “Change Track Width”

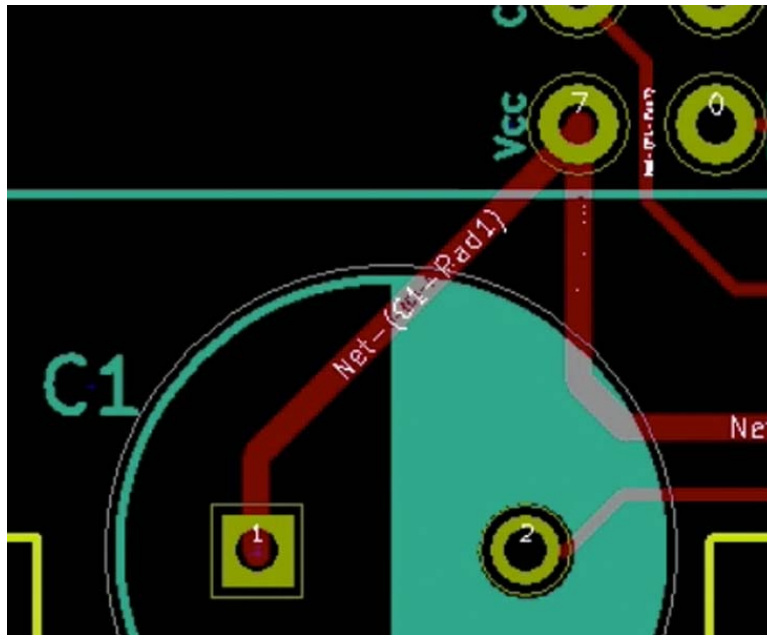


Notice that the track is now thicker.

Then, go to the track that you want to adjust the width for, right click on it and select change track width. You can see that it has become thicker. You can do the same thing for all the Vcc lines.

For the ground tracks, I would like to also make it thicker. Taking the available space into account, I will choose to make the Ground tracks 0.381mm wide. Again, choose this value from the Track Width drop down menu, then right click on the Ground tracks and

select Change Track Width.



This GND track is adjusted to 0.381mm width.

That should be okay. I will do a check in a second. Same thing for this. It will change the whole track not just the segment. If you want to just change the segment then you can just choose, change segment width.

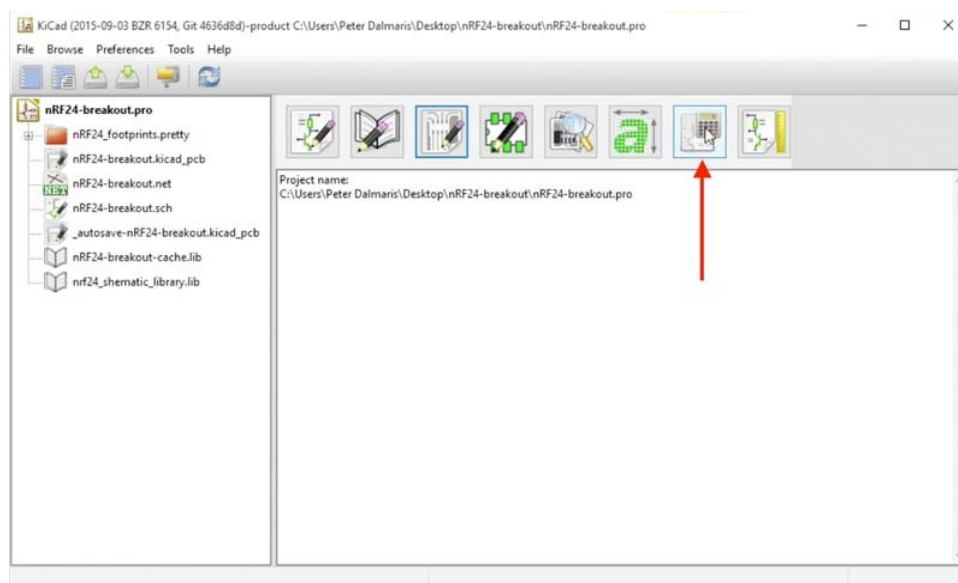
In this chapter we decided on the track width for the Vcc and GND tracks by simply guessing what values might be ok. There is not much science in this. In the next chapter, I will show you how to use Kicad's built-in track width calculator to calculate, instead of guessing, the appropriate track width.

Chapter 31: *Calculate the appropriate track width*

In the last chapter, you learned how to customise the width of a track. We didn't concern ourselves with figuring out the appropriate width for a track though, but only have to control the width.

In this chapter, I will show you how to use Kicad's built-in track width calculator to calculate (instead of guessing) the minimum track width. Notice that I used the word “minimum”. I did this because in general, bigger track widths are better, and the Kicad's calculator will give you the minimum width for which your board can operate properly based on the information you entered. This is why very often, at least for hobbyist PCBs, you can safely skip the calculator and just choose large width values with a lot of “margin of safety” in them.

This will become more clear after you complete this chapter.



Start the Track Width Calculator from the main Kicad window.

To start the calculator, go back to the main KiCad window and click on the PCB calculator button.

PCB Calculator

Regulators Track Width Electrical Spacing TransLine RF Attenuators Color Code Board Classes

Parameters

Current: 1.0 A

Temperature rise: 10.0 deg C

Conductor length: 20 mm

Resistivity: 1.72e-8 Ohm-meter

External layer traces

Trace width: 0.300387 mm

Trace thickness: 0.035 mm

Cross-section area: 0.0105135 mm x mm

Resistance: 0.0327197 Ohm

Voltage drop: 0.0327197 Volt

Power loss: 0.0327197 Watt

Internal layer traces

Trace width: 0.781437 mm

Trace thickness: 0.035 mm

Cross-section area: 0.0273503 mm x mm

Resistance: 0.0125776 Ohm

Voltage drop: 0.0125776 Volt

Power loss: 0.0125776 Watt

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 35A (external) or 17.5A (internal), temperature rises up to 100 deg C, and widths of up to 400mil (10mm).
 The formula, from IPC 2221, is

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

The Track Width Calculator is in the second tab of the PCB calculator app.

Click on the second tab, 'Track Width'. We know that from the documentation of our main component, the RF24, that at peak it will draw 40 milliamps of power. Type this figure in the Current field of the Parameters group, in Amps. So, you should type "0.014" (A).

For the temperature rise, we should set it 20 degrees Centigrade because I don't see that this component will be operating in temperature variations more than 20 degrees above room temperature and below room temperature. Therefore, a 20 degrees spread seems like reasonable value here.

The conductor length is not going to be more than 20 millimeters. You can measure that with the measurement tools that I showed you earlier but I don't think that the conductor length is going to be more than 20 millimeters. We don't need to be too accurate about this in our circuit because it is a low frequency, low power circuit. If you were working on a high-frequency circuit, parameter like this start to become very important. Let's leave that as 20 millimeters as the conductor length.

Regulators Track Width Electrical Spacing TransLine RF Attenuators Color Code Board Classes

Parameters

Current: 0.014 A

Temperature rise: 20.0 deg C

Conductor length: 20 mm

Resistivity: 1.72e-8 Ohm-meter

External layer traces

Trace width: 0.000546917 mm

Trace thickness: 0.035 mm

Cross-section area: 1.91421e-005 mm x mm

Resistance: 17.9709 Ohm

Voltage drop: 0.251592 Volt

Power loss: 0.00352229 Watt

After entering the desired values in the Parameters group, the calculated Trace width appears in the External layer traces group, on the right side, in the Trace Width box.

You can see that as I am filling these values in the parameters top left corner of the calculator, the calculator is automatically calculating the trace width and thickness for these characteristics. You can see in the External Layer Traces, the minimum Trace Width

is calculated to be 0.00054mm. This is the minimum trace width that would allow for this much current to flow through at this temperature rise for a conductor that is that long.

In the previous chapter, we set the ground traces to 0.381mm. This value is much, much larger than the calculated minimum. We are well and truly covered in terms of trace width. This is just confirmation that the thickness that I have chosen is sufficient.

Internal layer traces		
Trace width	<input type="text" value="0.00142277"/>	<input type="text" value="mm"/>
Trace thickness	<input type="text" value="0.035"/>	<input type="text" value="mm"/>
Cross-section area	4.9797e-005	mm x mm
Resistance	6.90804	Ohm
Voltage drop	0.0967126	Volt
Power loss	0.00135398	Watt

The Track Width calculator contains the Internal Layer Traces group. In it you will find the appropriate trace width for a trace that placed inside the PCB, as opposed to its surface.

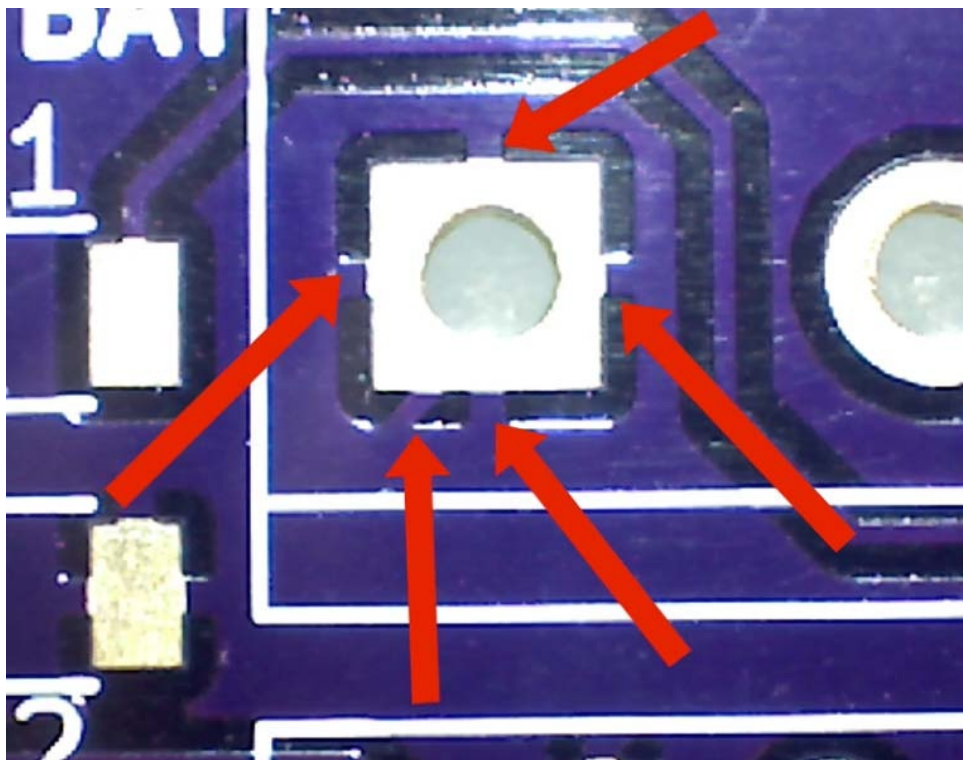
Notice that the track width calculator contains a group titled “Internal layer traces”. This group contains calculations for tracks that are placed inside the PCB, as opposed to its surface. In our example, the traces are all on the top layer. In the next two projects, they traces will be in either the top or the bottom trace. If you happened to be building a PCB with more than 2 layers, the you will need to use the numbers in this group to work out trace widths for the internal layers.

Based on the calculator findings it turns out that our current width for the Vcc track are orders of magnitude more than the minimum. But it’s better to be safe than sorry. This means it is better to have a wider track then a narrower track. These tracks are well within the minimum so I’ll leave them as they are. Before we end this chapter, do an ERC to make sure that we haven’t violated any rules.

In the next chapter, we will add copper fills to the board.

Chapter 32: *Adding copper fills*

An area on the PCB that is covered with a thin copper layer is called copper fill. Copper fills are used to create a ground plane. A ground plane is an area on the PCB that is grounded by being connected to one or more ground pads. Pads that should be connected to ground can be connected to the ground plane via small traces. These traces are often called “thermal reliefs”.

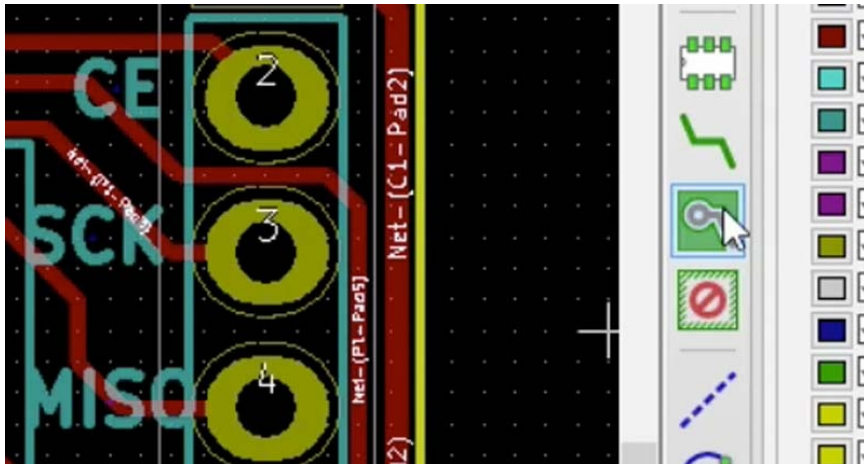


An example of a thermal relief. Notice the small traces connecting the bright coloured pad in the middle of this image, to the purple-masked copper fill.

The thermal relief allows for sufficient electrical connection between the pad and the copper fill, while at the same time restricting the amount of thermal energy that is dissipated from the pad to the copper fill. This is important for achieving good soldering. If the pad was connected directly to the copper fill, then too much thermal energy would

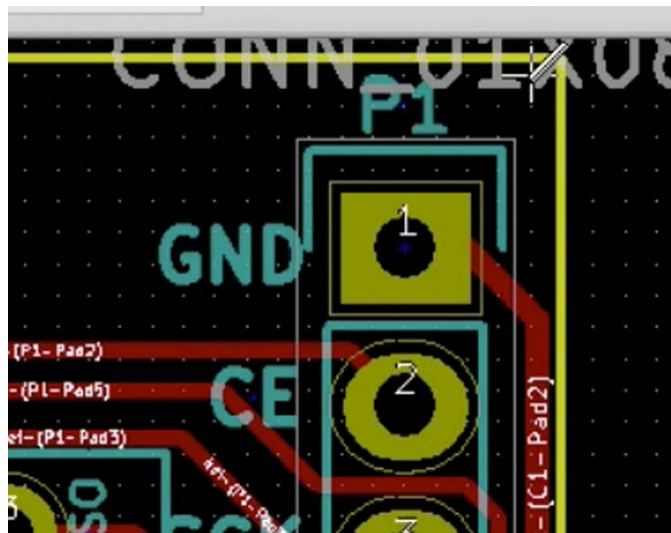
dissipate making it hard to keep a high enough temperature in the solder. This would result to a failed soldered joint between the pad and the component pin.

You can also create a copper fill that is not grounded. Instead, you can attached to a VCC pad. In 2-sided PCB, the typical configuration is to create a ground copper plane in the bottom layer, and a Vcc copper plane in the top layer. In the circuit for the current project we will create a ground plane and cover as much of the top copper area as we can. In the next two projects we'll be creating a Vcc plane on the top layer and a ground plane in the bottom area.

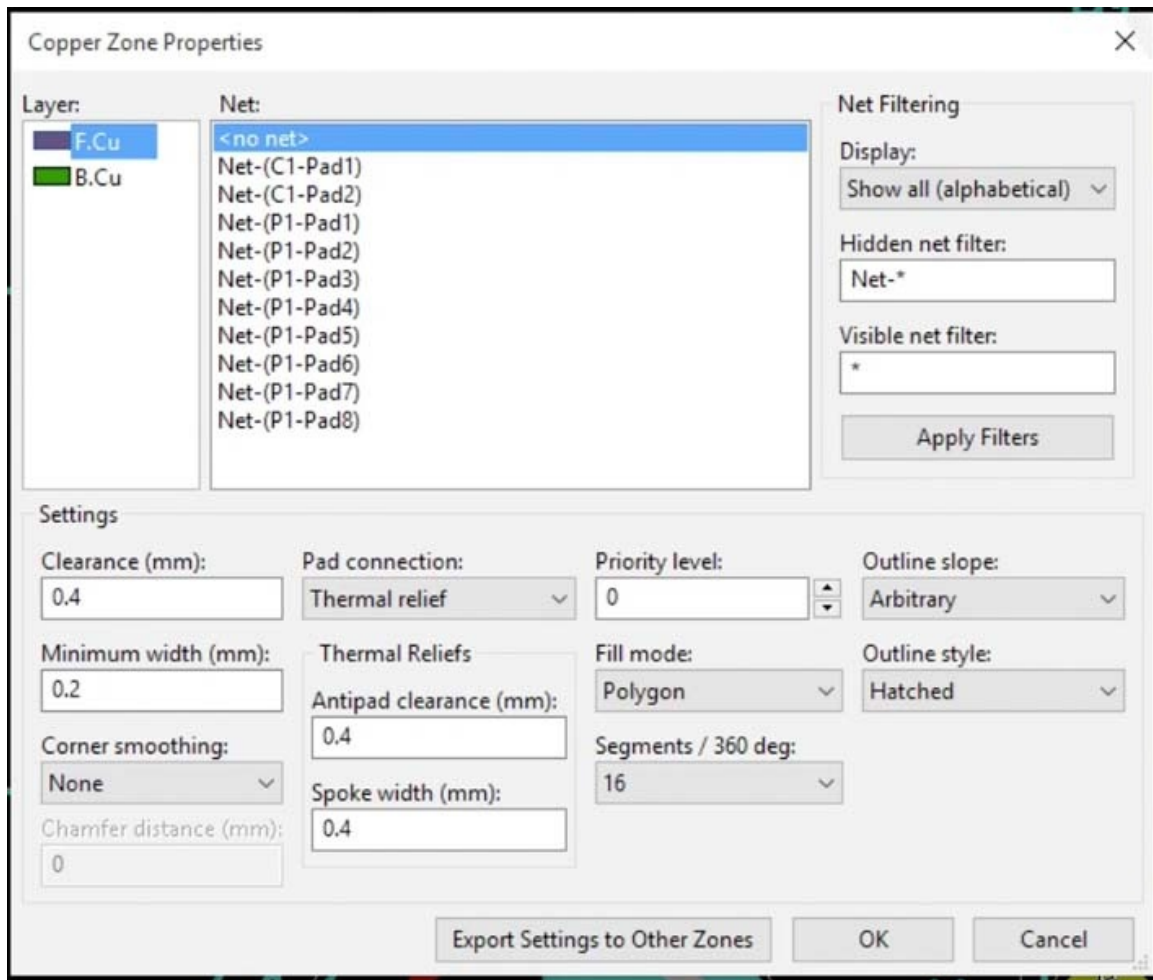


Start by clicking on the Add Field Zones button.

Let's get started. To create a copper fill start by clicking on the Add Field Zones button. This will change your cursor into a cross. You can then start creating the new zone. Make sure that the top copper layer is selected (F.Cu), and click on the top right corner of the PCB.

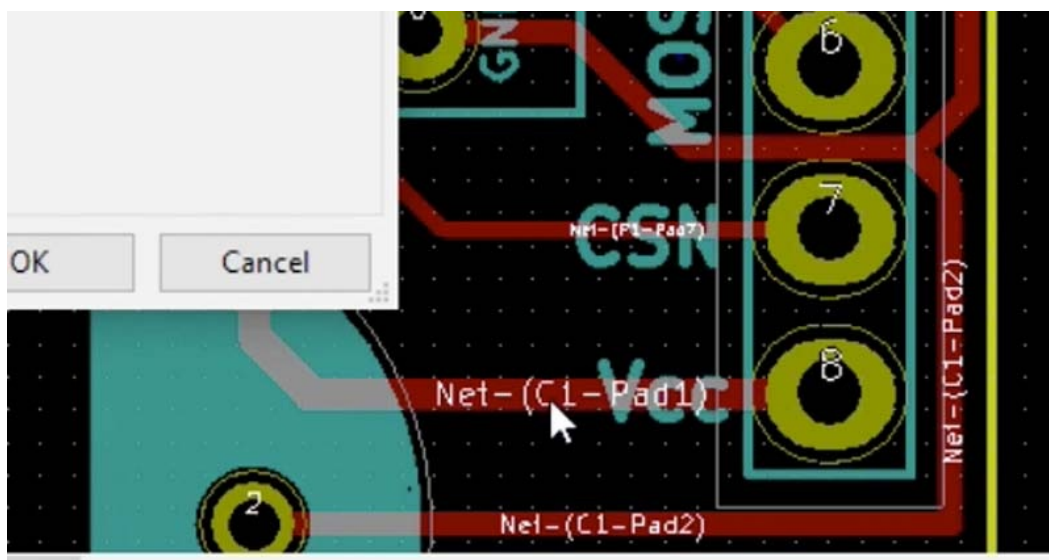


Working on F.Cu, click on the top right corner of the PCB to start creating the new zone.



The first zone click will bring up the Copper Zone Properties window.

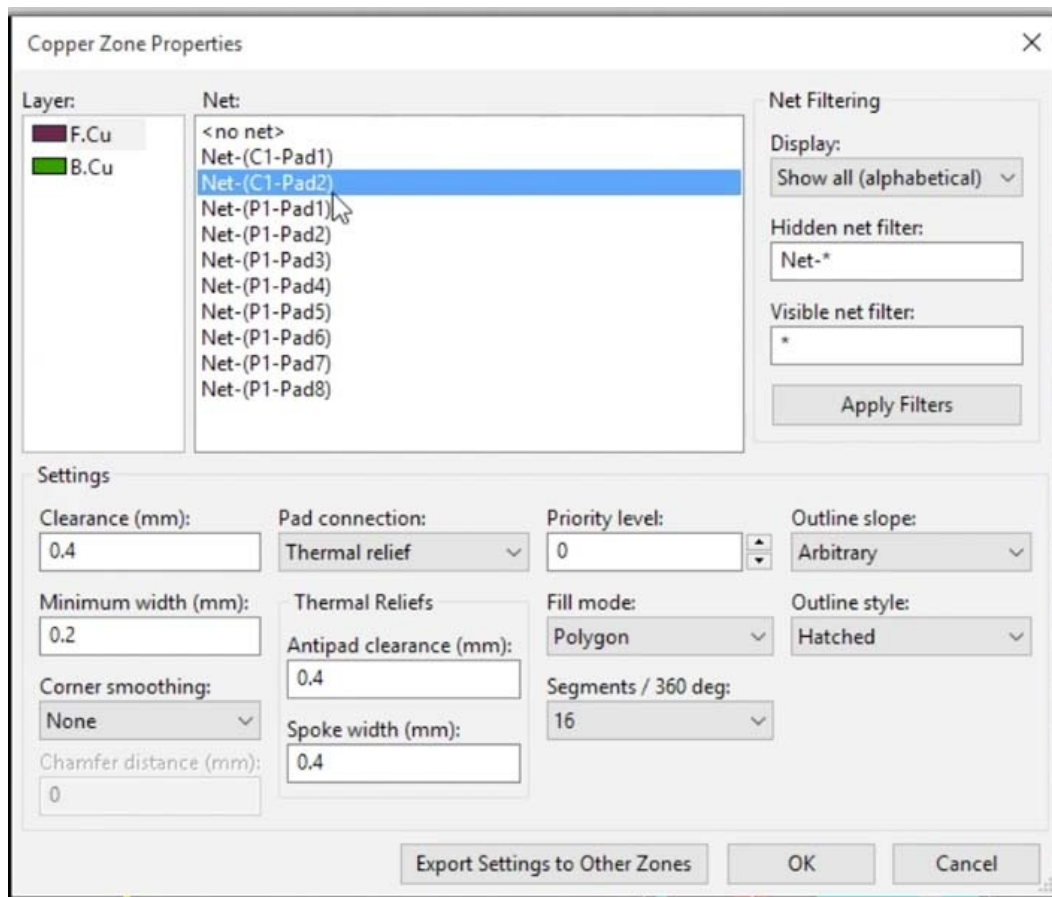
The first click will bring up the Copper Zone Properties window. The two important settings to make here are the layer for the fill, and the net you would like the zone to be connected to. In this project, we are working on the top layer only, so for the layer, choose "F.Cu". Next, in the Net box, there are several nets listed. Which one is the ground net?



Each trace has the name of the net to which it belongs written on it.

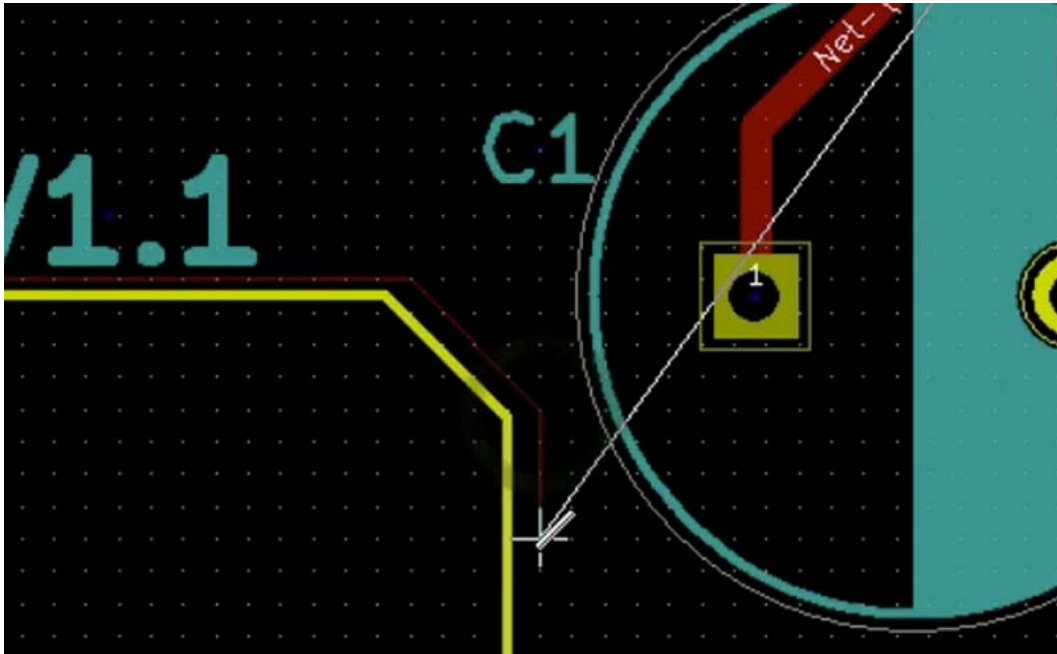
In the next project, I will show you how to give nets a custom, easy to recognise name. In this project, the nets were automatically named by KiCad. To figure out which of the

listed nets is the ground net, have a look at the traces. Zoom in if necessary. In the screenshot above, you can see that the trace connected to the Vcc pad has the text “Net - (C1 - Pad1)” on it. This is the name of the net to which this trace belongs to. Also look at the trace that is connected to the GND pads. The name of that net is “Net - (C1-Pad2)”. This way, you can determine the name of the net to which a wire belongs to.

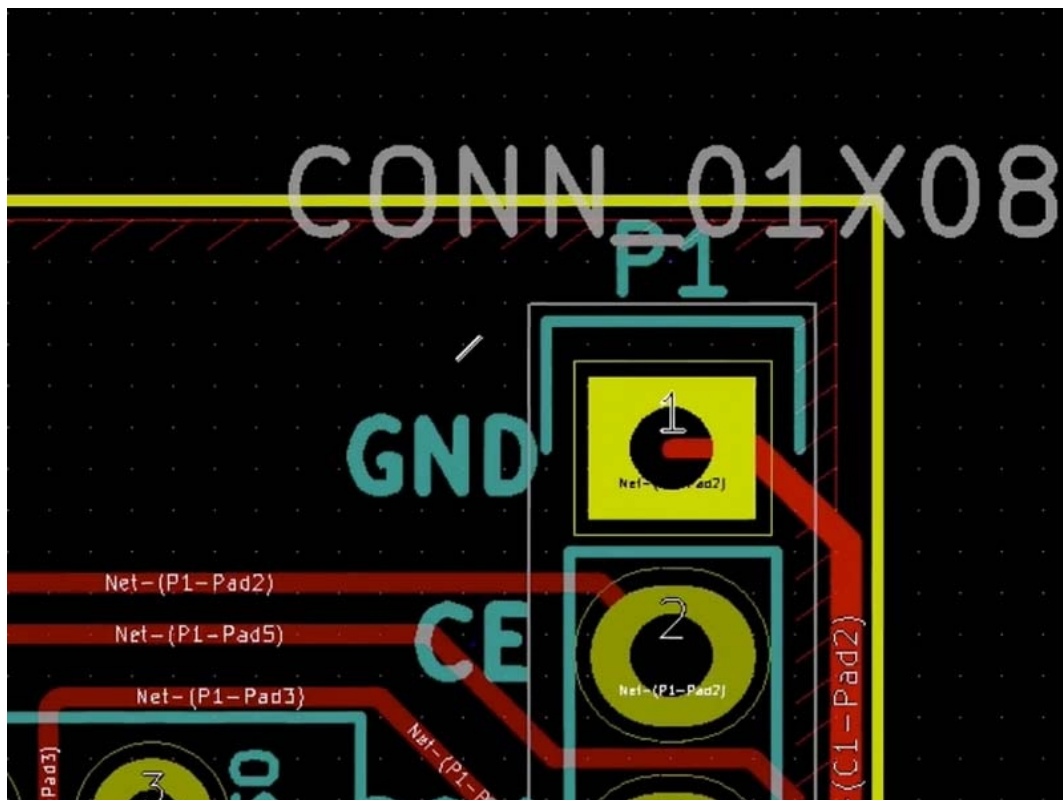


The Net-(C1-Pad2) is the ground net.

Now that you know that, you can go back to the Copper Zone Properties window and click on the “Net-(C1-Pad2)” net to select it. Then, click OK to close the Properties window. You can now continue drawing the new zone. The first edge of the zone is already selected, so move the cursor to the top left corner of the PCB to select the second edge, and click to commit. Continue around the edges of the PCB, like this:



Draw the zone edge along the edge of the PCB. Click to create an edge.

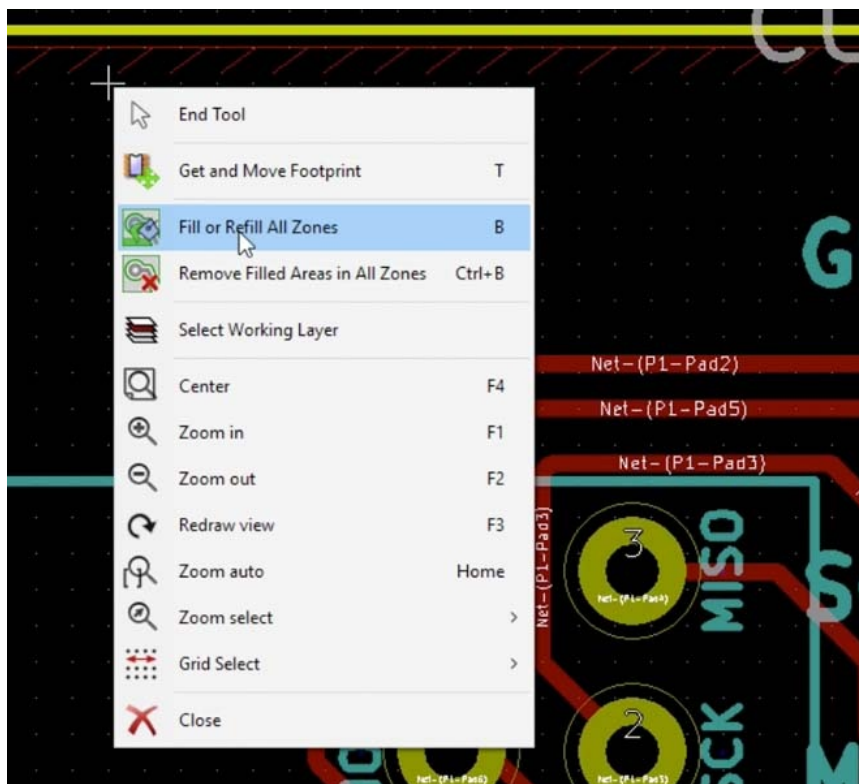


To complete the zone, double-click. Notice that the zone is marked by short red hairlines.

Draw the edge of the zone as close as you can to the edge of the PCB. Go around the edge of the PCB, clicking to create an edge. To close the zone, double click on the point where you started from, at the top right of the PCB. If all this is done correctly, the zone

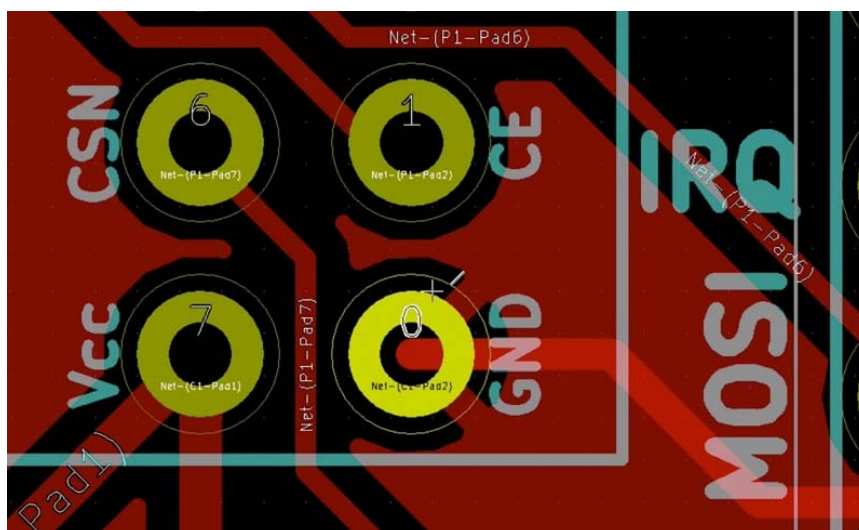
will be created. You will see the red outline, and the small red hairlines pointing inwards. They are red because we are working on the top copper layer, which is designated by the red colour. If we were working on the bottom layer, the traces and the outline of the zones would be coloured green.

The next thing to do is to fill this area with copper.



Right-click inside the zone, and select Fill or Refill All Zone.

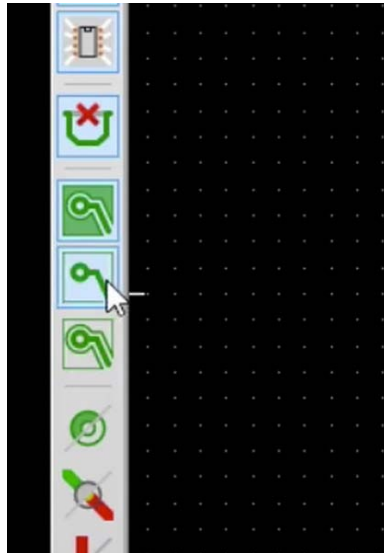
Right click anywhere inside the zone and then choose “fill or refill all zones”.



The zone is now filled.

The red area indicates that copper is filled where is possible. You can also notice the thermal reliefs in the ground pad.

Save the project.



This button allows you to toggle copper fills on and off.

If you would like to have a look at your board without a copper fills showing you can just make them invisible, so you can click on the “Make zone fills invisible” button. You can make the zone fill visible again by, clicking on the “Make zone fills visible” button.

Let’s do another ERC to make sure everything is ok.

In the next section, I will show you how to export the Gerber files upload the PCB to the manufacturer.

PART FIVE
Project 1: Fabrication

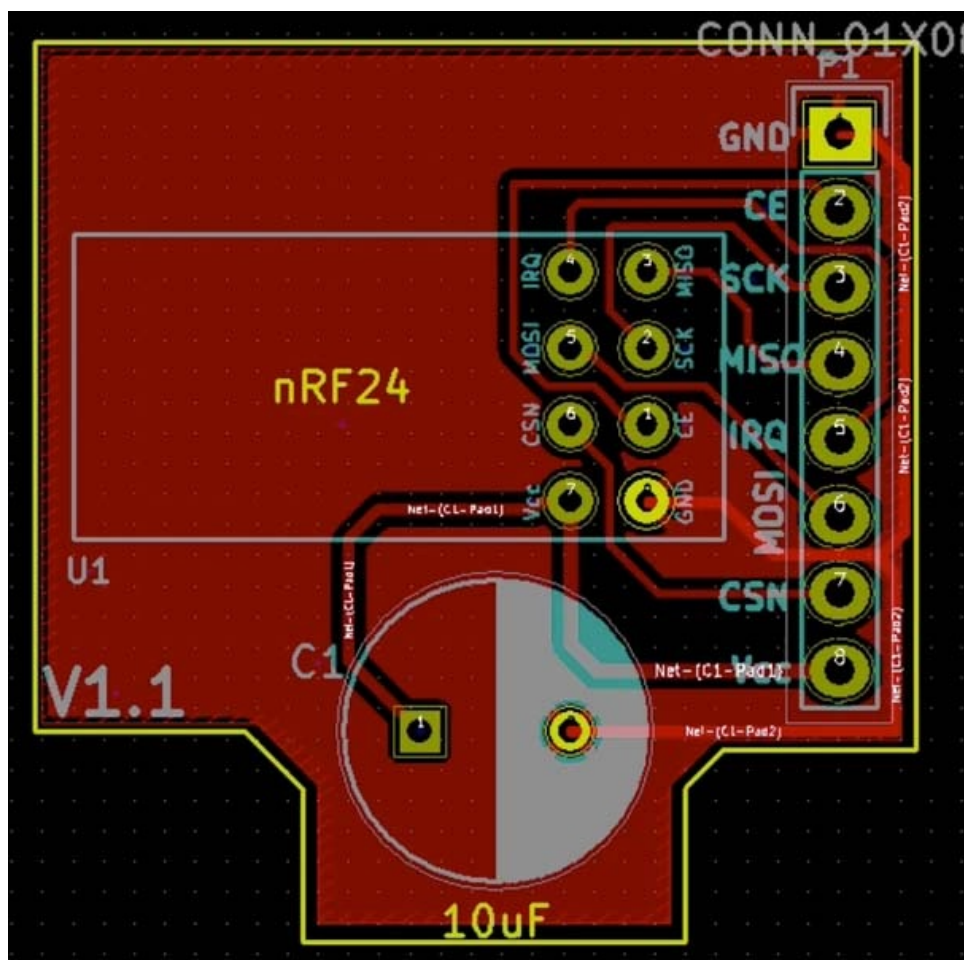
Chapter 33: *What is this part*

At this point we have completed the design of the nEF24 breakout and we are ready to send it off to a fabrication lab.

In this section, I will show you the process of doing this with OSHPark. The process is almost identical for most other online fabs.

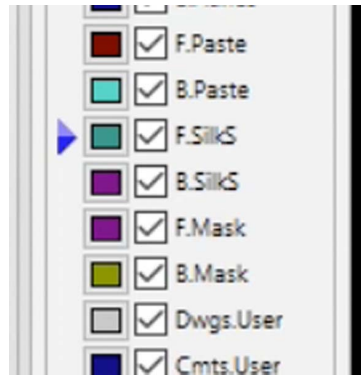
As an aside, in this section I will also show you how to add a decorative graphic to your PCB.

Chapter 34: *Creating the Gerber files and uploading to fabricator*



This board is almost ready to manufacture.

This board is ready to make. Before I upload it to the manufacturer, why not put your name on it, or the name of the board? Switch to the front silk screen and use the text tool to type in the name of the board, perhaps let's call it Peter's nRF24 Breakout. Or you can put your name on it.



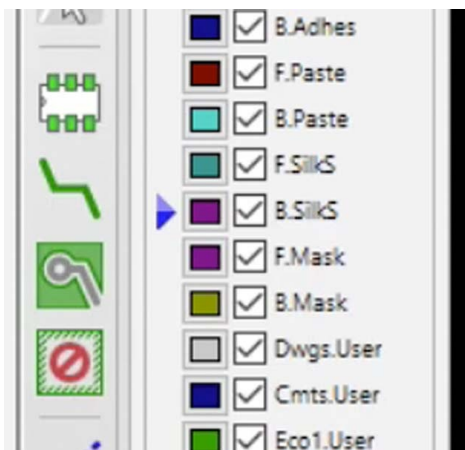
Choose the Front Silkscreen (“F.SilkS”) layer to add some text.

Edit the text and set the height and width to your desired values — maybe 1mm should be enough. You can also use the line tool to draw a box around the text. The end result is this:



The name of the board with a box around it.

You can also put things on the bottom silk screen layer.



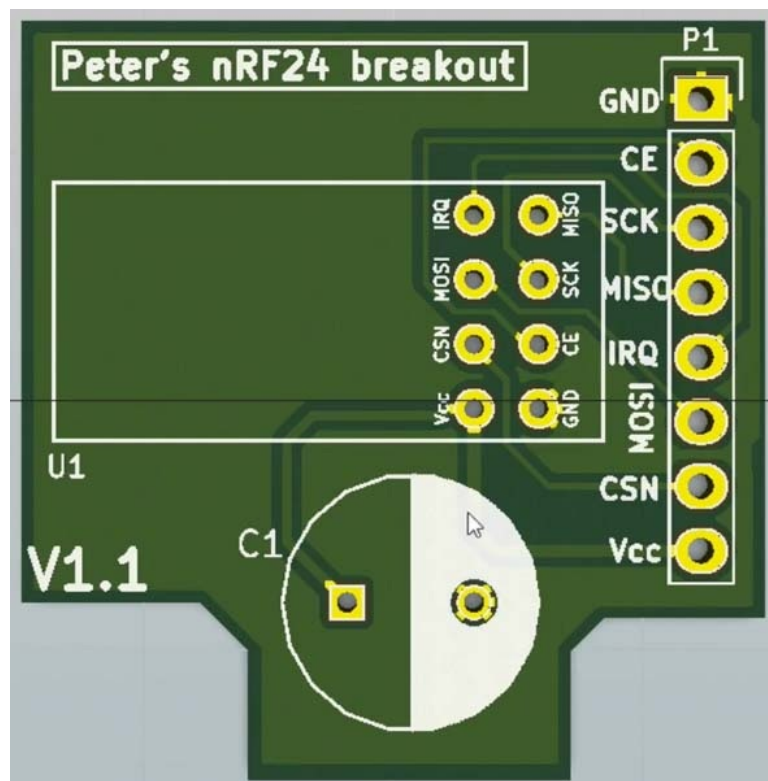
Select the B.SilkS layer to add text to the back layer.

You can print something like “Tech Explorations”, and the date you designed your PCB.

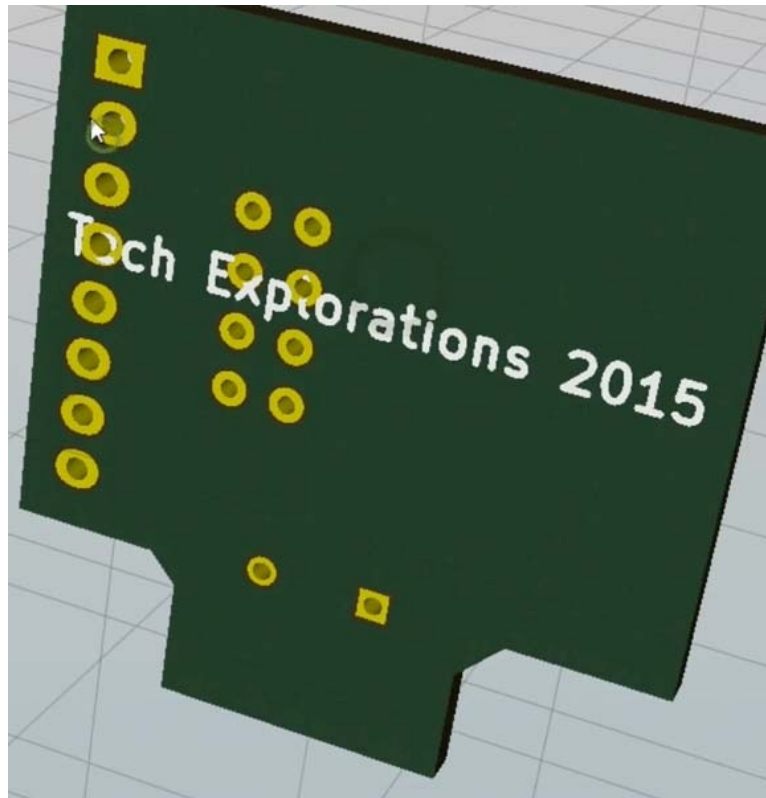


The text in the back silkscreen layer is represented in mirrored purple text.

You can have a look to see what it would look like once it's finished and manufactured using the 3D preview:

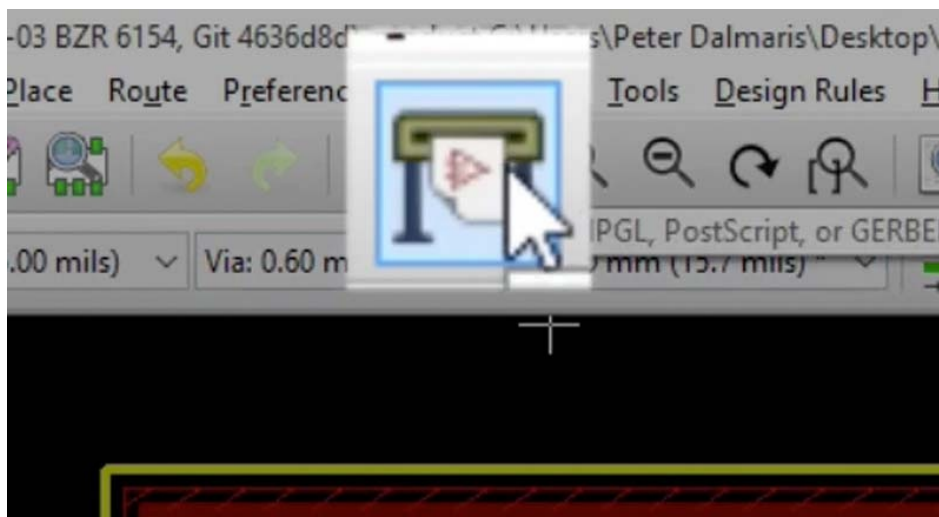


3D view of the front layer.



3D view of the back layer.

You can also put graphics on your board. I will show you how to do that in next chapter. You might want to read that chapter first before actually uploading this PCB to the manufacturer, if you'd like to include graphics with your finished, made board.



Click on the Plotter icon to generate the Gerber files.

And we are now ready to export the Gerber files. To do that, click on the plotter button and make sure that you've got the required layers selected. The required layers are:

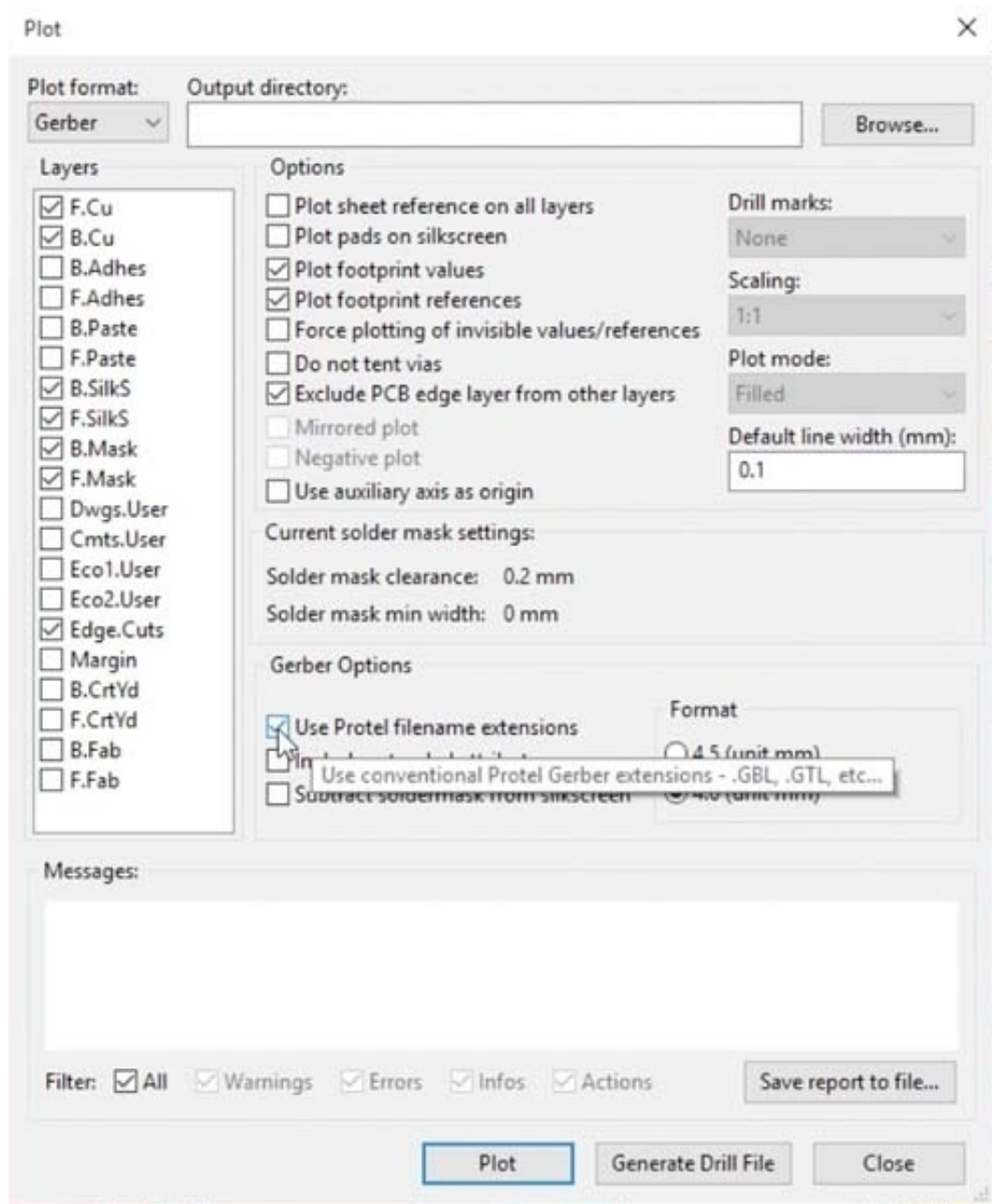
- F.Cu: Front copper
- B.CU: Back copper
- B.SilkS: Back silkscreen

F.SilkS: Front silkscreen

B.Mask: Bottom solder mask

F.Mask: Front solder mask

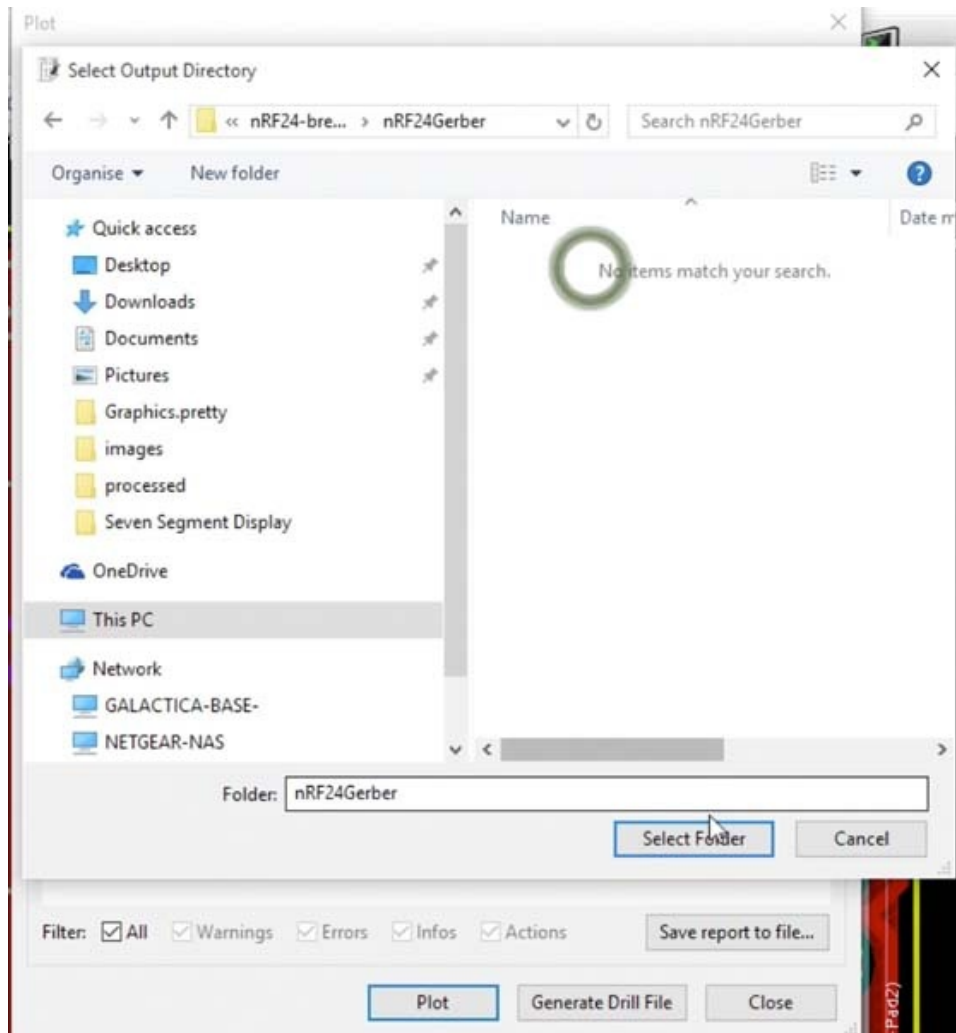
Edge.Cuts: Board outline



This screenshot contains the appropriate settings for manufacturing the project board.

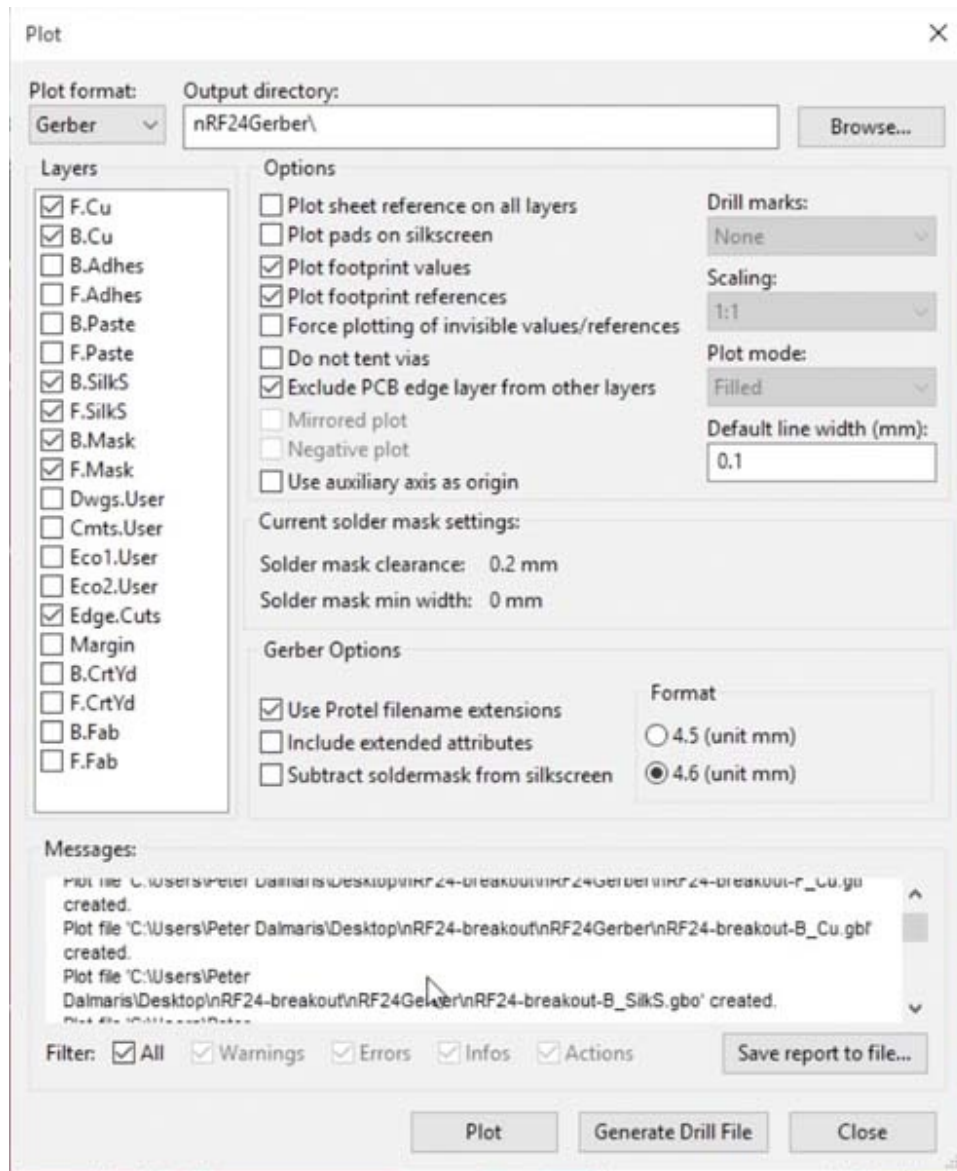
In the Gerber Options group, choose the Protel filename option.

For the output directory, choose a new directory inside your current project directory.



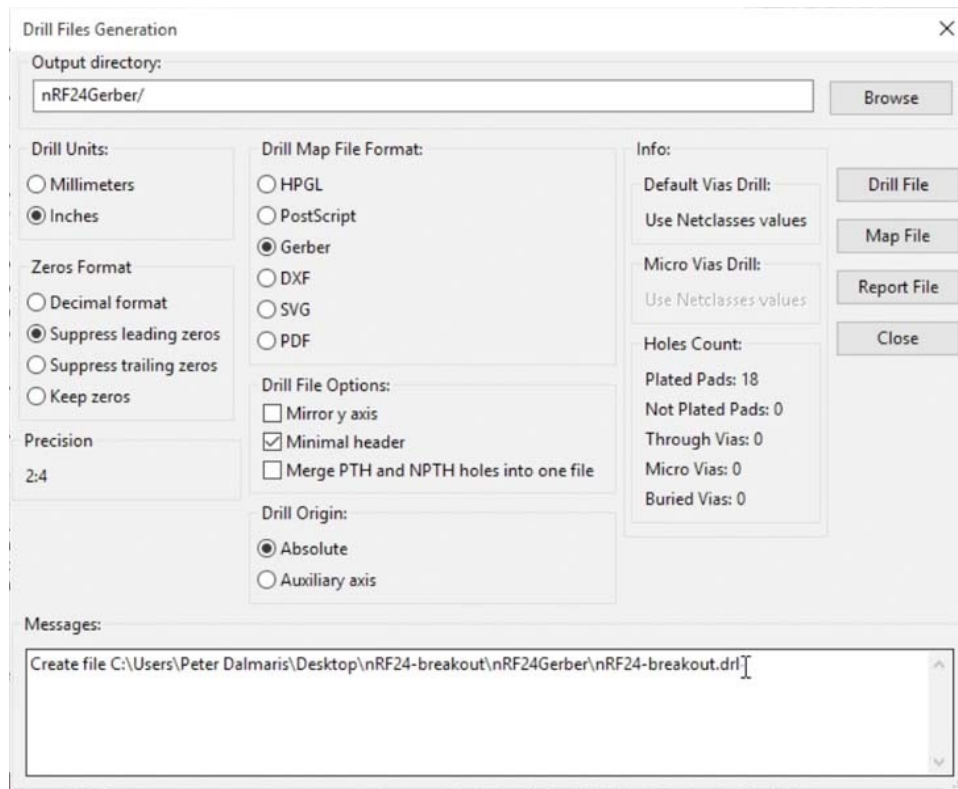
Create a new directory for the Gerber files. Place it inside the current project directory.

Next, click on the plot button and to generate the Gerber files for the layers.



Notice that the output directory is a sub-directory inside the current project directory. After clicking on the Plot button, information of the files that were created will appear in the Messages text box.

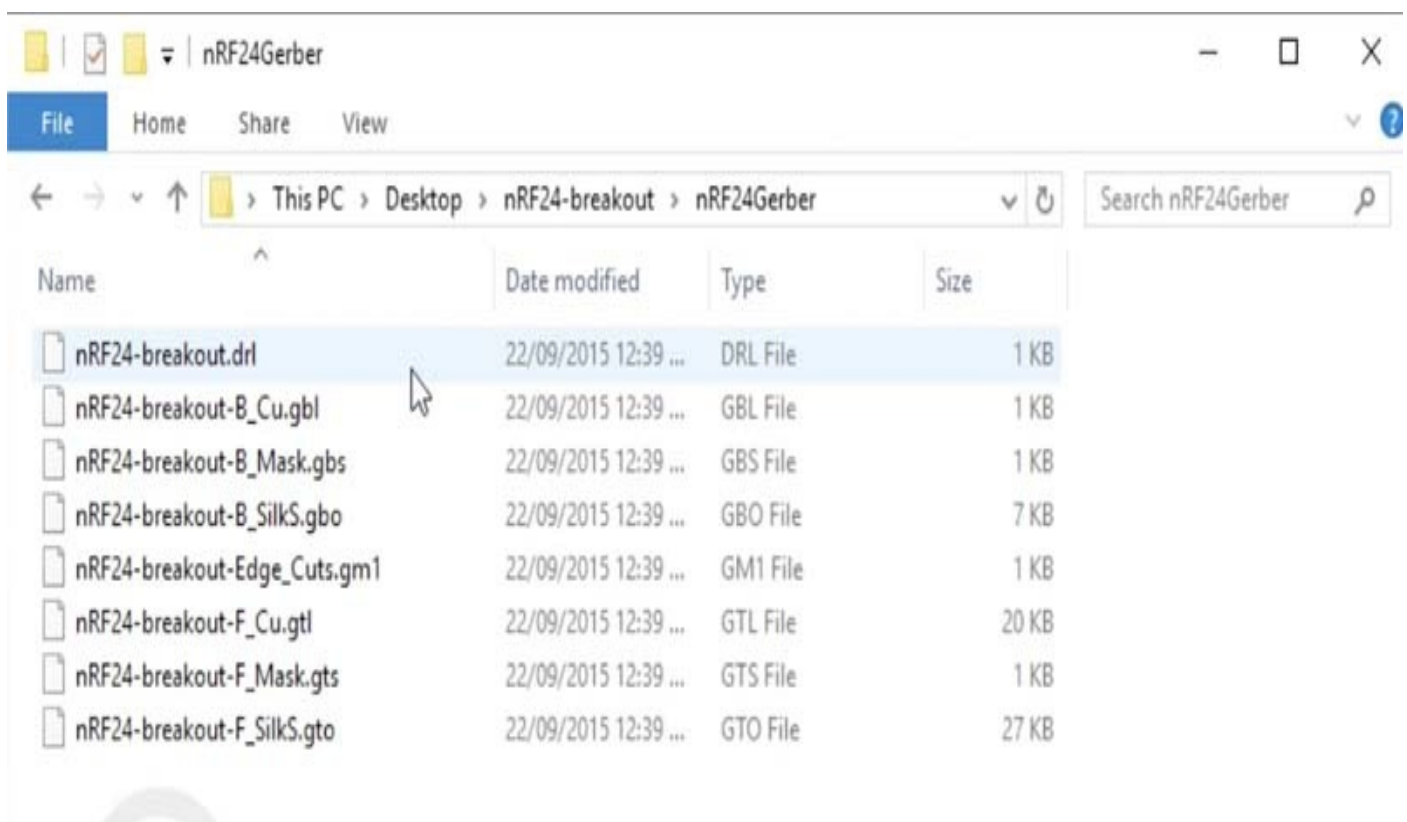
Next, we must also generate the drill file. The drill file tells the manufacturer where to drill for the pad holes on the PCD. Click on the “Generate Drill File” button.



The Drill Files Generation window.

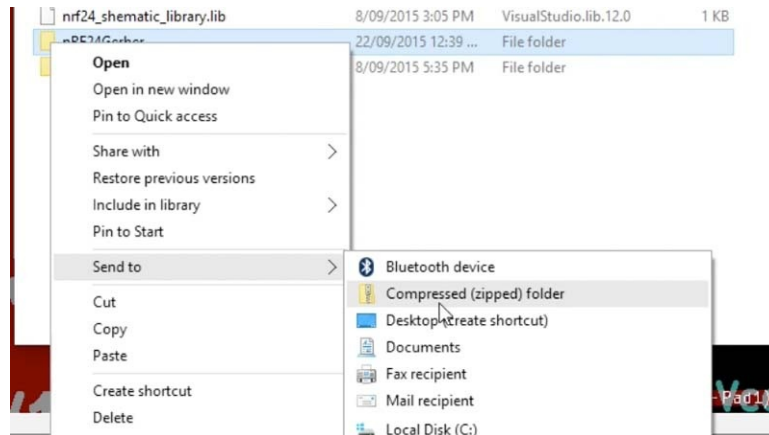
This will bring up another dialog box. All the settings should be correct by default, including the output directory. Click on the Drill File button, and this will generate the drill file with a “.drl” extension. You can close the Drill Files Generation dialog, and then close the Gerber Plot window.

Have a look inside your Gerbers directory.



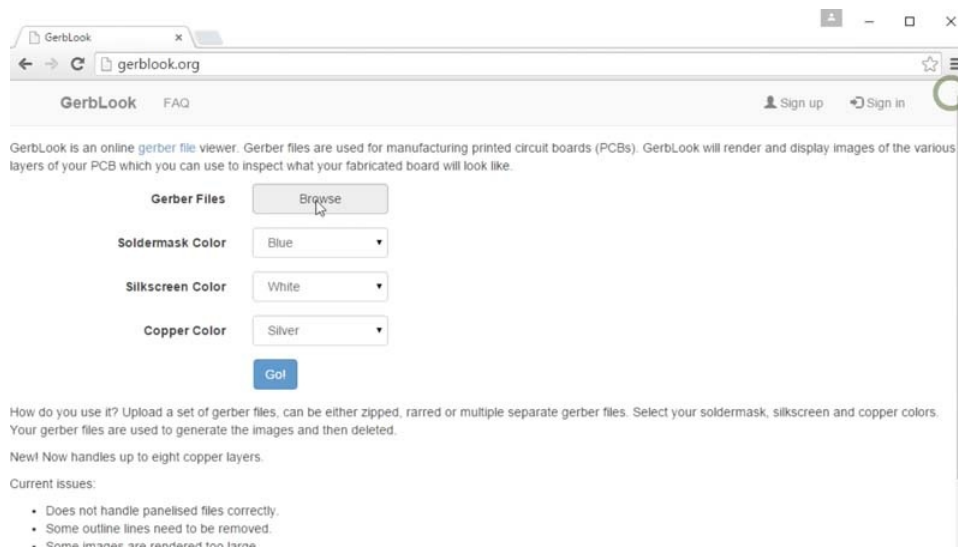
The Gerber files.

Let's inspect what we have. Inside the Gerbers directory you can see all the files that we just generated. We can now upload those files to the manufacturer. Before that though, we should make sure there are no errors. There is a very handy free service that we can use for this purpose, GerbLook.org. To upload our Gerber files to GerbLook, we first must create a Zip archive.



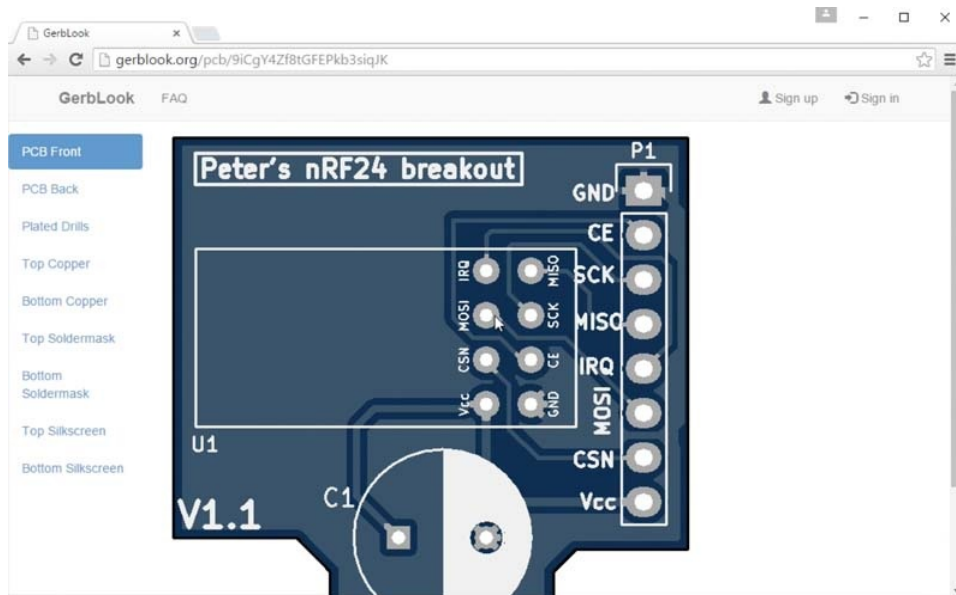
Create a ZIP archive from the Gerber directory.

Use your browser to go to gerblook.org. Then upload your Gerber ZIP file to the service.



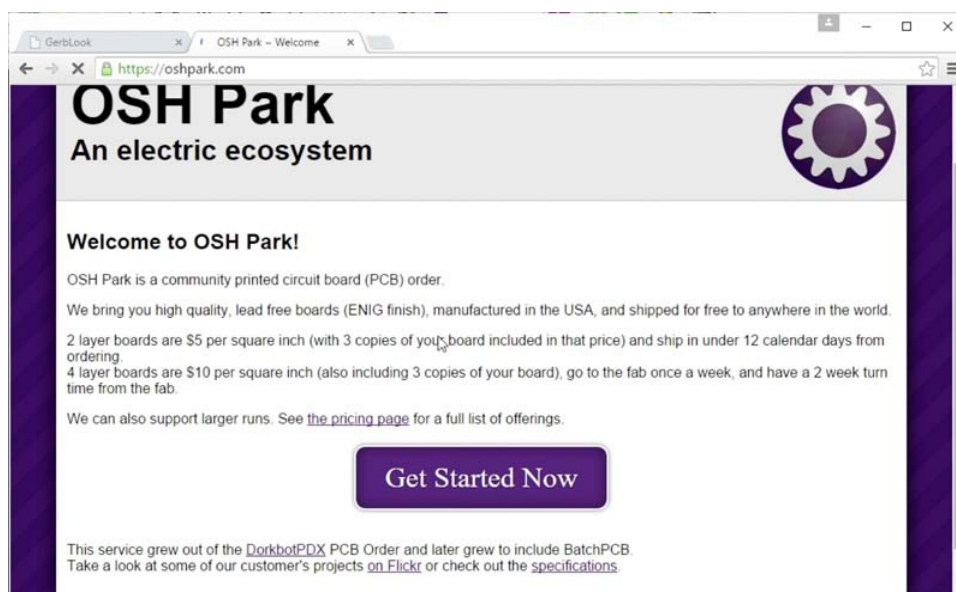
Go to gerblook.org and upload your Gerber ZIP archive.

Gerblook will inspect your Gerber files and let you know if there are any errors, like a missing layer or other problems. If everything is ok, you will see a rendering of each layer, like this:

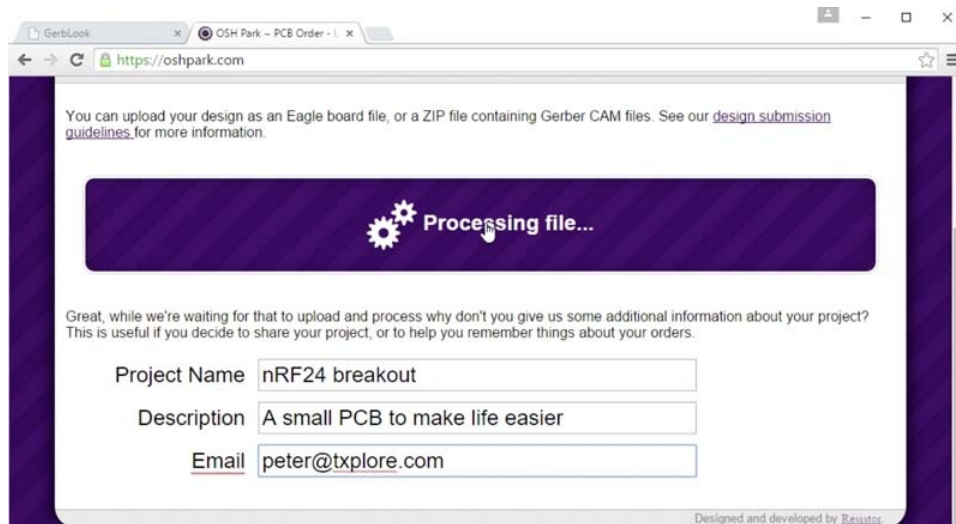


If everything goes well, Gerblook will show you a render of each layer.

Now, I am confident to go to OSHPark and start my ordering process. Go to oshpark.com.



The OSH Park home page. Click on Get Started Now.



Select the ZIP archive that you created earlier and start the upload. While the file is uploading and processing, you can fill in the project information.

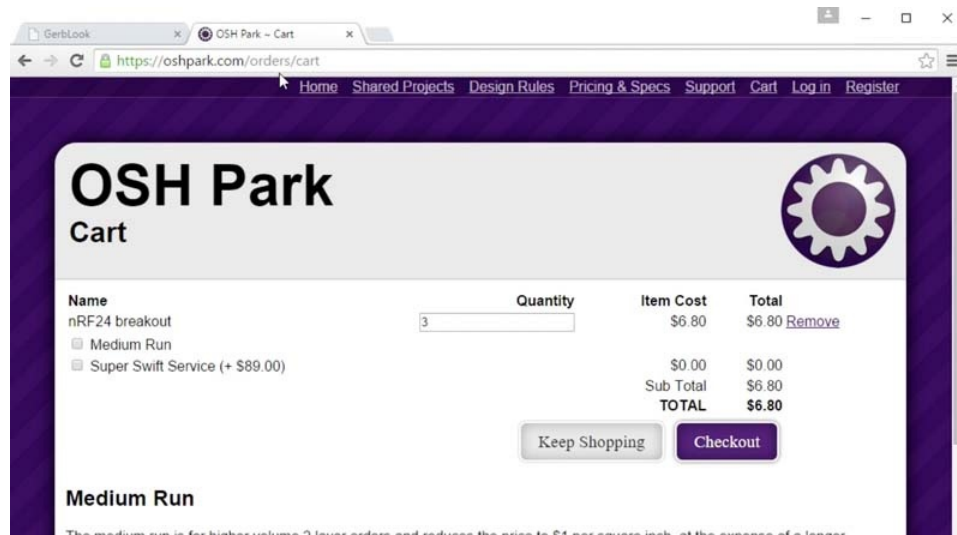
If everything goes well, once the processing finished, OSH Park will show you the layers that make up your board.



On the left you can see the OSH Park preview representation of the board. On the right, the actual board.

Okay, so it's finished uploading and processing. We can see the front and back of my new PCB.

If you are happy with what you see, you can proceed with the order.



At the timing of writing, OSH Park’s pricing is among the most competitive in the industry.

At the time of writing this book, the cost for this board was US\$6.80, a competitive price.

In the next chapter, I’ll show you how to add a custom graphic in order to decorate your PCB using the silk screen.

Chapter 35: *Adding a decorative graphic*

RECONSIDER IF YOU WANT TO INCLUDE THIS HERE, OR JUST LEAVE IT FOR PROJECT 2, WHERE IT IS DEMONSTRATED AT A REASONABLE LOCATION

In this chapter, I'll show you how to decorate your PCB with graphics on either the top or the bottom silkscreen layer. Decorating a PCB is usually the last thing you want to do before you export the Gerber files and send them to manufacturing. The decorative graphic could be a logo or some other symbol that you would like to show on your PCB. You can place the graphic on the top or the bottom silkscreen layer. For this project, I will take my logo graphic, convert it into a footprint (that has no pads) and then place it onto the PCB.

The way that KiCad works with graphics is by using a utility that converts a graphical file into a footprint. The utility produces a footprint without any pins or any holes or vias or any other elements other than the contents of the silkscreen, which you can place on the PCB.

The process starts with the image. The tool that Kicad provides to manipulate images is minimal in terms of functionality.

Here's my logo, an image in PNG format:



We will start with this PNG image.

It is around 2000 pixels in width, which is too large to fit on this PCB without resizing.

Step number one is to resize the image as big as your PCB is going to be. I'm going to go back into PCB new and do a quick measurement. I'm going to put my cursor up in the top corner, top left corner, hit the space bar to zero the dimensions and then move across to the right corner. You can see that in the X horizontal axis I've got 29.08 millimeters, so 29.08 millimeters. If I go down I have 22 millimeters. You should write these values down. That is 22.7 millimeters in the Y axis and 29.21 in the X axis. I can close that now.

The tool I'll be using to do the conversion of the png image into a footprint is Bitmap2 component. Let's open up bitmap2 component and go to the black and white picture tab and load the bitmap. You should go into desktop, project folder, this is a latest image. Looks a bit big, it was 72dpi, so I can change that to let's say 300. I'm changing the dpi, you can see that the size updates. This is a good way to control how big your image eventually will look on the PCB. I know that the dimensions of my PCB in width is 29.21 millimeters, so looks like this. It is still a bit too wide, so I will increase that maybe to 400, 500, I will increase them to—I'm going to go down to below 29 millimeters, so make that a 1000. No, a bit too much, maybe 800. Yes, 850, okay.

You need to keep track of your other side as well, 850. I think these settings at 850 dots per inch will produce an image, a footprint that is 22.4 millimeters in width and 14.9 millimeters in height. That should comfortably fit inside my PCB. Next I want to leave this as normal. Normal will produce a graphic with a wide silkscreen ink on the PCB instead of having a wide silkscreen ink everywhere outside the graphic itself. What I want is white on green or purple for Oshpack instead of purple on white. I hope that makes sense but you can see what happens if you choose negative.

Now, much that you have received the PCB from Oshpack, the black ink here will actually be purple since the PCB color from Oshpack is purple and then everything else around it is going to be white. I would rather go with the white marking for my logo. I will just leave it like that and everything else around it is going to be a default purple for the PCB, so the mask. You can play around with the threshold value but I find that about 80% is fine, so this basically tells you which part is white and which part is black. I've got black and white, anything above that reverses the colors. I will leave it at about 80%.

The format is PCB new since this is my target tool. All right, so I will export this and I will save the new graphic as a footprint in a new library inside my project folder. I go back to my project folder and then in here I'd like to create a new folder. I will call it something like graphics.pretty. Remember that the pretty extension is important because it denotes a library for footprints. Make sure you've got .pretty in here. Drill inside your new library folder and give your new footprint a name. Let's call it T-explore logo and save.

We're done with the tool. We got a new library for this footprint. We can now go into the PCBnew app. The first thing to do is to import the new library, just like we've done in the past with the custom component footprint. We go into preferences, then I use the footprint libraries manager and I will use the, append with wizard option to drill and find my new footprint library. That's in here somewhere. There you go, graphics.pretty is what

I'm looking for.

Next, it's looked inside. It's found that it contains footprints that are good. I'm going to add this to the current project only. I didn't add it to the global projects, globally accessible because the way that I have customized the dimensions for this graphic really makes sense only for this current PCB. I'm going to click okay to finish.

Now, I need to add this new footprint. I can click on this button here and I will just put the new footprint here. Now I will see if hopefully it's in my library folder because sometimes actually you have to restart PCBnew in order for it to load the library contents you've just added. Let's see if it's somewhere here. You should probably search for logo. There is T-explore logo, good. Okay and here it is.

Here is a logo that I've just added. I'm just going to put the logo here and what I want to do is to edit its location. I want to put this in the bottom side. Get the properties for the logo and choose site from top to bottom and use an M key to move it in place. Around here would be okay. I can also remove the text that I added earlier since I no longer need it. Just remove that, gone. All right. Refresh.

I'll actually move my logo a little bit higher, so around here and that looks okay. Let's check it out in the 3D view. There it is. There's the nice new graphic. Actually I should probably flip it, so edit these properties again and I want to rotate it by 180 degrees. I hope that would do the trick. Check out 3D viewer, just want to ease the orientation to be appropriate. This is the top, that the bottom. The orientation now is good, top and bottom. Nice. Don't forget to save your project. Let's refresh the drawings and there you have it. You can now decorate your PCBs with nice graphics. You can now take this PCB if you like and upload it to Oshpack and manufacture it and your graphic will be honored.

[00:11:10] [END OF AUDIO]

PART SIX

Project 2: a 7-segment display board

Chapter 36: *What is this part*

In the second project, we will design a PCB for a seven segment display circuit. The circuit is a slightly modified one from the Arduino Step by Step course, from the lecture on the seven segment display.

If you haven't watched that lecture, don't worry, it won't affect your ability to follow this project here.

We will start with a wiring diagram and convert it into a Kicad schematic. We will use components that already are available in the schematic library.

We will then convert the schematic into a 2-layer PCB.

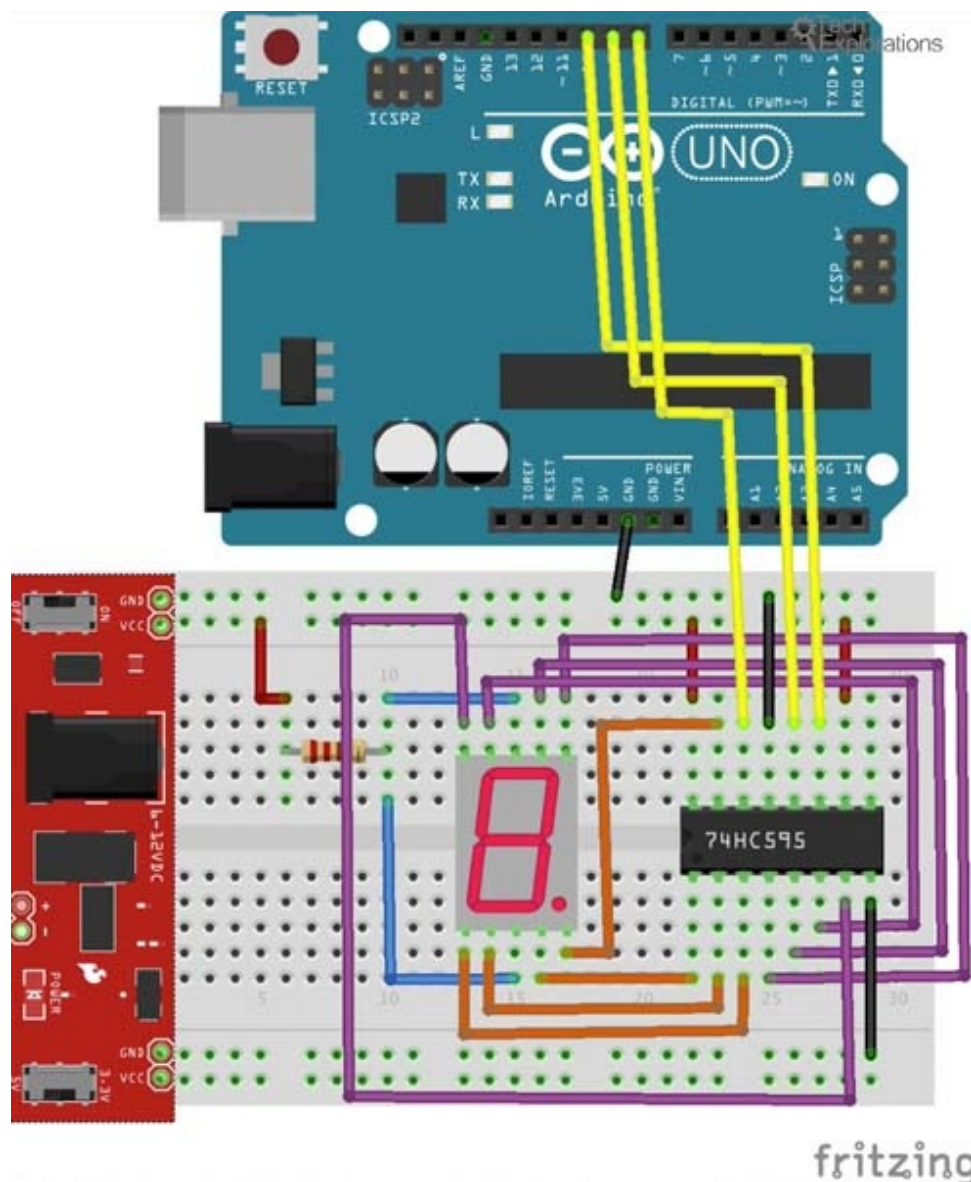
During this process, you will learn how to use several new Kicad features:

- * Configuring components with their values,
- * Working with integrated circuits and their hidden pins
- * All about nets and net labels,
- * The Power Flag,
- * Using busses to simplify connections between components,
- * The Unconnected flag,
- * Controlling the track width with Nets,
- * Vias,
- * and, Adding a decorative graphic

These, alongside what you learned in the first project, represent eighty to ninety percent of the skills and features you will need for most of your PCB work.

Chapter 37: *Create the schematic with Eeschema*

In this chapter, we will start the process of converting an Arduino circuit to a PCB.

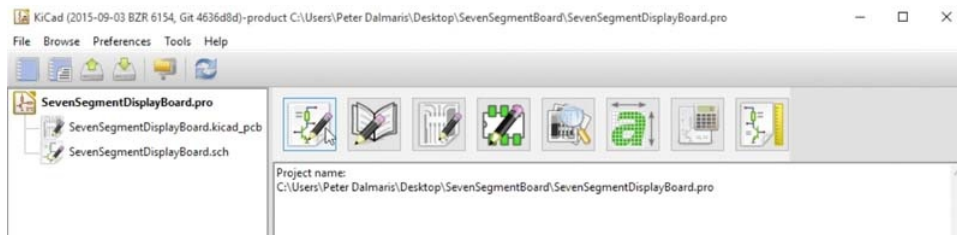


We will convert this Arduino circuit into a PCB

The wiring diagram that you can see above shows a breadboard with a seven-segment display, a shift register integrated circuit, a current limiting resistor, a breadboard power supply, and the Arduino.

We will design the PCB to contain the components on the breadboard, except for the power supply. Instead of a single current limiting resistor, our PCB will contain 8, one for each segment. The connection between the PCB and the Arduino will be made with a simple straight pin connector, like the one we used in project 1.

Start Kicad and create a new project.



Create a new project in Kicad, call it something like “SevenSegmentDisplayBoard”.

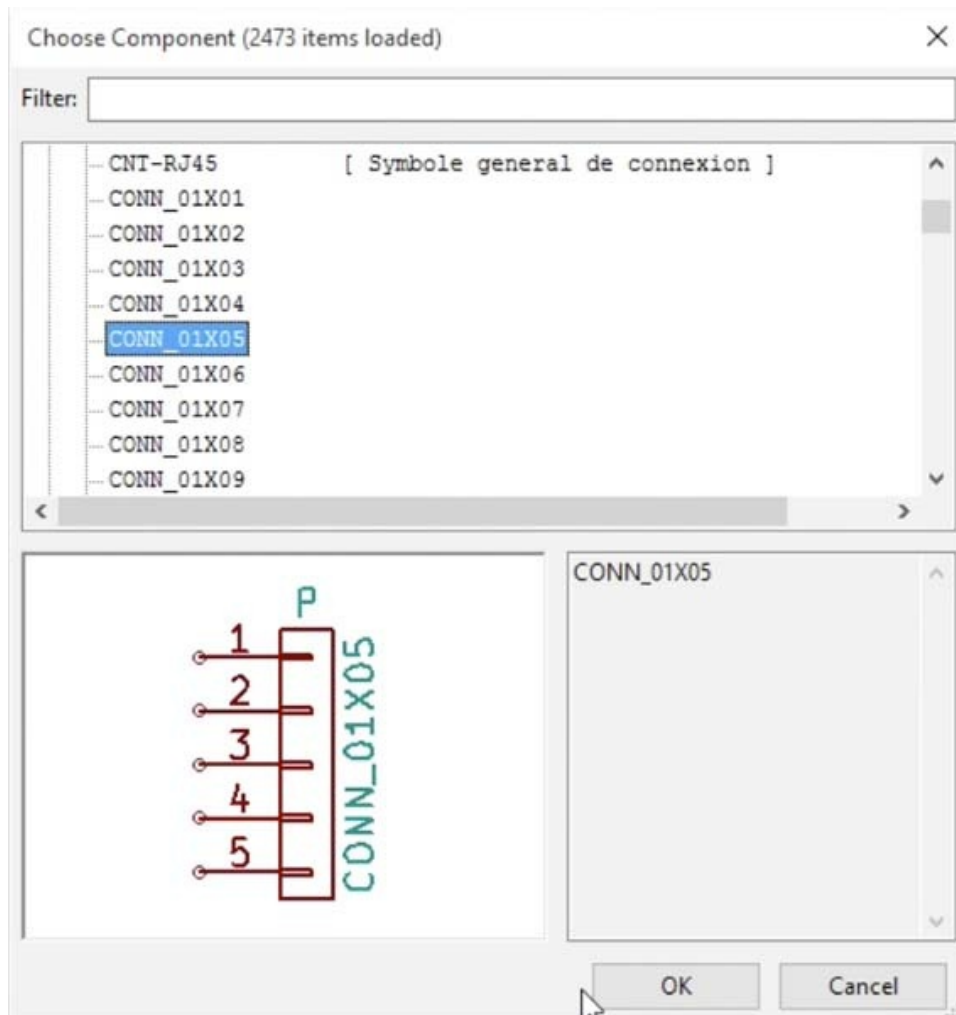
Let’s begin with Eeschema. It is a good habit to enter the schematic information. This is the text that appears in the legend of the schematic, in the bottom right corner of the canvas. To do that, go to File and choose Page Settings. In the end, the schematic legend should look like this:

Peter Dalmaris		
Tech Explorations		
Sheet: /		
File: SevenSegmentDisplayBoard.sch		
Title: Seven Segment Display board		
Size: A4	Date: 2015-09-23	Rev: 1.0
KiCad E.D.A. kicad (2015-09-03 BZR 6154, Git 4636d8d)-product		Id: 1/1

The schematic sheet legen, populated

Let’s continue with adding the components now. From the wiring schematic you can see that we need to add one seven segment display, a 595 shift register and eight current limiting resistors plus the connector.

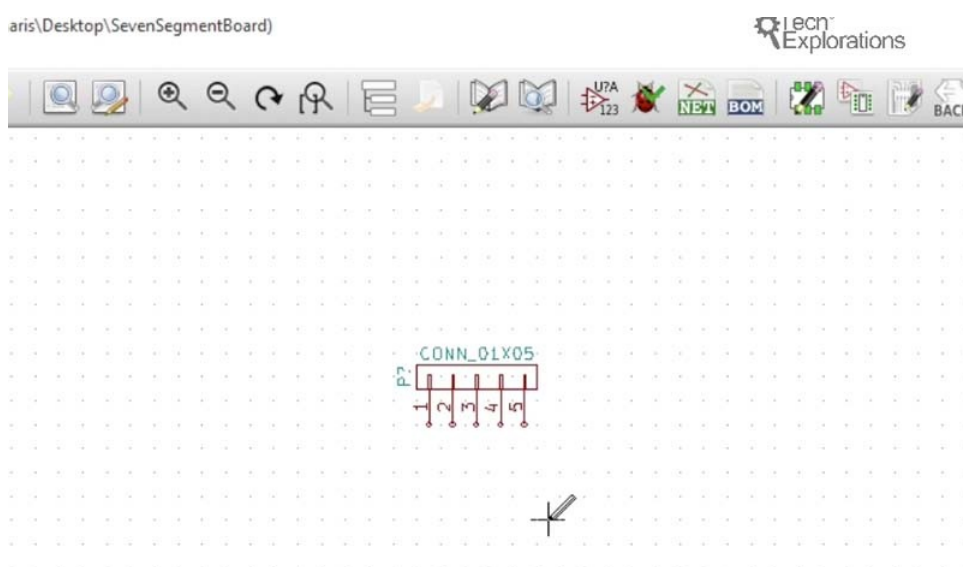
Let’s start with the connector, since it is the simplest component. I’m looking for a straight connector, one row and five pins. As per the wiring schematic, we need one pin for data, one for clock and one for latch. We also need one pin for ground and one pin for Vcc power. Therefore we will need a one row by five pins connector.



For the connector, we will use the 1x5 straight type.

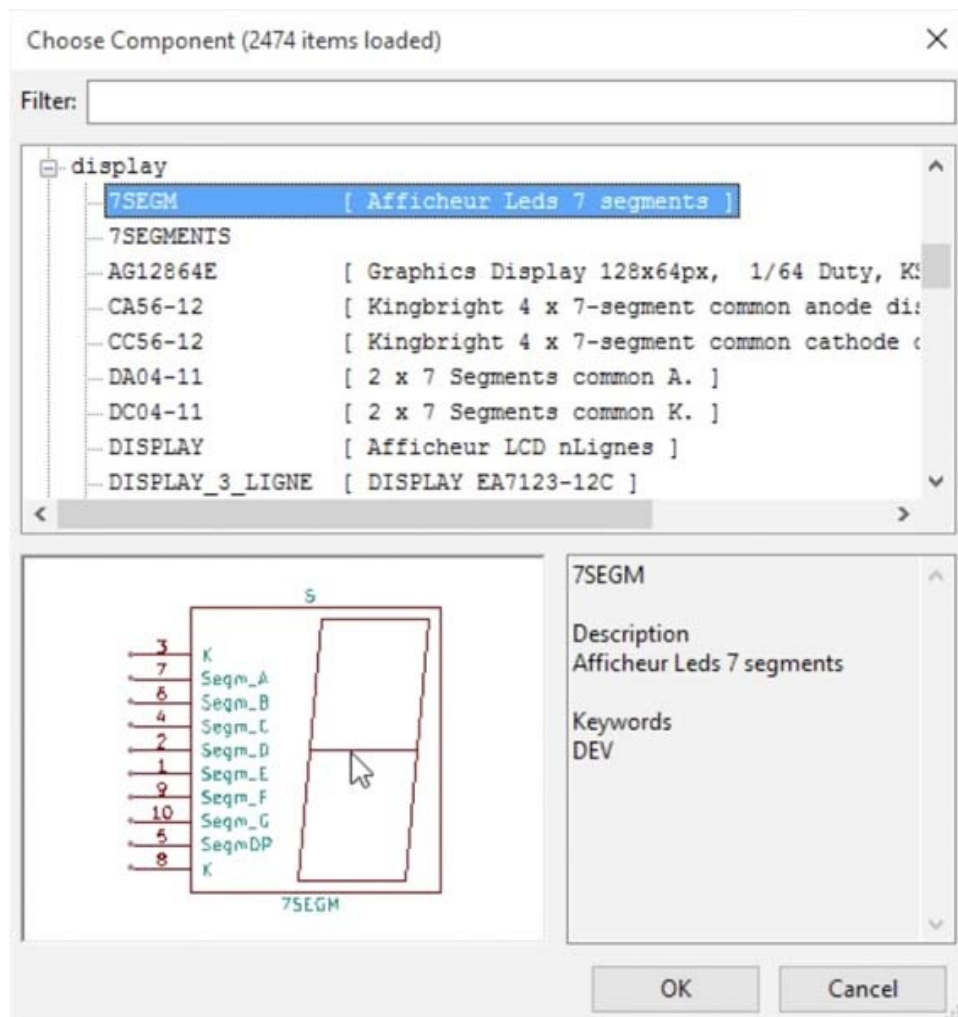
Hit the “A” to bring up the connector chooser window. You can browse to the CONN library and pick the connector you need, or type “CONN_01x05” in the filter. Once you find it, double-click to select it, and place it on the canvas.

You may rotate it and move it on the canvas, like this:



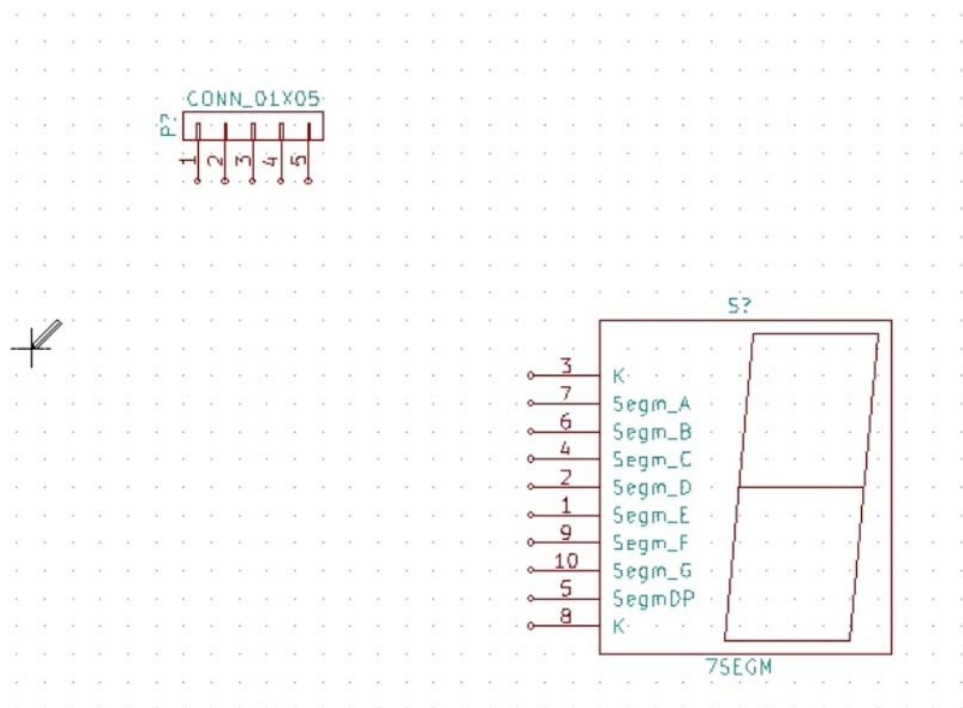
The first component of the board, the 1x5 straight connector.

Then, we need a seven segment display, so hit A and browse for it in the “display” library.



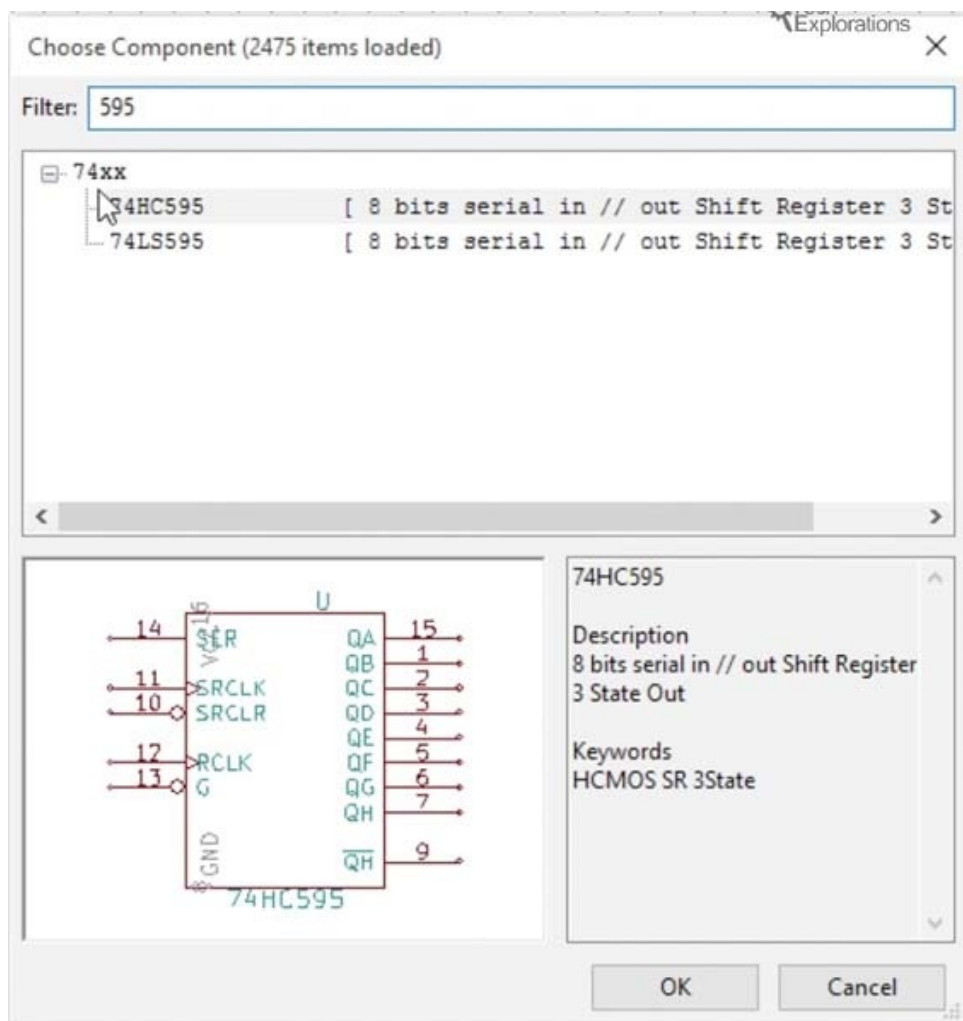
You will find the seven segment display component in the “display” library. Notice that there are a couple of choices available.

Double-click to select the 7SEGM component, and place it on the canvas, like this:



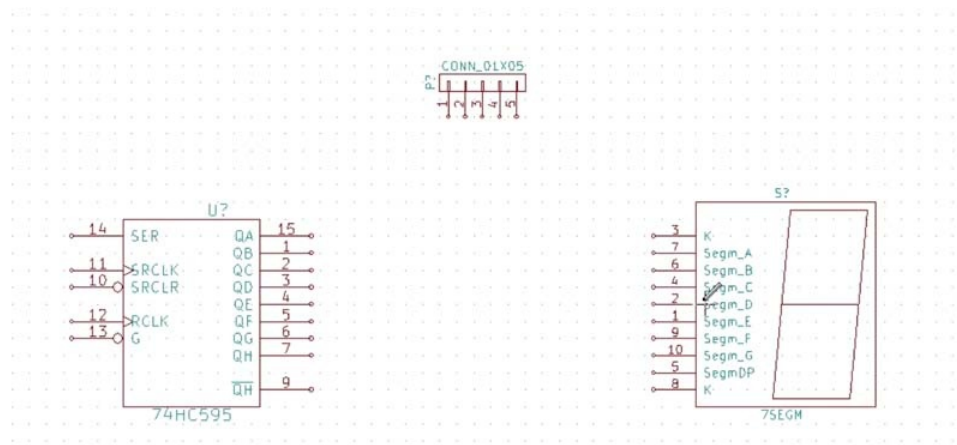
Added the seven segment display component.

Next one up is the 595 shift register. Hit the A key, and type “595” in the filter.



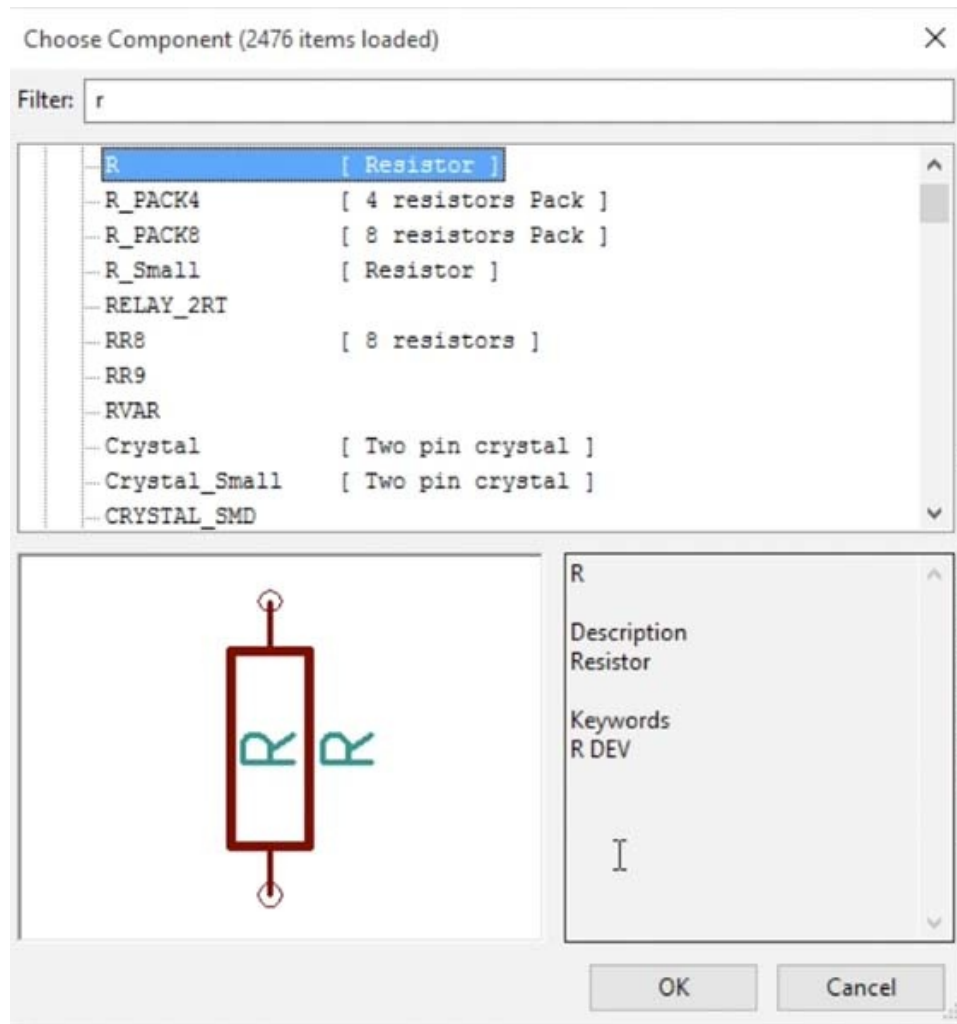
You can use the filter to quickly find a component. In this case, type “595” to look for available 595 shift register components.

The filter will return two available options. Look at the schematic preview of each option and confirm that it is pin-compatible with the actual part on the breadboard. Look at the pins one by one, the pin number and the pin role from the preview schematic, and compare it with the information available in the actual part data sheet. In this case, the part that we want to use is the 74HC595 version. Double-click to select it and place it on the canvas.

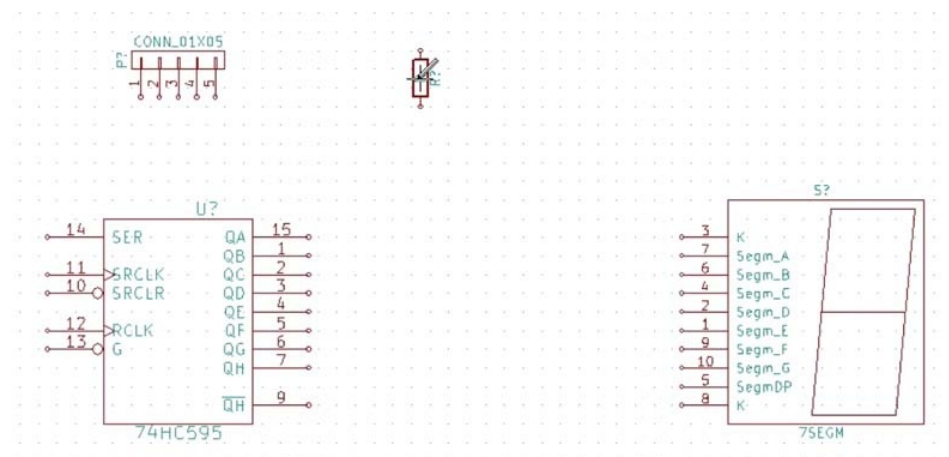


Added the 595 shift register to the schematic.

Finally, lets add the current limiting resistors. Type A to reveal the component chooser and look for a resistor. You can simply pick the generic resistor option, and place it on the canvas.



Choose a resistor.



The first resistor is on the board. You may need to move the connector over the shift register to make some room.

Edit the value of the resistor by hitting the “V” key.

Edit Value Field

Text: 220Ohm

Size (mm): 1.270

Options: ☒ Vertical ☐ Invisible

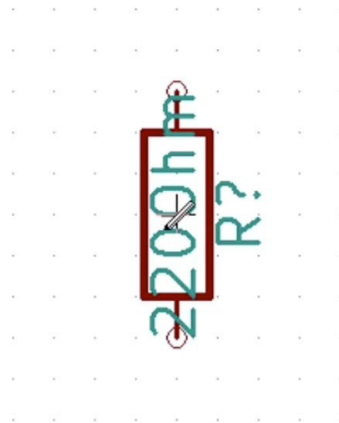
Style: ☒ Normal ☐ Italic ☐ Bold ☐ Bold Italic

Horizontal Justify: ☐ Align left ☒ Align center ☐ Align right

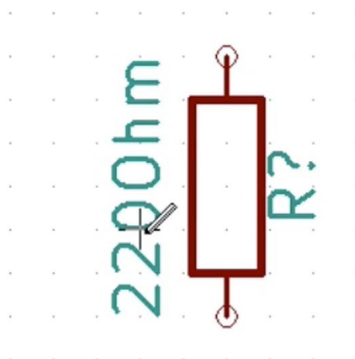
Vertical Justify: ☐ Align bottom ☒ Align center ☐ Align top

OK Cancel

Enter “220Ohm” in the Text box.



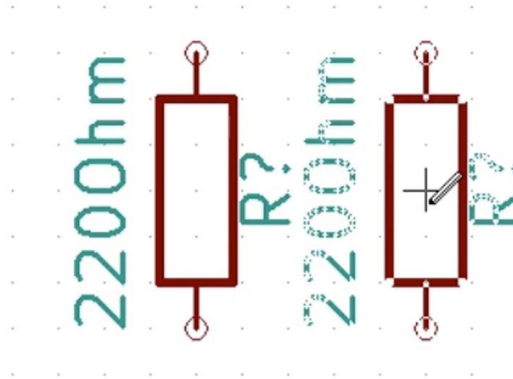
The value is visible with the component.



You can move the text to the side of the component so that it is more readable.

I need an additional seven resistors. Instead of adding them one at a time like we did for the first one, we can save time by copying the first already setup resistor component.

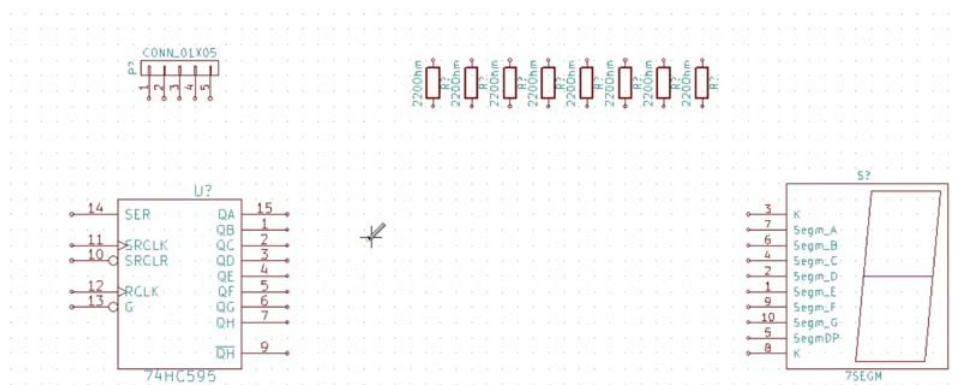
To do this, place your mouse cursor over the resistor, and type “C”. This will create one copy.



Use the “C” key to create copies of a component.

Repeat the process until you have a total of 8 resistors on the canvas. Move them as needed to space them out and align them nicely.

In the end, you should have something like this:



All the components are now on the canvas.

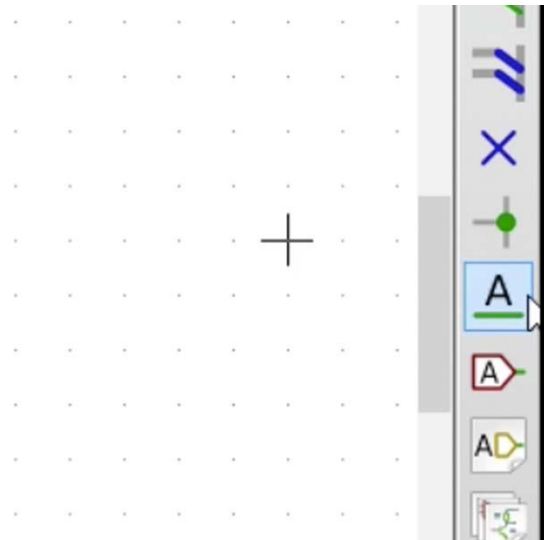
These are all the components that our schematic will have eventually. In the next chapter we will work on the wiring.

Chapter 38: *Create nets and labels*

In this chapter we will connect the components placed on the canvas. In the first project, you learned how to use individual wires to connect the pins of the components, manually, using the W key. As your schematic becomes larger and more complicated, this method of wiring will increasingly produce schematics that are difficult to read. Too many wires will be going to too many places, criss-crossing each other, and increasing the chances of errors.

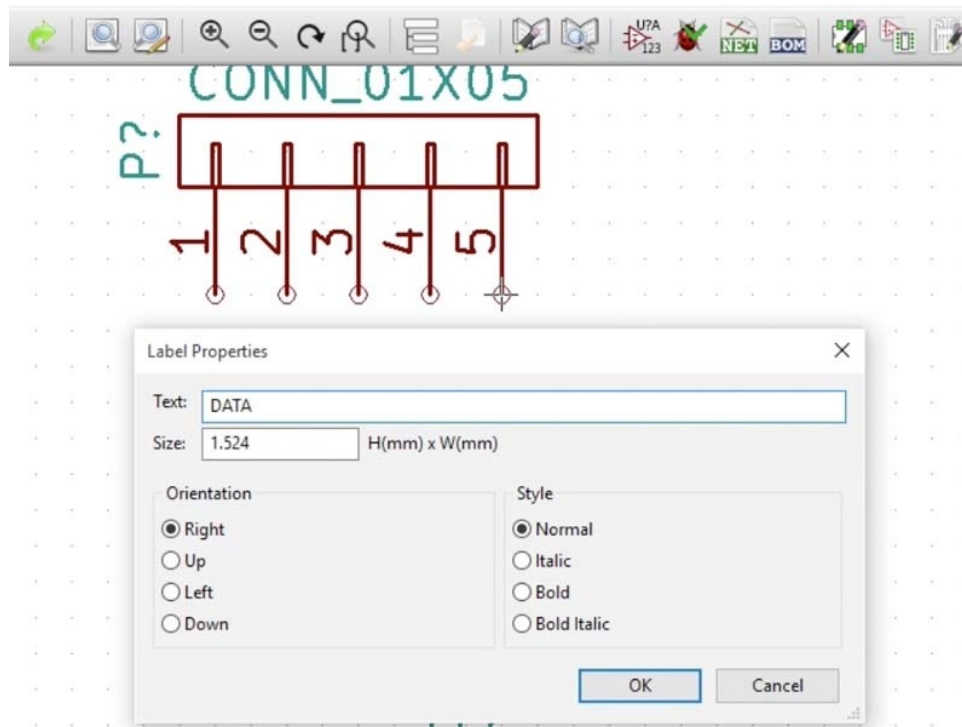
In this chapter I will show you how to use labels with which you can mark pins and logical connect them, without having to draw an actual wire. Let's work on an example to demonstrate.

Let's do the wiring for the Data pins.

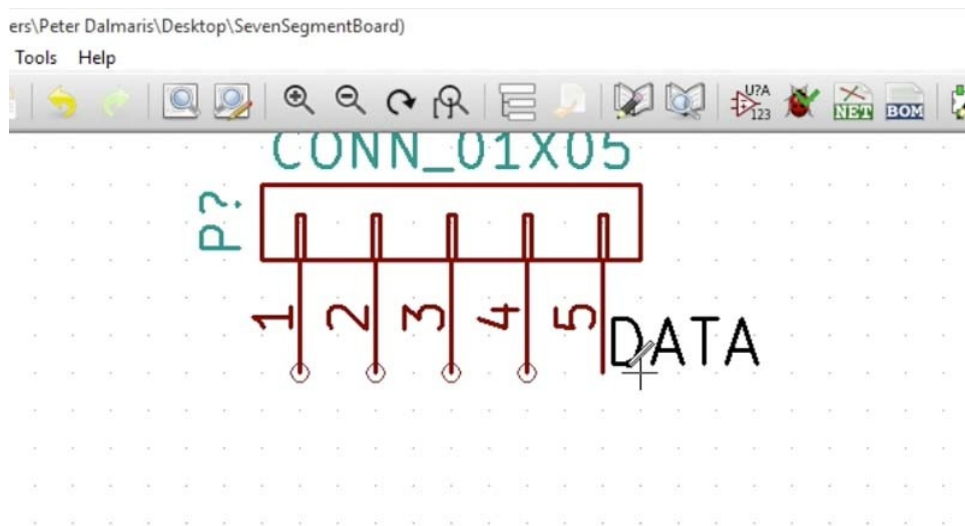


To add a wire label, click on the A key.

To add a label, click on the A button, as I show in the image above. Then, click on the wire that you want to label, in this case it is the pin 5 of the straight connector. This will produce the Label Properties window. Type in the name of the label, “DATA” and click OK.

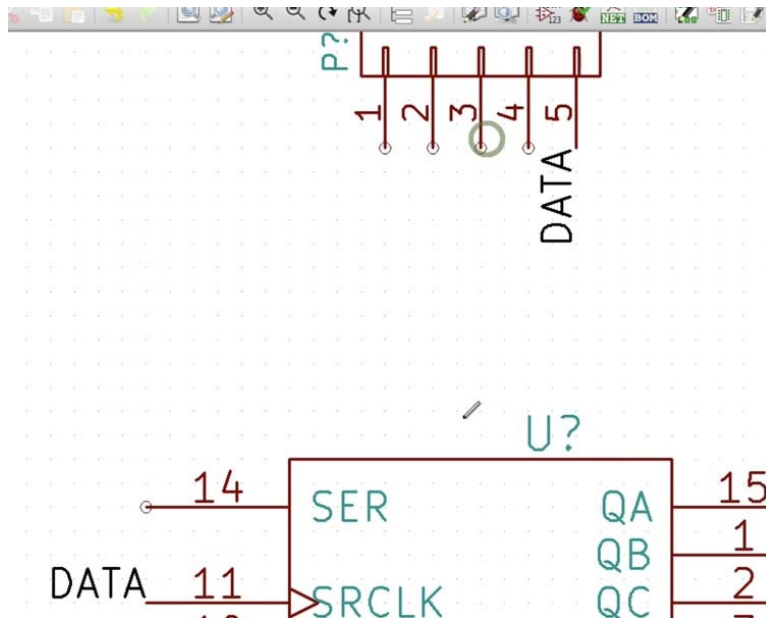


Once you are in label mode, click on a wire and give a name to your new label.



The new label is attached to the wire. You may need to rotate the label to align it.

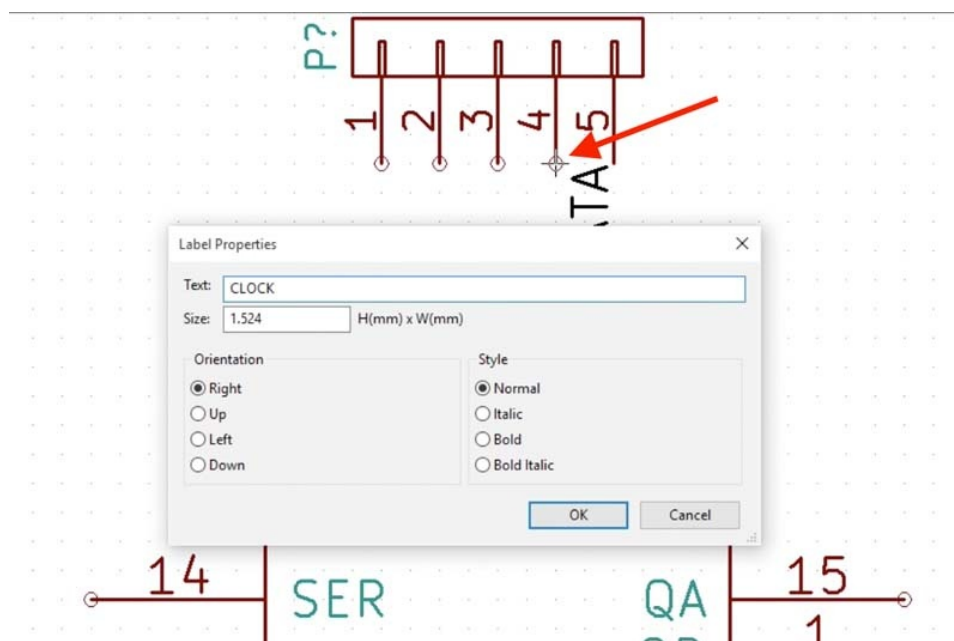
On the shift register, the pin where the data from the Arduino will arrive is number 11. Add a new label, also with the name “DATA” to pin 11 of the shift register. And again, rotate it if needed.



Attach a new label, also named “DATA”, to pin 11. Now, pin 5 of the connector and pin 11 of the shift register are logically connected, without a need for an explicit wire.

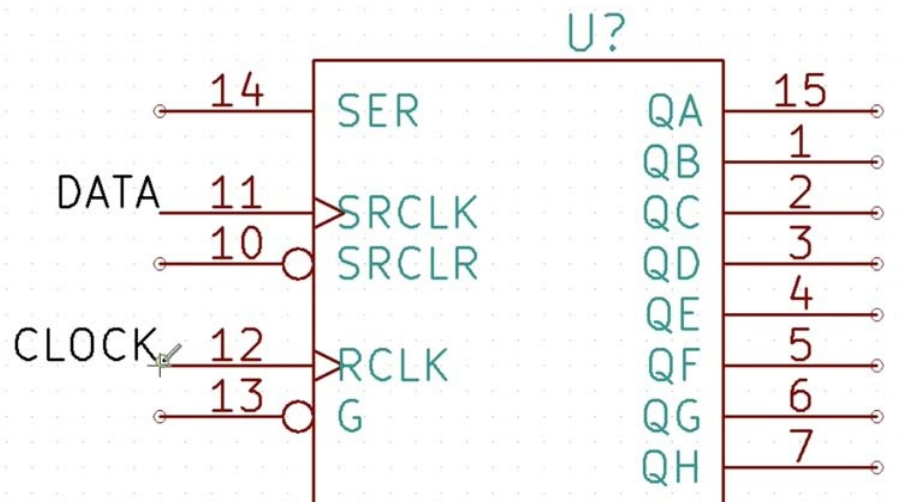
Now KiCad knows that pin 11 on the IC and pin 5 on the connector belong to the same net, so they will physically be connected in Pcbnew. You can also confirm that there is an actual electrical connection by doing an ERC check. The ERC check will show that indeed pin 11 on the IC and pin 5 on the connector are connected.

Let’s add another two labels. Label pin 4 on the connector “CLOCK”. Take care to use the “R” key to rotate the label, and to attach it to the end of the pin 4 line indicated by the small circle.



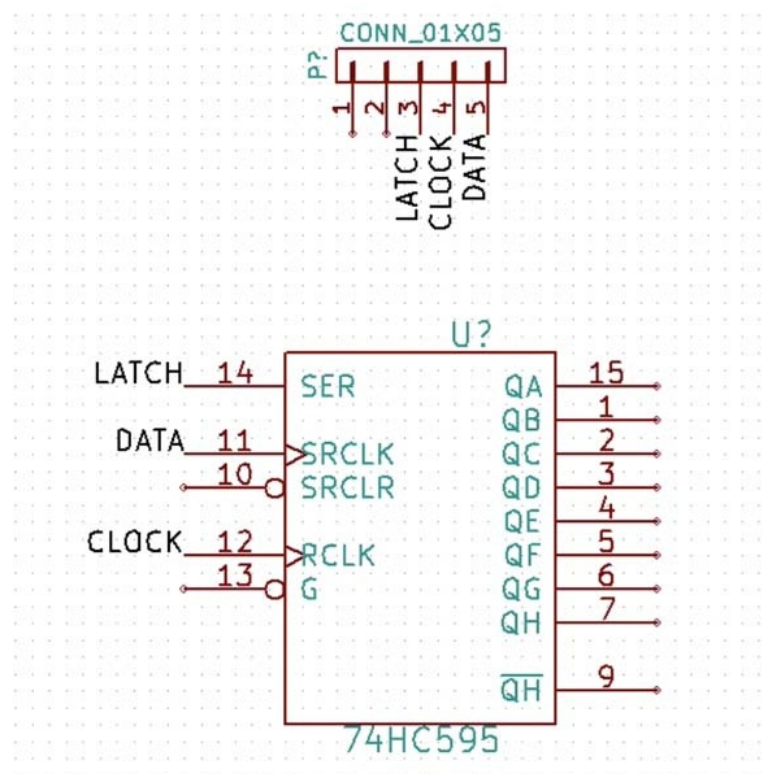
Pin 4 of the connector is labeled “CLOCK”. Take care to attach the label on the small circle that marks the end of the pin 4 wire.

On the IC, we want pin 12 to be connected to pin 4 of the connector, so we will also label it “CLOCK”, just like we did with the connector.



The IC pin 12 is also labeled “CLOCK”. This way, IC pin 12 and connector pin 4 are electrically connected (they share the same label).

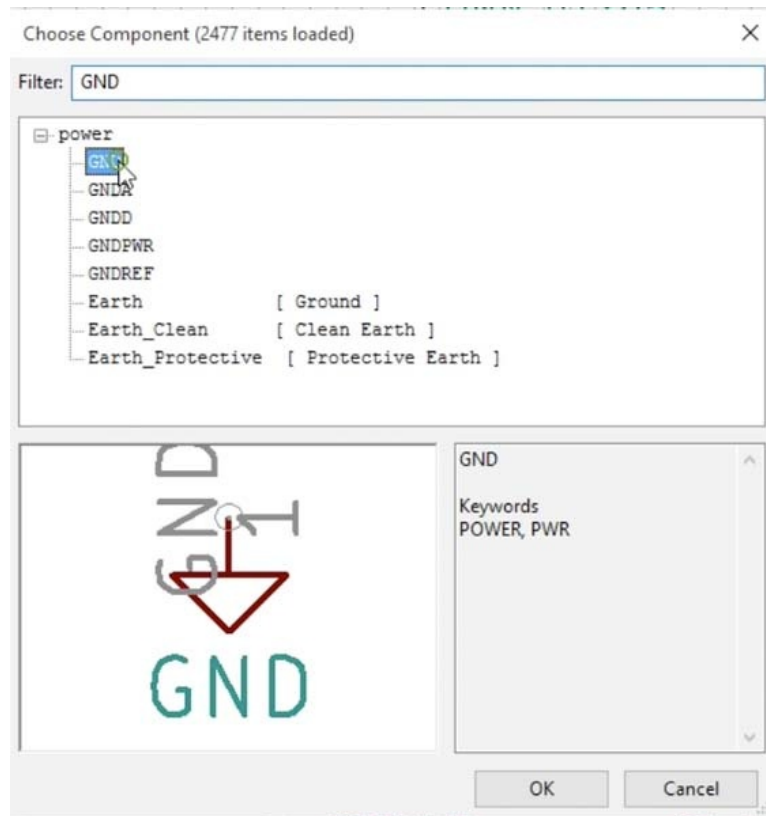
We now have two pins with the same label, and as a result, these two pins are electrical connected even though there is no visible wire in the schematic.



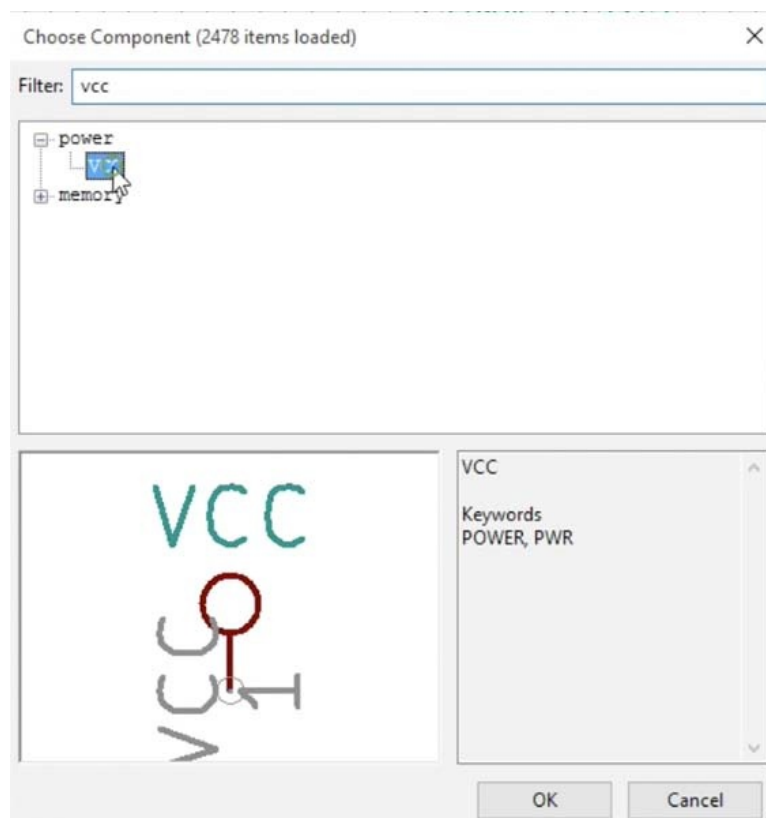
One more label added, “LATCH”, to pin 3 of the connector and pin 14 of the IC

Repeat the process for label “LATCH” as per the image above. We still have drawn no wires in the schematic, but we already have three completed and valid connections.

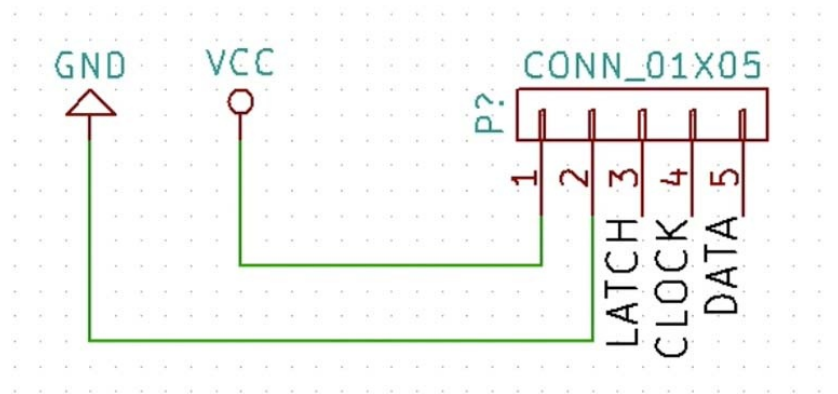
Let’s work on the power connections now. On the connector, I would like to attach Vcc to pin number 1 and GND to pin number 2. There are dedicated components for GND and for VCC. Type “A” to bring up the component chooser.



Use the filter to find the GND component. Double click to add it to the canvas

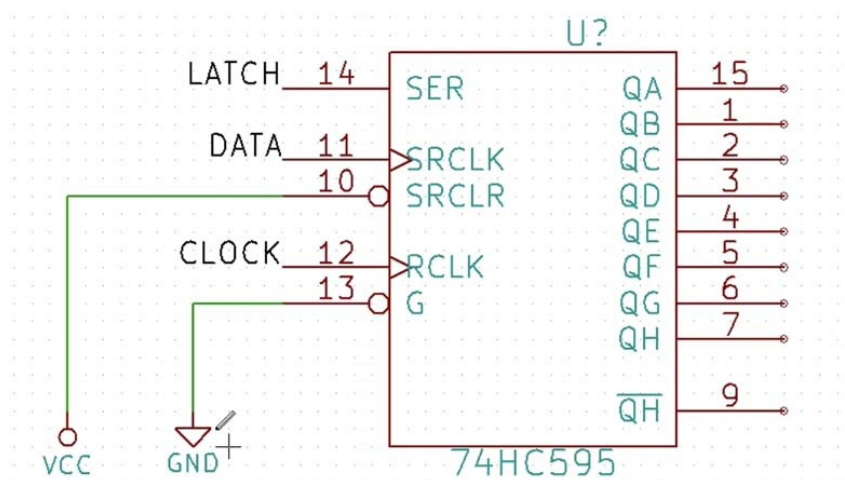


Use the filter to find the Vcc component. Double click to add it to the canvas.



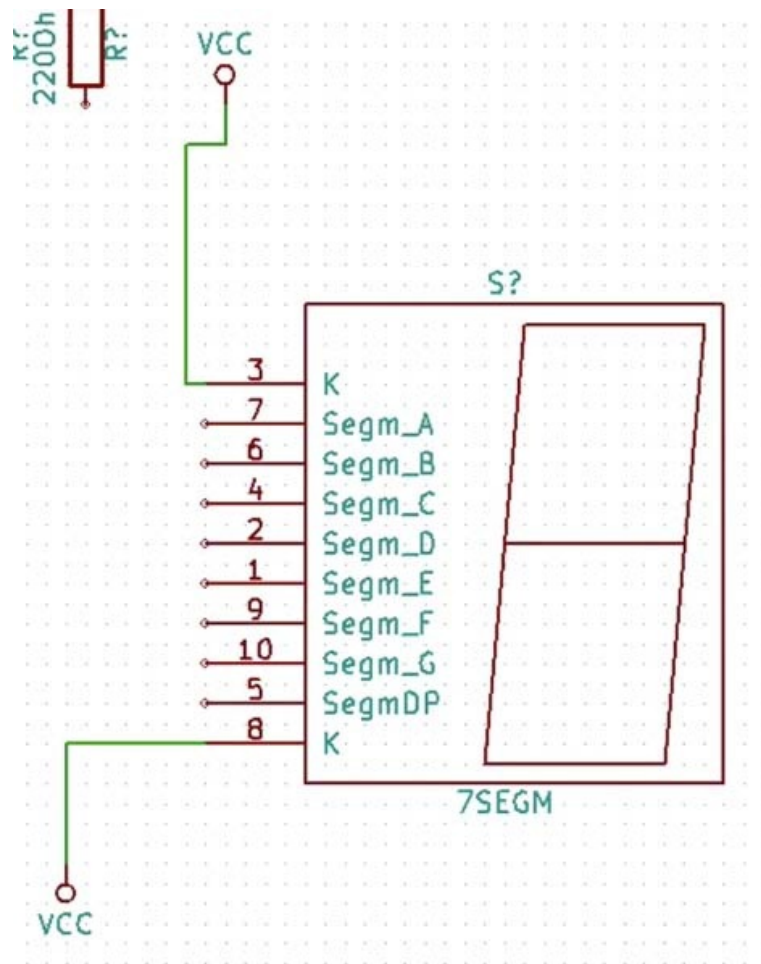
Rotate the GND and Vcc components and wire them to pins 2 and 1 of the connector respectively.

Use the filter to quickly find the GND and Vcc components in the component chooser. Double click on each component to select it and drop it on the canvas. Move them to a position adjacent to the straight connector, and use wires to connect GND to pin 2 of the connector, and Vcc to pin 1.



Repeat the process and connect Vcc to IC pin 10 and GND to IC pin 13. I could have used the existing GND and Vcc components, but this way creates a cleaner schematic.

Repeat the process for the IC. Add a GND and a Vcc component, and wire them to pin 13 and pin10 respectively.



Finally, add Vcc and GND connections to the seven segment display component.

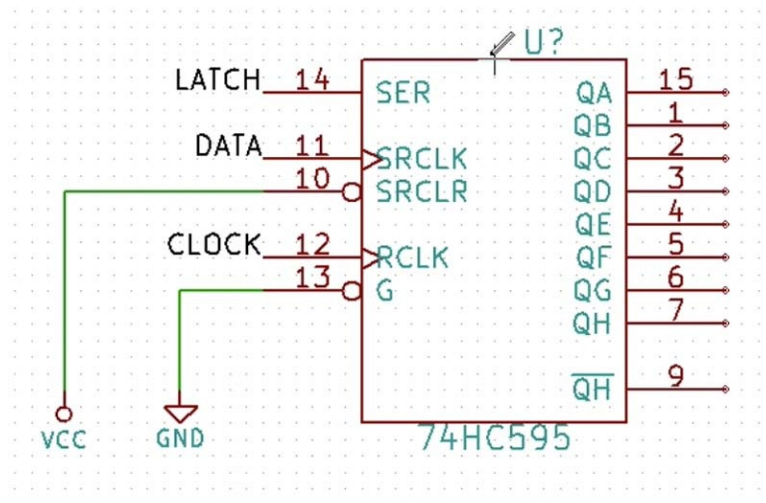
To finish the wiring of the Vcc net, go to the seven segment display. Add another Vcc component, and wire them to pin K, like in the image above.

The connections for the Vcc and GND nets are now complete. We have also used labels to connect three of the connector pins to three of the IC pins without using visible wires. This helps us to create a clean and easy to read schematic.

In the next chapter, I will discuss the concept of hidden pins and the power flag.

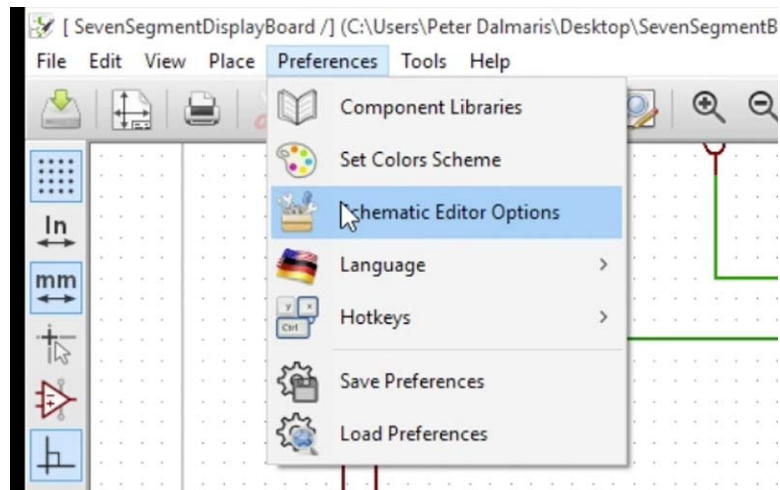
Chapter 39: *Hidden pins and the power flag*

In this chapter I will explain the concept of hidden pins. To do this, let's concentrate on the 595 integrated circuit. This IC is interesting because it contains hidden pins.

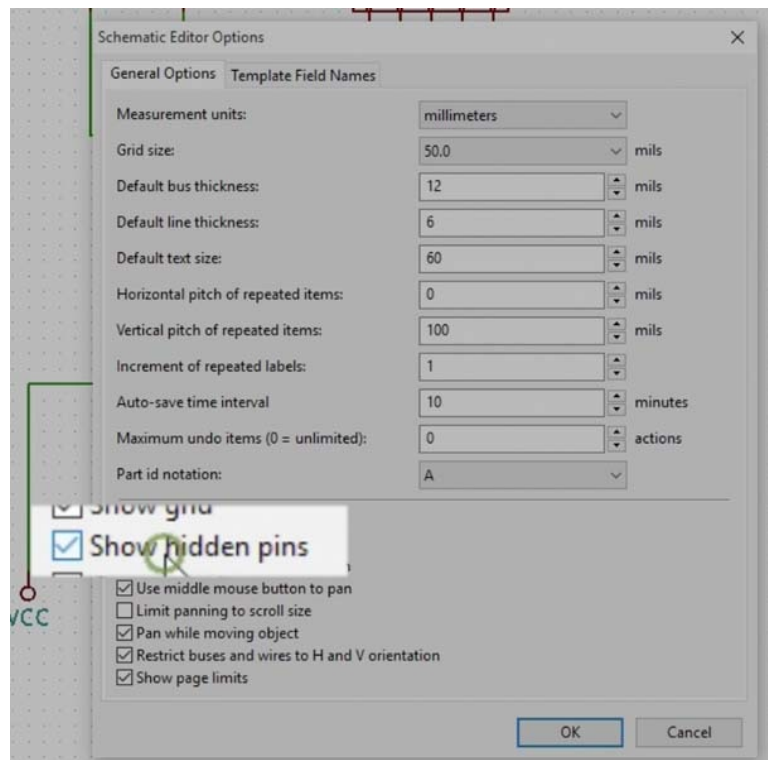


Take a close look at the 595 component. The power pins are not visible. They do exist in the schematic, but they are invisible.

You can see here that unlike the real 595 part, there are two pins that are missing, those are the power pins. The real 595 part has one pin for Vcc and one pin for GND. In the Kicad schematic component for the 595 IC, you can make those pins visible by going to the schematic editor options and clicking on show hidden pins.

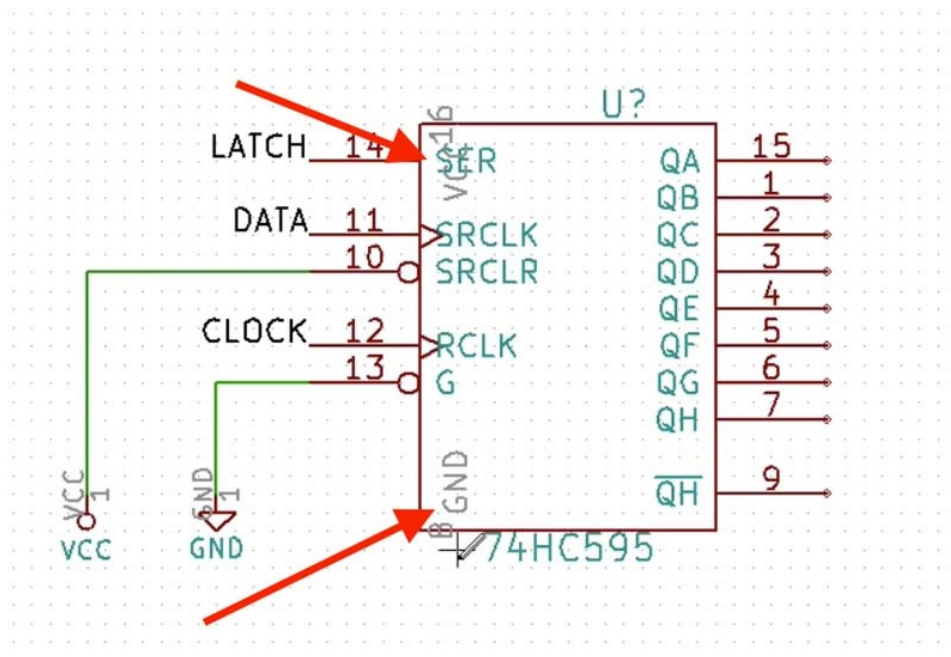


To make hidden pins visible, go to Preferences, Schematic Editor Options...



...Check the “Show hidden pins” option, and click OK.

When you do that, you’ll see pin 8, and pin 16, which are VCC and ground respectively.



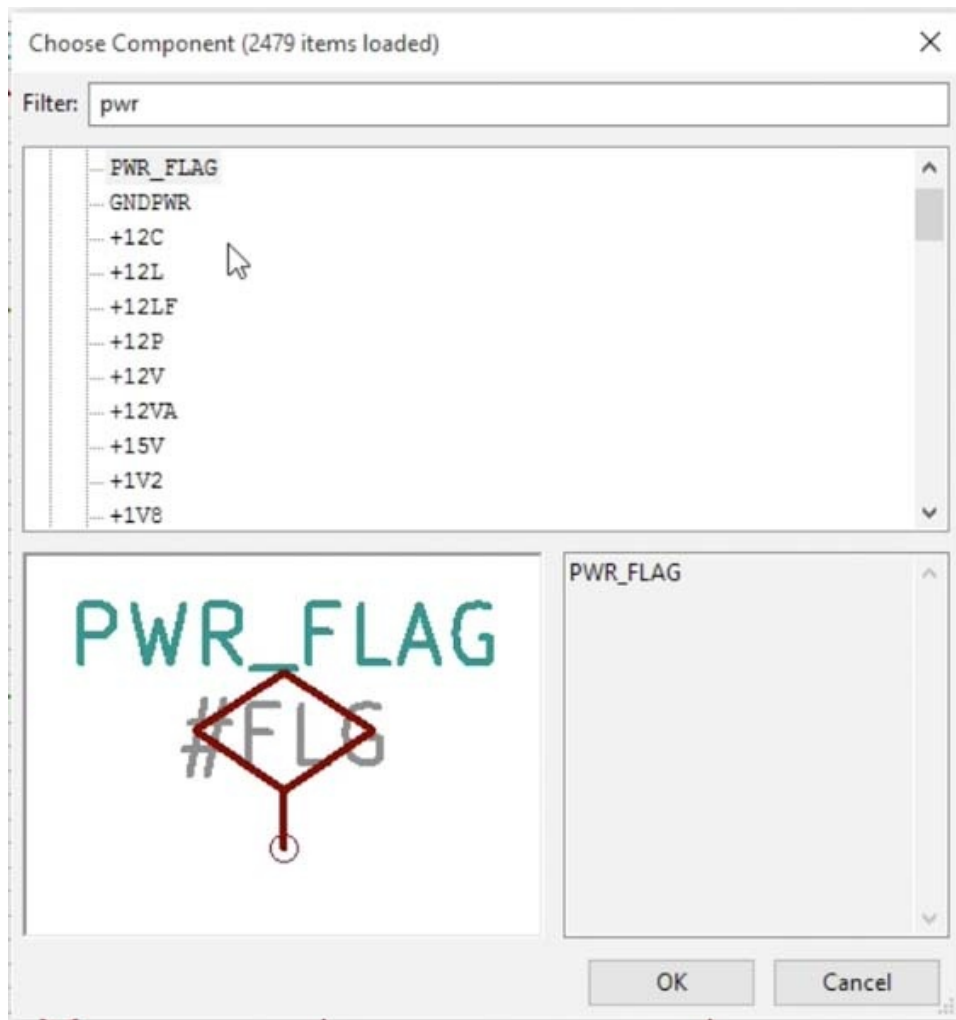
The hidden pins are revealed.

Hidden pins are... hidden because KiCad expects that somewhere in your schematic are nets that are marked as power nets. In the background, Kicad will automatically connect pin 8 and pin 16 to those nets. In our schematic, we already have Vcc and GND nets, but they're not marked as power nets yet.

Before we mark the power nets, restore hidden pins to their default state so that that they are invisible.

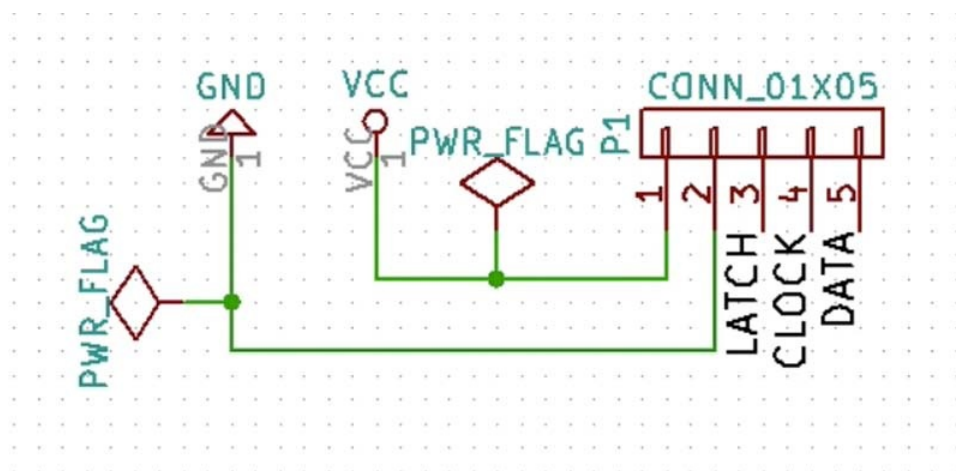
In order to mark a net or a connection or wire as a power net, you will need to use the power flag. The power flag is just another component, except that it is not really an electrical component. It's a KiCad component, or a component that KiCad uses with its rules check to confirm that hidden pins that require power are indeed connected to a power net.

Hit the "A" key to bring up the component chooser, and use the filter to search for the power flag. Type "PWD".



Use the filter to find the PWR_FLAG component in the components chooser.

Double click to drop the component to the canvas. Place it next to the GND net (wire) and use a wire to connect the flag to the GND net. Ensure that a solid green circle marks the joint (no circle means no connection). Repeat the process for the Vcc net, so that this is also marked with a PWR_FLAG. You should end up with something like this:



The GND and Vcc nets are marked with the PWR_FLAG.

If you do an ERC now, you will get error reporting that various pins are not connected to anything, but you will not get any errors concerning the GND or Vcc hidden pins of the IC. Those are now connected to the GND and Vcc nets in the background because Kicad is informed of their existence.

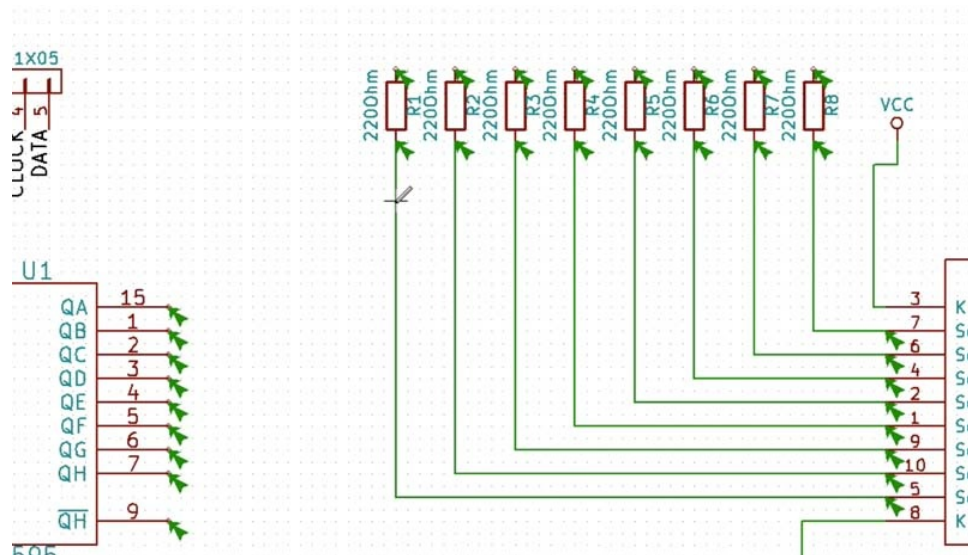
With this taken care of, let's move to the next objective: connecting the data pins of the IC to the resistors and the LEDs. We could use normal wires to do this, but you can imagine that the schematic would be very busy. Instead, we will setup a data bus. In the next chapter, I will show you how to make multiple connections using a data bus.

Chapter 40: *The data bus*

When you work with digital circuits, as we are in this project, it is often the case that two parts of the circuit transmit and receive a digital word, like a byte. The wires that transmit each bit of the word is a bus, and it can be thought of as a single bundle of wires. The keyword here is “single”. Instead of drawing the bus as seven separate wires, we can draw it as a single wire. To show to the reader, and to Kicad, that this is a bus, not a simple wire, we can draw it thicker than a normal wire.

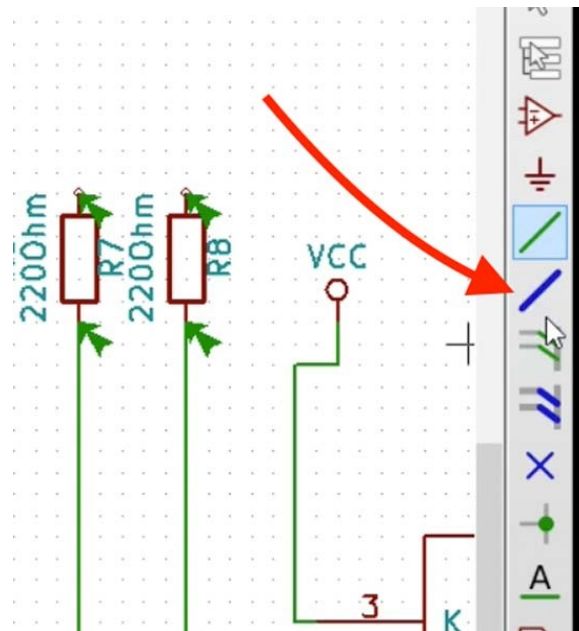
In this chapter, I will show you how to use a bus to connect the seven segment display to the shift register’s data pins.

Let’s start by wiring the seven segment display to the resistors, the “normal” way: by creating individual wires.



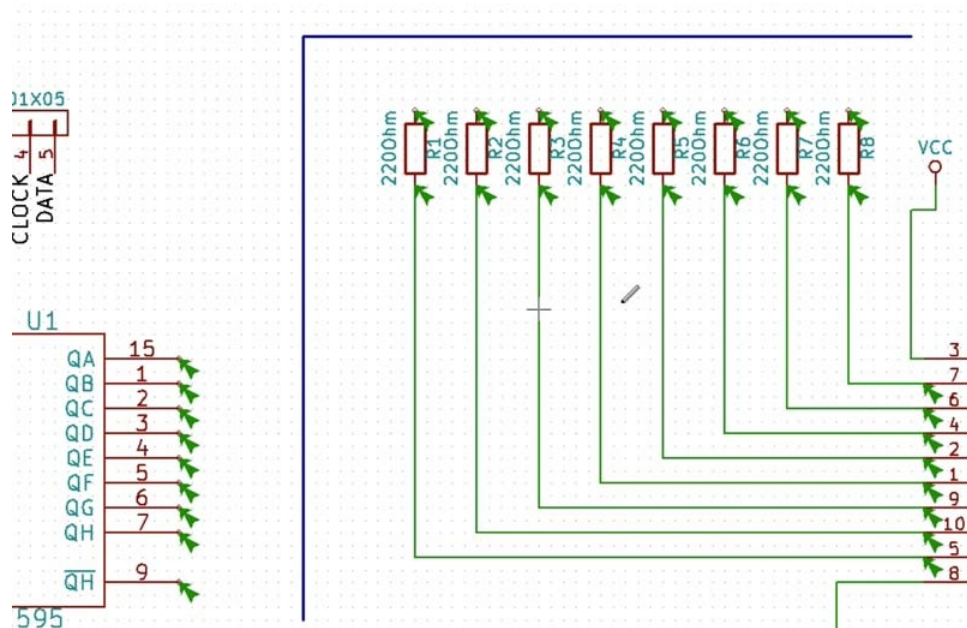
Nothing fancy here. The inputs of the seven segment display are connected to the resistors via wires.

Now, we will use a single bus to connect the shift register IC data pins to the other side of the current limiting resistors. To do this, start by clicking on the bus component button on the vertical tool bar.



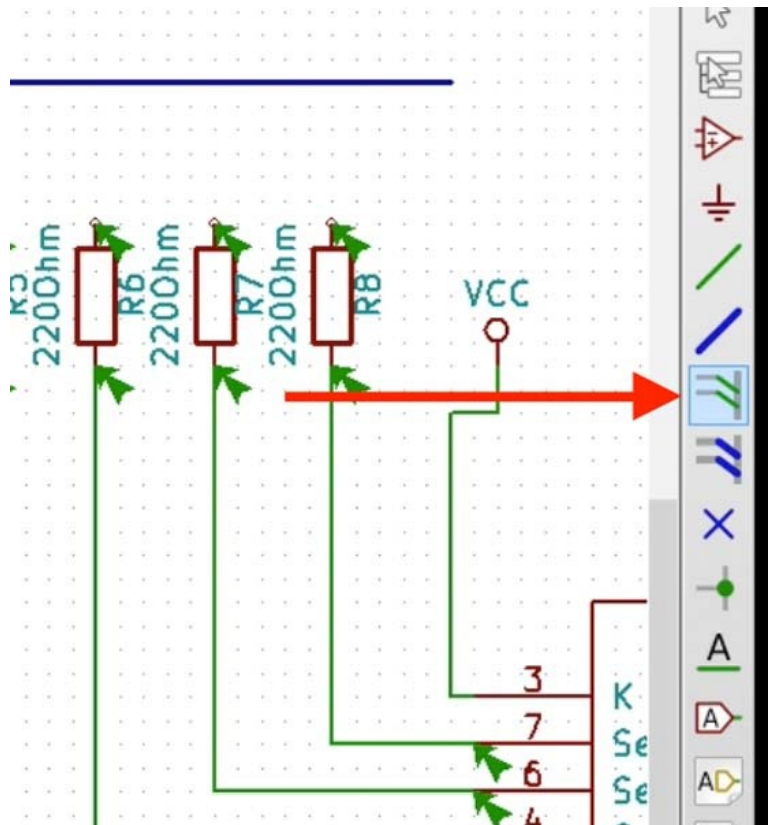
The bus component.

The bus component button looks like a straight blue line, as opposed to the normal wire that is a thin green line. Click on the bus button to enable it, and then draw the bus starting from above the right-most resistor to the bottom right corner of the IC, like in this image:



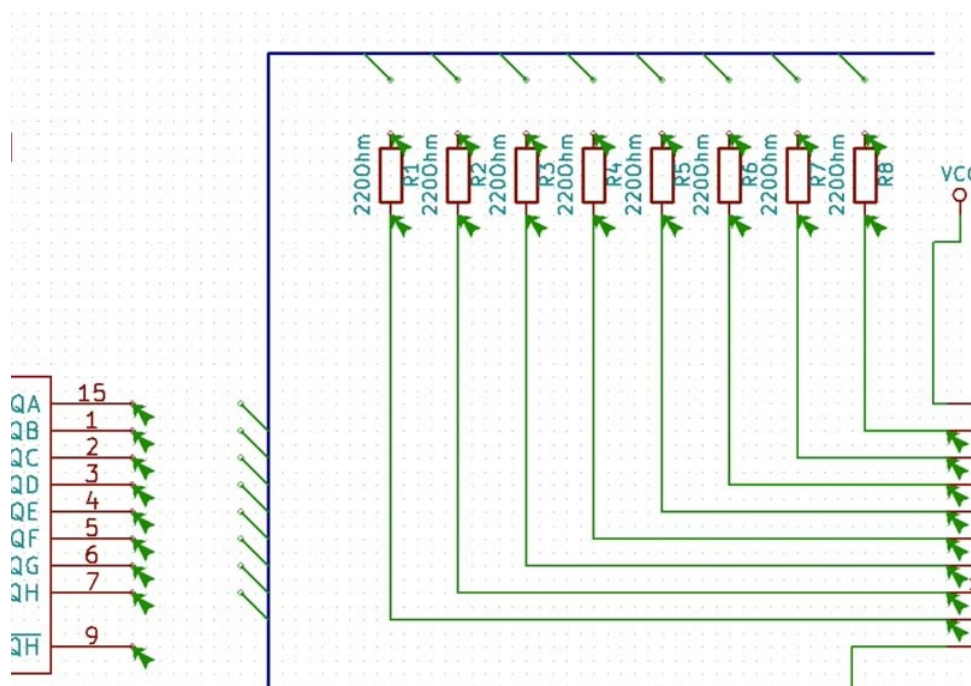
The blue thick line is a bus. Nothing goes in or comes out of it at the moment. We will add entries and exits next.

Next, let's add the entries and exits to the bus. We'll use the Bus Entries component for this task.



Use the Bus Entry component to add entries and exists to the bus.

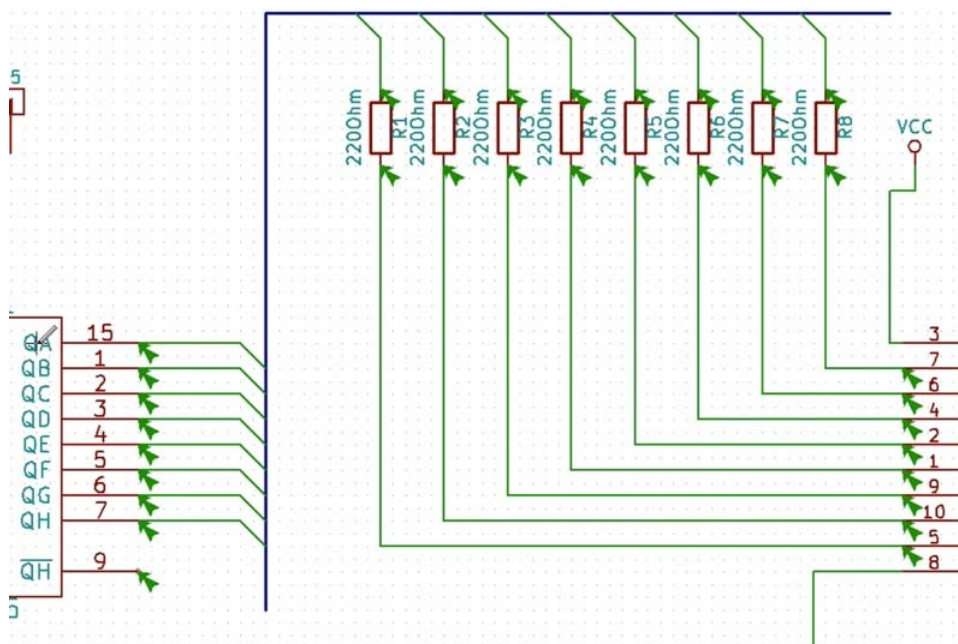
Click on the Bus Entry component to select it, then carefully start adding entry wires on the bus. This will take a few moments to get right when you attempt it for the first time. You need to work out the right spacing between the point where you click to add an entry and the bus. The bus entry wires are angled, and this makes it tricky to judge the correct distance at first. At the end of the process, you should have something like this:



The bus entries, added to the bus adjacent to the IC, and the exits over the resistors.

Once you have bus entries and exist added to the bus, you need to use normal wires to

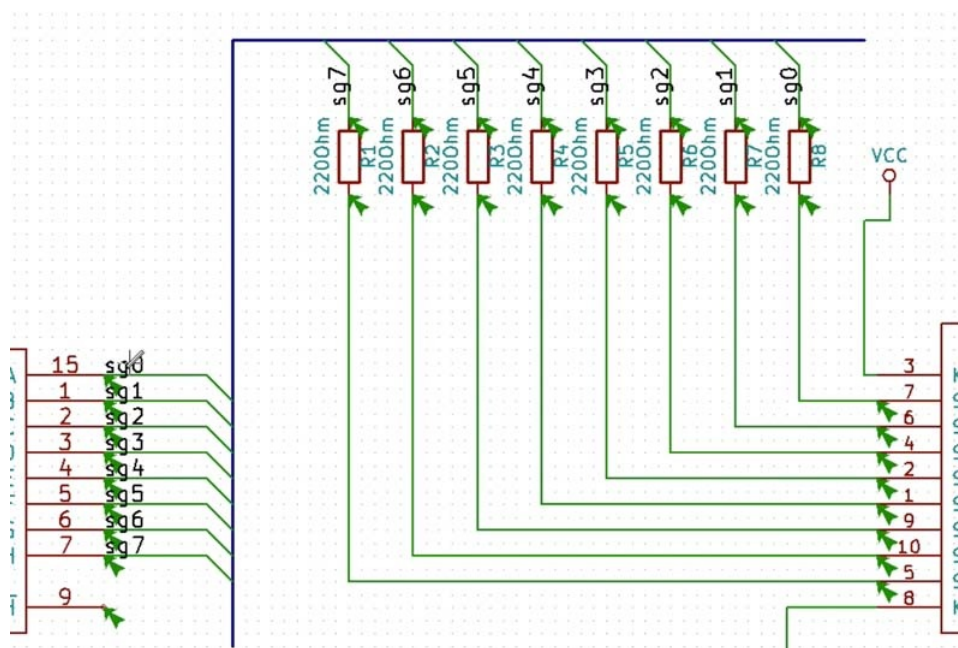
finish the connections between the bus entries and exists and the pins. Click on the wire button, and finish the connections. In the end, you should have something like this:



The bus entries/exists connected to the pins.

You may be wondering now, how does KiCad know that, for example, the QA or pin-15 of the IC is supposed to be connected to pin 5 on the seven segment display? The answer is that Kicad doesn't know unless you label both pins with the same name. To do this, we will add labels to all pins that are meant to be connected via the bus, like we did in the previous chapter.

Go ahead and start adding labels. Start with the IC and pin 15, and label that as "sg0", then pin 1 is "sg1" and so on. On the side of the resistors, label the right-most resistor as "sg0" (same name as pin 15 of the IC), then next resistor to the left will be "sg1", and so on. In the end, your schematic will look something like this:



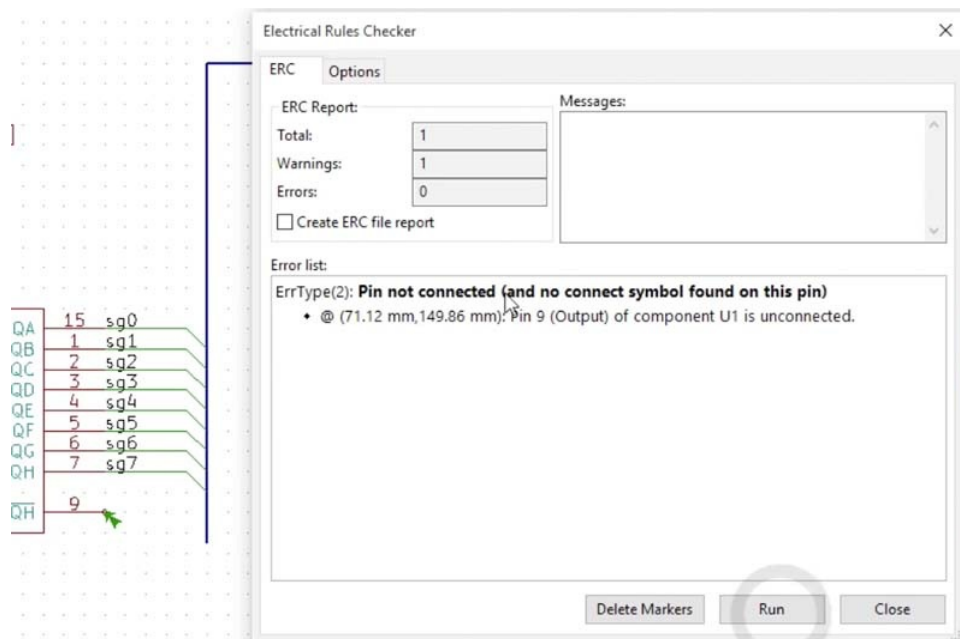
We have added labels to all pins that are connected to the bus.

At this point, our schematic contains all data pins of the IC connected to the seven segment display, via a bus. The power wiring is also done, and so is the wiring between the connector and the IC. Notice that pin 9 on the IC is still not connected? In fact, we don't want to connect it to anything. But, if we leave it like this, Kicad's ERC will complain about it. I will show you how to deal with this issue in the next chapter.

Chapter 41: *The unconnected component*

In the previous chapter, we completed all the wiring, but left IC pin 9 unconnected. In actual fact, we don't want to connect this pin to anything, as it is only useful if we want to cascade multiple shift registers, one after the other.

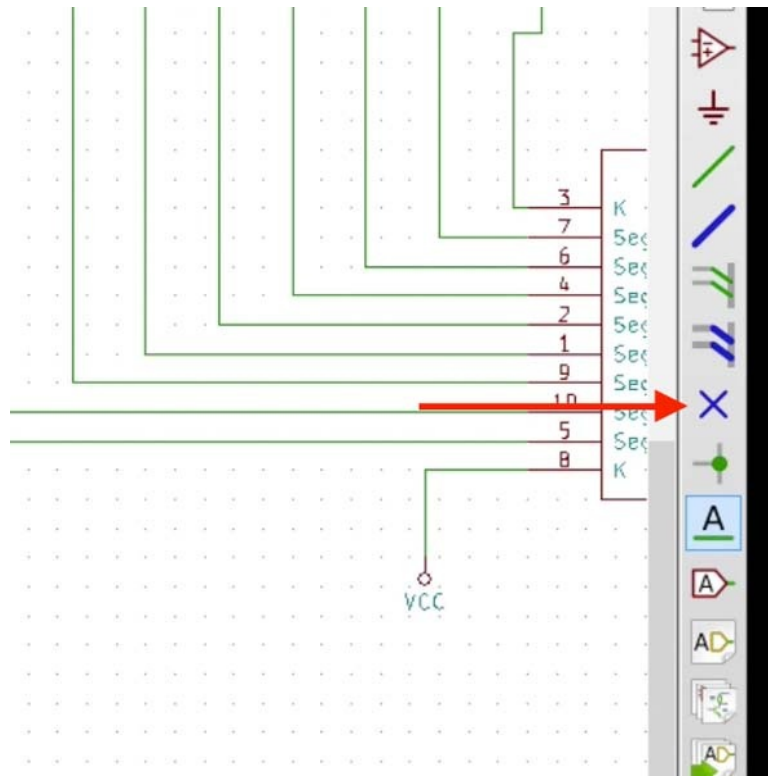
But, if we do an ERC now, we will get this report:



ERC complains that pin 9 is not connected. But we want to leave it unconnected!

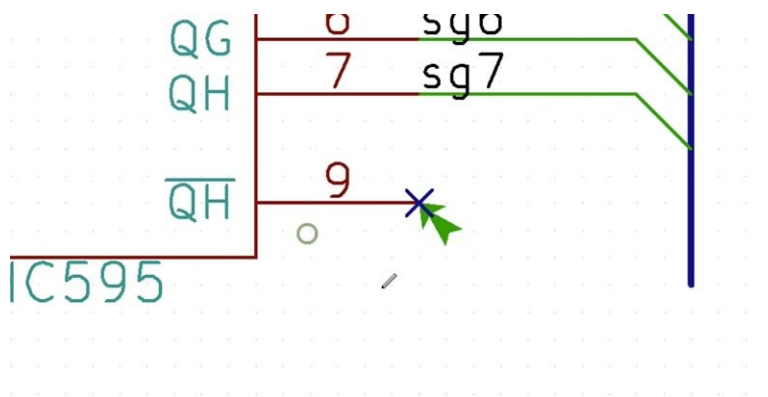
There is only one error. Pin 9 is left unconnected. Since we actually want to leave this pin unconnected, we must tell KiCad about our intention so that it doesn't bring it up as an error when we do an ERC.

There is a special component for this, it's the "not connected flag".



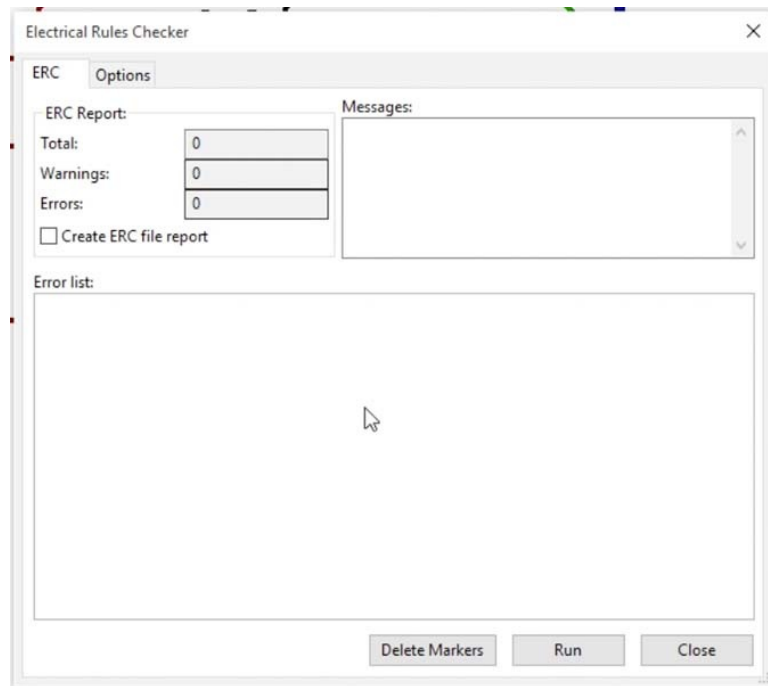
The “unconnected” component

Click on the Unconnected component button to select it from the vertical tool bar, and then click on the small circle that marks the end of the IC pin 9 wire to place it there. Now KiCad knows that I shouldn't have to worry about this pin not being connected.



Pin 9 is marked with the Unconnected flag. Kicad now knows that it is correct to leave it unconnected.

Let's do the ERC again:



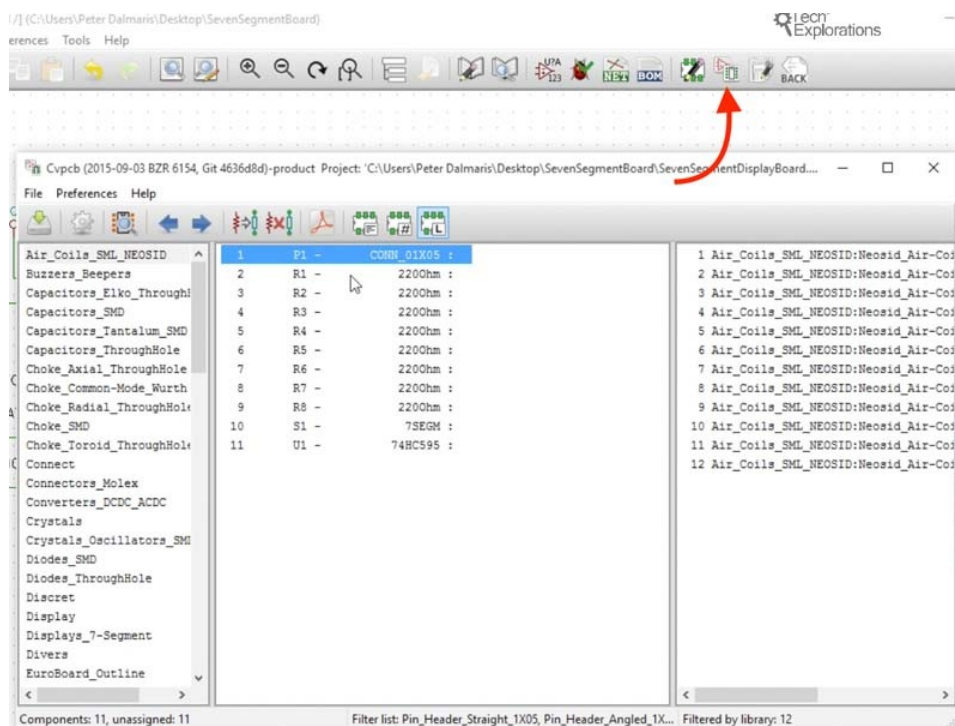
The ERC now shows no errors!

No errors showing in the ERC! This means that our schematic is complete!

Let's save our project now, and move onto the next task, which is to do is to do the associations between the schematic parts and the footprints before exporting the net list file. We'll do that in the next chapter.

Chapter 42: Component - footprint associations

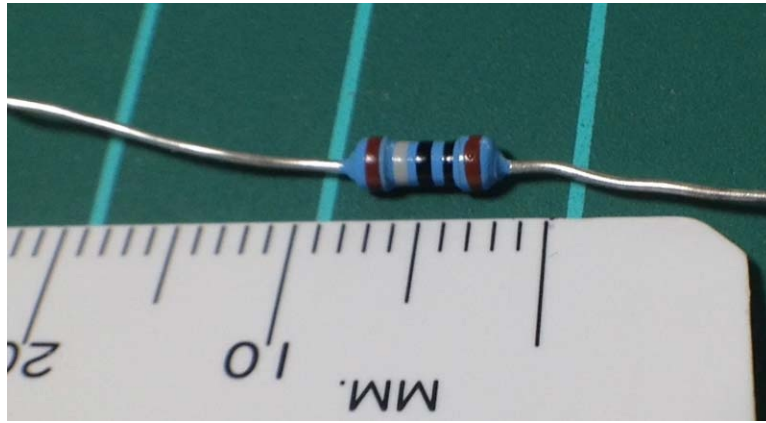
I'm happy with the schematic the way it is now, so I think we are ready to move on to the next step. The next step is to associate the schematic components with the footprints. After that, we will go into Pcbnew and do the layout. To do the associations, we will use Cvp pcb. Click on the Cvp pcb button to start this app.



Start Cvp pcb. The components that must be associated with footprints are in the middle pane.

It will take a few seconds for the panes to be populated with data from Github. The middle pane contains the components found in the schematic. We will associate these components with appropriate footprints.

Let's start with the resistors first. In this project we are working with through hole components, so we will use resistor footprints from the "Resistors_ThroughHole" library. There are a lot of options to choose from in this library. Since we are working with footprints, we just need to find one that has the correct dimensions.



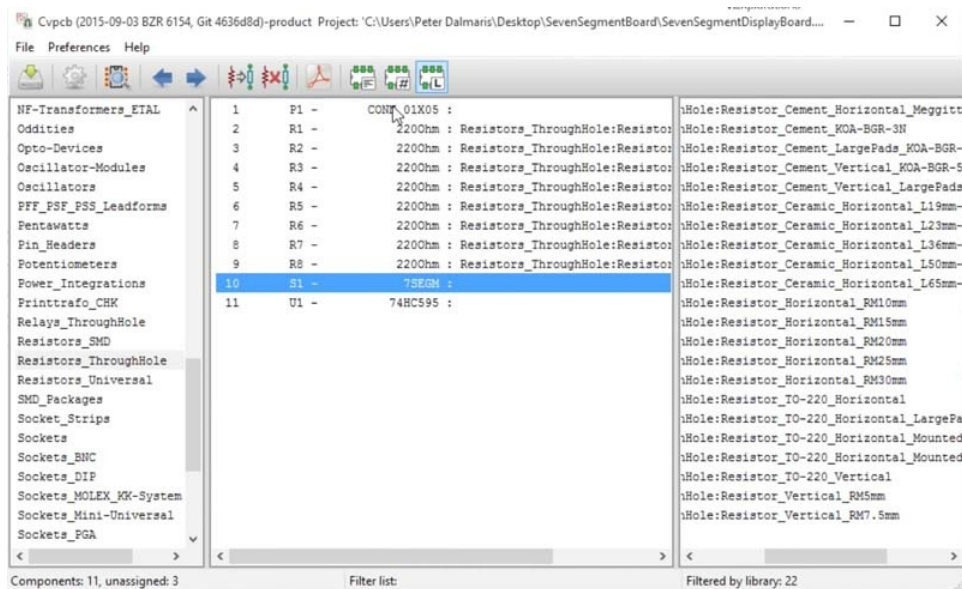
I want to use resistors with a 10mm pitch.

Take out your ruler and measure the pitch (distance) between the two ends of the resistors you would like to use on your PCB. In my case, the pitch is 10mm. With this information, I will look for a footprint that has a distance between the pads of 10mm.

```
:Resistor_Cement_Vertical_KOA-DGR-5N- /N  
:Resistor_Cement_Vertical_LargePads_KOA-  
:Resistor_Ceramic_Horizontal_L19mm-W8mm-  
:Resistor_Ceramic_Horizontal_L23mm-W9mm-  
:Resistor_Ceramic_Horizontal_L36mm-W11mm-  
:Resistor_Ceramic_Horizontal_L50mm-W14mm-  
:Resistor_Ceramic_Horizontal_L65mm-W16mm-  
:Resistor_Horizontal_RM10mm  
:Resistor_Horizontal_RM15mm  
:Resistor_Horizontal_RM20mm  
:Resistor_Horizontal_RM25mm  
:Resistor_Horizontal_RM30mm  
:Resistor_TO-220_Horizontal  
:Resistor_TO-220_Horizontal_LargePads  
:Resistor_TO-220_Horizontal_MountedFrom]
```

This option is a resistor with a distance of 10mm between the pads.

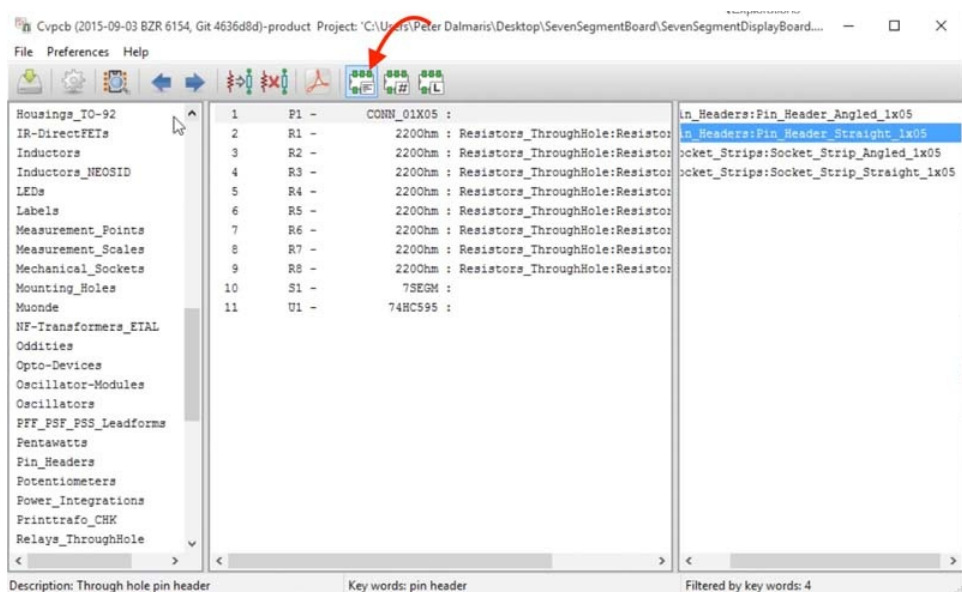
Make sure the L filter is selected, and scroll through the resistor options until you find one marked with “RM10mm” in its name. This resistor has a footprint of 10mm, which is the one I am looking for. You can confirm by creating a 3D view of the footprint (click on the button with the magnifying glass icon). Double click to select it and associate it with component R1. Continue double-clicking until all of the resistors are associated with the same footprint. In the end, your Cypcb window will look like this:



The resistors (middle pane) are associated with the 10mm through hole resistor footprint.

In that library, So there is a library with resistors somewhere here, yes, I'll manually going to look through it. All of the components in this project are through hole components. In the next project, in the last for this course, all of the components will be SMD, so for now I'm looking inside the resistor's through hole library.

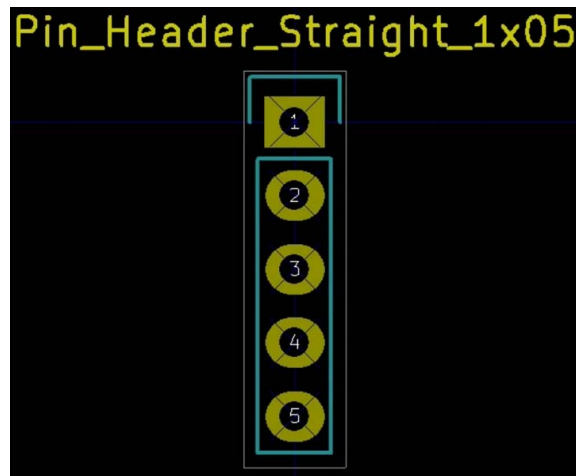
Next, let's work on the connector. We are looking for a connector with five pins in a single row. So let's look for this connector. A quick way to look for this footprint is by using the name filter. With the CONN_01x05 component selected in the middle pane, click on the name filter (the third button from the right). This will return 4 compatible footprints in the right pane:



The name filter can speed up searching for a footprint.

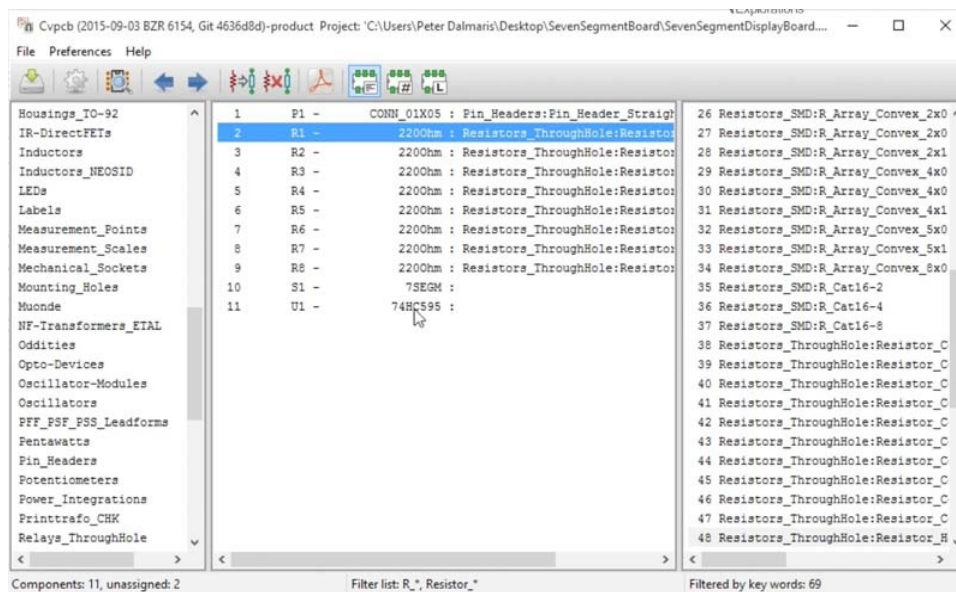
The options returned all involve 5 pins in 1 row configurations, either angled or straight. Use the pre-viewer to see what each one will look like on the board to help you

decide which one to use.



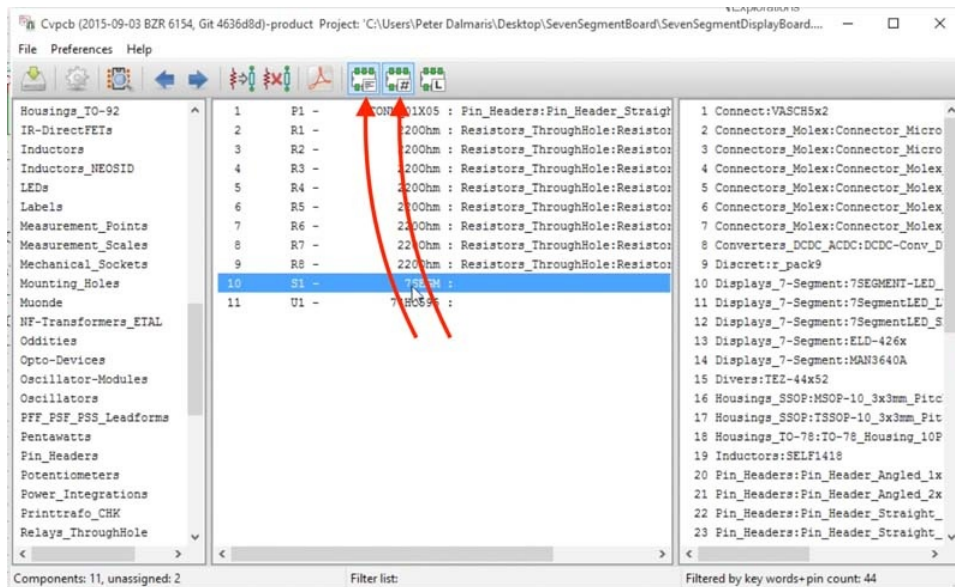
The preview of the 5x1 straight connector. This is the one we'll use.

So these footprints are here and there's 1x05 pins that is angled and one that is straight. Double click on the first option to do the association. Your Cypcb should now look like this:



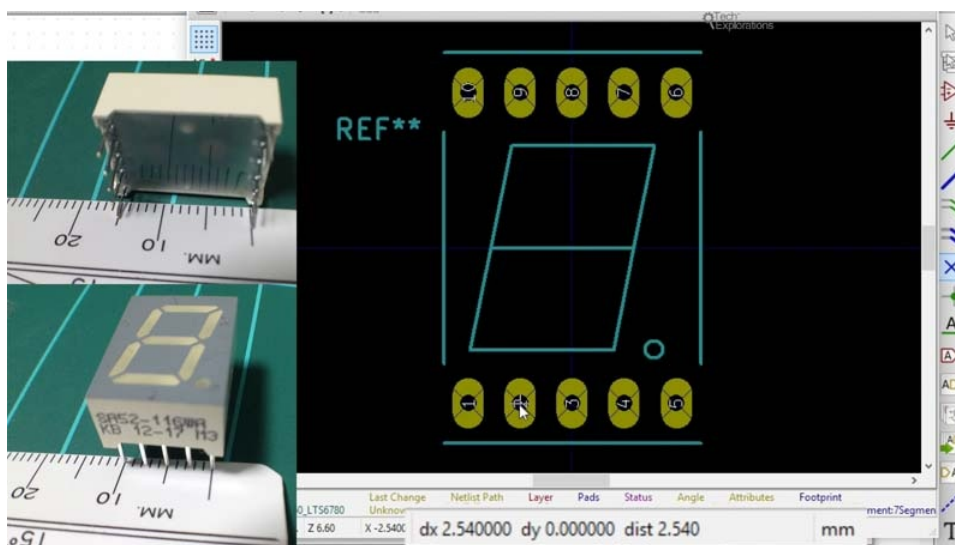
The connector is now associated with the 1x5 straight connector footprint.

Next lets work on the seven-segment display. The keywords filter does not return anything useful, maybe it does, but maybe we need to augment it with a number of pins filter.



With the keyword and pins filter selected, the right pane contains a small list of footprints that I can potentially associate with. We must find the best one manually.

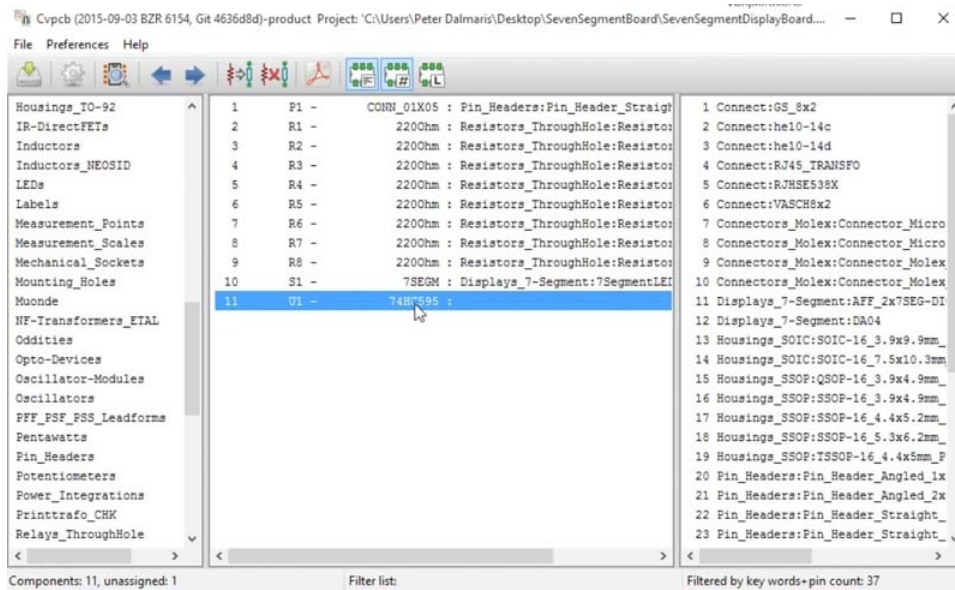
To find the best footprint for our seven segment display we must ensure that the type of pins (through hole) and the dimensions are correct. The name and the preview of each footprints will help us in this. Avoid components that contains the letters “SMD” since they represent surface-mounted components (we’ll work with them in the next project). Look for footprints that contain “7-segment” as a part of their name.



Footprint “7SegmentLED_LTS6760_LTS6780” has the right measurements and pin type.

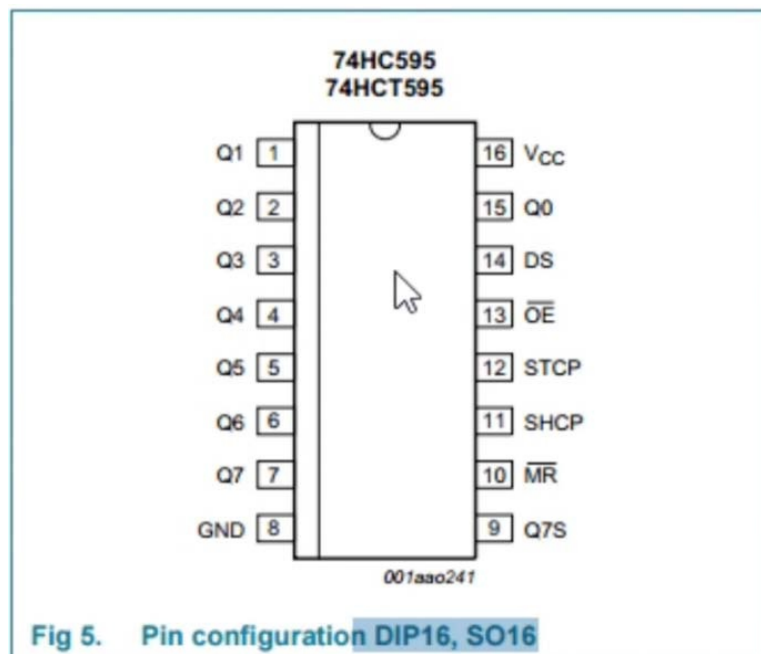
Looking through the list, find one titled “7SegmentLED_LTS6760_LTS6780”. This footprint has through-hole pins, and the dimensions between the pins are an exact match of the seven segment display I would like to use on my board (see image above). You can use the previewer to make manual measurements in order to confirm that match. The status bar in the bottom of the previewer window shows the measurement results. Remember to press the space bar to zero the counters. The distance between the neighbouring pins is 2.54mm and the distance between the two rows of pins is 15.24mm.

These are the dimensions of the real part as measured with my ruler. Double click on it this footprint to associate it with the seven segment display component. Your Cypcb should now look like this:



Completed with associating the seven segment display component with a footprint.

Finally, let's work on the shift register. It's a 74HC595 and I looked up the documentation and its data sheet, it's up here in the NXP website. And the part that I want to use is this one here and you can see that the name of the package that my shift register uses is DIP16 or SO16.

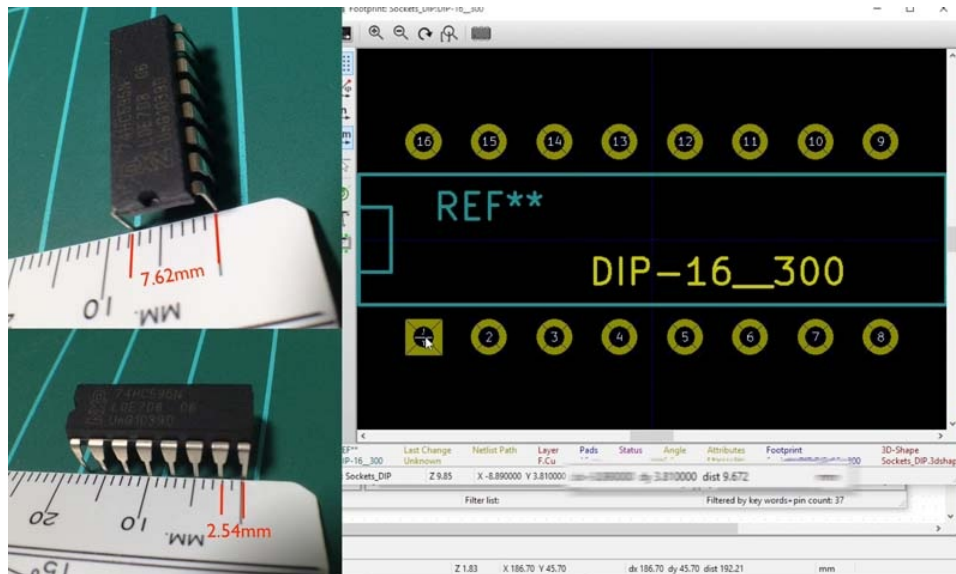


As per the data sheet, the 595 shift register I wish to use comes in a DIP16 package.

So, we should search for a DIP16 footprint. The way that we configured the filters

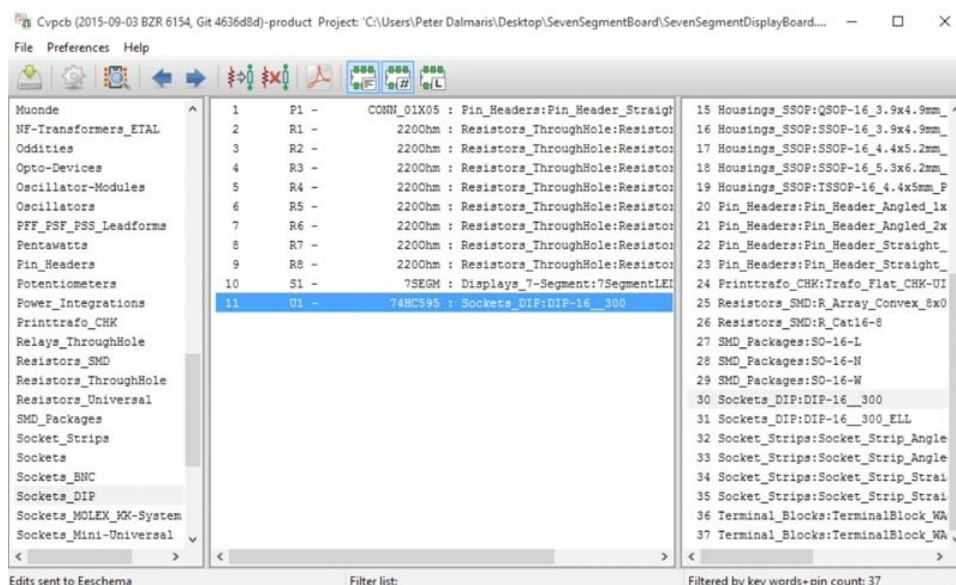
earlier return several options, one of which is the DIP-16_300 footprint, inside the “Sockets_DIP” library. In Kicad 4, this footprint is inside the “Housings_DIP” library.

Preview this footprints and ensure that its dimensions match the dimensions of the actual part.



It is important to cross-check that a candidate footprint matches the dimensions and pin type of the real part.

Just like with the seven segment display, measure the distances between the pins and rows of the real part and compare them against the preview of the candidate footprint. In this example, I have a perfect match (DIP16 packages are very common and standardised in the industry). Double click to setup the last association. Your Cypcb will now look like this:



All associations are complete!

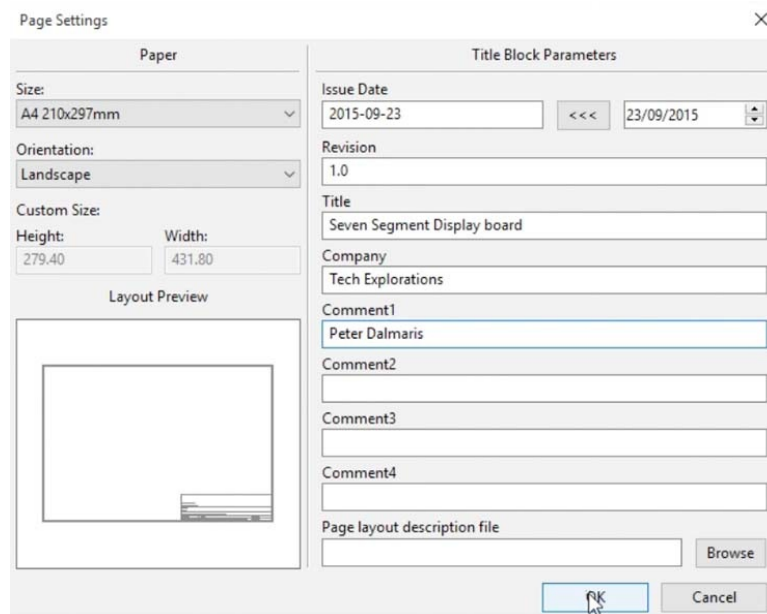
Now that the associations are complete, click on the Save button to commit them, and return to Eeschema. Before we forget, let’s save the schematic in Eeschema. The last thing to do before we move on to PCBnew is to export the netlist. Create the netlist and save it

in the project directory. In the next chapter we will start working on the layout in PCBnew.

Chapter 43: Create a 2 layer PCB in Pcbnew

In this chapter we will use Pcbnew to work on the layout and wiring of the second project PCB.

Start Pcbnew. Type the project details for the canvas legend by selecting Page Setting in the File menu. Type in your own details. Here's an example:

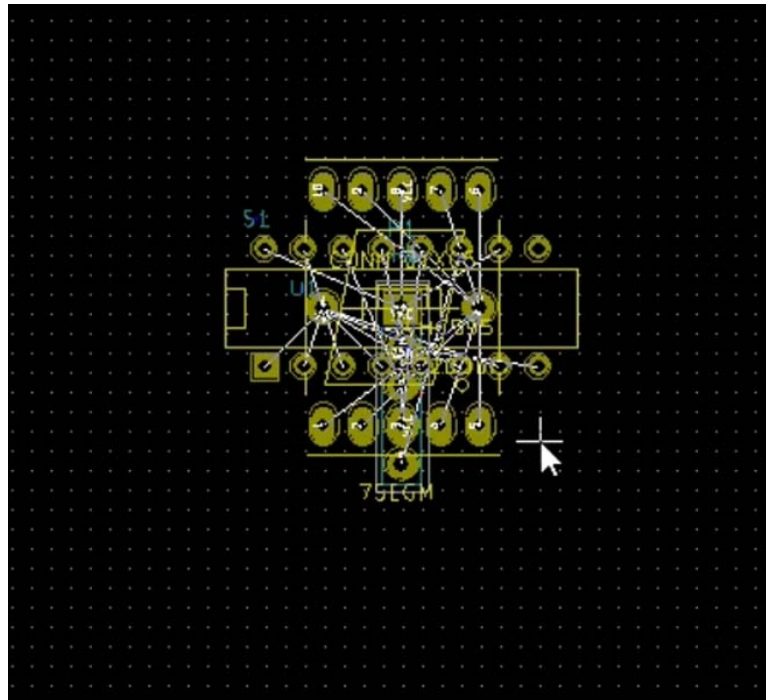


Setting up the Pcbnew project information.



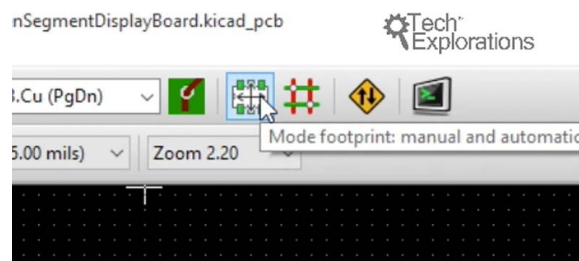
The project information appears in the Pcbnew legend.

Next, let's import the netlist file. Click on the "read netlist" button, navigate to the netlist file you created in Eeschema, and import it by clicking on the Read Current Netlist button.



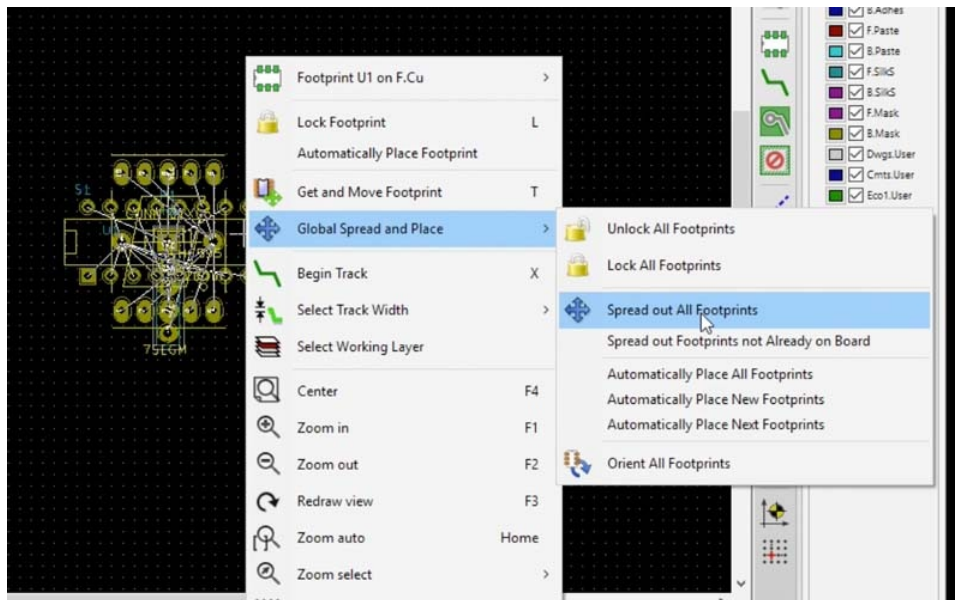
All the footprints from the netlist appear one on top of the other in Pcbnew.

Pcbnew will read the netlist file, import the footprints for the project, and place them one on top of the other. You could separate them by moving each one individually, but with so many components bundled together this is boring work. Instead, we will get KiCad to separate the components.



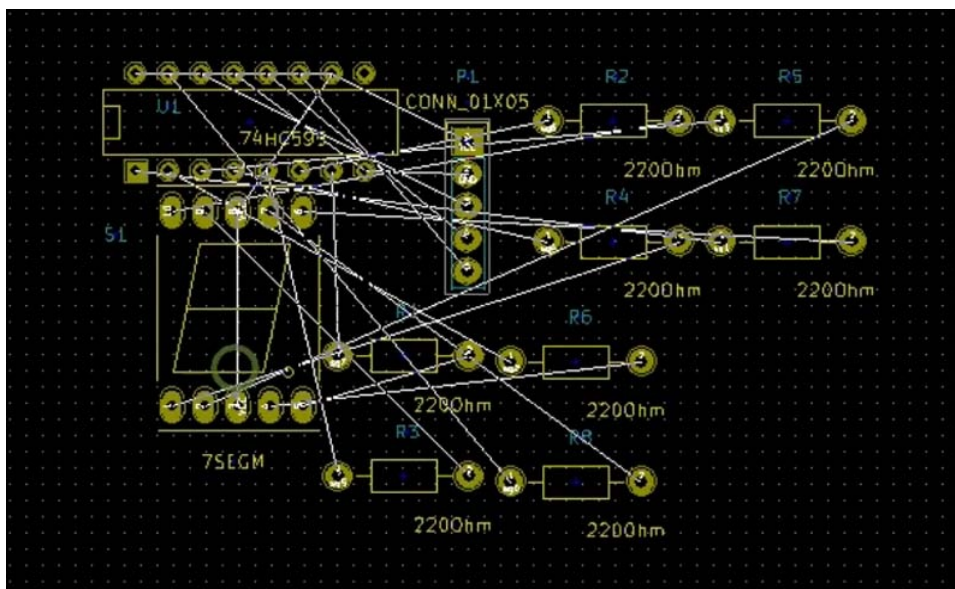
To separate the footprints, start by going into the automatic footprint mode.

First, enter automatic footprint mode, a mode that gives you access to several useful features in the Pcbnew canvas. Look for the Mode Footprint button in the image above.



Right-click anywhere in the canvas to bring up the contextual menu, and select “Spread out All Footprints” from the “Global Spread and Place” menu.

Once you enter Automatic Footprint mode, right-click anywhere in the canvas and select “Spread out All Footprints” from the “Global Spread and Place” menu. You will receive a warning that any footprint not locked will be moved. This is ok, as none of the footprints we just imported are locked. Locking a component makes it immovable by Kicad. Notice the Lock Footprint menu item in the contextual (right-click) menu.



Kicad has spread out the footprints. We can now start positioning them.

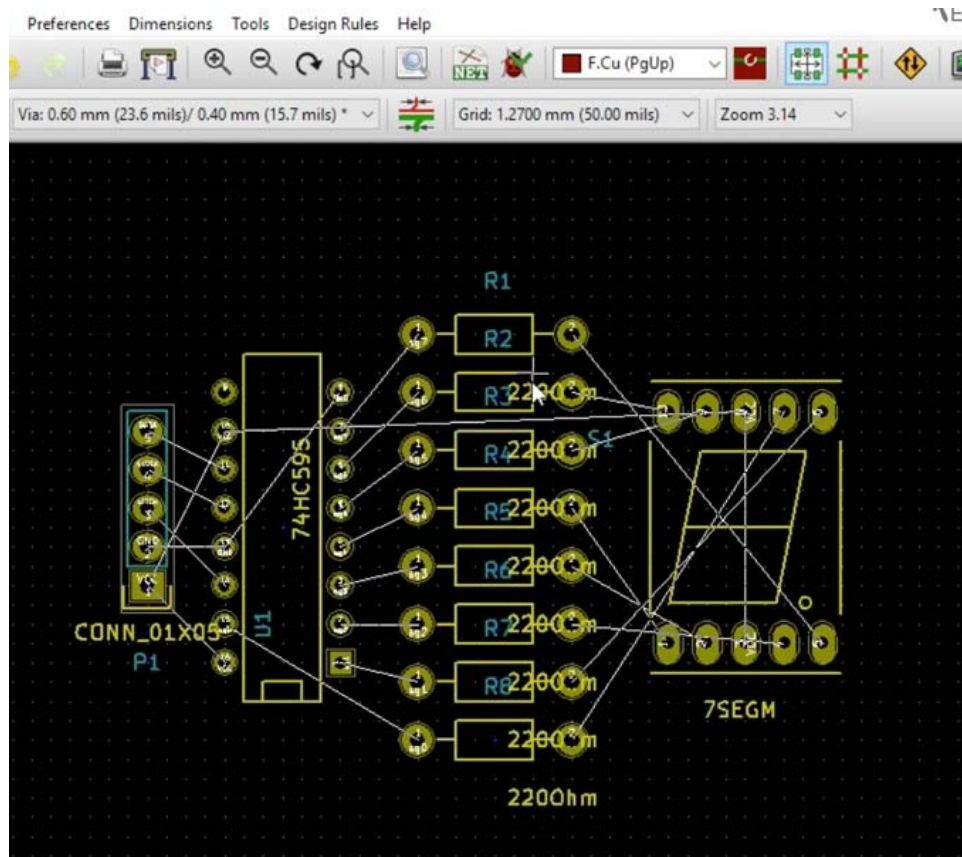
Click OK to dismiss the warning, and Kicad will spread out the footprint so that none are overlaying any others. We can now start positioning them.

We will be building a two-layer PCB. This means that although the actual parts will be mounted on the front side of the board, the traces will be routed on both the front and the back side. As much as possible, we will try to keep all the GND related wirings in the back copper layer. We will also try to keep the Vcc and data traces to the front copper

layer.

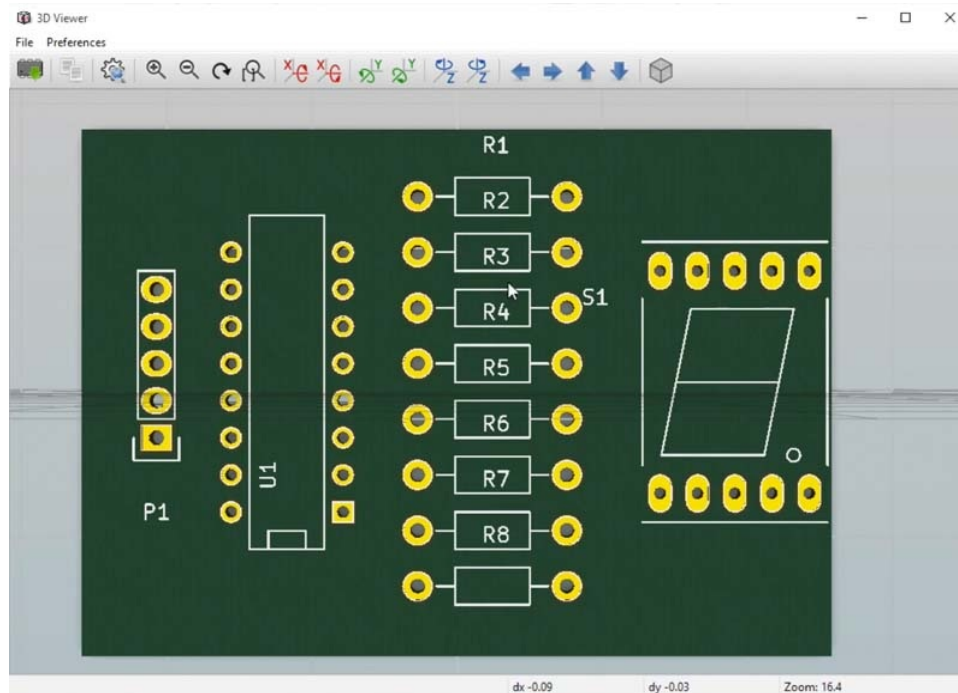
Let's start by separating the components and putting them approximately to the location that we would like them eventually to be on the board. We will use the M key (for "M"ove) a lot. Consider changing the grid to something bigger in order to make placement on a grid faster. I set my grid to 1.27 millimeters.

At the end of the placement process, my canvas looked like this:



This is, approximately, the final position of the footprints on the PCB. Also notice the grid setting.

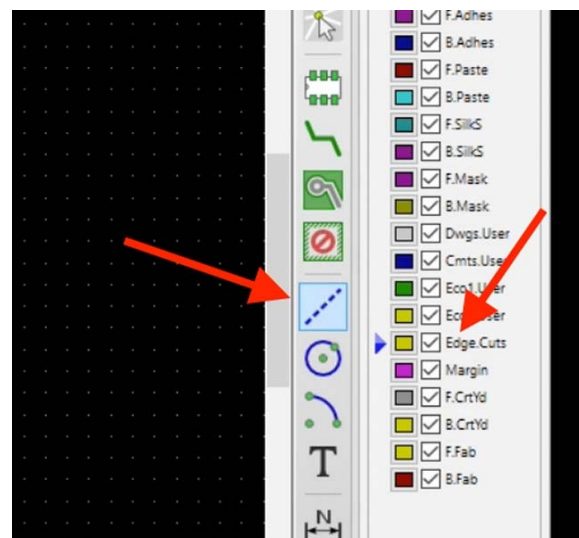
With this placement of the footprints, the 3D viewer rendering looks like this:



A 3D rendering of the board with the current placement of the footprints.

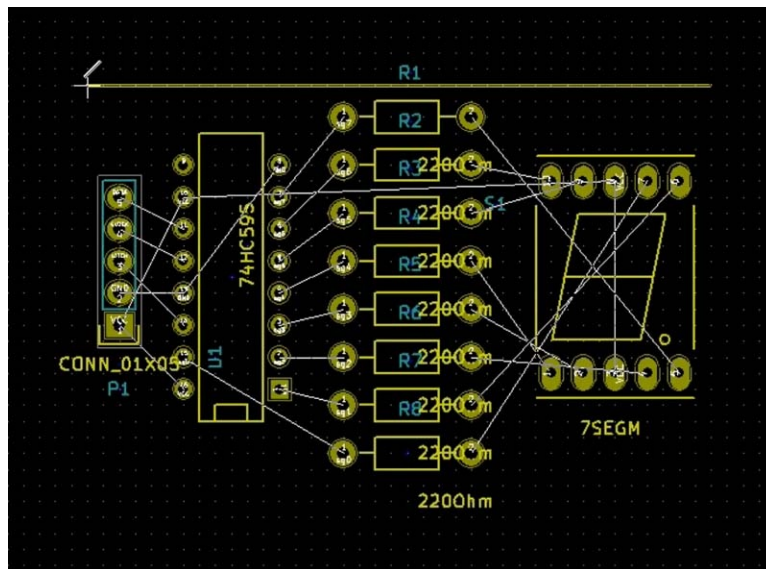
You can see that the silkscreen markings for the resistors (“R1”, “R2”, etc), appear to be misaligned. Let’s put a reminder to adjust the markings so that they are correctly inside the rectangle that denotes each resistor. This is a aesthetic adjustment, so I rather work on it towards the end of this chapter.

The next thing we want to do is to add the edge cuts perimeter for the PCB. This will define the limits of the PCB and therefore is useful to do before we start work on the the wiring. Select the edge cuts layer (Edge.Cuts), and then click on the polygon button. I think that the current grid setting (1.27mm) is appropriate for drawing the edge cut.

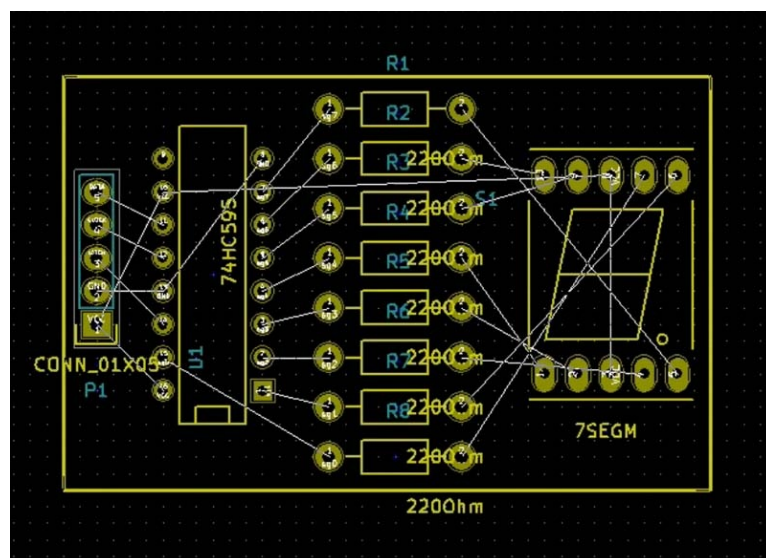


Ensure you are working in the Edge.Cuts layer. The select the polygon tool.

From the top right corner of the PCB, start drawing the edge cut.



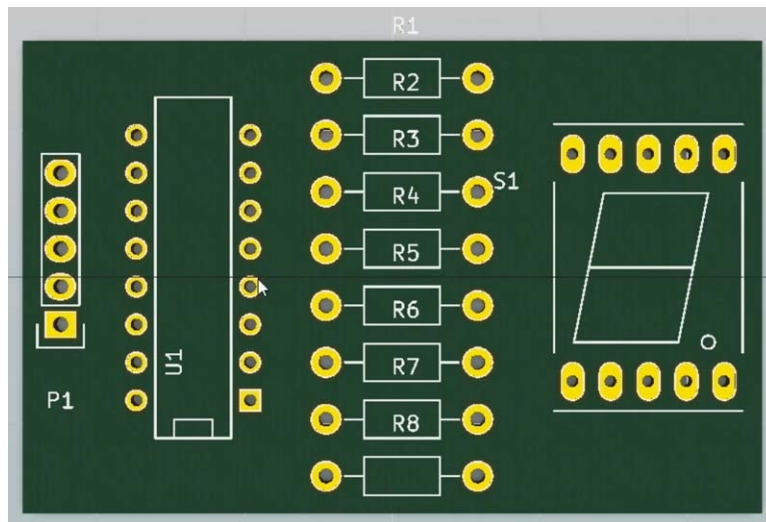
Start drawing the boundary of the PCB front the top right corner...



...And complete drawing where you started.

Because the price of a PCB depends greatly on its size, try to create a boundary that make it as small as possible. The positions of the footprints, of course, influence the way that the boundary wraps around them.

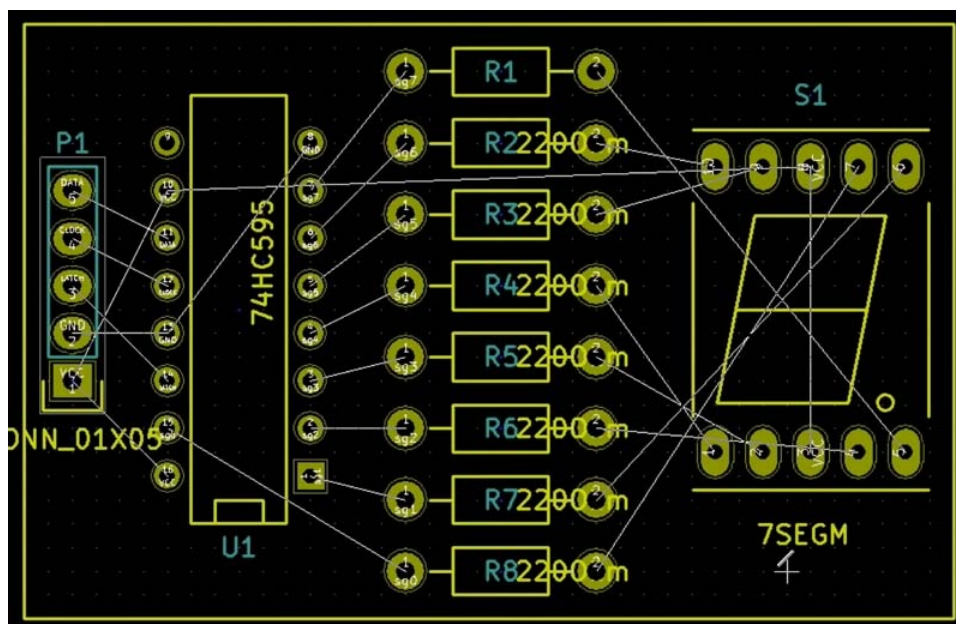
Start drawing the perimeter.



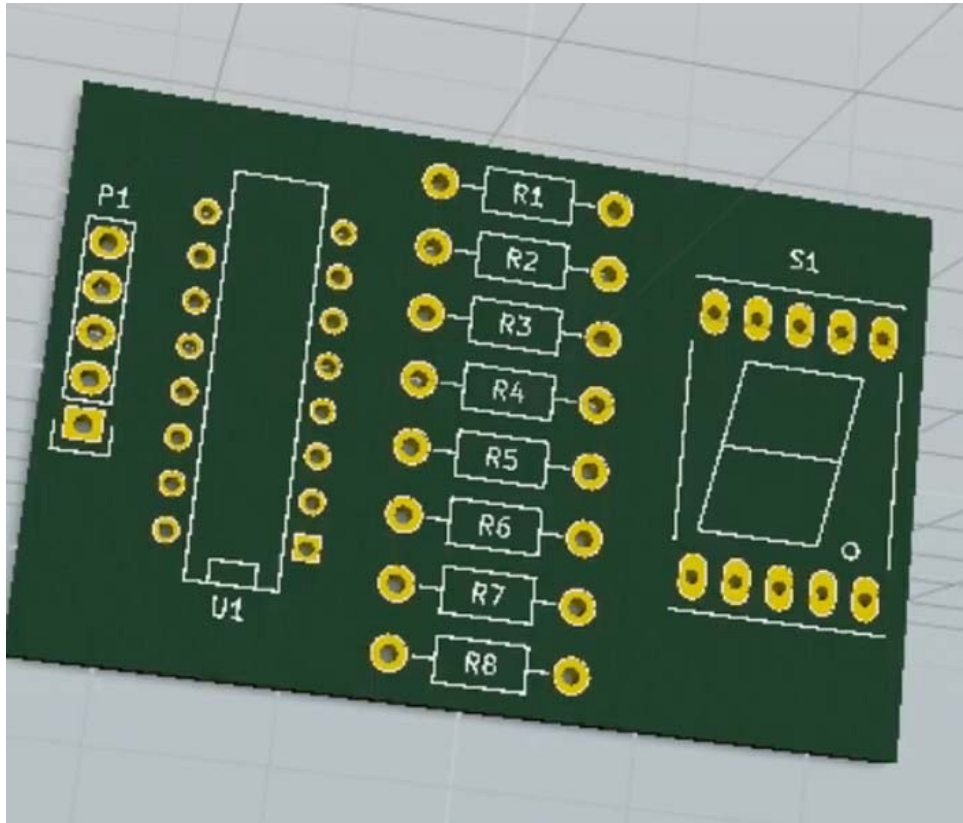
A 3D view rendering of the PCB, after the boundary is defined.

This looks fine so far. Before we continue with the wiring, I would like to correct the positions of the silkscreen labels, “R1”, “R2” etc. Let’s move them to appropriate positions inside the resistor boxes. Let’s also adjust the positions of the rest of the silkscreen labels so that they are better aligned with the footprints they represent.

Use the “M” key to move the labels around. By the end of this process, your PCB should look a bit like this:



The R1..R8, S1, U1 and P1 labels were moved (in blue).



The 3D rendering of the board shows the new label locations.

With the label positions adjusted, we can continue to the next step: the Wiring. In the next chapter I will show you how to do the wiring so that the track width is automatically set to the correct value based on the net name. Remember that in project 1, I showed you how to control the track width manually.

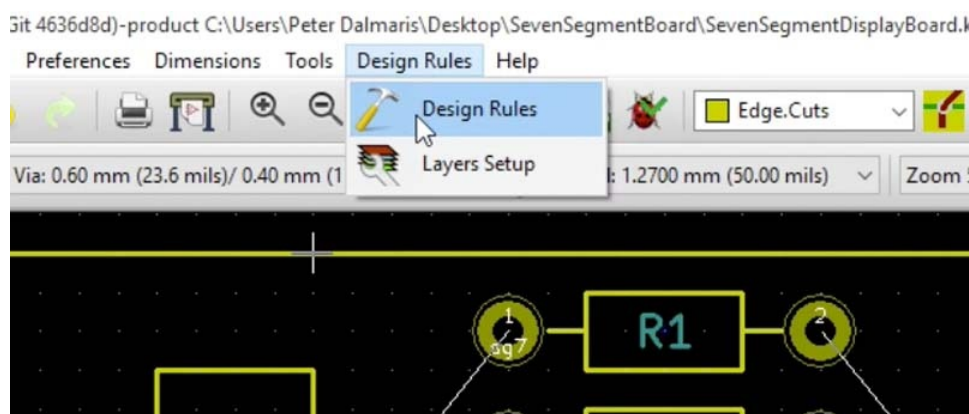
Chapter 44: *Control track widths with nets*

In this chapter I will show you how to wire the tracks on both top and bottom layer of the PCB, and have the width of each track set automatically by Kicad. To make this possible, we will define the desired width of Vcc and GND tracks, as well as the default track width (that is, the width for tracks that are neither Vcc or GND). Then, Kicad will set the width for a track based on this information and the name of the net that the track belongs too. If this sounds a bit complicated, rest assured it isn't; it will all make sense by the end of this chapter.

In the first project we controlled the width of each wire manually. We did this by creating a custom track width and then we used the track width drop-down menu to manually choose the width of a particular track. What I would like to do now is to get Kicad to automatically set the correct track width.

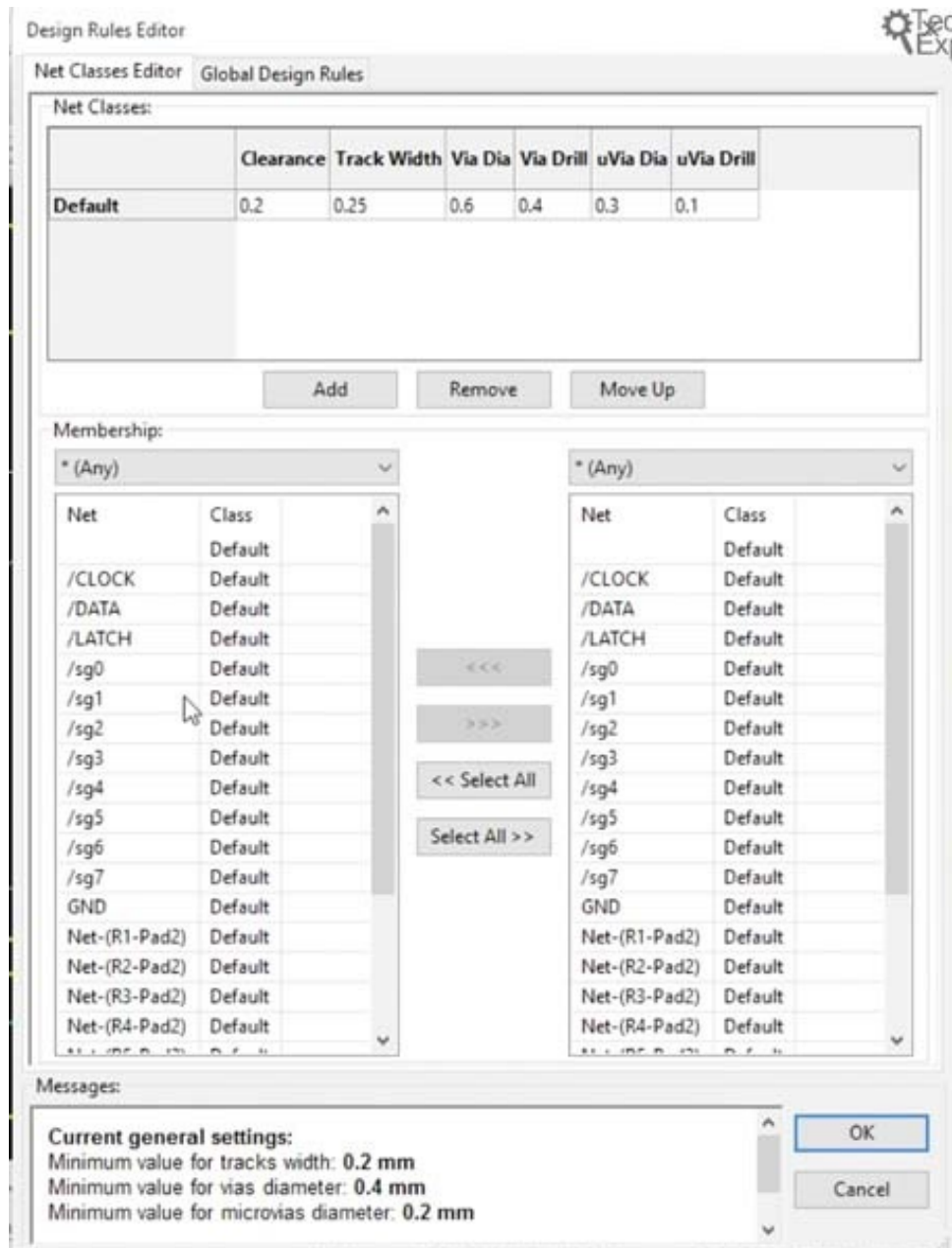
Here is how this can be done:

Start by bringing up the Design Rules editor from the Design Rules menu.



Go to the Design Rules editor.

You have seen the Design Rules editor in the past, when you created several custom track widths in the Global Design Rules Tab.



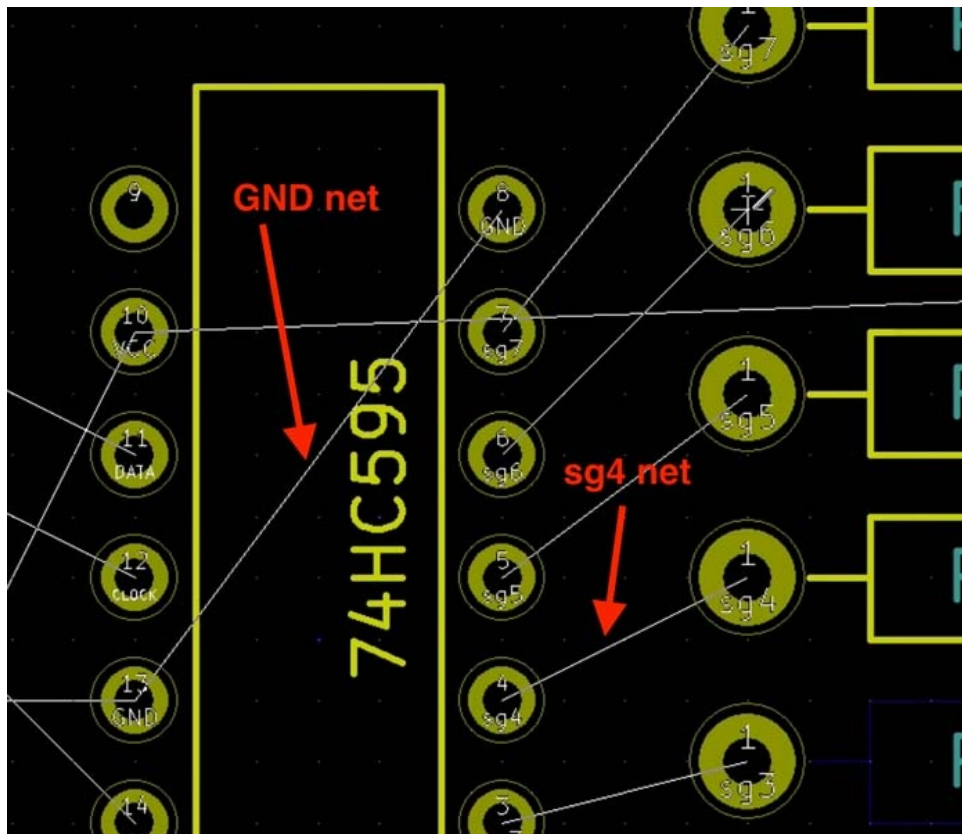
We will work in the Net Classes Editor tab.

In this chapter, we will work in the Net Classes Editor tab, instead of the Global Design Rules tab.

There's a few things to notice here. First, at the top of the editor there is the Net Classes matrix. There is a single net class, "Default", with the various settings for clearance, Track Width etc.

In the middle of the editor are the Memberships. On the left side, you can see that all the nets belong to the Default class. There is no other class to which any of these nets could have belonged to anyway. You can see the same information on the right side. At the top of each list is a drop down menu. This allows you to filter out nets of a particular class, and only display those in the list. With two lists, one next to the other, you can select nets of a desired class in each list, and then use the four buttons between the lists to move nets from one class to another.

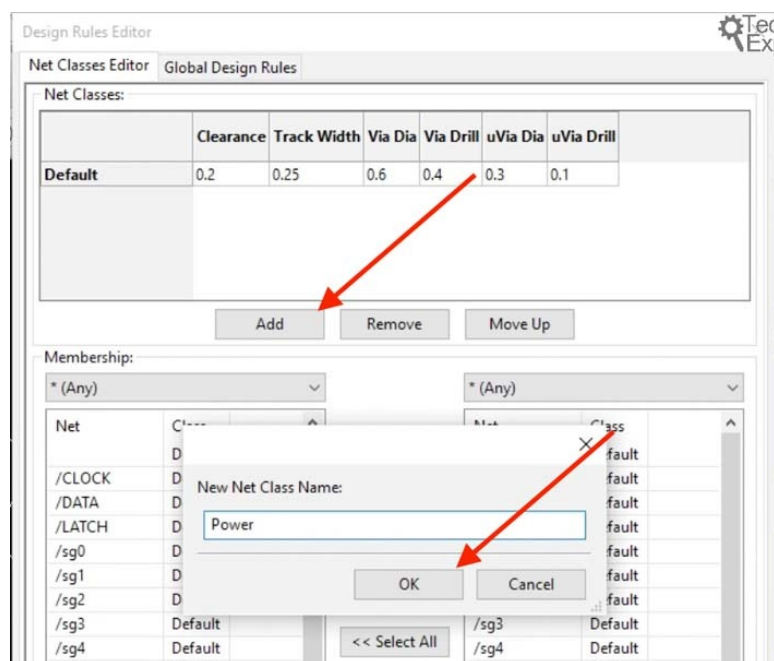
A net is a wire, and it gets its name from the names of the pads that it connects.



A net gets its name from the names of the pads it connects

In the example image above, you can see that the wire that connects any “GND” pads is named also “GND”. The wire that connects the two “sg4” pads is also named “sg4”. If you wish to change the name of a net, you must change the names of the pads that it connects.

To create a new Net Class so that tracks that belong to this class have an individual set of values for their various characteristics, start by clicking on the Add button in the Net Class Editor tab.



To add a new net class, click on Add, give the new class a name, and click OK.

Give the new class its name, Power. Then click OK. Now, edit the values for the Power class. I set those values as per the screenshot below:

Net Classes:						
	Clearance	Track Width	Via Dia	Via Drill	uVia Dia	uVia Drill
Default	0.2	0.25	0.6	0.4	0.3	0.1
Power	0.3	0.4	0.7	0.5	0.3	0.1

AddRemoveMove Up

The values for the new Power net class.

Great, we have a new net class, for Power. We would now like to assign nets to this class, so that the power class characteristics are inherited by these member nets.

Membership:

* (Any)

Net	Class
	Default
/CLOCK	Default
/DATA	Default
/LATCH	Default
/sg0	Default
/sg1	Default
/sg2	Default
/sg3	Default
/sg4	Default
/sg5	Default
/sg6	Default
/sg7	Default
Net-(R1-Pad2)	Default
Net-(R2-Pad2)	Default
Net-(R3-Pad2)	Default
Net-(R4-Pad2)	Default
Net-(R5-Pad2)	Default

Power

Net	Class
GND	Power

<<<

>>>

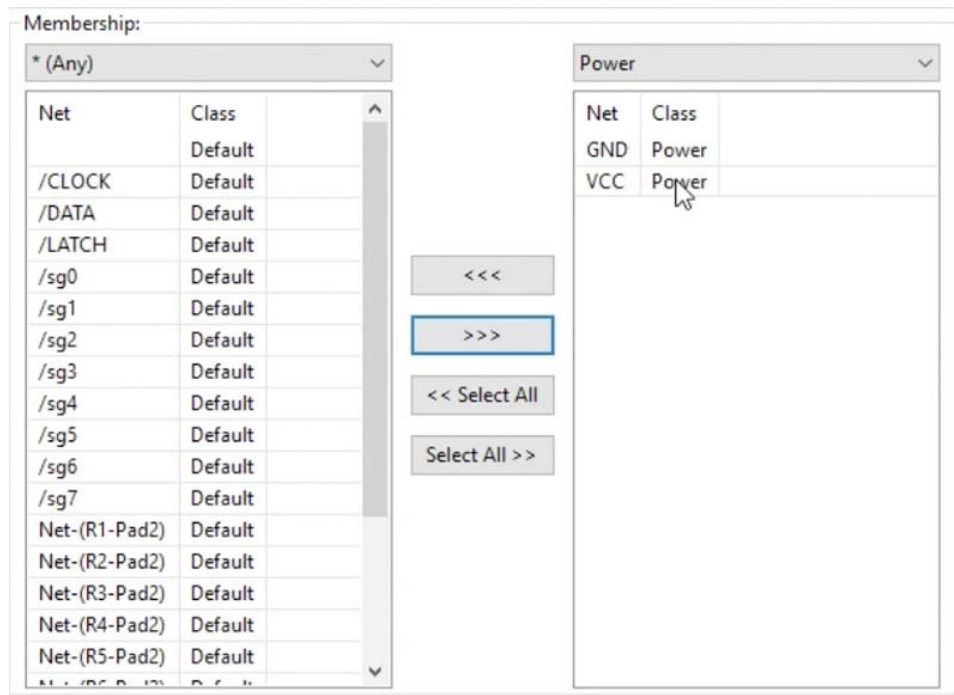
<< Select All

Select All >>

Make the GND net a member of the Power net class.

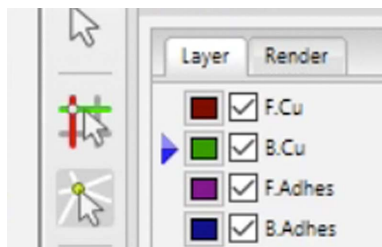
To do this, first use the drop down filter menu of the right net list to select the Power class. Then, select the GND net from the left list of nets. With the GND net selected, click on the “>>>” button to move the GND to the right side list. That’s it, the GND net is now a member of the Power class.

Repeat the same process for the Vcc net, and what you will end up with is this:



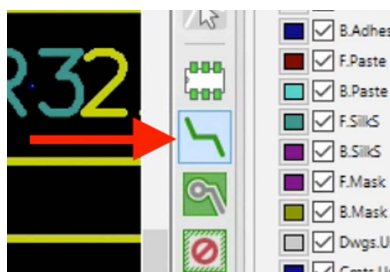
The Vcc net is also a member of the Power net.

At this point, we have the GND and Vcc nets as members of the Power net class. Let's see the effect that this has on routing tracks. Let's connect a couple of wires, starting with the GND pads. Because I prefer to place GND tracks in the back copper layer, select the B.Cu layer first.

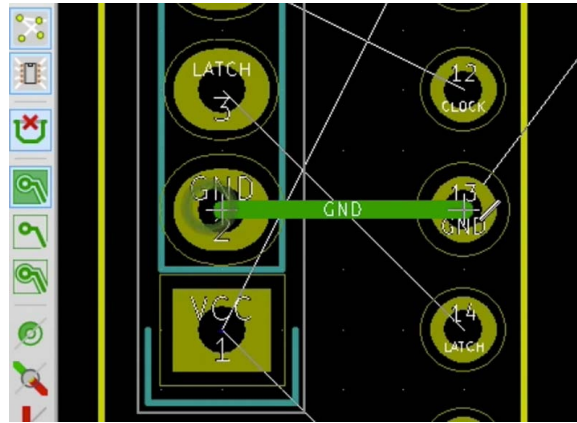


We will place the GND track in the back copper layer.

Then click on the green wire button to select it, and connect any two GND pads.



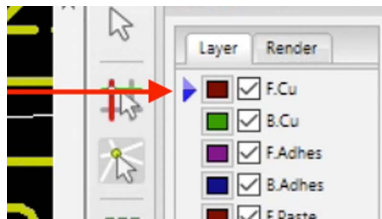
Click on the green wire button to go into the wiring mode.



Wire the connector and 595 IC GND pins. Notice the thickness of the track?

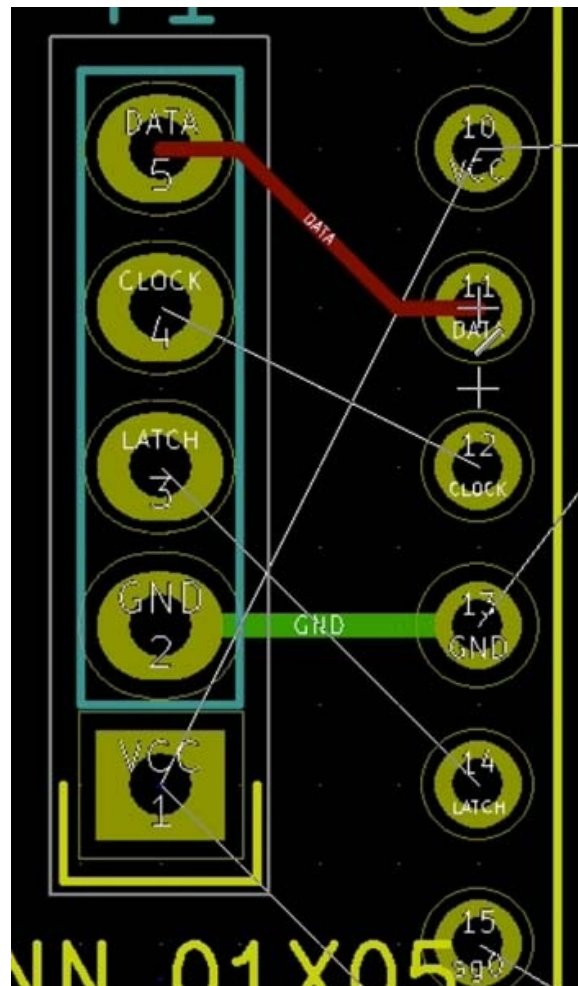
I connected the GND pins between the connector and the left 595 IC. Notice that the new track is thicker than the default thickness. This is an indication that KiCad applied the correct track thickness value best on the membership of the GND net.

Let's do the wiring for a non-power related set of pins. Let's connect pin 5 of the connector to pin 11 of the 595 IC. Since this is a normal data signal, not GND, we will switch to the front copper layer.



Select the F.Cu layer in order to do wirings in the front copper layer.

Then do the connection between these pads. This is what you should see:



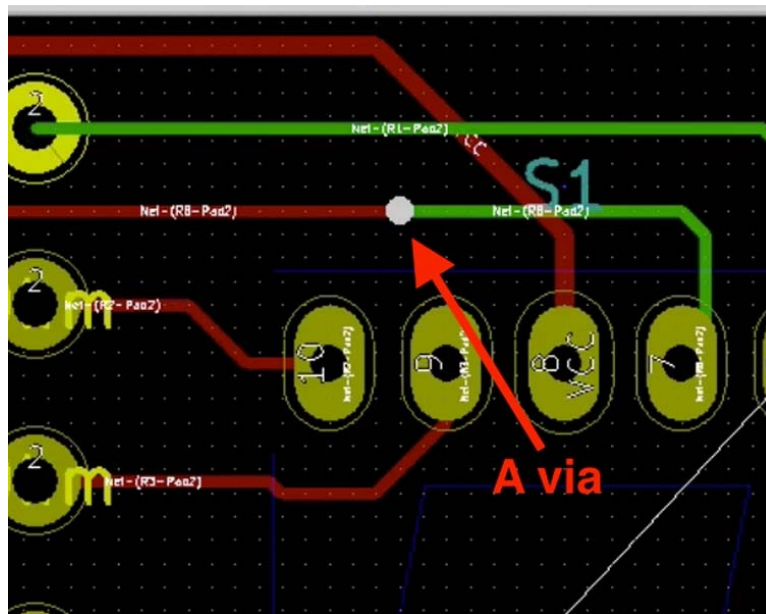
The red track connects two non-power pads with a net that does not belong to the power net class. It is thinner than the green track because Kicad applied the default track settings to it.

Notice the difference between the thickness of the green and the red tracks. The red track (indicating it is placed on the front of the PCB) is thinner than the green one. This is because the Data net (the net that connects pin 5 of the connector to pin 11 of the IC) belongs to the default net class. This class has a thinner value set for the thickness. Again, we didn't have to do anything specific to apply the thickness value. Kicad used the net class settings to work out the correct thickness for each net.

Let's continue with the rest of the wirings. I prefer to do my wiring from left to right when possible. Spend a bit of time experimenting with the routing, try to figure out the shortest path from one pad to another, and try to make the tracks as neat as possible. Try to lay the tracks in parallel paths, with adequate space between them. Do a Design Rules Check occasionally to make sure the design is valid at all times. Usually wiring the back copper layer is easier since it involves mostly the GND pads. Then move to the front layer.

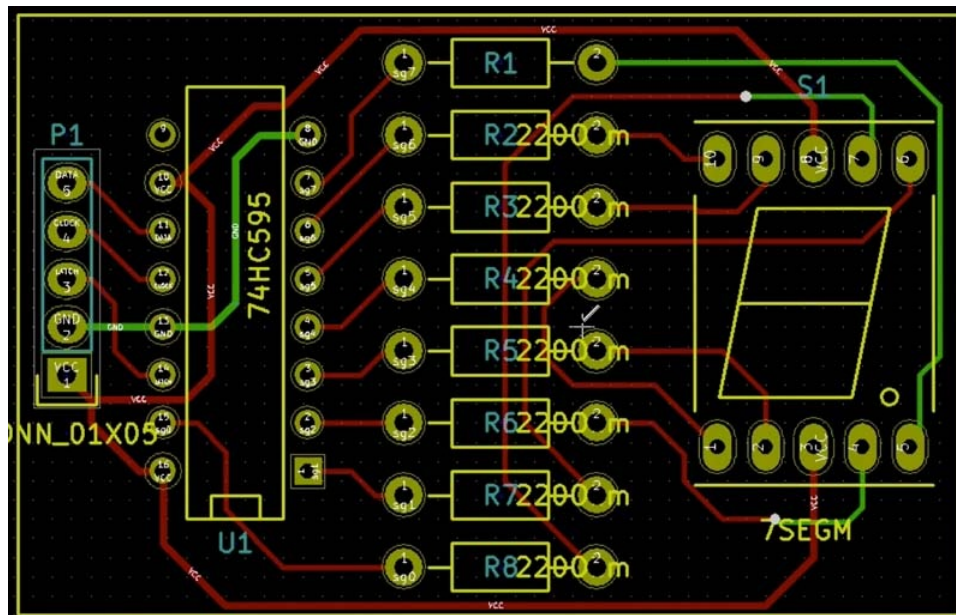
In some cases, you may not be able to find a route between two pads because of obstacles, like existing tracks or pads. You can try to re-arrange existing tracks by deleting them and re-drawing. You may also be able to create a route by inserting vias. A via is a small hole that implements an electrical connection between layers. This way, a track that starts in the front layer can continue in the back layer, and then come back to the front to finish a connection. To insert a via, just type "V". This will create a via representation on

the canvas that looks like this:



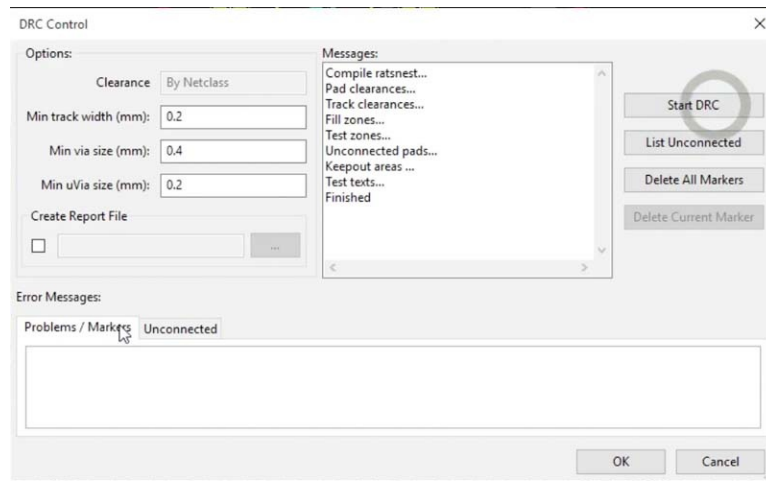
A via is a connection between layers that allows tracks to move from one layer to another. In this example, a via is used to make allow for a front layer track to continue to the back layer.

My track routes eventually look like in the image below. Yours may be different.



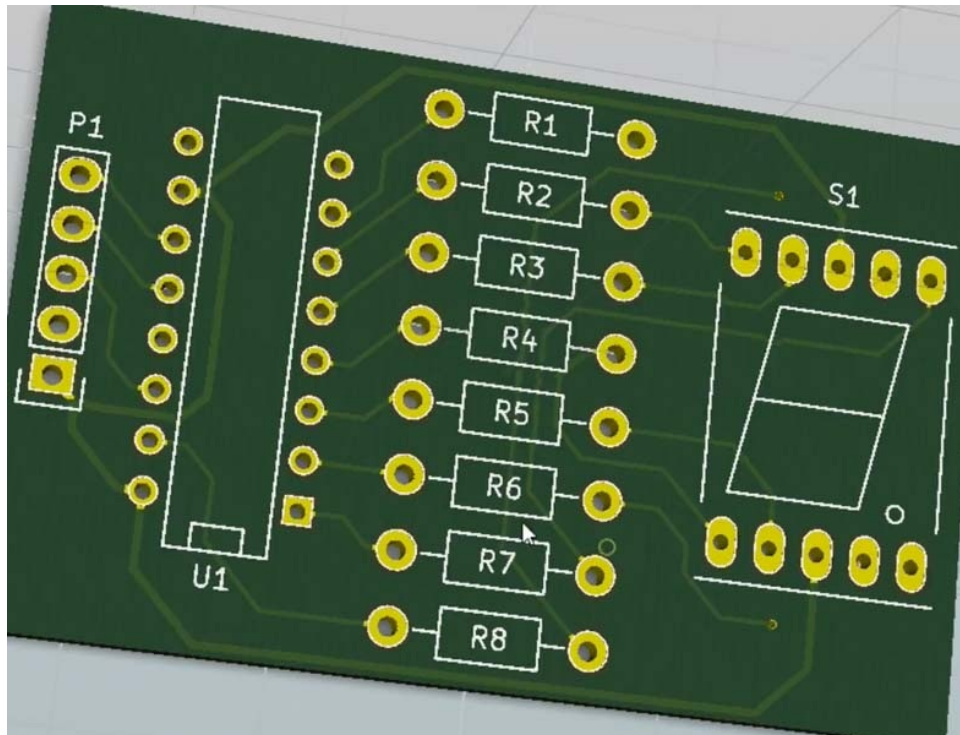
A possible wiring. Your actual routes may be different.

The DRC is also clear:



A clear DRC report!

You can also have a look at the 3D render of your board, and admire the tracks you just created. Look at the back layer to notice how the thicker GND net is clearly visible.



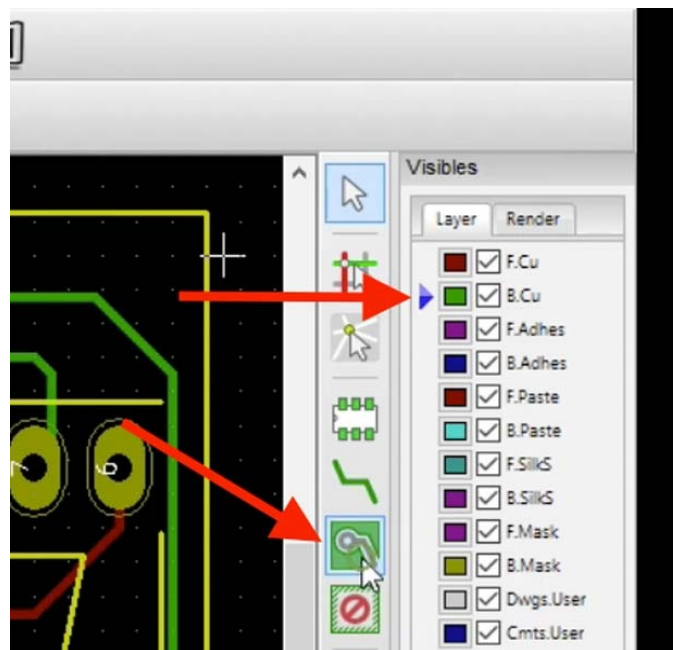
The track are clearly visible in the 3D rendering of the board.

Let's save the project. In the next chapter, we will add copper fills for Vcc and GND.

Chapter 45: Add GND and Vcc copper fills

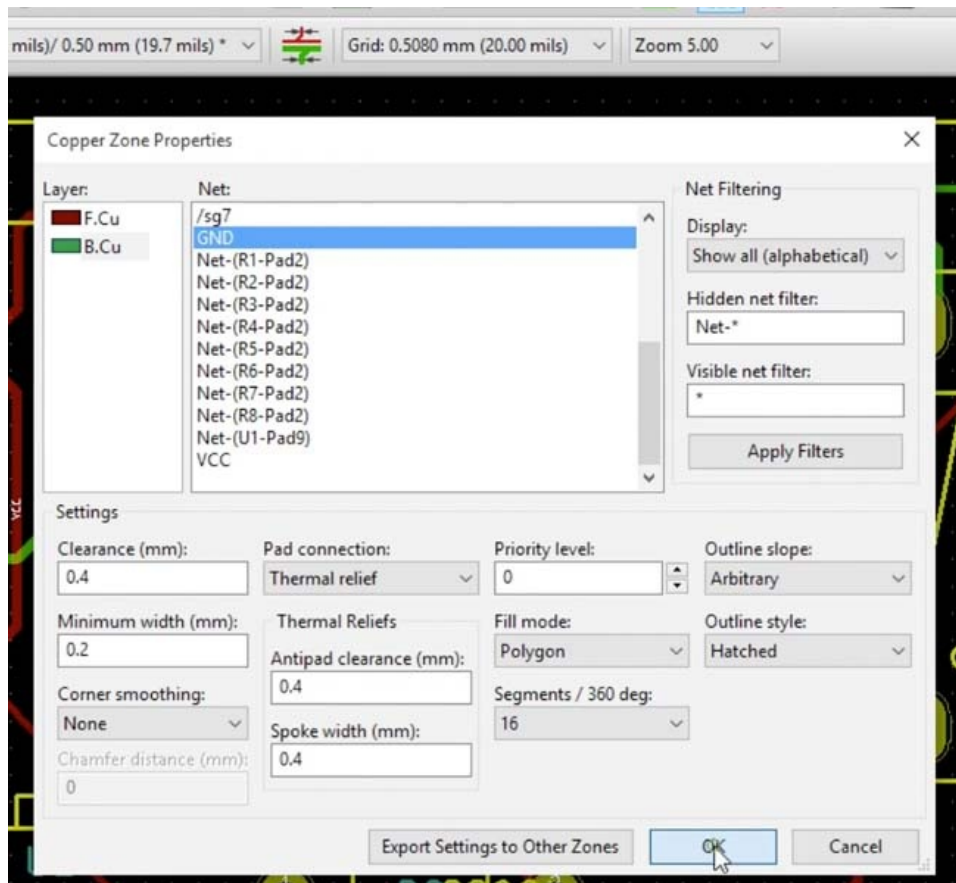
In this chapter, we will add the copper fills for the ground and Vcc planes. Since we placed the ground tracks in the back copper layer, and the Vcc and data signals on the front copper layer, we will create copper fills in the back and front layers.

Let's start with the bottom layer. Select the bottom layer "B.Cu" from the Layer chooser.



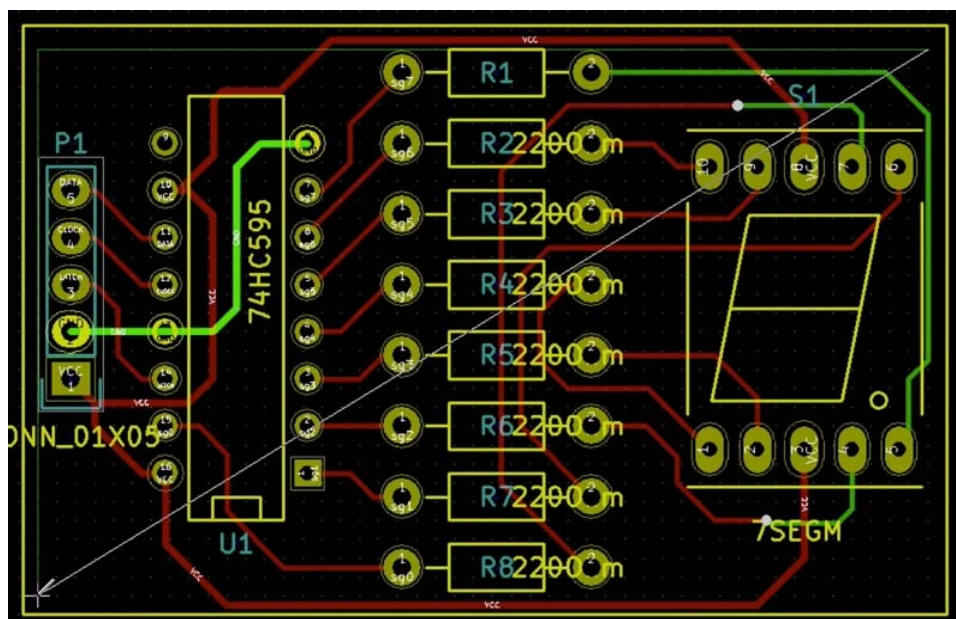
Select the appropriate layer, then enable the copper fills function button.

You may get a warning like this:

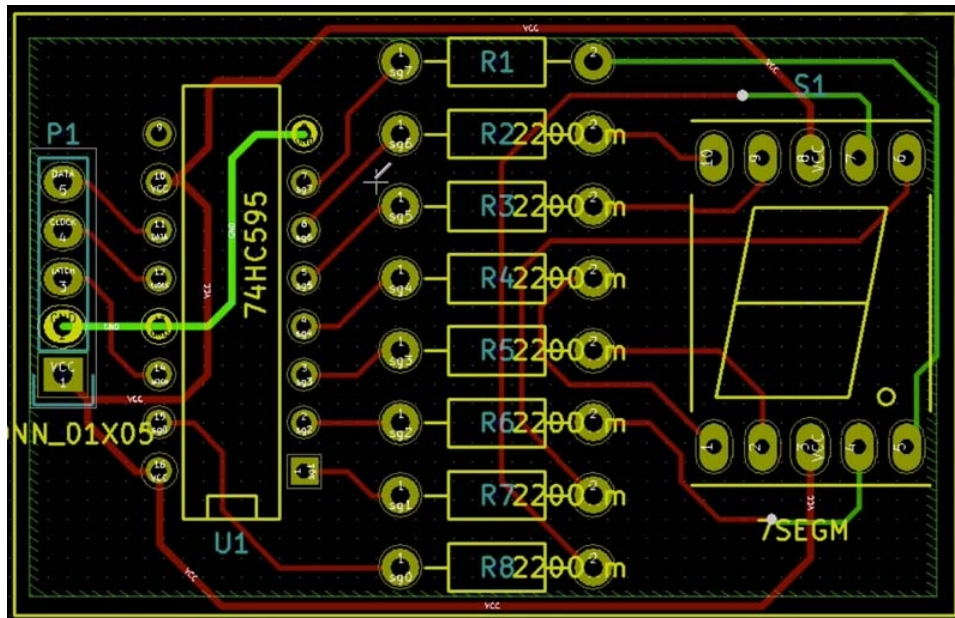


To add a copper fill in the bottom layer for the GND net, select the layer and net from the Copper Zone Properties window. Also notice the Grid setting (0.508) in the drop-down menu above the properties window.

Click OK to dismiss the properties window, and start creating the box. A single click will complete an edge, and a double click will finish the box.

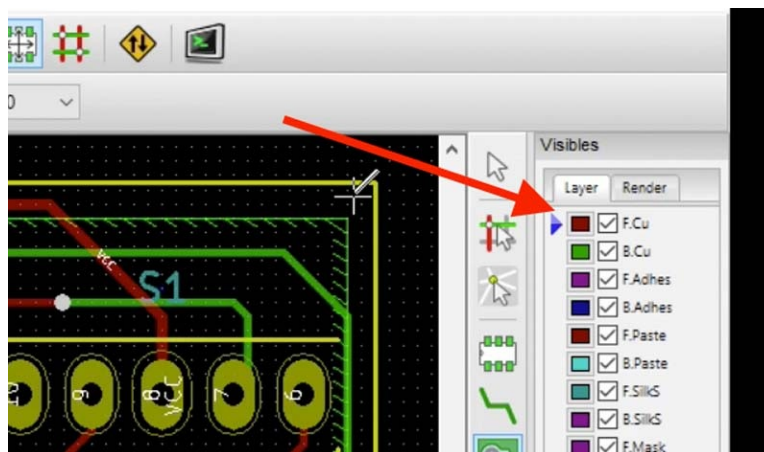


Click to create an edge of the box...



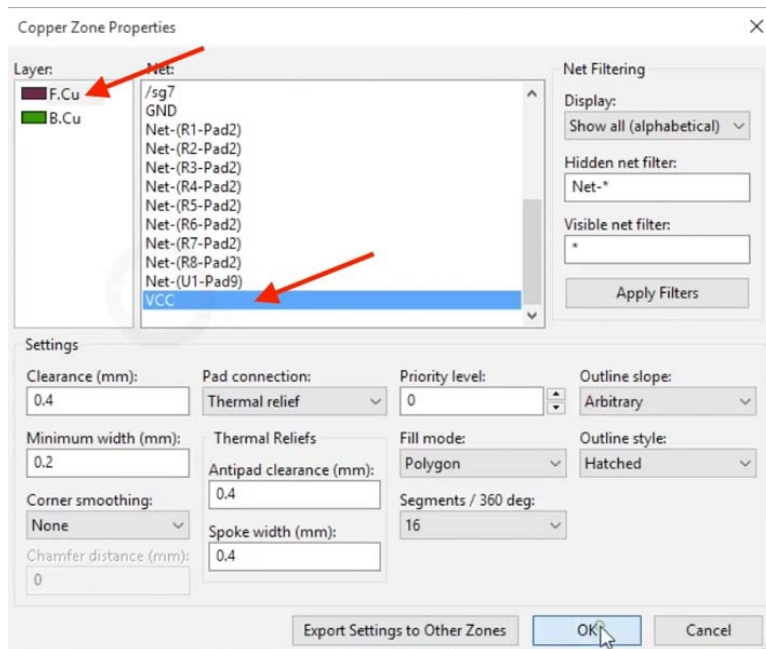
Double-click at the first corner of the box to complete the fill.

The bottom layer copper fill (marked in green) is ready. Let's move to the front layer and create the copper fill there. Choose the front copper layer (F.Cu):



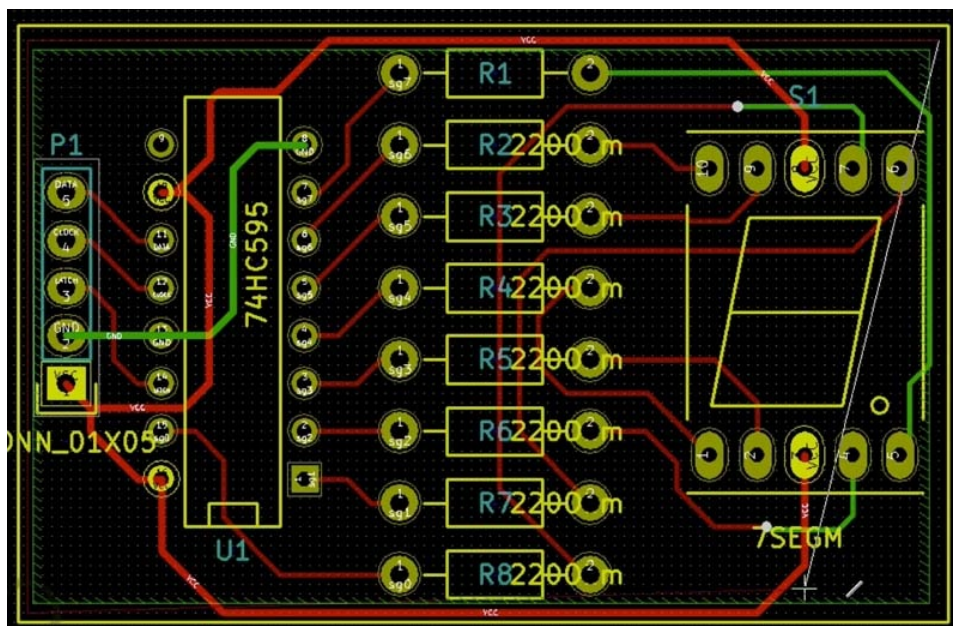
Move to the front copper layer.

The fill zones tool is already selected, so we can go ahead to start the box for the copper fill in the front copper layer. Again, start by clicking on the top right corner of the PCB (this is not a requirement, only my preference). The properties window will come up:

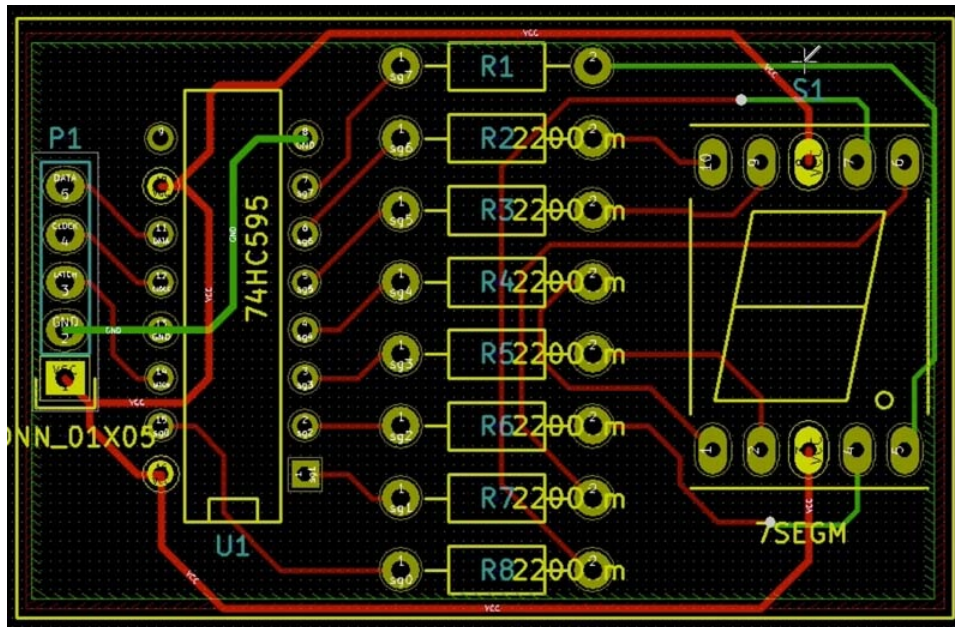


F.Cu will be already selected. Choose the VCC net, and click OK.

Since we are already working on the front layer, the F.Cu layer will be already selected in the Copper Zone Properties. In the Net list box, scroll to find the VCC net, and selected. Click on Ok to dismiss the properties box. Begin drawing the box close to the edge of the PCB:

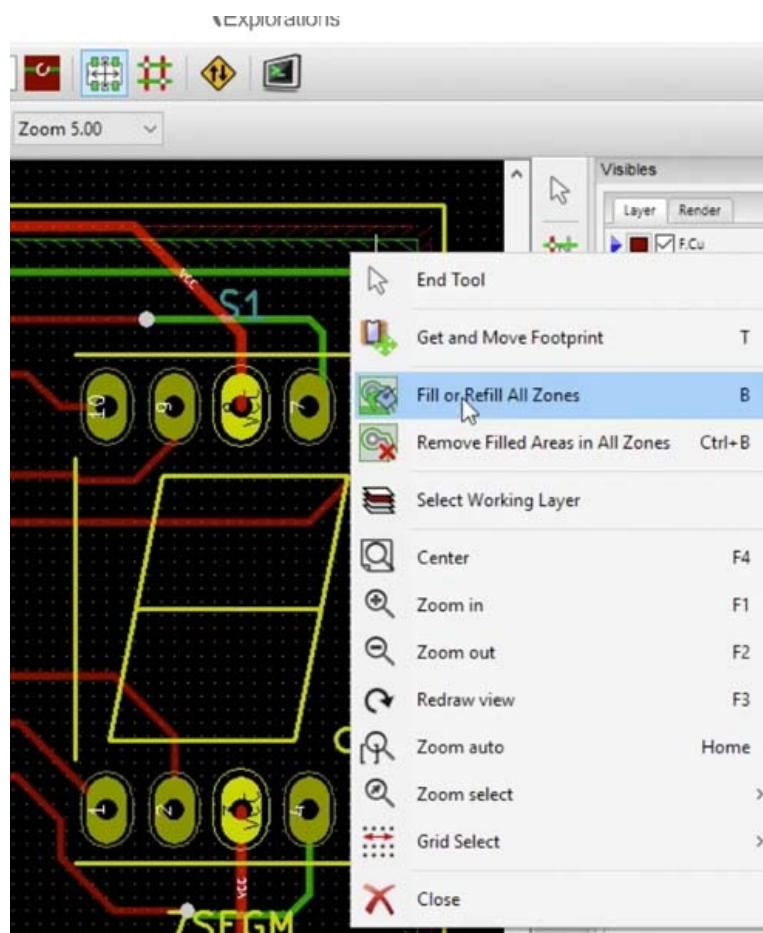


Drawing the front layer copper fill...

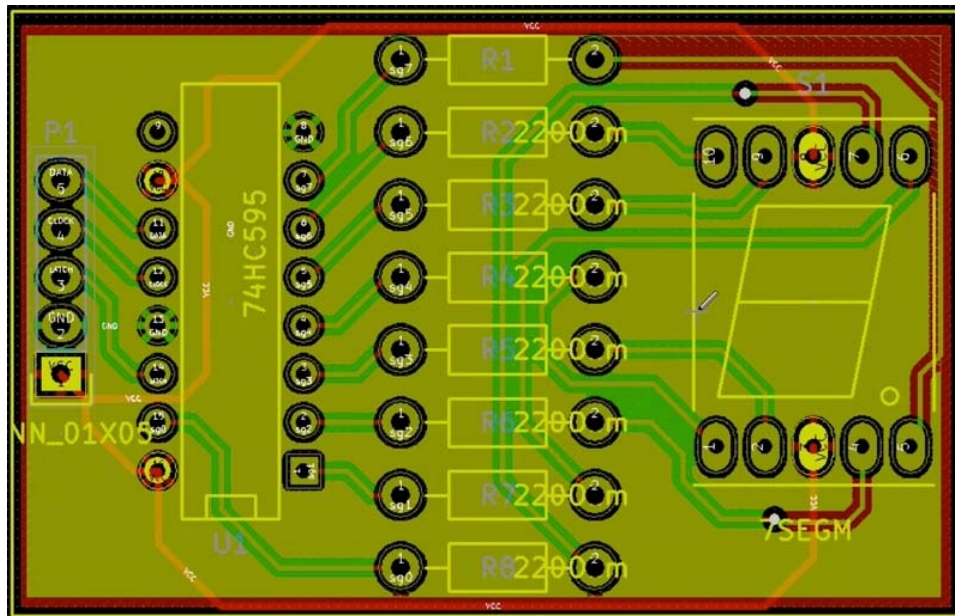


The front layer copper fill box is complete. Notice the thin red line outlining it.

We now have the outline for the top and bottom layer fill zones, but they zones are still empty. To actually fill them with copper (at this point, virtual copper!), right-click anywhere inside the fill zones and click on “Fill or Refill All Zones”) from the contextual menu:

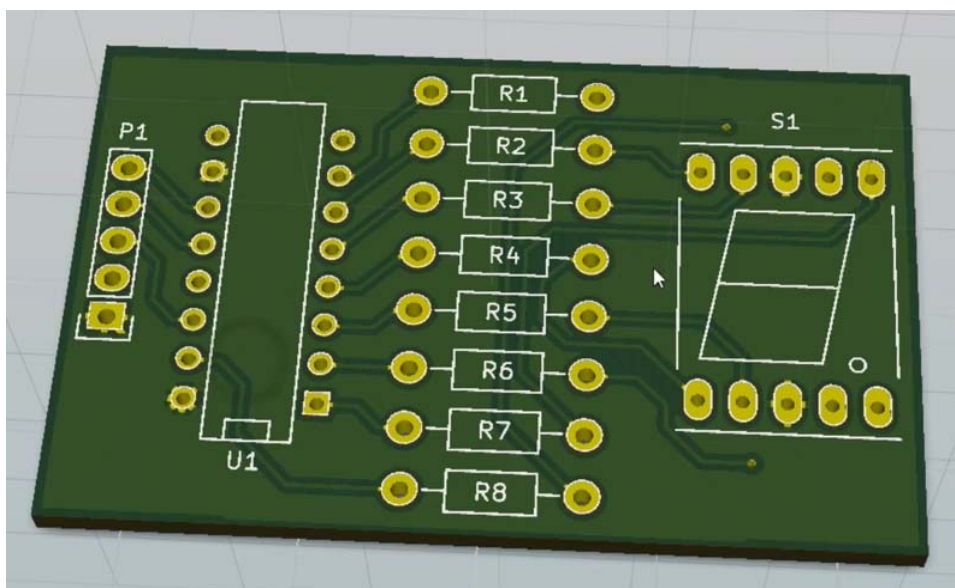


To fill all zones, right click anywhere inside the zone and select “Fill or Refill All Zones”.

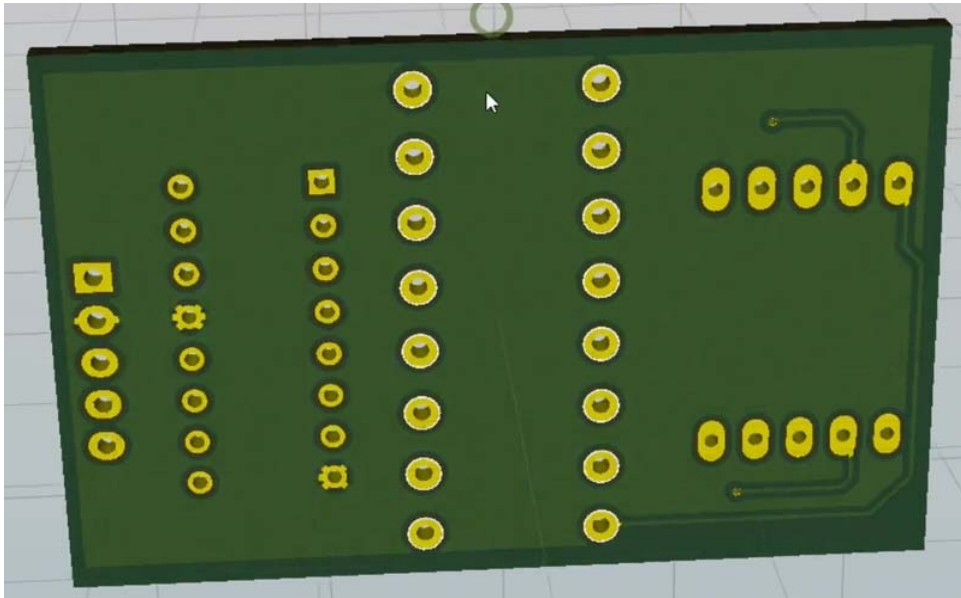


Our PCB, with its fill zones filled with virtual copper.

Our board is now filled with copper in its fill zones: red for the front, and green for the back. The 3D rendering shows the copper filled area with a lighted shade of green when compared with areas that have no copper:



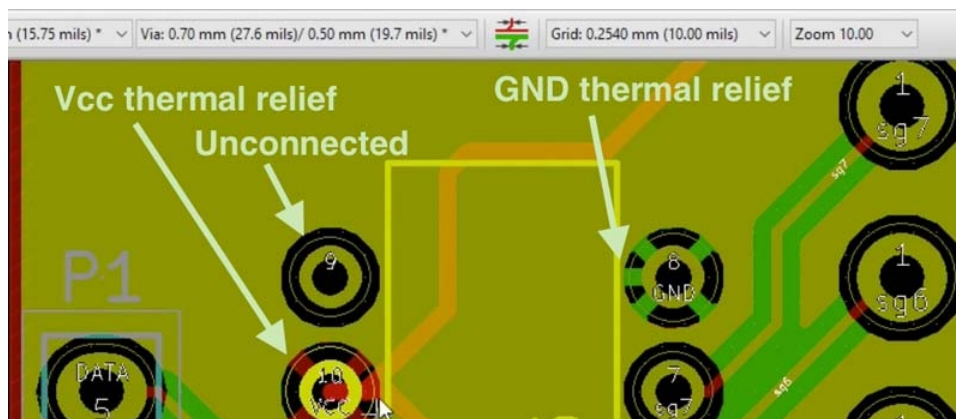
A 3D rendering of the front layer. Notice that the areas filled with copper have the same shade of green as a normal track. Darker green indicates an area that has no copper.



The back layer, almost completely covered with copper.

I like to preview a 3D rendering of the front and back layers before doing any more work, in order to visually inspect the board. It is also a good practice to do regular DRCs. In my case, there were no errors, which is how I like it!

Before we move on, have a look at one of the Vcc pins, like pin 10, GND pin 8, and unconnected pin 9:



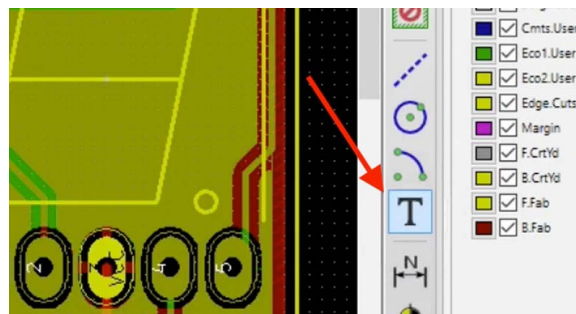
Small but important details: pins 8, 9 and 10.

Notice how pin 9, which in Eeschema we marked as “unconnected” is actually not connected to anything in Pcbnew. Thermal reliefs were also added to pins 8 and 10, GND and Vcc respectively, since we chose to connect these nets to the back and front layer copper fills.

The bulk of the layout and wiring work is complete. In the next lecture we will add text labels in the silkscreen layer.

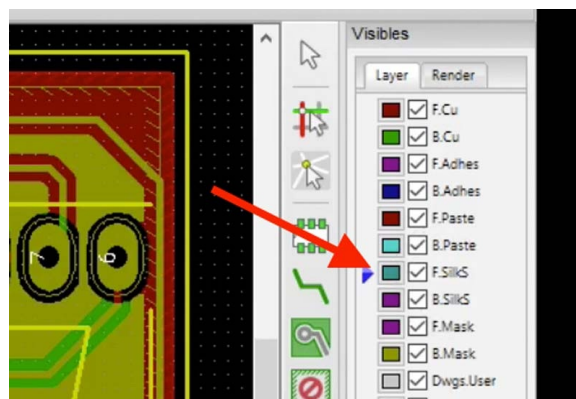
Chapter 46: *Add text labels*

In this chapter we will add a few text labels in the silkscreen layer. Text labels provide a convenient way to provide useful information to the end user of the PCB, like the name or model and version number. Select the Text tool from the right vertical tool bar:



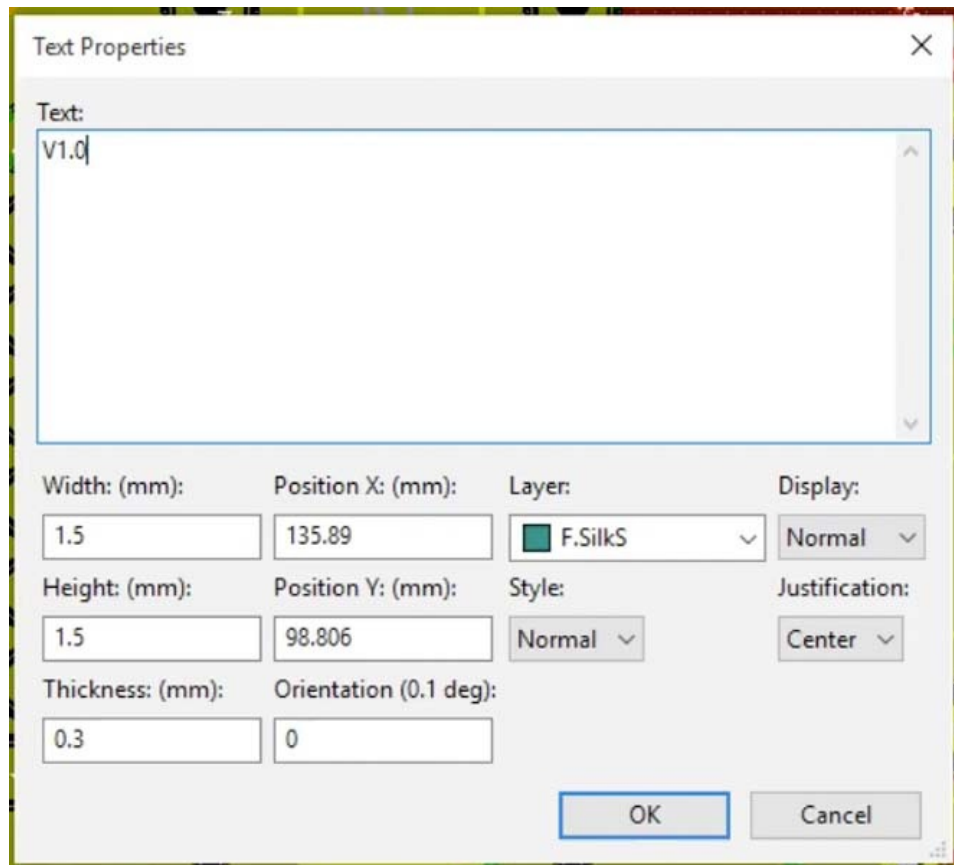
The text tool button

Select the front silk screen layer, “F.Silk”, where we would like the text to go:



We will place the text labels in the front silk screen layer. You can also place text labels in the back silk screen layer.

Start with the version number. There is a bit of empty space in the bottom left corner of the board, so click there. The Text Properties window will come up. Type “v1.0” in the text box:



The Text Properties window

The default settings for the text label are appropriate for this text. Notice that the Layer setting is “F.SilkS”, which is carried over from our earlier layer selection. Click OK, and then move the mouse so that the level (which is now tethered to the mouse pointer) is positioned close to the bottom left corner of the board. Click to place the label in position, like this:

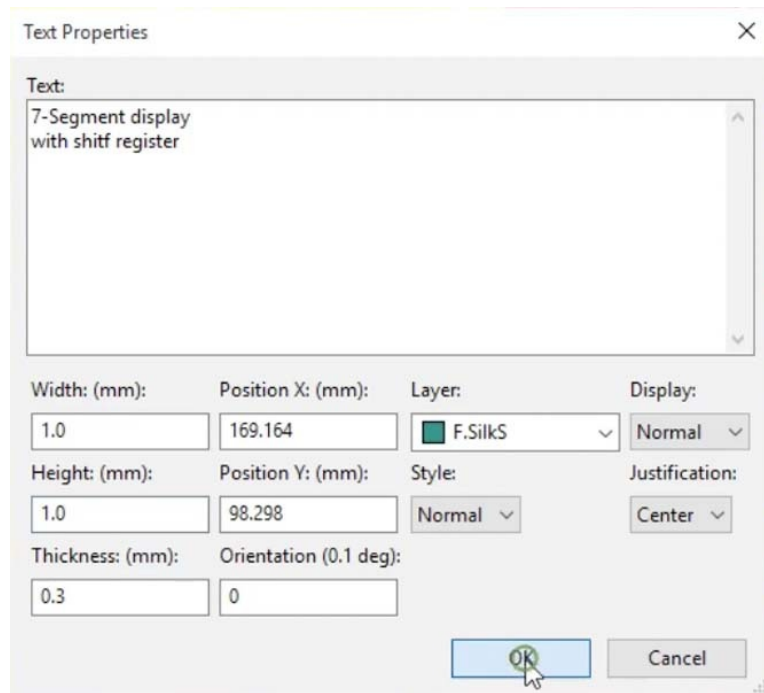


The “V1.0” text label in place.

In the screenshot above, you can see that the text (in light-gray) looks “unclean” and there is a ghost image in red. This is an artefact the KiCad generates as the label is moved around. You can force a redraw by pressing F3. This will clean up any such artefacts.

Create another label with information about the board. Something like this will be

good:



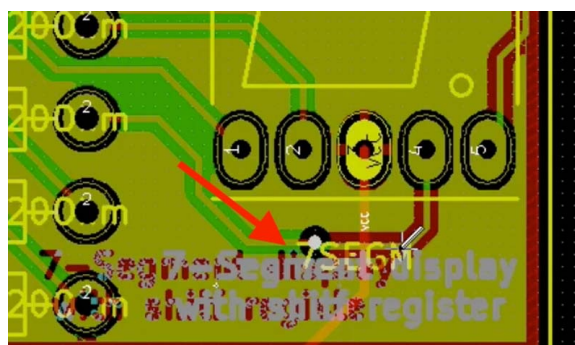
A new label. The width and height settings are smaller in order to produce smaller text.

I would like this text to be smaller, so set the width and height values to 1mm. When you click OK you will see a warning. This is telling you the Kicad needs to adjust the thickness of the text to accomodate for the smaller text size. This is fine, so click OK again to dismiss the warning. Use your mouse to fine-tune the position of the label in the bottom right corner of the board, and click to lock it in place. The label will look like this:



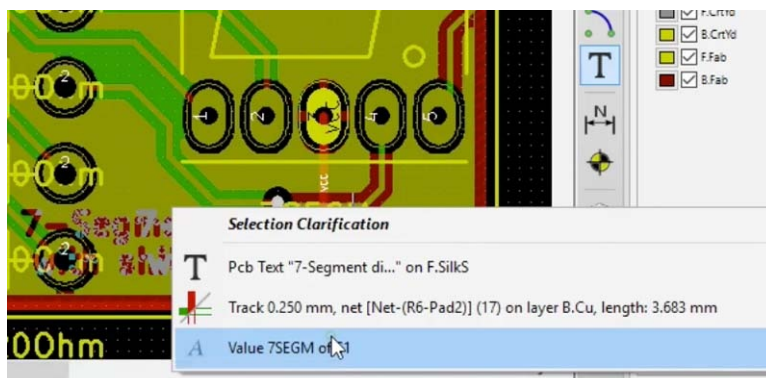
Another label with drawing artefacts. Type F3 to redraw and clean it.

While looking at the bottom right corner of the board, notice an existing label, "7SEGM" that is part of the display footprint:

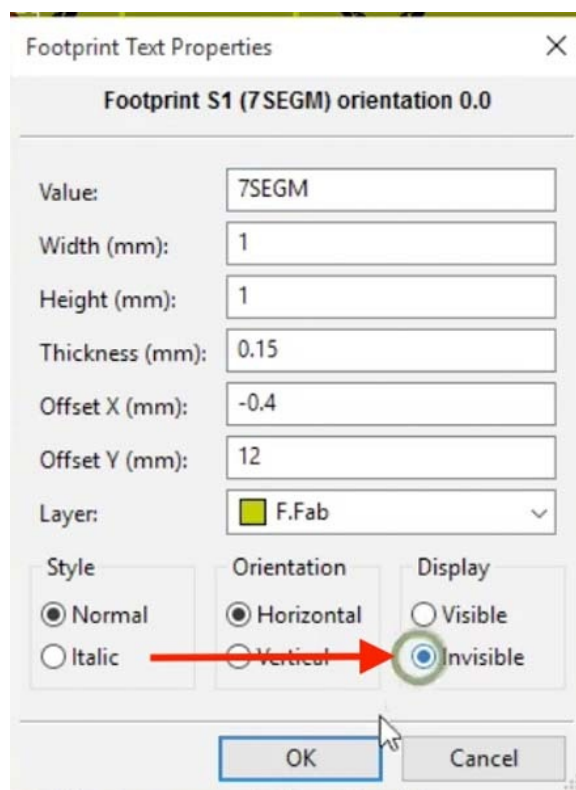


The label "7SEGM" is not useful. Let's make it invisible.

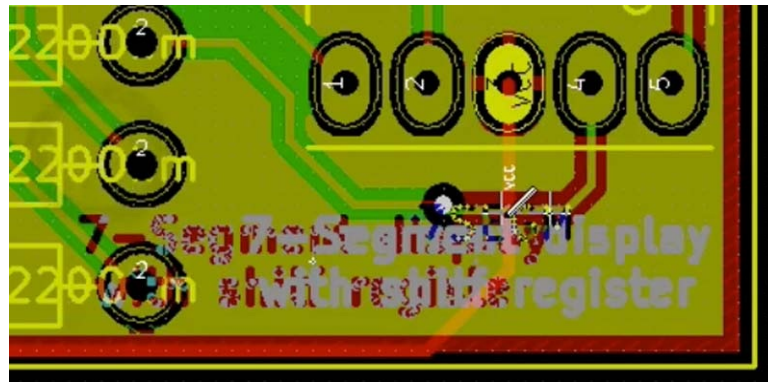
This label is not very useful, so I would prefer to make it invisible. Place your mouse pointer over the label and type “E” (for edit). A small menu may appear if more than one items are sharing the same space, asking you to indicate which item you would like to select.



If multiple items are sharing the same space, a menu will appear to help you select one of them.

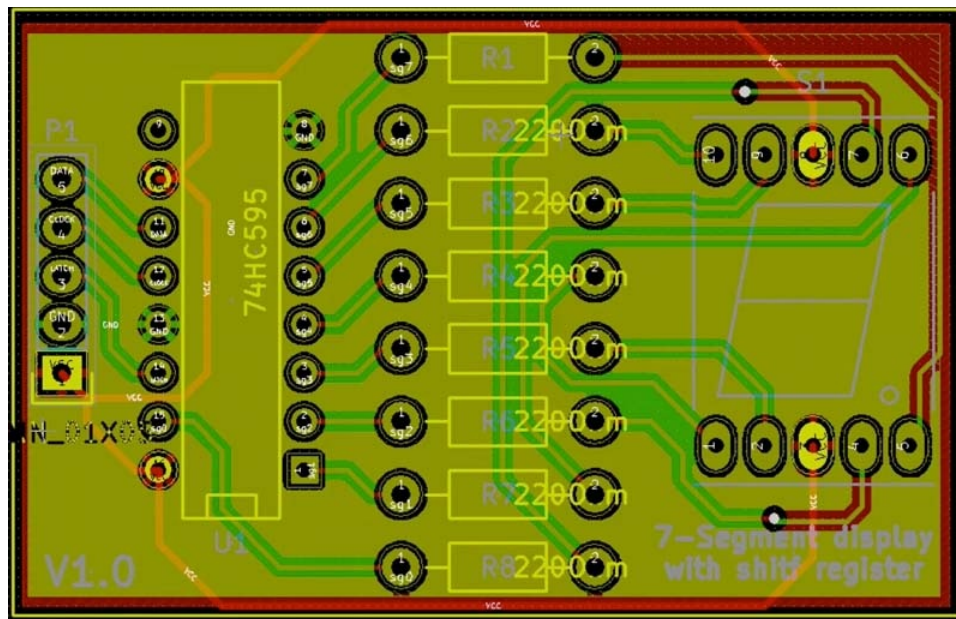


You can make a label invisible by selecting the “Invisible radio button”.



The text label has disappeared.

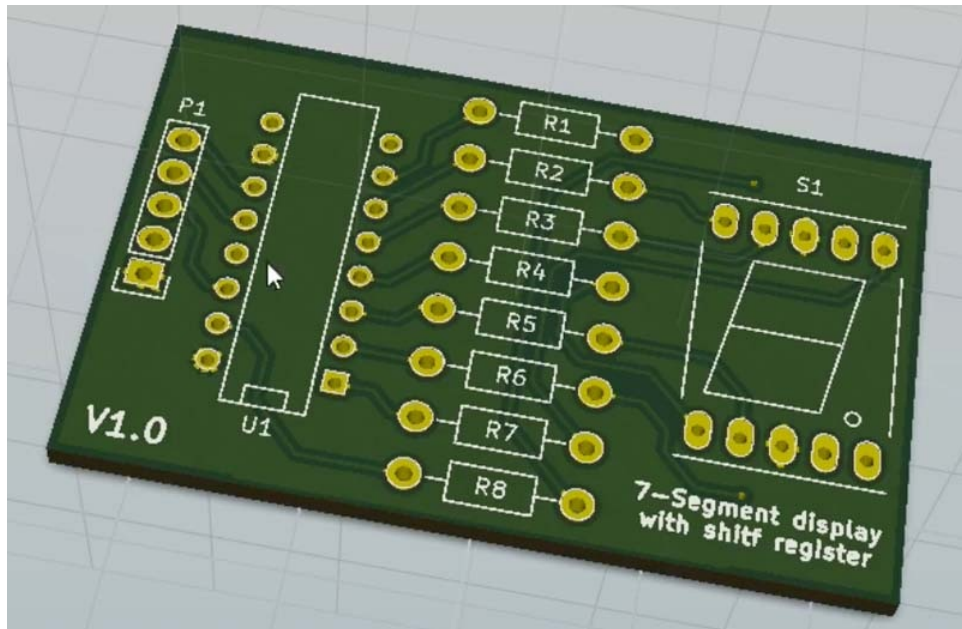
Our board, after a re-draw, now looks like this:



The board after a redraw (F3).

There are several yellow-coloured labels. These labels are located in the F.Fab layer, which we will not be sending to the fabricator anyway. Therefore, these labels will not be present in the final printed circuit boards, even though they are visible in this view.

The 3D preview of our board is this:



The 3D rendering of our board, with the new text labels in the silkscreen.

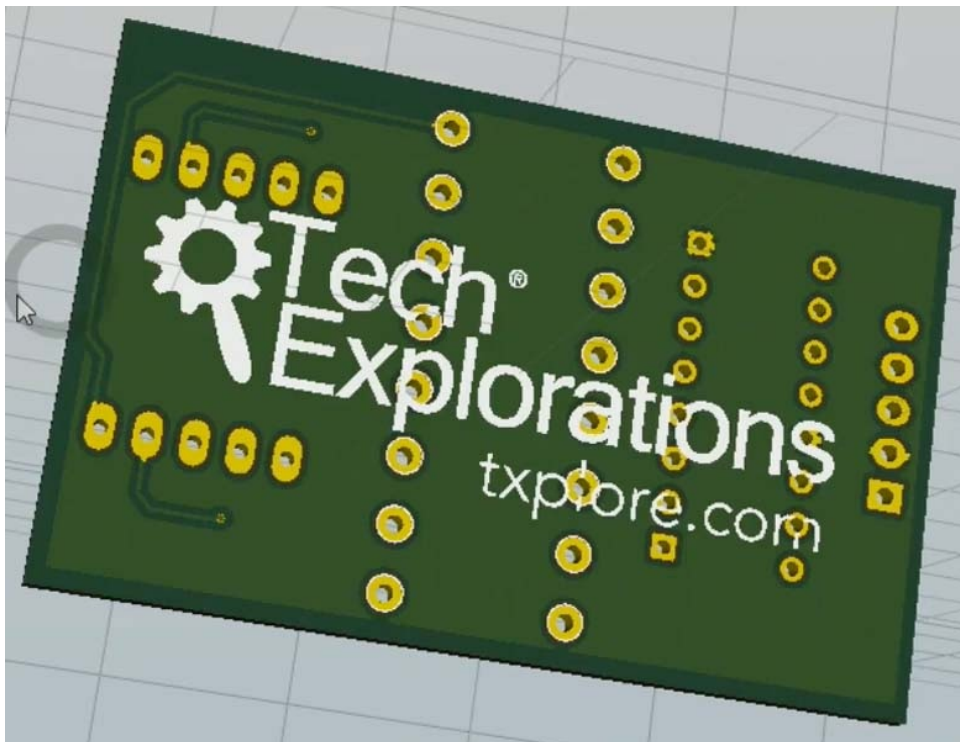
I think our board looks good! Save the project.

In the next chapter we'll create a decorative graphic to put in the back of the PCB with a logo.

Chapter 47: *Add a decorative graphic*

In this chapter I will show you how to add a decorative graphic on the back layer of our PCB.

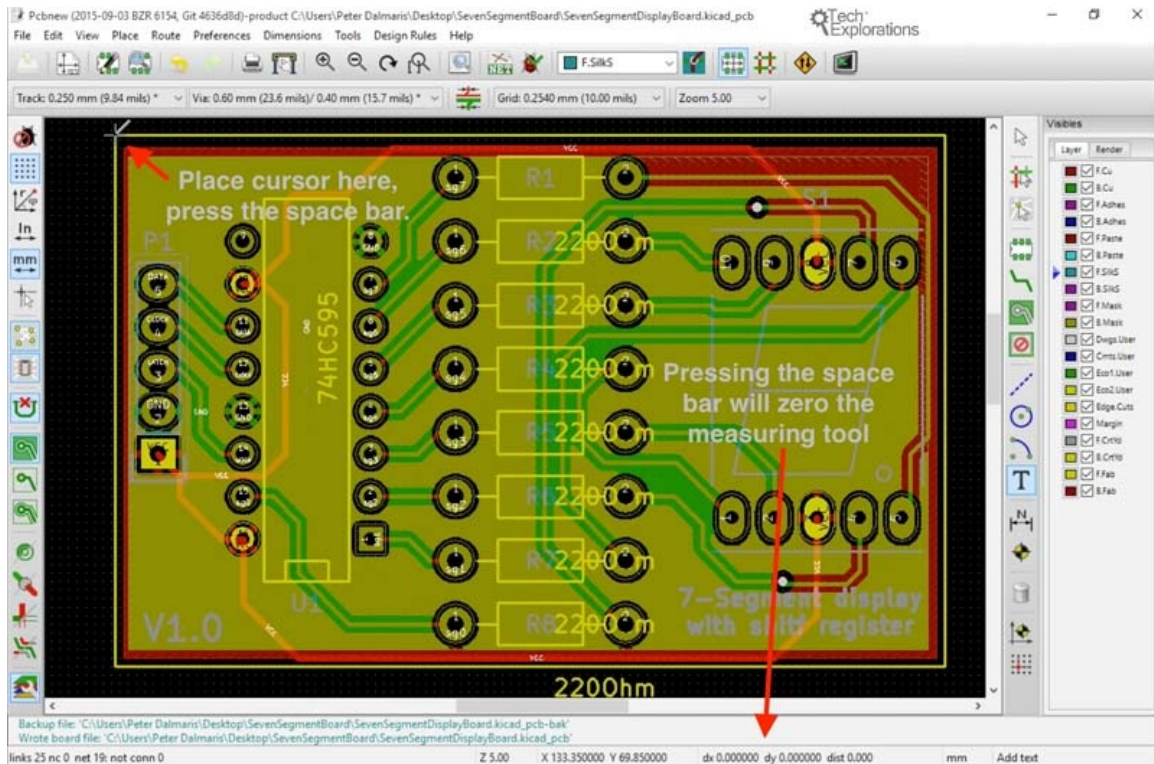
To better understand what we will achieve in this chapter, let's start from the end. I would like to add a graphic in the back silkscreen of the board, so that it will look like this:



In the end of this chapter, the back layer of our PCB will look like this.

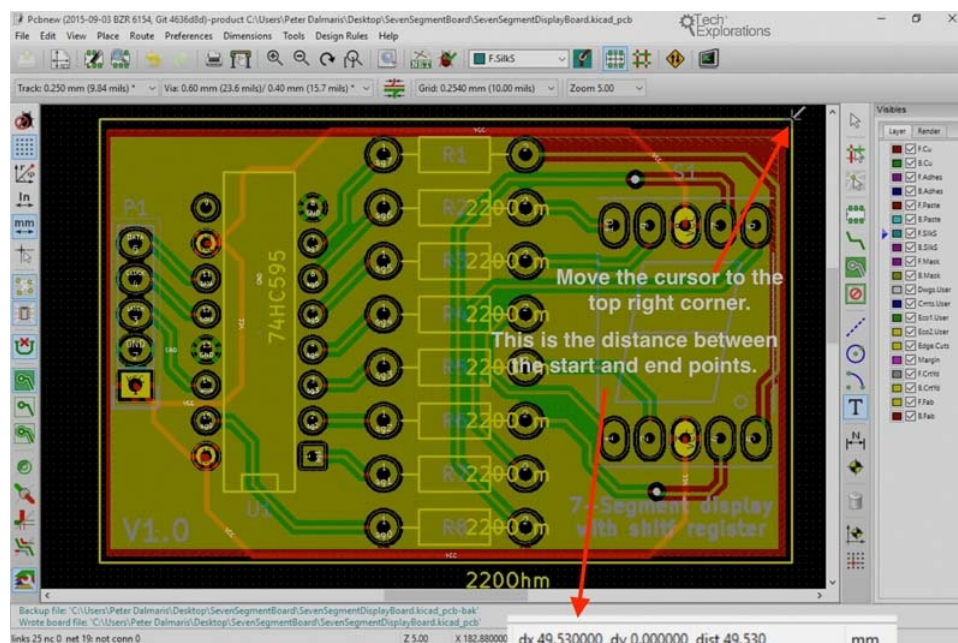
Of course, you can use whichever graphic you like!

First, we need to find out the dimensions of our board: width and height. Kicad contains a measuring tool in Pcbnew. Starting with the width, place your mouse cursor over the top left corner of the PCB and hit the space bar. This will zero the measuring tool at the right side of the status bar (at the bottom of the Pcbnew window).



Pressing the space bar will zero the measuring tool.

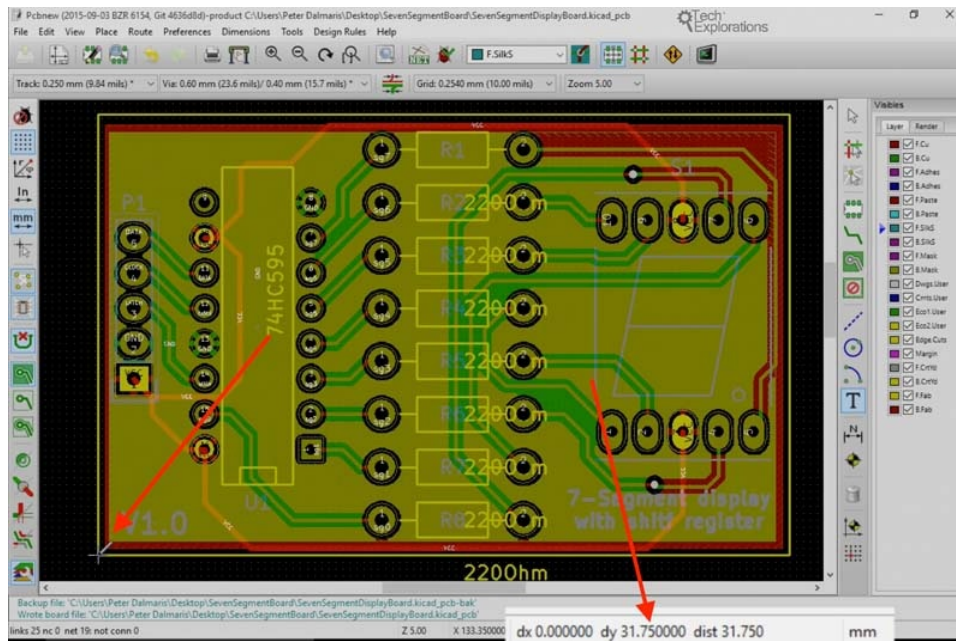
Then, move the mouse cursor to the top right corner of the board to measure the horizontal distance.



The distance, in the X and Y axis, between the start point (the position of the cursor when the space bar was pressed) and end points where the cursor is now showing in the status bar.

In the right side of the status bar you can see the horizontal (X-axis) distance between the original point of the cursor (when the space bar was pressed) and the current position. This tells us that the width of the board is 49.53 mm.

Move the cursor to the bottom left corner of the board to measure the height:



The height of the board is 31.75 mm

We now know that the height of the board is 31.75 mm. Because we are taking these measurements in order to calculate the maximum dimensions of our decorative graphic, we don't need to be too accurate. We can round these dimensions down to round numbers. Let's make the width 49 mm and the height 31 mm.

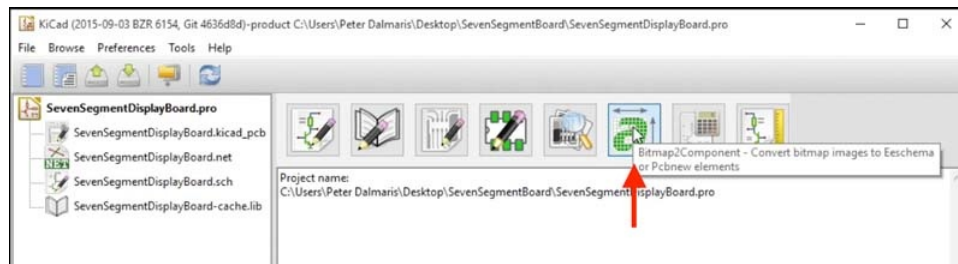
The process of creating the decorative graphic involves using a simple Kicad utility that convert a PNG image file into a Kicad footprint. Of course, we need a graphic file. I will use this:



I will convert this graphic into a Kicad footprint. Feel free to use your own graphic.

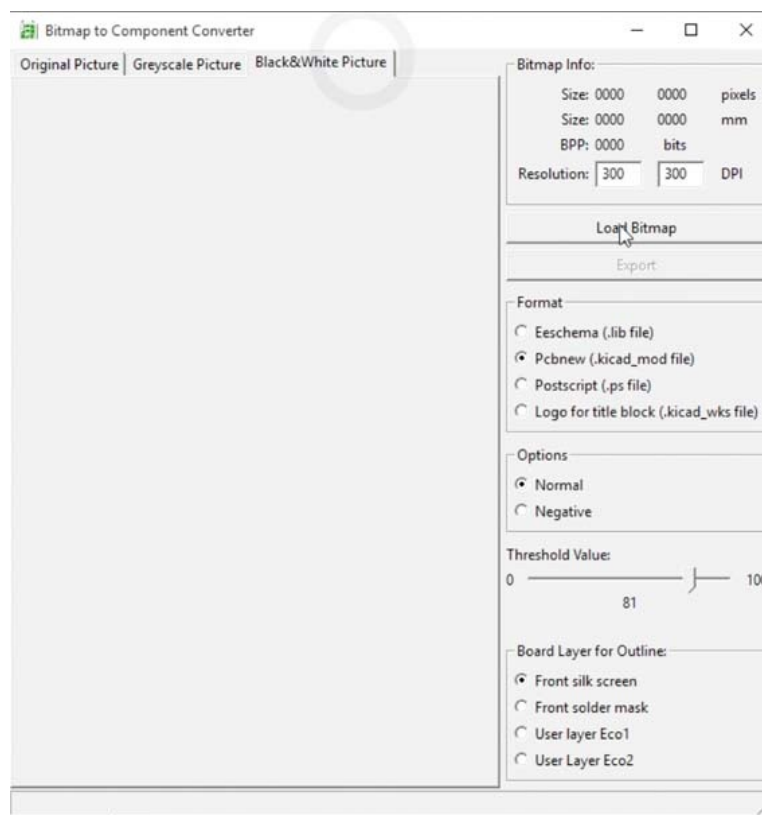
It is better to start with a graphic that has high resolution. Part of the process that will follow involve scaling down the image in order to reduce its size and make it fit inside our PCB.

From the main Kicad window, click on the button with the “a” icon to start the Bitmap2component utility.



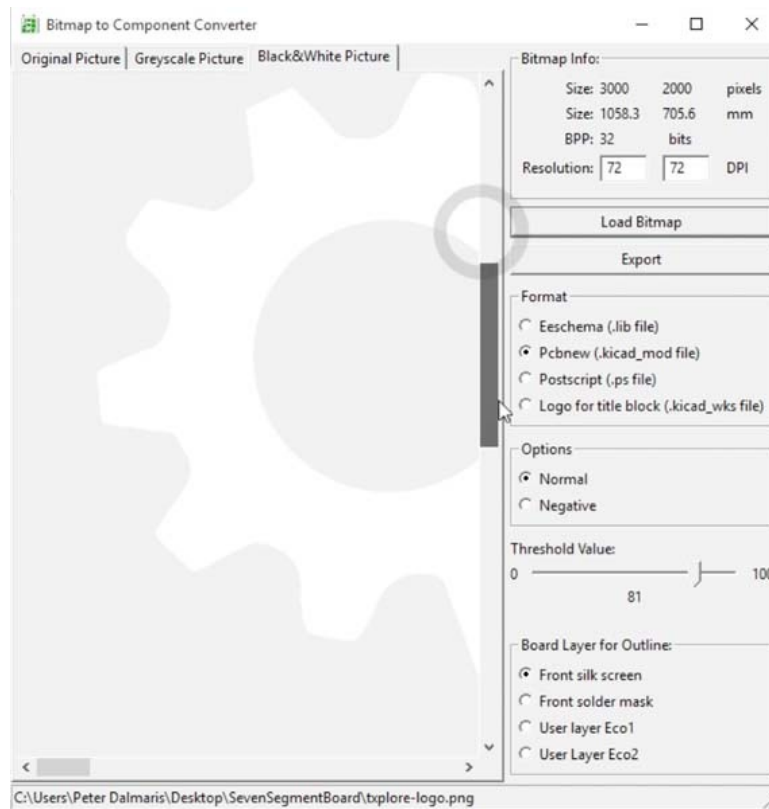
Click to start Bitmap2Component.

When the utility starts, notice that there are three tabs that allow you to convert different types of images into components. Because silkscreen are typically able to print in a single colour only, because my original image is black and white, I will select the “Black and White Picture” tab:



In most cases, we will be dealing with black and white PCB decorative images.

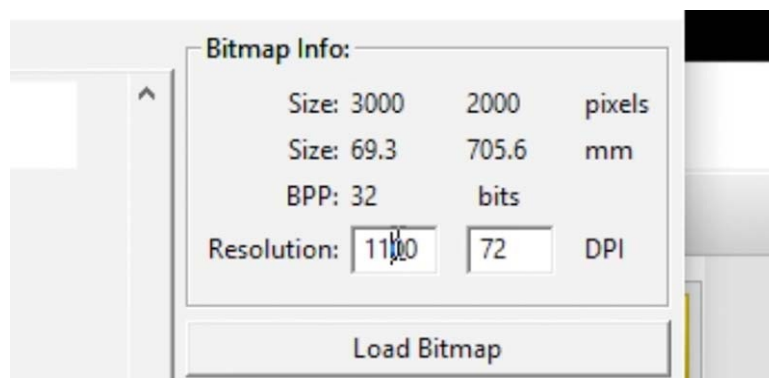
To load the image file, click on the “Load Bitmap” window. Browse to select the image, and once it loads, the utility will show you a full-sized preview:



The tool will provide a full-size preview of the image.

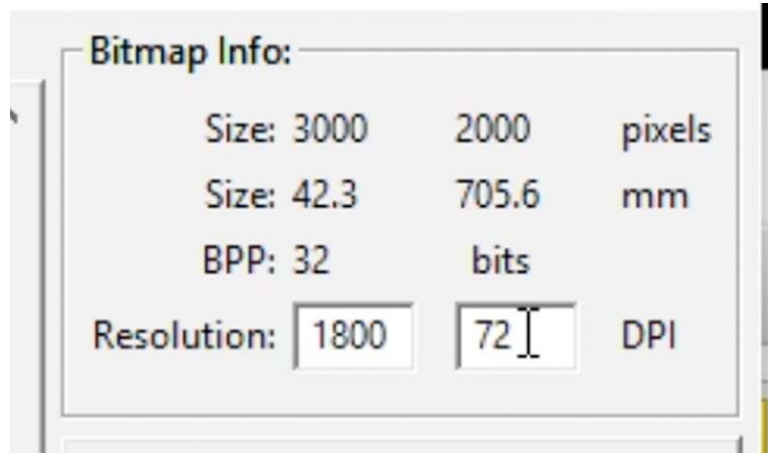
At the top right corner of the tool you can see the statistics, and a couple of fields where you can specify the resolution of the final footprint. You can also see the dimensions of the footprint. The process is simple: change the DPI resolution until the size of the image is smaller than the size of the board.

When we start with this image, the footprint contains 3000 pixels horizontally and 2000 pixels vertically, and this translates to a footprint of 1058.3 mm by 705.5 mm. This is a very large image and will not fit on our board, which measures only 49 mm by 31 mm. Let's edit the resolution down, and see how this changes the dimensions.



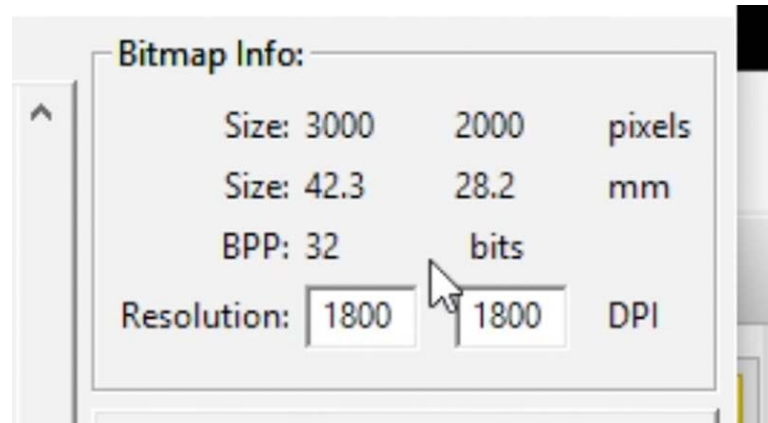
We can change the dimensions of the footprint by changing the resolution.

If we change the horizontal resolution to, say, 1100 DPI, the horizontal size will be reduced to 69.3 mm (from 1058.3 mm). Still, too large, so we need to increase the resolution further.



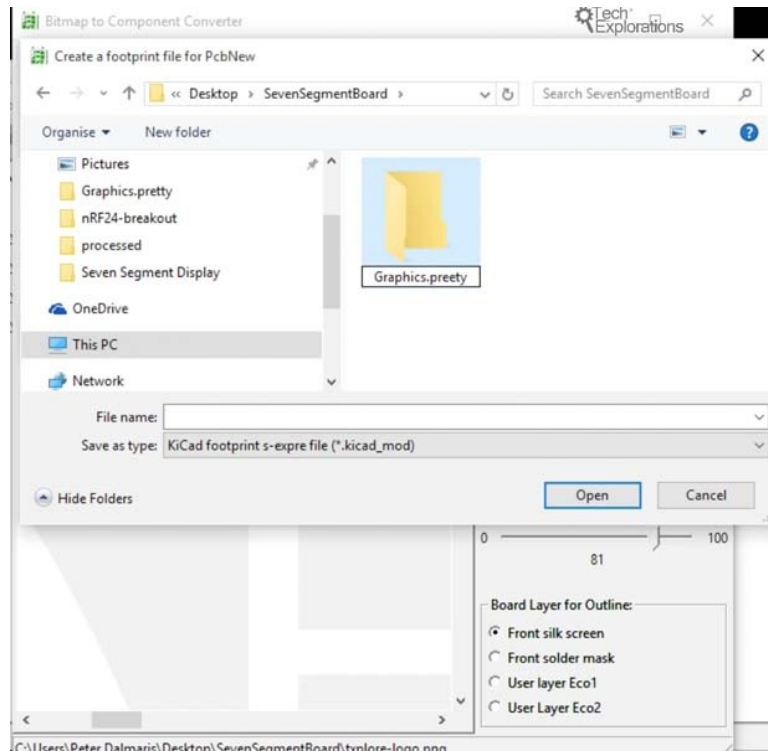
At 1800 DPI, the horizontal dimension is 42.3 mm.

If we increase the horizontal resolution to 1800 DPI, the horizontal dimension is reduced to 42.3 mm, which is fine for our board (considering that the image has a border around it). Let's make the vertical resolution also 1800 DPI in order to preserve the aspect ratio.



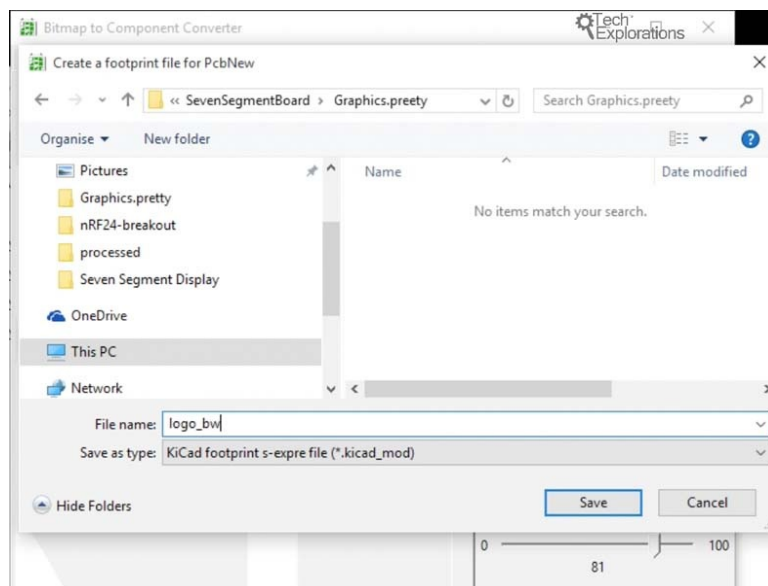
At 1800 DPI for both horizontal and vertical axes, our image is resampled so that it will fit inside the PCB.

At 1800 DPI resolution, our image is scaled down to 42.3 mm by 28.2 mm, which will fit inside our PCB. To create the footprint, click on the Export button. We must store the footprint file in our project directory, inside a new ".pretty" named directory.



We will store the new footprint in a directory titled “Graphics.pretty”, inside the project directory.

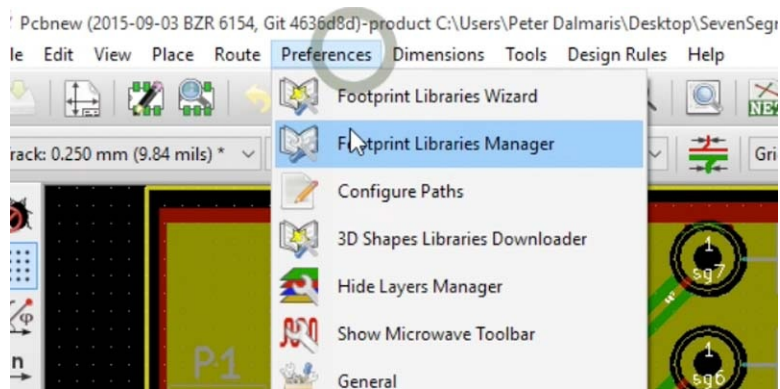
Name the footprint file “logo_bw”:



We will name the new footprint file “logo_bw”. Kicad will give it the “.kicad_mod” extension.

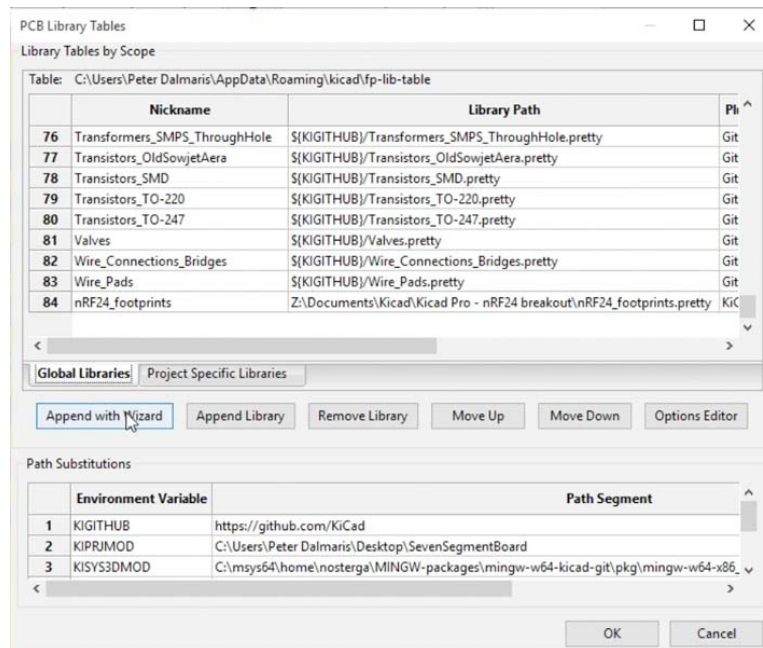
Click on the Save button, and close the utility. Our new footprint is now saved inside the project directory.

Return to Pcbnew and bring up the library manager in order to import the new footprint.

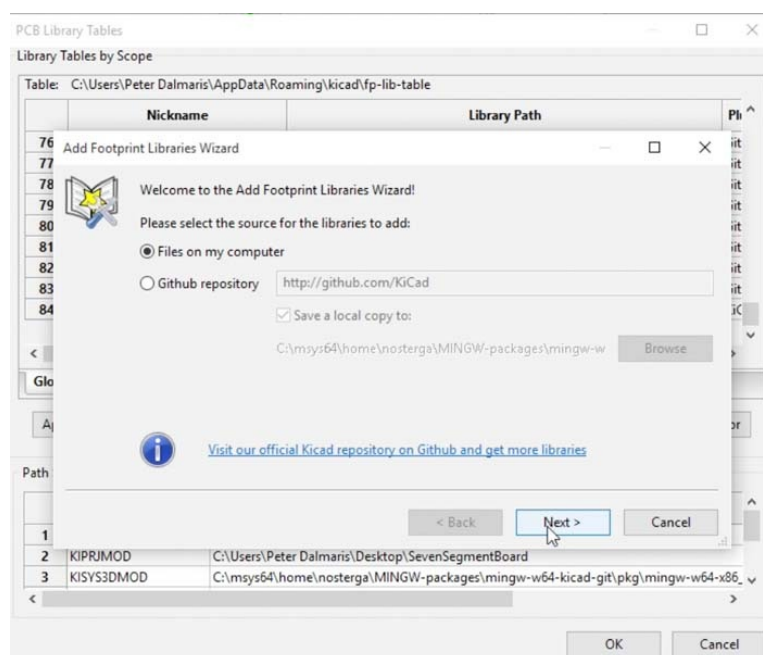


Start the Footprint Libraries Manager in Pcbnew.

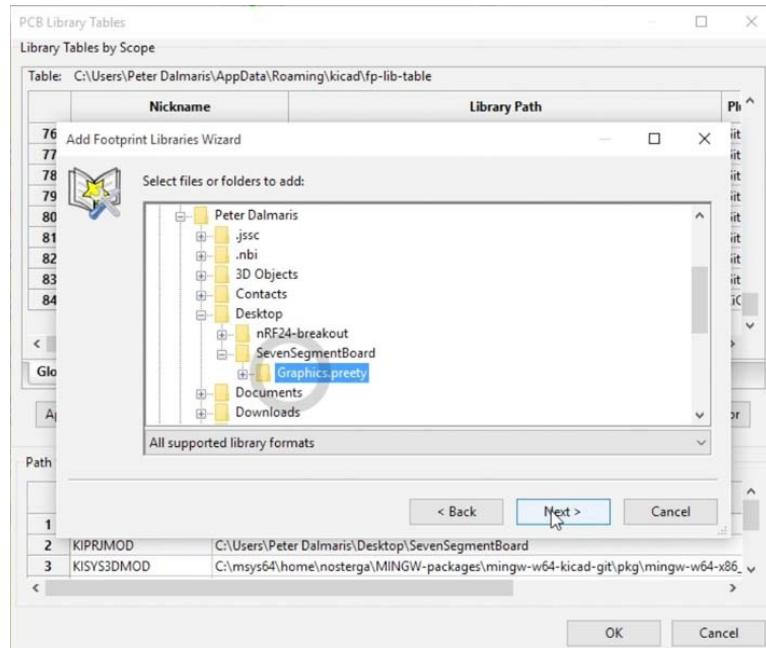
We will use the library Wizard to import the new footprint library. In the libraries manager window, click on the Append with Wizard button to start the Wizard:



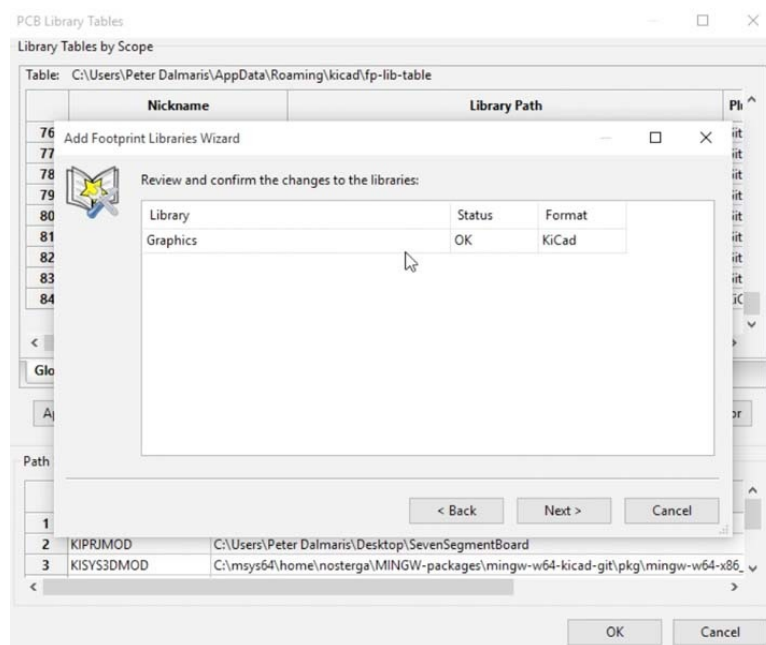
Start the Wizard.



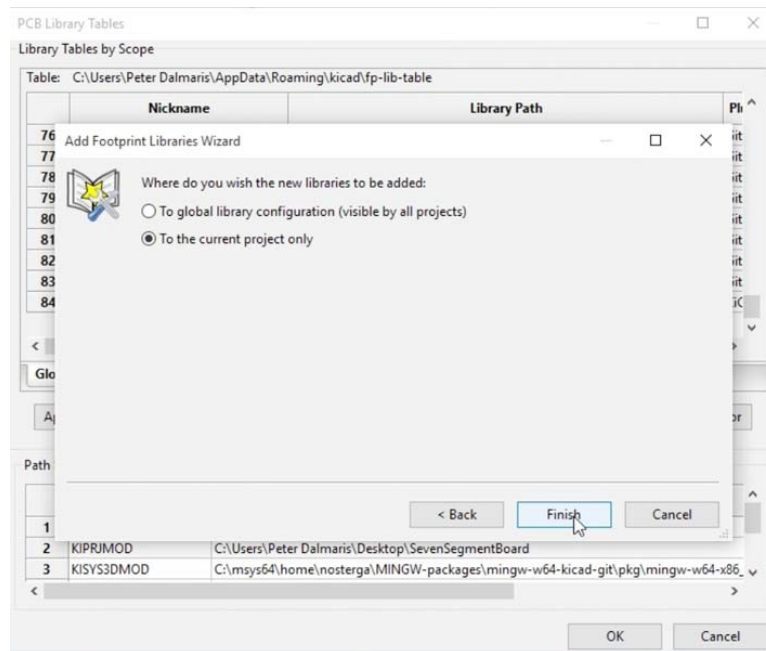
The file we are importing is on the local machine.



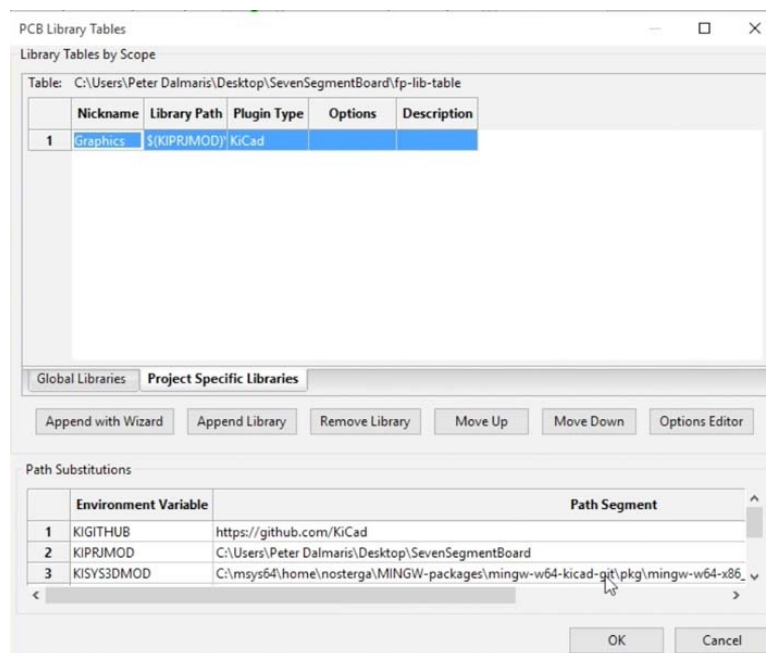
Browse to the local directory where we created the Graphics.pretty folder with the footprint file.



The Wizard recognises this as a native Kicad footprint.

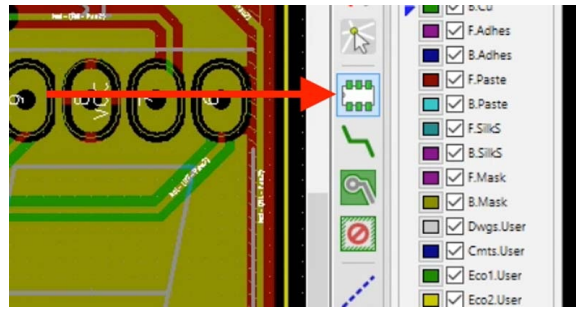


We will make this library available to the current project only. If you want it available to all your projects select the first radio button. Click Finish to ... finish ... Wizard.



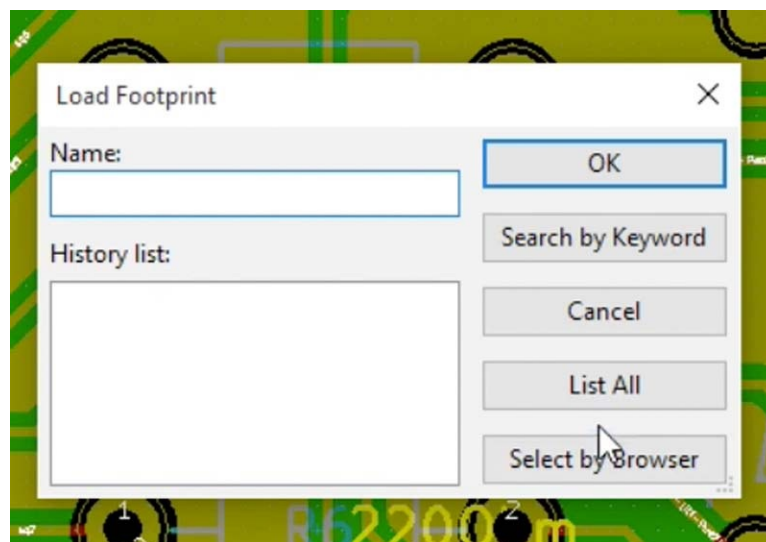
When the Wizard finished, you will be shown the Project Specific Libraries tab in the Library Manager. Our newly imported library is included!

Now we can add the footprint we just created to the back silkscreen of our board. Click on the Add Footprint button:



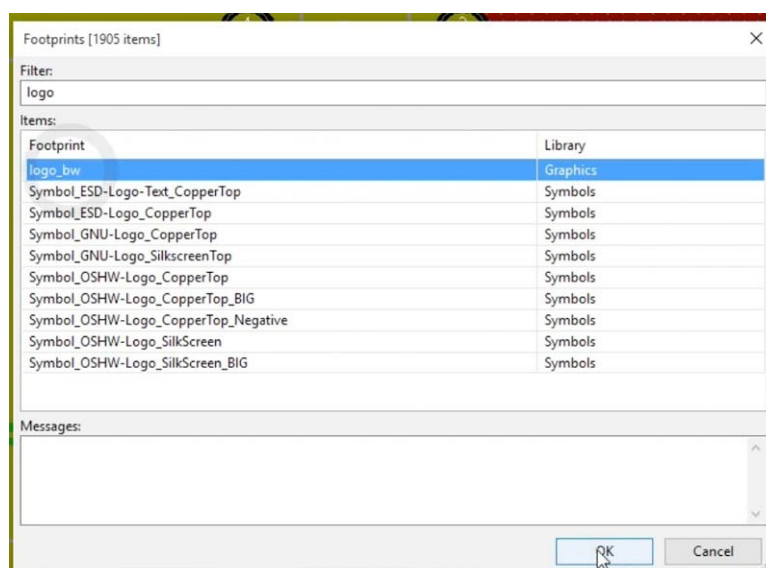
The Add Footprint button.

Click anywhere on the canvas to add the footprint, and the footprint chooser will appear:



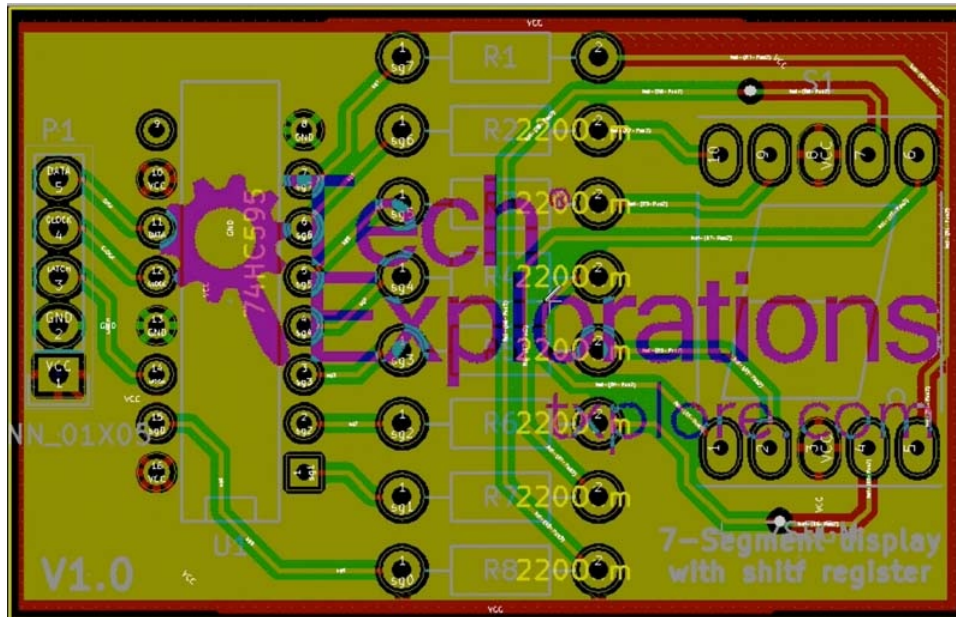
The component chooser. Click on List All.

We can use the filter to look for the new footprint. Click on List All:



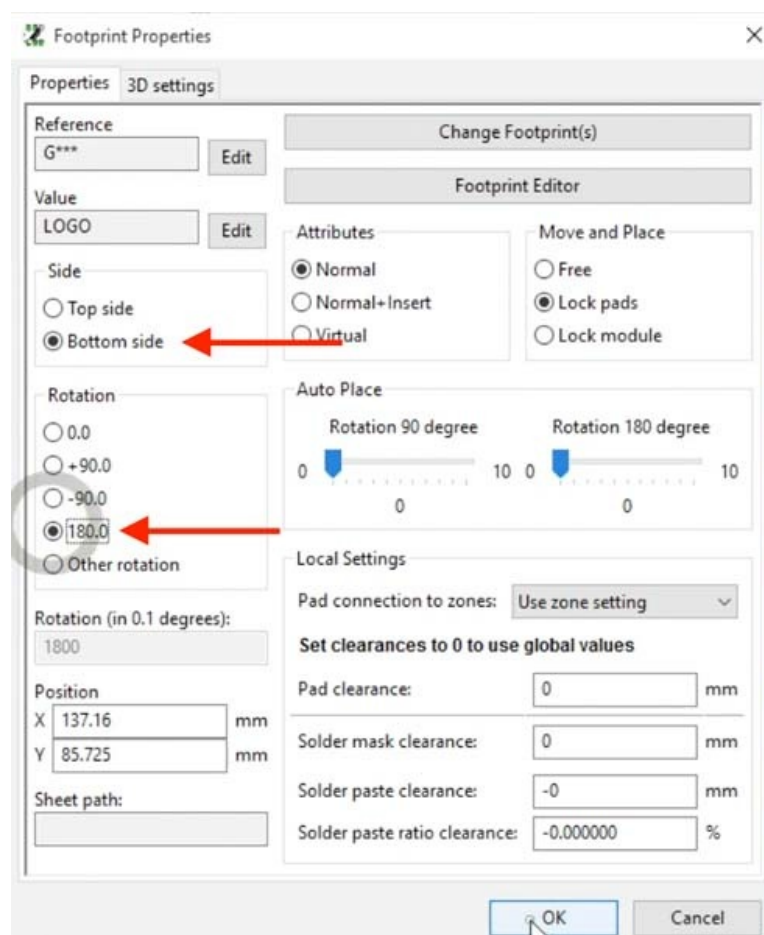
In the filter box, type a part of the footprint name, like “logo”.

Select the footprint and click OK. This will drop the graphics footprint on the canvas:



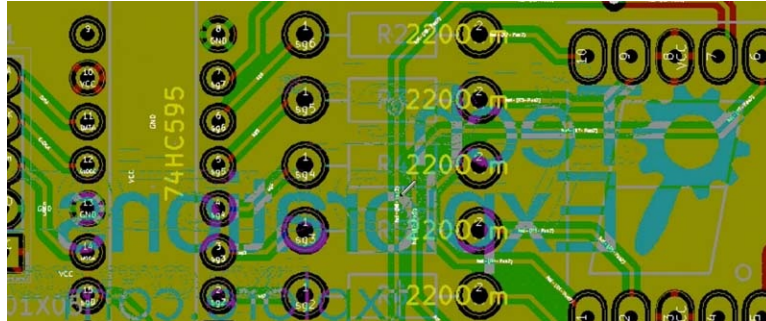
The decorative footprint. Since we want to place it in the bottom layer, we need to reverse it.

At the moment, the footprint is placed on the top layer by default. We would like to place it on the bottom layer instead. To do this, place you mouse over the graphic, and type “E” to reveal the footprint properties window:



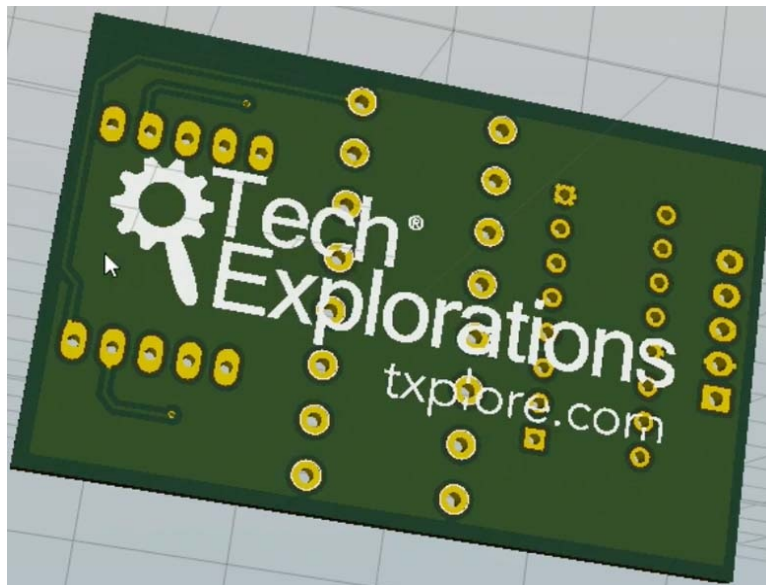
In Footprint Properties, choose “Bottom side” in the “Side” group, and “180.0” in the “Rotation” group.

In the Footprint Properties window, we will choose to place the footprint in the Bottom side. We also have to mirror the footprint in order to orient it properly, so choose “180.0” in the Rotation group. The board now looks like this:



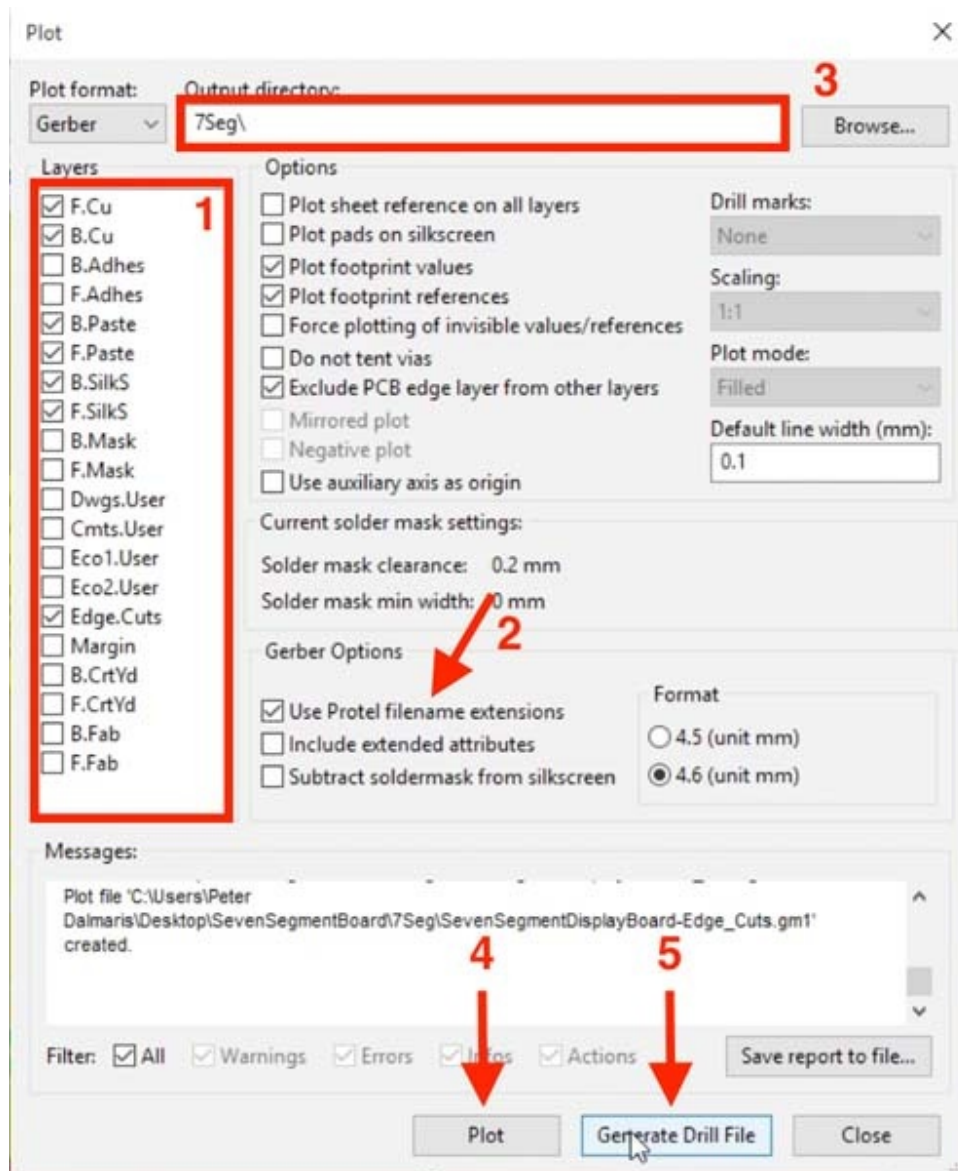
The decorative graphic is now in the back layer, properly oriented.

The PCB view is not very clear, so let's check out the 3D rendering:



Here is the decorative graphic, nicely showing in this 3D rendering.

I really like this, I hope you agree! With the decorative graphic footprint in place, we have completed the implementation of the PCB. The only thing left to do is to export the Gerber files, test it with Gerblook, and upload it to the manufacturer. We will do this in the next chapter.

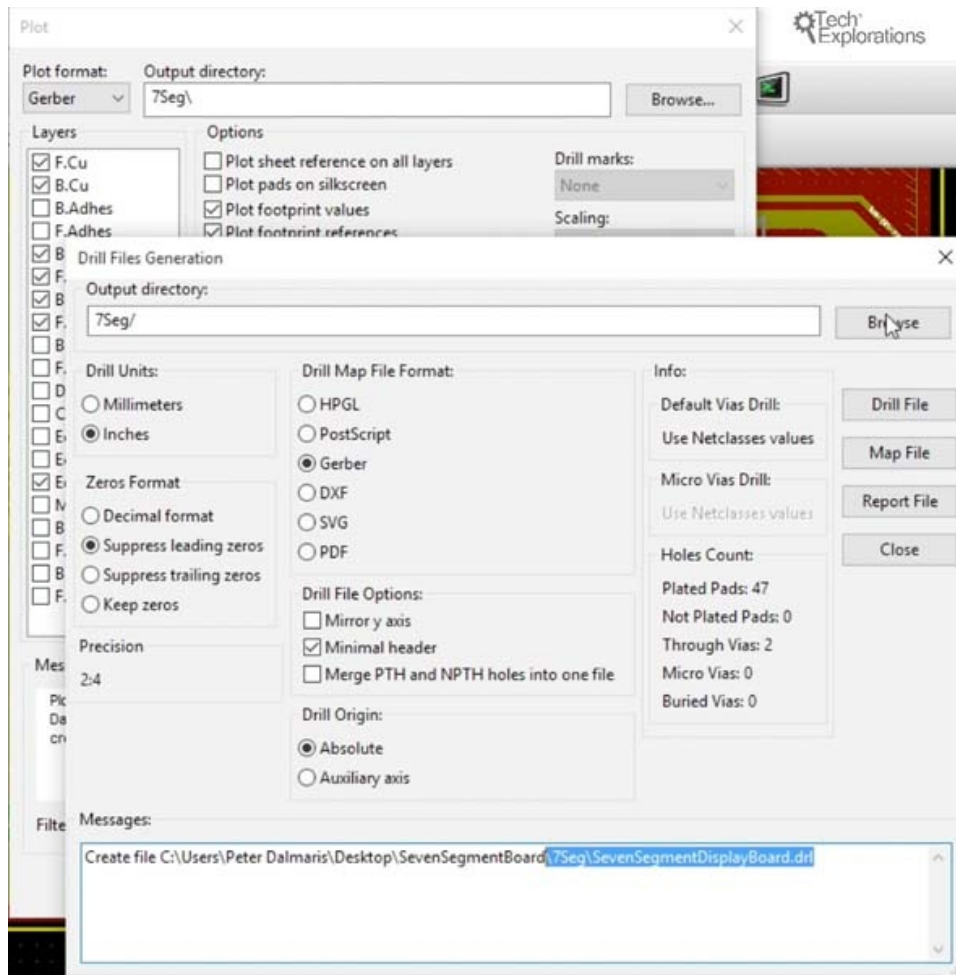


The Plot window controls the creation of the Gerber files. The numbers indicate the order of interaction with the window.

In the Plot window, start by selecting the layers to export. Just like in the first project, we are going to export the F.Cu, B.Cu, B.Paste, F.Paste, B.Silk, F.Silk and the Edge.Cuts layers. As you will see later, one of the required layers is missing, but let's continue with this omission to see what will happen later in the process.

In the Gerber Options group, check the “Use Protel filename extensions”, and select the output directory. Create a new directory so that we can create a ZIP archive from it later.

Next, click on the Plot button to have the Gerber files for these layers created. You will see the Messages box populated with the status of the Gerber generation process. The last thing to do here is to create the Drill file, which tells the manufacturer where to drill for pad holes and vias. Click on the Generate Drill File button.

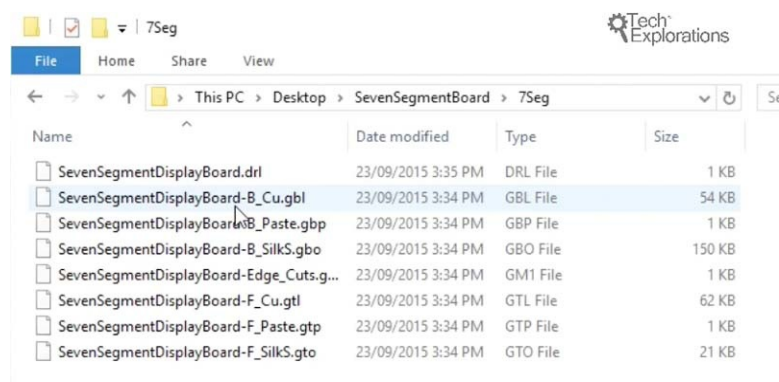


All the defaults in the Drill Files Generation window a good.

In the Drill Files Generation window simply inspect the settings as the defaults are correct. Click on the Drill File to generate the file, and confirm the operation by looking in the Messages box.

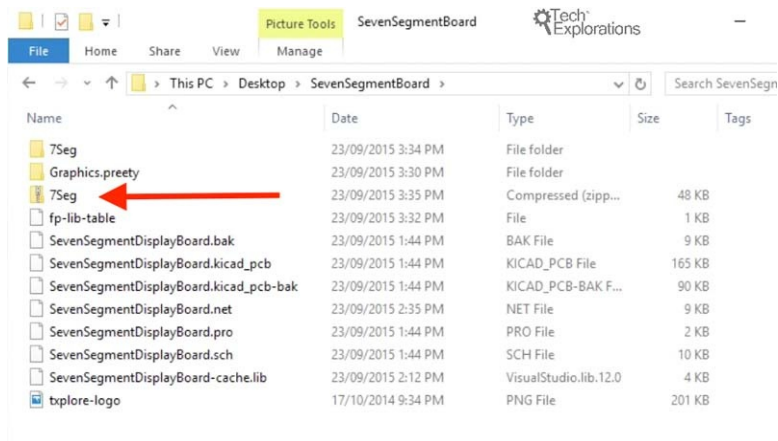
The Gerber files are now saved in the Gerber directory, so click on Close to close the Drill Files Generation window, and Close again to dismiss the Plot window.

Here are the Gerber files:



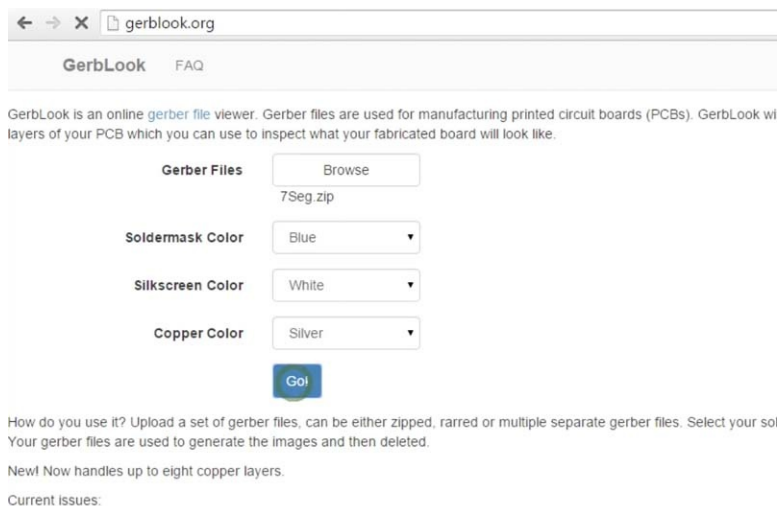
The Gerber files. One for each layers, plus the drill file.

Create a ZIP archive from the Gerber directory:

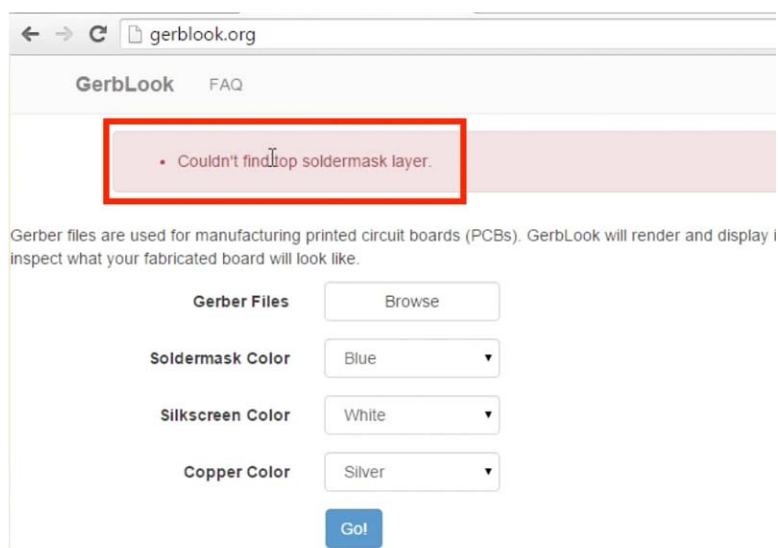


The Gerber files ZIP archive.

We will use [Gerblook.org](http://gerblook.org) to confirm that our Gerber files are valid and that we have not forgotten any of the layers (which is often the case!). Use your browser and go to gerblook.org. Then upload the Gerber ZIP archive:



Go to gerblook.org and upload the ZIP archive that contains the Gerber files.

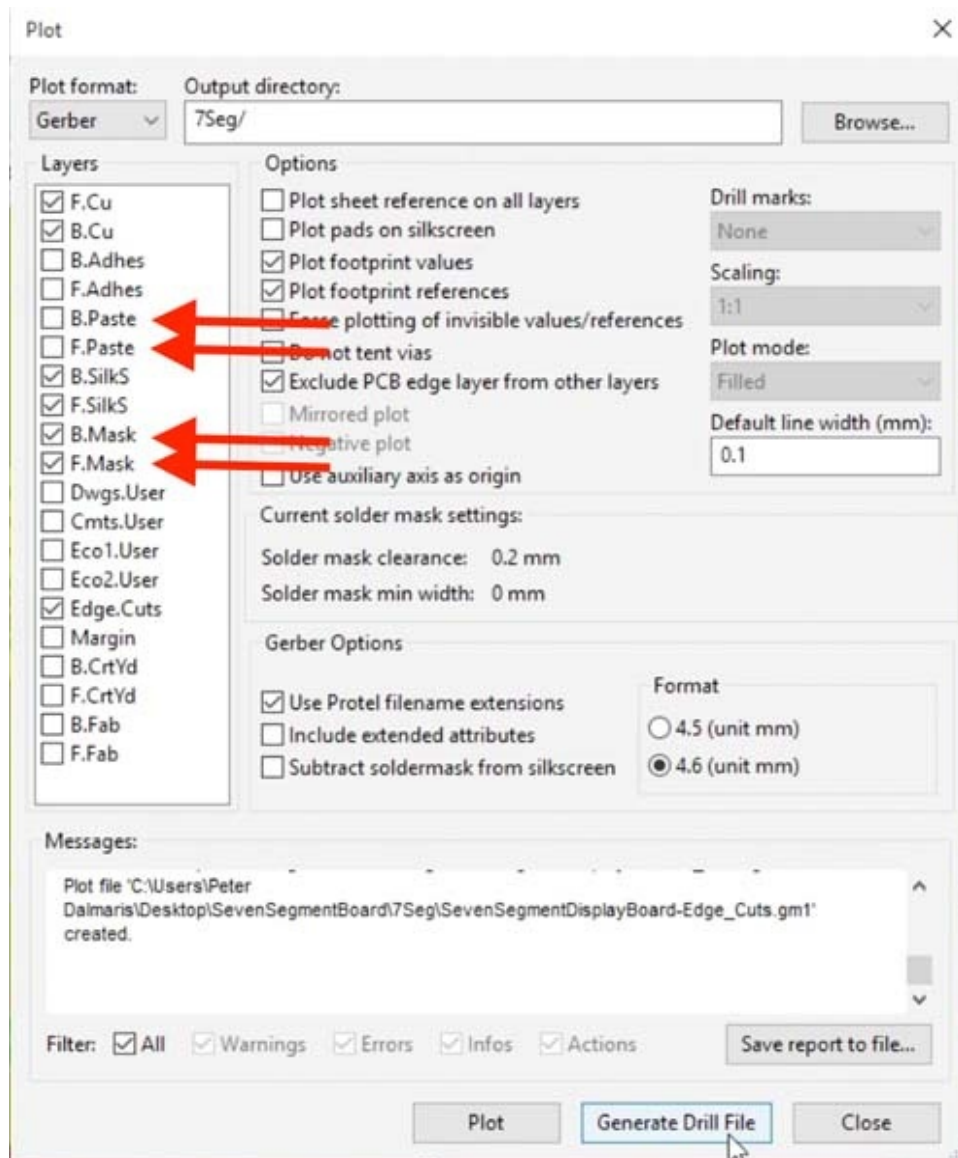


Oh no! It seems that we forgot to include a required layer!

Gerblook detected that a required layer, the solder mask layer, is not included in the ZIP archive. If we had proceeded and uploaded this ZIP archive to a manufacturer, the

board would be produced anyway, but with this layer missing the board would be useless. Therefore, it is a good practice to always check your Gerber ZIP archive with a service like Gerblook or similar to make sure that the archive is valid!

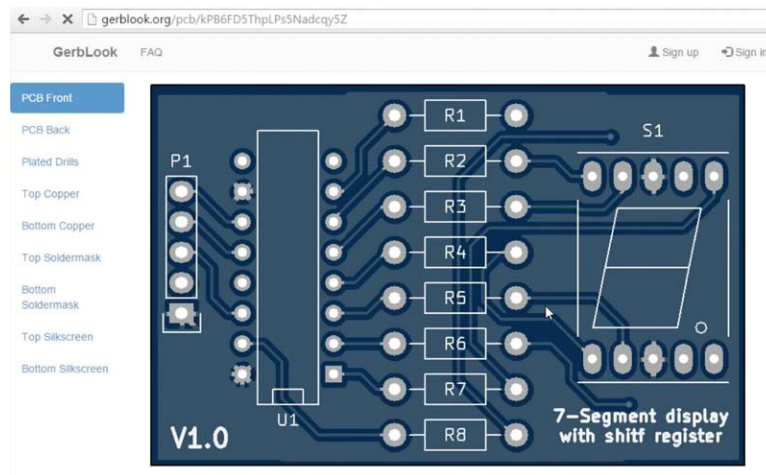
Go back to the Plot window in Pcbnew, and click to select the B.Mask and F.Mask layers. You should un-select the B.Paste and F.Paste layers that I incorrectly selected in my first attempt.



Select the back and front solder mask layers. Unselect the front and back paste layers.

Click on the Plot button to create the missing Gerber files, and the Close to dismiss the window. We don't worry about the Drill file since we generated it earlier.

Delete the existing ZIP archive, and create it again from the Gerber file directory. Return to your web browser and Gerbview, and upload the new ZIP archive. Gerbview will render the layer and present them in a single page:



The new Gerber ZIP archive passed the Gerblook test!



The back of the PCB on Gerblook.

Gerblook reported that the new Gerber ZIP archive is in order.

We can now proceed with our order. I got mine from OSH Park, but feel free to choose your own manufacturer. The ZIP archive we have created is a de-facto industry standard and I had no trouble using it with various manufacturers.

PART SEVEN

Project 3: a full-SMD 16-LED board

Chapter 49: *What is this part*

In the third project for this course, we will build a 16-led array PCB, using two layers and surface-mounted components.

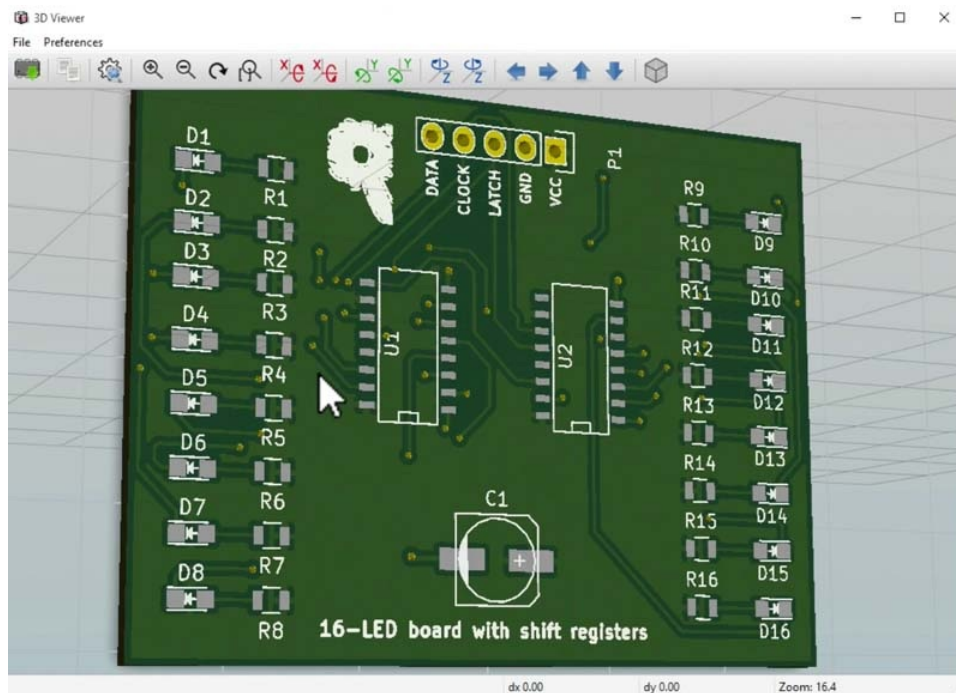
Using surface-mounted components does not change materially the process of designing the PCB. In fact, in the schematic editor you don't even have to think about it. Any changes to the design process apply at the layout and wiring stage, in Pcbnew.

In Pcbnew you need to be mindful of the fact that the components are placed on pads that only exist on one layer, instead of pads that are connected on both sides of the PCB through a hole.

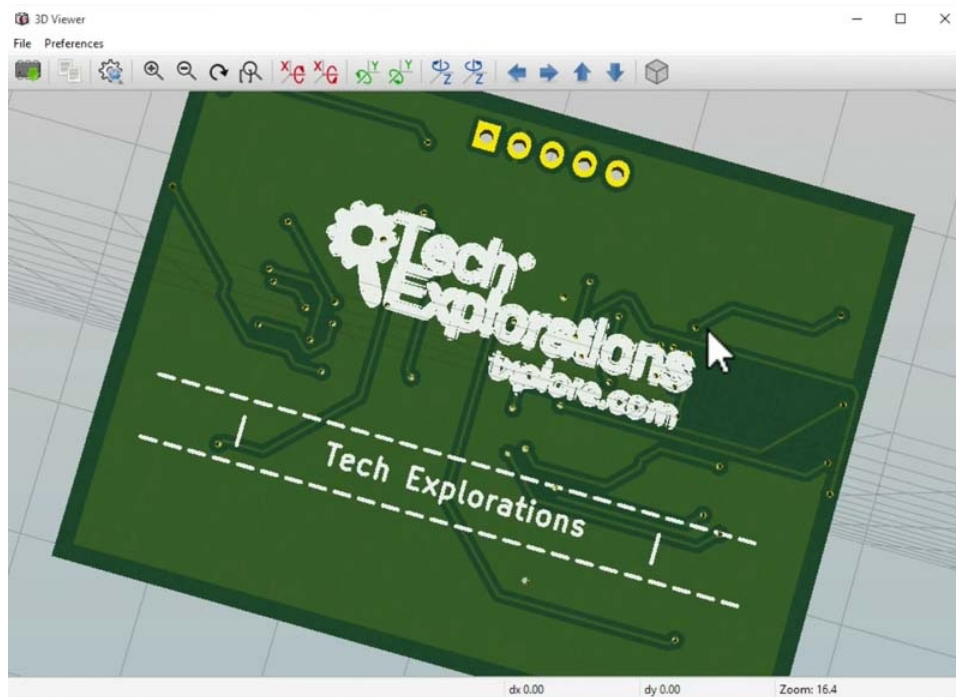
In this project, apart from using surface mounted components, you will consolidate the features and techniques you learned about up to know.

You will work with components, nets, labels, busses, power flags, the electrical rules check, netlist, footprints, track widths, copper fills, graphics, and Gerber files.

This is what the final board will look like:



The final PCB for the third project, front side.

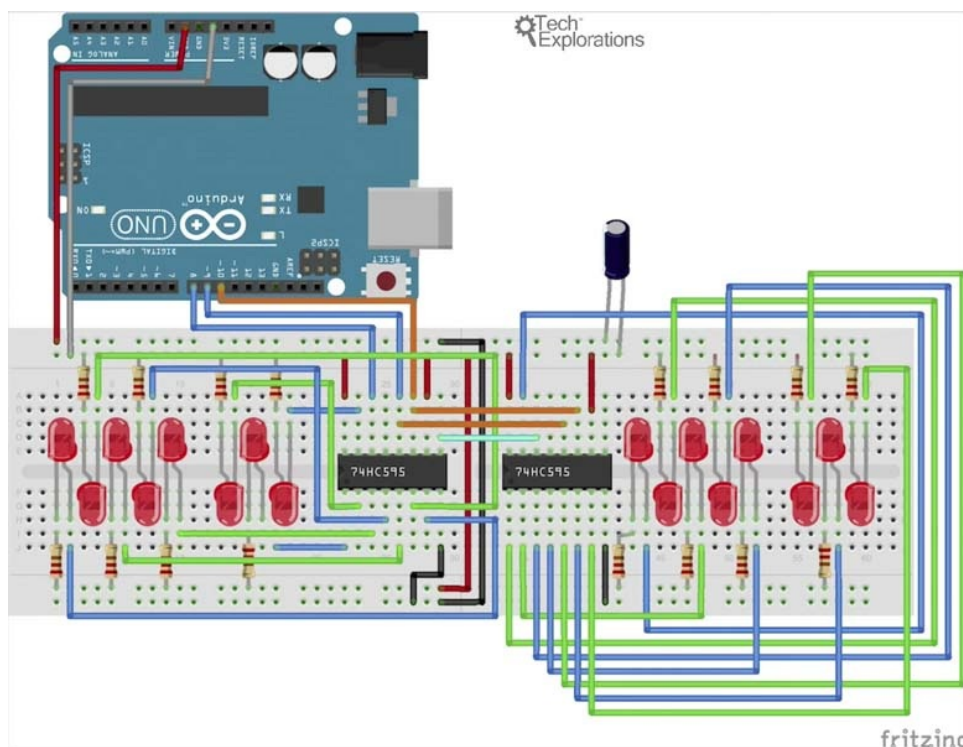


The final PCB for the third project, back side.

Chapter 50: *The circuit*

Before we start with the design work of the PCB in Kicad, let's have a look at the original circuit.

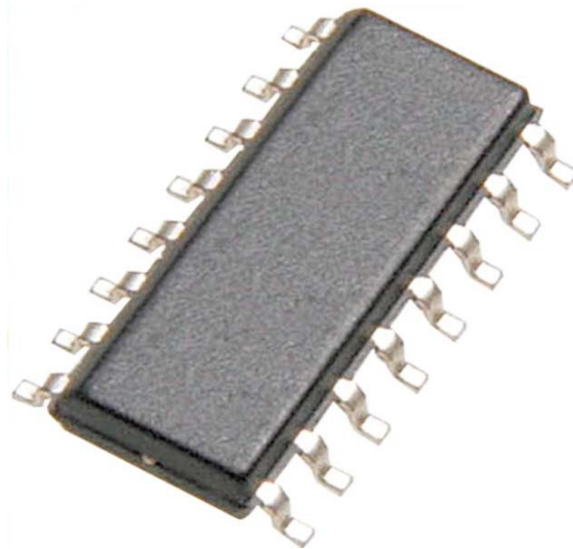
The circuit, implemented on a breadboard, comes from one of my Arduino step-by-step lectures. If you haven't completed that lecture from Arduino step-by-step, again, don't worry about it. You'll be perfectly able to follow along with what we're doing here.



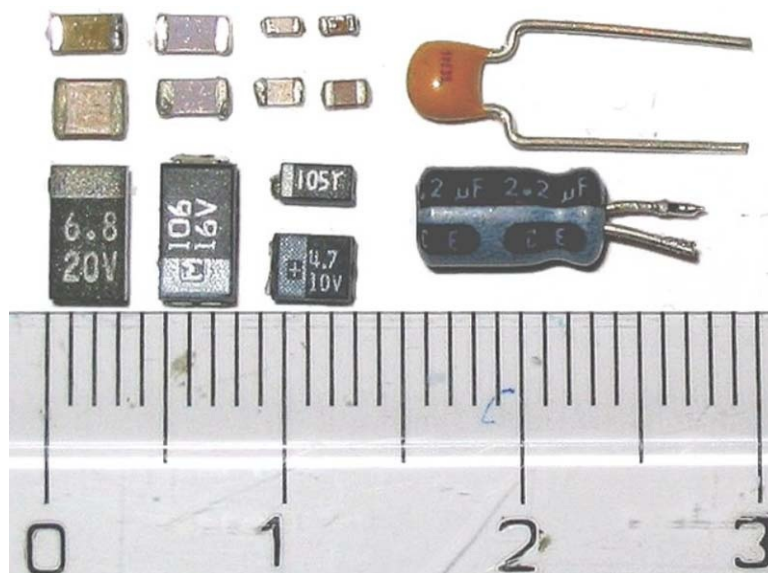
The circuit, on a breadboard. We will design a PCB for it.

As you can see in this circuit, there are 16 LEDs in total on the breadboard. I would like the PCB to provide pads for common 0805 surface mounted LEDs supported by similarly sized surface mounted resistors. The circuit will also contain two surface mounted 595 shift register integrated circuits, and a surface mounted electrolytic capacitor.

The only through-hole component on the PCB for this project will be the 5 pin connector.



The two ICs will be contained in an SMD package like this.



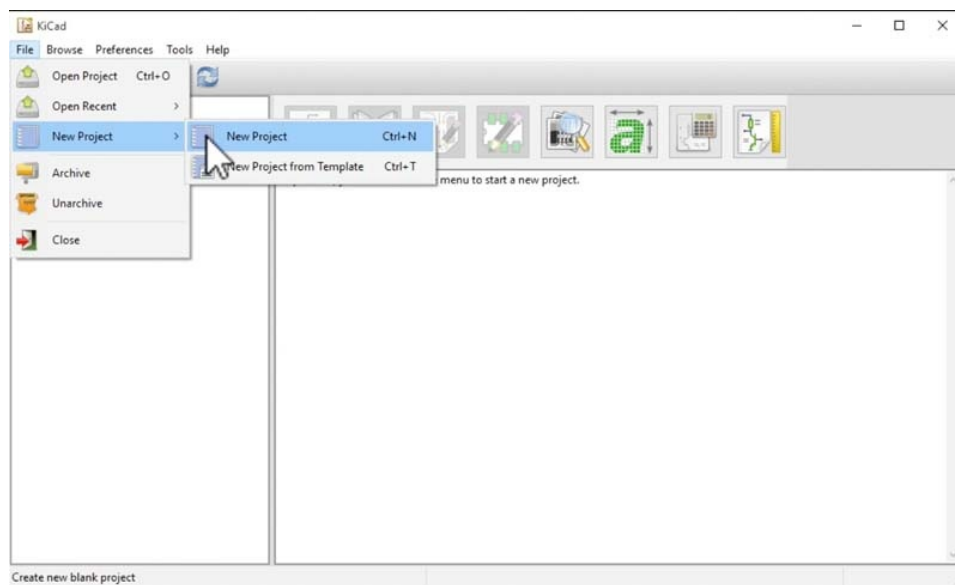
Examples of capacitors. The left of the image shows several surface mounted capacitors.

Let's begin with the work in Kicad!

Chapter 51: *Create the schematic in Eeschema*

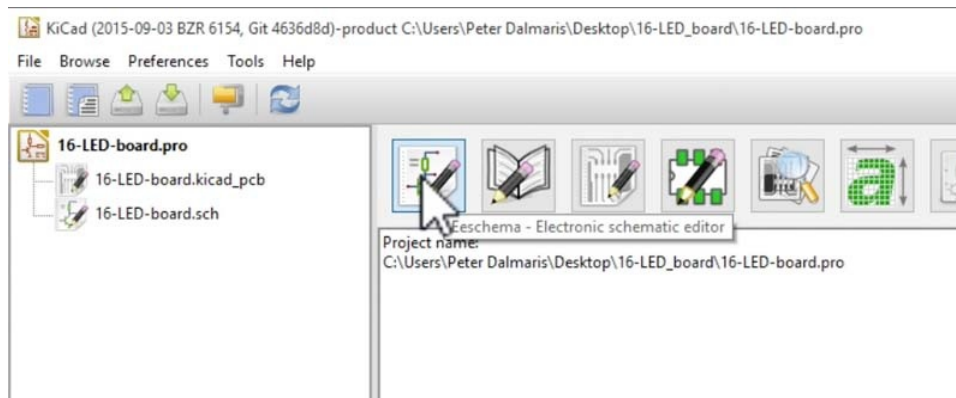
In the last chapter we looked at the breadboard wiring diagram for the circuit that we will work to create a PCB for. In this chapter, we'll start the process by creating a new Kicad project, and doing the schematic in Eeschema.

Let's create the new project. Start Kicad, go to the File menu item, then New Project, and then click on New Project.



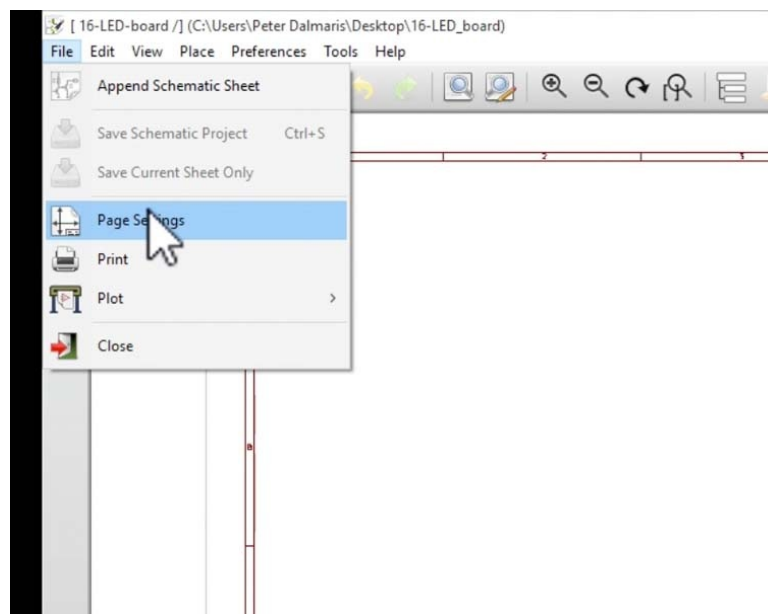
Starting a new project.

Create a new directory for the project, and give it a name like “16-LED-board”. Give your project a reasonable name, like “16-LED-board”, and the Kicad main window will look like this:

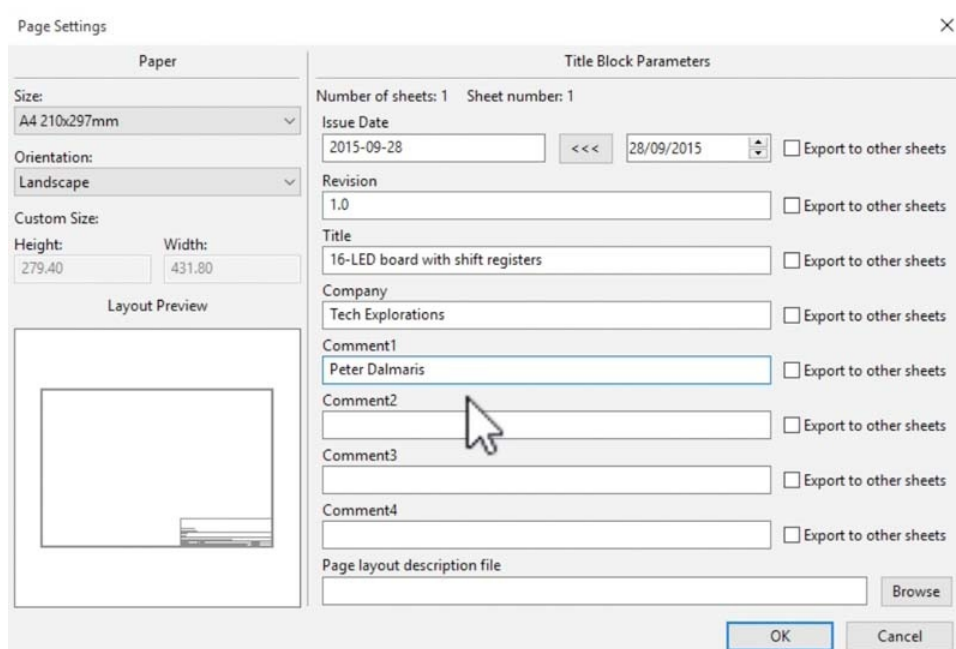


The new project is created, ready for Eeschema!

Let's start Eeschema. Click on the Eeschema button. Once Eeschema launches, edit the page properties so that the project information is shown in the canvas legend.

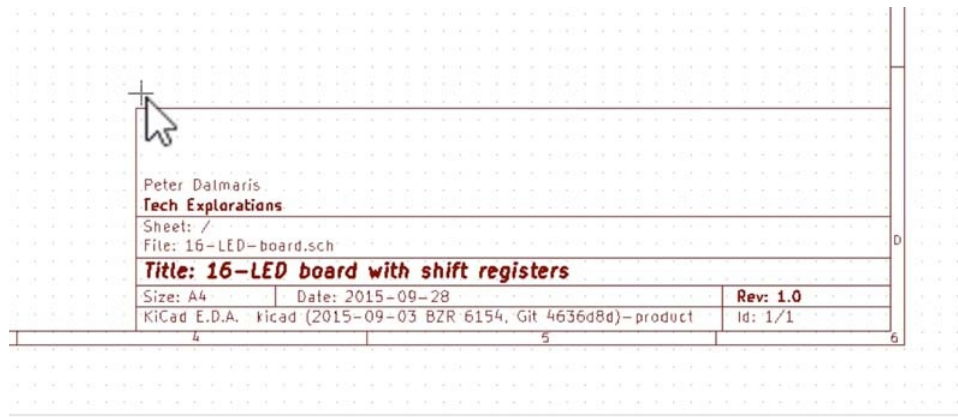


Bring up the Page Settings window to edit the project information.



Populate the project information text fields.

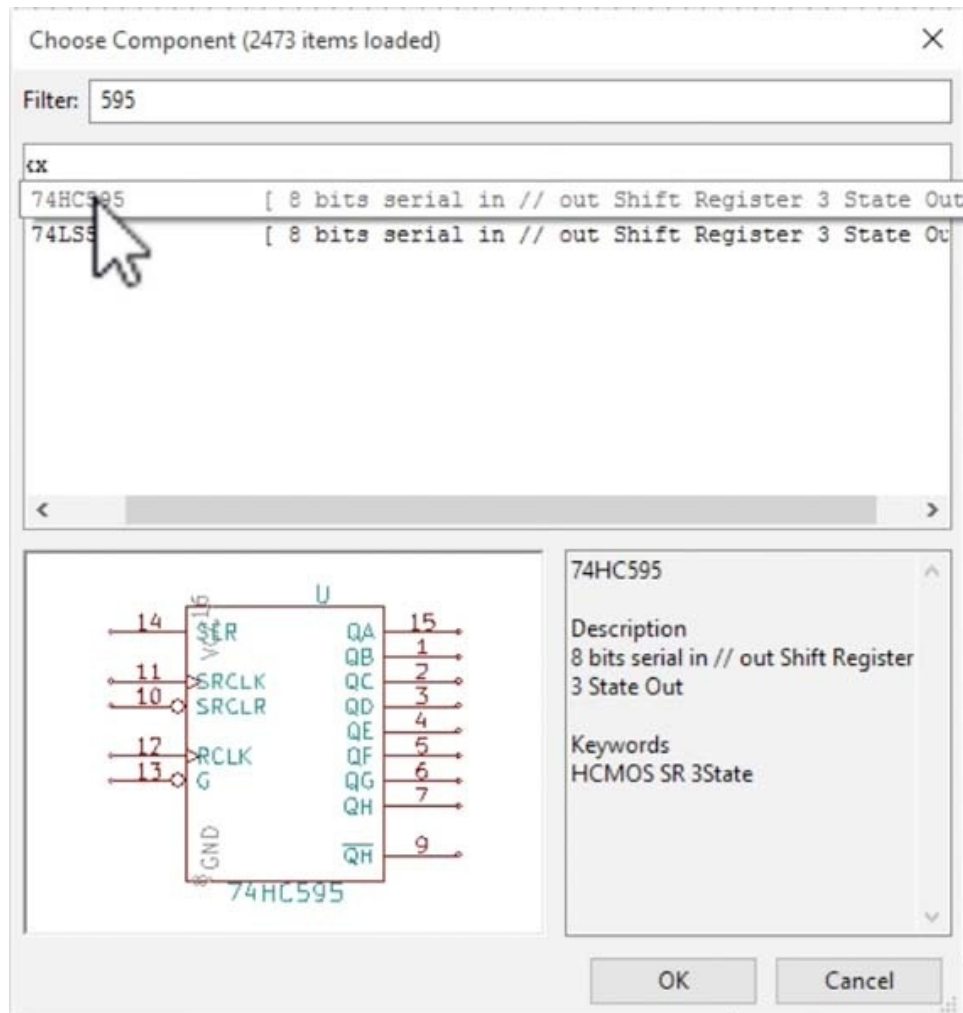
Once all the details are completed, click OK. The canvas legend will look like this:



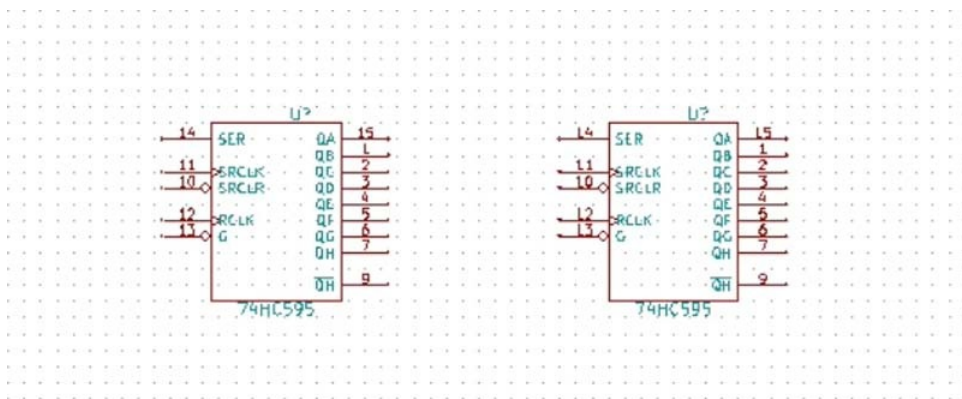
The canvas legend with the project information.

We will start again as usual by dropping in the components into the canvas, making extensive use of pin labels and buses. We will use lines, boxes and text labels to annotate the schematic to make it easy to read if you decide to print it out.

Let's start the process by dropping the components to the canvas. Hit the "A" key to bring up the components chooser. Look for the larger components first, like the 595 shift register. Just like in previous project, we will use the HC595 version of the shift register. We will need two for these ICs. We can simply copy the first one by placing the mouse pointer over the first component and typing "C" to make a copy. We now have two shift registers on the canvas. Here's what your canvas will look like now:

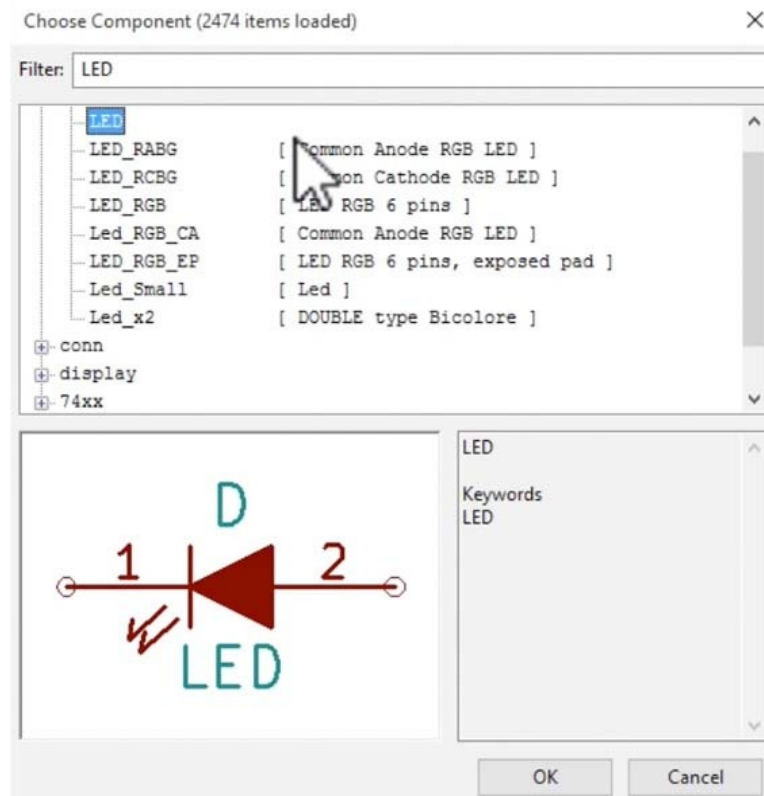


Select the HC595 IC and click OK to drop it to the canvas.



Make a copy of the first IC with the “C” key. We now have two ICs on the canvas.

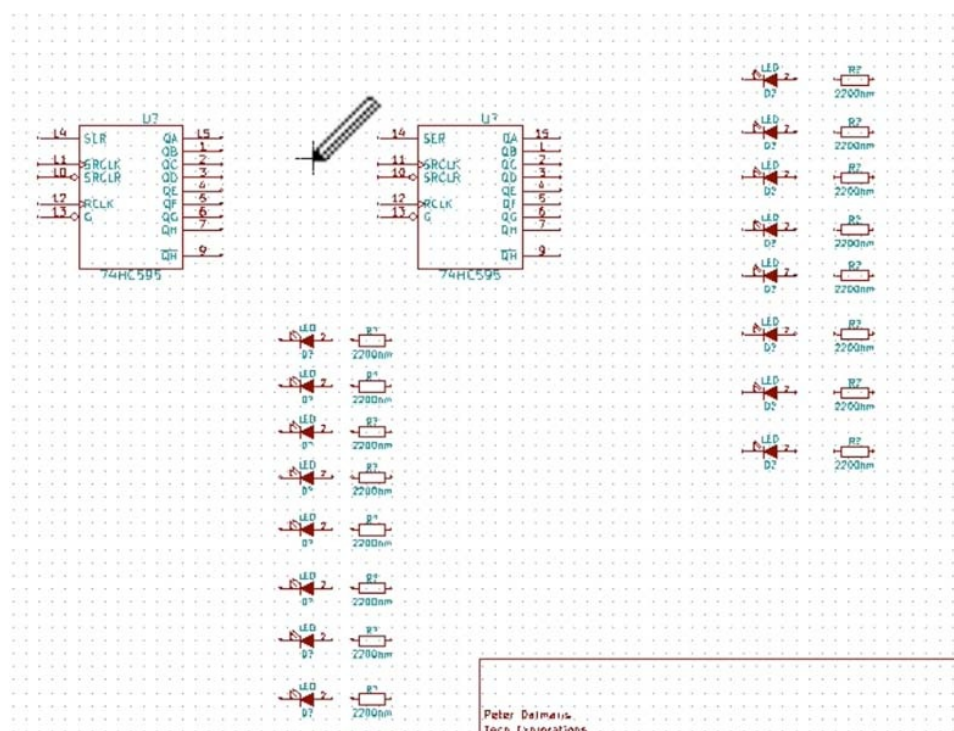
Next let’s add the LEDs. Hit the “A” key to bring up the component chooser, and select the LED component:



Select an LED component.

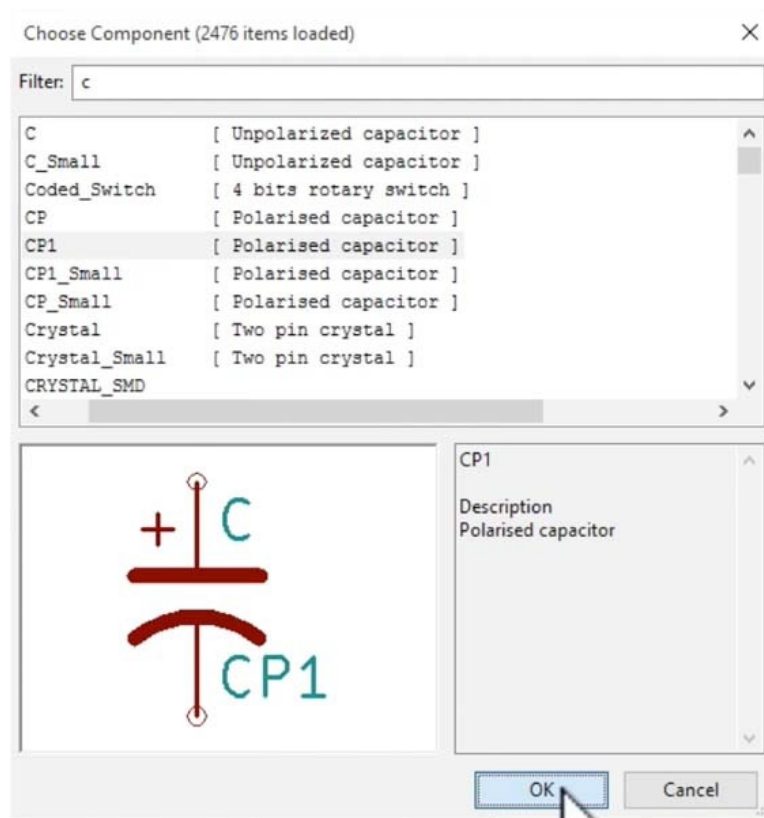
We will need 16 of the LED components, arranged in two rows of 8 LEDs each. Just like with the IC, the easiest way to add more of the same is to make copies of the first LED, and arrange them in two rows. Then, do exactly the same thing and add 16 resistors, also arranged in two rows.

At the end of this process, your canvas should look like this:



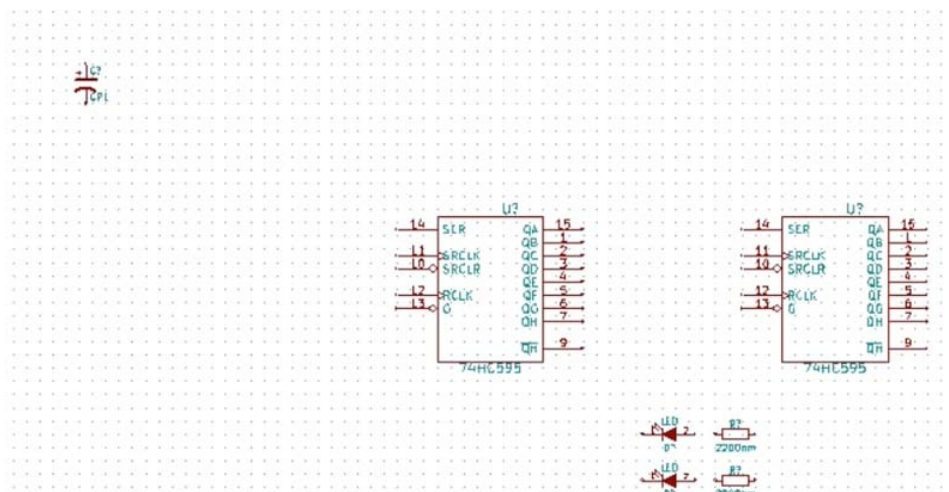
The canvas now contains 2 595 ICs, 16 LEDs and 16 resistors.

Check the breadboard schematic, what else do we need? Of course, we need a capacitor, so let's add one. We're using an electrolytic capacitor so we want to add a polarized capacitor to the canvas. In the component chooser, use "C" as the filter keyword, and choose one of the available polarised capacitors:



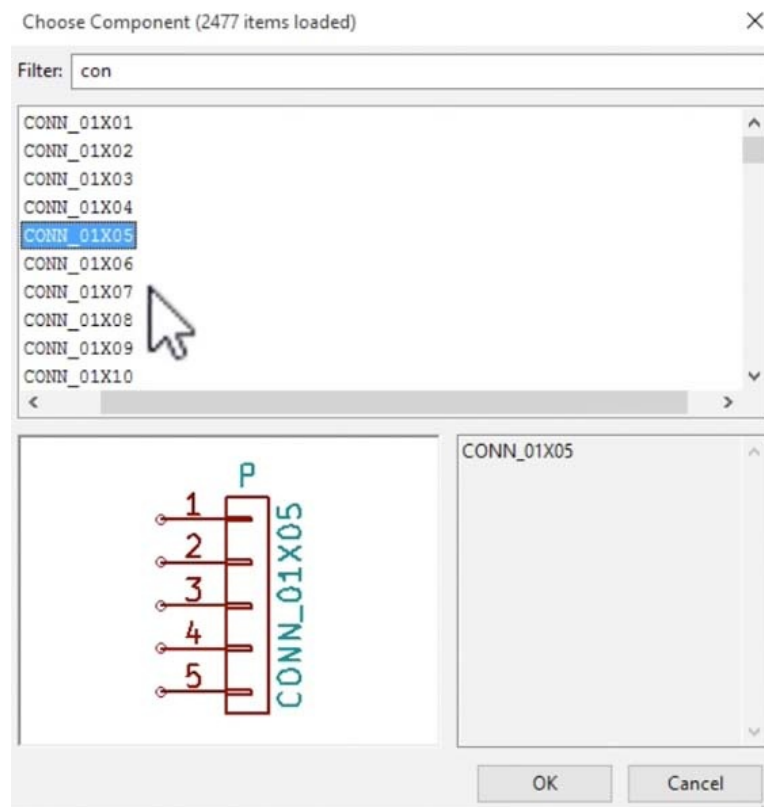
Add a polarised capacitor.

Because we will connect the capacitor to the rest of the circuit via labels, instead of actual wires, we don't have to place it close to the other components. In fact, because there will be a lot of wires, buses and labels close to the two ICs, I prefer to place this capacitor further away from them. Let's place it at the top left corner of the canvas, where there is plenty of space:



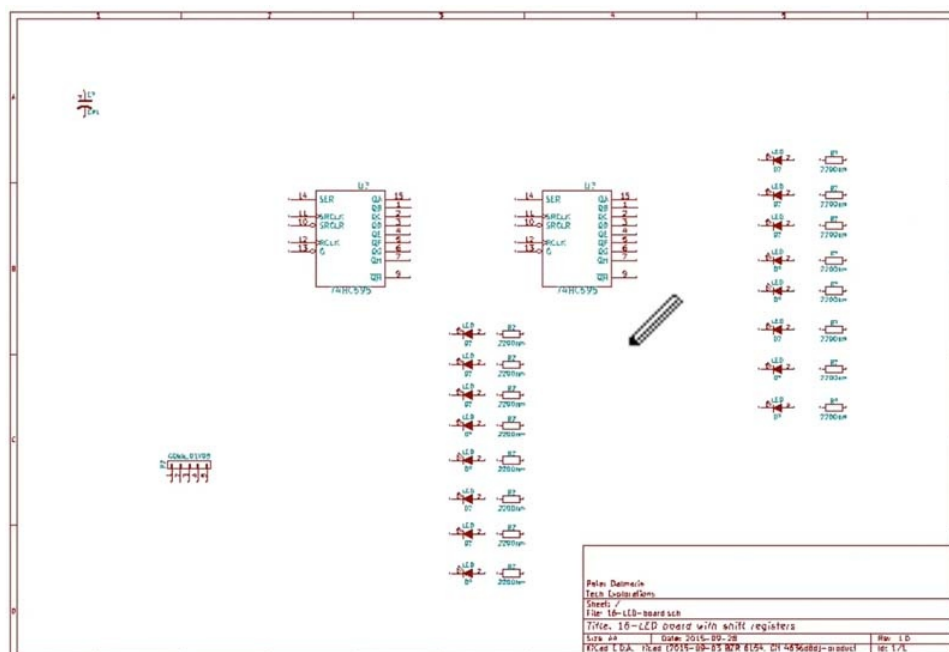
The capacitor is at the top left corner of the canvas.

The last thing we need is the connector. Like with the other projects, we will use a straight 1 x 5 connector. Find it in the component chooser:



The 1x5 connector in the component chooser

Like with the capacitor, we don't need to place the connector in close proximity to the rest of the components in the circuit because we will be using labels instead of wires to do the connections. Feel free to spread out your components to produce a balanced and easy to read schematic. Here is the current version of the schematic, with the connector at the bottom left corner of the canvas:



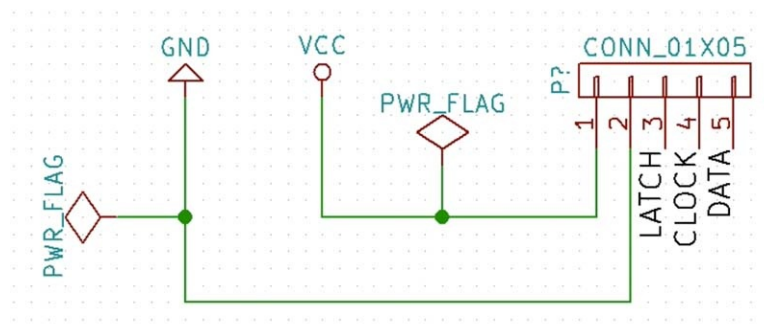
The current version of the schematic.

We now have all the components of the circuit on the canvas. In the next chapter, we will start working on the wiring.

Chapter 52: Schematic wiring, Part 1

In the previous chapter we added the circuit components to the canvas. In this chapter, we will start work on the wiring. We'll make full use of the techniques we learned in the previous two projects, and especially buses and labels.

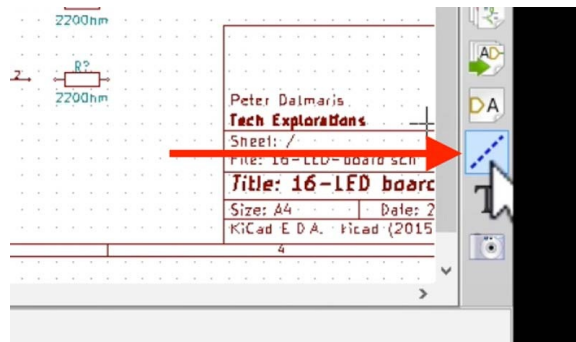
Let's begin with the connector. Consult the breadboard wiring diagram to find out the role of each pin. We will assign pin #1 to be connected to five volts, pin #2 will be connected to ground, pin #3 will be latch, pin #4 will be clock, and pin #5 will be data. So, let's put in a VCC component. So that will go say here, and I put in a ground component as well GND. So that will go rotated, that'll go here. And I can do the wiring, so W. Just start the wiring, and it will go around here to pin number one. Same thing for ground, click to start the wire and that would be connected to number two. This is very similar to how we configured the connector in project two, so I will show you the way the connector will look like on the schematic now:



The wiring of the connector.

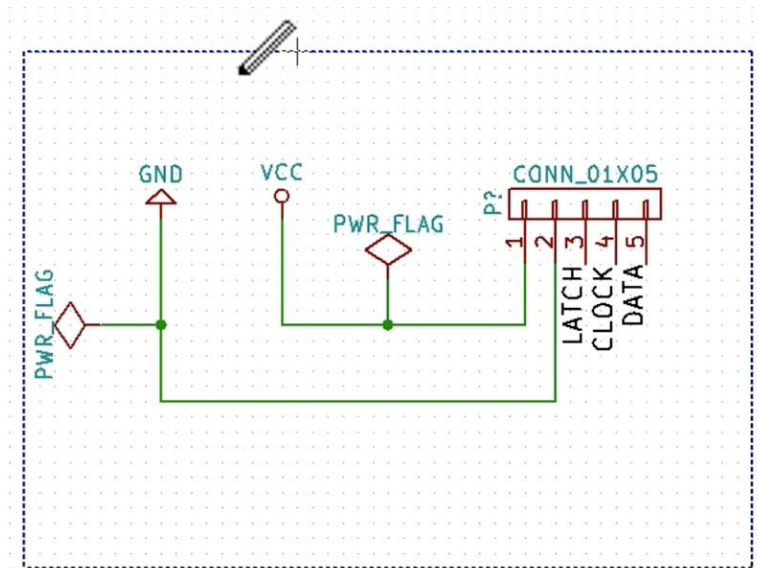
In the connector schematic, notice the GND, VCC and PWR_FLAG components, all added via the component chooser. We use labels for pins 3, 4 and 5, though which we will achieve connection with other parts of the circuit.

To make the connector part of the schematic easier to identify, I would also like to wrap it in a box, and give it a name. First, select the line tool from the right vertical toolbox:



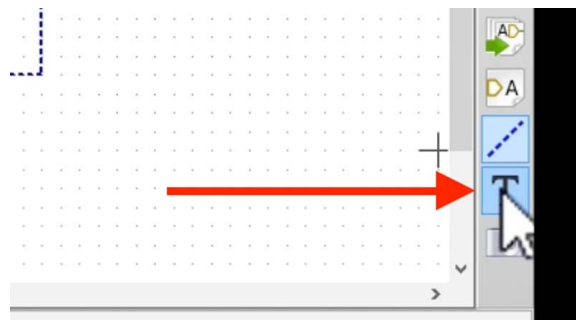
With the line tool you can draw boxes and other shapes.

Draw a box around the connector schematic:



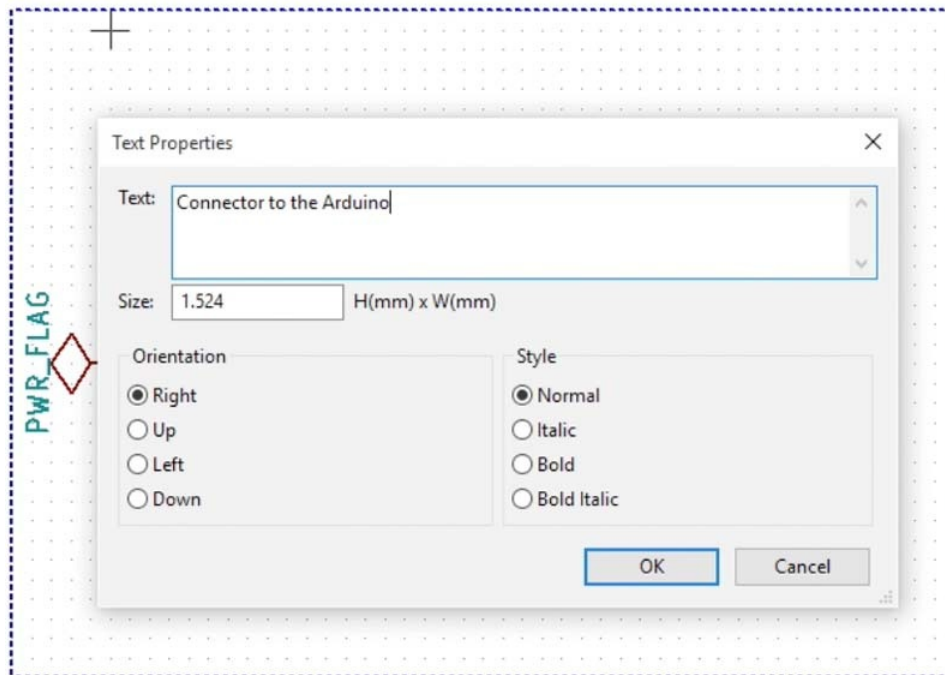
With a box outline, we can mark out parts of the schematic.

Add a name to this part of the schematic. Use the Text tool:



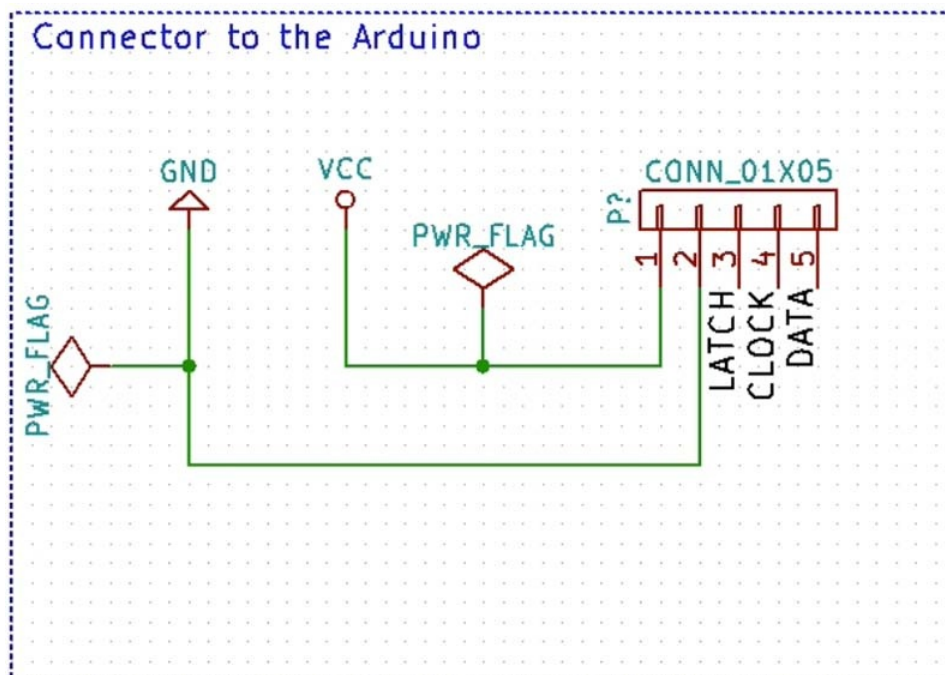
With the Text tool you can add arbitrary text to the schematic.

Click inside the box you just created, somewhere in the top right of the box. The Text properties window will come up. Type something like “Connector to the Arduino” in the text box:



Type some text in the text box, and click OK.

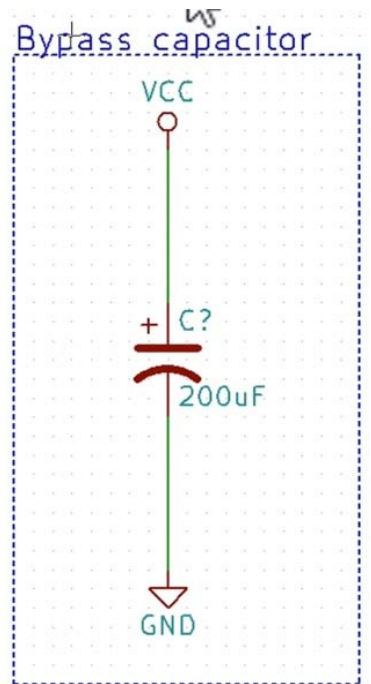
The text label you just created is now tethered to the mouse pointer. Place it in the box, so that at the end you have something like this:



This part of the schematic is now marked and titled.

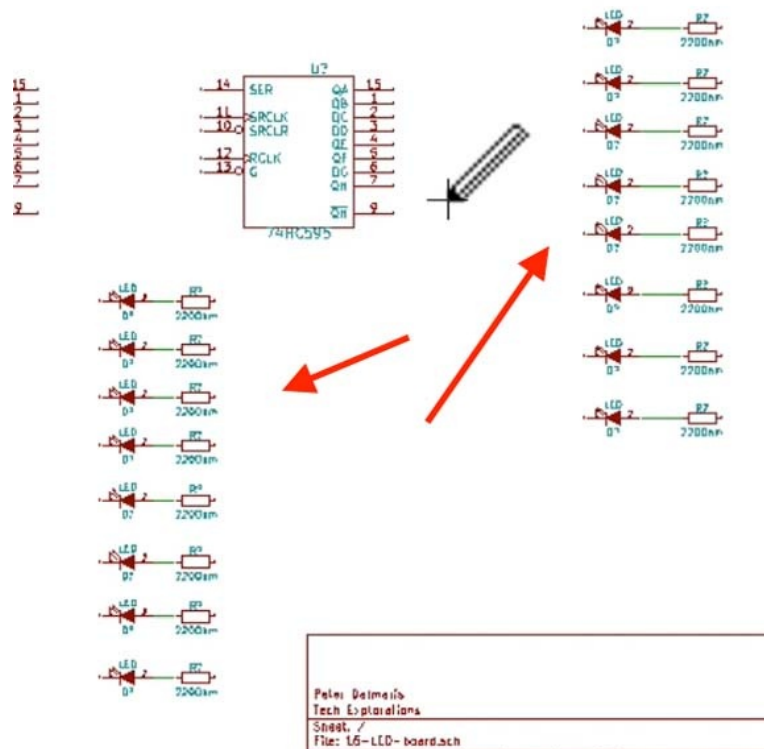
The box and the title we just added are not part of the circuit! They only serve in marking out special parts of the circuit. They just make the schematic easier to read.

Do the exact same thing for the capacitor. Add the VCC and GND components, place the area in a box, and give it a title. It should look like this:



The capacitor part of the schematic, boxed and titled.

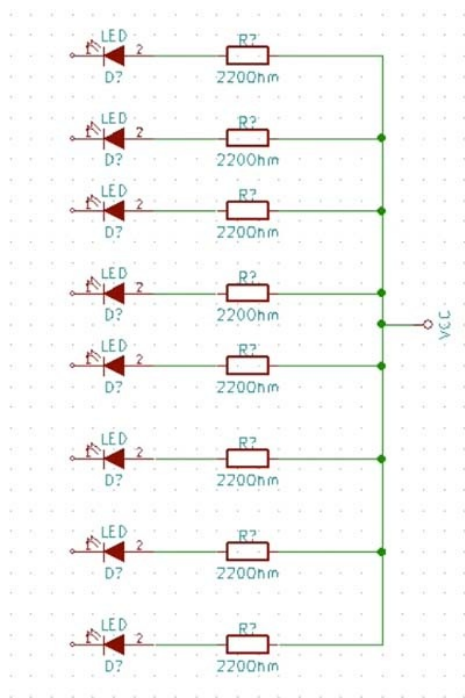
Next, let's switch our attention to the resistors and LEDs. Because of the way we positioned these components in the previous chapter, one next to the other in two columns, we can simply use wires to connect them. Type "W" to select the wire tool, or click on the Wire tool button from the right tool bar, and draw the wires.



Use normal wires to connect the LEDs to the resistors.

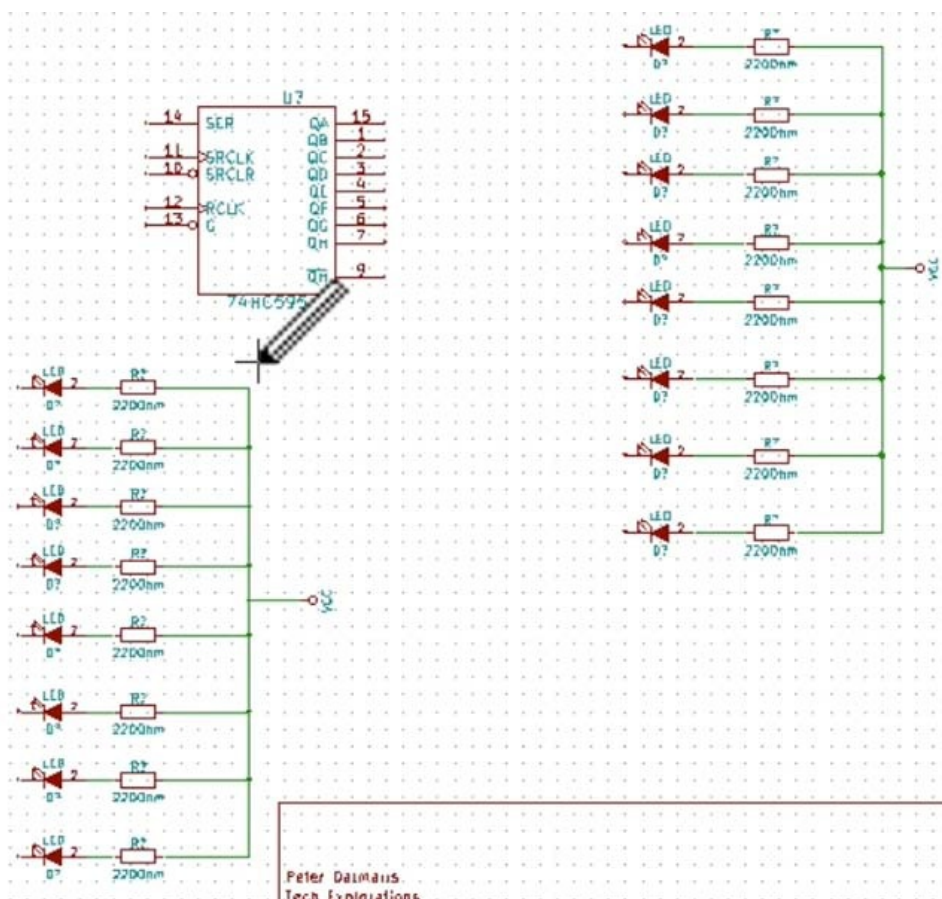
The circuit requires that the anode of each LED is connected to VCC. The cathodes will be connected to the shift register data pins. So, lets add the VCC component next, and

connect the resistors to it, for both LED banks. The first LED bank circuit will look like this:



Added a VCC component, and completed the connections with the LED anode, via the resistors.

Notice the solid green dots indicating that the wires actually connect instead of only intersecting. The two LED banks, completed, look like this:



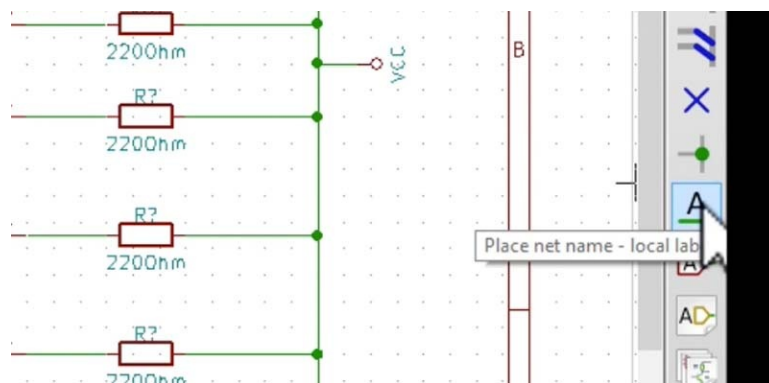
Both LED banks connected to the VCC component via the current limiting

resistors.

We'll continue with the data pins and wires. Because we have 8 data pins for each LED bank, it is preferable to bundle them together via buses. We will create one bus for each LED bank, and assign 8 data signals to each bus.

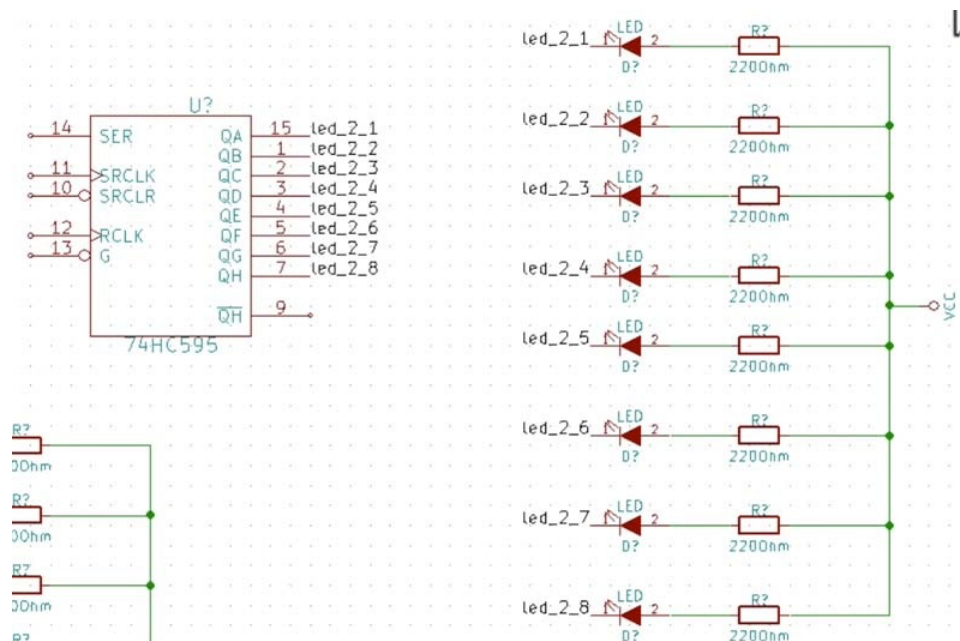
Start the process for the first bank by labelling the data pins on the shift register and the LEDs, then draw the bus, add the bus entries, and finish with the wiring between the bus entries and the pins.

Click on the Local Label button:



Select the Local Label button.

Next, add labels to pins QA to QH of the shift register, and the 8 LEDs that are across it. The labels that you create should contain the same name for each data pin and LED pair. The schematic should look like this:

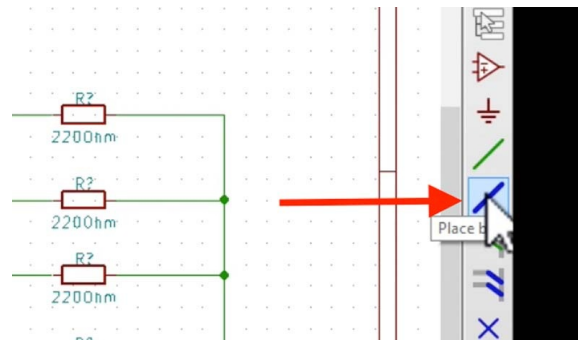


The data pins and LEDs are now labeled.

Notice how a label is structured according to a convention: “led_x_y”, where “x” is the number of the LED bank (in this example, we are working on bank 2 of 2), and “y” is the number of the data pin (1 of 8, 2 of 8 etc). Also notice that the same label is used for pins that are meant to be electrically connected. So, the shift register pin QA is meant to be connected to the LED at the top of the bank. The names of the labels will be used to name

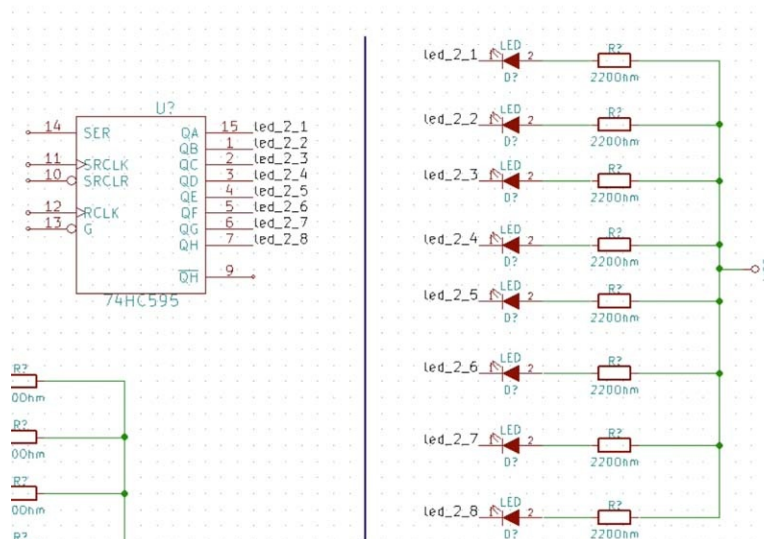
the corresponding net. Later in Cvpcb, we will see these nets again, so it is important to choose names that have meaning otherwise there is a good change you will get confused later!

Let's create the bus next. Select the bus too from the right tool bar:



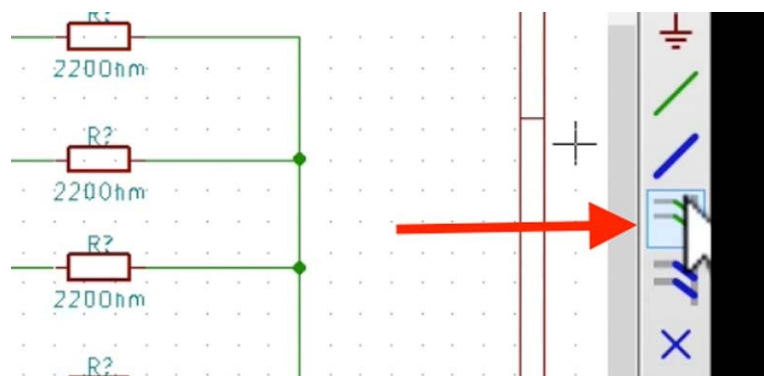
The bus tool button.

Draw a bus as a single line between the shift register IC and the LEDs:



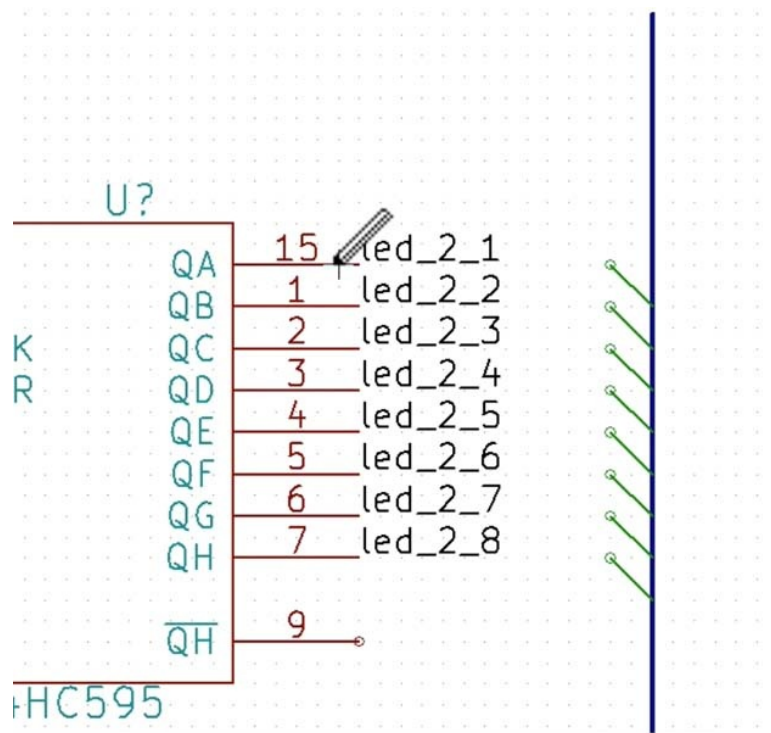
The blue line represents a bus.

Select the bus entry tool from the right tool bar:



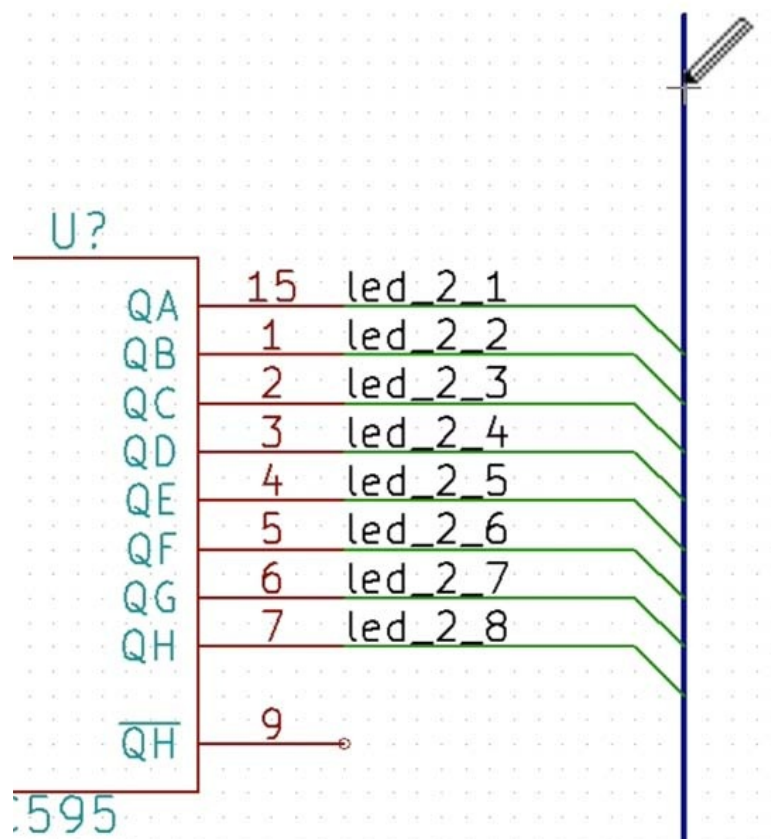
The bus entry tool.

And create entries on the IC side first:



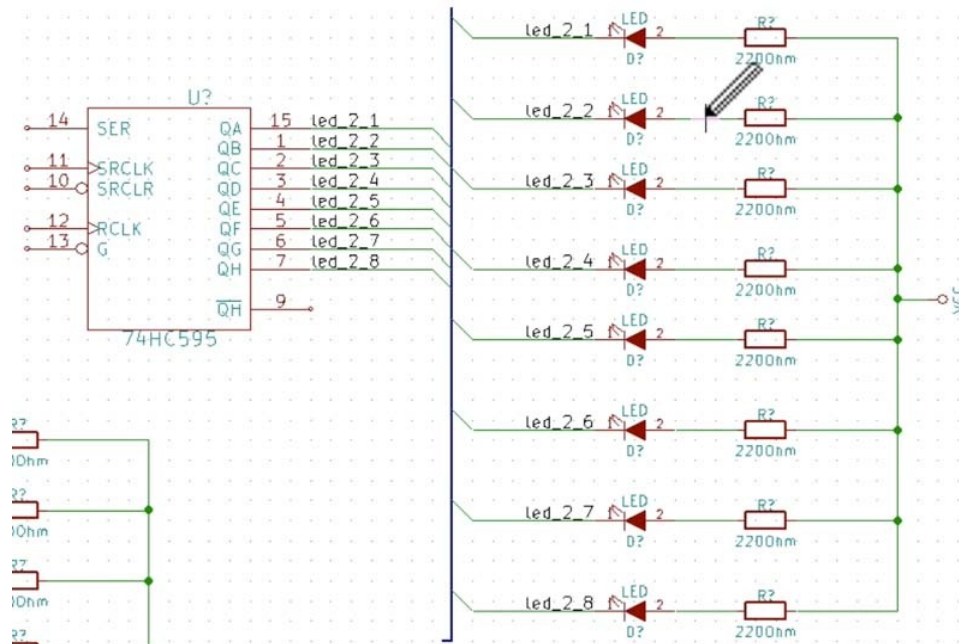
Added bus entries. They are very similar-looking to normal wires, except that they are at a 45 degree angle.

Use a normal wire (“W” key) to connect each pin of the shift register with a bus entry:



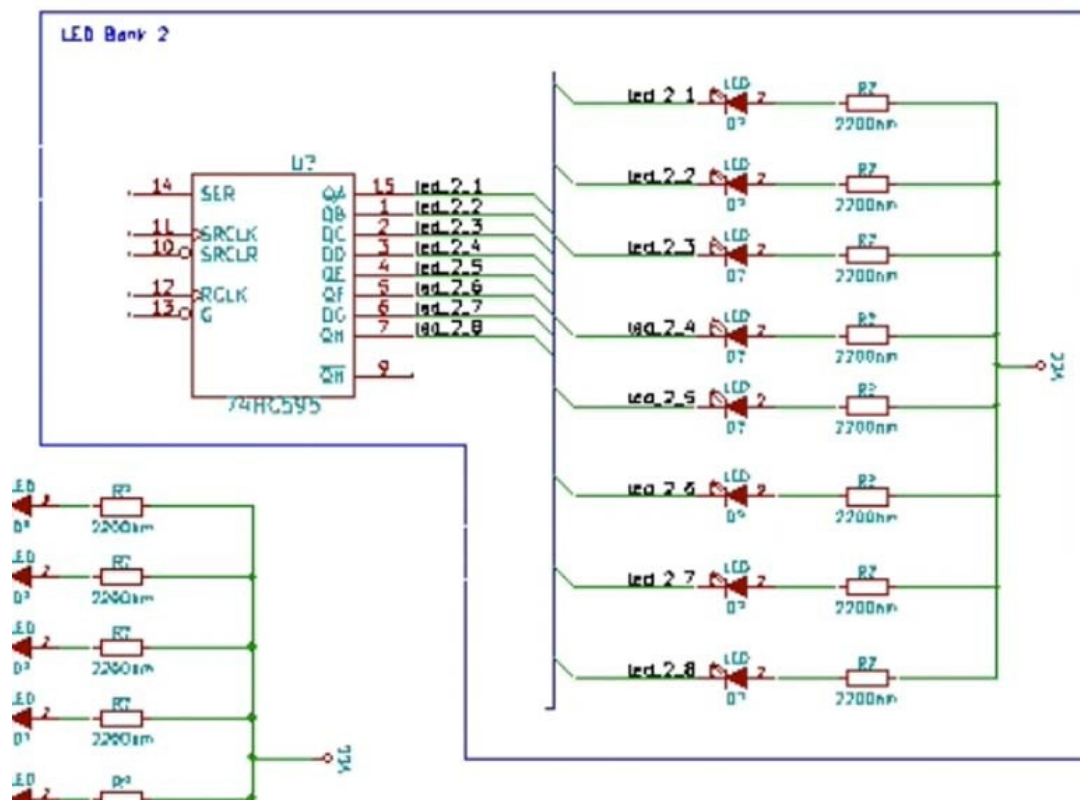
The pins are now connected with a corresponding bus entry.

Repeat the process on the LED side: add bus entries and connect them to the LEDs:



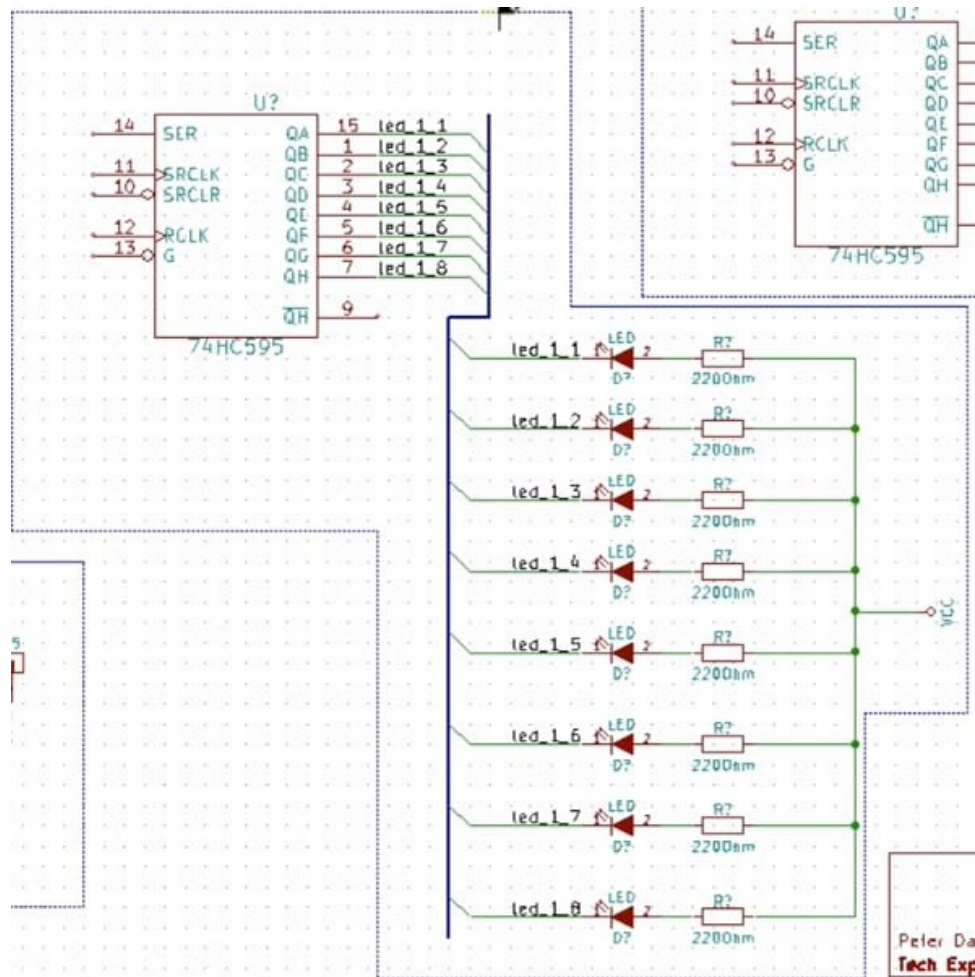
The LEDs are now connected to the bus.

The shift register is now fully connected to the LEDs. Draw a boundary around this part of the schematic, and give it a name, like “LED Bank 2”:

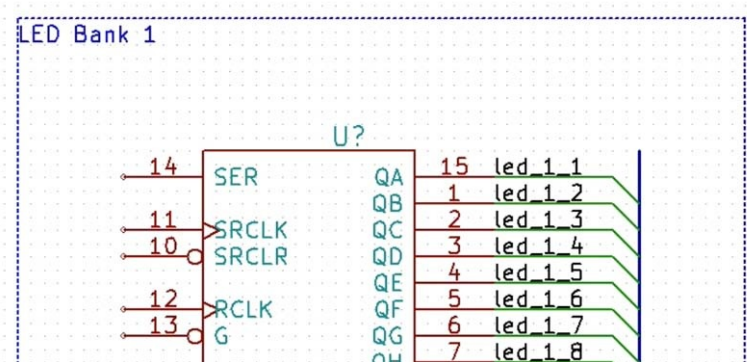


This part of the schematic highlighted and titled.

Repeat the exact same process for the other shift register IC and its LEDs. To avoid boring you, I will just show you the end result:



This is the LED Bank 1 segment of the schematic, complete with a bus, bus entries, net labels and wires.



Detail, showing a text label with the name of this segment for the schematic.

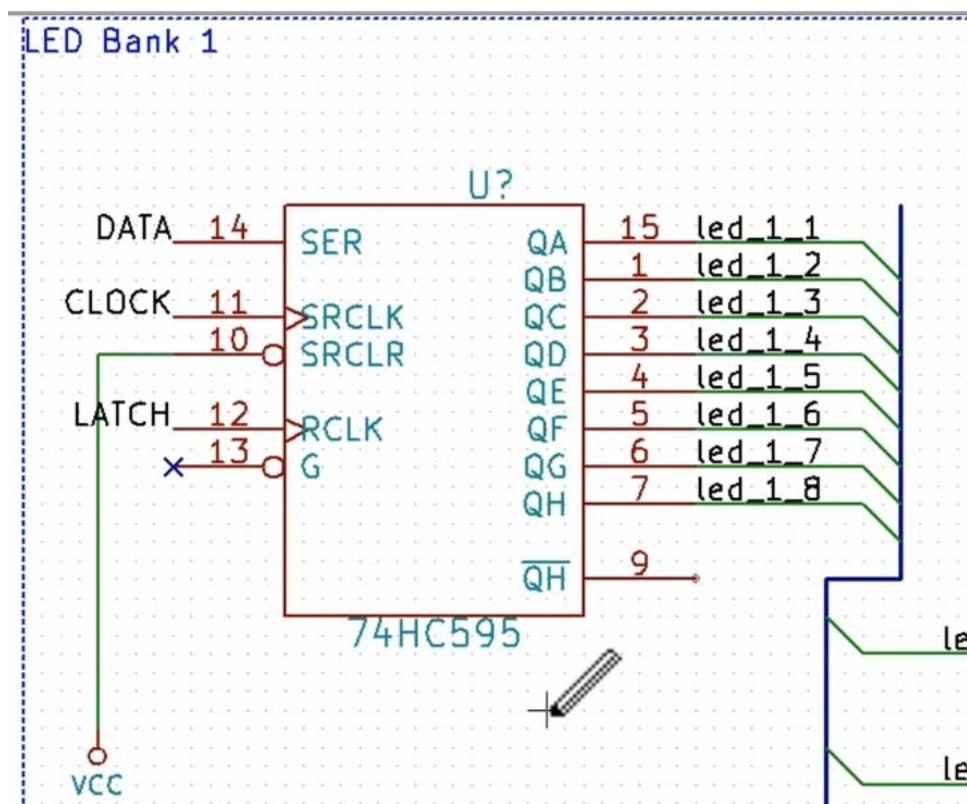
We have made substantial progress so far: the shift registers and the LED as connected via buses, and the connections to the capacitor and the connector are complete.

Next, we need to complete the circuit by connecting the shift register communications pins to the connector. We will use net labels for this. Let's proceed to the next chapter and to get this done.

Chapter 53: Schematic wiring, Part 2

In this chapter we will continue where we left of last time. We will complete the wiring of the shift registers.

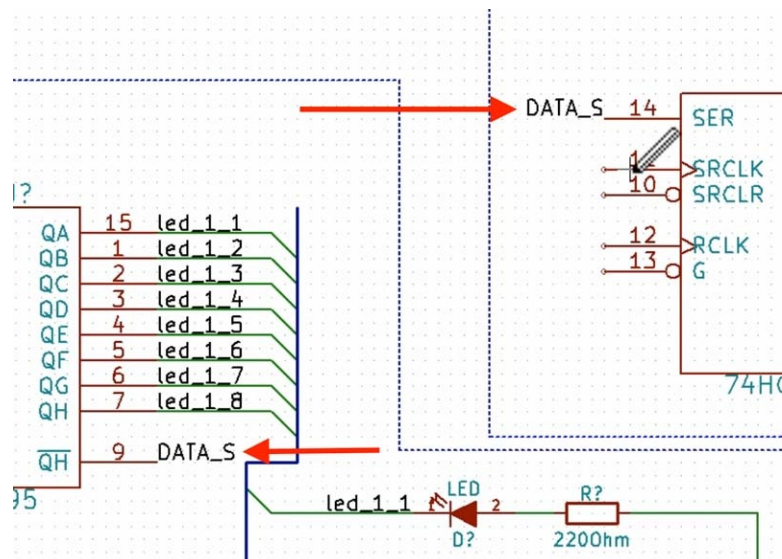
The shift register that is connected to LED Bank 1 will be wired directly to the connector. We will start by adding net labels. Pin #14 is data, pin #11 is clock, pin #12 is latch, pin #10 is Vcc, and pin 13 will be left unconnected. Here's the result:



Added labels to pins 14, 11, 12, connected pin 10 to VCC, and marked pin 13 as unconnected.

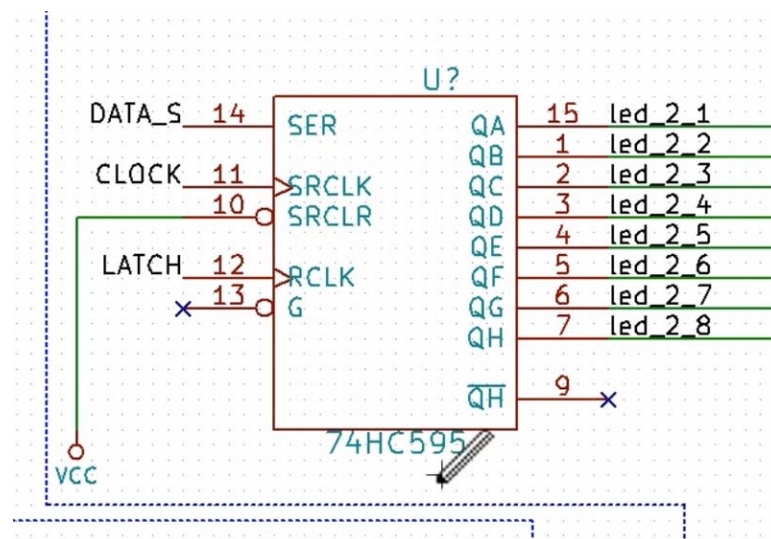
On to the second shift register now. The second shift register receives its data in pin 14 from pin 9 of the first shift register. We could use a wire to do this connection, but the schematic will look better with a label, so let's do that. Add the net label "DATA_S" ("S"

stands for serial) to both pin 9 of the first shift register, and pin 14 of the second shift register. The end result is this:



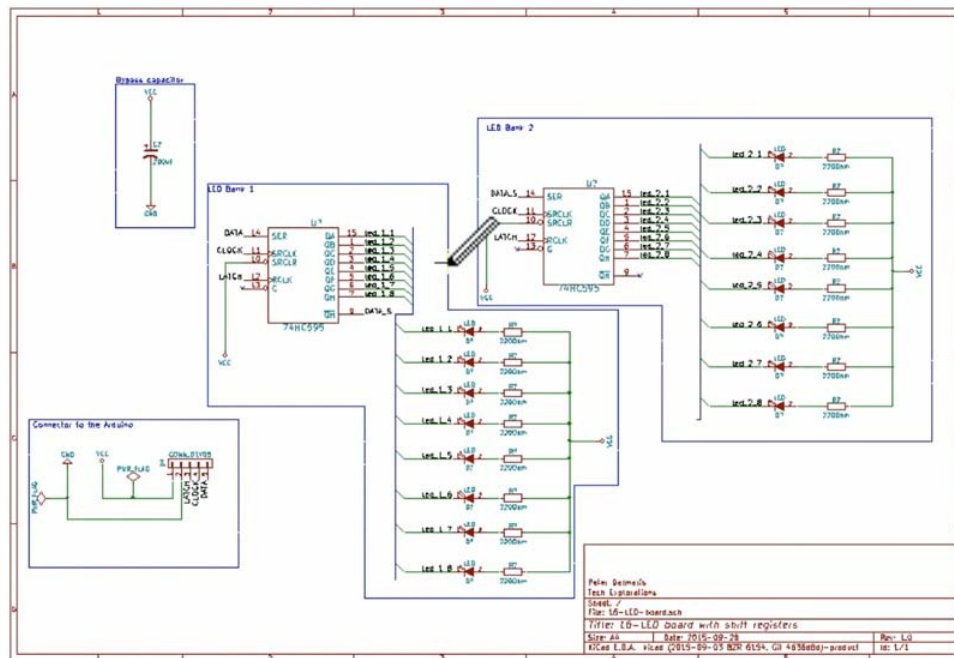
Pin 9 of register 1 and pin 14 of register 2 are connected via net labels.

Let's finish the labelling of shift register 2. Create labels, as you did for shift register 1. In addition, attach the Unconnected component to register 2 pin 9. Eventually, register 2 will look like this:



Register 2 labelling complete!

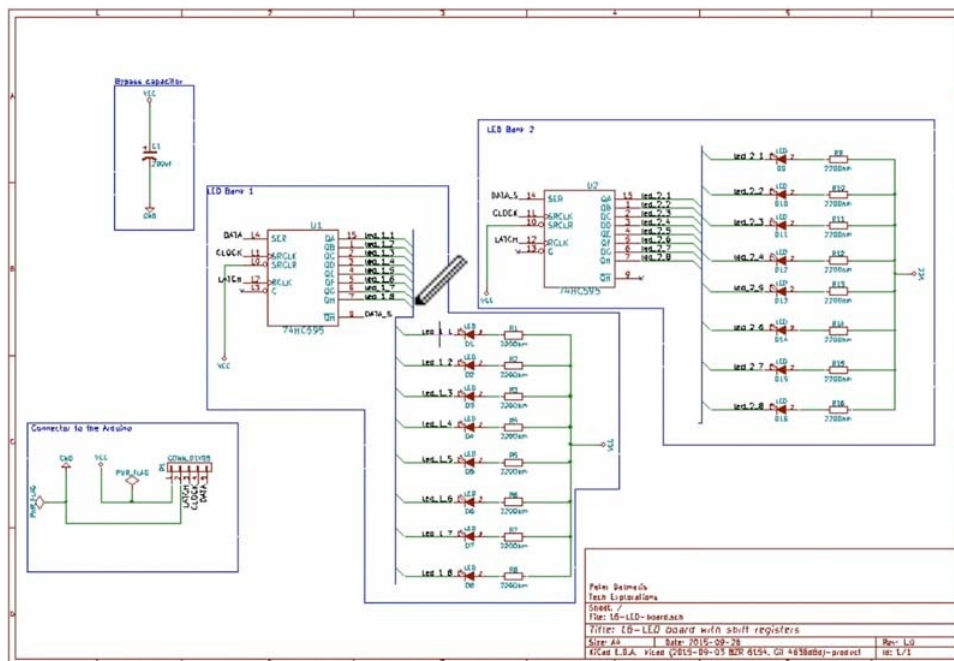
With this, the wiring of the circuit is complete. Save the schematic, and zoom out to see what we have:



The complete 16-LED project schematic.

We have got all the LEDs with their current limiting resistors, connected to VCC on side, and then via a bus to each one of the shift registers. We have the connector down the left side, and we've got the bypass capacitor at the top.

Let's do the annotation now. Click on the annotate schematic components button. The default settings are good, so we can accept them as they are. Click on Annotate.

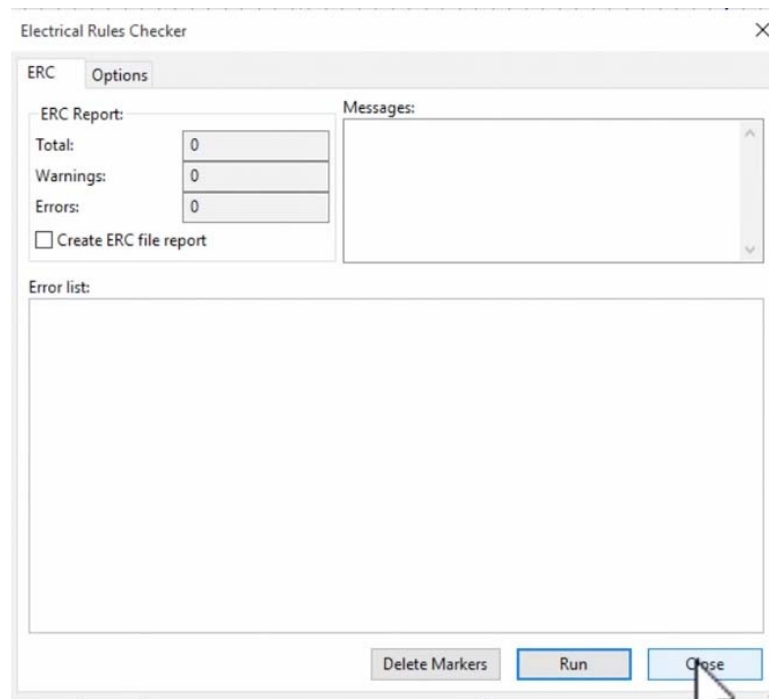


The schematic is now annotated.

All the components now have a unique designator. The connector is P1, the first shift

register U1, the second U2, and then all the LEDs also have got their own designators. Save the project again.

Finally, let's do an electrical rules check:



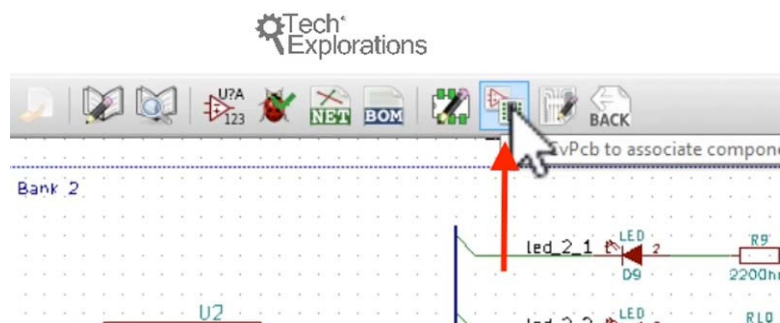
Nothing to worry about!

Excellent, no errors!

In the next chapter we'll do the component-footprint associations and export the netlist for Pcbnew,

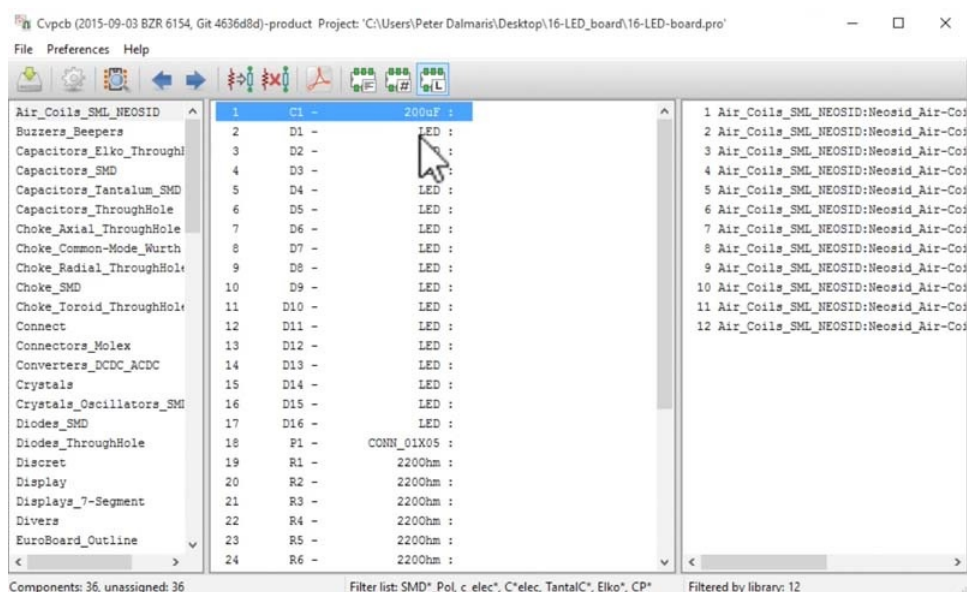
Chapter 54: Associate components with footprints

In this chapter we'll create the component-footprint associations and then we will export the netlist file. Let's begin with starting Cvpcb. If you are still in Pcbnew, click on the Cvpcb button:



Start Cvpcb.

You will see the Cvpcb window, with the middle pane containing the components from the schematic.



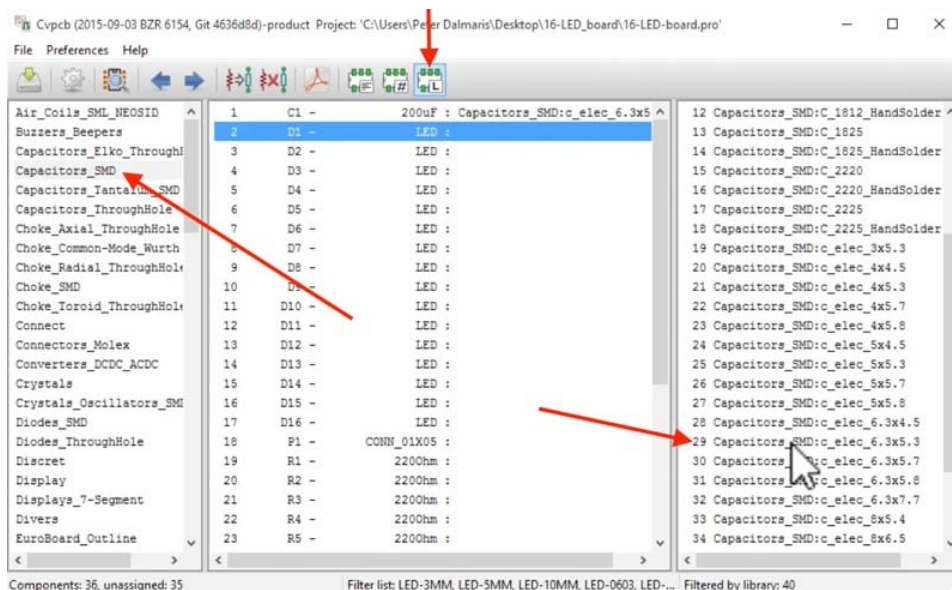
Cvpcb lists the schematic components in the middle pane.

We can start with the capacitor. For the capacitor, we're looking for an SMDs type package. The one I would like to use is an electrolytic SMD aluminium capacitor, measuring 6.3 mm by 5.3 mm, like this one:



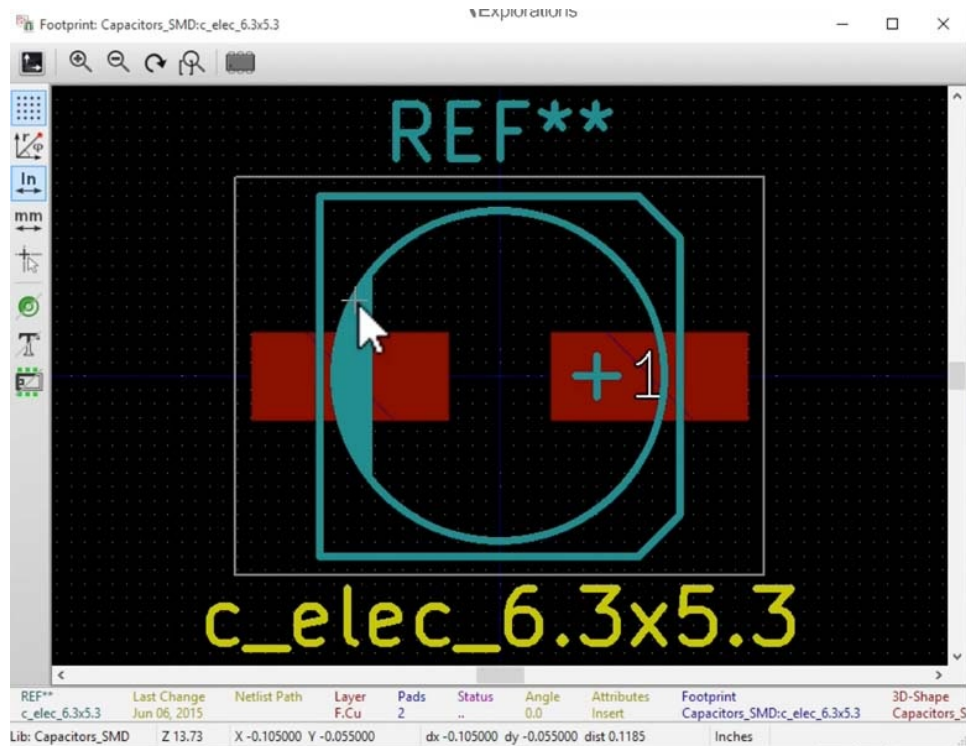
An electrolytic SMD capacitor.

Select the Capacitors_SMD library from the left pane, and enable the Library filter. In the right pane there is a long list of candidate footprints. Scroll down until you find one that measures the correct dimensions:



Select the library, filter, and look for the right footprint.

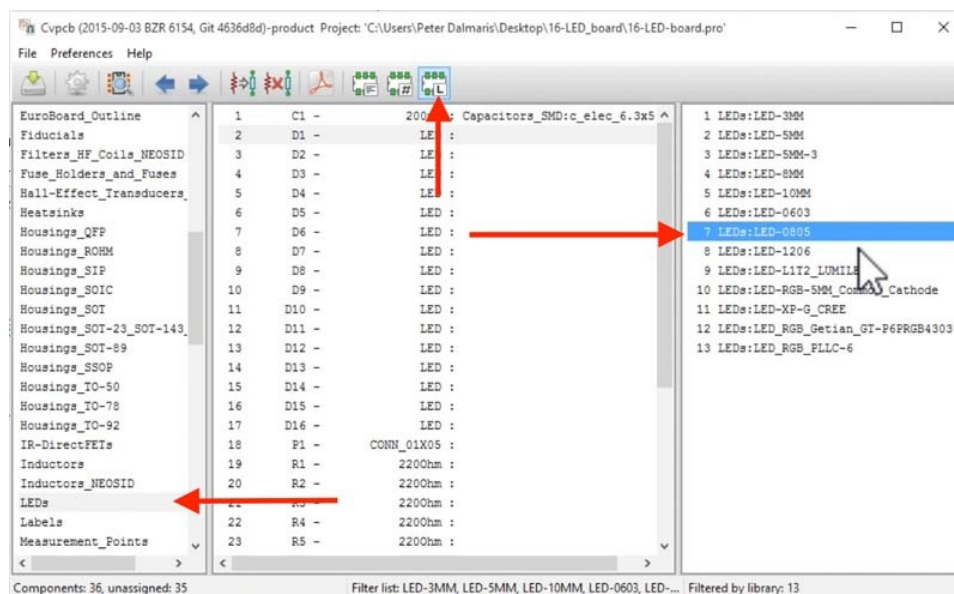
Confirm that you have the right footprint by looking at its preview:



This is the footprint of an SMD electrolytic capacitor.

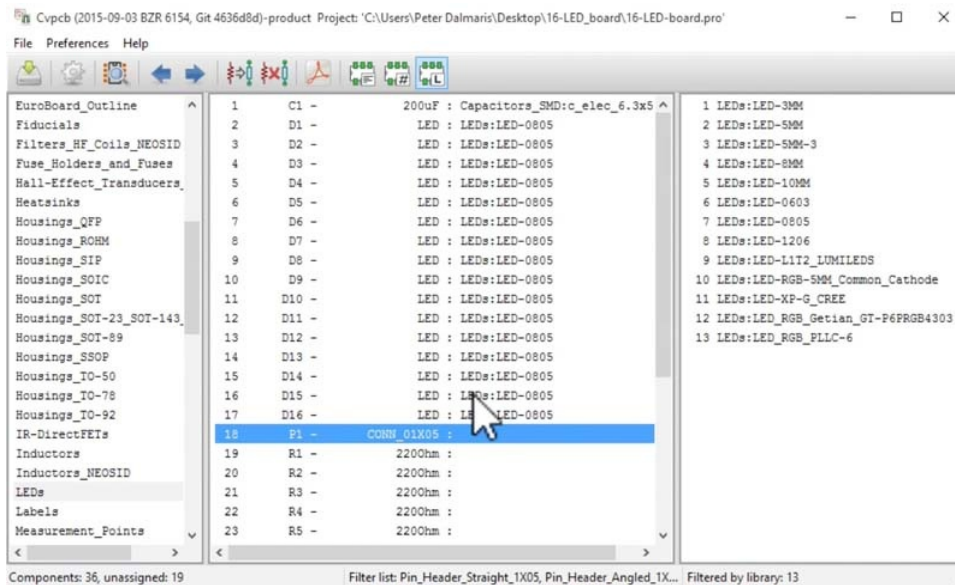
Double click to select this footprint and associate it with the capacitor component.

For the LEDs, again I am looking for an LED surface mounted device of 0805 type. Select the LEDs library, and inspect the list in the right pane.



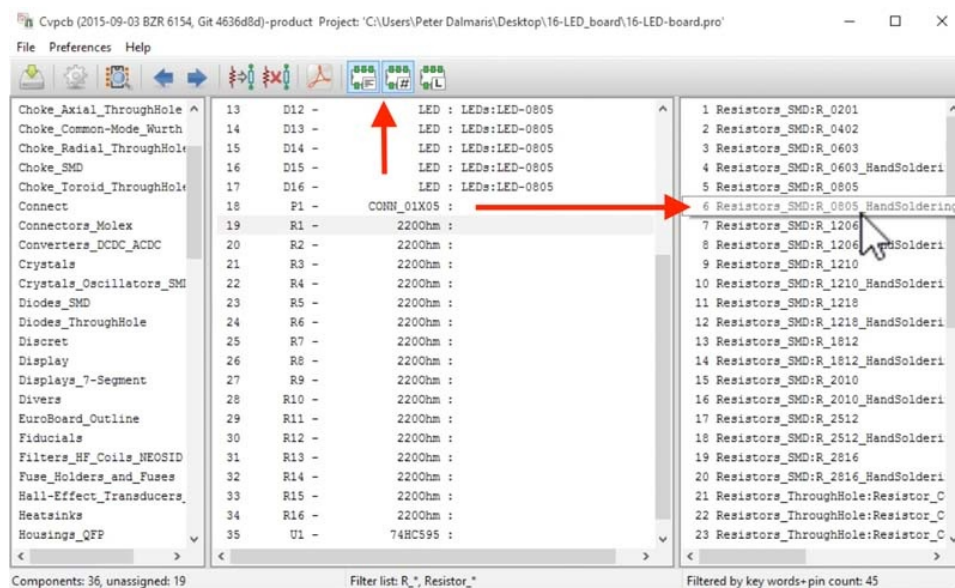
Candidate footprints for the LEDs.

There are several options for 3mm, 5mm, and 0805 here, among others. I am looking for the 0805, so double click on that to select it. Continue to double click until all LED have the 0805 footprint associated.



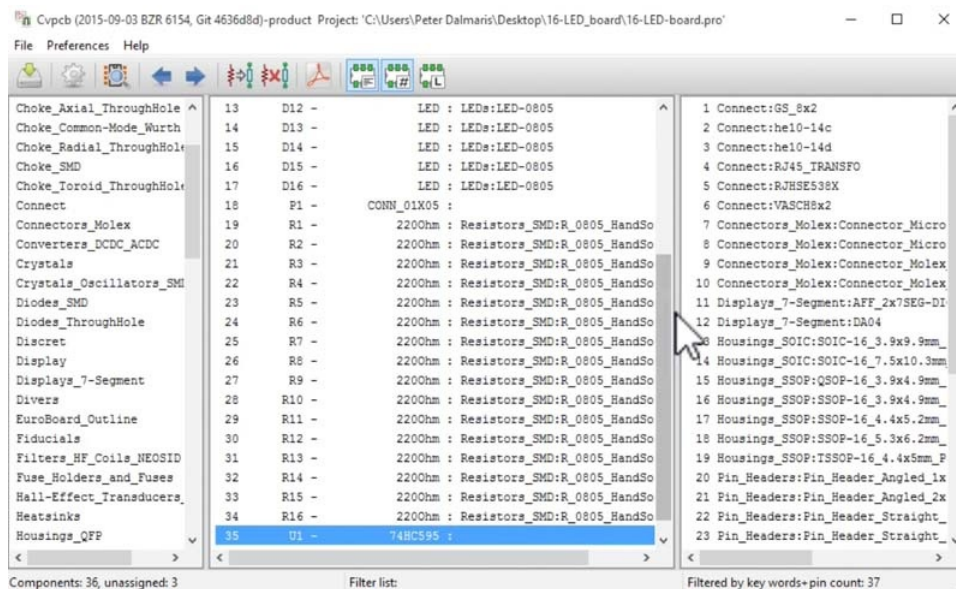
The LED components are associated with the 0805 footprint.

Let's skip the connector and work on the resistors next. Instead of selecting a library first, I will use the keyword and pin count filters to find the right footprint.



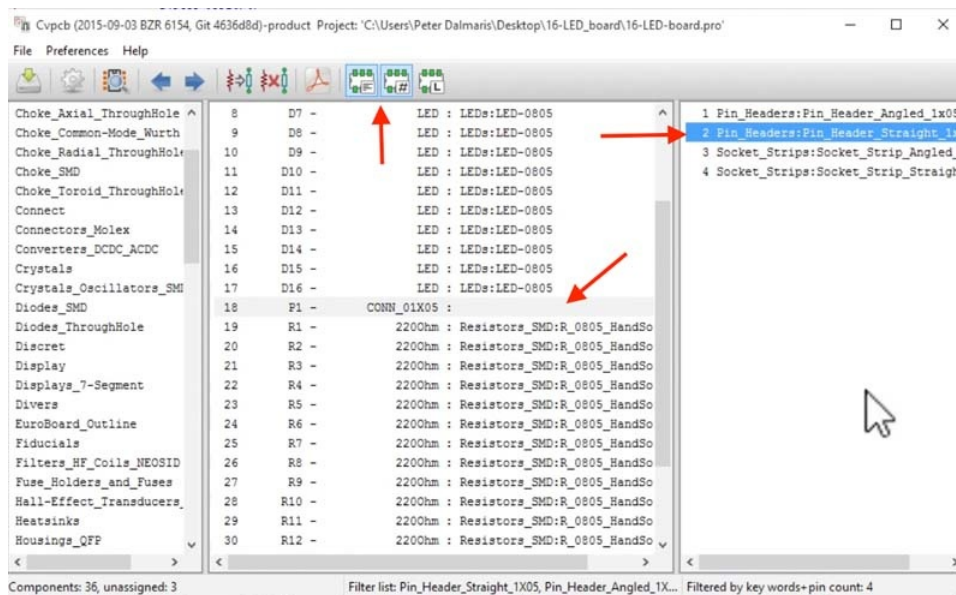
With the keyword and pin count filter, I can quickly find the correct Resistor footprint.

Look for the 0805 type resistor, and double click several times to associate this footprint with all of the resistor components. Here is the end result:

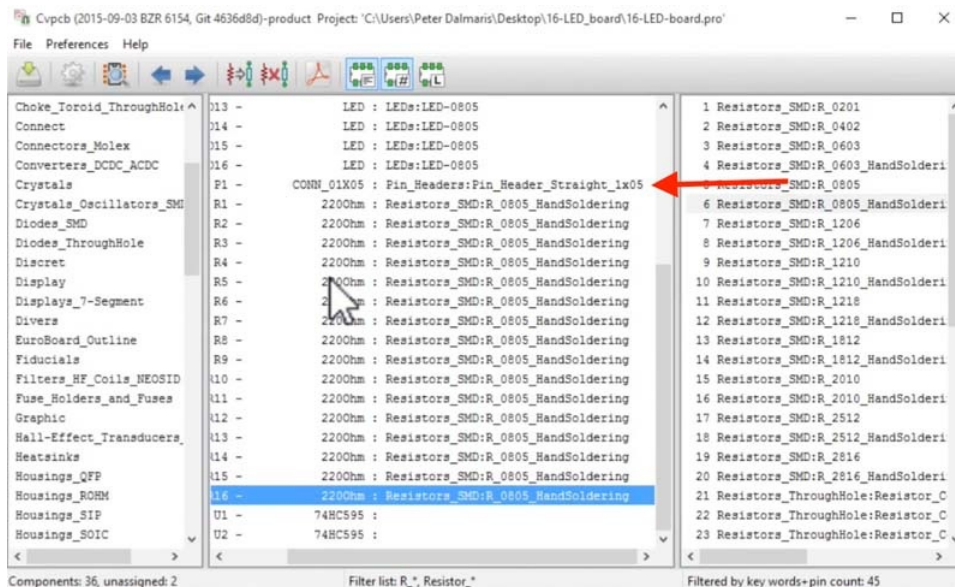


The resistor components are now associated with the 0805 footprint.

Let's continue with the connector. Click on the CONN_01X05 component in the middle pane, and with the filter settings unchanged, the suggested footprint in the right pane are just four. I'd like to use the straight header connector, so double click on the second option to select it.

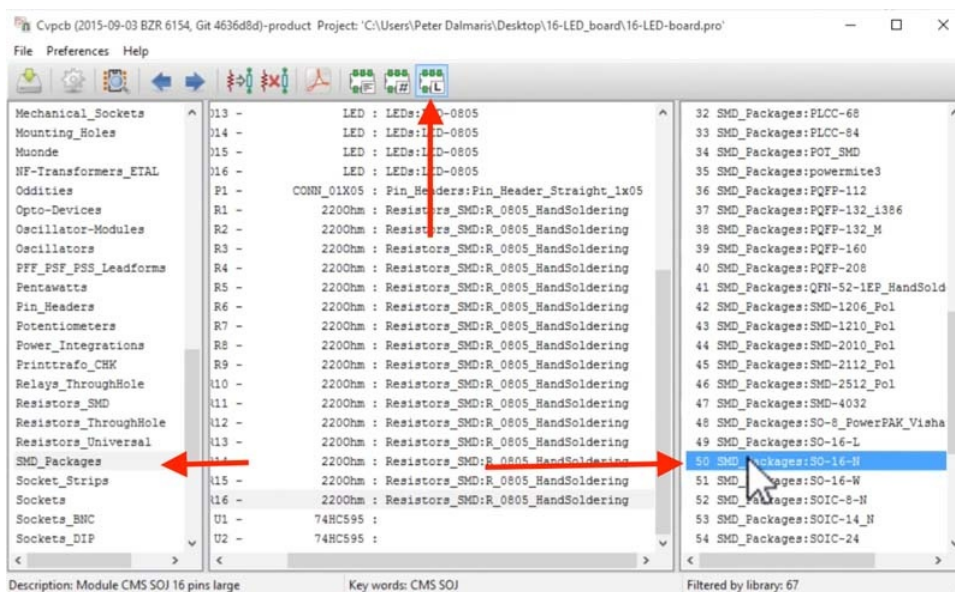


The options of the connector.



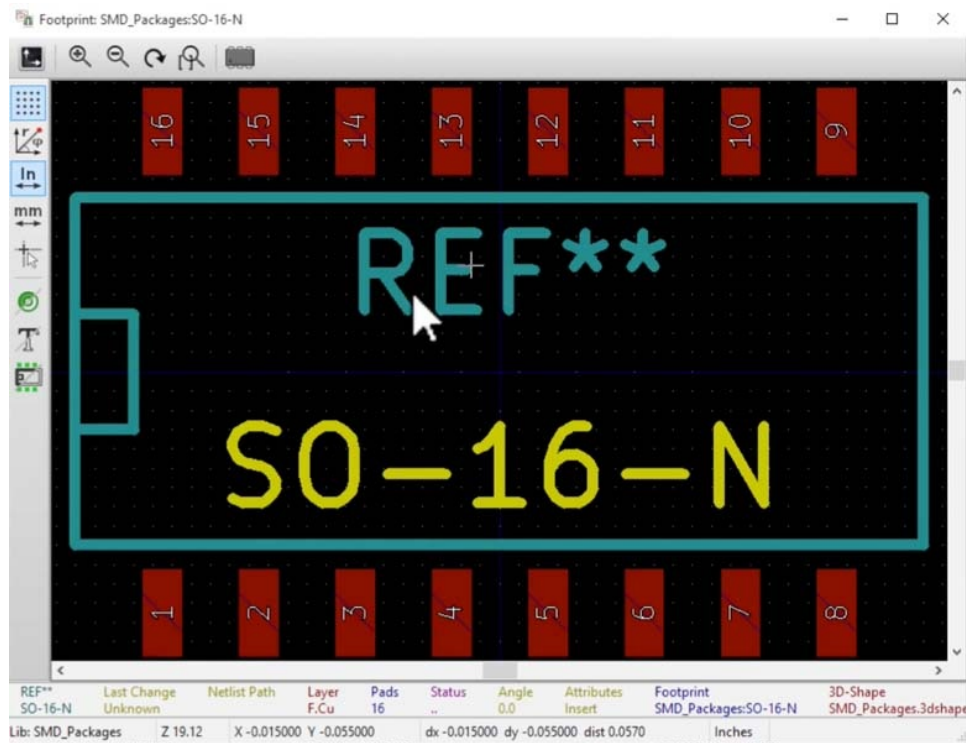
The connector component is associated with the straight header.

Lastly, for the shift register, we need to look in the SMD package to find the appropriate footprint. Click on the library filter button, and click again on the other two filters to disable them. This part is more complicated than the resistor and capacitor, so I prefer more control over the browsing process. The integrated circuit I would like to use on the board comes in an SO16 package, so we need to find one of them in the right pane.



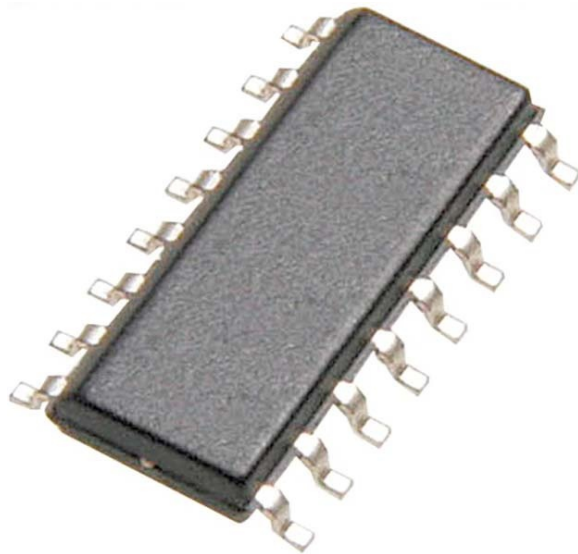
Only the Library filter is used for the search for the SO16 package.

The correct footprint for our component is most likely the SO-16-N option. But because there are three varieties of this footprint, it is better to make sure we have the right one. Let's bring up the footprint preview:



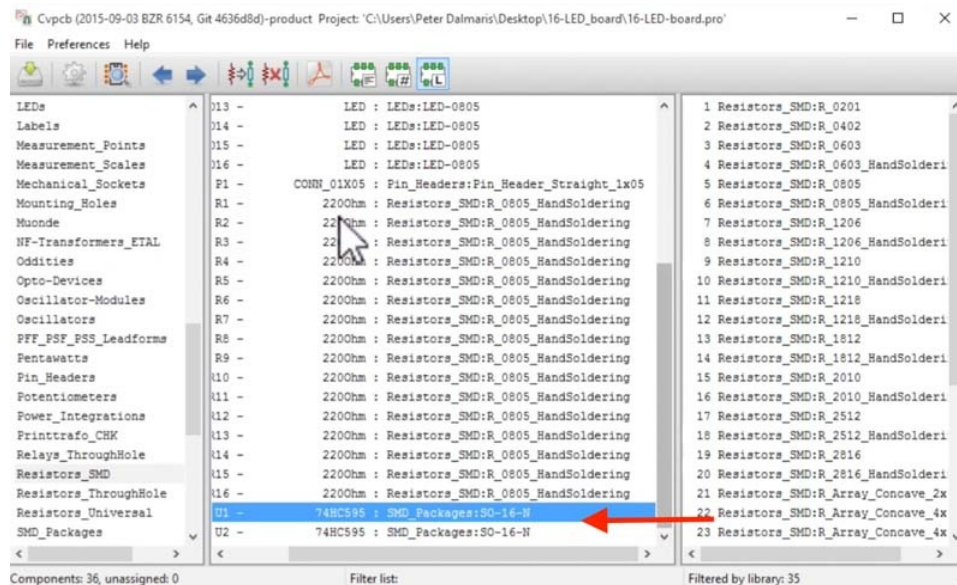
The SO16-N footprint preview.

Measure the dimensions between the pads, between the rows, and of the package itself. The actual part that I have in my toolbox is this:



The actual IC I would like to use comes in an SO16 package like this one.

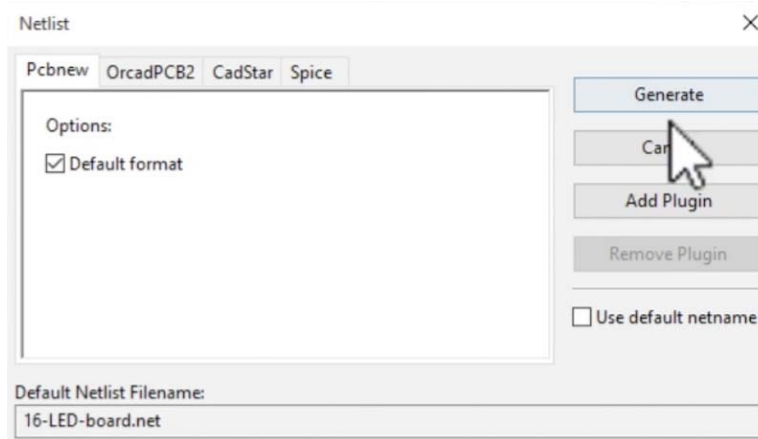
I used my rule to measure the distances and found that the pitch between the pads is 1.27mm and that the distance between the two rows is 6.35mm. Use the measuring tool in the footprint previewer to make sure the the footprint you are about to select has the same measurements. This is the correct footprint, so double click to select it for both ICs:



The shift registers are associated to the correct footprint.

We now have all our components associated with their footprints, so we can save the associations and return to Eeschema.

Before we close this chapter, let's export the netlist file, store it in the project folder:

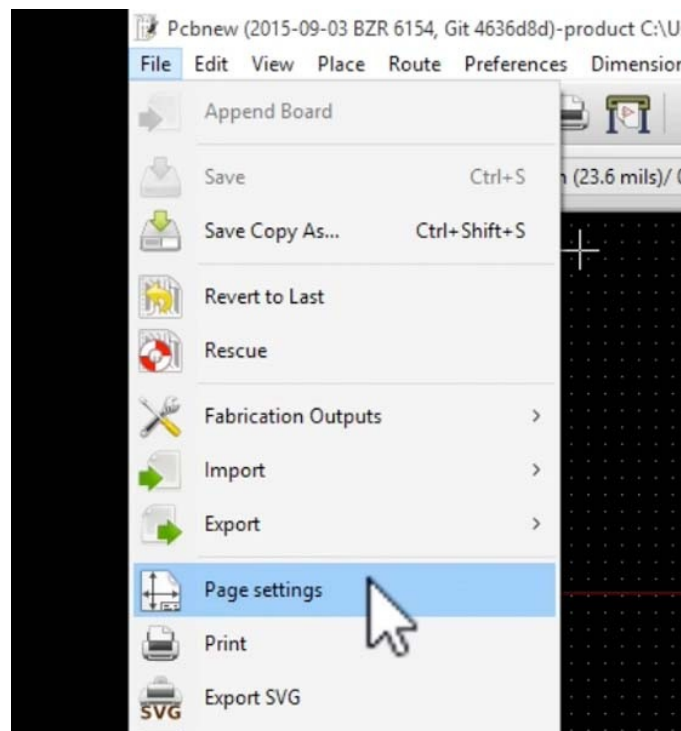


Generate the netlist file.

Let's continue with Pcbnew now and work on the layout and the wiring.

Chapter 55: *Create the PCB in Pcbnew*

After creating the netlist file in Eeschema in the previous chapter, we'll move into Pcbnew to work on the layout, the wiring, and produce the final pcb. Start Pcbnew and fill in the page details details. Bring up the Page Settings window by clicking on File and Page Settings:



Bring up the Page Setting window.

Fill in the page and project information, like in this example:

Page Settings


Paper

Size:
A4 210x297mm

Orientation:
Landscape

Custom Size:
Height: 279.40 Width: 431.80

Layout Preview



Title Block Parameters

Issue Date
2015-09-28 <<< 28/09/2015

Revision
1.0

Title
16-LED board with shift registers

Company
Tech Explorations

Comment1
Peter Dalmaris

Comment2

Comment3

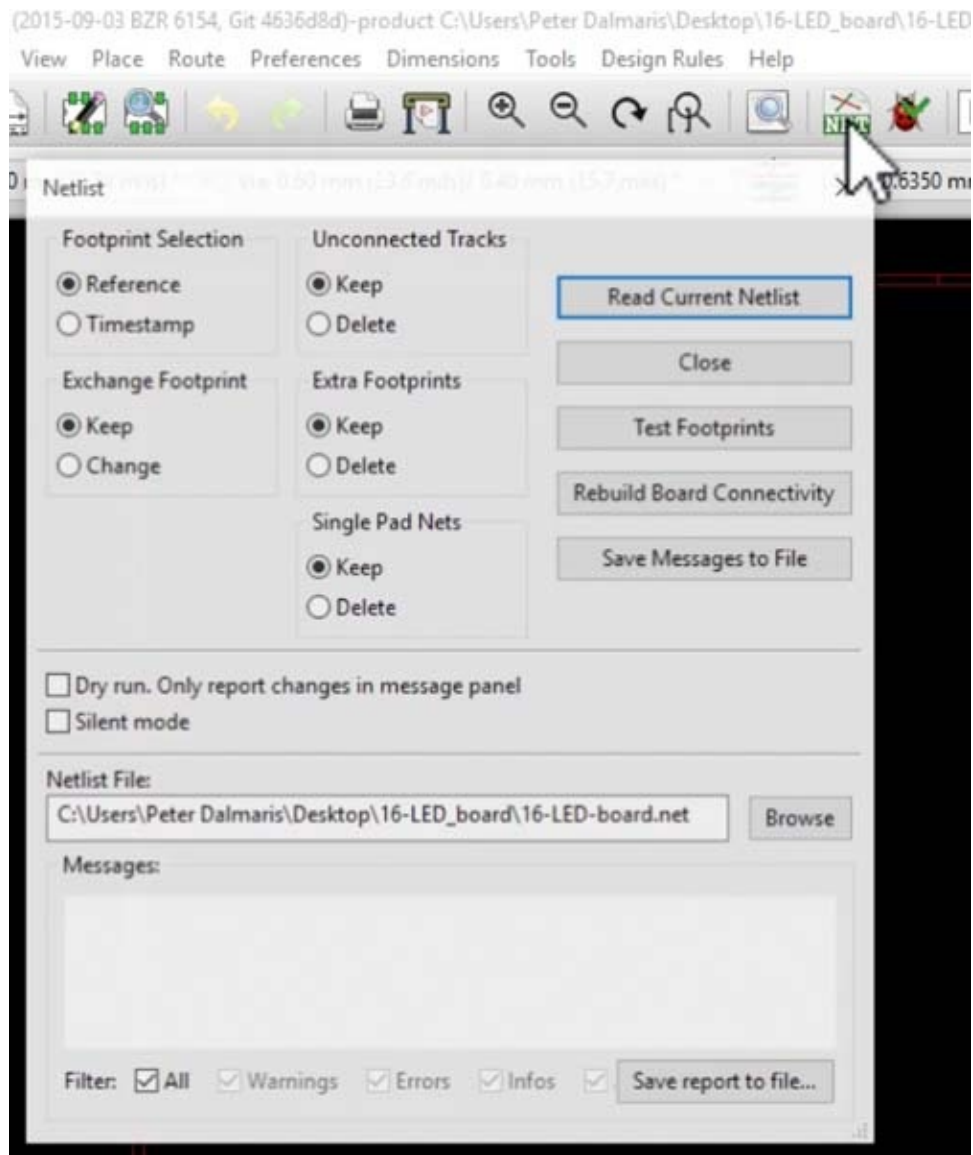
Comment4

Page layout description file
Browse

OK Cancel

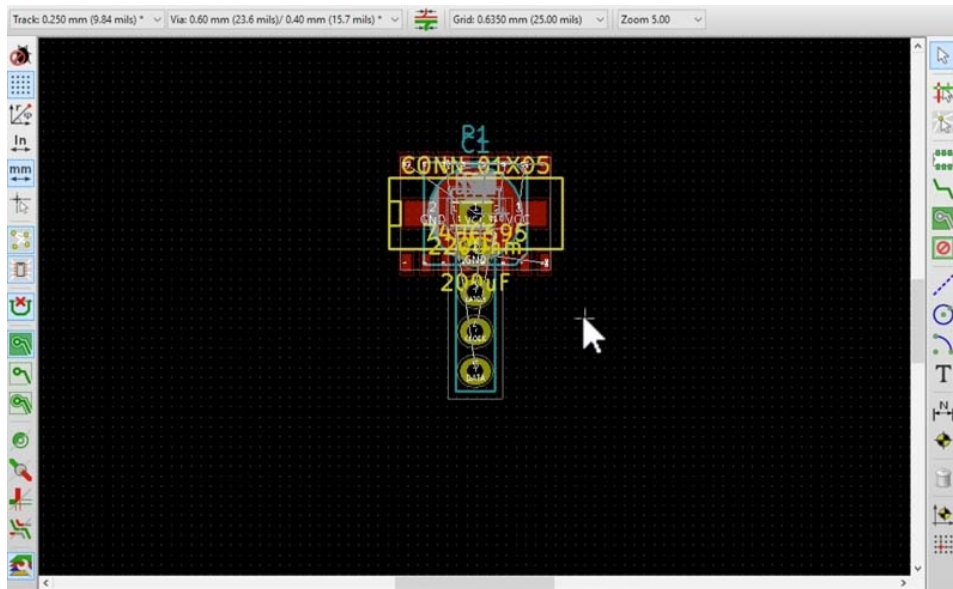
The page and project information, entered.

Now, let's go ahead to read the netlist file.



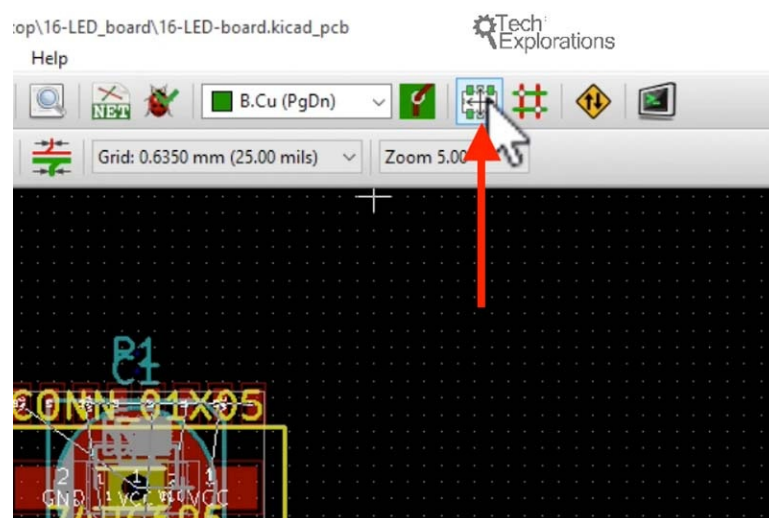
Read the netlist file. The path to the file should be correct by default.

The default path should be correct if you saved the netlist file in the project directory. Click on the Read Current Setlist file to read it, and then close the window (click on the Close button). The footprints will be bundles together on the canvas. You will need to zoom in and pan to be able to distinguish the individual components in the bundle.



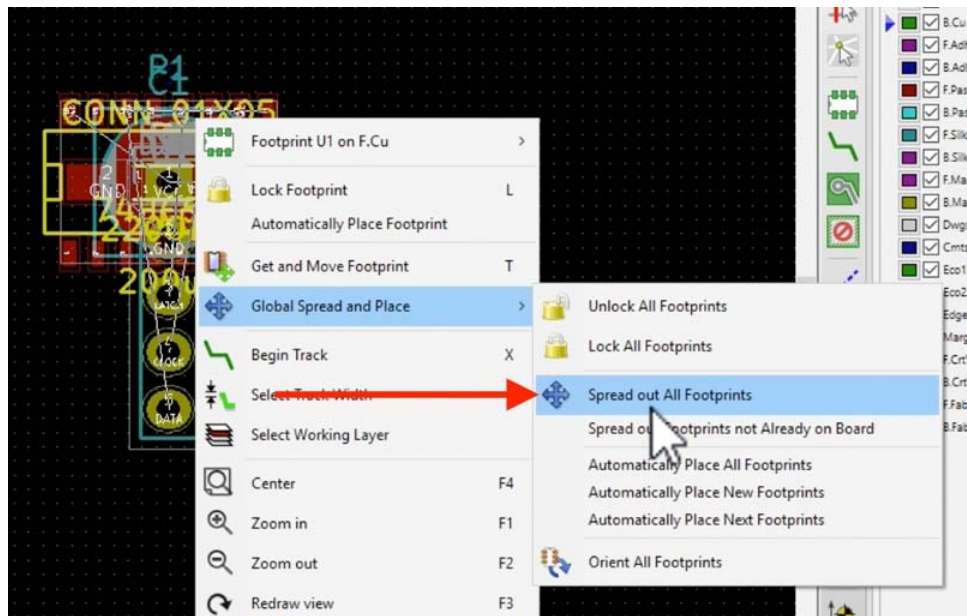
The footprints are placed in the canvas, in a bundle.

Picking and moving individual footprints is too much trouble, so let's get KiCad to help us out here. Click on the Mode Footprint button to enable automatic mode.



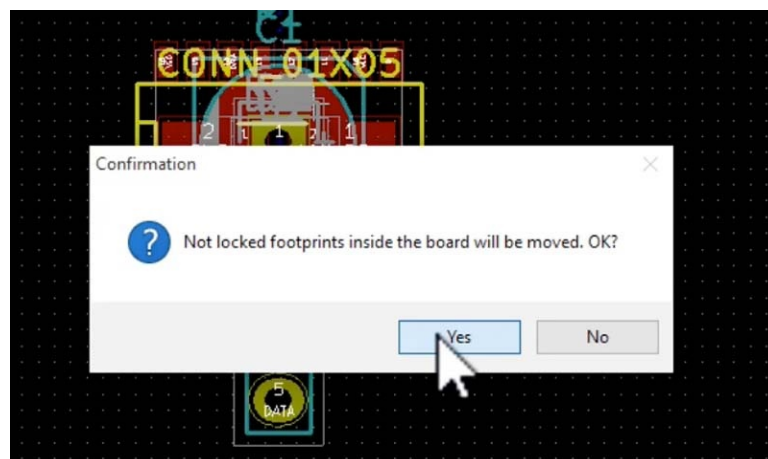
Enable Automatic Footprint mode.

Then click anywhere on the canvas and select Spread out All Footprints, to do what this items says it will do!



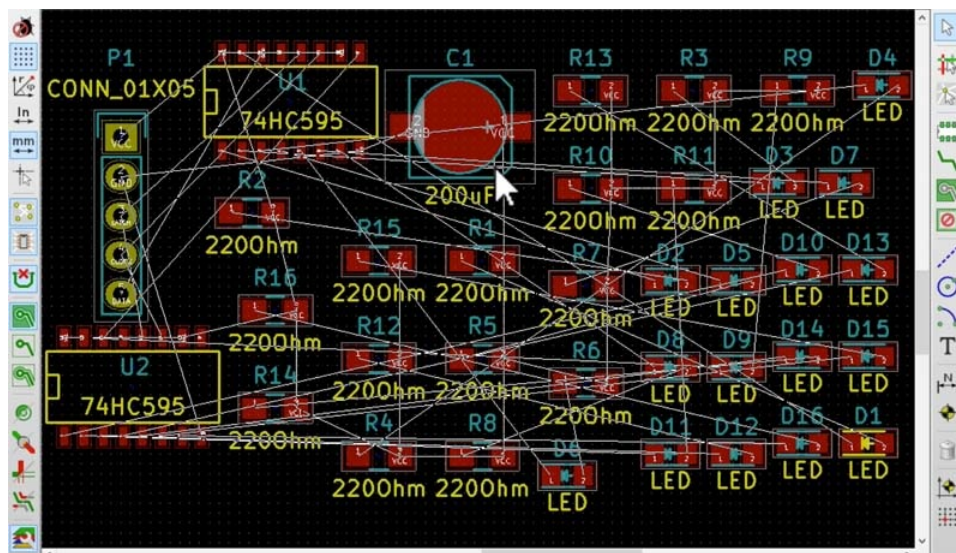
Select Spread out All Footprints.

You will get a warning informing you that any unlocked footprints will be moved. That is fine, so accept the warning. I will show you how to lock footprints in place in a separate chapter.



None of our footprints are locked, so all of them will be spread out.

Kicad will spread out all footprints and produce a layout similar to this:

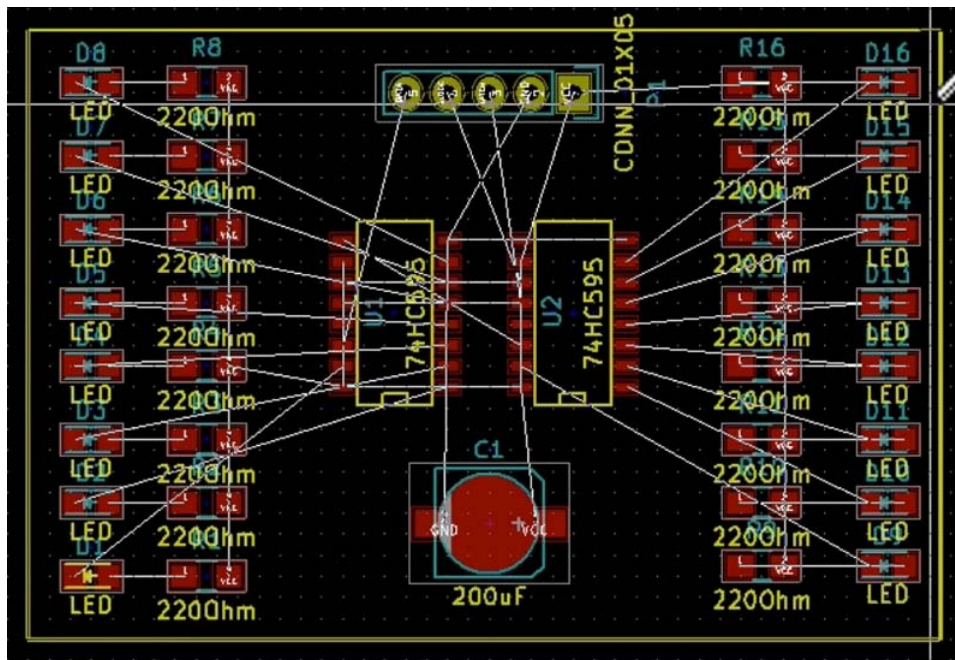


The footprints from the bundle have been spread out so none is overlapping.

All the footprints are now spread out somewhere in the canvass, in a grid-like manner. It is now much easier to find individual footprints and place them to a position that we think is appropriate.

We can now go ahead working on the layout. I would like to position the two integrated circuits in the middle of the board, and the LEDs with the resistors along side the shift registers. I would like to place one bank of the LEDs on the left side of the PCB, and the other bank on the other side. I will also try to place the footprints in a way that minimises the total area for the PCB in order to reduce its cost.

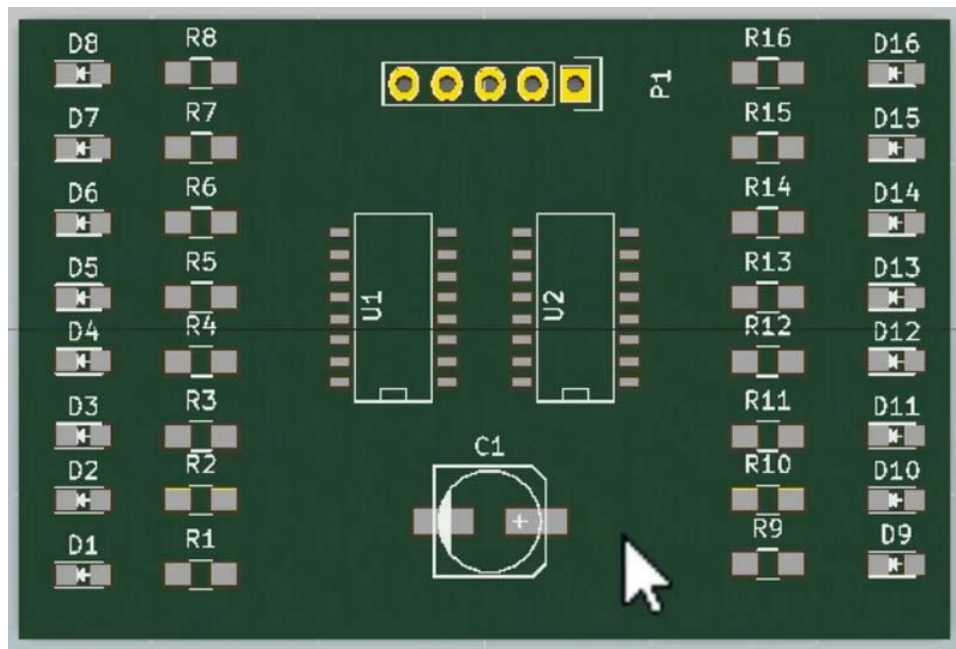
With a bit of trial and error, and after several iterations, I ended up with this layout:



The final example layout, with the edge cut set.

The screenshot above also includes the edge cut border. Don't forget to select the Edge.Cuts layer and the polygon tool in order to draw it.

The 3D preview of this PCB looks like this:

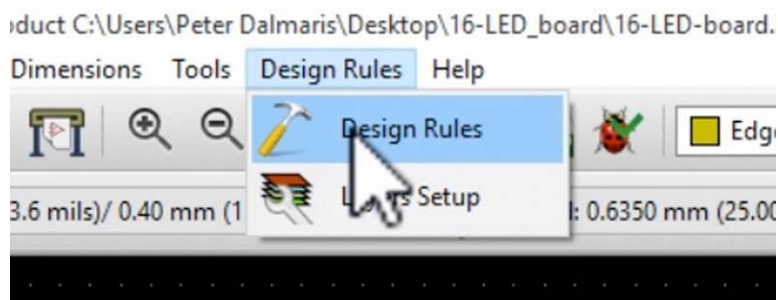


The 3D preview of the current PCB.

Now that the placement of the footprints is complete, we can proceed with the wiring. We'll do that in the next chapter.

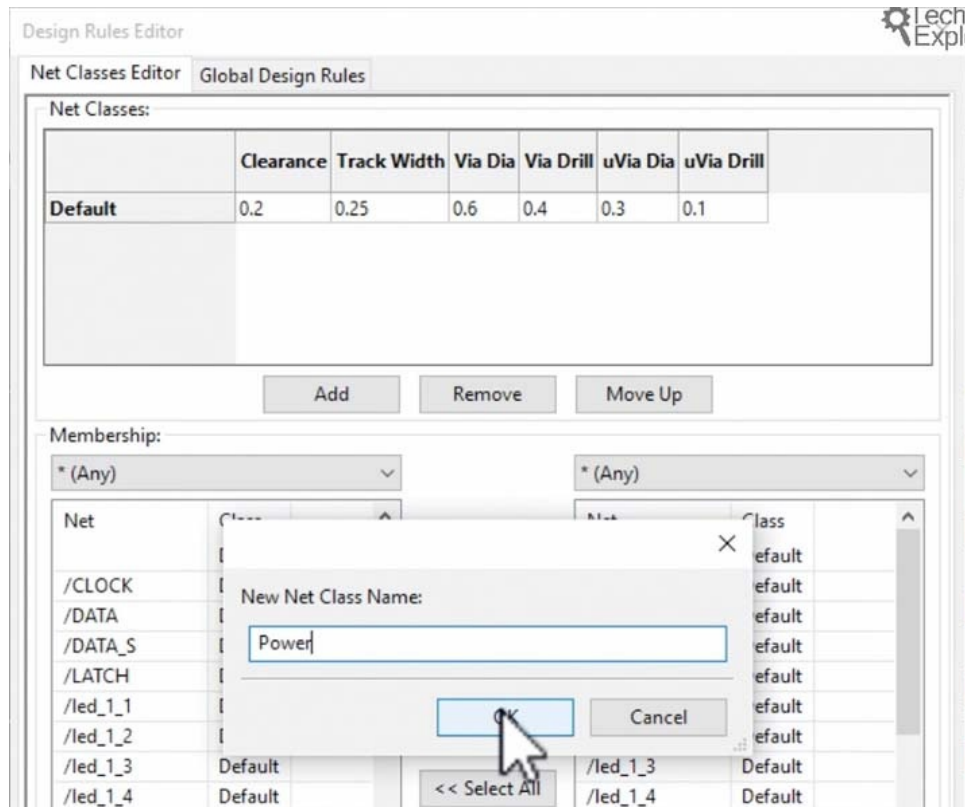
Chapter 56: *Wiring in Pcbnew*

Now that our footprints are placed on the PCB, let's go ahead with the wiring! First, let's edit our design rules so that KiCad can automatically select the appropriate width for each track. Bring up the Design Rules window:



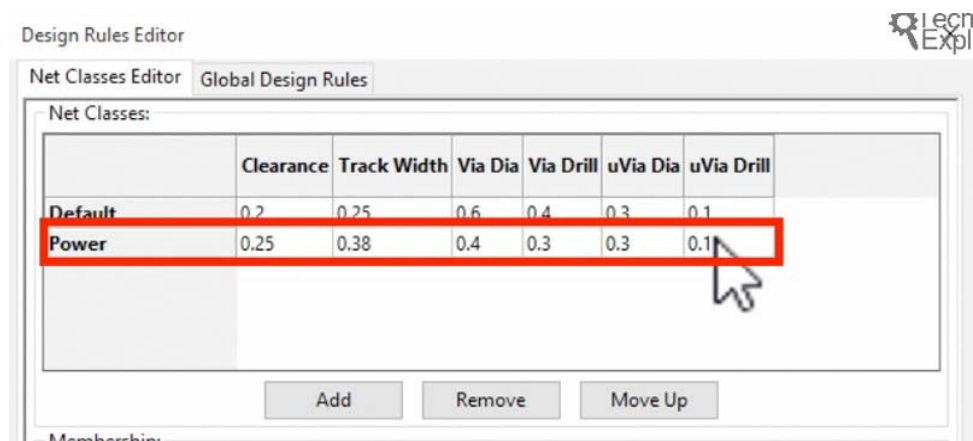
Bring up the Design Rules window

Click on the Add button to create a new Net Class:



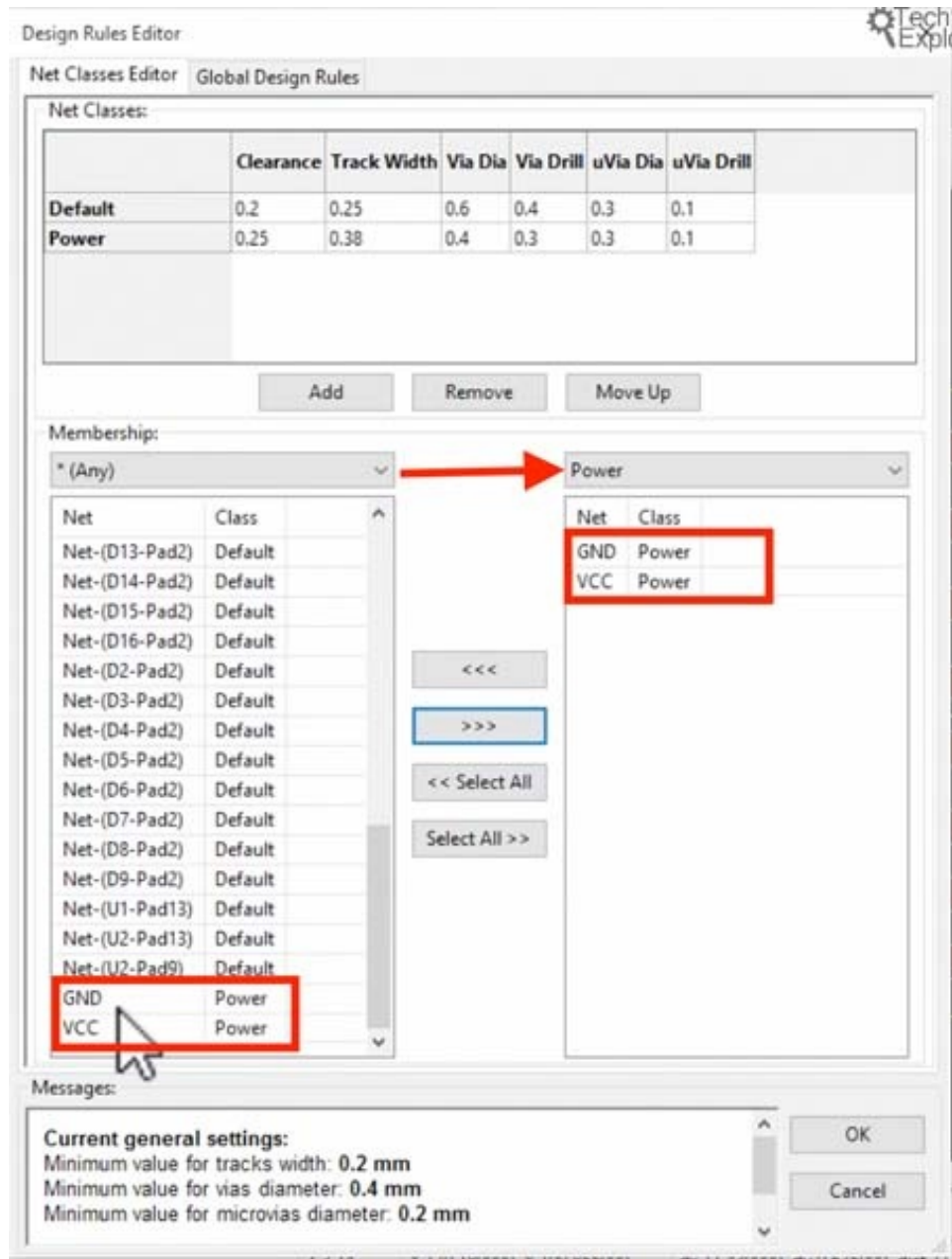
Create a new Net Class.

Configure the new Power class with the values from this example:



The values for the new Power net class.

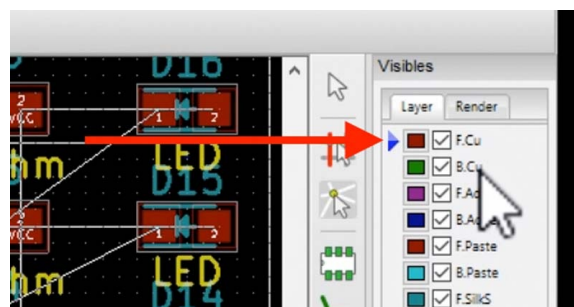
Finally, move the VCC and GND nets into the power net class:



Move the GND and VCC nets into the Power net class.

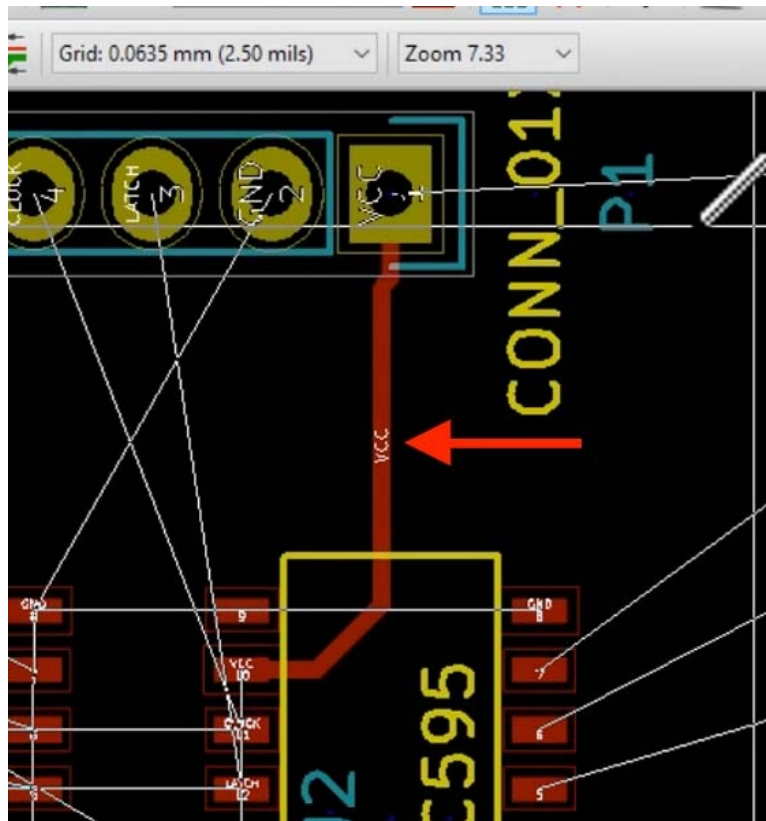
In the screenshot above, notice that the filter of the right membership list is set to Power. After you set the filter, find the GND net in the right membership list, select it, and click on the “>>>” button to move it to the right list. Do the same for the VCC net.

Let’s start laying out some tracks now. We can start with the VCC net. I would like the VCC signals to go on the top copper layer and ground as much as possible on the ground copper layer. Just like in the previous project, we will have to switch signals from top layer to bottom layer through vias. Select the F.Cu layer:



Select the F.Cu layer.

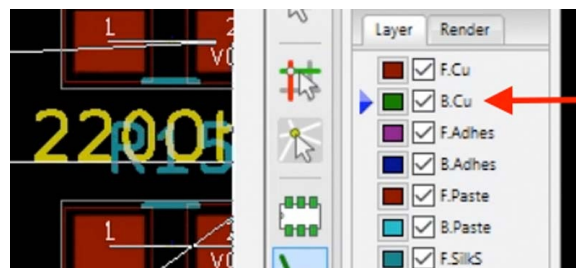
Then, click on the green Wire button or type “W” to select the wiring tool, and create a wire between the Vcc pin of the connector and the Vcc pin of U2.



Just wired the first VCC net.

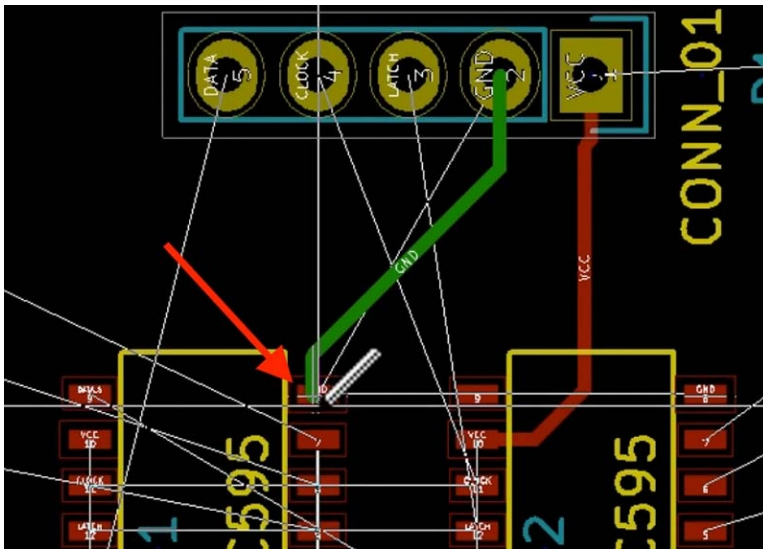
Notice that the name of the wire is written on it: “VCC”, the name of the net.

Let's work on one of the ground wires next. Switch to the B.Cu layer:



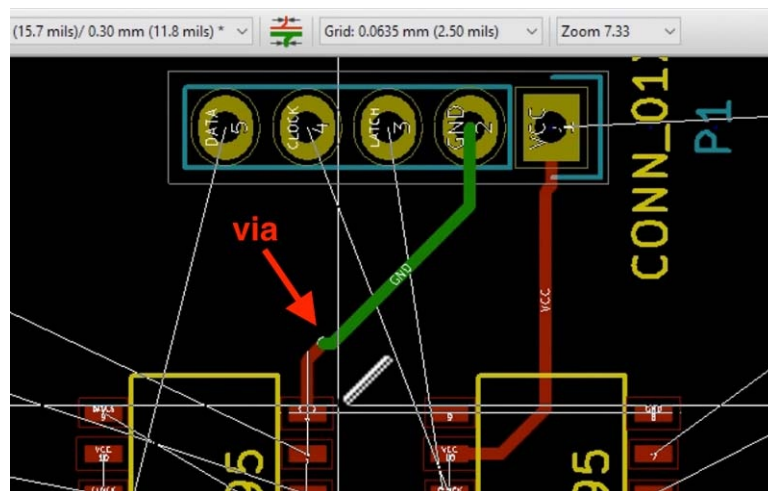
Select the B.Cu layer for the ground signals.

Now try to create a wire between the GND pin of the connector (the second pin from the right) and the GND pin of U1:



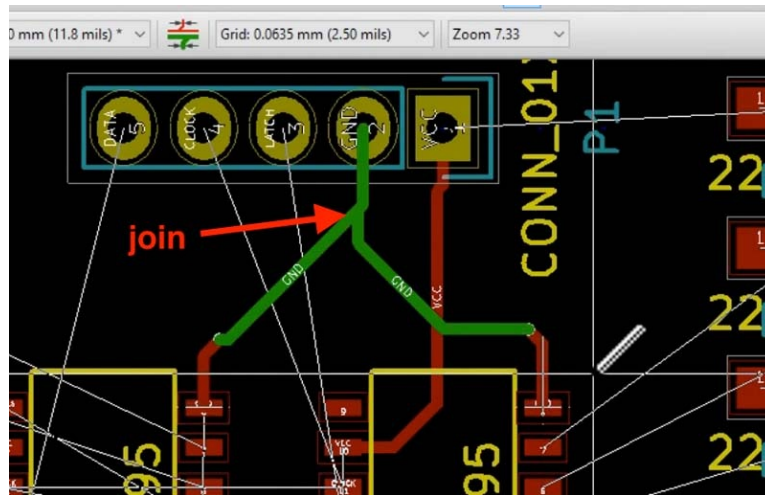
You will be unable to complete the green wire!

No matter how hard you try, you will not be able to complete the green GND wire. This is because footprint U1 is surface mounted, and its pads are only accessible on the top layer. Therefore, we must use a via to switch the wire from the bottom layer to the top layer, and then complete it on the GND pad of footprint U1. Use the “V” key to create a via close to U1’s GND pin, and then double click to complete the wire on the pad:



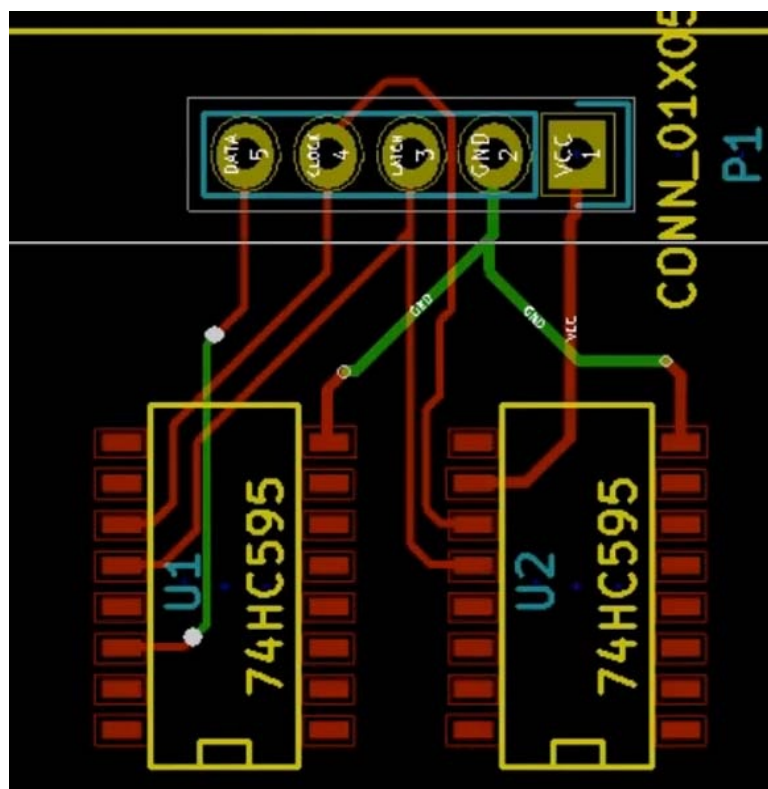
Use a via to switch a wire between layers.

Connect the GND pad of U2 to the same GND wire, again using a via to switch between layers:



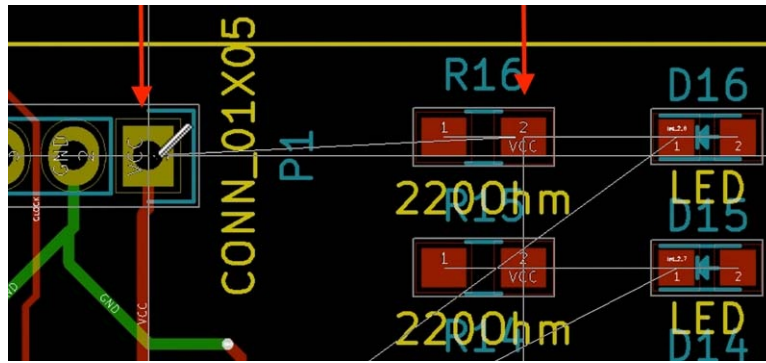
Connect the GND pad of U2 by joining a new wire to the wire of the U1 GND.

We can continue by wiring the rest of the connector pins. Switch to the top layer, and follow the ratsnests to help you wire the connector pins to the U1 and U2 pads. You can use vias if needed to keep the tracks from being too long or having too many twists and turns. My wiring ended up like this:



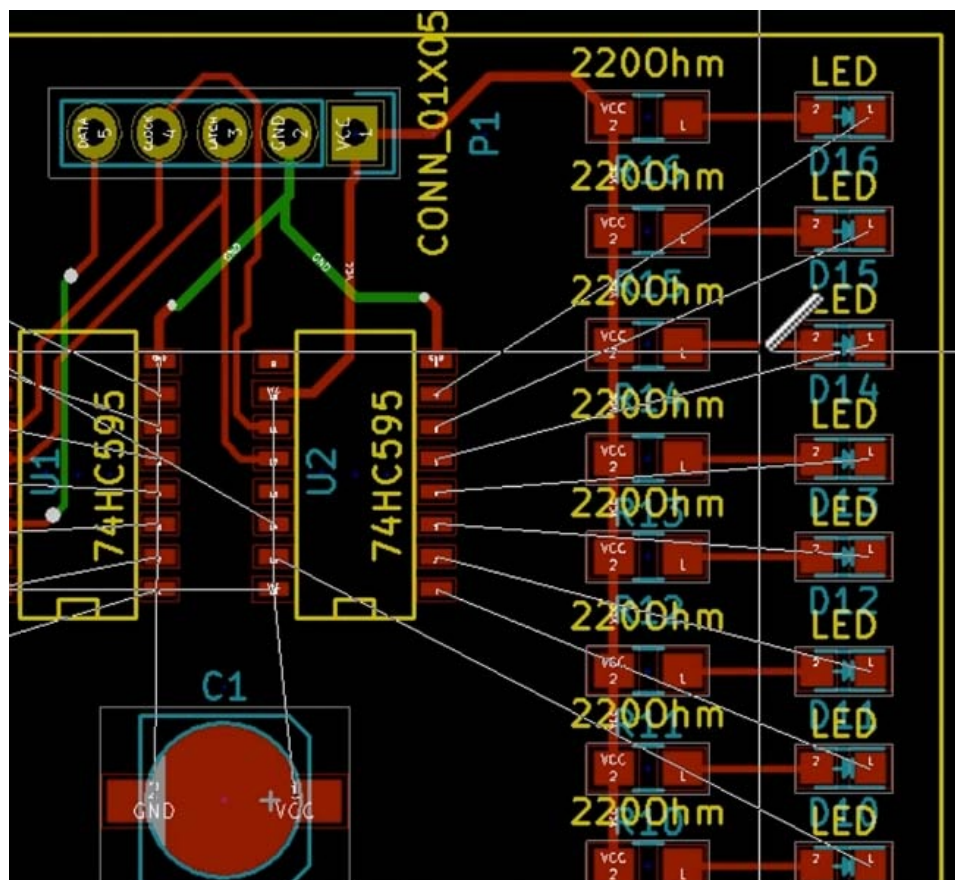
Wiring the connector pins to U1 and U2.

Now let's have a closer look to the wiring between the VCC pin in the connector and the resistors on the right side of the PCB. At the moment, the layout is this:



The orientation of the resistors can be improved in order to minimise the length of the VCC wire.

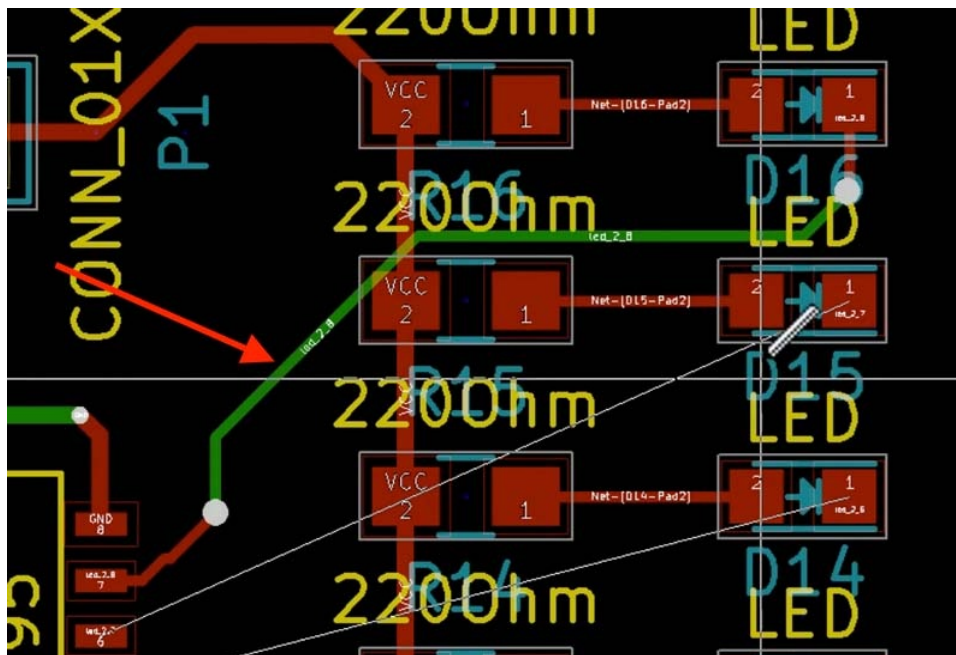
Notice how the VCC pad of the resistor is oriented away from the VCC pad of the connector. At this orientation, the wire that connects the two would have to be routed around the back of the resistor. If we simply flip the resistor over so that its VCC pad is right opposite the VCC pad of the connector, the length of the wire would be minimal, without any angles. It would also make it easy to connect the VCC pads of all other resistor in the bank to the same wire. So, that's what you should do next: place your mouse pointed over an resistor, hit the "R" key to flip it. Repeat for all resistors. Then, connect the VCC pad of the connector to the VCC pad of the first resistor, and extend the wire downwards to connect the VCC pads of all resistors. Then, also connect the LEDs to their current limiting resistors. The PCB will look like this:



The Connector and Resistor VCC pads are connected. The LEDs are also connected to their resistors.

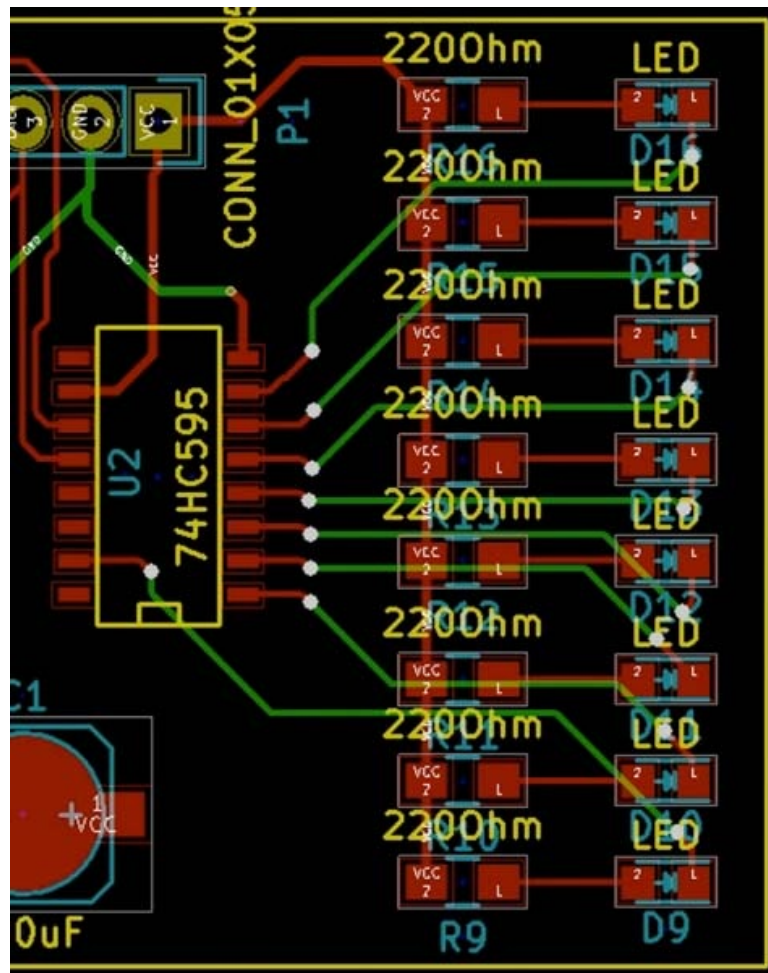
It's a fairly clean and straight forward route for the VCC wire. The unconnected pad of

the LEDs must be connected to the data pins on U2. To do the wiring we will need to use vias so that part of the wire is routed in the back copper layer so that it does not cross the long VCC wire that connects the resistors. Here is how you can route this wire:



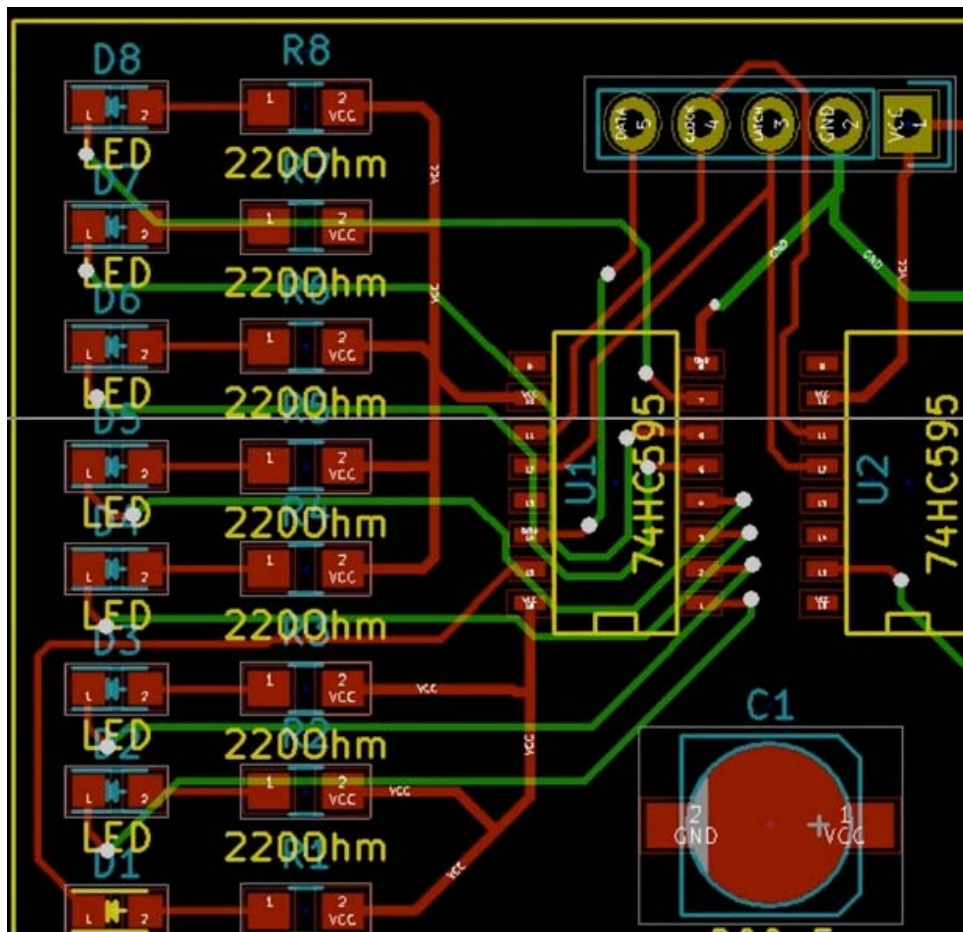
Routing the LED to the data pins of U2.

In this example, starting from the first LED pad on the front layer, use a via to switch to the back layer, continue the wire until it gets close to pin #7 of U2, then use another via to continue on the front copper layer, and end the wire on pin #7. Repeat the same process for the rest of the resistors, until you complete the wirings on the right side of the board. The PCB will look like this:



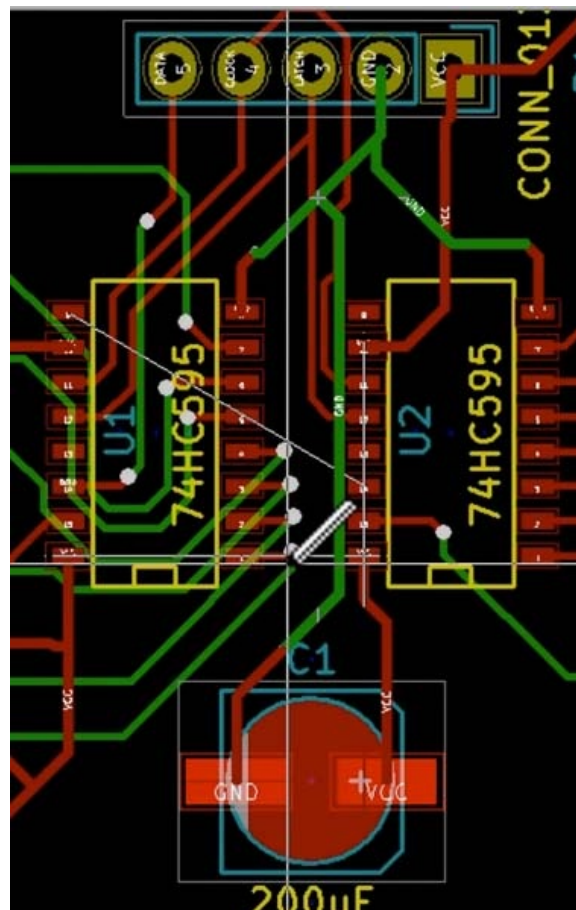
The wiring on the right side of the PCB is complete.

Repeat the process to connect the LEDs and resistors on the left side of the board. Don't forget to do a Design Rules Check occasionally. This will tell you if you have left unconnected pads. When you complete the left side of the board, the PCB will look like this:



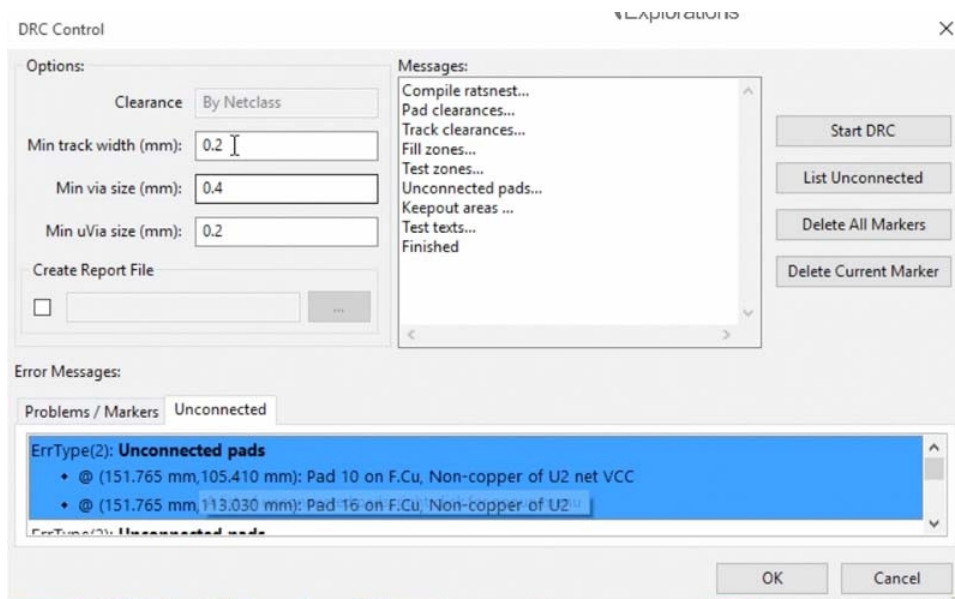
The left side of the PCB is complete.

The last footprint to connect is the capacitor. We can connect pin 1 to any existing VCC net, and pin 2 to any existing GND net. Here is the way I wired mine:



The capacitor is wired.

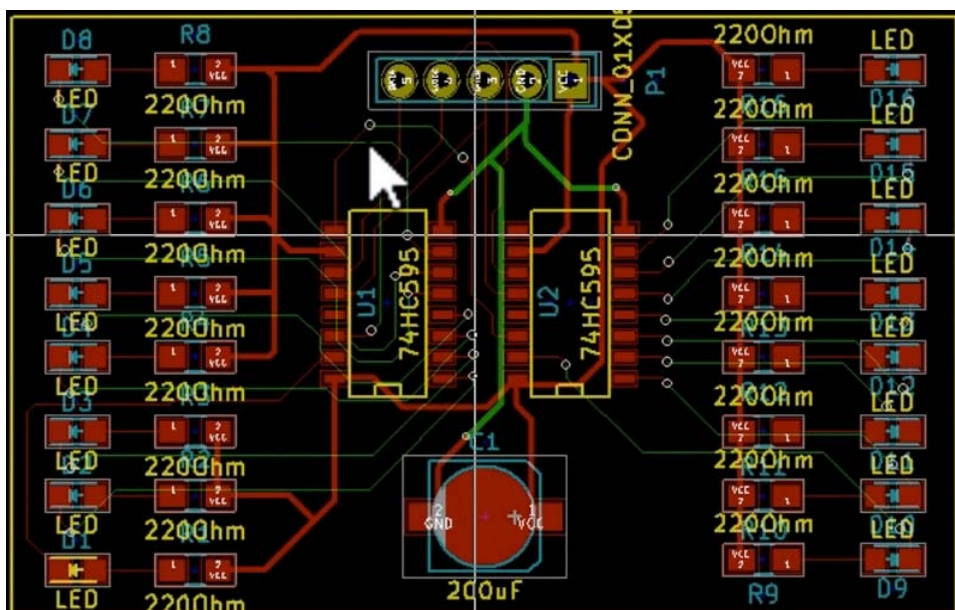
At this point I can see a couple of ratsnest threads, which means that I have unconnected pins. The canvas is crowded, so I find it hard to see exactly which pads are left unconnected. A good solution is to do an DRC, and get a list of unconnected pads.



The DRC gives a handy list of unconnected pads.

The DRC tells me that pad #9 of U1 should be connected to pad #14 of U2, pad #10 of U2 should be connected to pad #16 of U2, pad #16 of U1 should be connected to pad #16 of U2, and pad #2 of R4 should be connected to pad #2 of R3. Let's do these connections.

Here is the final wiring:



The final wiring

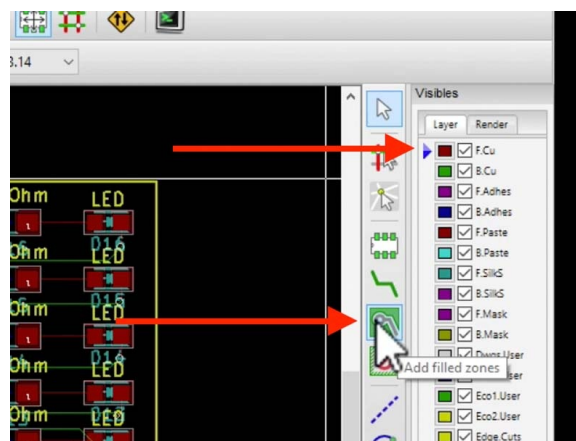
Wiring takes some time to do properly, and several iterations. You should not be afraid of deleting traces and redoing them better!

In the next chapter, will will work on the copper fills.

Chapter 57: *Adding copper fills*

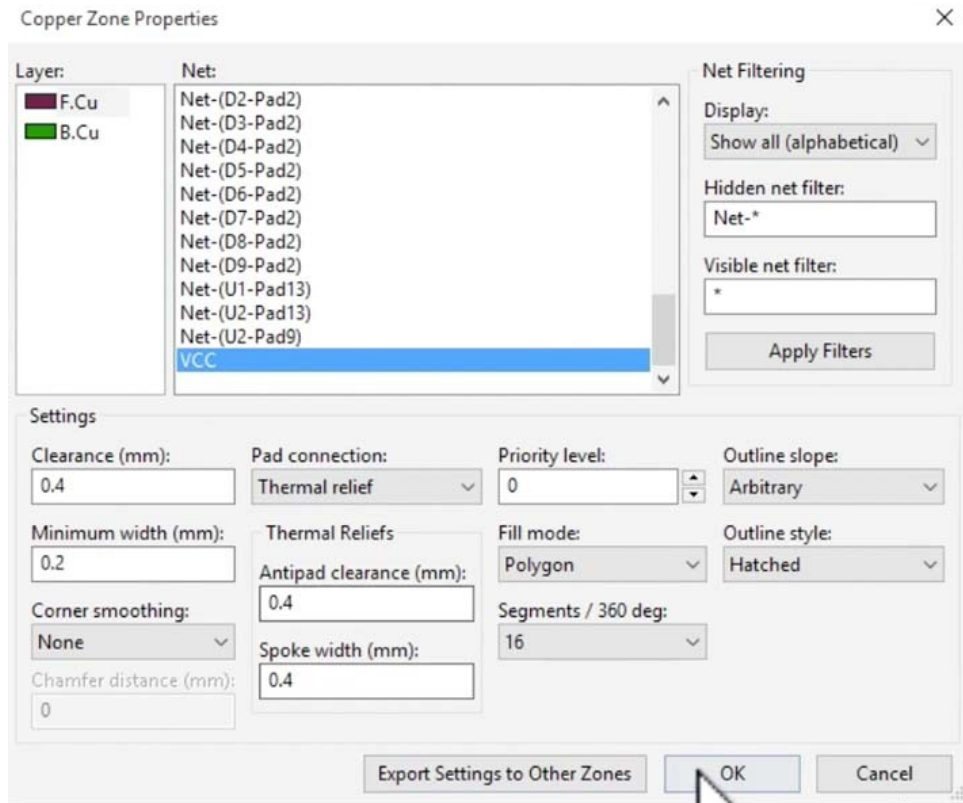
In this chapter we will add the copper fills for the ground (in the back layer) and Vcc (in the front layer).

Start with selecting the F.Cu layer and the Filled Zones tool.



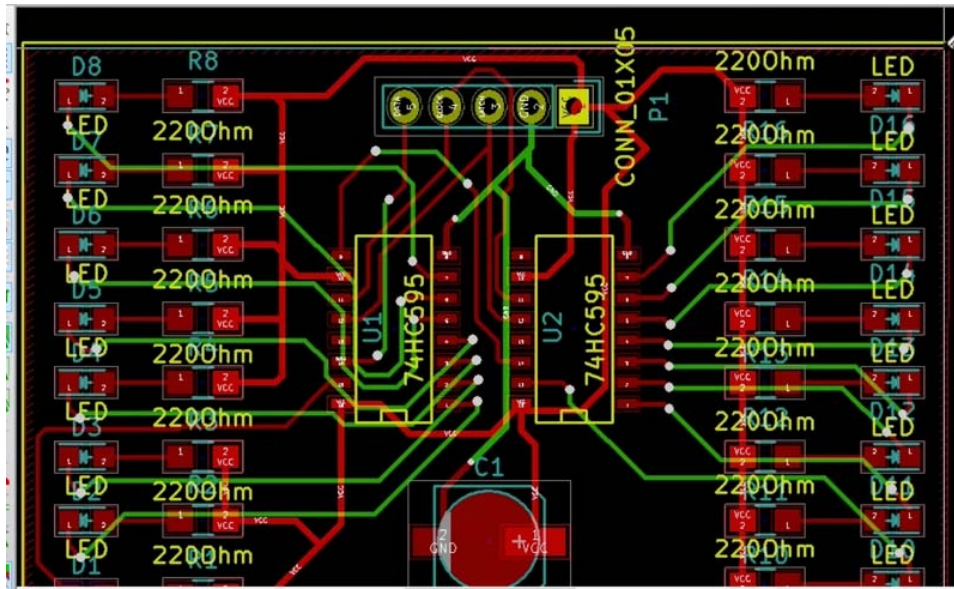
Select F.Cu and the Filled Zones tool

Configure the front copper fill so that it is connected to VCC:



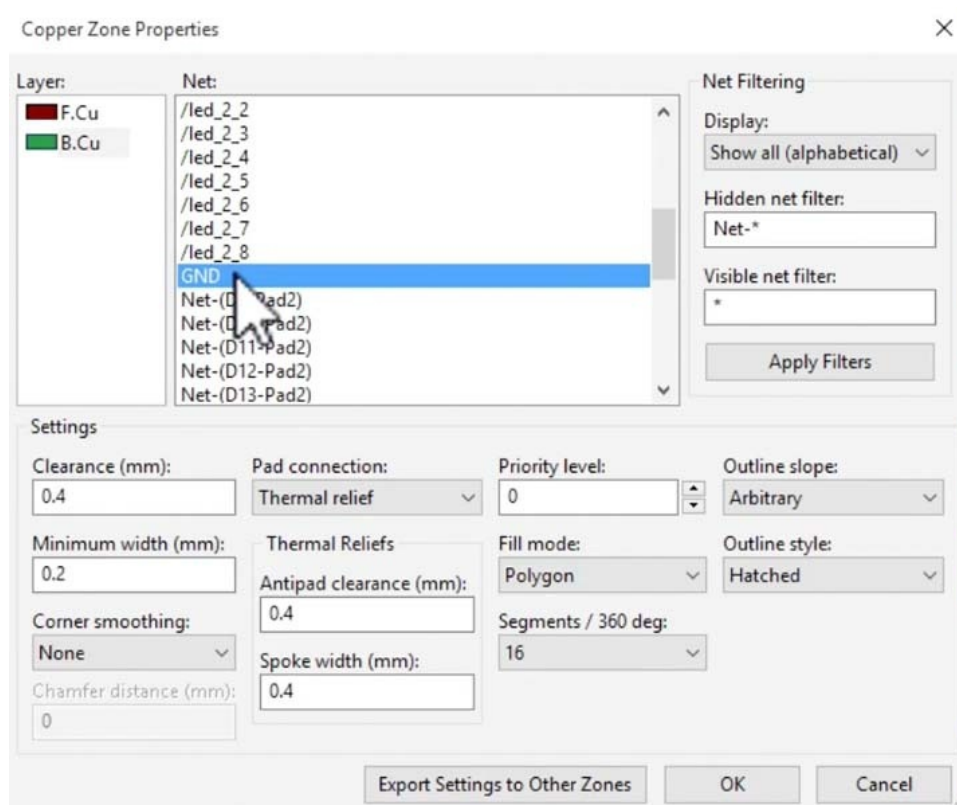
The front copper fill will be connected to the VCC net.

Draw the fill rectangle close to the edge of the board:



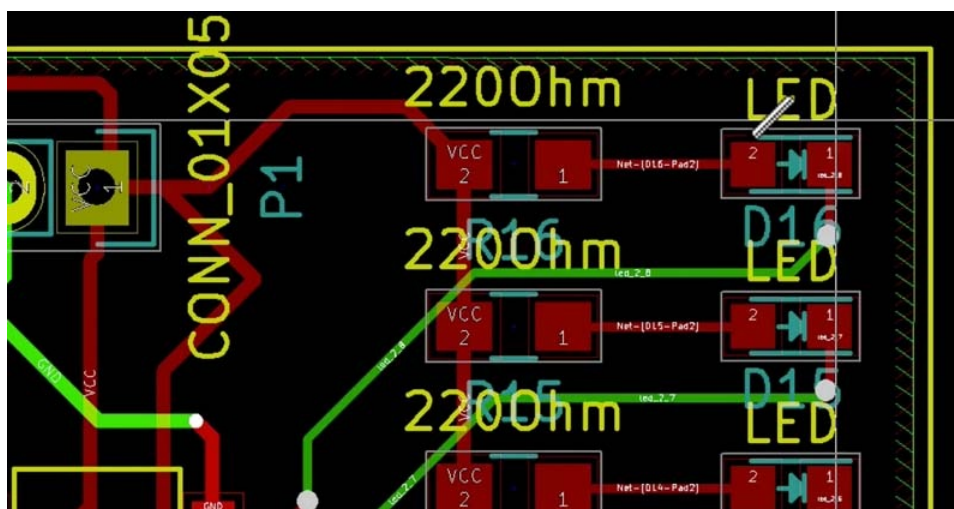
The copper fill for the front layer, connected to VCC.

Click again at the top right corner of the board (inside the yellow edge cuts line!) to start setting up the back copper fill. Choose the B.Cu layer, and connect it to the GND net:



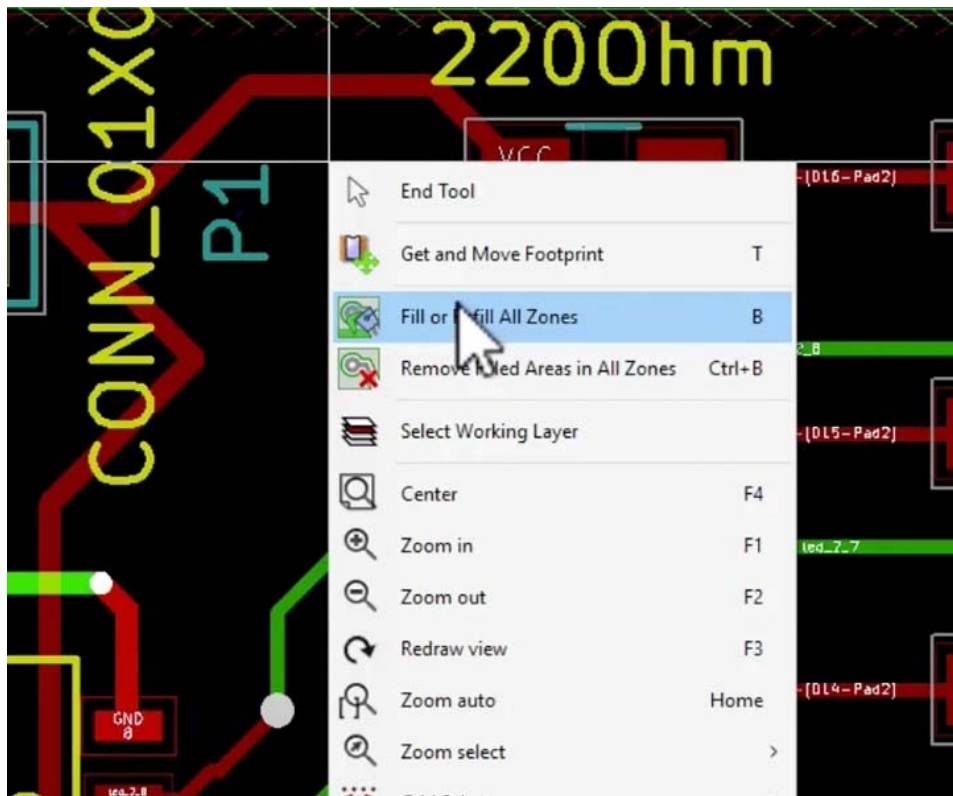
Configure the back copper fill.

Set the boundary for the back copper fill:



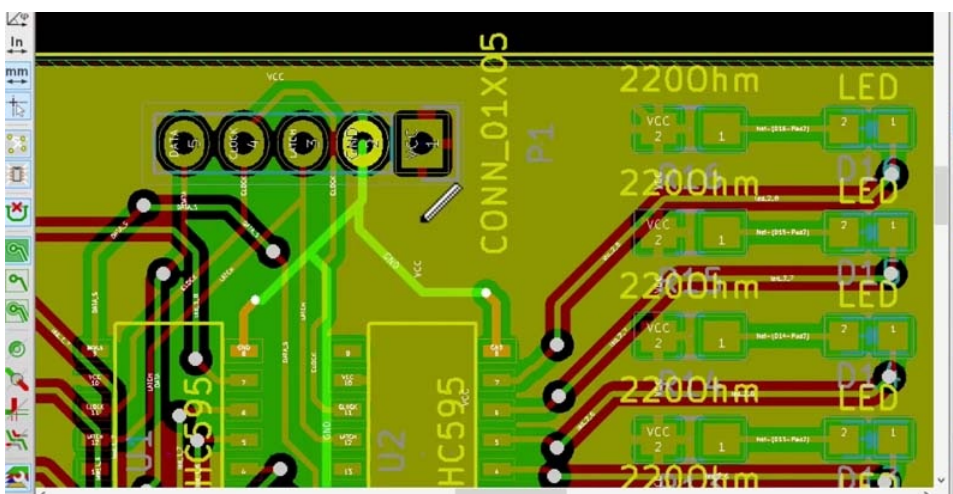
The back and front copper fills defined (detail).

Both copper fills are now defined. Let's fill them with copper. Right-click anywhere inside the fill zones and select "Fill or Refill All Zones":



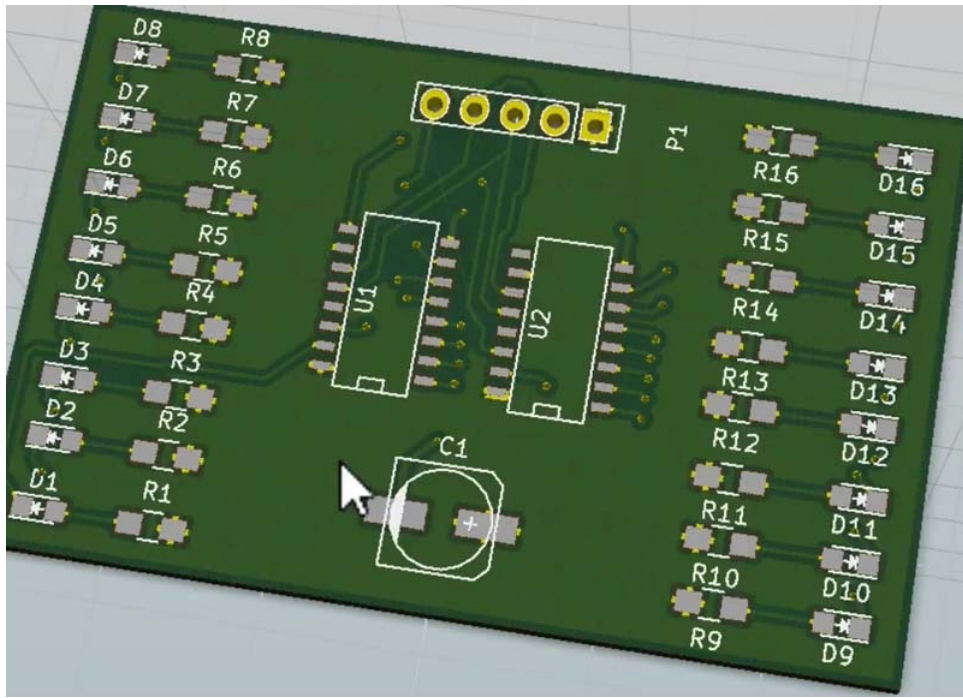
Fill the zones with copper.

The PCB will look like this once the zones are filled:

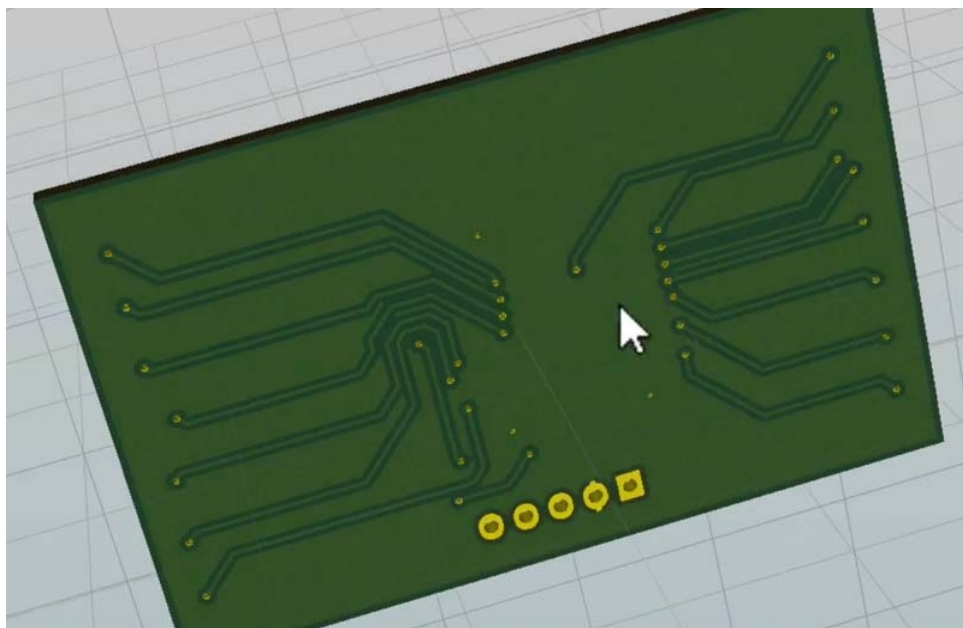


The copper zones are now filled.

Let's also have a look of the board in 3D:



The front of the PCB, in 3D.



The back of the PCB, in 3D.

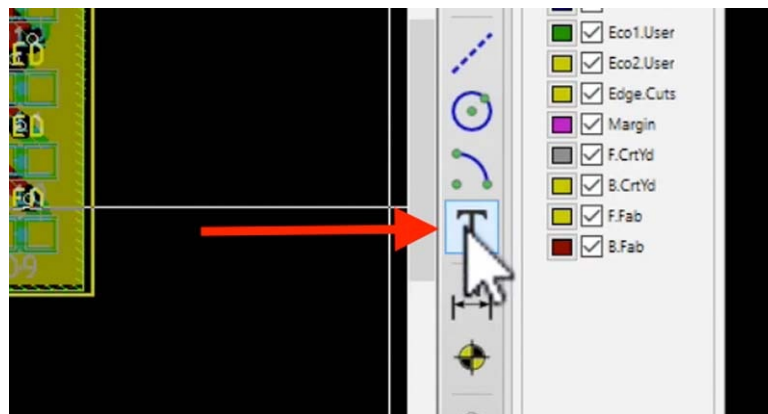
Notice the different shade of green that marks areas that are filled with copper and those that are not.

The electrical work on the board is now complete. In the next chapter we will finish the board by adding text labels and a decorative graphic.

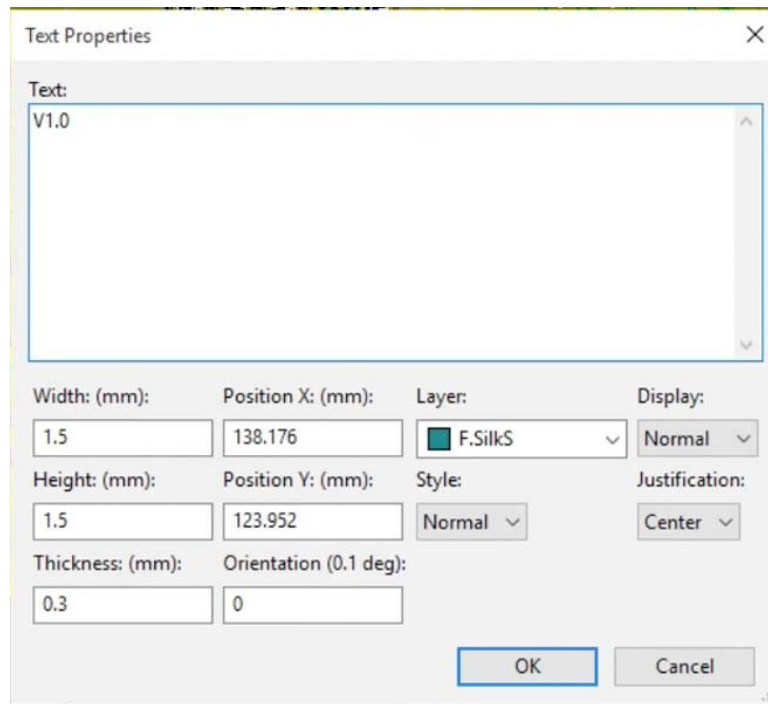
Chapter 58: *Adding text labels and decorative graphics*

We are approaching the end of this project. One of the last few things to do is adding text labels with information about the PCB, and if you choose, a nice decorative graphic.

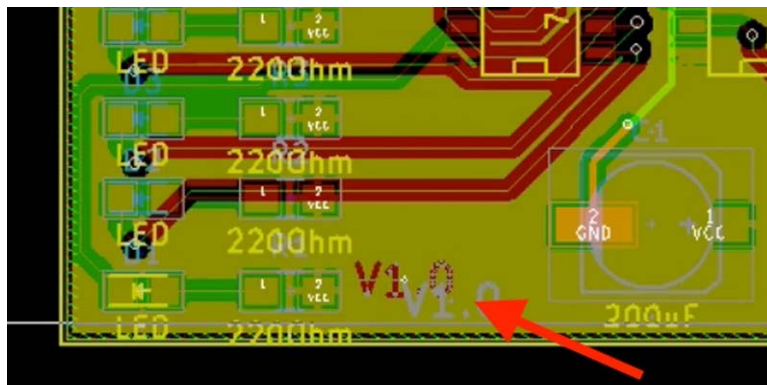
Let's add some text labels. First, I would like to put one with the version of the project. Choose the front silk layer and place the label inside a clear area towards the bottom of the PCB:



Select the Text tool.

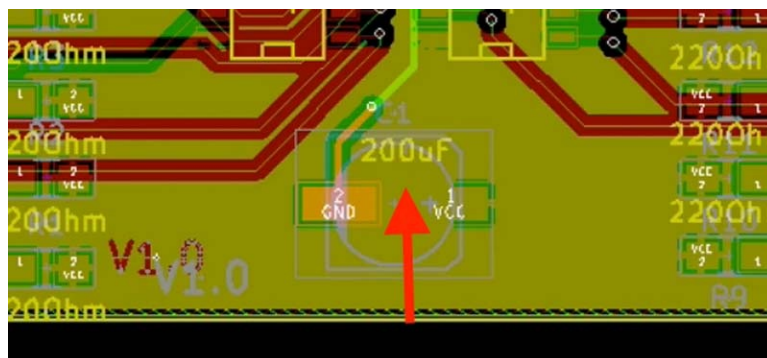


Click inside a clear part of the PCB and type the content of the label in the text box.



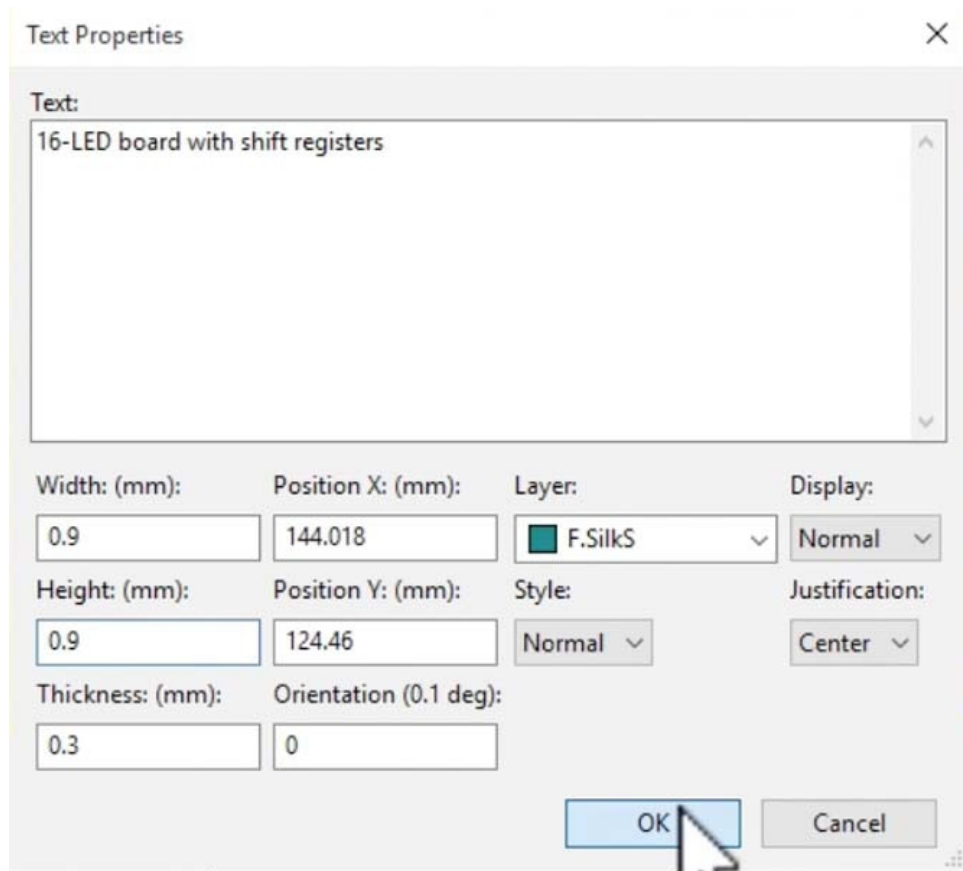
Fine tune the position of the label with your mouse, and click to set it.

I would like to make some more space available just above the bottom edge of the board. At the moment, the “200uF” label is there, so let’s move it up:

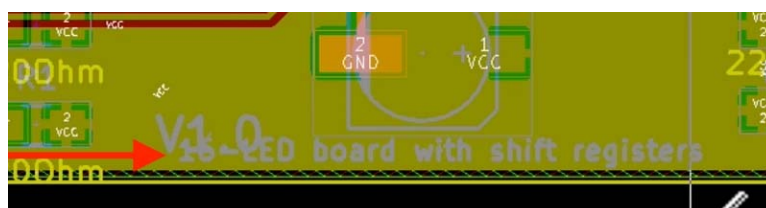


Move this label up to make room.

Let's add a label with the name of the board. The name of the board is "16-LED board with shift registers". Reduce the size of each character to 0.9 mm for both width and height:

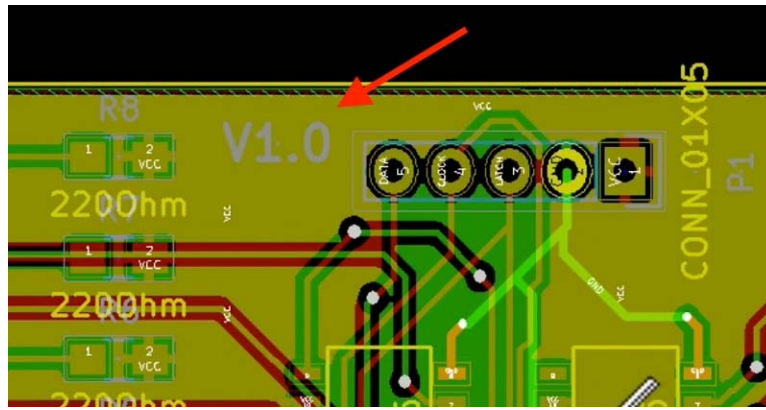


Add a label with the name of the board.



Added a label with the name of the board.

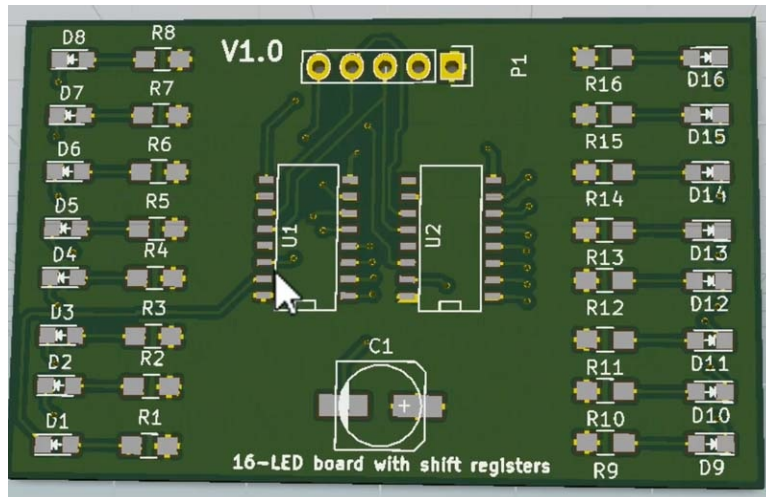
The version and name labels overlap, so let's move the version label at the top of the board where there is enough space:



The version label, repositioned.

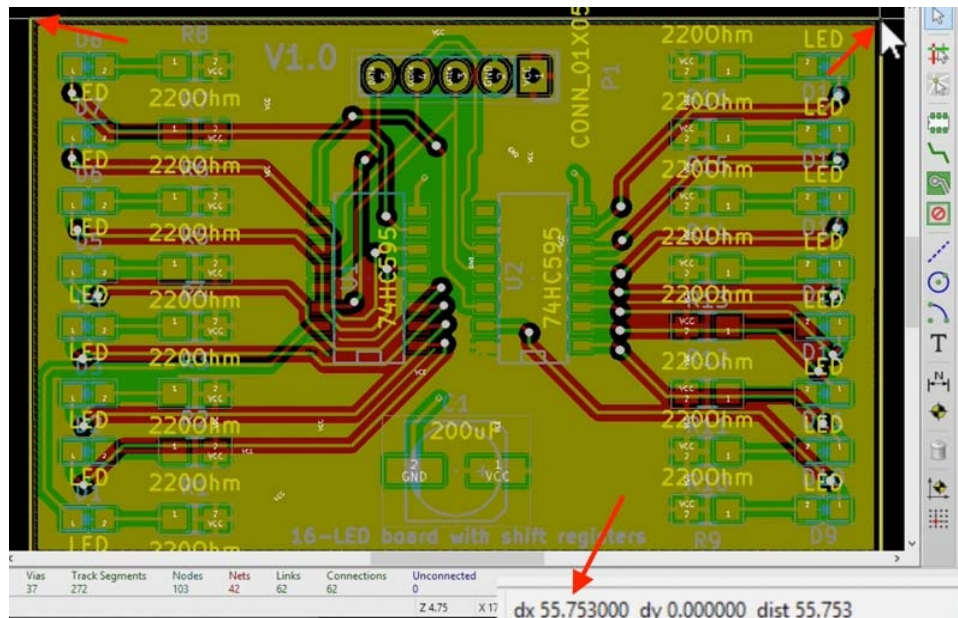
It's all aesthetics really at this point. I think that what we have now is sufficient as far as text labels are concerned. It's really a personal thing so decide what messages or text or numbers you'd like to put in.

Let's have a look at the 3D preview:

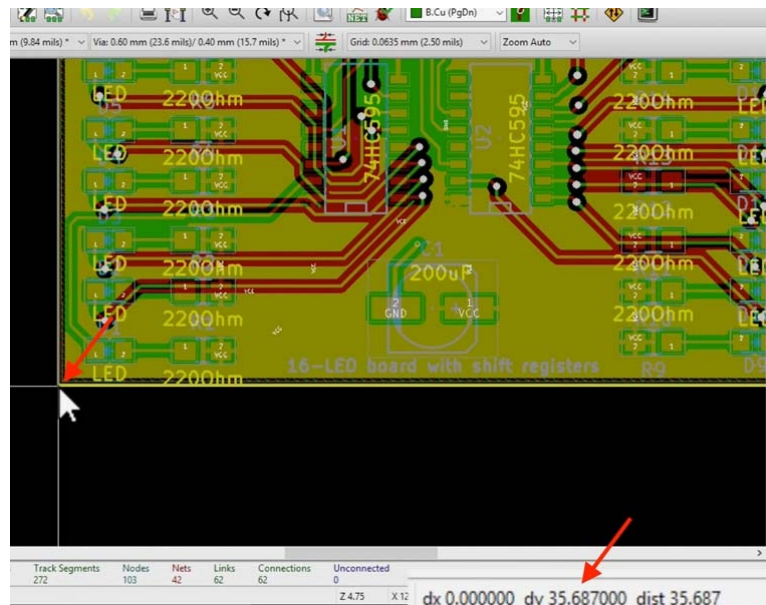


A 3D preview of the board.

Save the project. Let's work on the decorative graphic next. We will need the dimensions of the board to use with the Bitmap to Component Converter, so let's use the Pcbnew measuring tool.



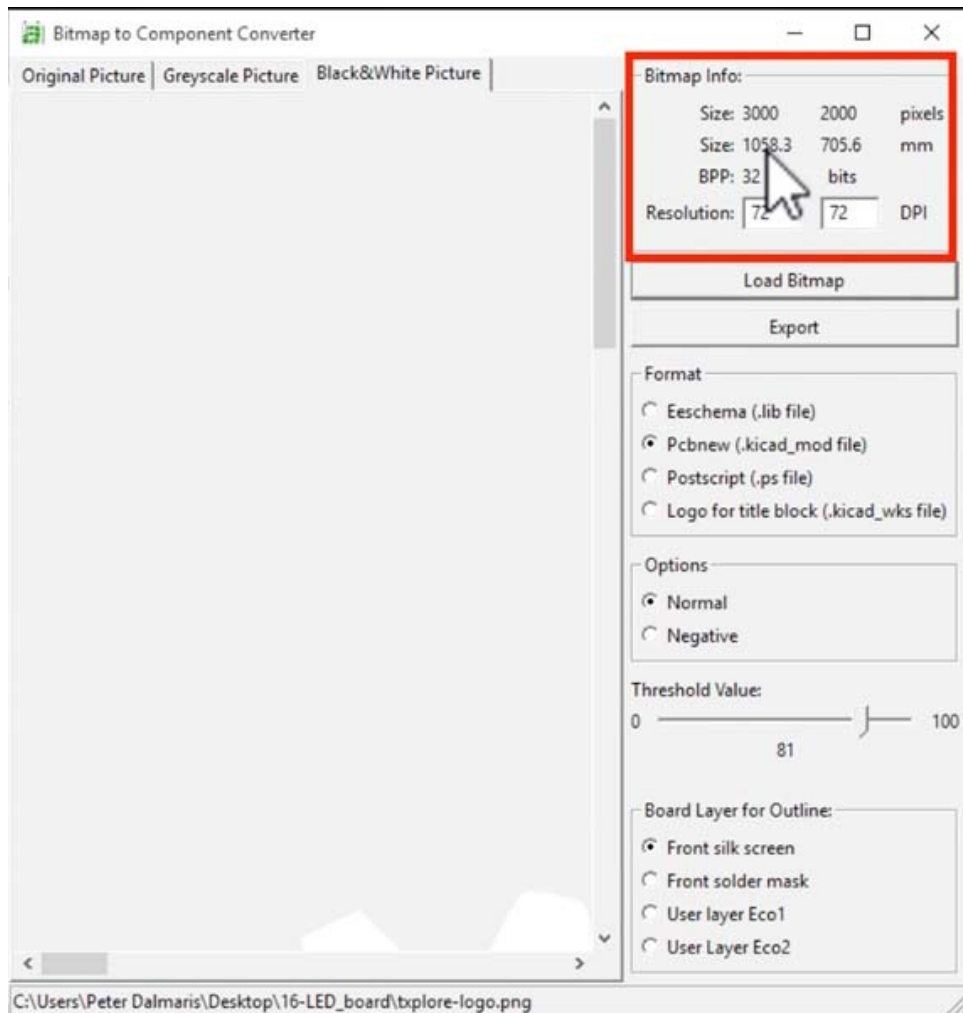
Measuring the horizontal size of the board.



Measuring the vertical size of the board.

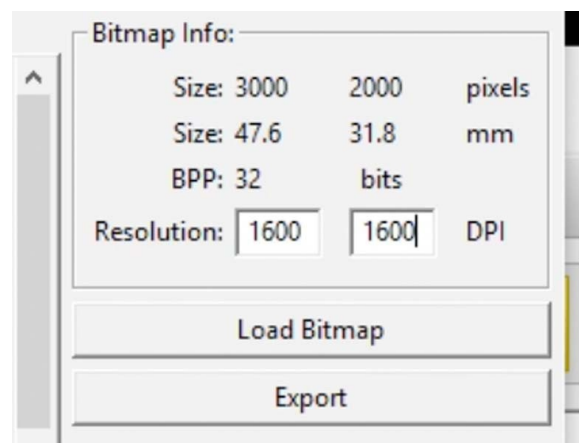
I will use the same logo as in the previous project, but you feel free to choose something else. We need to find out the dimensions of the PCB. My measurement for the horizontal size is 55.75 mm, and for the vertical 35.5 mm. I will write down 55 mm x 35 mm, rounded down to the next decimal. We don't need to be too accurate about this because all we are trying to do is create a footprint that will be within the boundaries of the PCB. Therefore rounding the exact size down is fine.

Close Pcbnew and start the Bitmap2Component app. Load your graphic PNG file.



Bitmap to Component Converted with the graphic loaded.

You can see that the image size as it is around 1000 mm by 700 mm, too big for the PCB. Use the DPI setting to reduce the size to somewhere smaller, around 55 mm width.

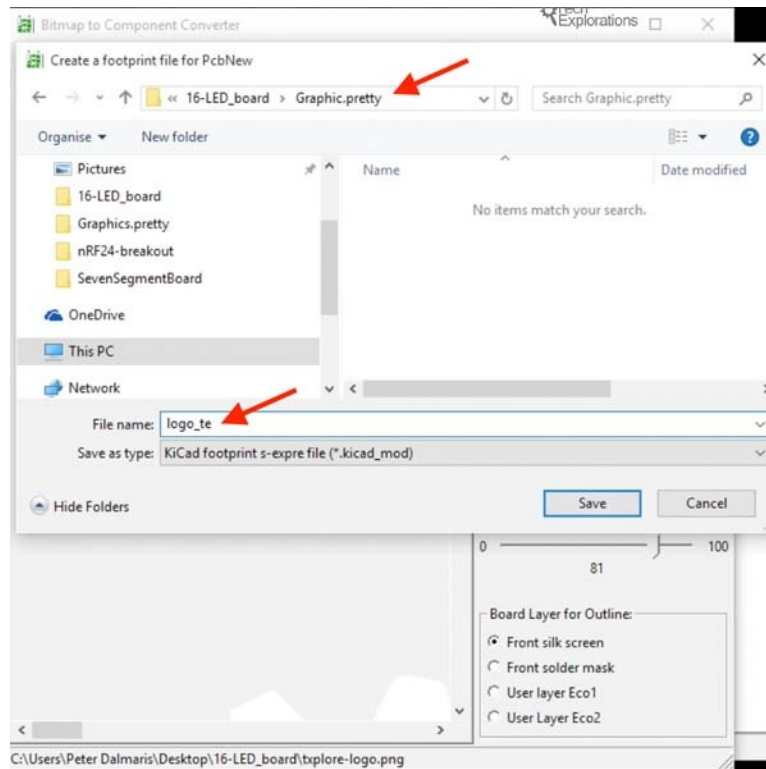


At 1600 DPI, the graphic's size will be able to fit inside the PCB.

At 1600 DIP, the graphic will be resized to 47.6 mm X 31.8 mm, which will allow it to fit easily inside the PCB. So we can accept this value and export the footprint.

Create a new folder in the project directory to hold the new footprint. Call it "Graphic.pretty". "Pretty" is the required extension here. Go inside the library directory

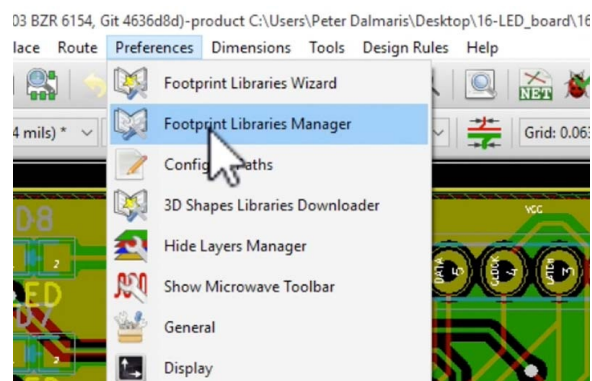
and call this footprint as “logoTB” text boration’s logo.



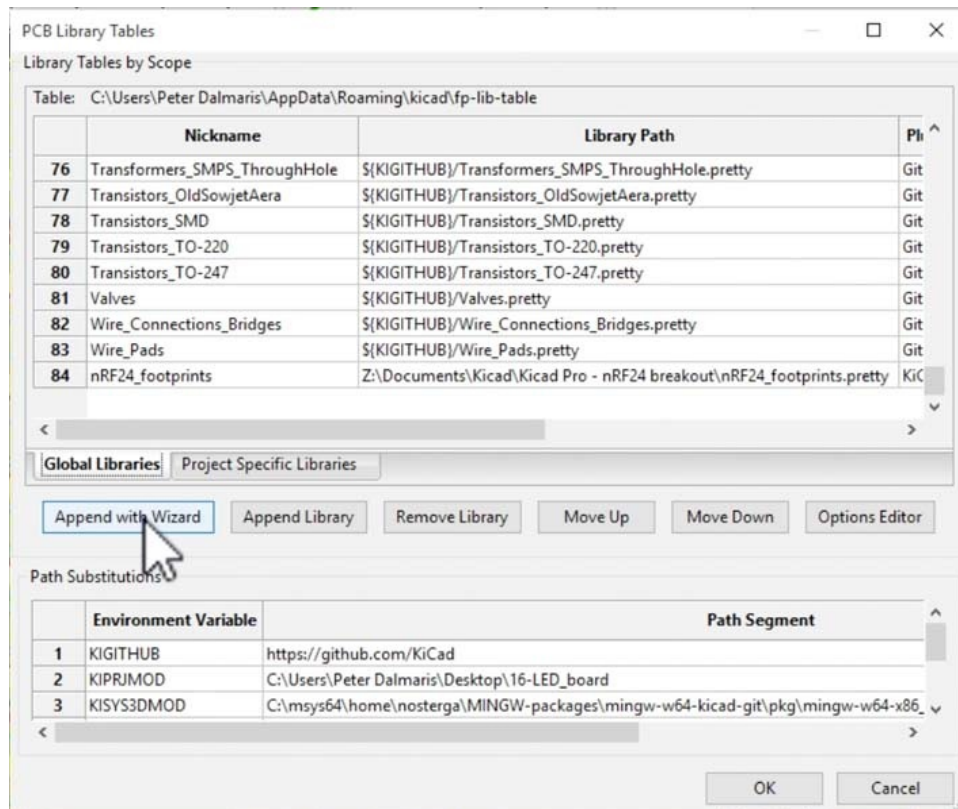
Store the new footprint in a new directory called Graphic.pretty, with file name “logo_te”. The Graphic.pretty directory is stored inside the project directory.

Click on Save to finish with the footprint generation, and restart Pcbnew.

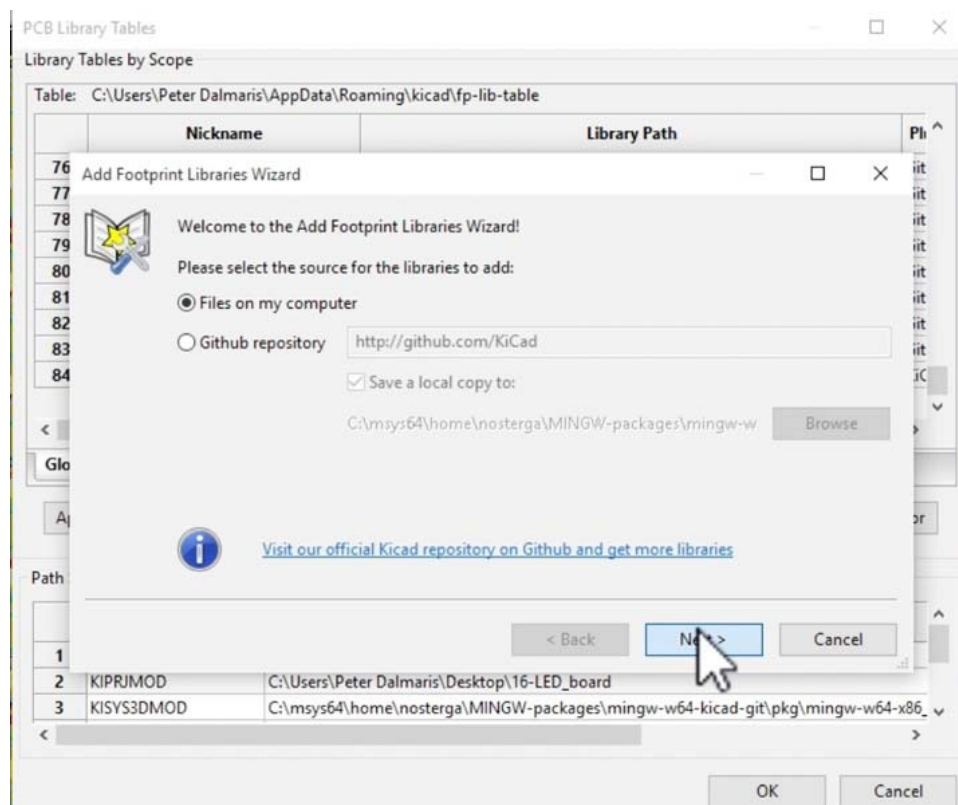
In Pcbnew we will import the new footprint so we can add it to the board. Go to the footprint library manager and add a new library with the wizard.



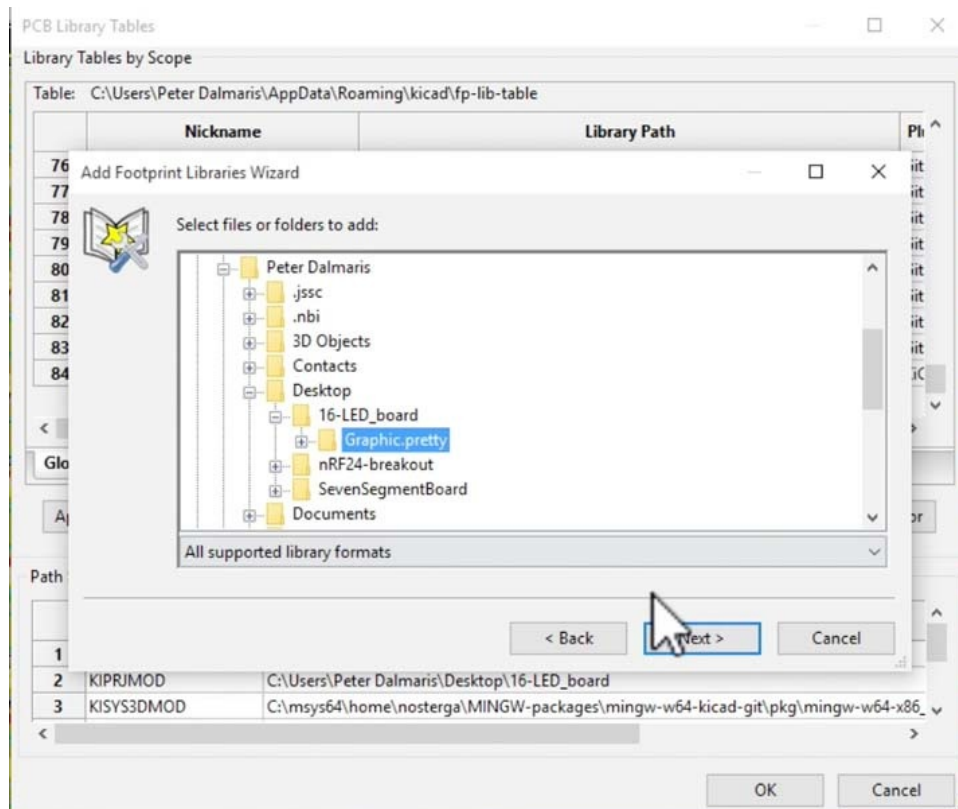
Start the Footprint Libraries Manager



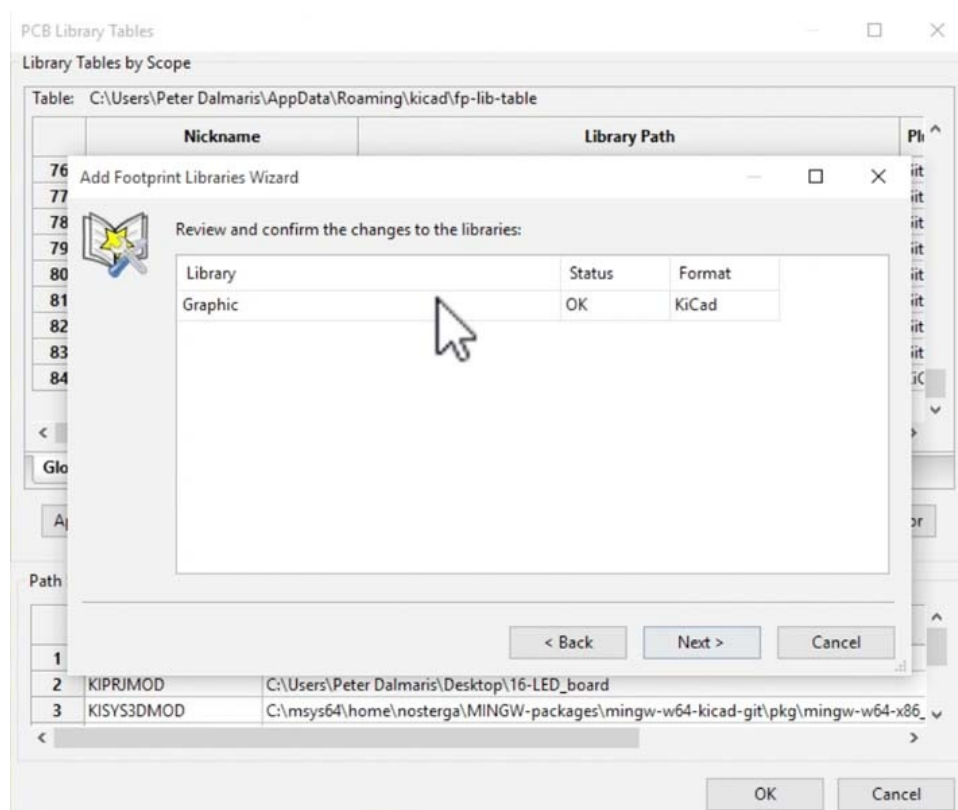
Start the Wizard.



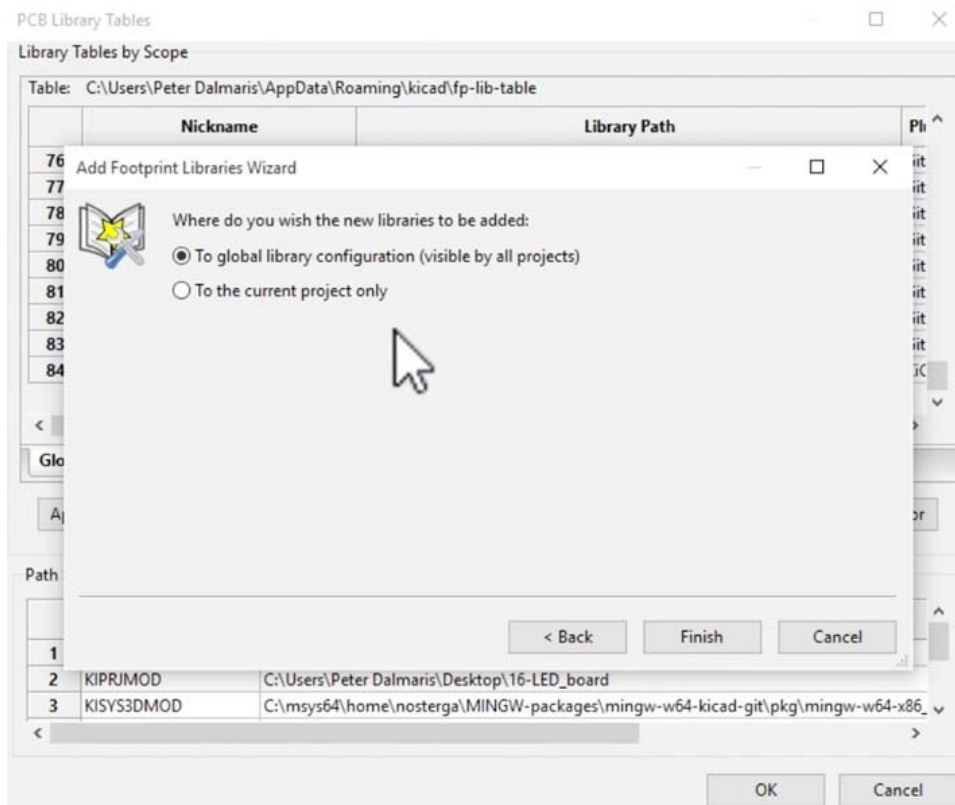
Look for files on your computer.



Look for the library on your filesystem.

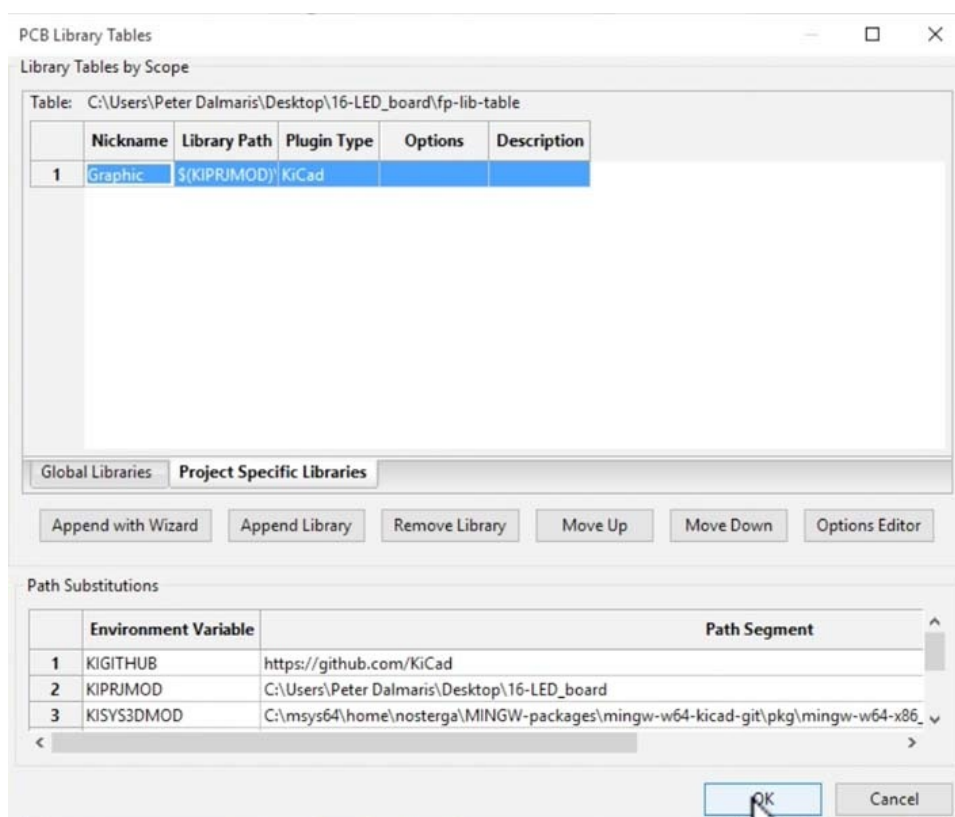


Kicad will detect the footprint library inside the .pretty directory.



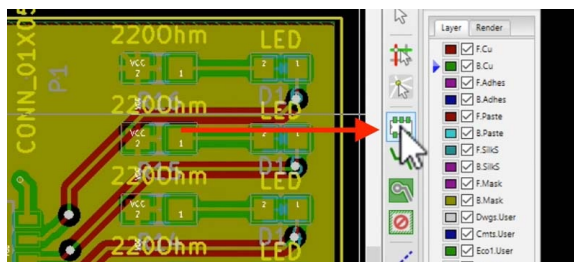
Choose the accessibility level for the new footprint. I selected to use it for the current project only.

Click “Finish” to complete the import process. The new footprint will now be available to use in the current project.

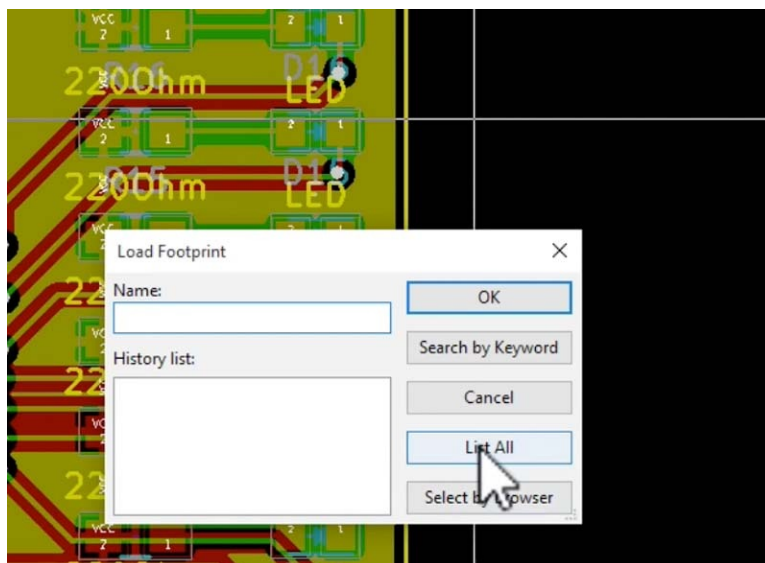


The new library is now imported, and is available for the current project.

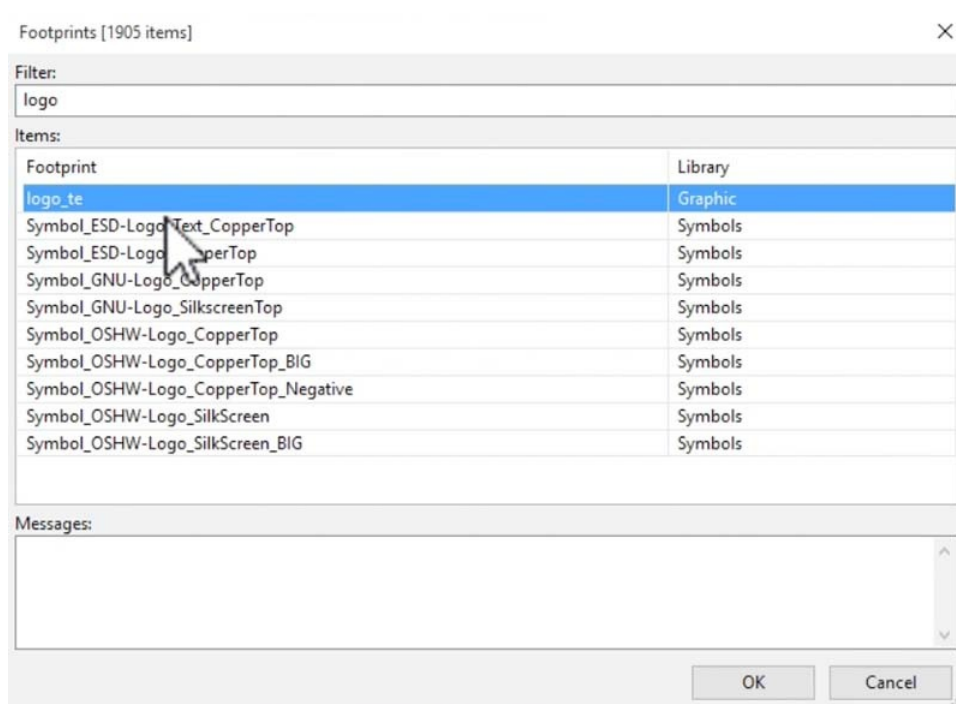
Let's add the new footprint to the back silk screen. Click on the Add Footprint button and click somewhere in the canvas:



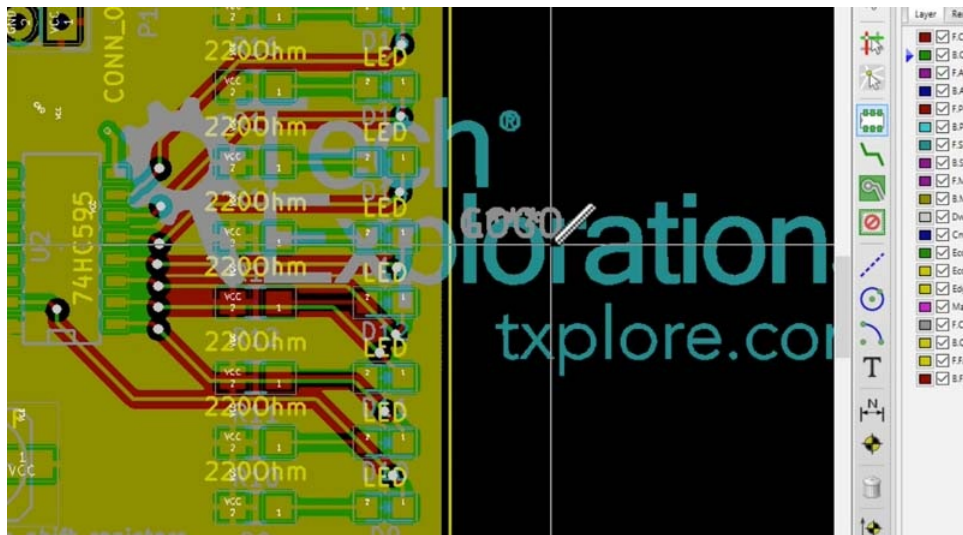
The Add Footprint button.



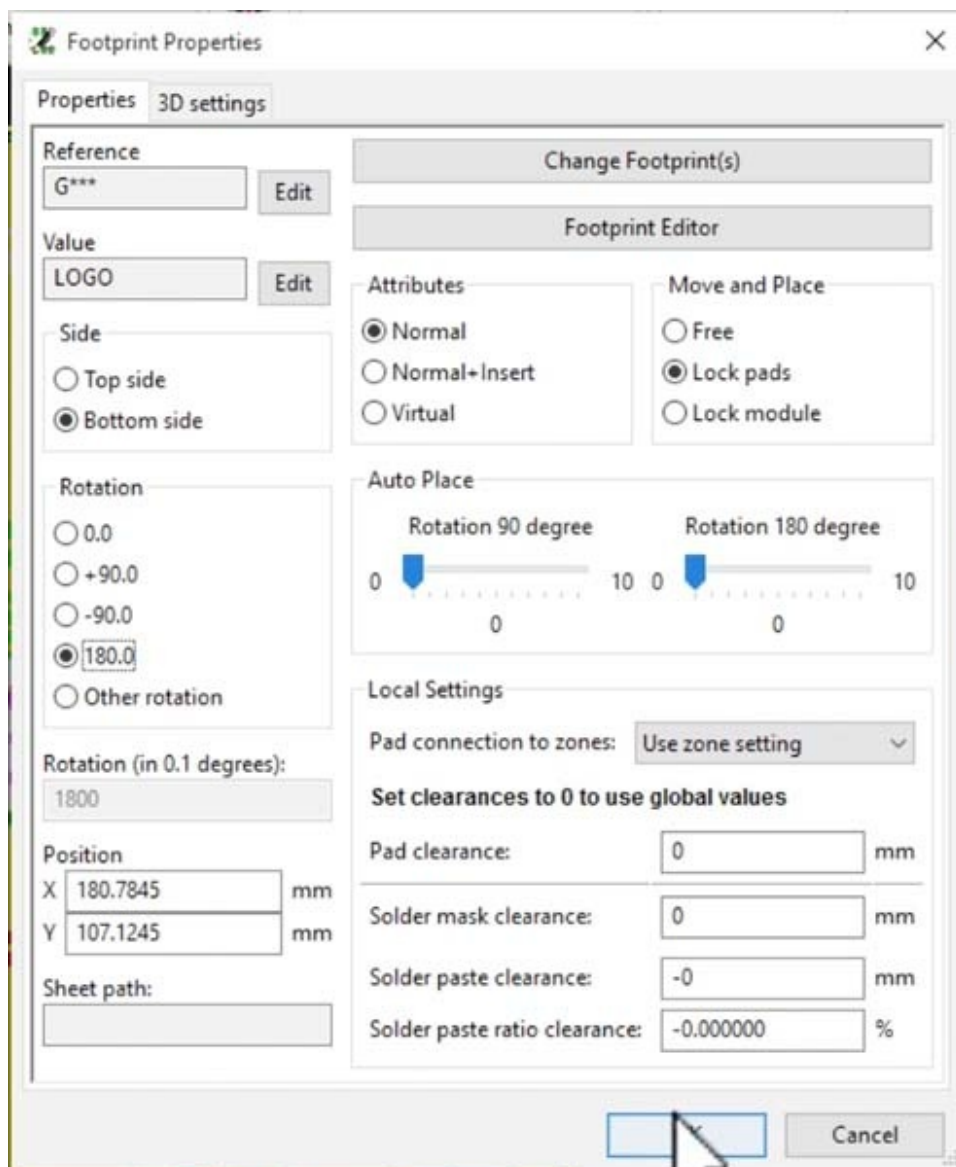
The Footprint chooser will appear. Click on List All.



Type part of the name of the footprint in the filter, and double click on the footprint to add it to the canvas.

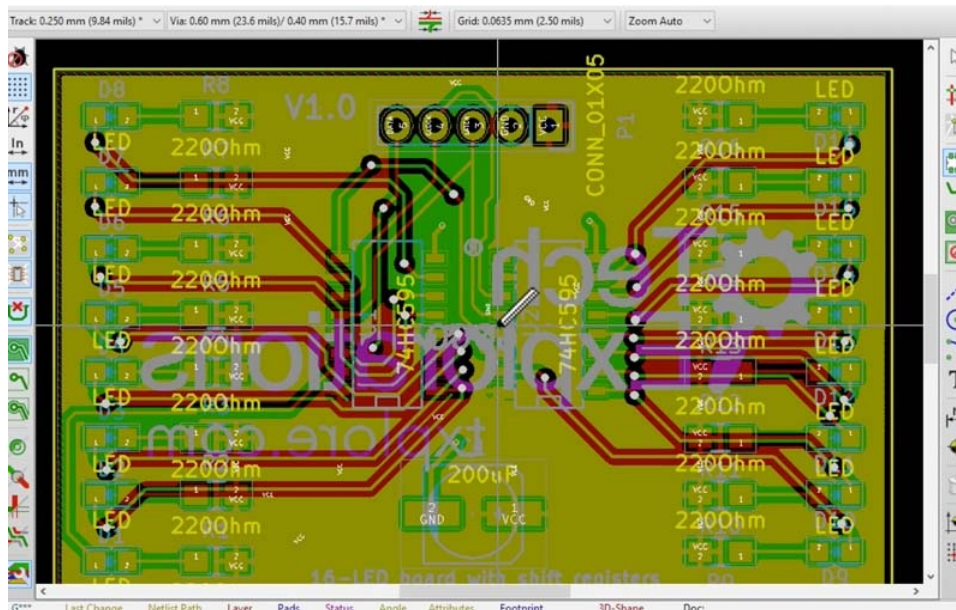


The footprint is on the canvas, but it is placed in the front silkscreen layer. Put the mouse over the footprint and type “E” to edit its properties.



Select the Bottom Side radio button, and the 180.0 radio button for the Rotation. Click OK.

Select and move the footprint in place in the middle of the board. The new footprint is now in place in the back silk screen layer. It looks like this:



The decorative graphic is now in place.

Let's have a look in 3D preview:

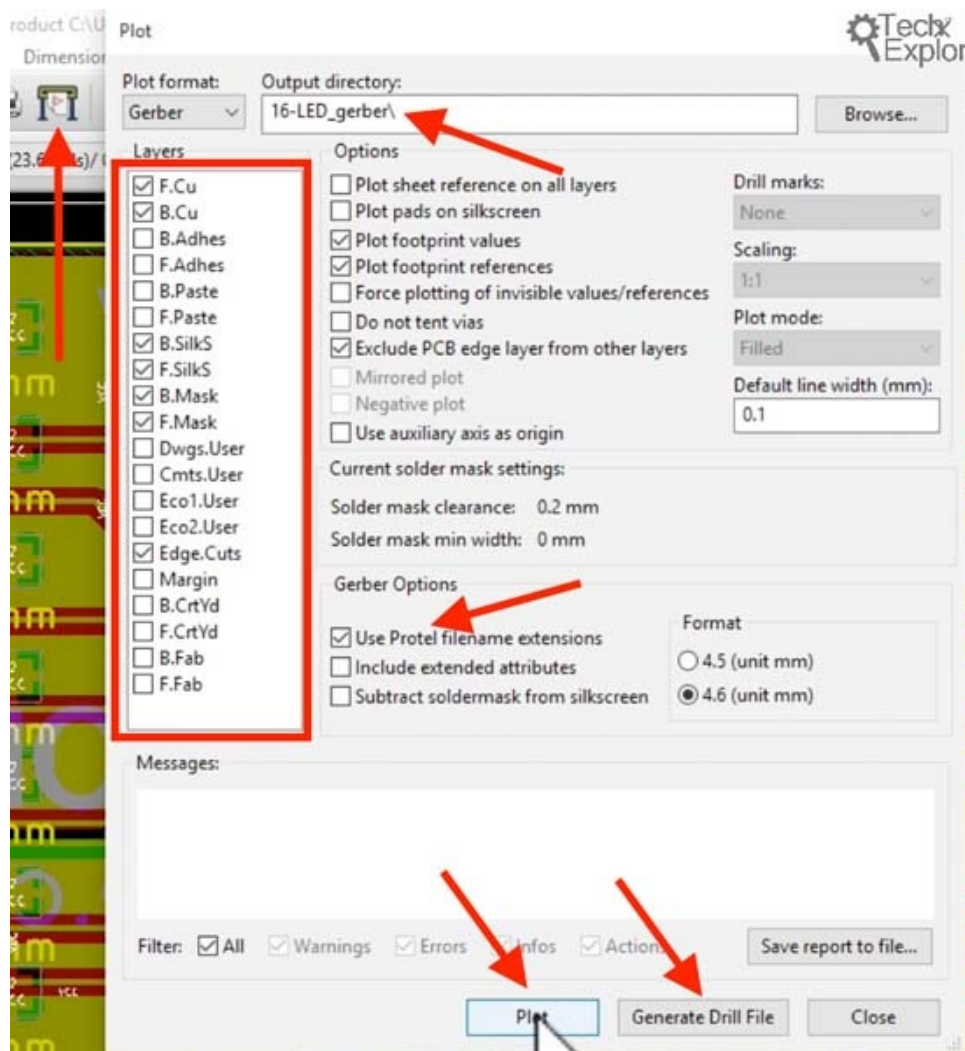


The decorative graphic, in the back of the PCB.

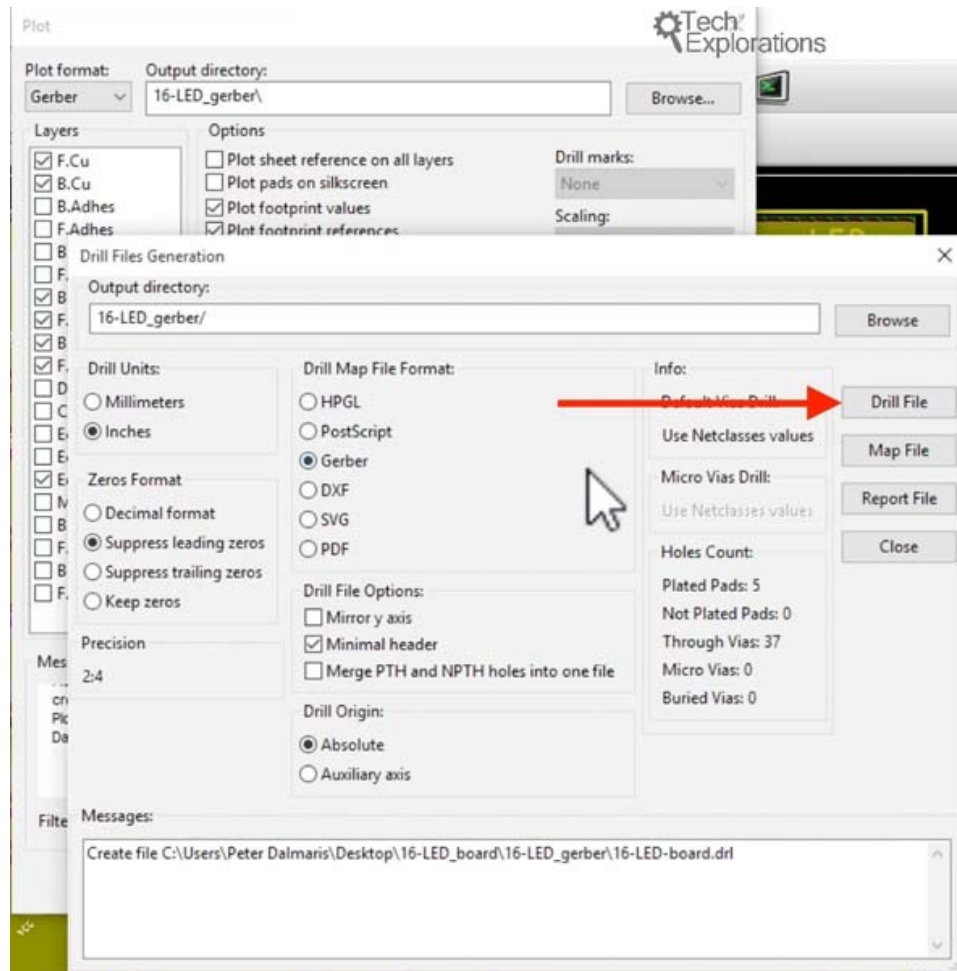
Save the project. In the next project we will complete the project by exporting the Gerber files and uploading to the manufacturer.

Chapter 59: *Export the Gerber files*

We can now go ahead and export the Gerber files. Click on the plotter button:



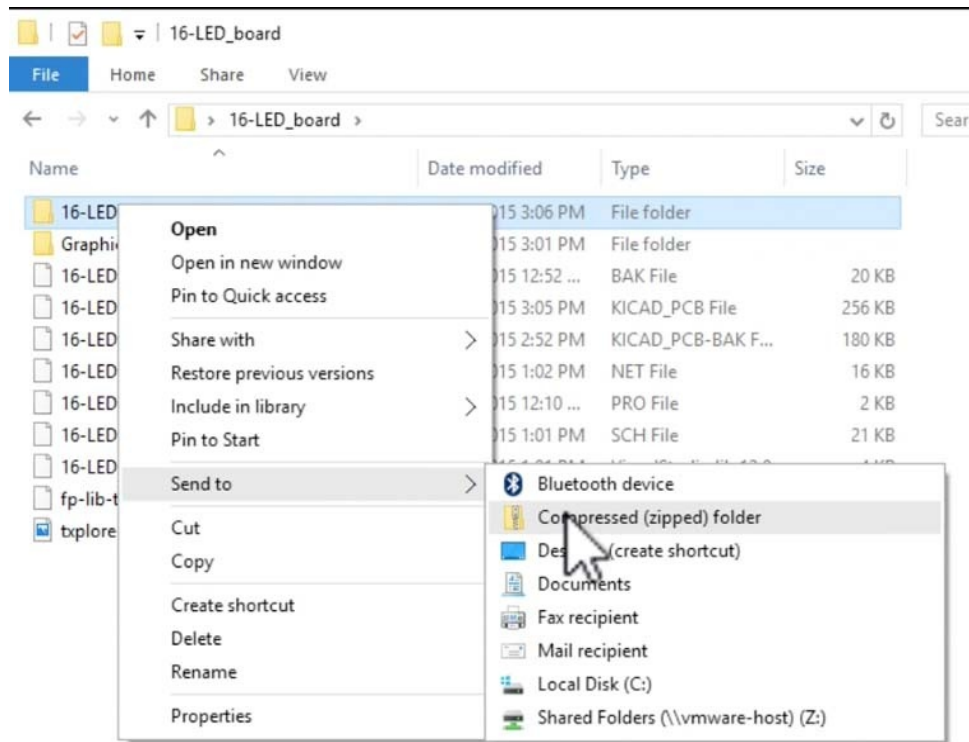
Click on the Plotter button to bring up the Plotter window. Choose the required layers, set the export directory and Gerber options, and click on Plot. Then click on Generate Drill File.



The default settings in the Drill Files Generation window are fine. Click on the Drill File button.

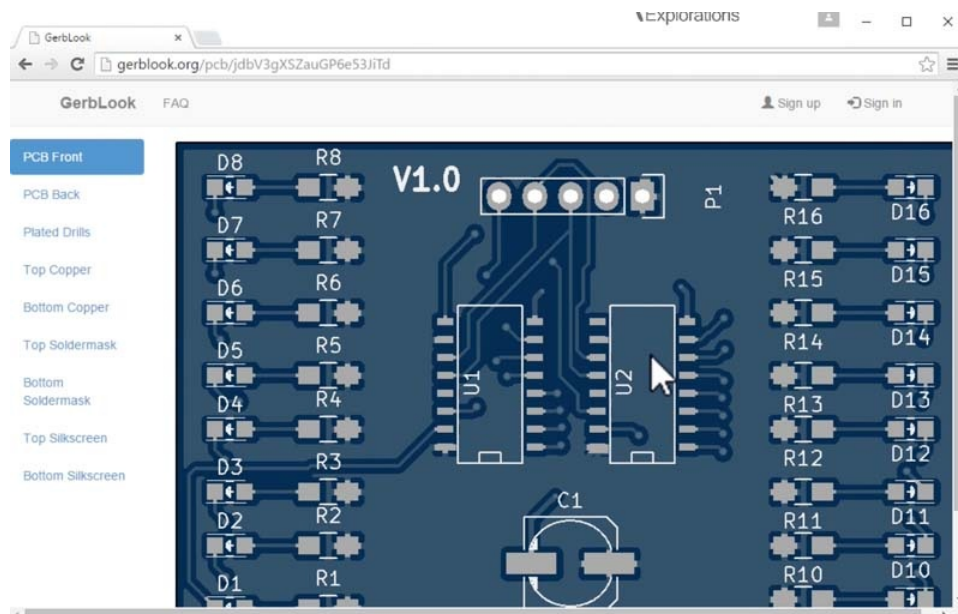
Close the two Gerber generation windows.

Use your file system browser to find the directory that contains your Gerber files, and create a ZIP archive:



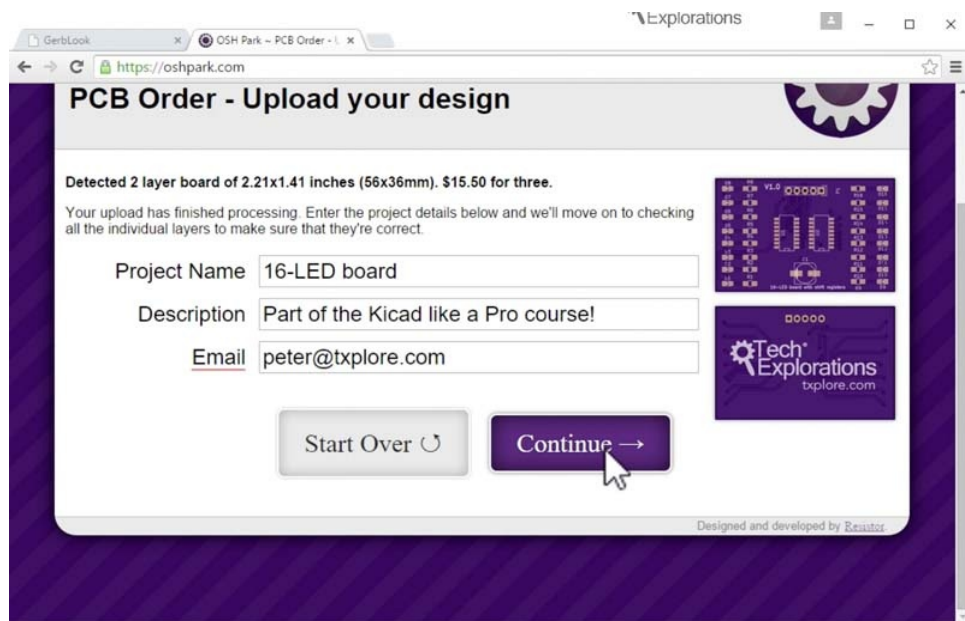
Create a ZIP archive from the directory that contains the Gerber files.

Use your web browser to go to gerblook.org so that we can test our Gerber archive. Upload the ZIP file to Gerblook, and if everything goes well, the different layers will be rendered in the browser:



Gerblook renders the layers in the Gerber archive, so it is a valid archive!

Once Gerblook confirms that we have a valid Gerber archive, we can go ahead with the order. In the case of OSH Park, I uploaded the ZIP archive to my account, and ordered my boards:



Ordering my boards from OSH Park!

A few weeks later, I received the PCBs in the mail!

This project is now complete, so what's next? Please read the conclusion for a few ideas.

PART EIGHT

Conclusion

Chapter 60: *What's next?*

Congratulations!

By completing this course, you have gained the skill set you need to design the vast majority of PCBs that a hobbyist, and in a large extend a professional, maker would ever dream of.

Going forward, the best advise I can give you is to use these skills as much and as often as you can. Look around your desk and you lab. Look inside drawers and boxes. Find breadboarded projects that you placed aside because you didn't want to take them apart. Convert these breadboard circuits into PCBs. Start with the simple ones, move on the the more complicated ones.

Try to minimize the size of your boards, and increase their elegance. Place components in nice tidy grids, minimise the total number of vias, and minimise the total length of each track.

Once you master this process, think about Kicad's more advanced features.

Let's say that at some point you wish to build a PCB that contains a high-speed RAM, or a microprocessor running at a few hundreds megahertz. You will need to extend your skills in order to tackle a problem like this. Look for documentation that can be of assistance. Or, simply look inside your computer, appreciate the skill of the engineers that created such elaborate and efficient designs and learn from them.

Like much in engineering, there is no limit to what you can learn and make!