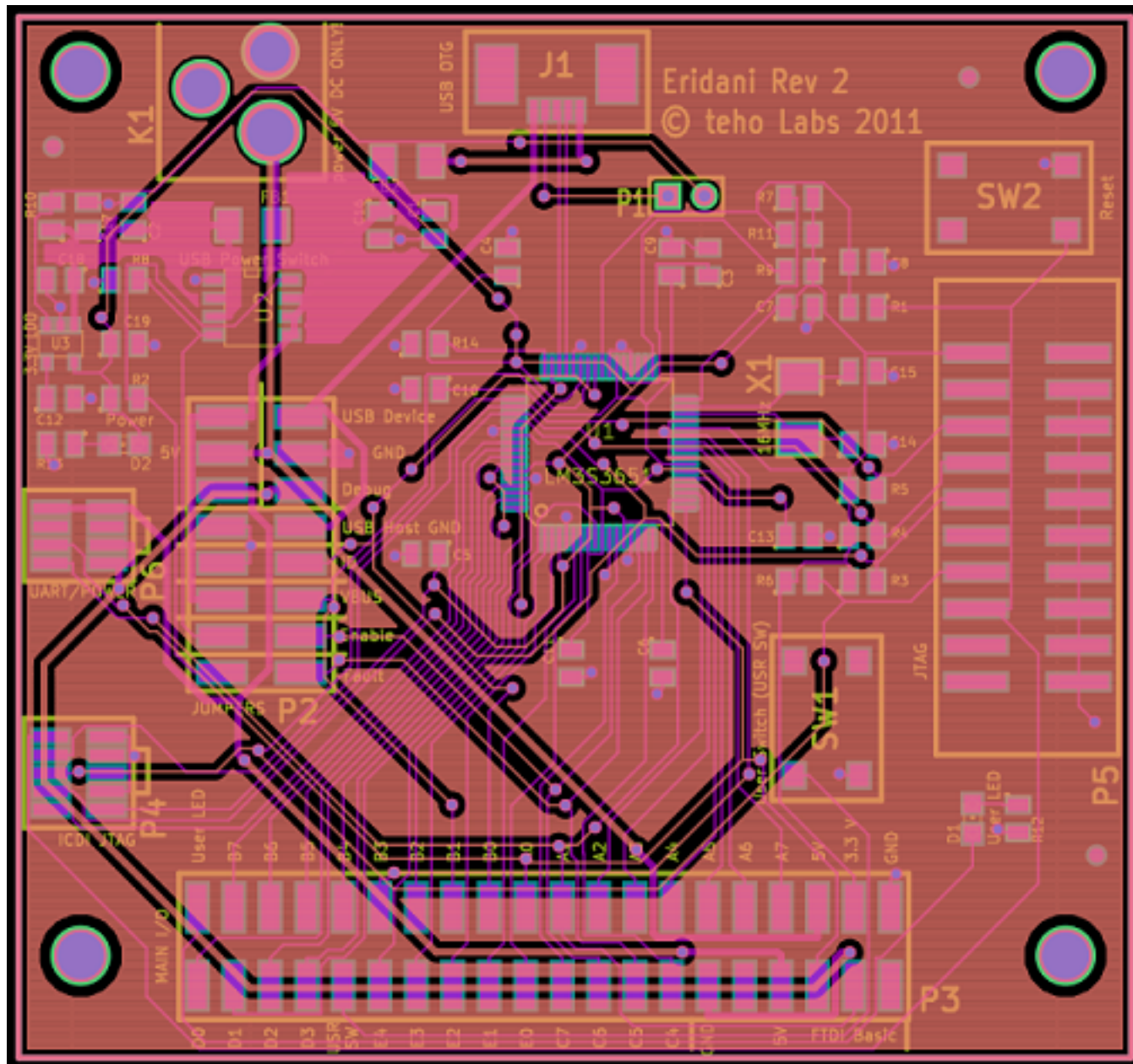


We are often asked what tools we use. We like to stick to free tools here at teho Labs. There are two good reasons to do this: one a large user base for GPL software helps insure it remains actively maintained, and two it saves a lot of money which keeps overheads low.

For the hobby community or small business there are essentially 3 choices for PCB layout: Eagle, gEDA and KiCad.

We use KiCAD to design our PCBs like this one, for Eridani:



While Eagle is certainly the most popular with people starting out we don't think it is the best choice. Why you shouldn't use Eagle? Eagle is fine for what it does. Cadsoft is nice to give away a free version for hobby use, but their license is non-commercial. What that means is you aren't supposed to make things you are going to sell with it. That might sound fine if you are just starting out with PCBs, you might only want them for your own projects, but one day you might have a bright idea that you do want to sell something you made and then you are trapped. You won't want to learn a new package and you won't want to remake all your custom parts, so you will pay perhaps even 750 dollars to get what you need, it isn't a good deal. What is more Eagle isn't even a great CAD tool, the free version is restricted in many ways, as is the cheap light version.

Your other options are gEDA and KiCAD. gEDA is really a big electrical design tool suite that happens to do PCBs as such it isn't all that integrated, this makes it harder to use than KiCAD.

KiCAD is free, complete, has lots of footprints built in, and no restrictions on how you can use the designs you produce with it. It integrates with a free to use very nice autorouter for parts of the circuit that aren't signal critical. It is GPL so it will always be free.

In this tutorial you will learn how to make a part, footprint, schematics and PCB layout. All of these skills are essential though it does make the tutorial a lot longer than it would be otherwise. We cover many items you will have to do only once as well. If you can get through the tutorial you will know enough KiCAD to start making lovely PCBs.

For Eagle users it should be noted KiCAD is setup logically differently in that parts are not so closely linked with footprints. We find this a great feature for parts with multiple packages, some people find it annoying. We will show how it can be useful along the way.

First you'll need KiCAD you can grab it here: <http://www.kicad-pcb.org> We will assume you are using Windows but on other platforms this should be nearly the same. Once you have downloaded it you can just install it on your system with the defaults.

The main KiCad window consists of 5 buttons. In order left to right the sub programs are:

- eeSchema - a schematic program
- CVpcb - a program linking schematics to footprints
- PCBnew - a PCB layout tool
- GerbView - a gerber viewer
- Bitmap2Component - a program for converting bitmaps for use on PCBs etc.

We like gerbv more than GerbView, and Bitmap2Component isn't essential so we will just do the first 3.

In this tutorial we will make a simple PCB breadboard power supply including exporting gerbers for fabrication.

The first step is to make a new project. *Click file -> new* and give it a name. Note there are buttons for many of these things but we will generally use menus to avoid millions of screenshots. Let's call the project "breadPower".

First we need to decide what parts we need. We will need a barrel jack, a diode, 2 capacitors and a 3.3 V LDO for this project.

We have selected PJ-202A for the barrel jack (the standard jack at SparkFun but also available for less from Digikey and no doubt others), NCV551SN33T1G for the LDO, we will assume generic through hole diodes and electrolytic capacitors for the rest.

Let's get started. *Click on eeSchema*. You may get a box saying the file doesn't exist, that just *click OK*. Let's dive right in and make some parts. We could do this with the library editor but there is an easier way online: <http://kicad.rohrbacher.net/quicklib.php> We actually only need to make 1 part: the voltage regulator. Taking a look at the datasheet it is a 5 pin part with a counter clockwise pin numbering. http://www.onsemi.com/pub_link/Collateral/NCP551-D.PDF

Change component name to NCP551. We want "DIL" and N = 5. Click on assign pins.

The pins 1-5 are: Vin, GND, En(able), N/C, Vout.

Type those in the boxes. Next change the types to: power in, power in, input, unspecified, power output.

These pin types are just for DRC checking, it is good to assign them but you can just specify everything as bidirectional if you don't want to bother with it.


Click the preview button. If you don't abbreviate Enable as En, it will run into the part label. You can increase the horizontal margin if this matters to you. You also can change the location of these labels in the symbol editor, this tool is just faster for making simple parts.

Everything looks nice on our preview, *click build library component*. This will give us a file to save. Let's *rename it on save as "myParts" and save it a new directory under our project's folder called libraries*.

The library is a text file so when we make additional parts we can append it to the library file we just made, just cut and paste. Parts start with "DEF" and end with "ENDDEF".

We need to add our new library to the project. *Click preference -> library*. Then *click on the add button and browse to the library you just saved! If you are running KiCAD in Windows 7/Vista, be sure to save the libraries to a folder that doesn't need administrative privileges (IE not the libraries folder in program files/kicad/)*.

You also may define a search path for libraries with this dialog box. You will be asked to save the project when you click ok. *Do so*.

Having added our one symbol we needed we can make our little circuit! *Click place -> component, select by browser, "myParts", NCP551, then click insert component in schematic* . *Now click in the schematic somewhere to place the LDO.*

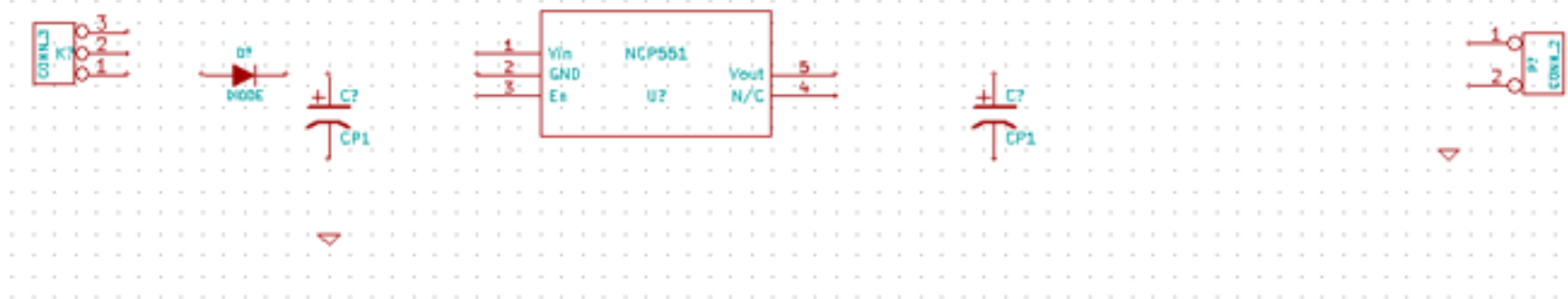
In KiCAD most things have shortcuts. 'a' is the shortcut for placing a component you can see this in the menu system <A>. Right clicking is also vital in KiCAD, you can right click on the part you just placed. See how it says Move <M> you just learned another vital shortcut key.

Hit 'a', now type cp1 and hit enter. As you can see CP1 is the name of a polarized capacitor. CP is also polarized, while C is unpolarized. We need to add a diode and a barrel jack.


Oh no! Don't we need another component for the barrel jack? Well maybe. But items that are really nothing more than pins we tend to use the 'conn' library. Because KiCad lets you link parts to any footprint we don't have to really make unique symbols for everything! We just need a connector with 3 pins. (We consider this an advantage to KiCAD over the Eagle like approach.) *Look for one in the browser under 'conn'*. Find it? CONN_3. That's good for input, how about output? CONN_2. Last we need a diode, this is under device 'DIODE'. *Place these three parts on your schematic*.

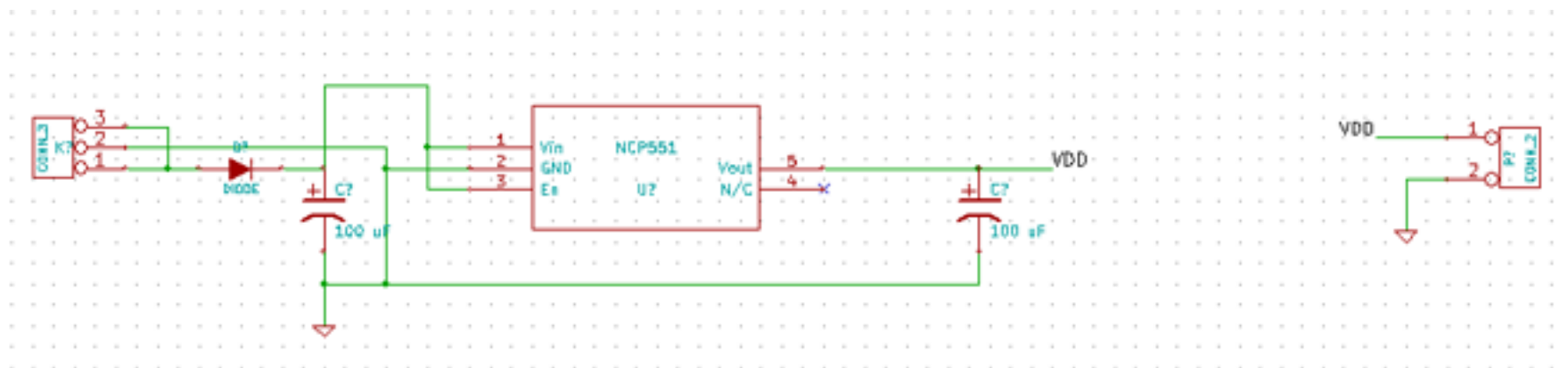
We need one last thing and that is the ground symbol. Let's add two of them, you'll see why in a moment. *Click place -> power <P>. (Or as you now see just p). Type in GND and place two on the schematic*.

Now move the parts around till they look something like the screenshot below. You can rotate a component by right clicking and selecting rotate or the 'r' key!.



If we look at the [datasheet for PJ-202A](#) we can see pin 1 is high, pin 2 is GND, and pin 3 is connected to GND when there is no barrel jack in.

With this info we can wire up the circuit. *Click place -> wire <W> (or the icon ).* You should end up with the following design seen below:




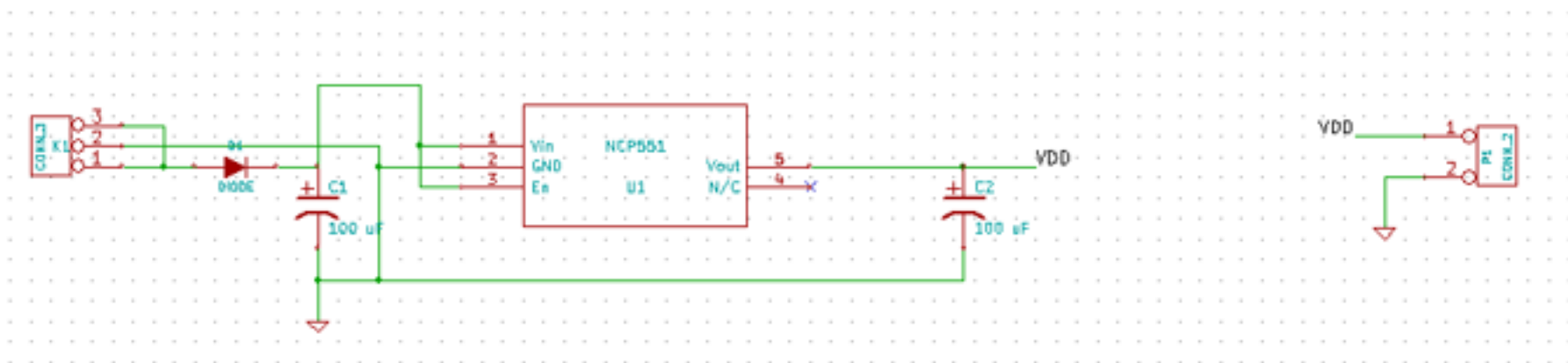
You should notice two things on my schematic you may not have done a small X and two labels of VDD. *If you attached wires delete them.* (You can right click, use the del key or the eraser icon). See the X icon? That is the not connected, *use it or place -> no contact flag to add the X to the N/C pin.*

Next let's add the labels. Labels are a good way of making a schematic readable. They tell the program what nodes of the circuit are attached without using wires. This lets you use wire to connect circuit blocks then labels to connect blocks.

Place a label and call it VDD on the Vout pin and the pin we will use for our output. *Click place -> label <L>.* The small square on the label will disappear when it is connected correctly to a node of the circuit.


The next step is to add values and annotations to the schematic. Let's give the capacitors a value, say 100 uF each. To change the value you can *just hit 'v'* when you are over the component with the select tool, or you can find it with the right click menu system.

The next step is to annotate. The annotate schematic button is this one  in the top bar. Pick any options you like the defaults are fine and *click 'Annotation' and then 'Close'.* You should now have this:




The components now have unique labels. *Save your schematic!*

You could run a DRC check now and eeSchema will try to help you find where you made mistakes connecting inputs to outputs and such. Given the symbols I choose, I get 2 errors, both aren't real mistakes.

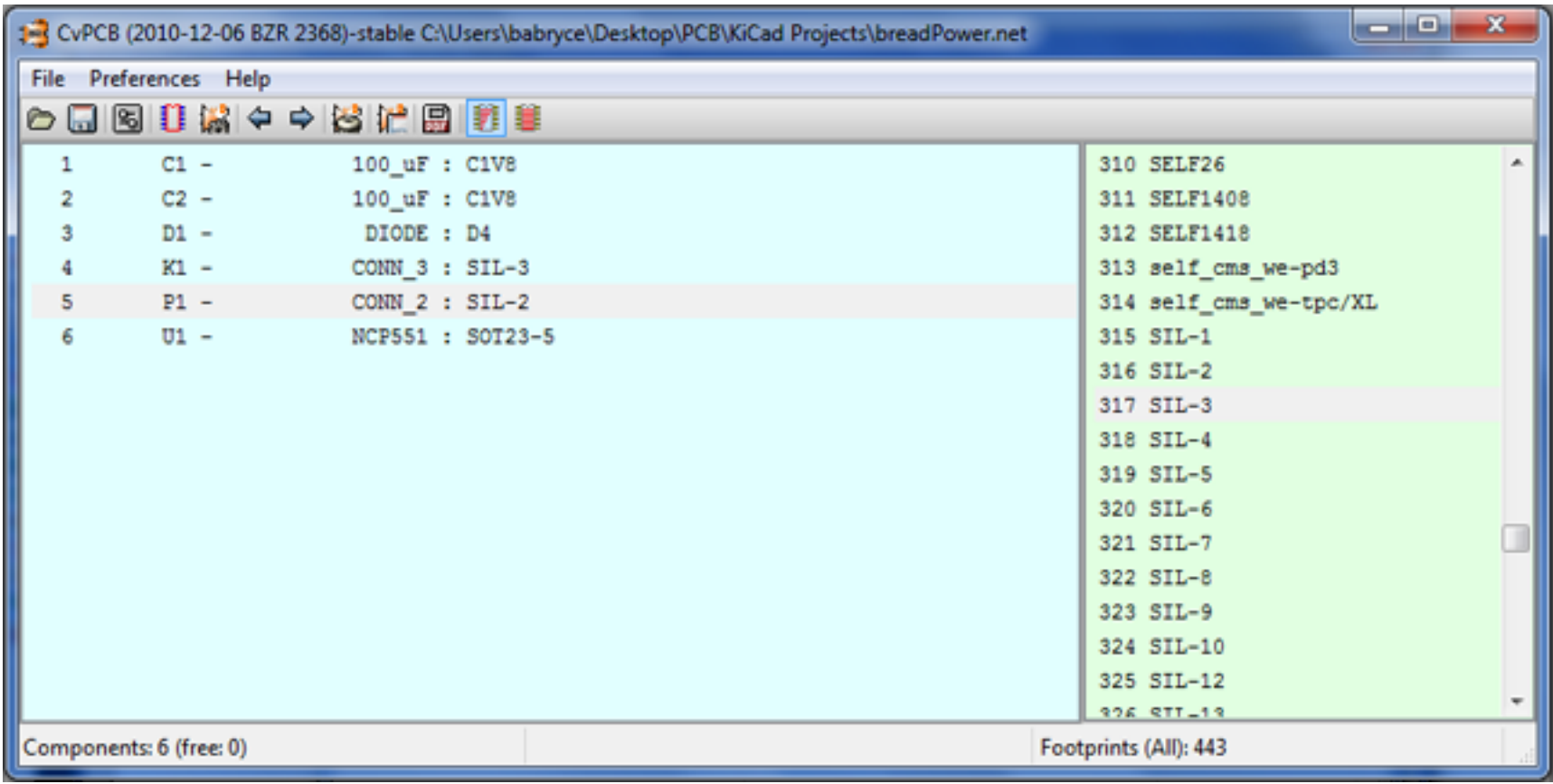
The final step in eeSchema is to generate the netlist. To do this: *click the generate netlist button , then netlist (in the default PCBnew tab). Save the file.*

We are done with eeSchema, you can close it. Now we will assign footprints to our components with CVpcb. *Click on CVpcb.* You may get an error about a library being missing just click okay. Now we see a list of our symbols at the left and possible footprints at the right.

The default view is filtered, that is a symbol can restrict the options of footprints that should be assigned to it if you know better you can *click the "display full footprint list" icon .* In this case I will do that because I want to use C1V8 as my footprint for the polarized caps.

What is C1V8? I hear you ask. Well there are a set of footprints that come with KiCad that are pretty extensive. To see what they look like the easiest way is to look at the footprints.pdf in the share/modules/footprints_doc directory under where KiCad is installed. If you print this PDF at 100% you can check the footprints against the parts you have!

Assign the components the following footprints by double clicking the name at right, to get the C1V8 footprint you'll have to turn off the list filtering, otherwise just use CP4 or whatever you like, it is just axial verse radial part. At this point you should realize we don't have a footprint for the barrel jack. We will have to make one, for now assign anything you like.



You can get to the first letter of the part by typing it. But then you will need to scroll. Once you are done this *click save*.

Now open PCBnew! Again just ignore the error and click OK.

The first thing to do is to fix what we know is wrong: the footprint that is missing.

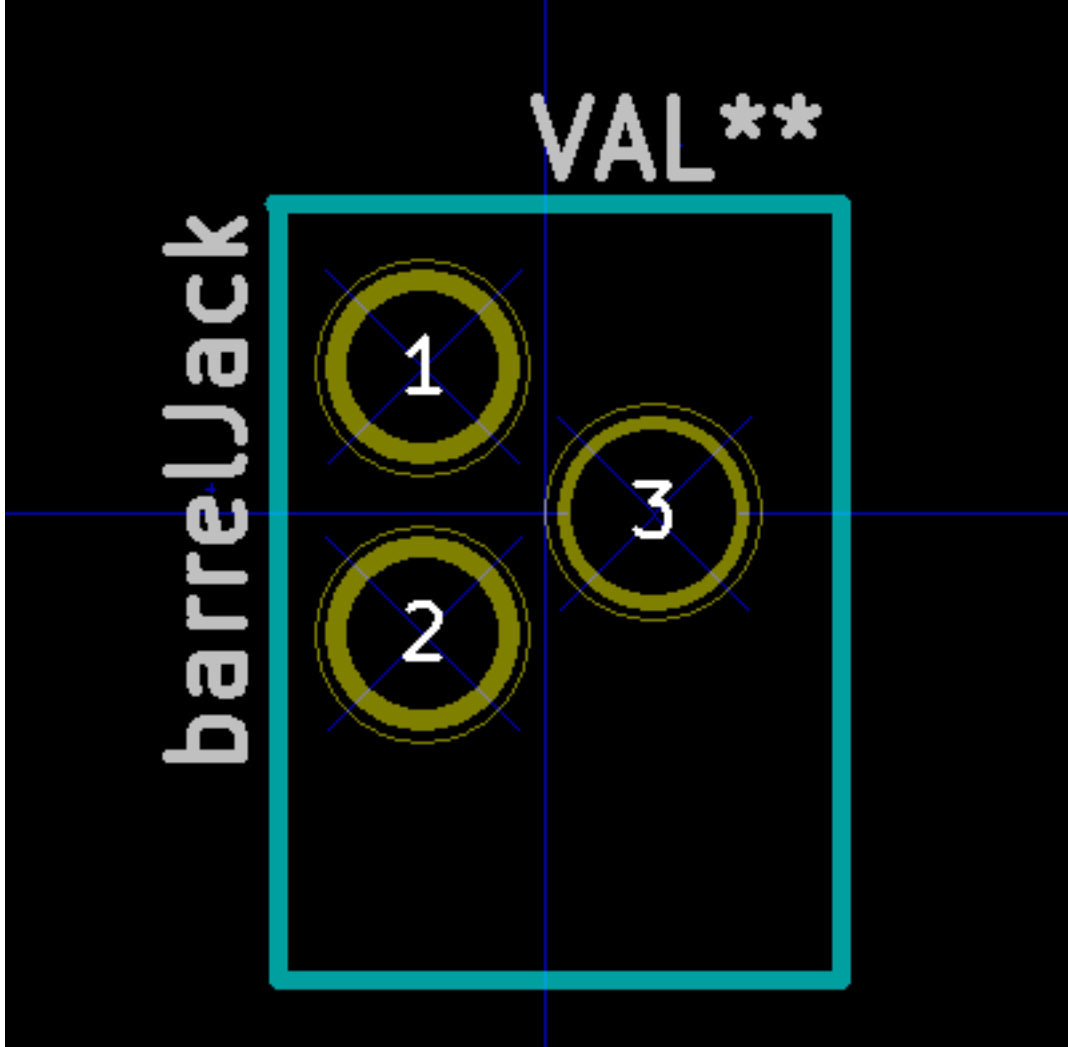
Click on the open module editor. 

Normally you would now choose a working library, but since we don't have any libraries of our own yet we won't. Click on the new module button  and call it barrelJack. Next click on add pads  and place 3 pads on the module. Right click on the first pad and click edit pad <E>.

You will see there are both SMD and through hole (standard) options here, this happens to be a though hole component but we mostly find that we make SMD components. The layers at the right can be left as is. However we need to change the hole sizes, and positions of these holes.

We choose 0.13 and 0.16 for the drill and size respectively. From the datasheet you can figure out the positions or just move the pins around however you want unless you are actually going to make one of these. (For pins 1,2,3 the [x,y] cords should be: [-0.1, -0.12], [-0.1, 0.1], [0.09, 0], though you could offset or rotate the part).

Add a line around your part and move the labels to reasonable places. It should look something like this:



Now we, will save it to a new library. *Click file -> save module to new library. Create a new folder called modules in your work folder like you did for the libraries (not in program files) and save it as 'myParts'.*

In the future you can add your parts to the 'myParts' module library you just made.

Close the module editor.

Click preference -> Libraries and add the new module library you just made, you can also add a search path if you wish. Click okay and save the project.

Go back to the module editor and 'select working library', 'myParts'. Next click load module from working library, list all, barrelJack. Finally, 'save module in working library'. Exit the module editor.


If you have a working library to start with these last extra steps just like adding the library to the path won't be needed, but for whatever reason the save module to new library process does not make one of the files you need!!! By the action we just took we created it.


Making modules isn't that hard as you can see. And about 50% of the work we just did we don't have to do next time because everything will be set up right already.

Close PCBnew! (Save the board on exit).


Open CVpcb again!

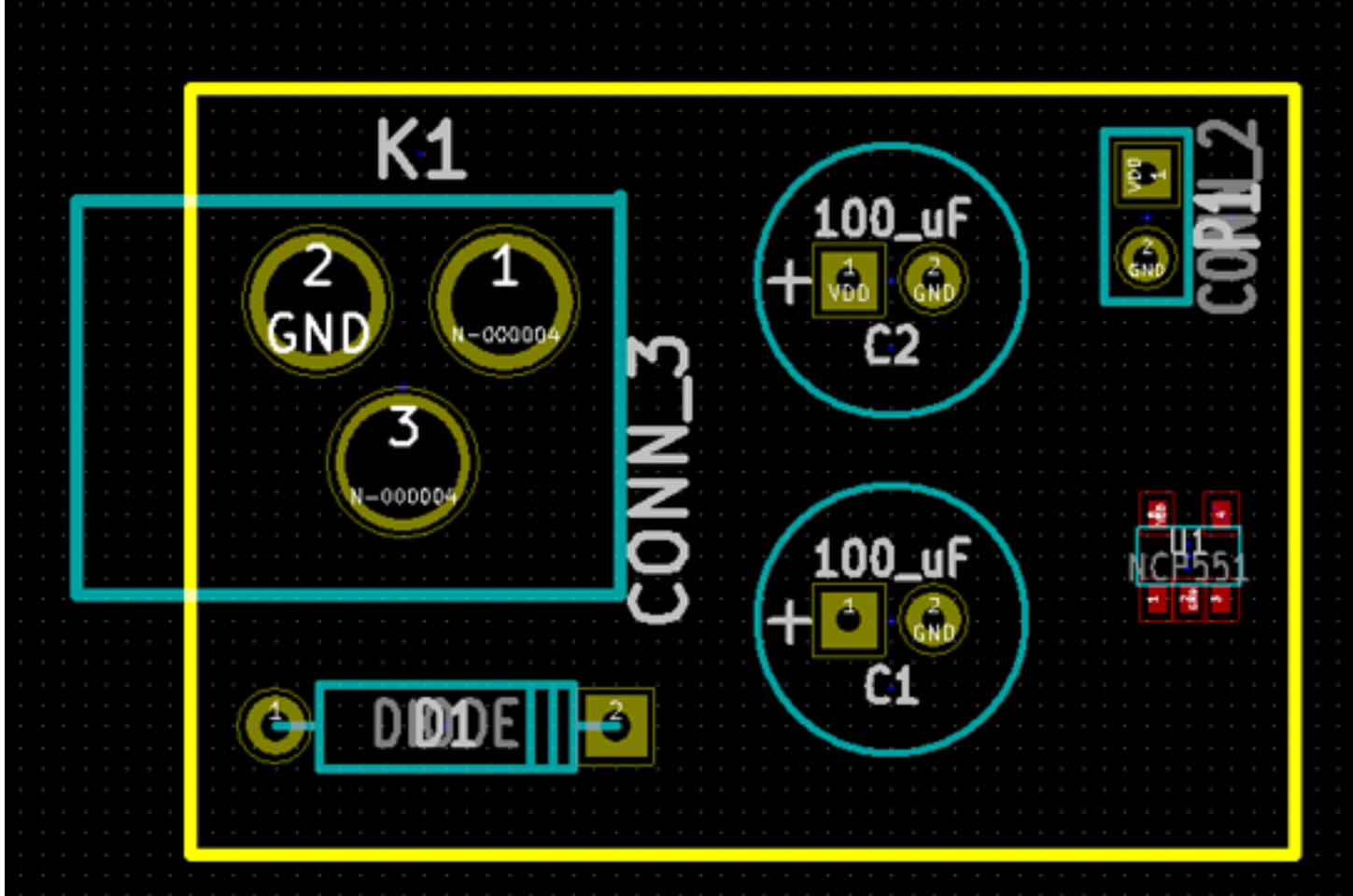
Goto K1 and click on the right hand panel and type 'b', look there is our barrelJack footprint! Double click it and resave the netlist as before.

Open PCBnew! Click on 'read netlist' . The path should be correct for the netlist you just saved with CVpcb, but you can change it if it isn't. Click read current netlist. The footprints will appear! Click close.

Now click on footprint mode . Next right click in the center of the sheet and select Glob Move and Place -> Move all modules. OK. Now the modules will be where you clicked and spread out.


Use 'm' and 'r' to move and rotate your modules.

After that is done select PCB edges from the layer bar at right and then the line tool . Single click where you want to start drawing the board edge and draw a box around your layout. It should look something like this:




Next we want to define some node classes. *Click on design rules -> design rules*. Here you see the default clearance, line width, etc for PCB lines. This LDO actually isn't high enough power to worry about this but we are showing it to you for when it will matter.

Let's add a new class called 'power' and change the track width to 24 mil from 8. Next add everything but ground to the 'power' class (select 'default' at left and 'power' at right and use the arrows).

This is a two layer board and we will use the backside for a GND plane. *Select the 'front' as the layer, and click the track button* . You will see the so called ratsnest. *Using 'v' to place vias and change layers you can route the whole board by hand now (you also may change layers by clicking on the layer at right).*

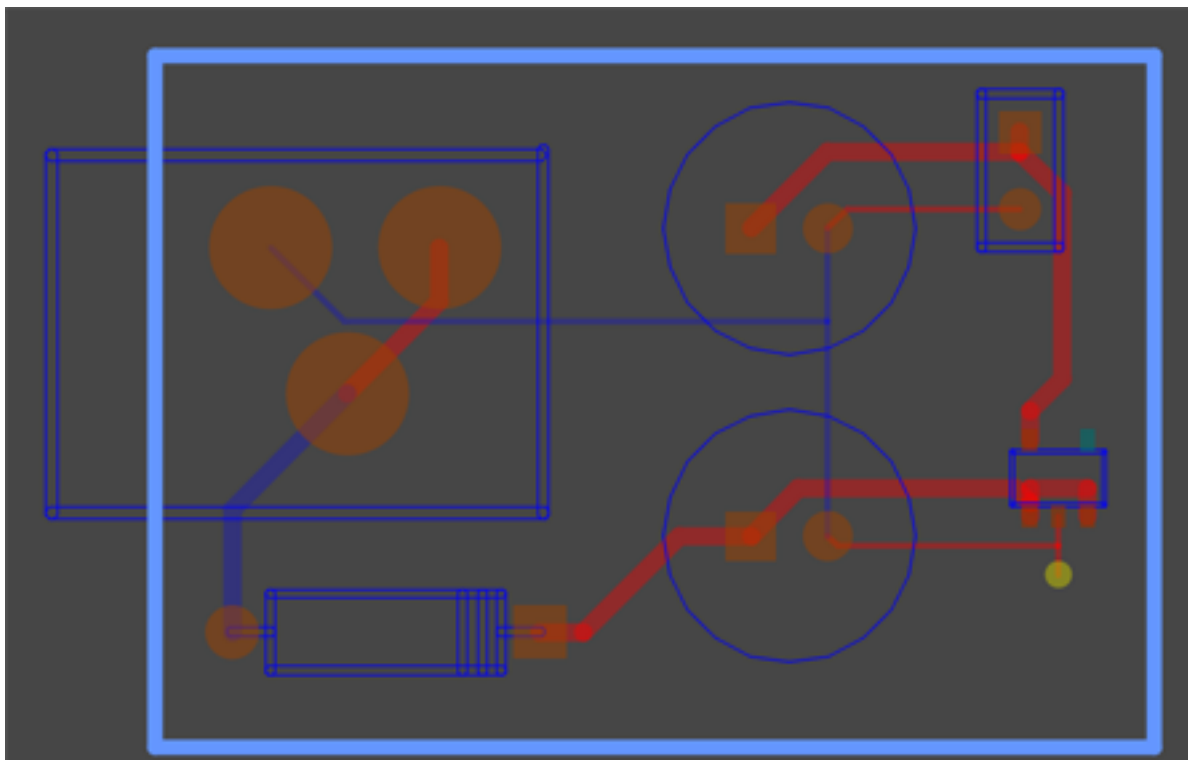
Let's not route the whole board because I want to show you how to do autorouting also. Since we want to use the back as ground, *click on the LDO GND pin and draw the wire out some way type 'v', then double click to end the line.*

For a board this simple routing by hand would be quick and easy, however, for board with lots of pins you might want to use an autorouter. You should always route critical traces by hand first though! The autorouter we will use is very nice freerouteing.net router, which will not change any of the traces you put down before you ask it to route.

To use this router, *click autorouter icon* . *Click export DSN file and save it. Next click the launch button, you will need java to use this router. The program will launch. Click on open design and browse for the DSN file you just saved.*

This router has some nice features that KiCad doesn't have built in yet for manual routing as well but we are here for the autorouter. The autorouter has two main steps, first it will route the board and then it will optimize it. Full optimization can take a long time, however, you can stop it at any point after it routes your board. If the pass count to route the board gets above 30 your board probably cannot be autorouted with this router. Spread your components out more or rotate them better and try again. The goal in rotation and position of parts is to lower the number of crossed airlines in the ratsnest.

Click autorouter to run the autorouter. For this simple board it won't take very long. This is the result I got:

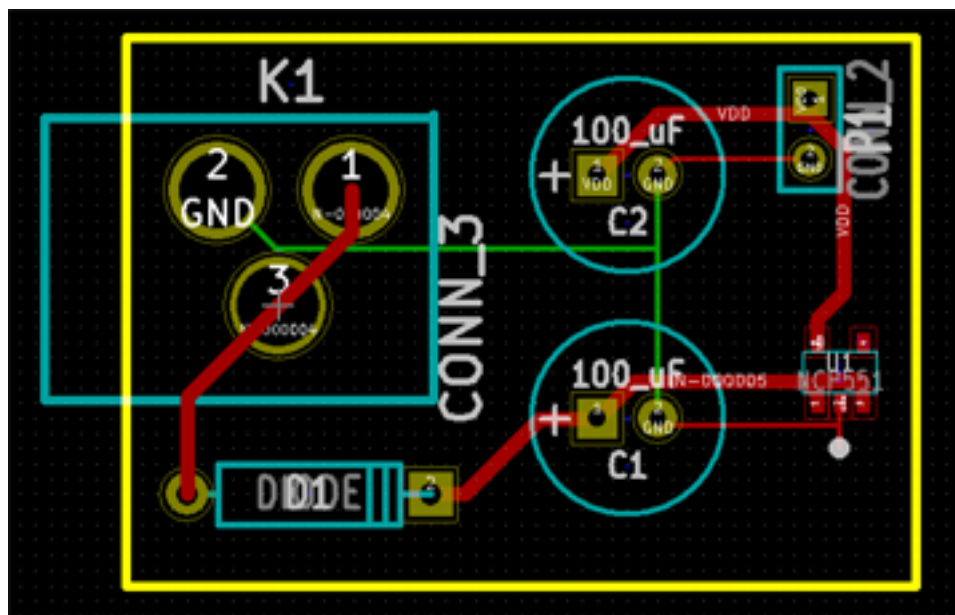



Click file export session and save it with the extension .ses. We never bother to save the rules file.

Back in PCBnew, click on the back import .ses button and select the file you just saved. Click okay to any dialog boxes and close the autorouter box.

There is one thing we don't like which the autorouter did and that was to place the trace to the diode on the backside, so I am going to delete those traces and route them myself, using the delete key and the routing tool. *Try to do this yourself.*

There all better:

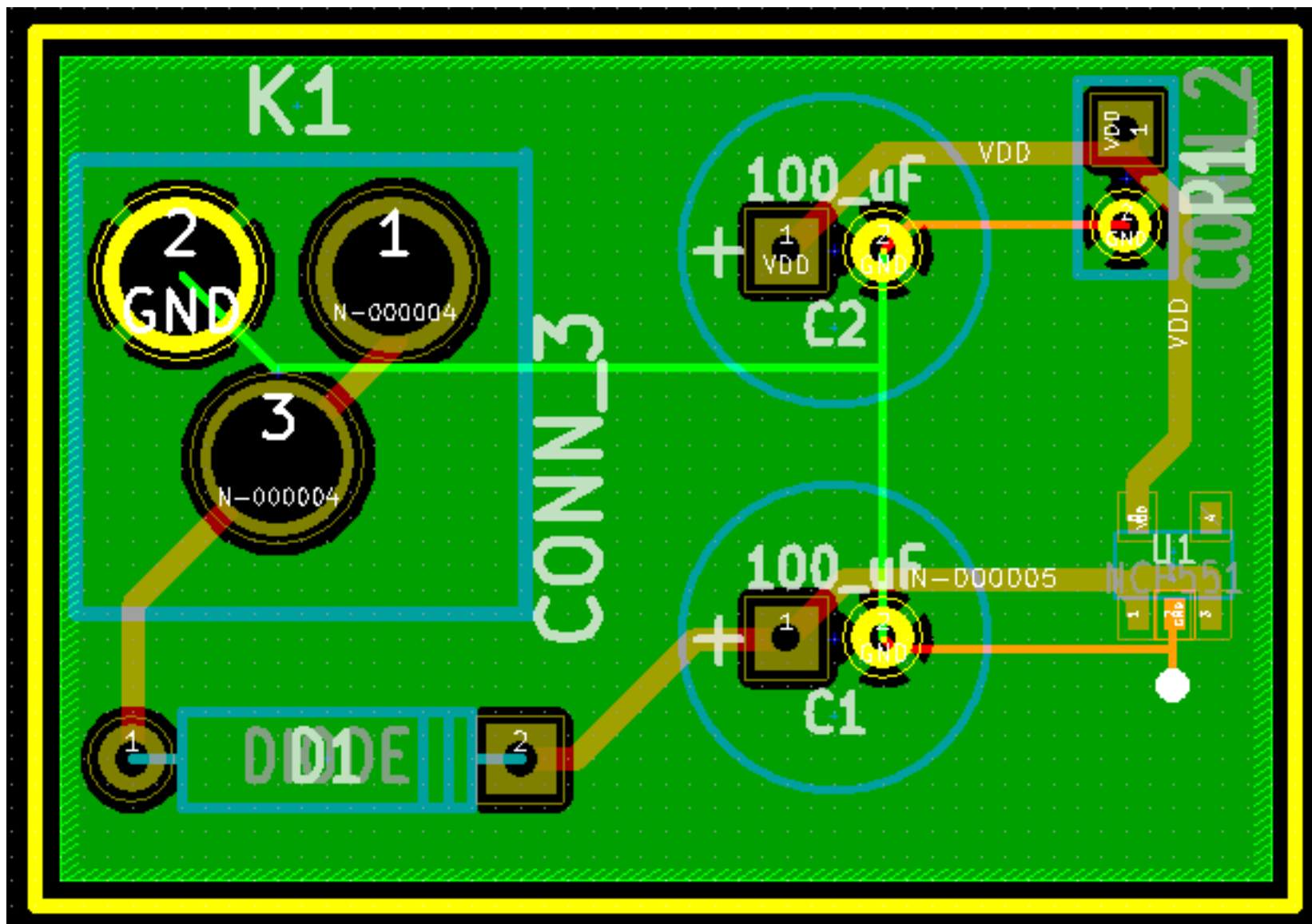


Finally let's add a zone to fill the back with the GND layer. Select back as the layer and then the zone tool . Click inside your PCB outline just like you were about to start a rectangle just smaller than the outline (in a corner). A dialog box will come up, you may choose to include or exclude pads here what the clearances need to be, etc. We will just leave everything as default and select the GND node and then click okay. Now, draw a box and double click to finish at the corner you started in.

Next right click in zone and select 'Fill or Refill all Zones'.

Basically you are done now with the board but if you want to manufacture it we should do a few more things. First, for the barrel jack footprint we designed sticks over the board edge. We want to save cost on the board so it is okay for this to happen but if we turn it in this way the fab house might not cut the board down. So let's edit the module in place and trim it. Hit 'e' on the module and select module editor. Redraw the silk outline (erase the old one and put a new one in). Finally choose update module in current board. This changes just the instance in the board you are working on. You may ask why have the longer outline at all, well you don't have to but for us we find it helps indicate the rotation of these kinds of parts as well as gives us the full size of the part for clearances.

We now have this:



I am quite happy now, though I could change some names and move some labels.

Let's export it as gerbers for BatchPCB or similar to use, the settings below are for BatchPCB. Click File -> Plot Your dialog box should look like this:

Layers:

Copper Layers:

- ☒ Front
- ☒ Back

Technical Layers:

- ☐ Adhes_Front
- ☐ Adhes_Back
- ☒ SoldP_Front
- ☐ SoldP_Back
- ☒ SilkS_Front
- ☐ SilkS_Back
- ☒ Mask_Front
- ☒ Mask_Back
- ☐ Drawings
- ☐ Comments
- ☐ Eco1
- ☐ Eco2
- ☐ PCB_Edges

☒ Use Proper Gerber Extensions

☐ Exclude pcb edge layer

☐ Print sheet reference

☐ Print pads on silkscreen

☒ Print module value

☒ Print module reference

☒ Print other module texts

☐ Force print invisible texts

Pads Drill Opt

- ☐ No drill mark
- ☒ Small mark
- ☐ Real drill

Scale Opt

- ☐ Auto scale
- ☒ Scale 1
- ☐ Scale 1.5
- ☐ Scale 2
- ☐ Scale 3

Plot Mode

- ☐ Line
- ☒ Filled
- ☐ Sketch

Plot Origin

- ☒ Absolute
- ☐ Auxiliary axis

Plot Format

- ☐ HPGL
- ☒ Gerber
- ☐ Postscript
- ☐ Postscript A4
- ☐ DXF Export

☐ Plot mirror

☐ Vias on mask

Default pen size

0.0060

HPGL Options:

Pen size ("):

0.015

Pen Speed (cm/s)

20.000

Pen ovr ("):

0.002

PS Options:

☐ Plot negative

Output Directory:

C:\Users\babryce\D

Browse...

Plot

Save Options

Generate drill file

Quit

Messages:

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-Back.gbl> created

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-Front.gtl> created

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-SoldP_Front.gtp> created

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-SilkS_Front.gto> created

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-Mask_Back.gbs> created

Plot file <C:\Users\babryce\Desktop\PCB\KiCad Projects\breadPower-Mask_Front.gts> created

We have included the solder paste layer for submission to a stencil maker. But the PCB house will not need it.

Click on plot.

Now let's create the drill file, *click on generate drillfile*, it should look like this:

Drill Units:

- ☐ Millimeters
- ☒ Inches

Zeros Format

- ☐ Decimal format
- ☐ Suppress leading zeros
- ☒ Suppress trailing zeros
- ☐ Keep zeros

Precision

- ☐ 2:3
- ☒ 2:4

Drill Origin:

- ☒ Absolute
- ☐ Auxiliary axis

Drill Sheet:

- ☒ None
- ☐ Drill map (HPGL)
- ☐ Drill map (PostScript)
- ☐ Drill map (Gerber)
- ☐ Drill map (DXF)

Drill Report:

- ☒ None
- ☐ Drill report

HPGL plotter Options:

Speed (cm/s)

1

Pen Number

1

Options:

☐ Mirror y axis

☒ Minimal header

Info:

Default Vias Drill:

Use Netclasses values

Micro Vias Drill:

Use Netclasses values

Holes Count:

Pads: 11

Through Vias: 1

Micro Vias: 0

Buried Vias: 0

OK

Cancel

Click ok and save the drl file.

You are **DONE!**

You should review your gerbers with gerbv or gerbview but that's it. Zip the files and send them off to a PCB fab house.

We know that seems like a lot, and truthfully it is, but you just did all the hard setup stuff, and almost all of the common tasks for a low frequency 2-layer PCB.

For more useful info be sure to check out our [docs section](#).