

Keyboard PCB Guide

So you want to make a PCB for a keyboard? Don't know where or how to start? Well you've come to the right place!

Table of Contents

- [Setting Up](#)
- [Schematics](#)
- [Associating Components and Footprints](#)
- [Generating Netlist](#)
- [PCB](#)
 - [Component Placement](#)
 - [Edge Cuts](#)
 - [Routing](#)
 - [Mounting Holes](#)
- [Production](#)
 - [Gerber Files](#)
 - [Manufacturer](#)
 - [Components](#)

Setting Up

We're going to need [KiCad](#). Download it, install it, and you should be ready to go!

... almost.

We're going to want some libraries, too. I like to use Hasu's keyboard_parts [component library](#) and [footprint library](#). /u/techieee also has a good [switch footprint library](#).

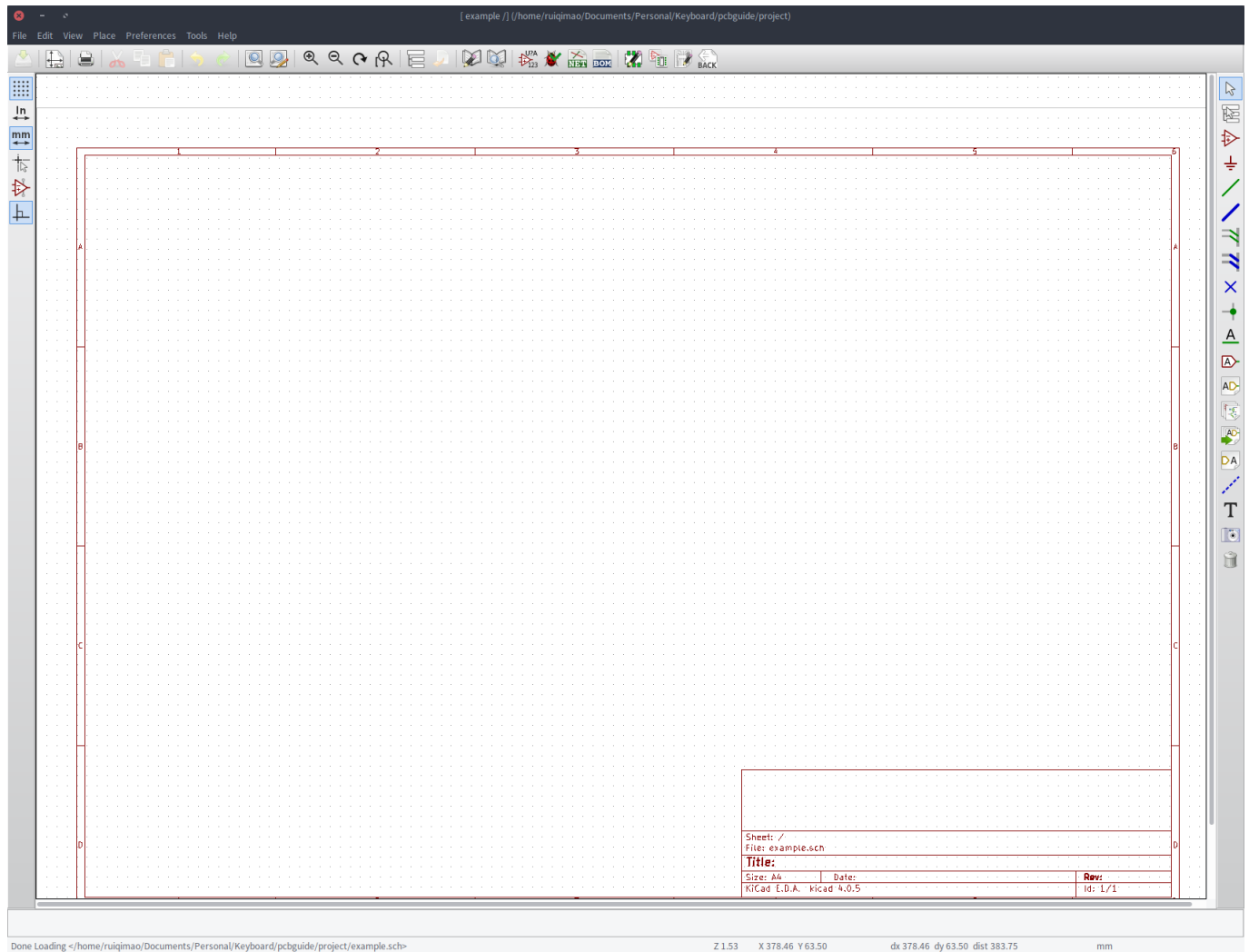
Download all of these and we should be good to go!

Make sure you also have the official KiCad libraries as well. Those should be included with your KiCad installation.

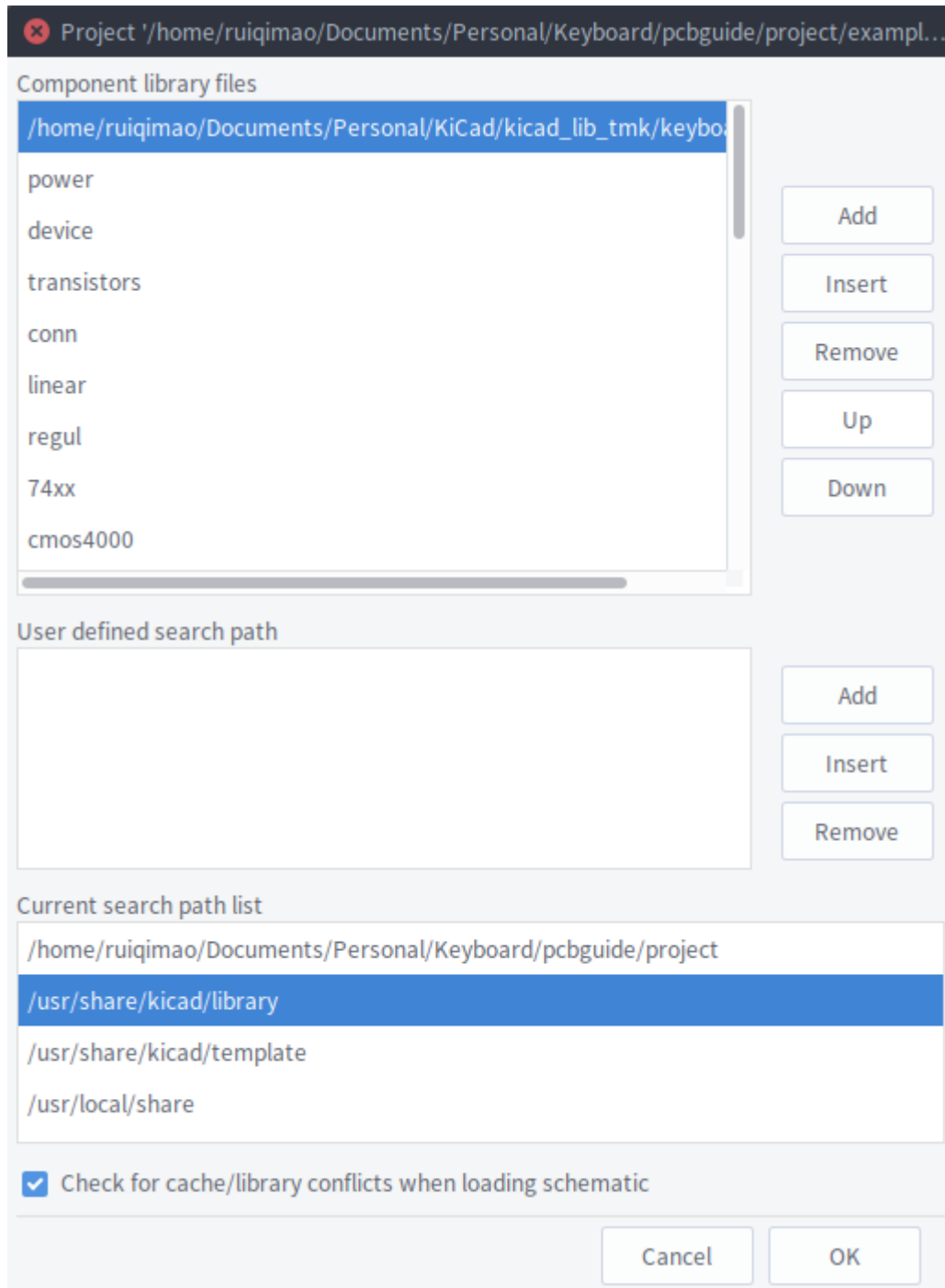
Schematics

Start up KiCad and create a new project (File > New Project > New Project). Name the project whatever you want. For the purposes of this guide, I'll be calling it "example". Very creative, I know.

We'll start by designing our schematics. Double click on your .sch file and you should be greeted with an empty schematic sheet:



Let's add our component library. At the top of the window, click on Preferences > Component Libraries. Then, click on "Add" and find the `keyboard_parts.lib` file from Hasu's library. Scroll down to the bottom of the component library list and find the library you just added. We want to move that to the top of the list, so your list should look like this:

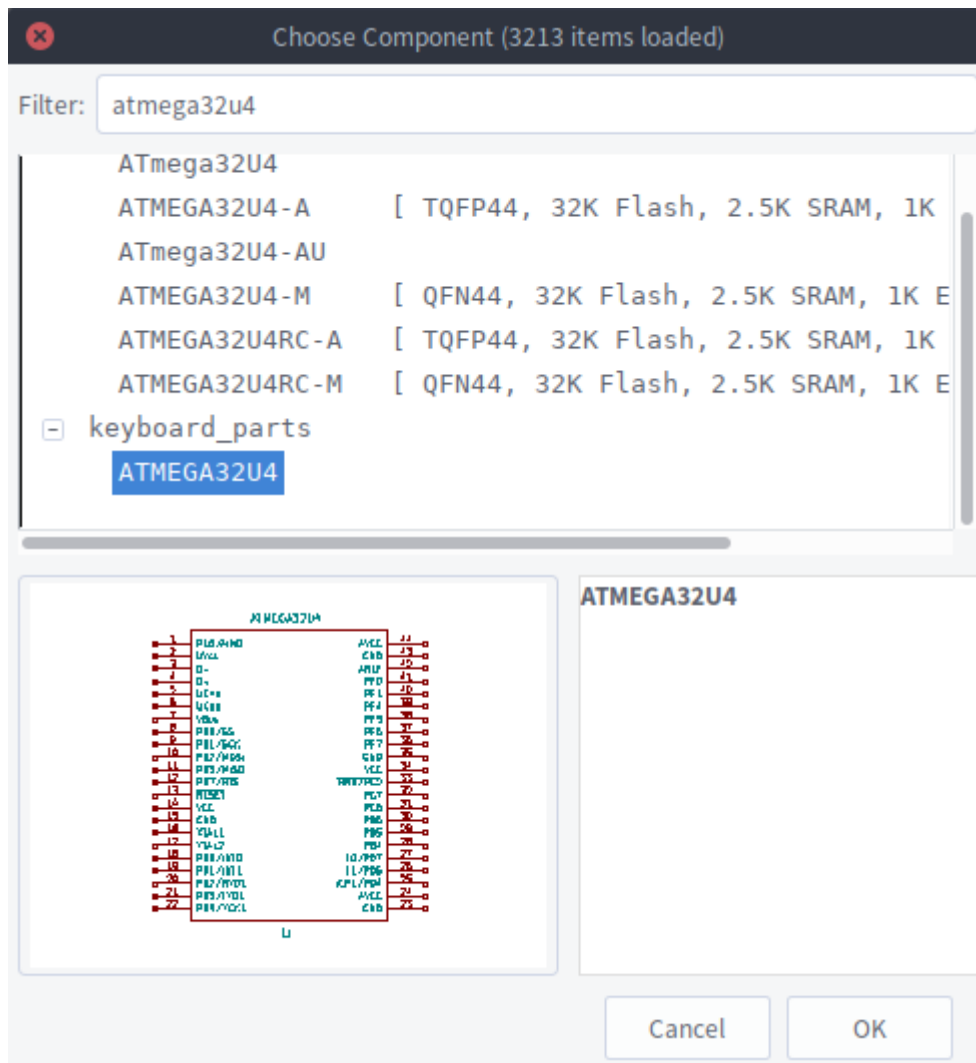


Click "OK" and we're ready to go. We're about to get real technical here, so buckle up.

To start off, here's a list of basic commands:

```
m: pick the component up and move it
g: drag the component up and move it while keeping wires attached to it
c: copy the component
e: edit the component
r: rotate the component
y: mirror the component
del: delete the component
esc: abort!
```

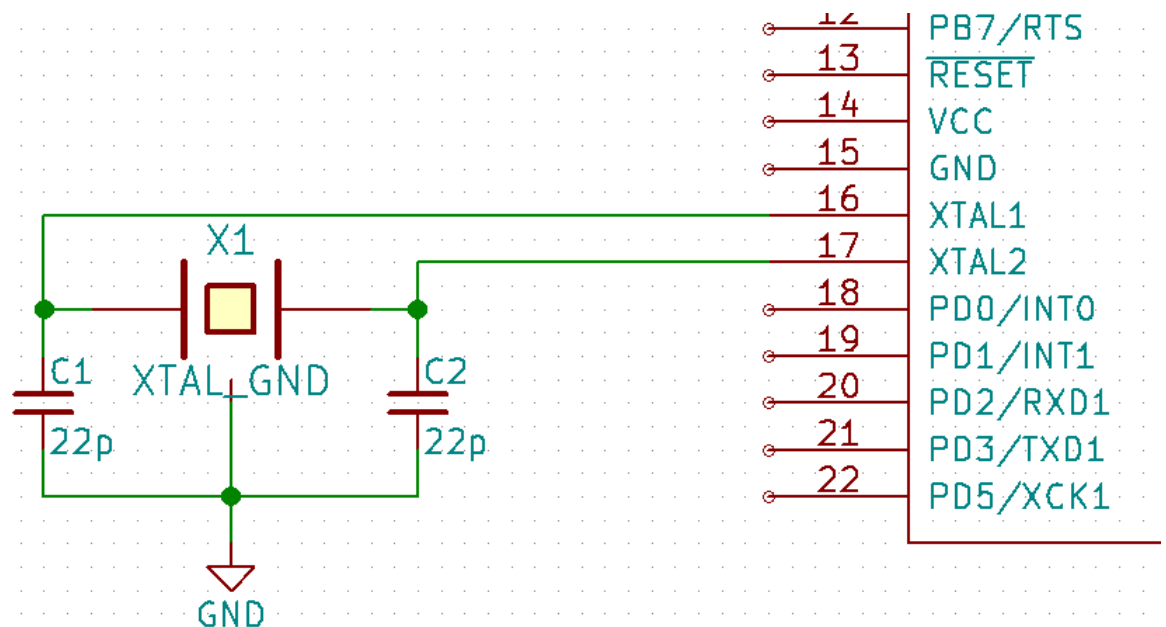
Do Place > Component. Your cursor should turn into a pencil. Click anywhere on the sheet. Look for ATMEGA32U4 in the keyboard_parts library:



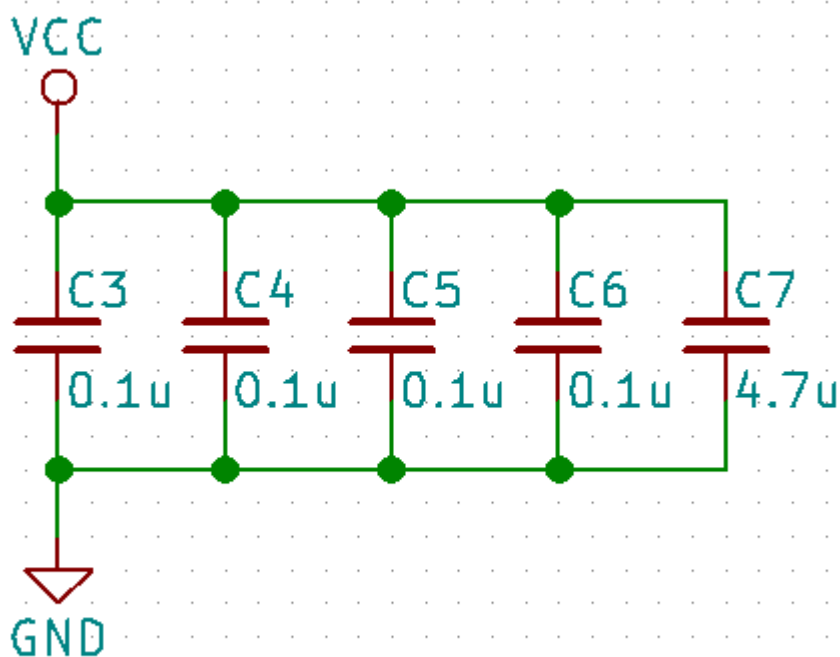
Click OK, then click on the schematic sheet again to place the component. This is our controller. Edit the component and change the reference from "U?" to "U1". This is the unique name that we're going to use to refer to this particular component.

The next part we'll want to place is the crystal, which is the part that tells the controller how fast to run. Look for the XTAL_GND component and place it next to the controller. Change the reference to X1.

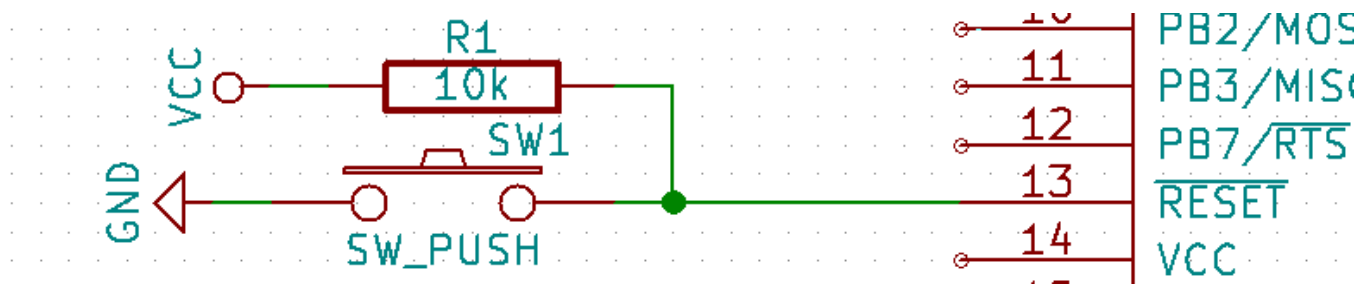
Next, we're going to want to add 2 decoupling capacitors (C_SMALL). These capacitors will basically help prevent the signal to the controller from accumulating too much noise. There's a formula for determining the capacitance you need for these capacitors, but for now, we'll use a crystal with 18pF load capacitance, so these decoupling capacitors will be 22pF. Name them C1 and C2, and change their values to 22p. Also add a GND symbol to represent ground, and connect everything using the wire tool (green line on the right) like so:



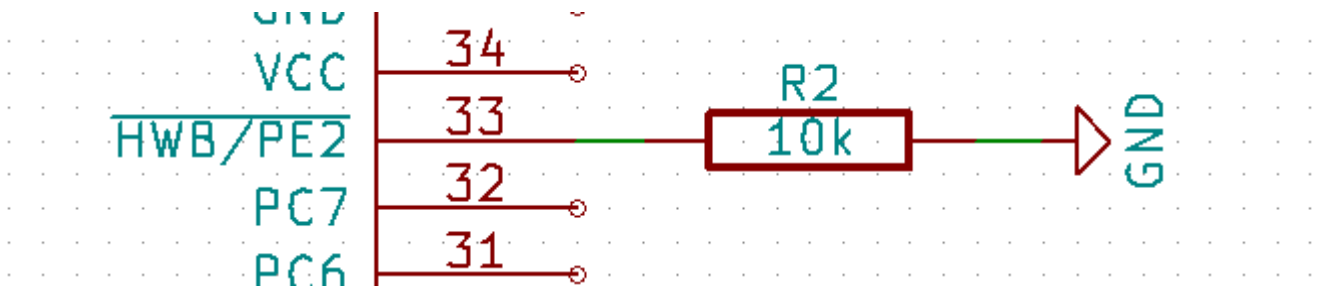
Next, we'll add decoupling capacitors for VCC, our power source. We will generally want one 0.1uF capacitor for each VCC/AVCC on the controller and one 4.7uF capacitor for UVcc. In our case, we want 4 0.1uF capacitors and 1 4.7uF capacitor, like so:



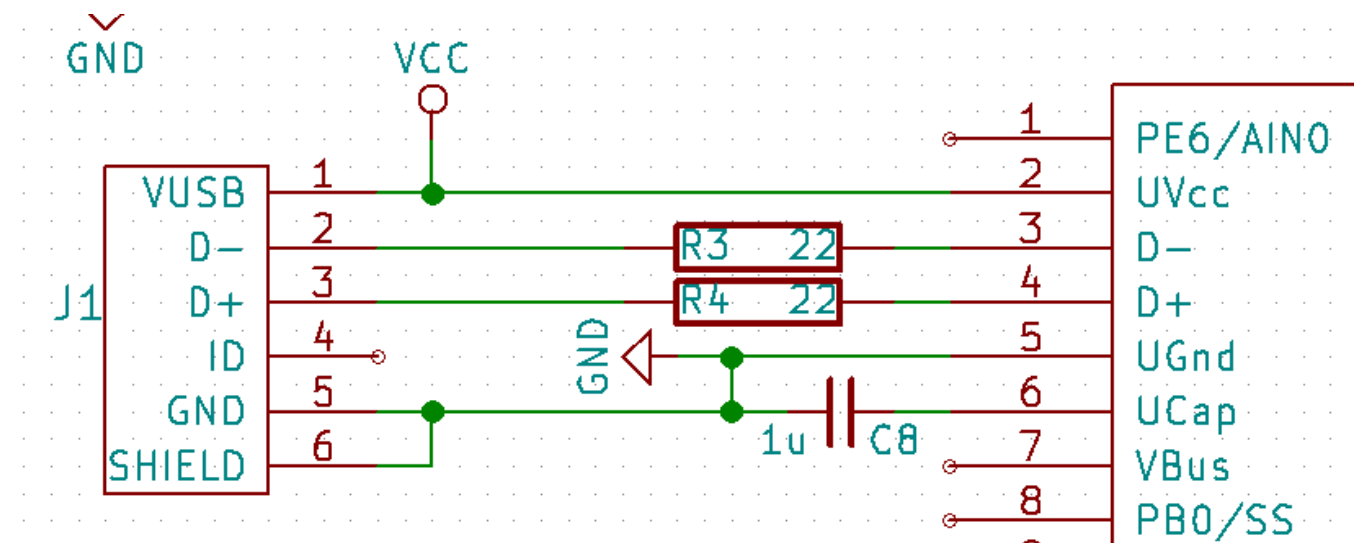
Let's hook up a reset switch. For this, you'll want a switch (SW_PUSH) named SW1 and a 10k resistor for pullup (R) named R1. If you want to know why we want a pullup resistor and what a pullup resistor even means, [here](#) is a good explanation from Sparkfun. But for now, here's how it should be hooked up:



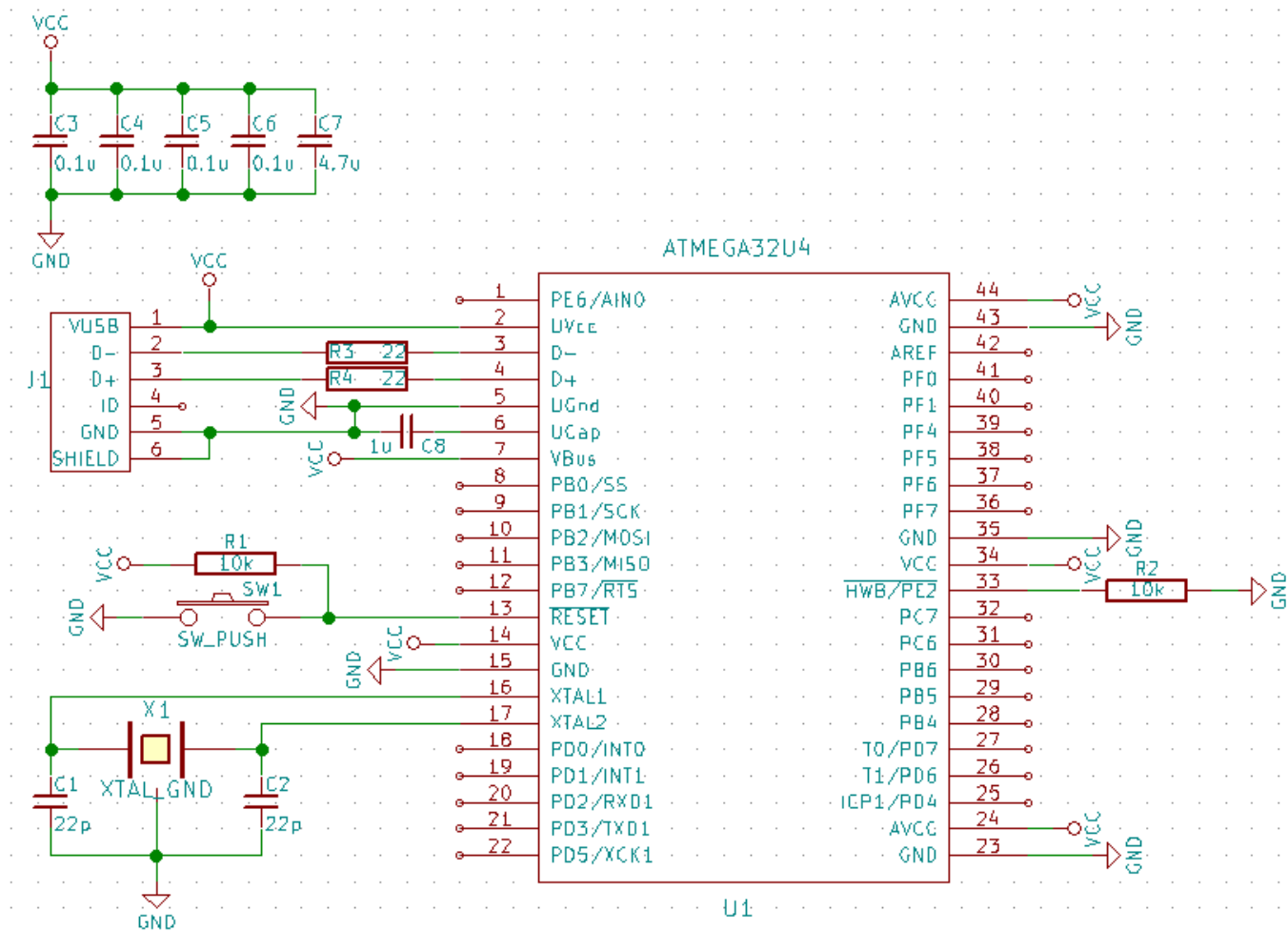
Now let's put a 10k resistor named R2 on HWB/PE2 pin and connect it to ground. We want a resistor here because it tells the microcontroller that when we press the reset button, we want to go into the bootloader so that we can flash a new layout onto it!



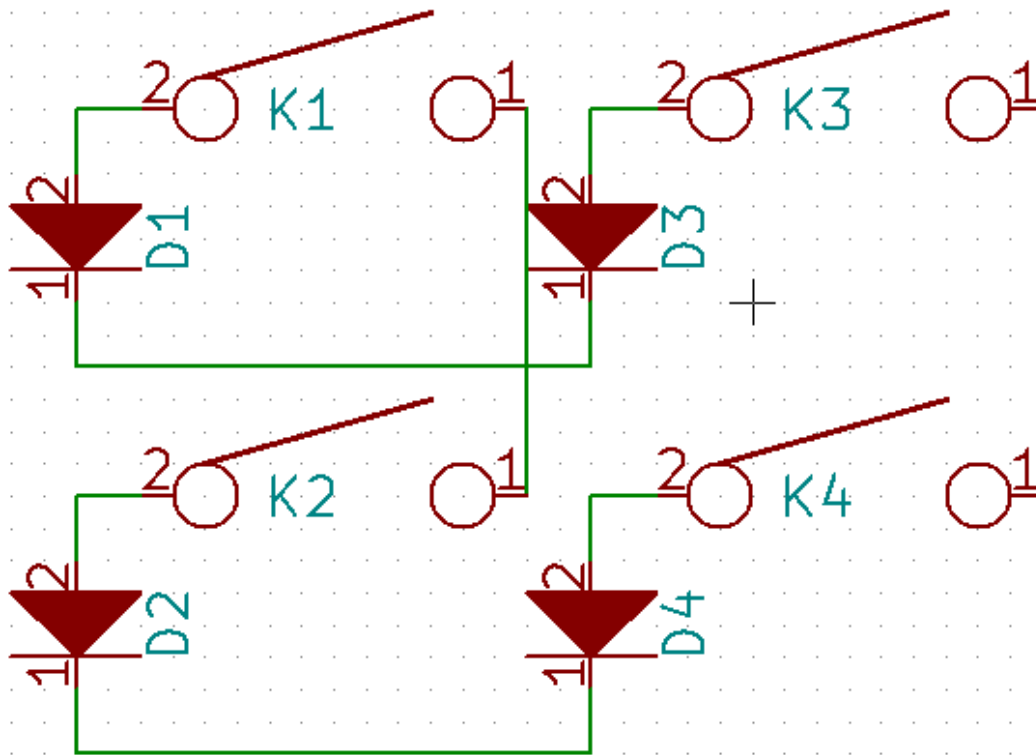
Next, let's add our USB port. Add the USB_mini_micro_B component from the keyboard_parts library and call it J1. Connect VUSB to VCC and Uvcc, and put two 22 ohm resistors R3 and R4 between the D- and D+ connections. Connect GND and SHIELD together and connect them to ground. And lastly, put a 1uF capacitor C8 between UCap and GND:



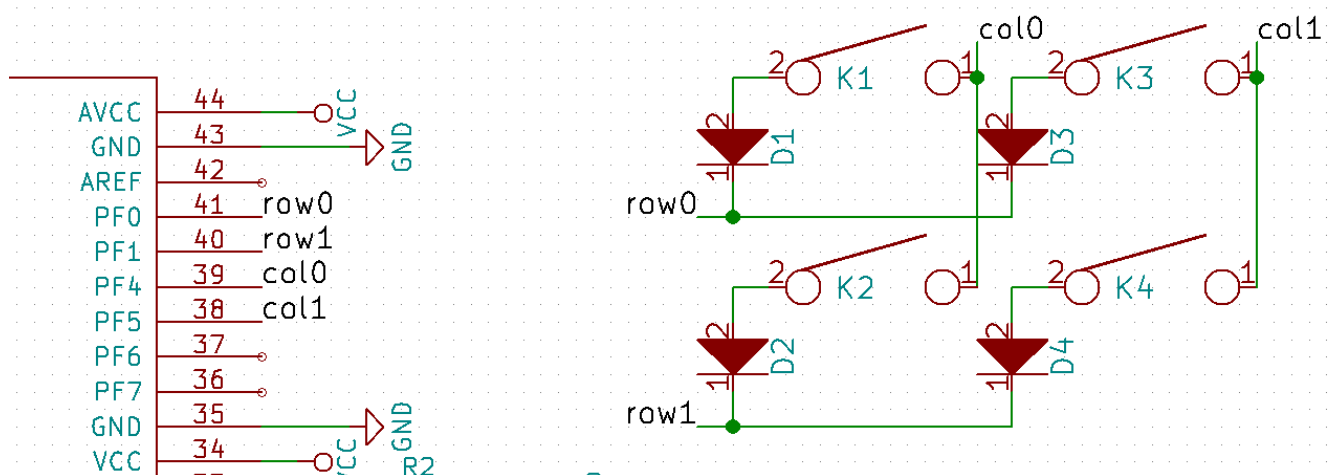
Let's connect all the VCC connections together and all the GND connections together. Normally, you would place a capacitor between AVCC and VCC if you were using the built-in ADC (analog to digital converter), but we don't care about that for a keyboard, so just directly connect them. Here's what everything look like at this point:



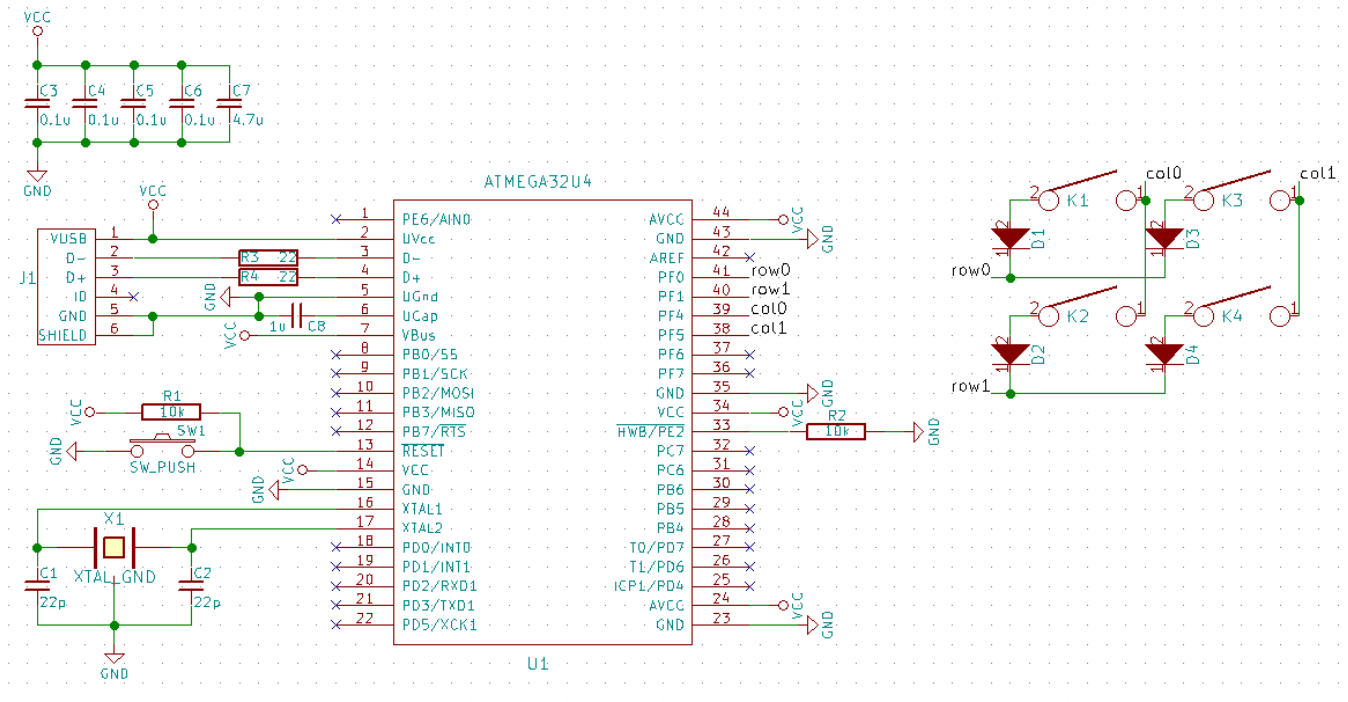
Now let's build our switch matrix. For the purposes of this guide, we're simply making a nice and easy 2x2 matrix. We're going to want to use the KEYSW and D components for our switch and diode components, respectively. Just connect them like you would a handwired board, and don't forget to name them. K1 should correspond to D1, K2 should correspond to D2, and so on:



Now we want to connect this matrix to the controller. We'll use labels for ease (A with a green line underneath on the right). For our example board, we'll use PF0 for row0, PF1 for row1, PF4 for col0, and PF5 for col1:



Finally, let's label all the unused pins as not connected. Use the no connect tool (blue X on the right) and click on all the unconnected pins on the controller and the ID pin on the USB port. This is also a good chance to make sure you didn't miss any VCC or GND pins earlier! Our final schematic should look like this:

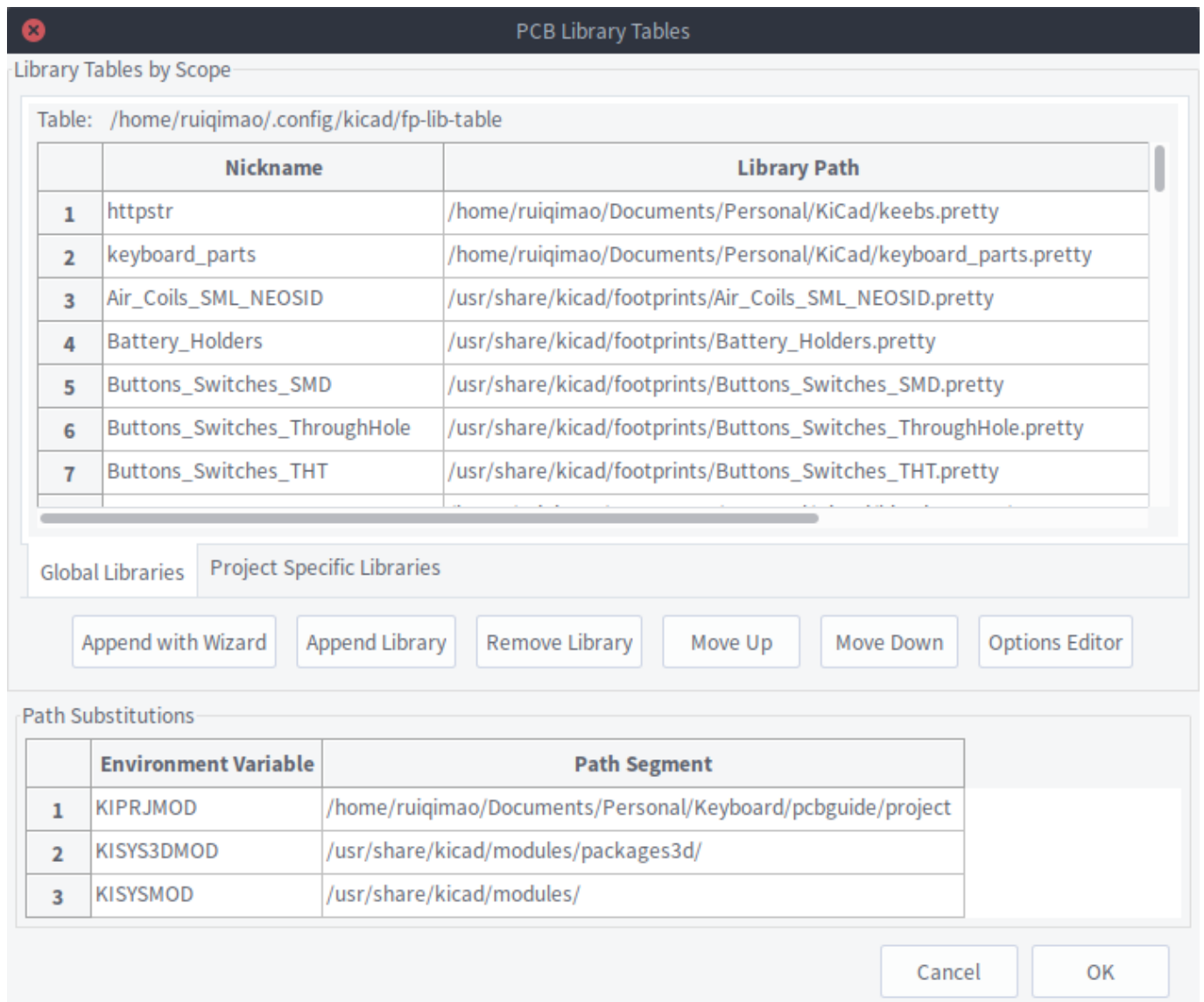


Associating Components and Footprints

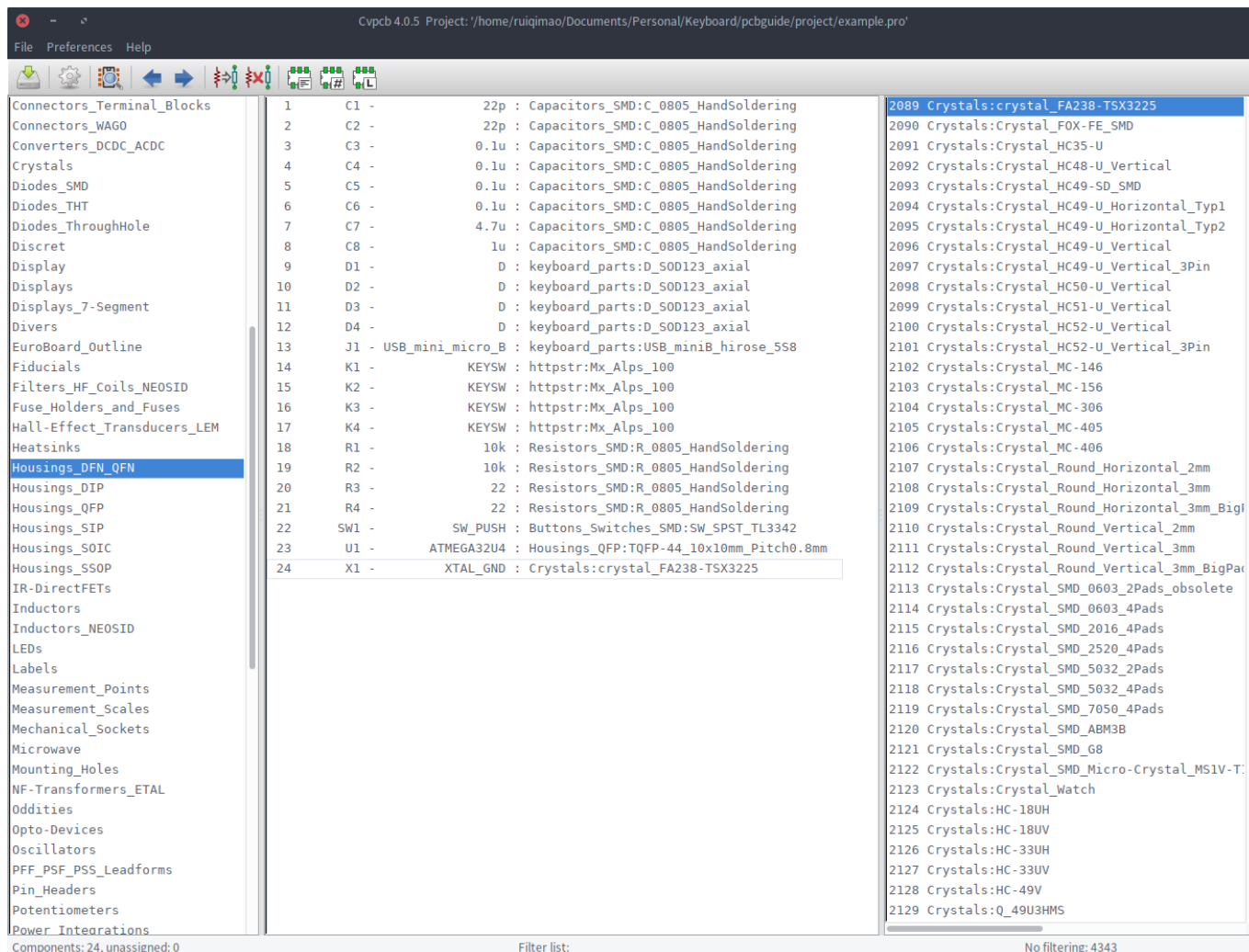
We have to tell KiCad what each of these components means. Click on the icon for CvPcb at the top:



If this is your first time running CvPcb, we're going to need to add the footprint libraries we downloaded earlier. Click Preferences > Footprint Libraries, and in the window that shows up, use the "Append with Wizard" button to add the "keeps.pretty" and "keyboard_parts.pretty" folders that we downloaded earlier. You may also need to manually add the built-in KiCad libraries. Your list of libraries should look something like this now:



We're going to assume that all of our capacitors and resistors are 0805 imperial size. Our ATmega32U4 is going to be in a TQFN package. We're going to use the very handy hybrid through-hole and surface mount footprint from Hasu's library for our diodes. Switches are going to be /u/techieeee's 1u switch footprint. The crystal will be an FA-238 series crystal. The reset button will be a TL3442 series button, and the USB mini B port will be a Hirose 5S8 connector. For each component, go through the list of footprints and double click on one to associate it with the currently selected component. Here's what all the associations should end up looking like:



Save the associations and close the window.

Generating Netlist

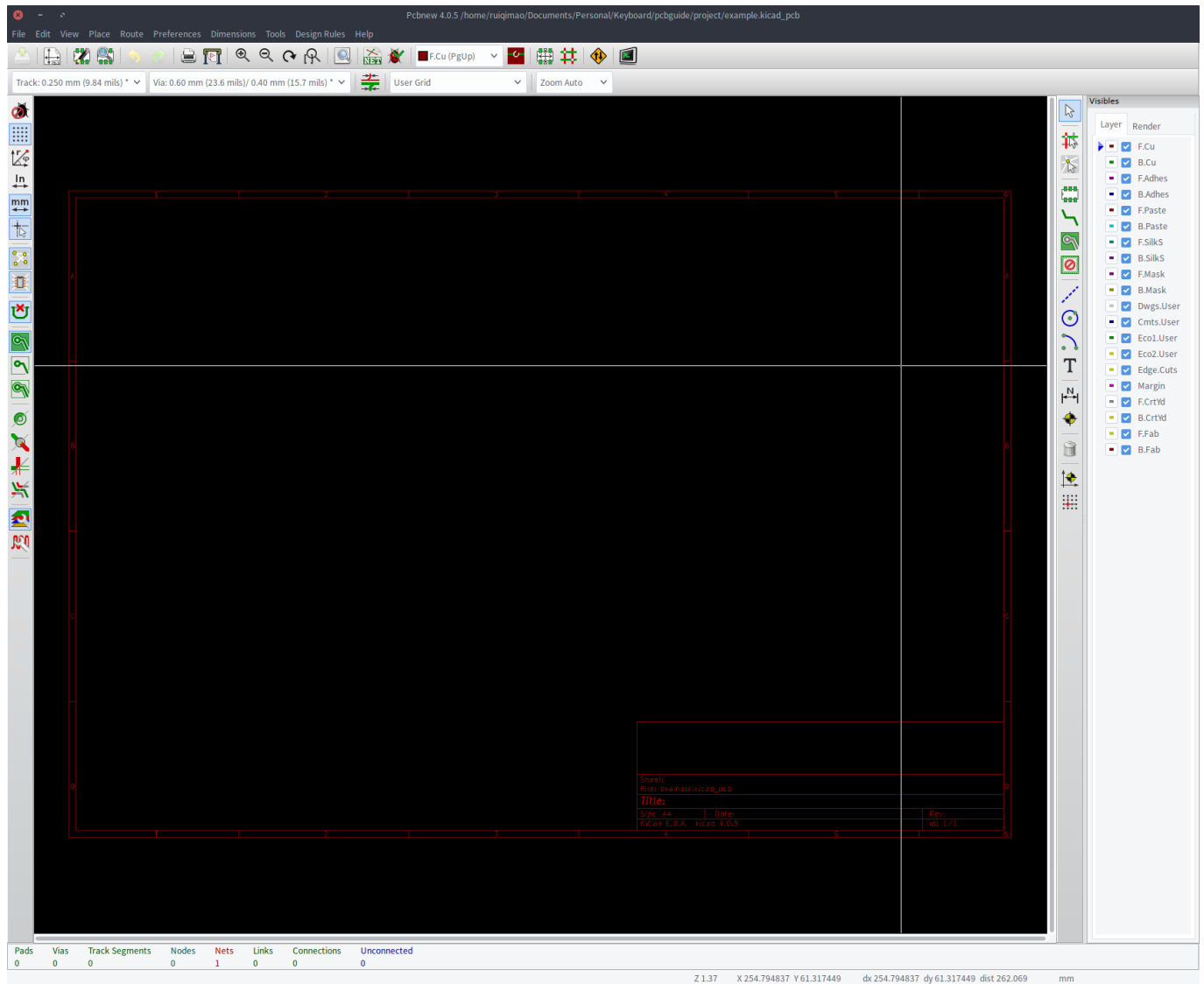
Now we want to generate the netlist, which is essentially a list of connections in our schematic. Click on the netlist button:



In the dialog that opens, simply click "Generate". Use the default netlist name in the save dialog. If everything was laid out and named properly, KiCad should not ask you about annotations. If so, click "Cancel" and double check all of your references then try again.

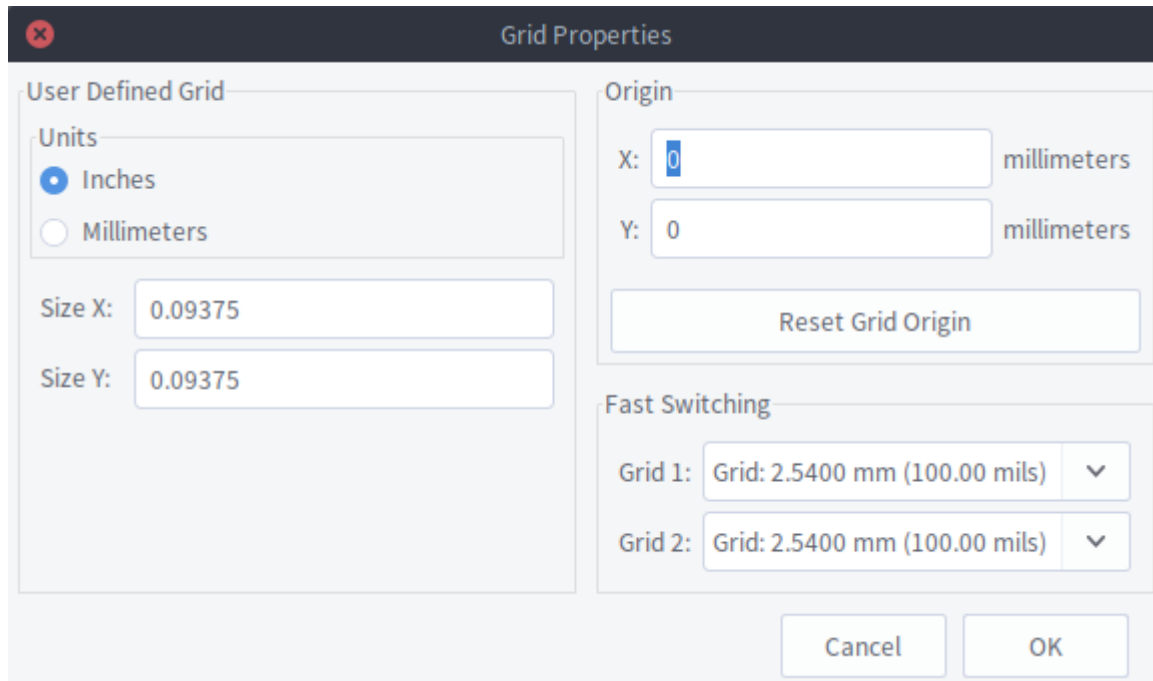
PCB

Now we get to create our PCB! Save and close out of the schematic editor. Then, go back to your project and open your ".kicad_pcb" file. You should be greeted by a blank PCB editor:



The first thing we're going to do is double check that all of our footprints are still here. Go to Preferences > Footprint Libraries Manager and make sure that all the footprint libraries you imported earlier are still there. If not, then simply import them again.

Next, we're going to set our grid. Click on Dimensions > Grid, and set Units to Inches and Size X and Size Y both to 0.09375, like so:



The image shows a 'Grid Properties' dialog box with a dark title bar. It is divided into two main sections. The left section, titled 'User Defined Grid', contains a 'Units' group with 'Inches' selected and 'Millimeters' unselected. Below this are input fields for 'Size X' and 'Size Y', both set to '0.09375'. The right section, titled 'Origin', has input fields for 'X' and 'Y', both set to '0', with 'millimeters' indicated to the right of each. A 'Reset Grid Origin' button is located below these fields. At the bottom of the right section is a 'Fast Switching' group containing two dropdown menus, 'Grid 1' and 'Grid 2', both showing 'Grid: 2.5400 mm (100.00 mils)'. At the very bottom of the dialog are 'Cancel' and 'OK' buttons.

User Defined Grid	
Units	
<input checked="" type="radio"/>	Inches
<input type="radio"/>	Millimeters
Size X:	0.09375
Size Y:	0.09375

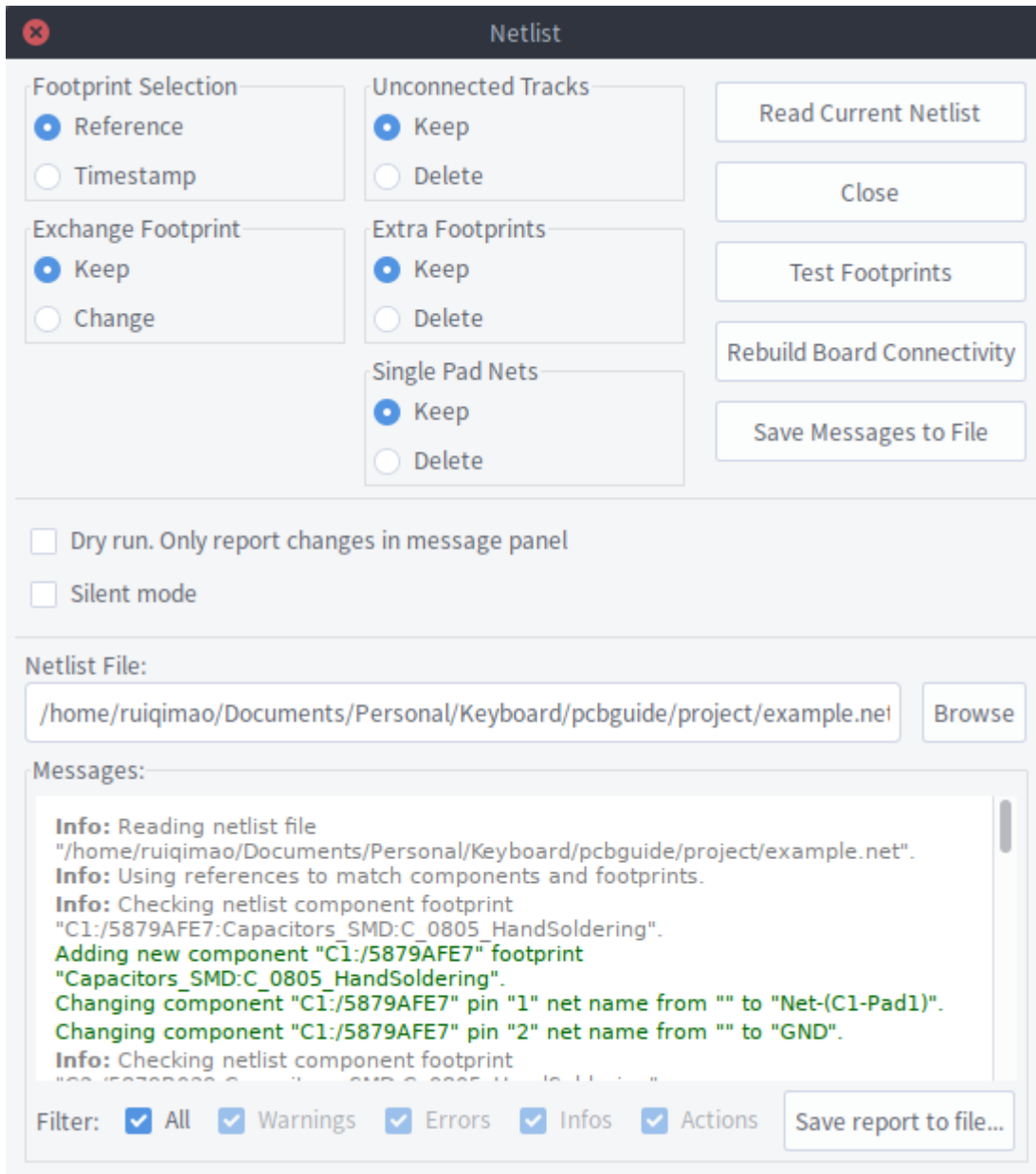
Origin	
X:	0 millimeters
Y:	0 millimeters
Reset Grid Origin	

Fast Switching	
Grid 1:	Grid: 2.5400 mm (100.00 mils) ▼
Grid 2:	Grid: 2.5400 mm (100.00 mils) ▼

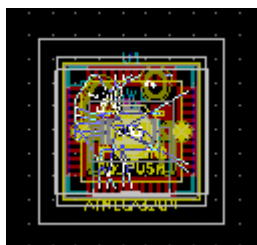
Cancel OK

Then, we want to tell the PCB editor to use our user-defined grid. Change the grid option at the top to "User Grid".

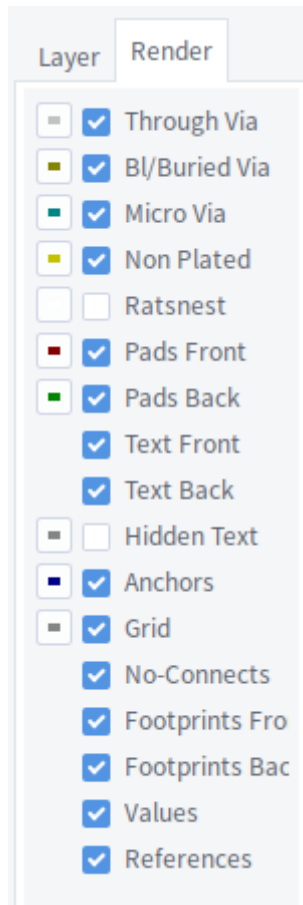
The easiest way to get all our footprints onto the board is to read the netlist we generated earlier. Click on the netlist button, which should look the same as before, and simply click on "Read Current Netlist". A bunch of messages should show up, and the dialog should look something like:



Now, click "Close". You'll notice that there are now a bunch of footprints in the middle of the screen all stacked on top of each other:



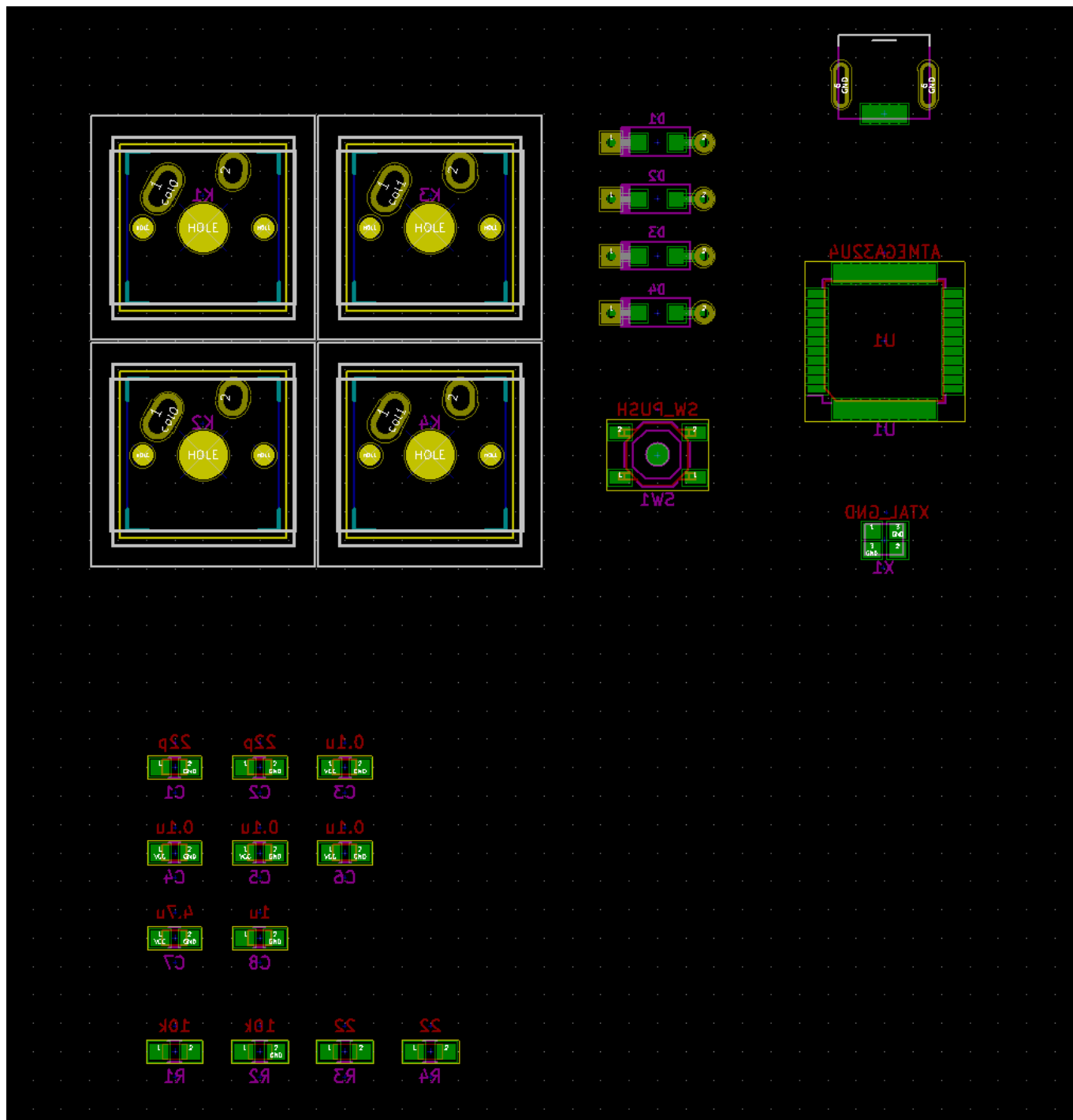
Before we separate them, let's hide the ratsnest, which is essentially the lines that detail the electrical connections in the board. Go to the "Render" tab on the right and uncheck "Ratsnest", like so:



Here are some useful commands for the PCB editor:

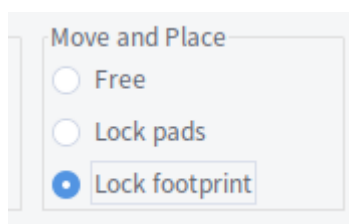
```
m: move the footprint
g: drag the footprint while keeping connectivity
e: edit the footprint
r: rotate the footprint
f: flip the footprint
del: delete the footprint
esc: abort!
```

Let's separate our footprints and put them on the correct side of the PCB. The only footprints that will be on the "front" of the PCB will be the switch footprints. Everything else will be on the "back" of the PCB, so make sure everything but the switches are flipped:



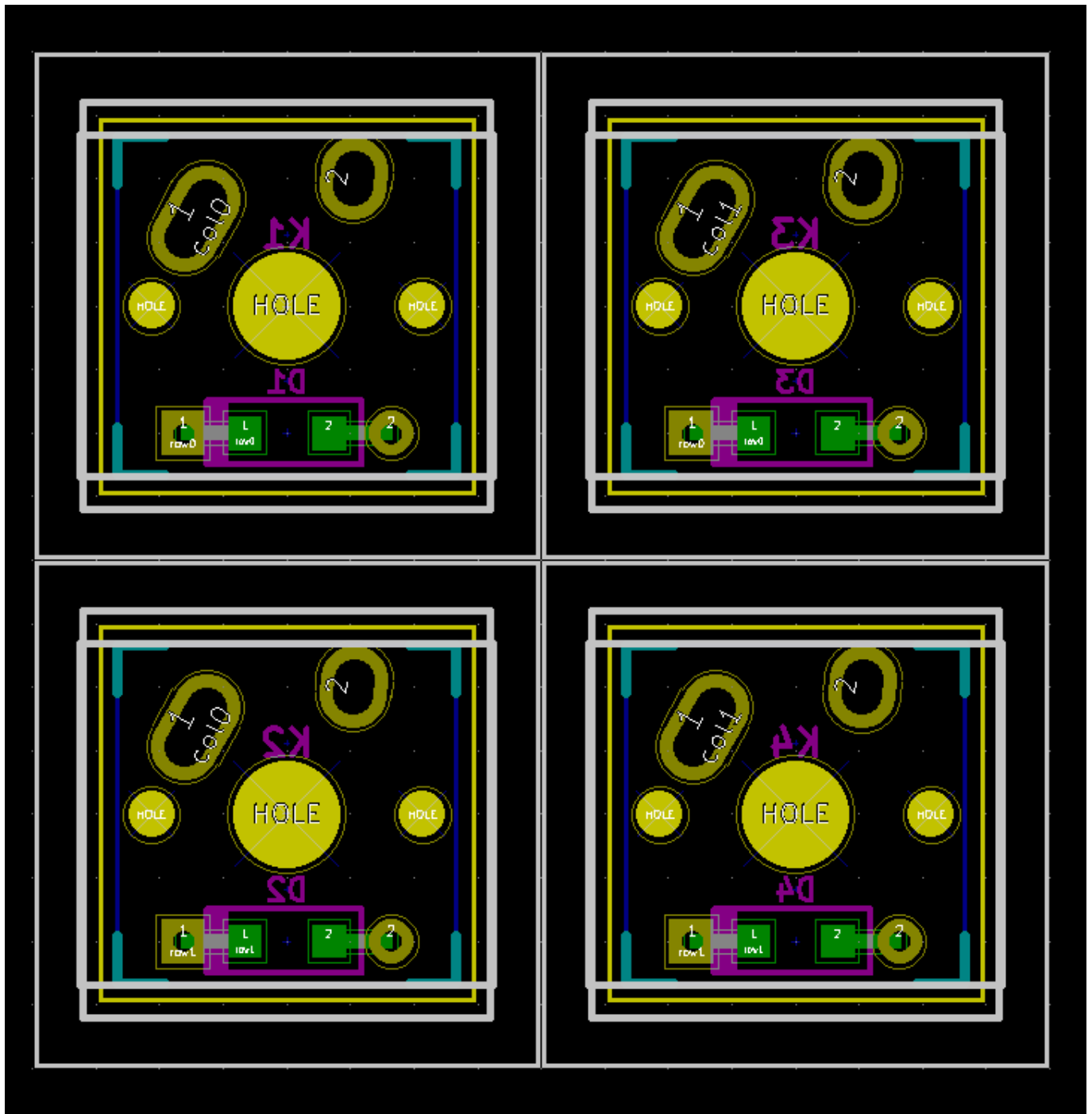
Component Placement

Arrange your switch footprints as shown if you haven't already. Then, edit each of the switch footprints and change the "Move and Place" option to "Lock footprint" so that we don't accidentally move them:

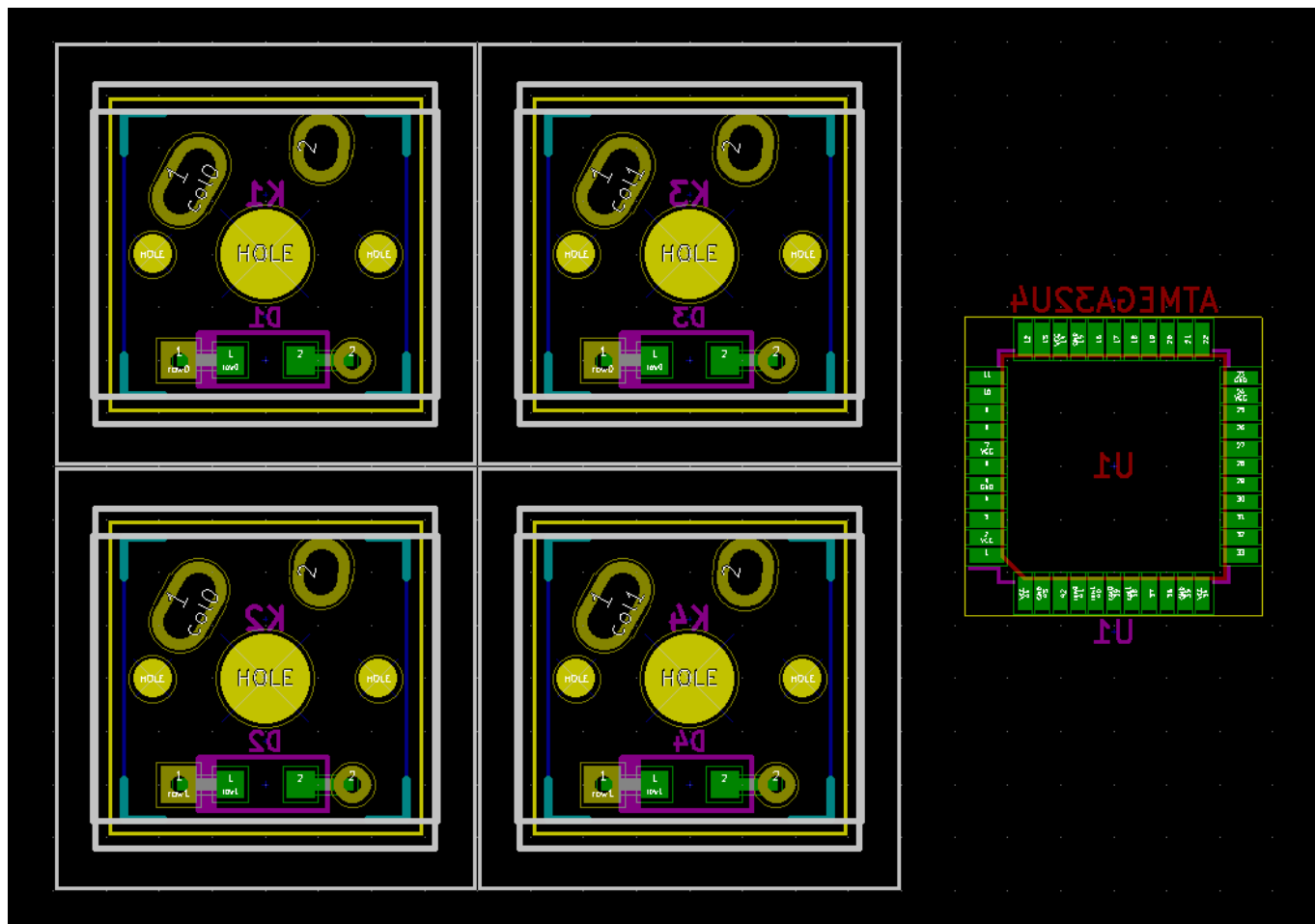


Let's put our diodes under our switches first (**THIS PLACEMENT WILL ONLY WORK FOR SMD**

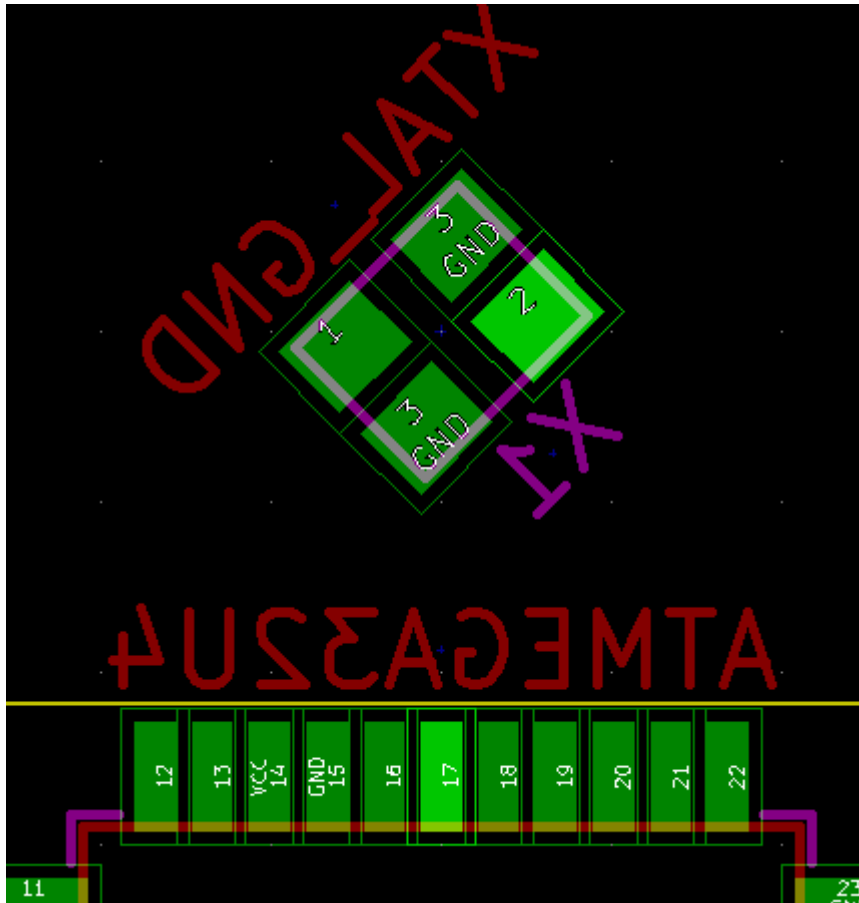
DIODES). Make sure each diode corresponds to its switch:



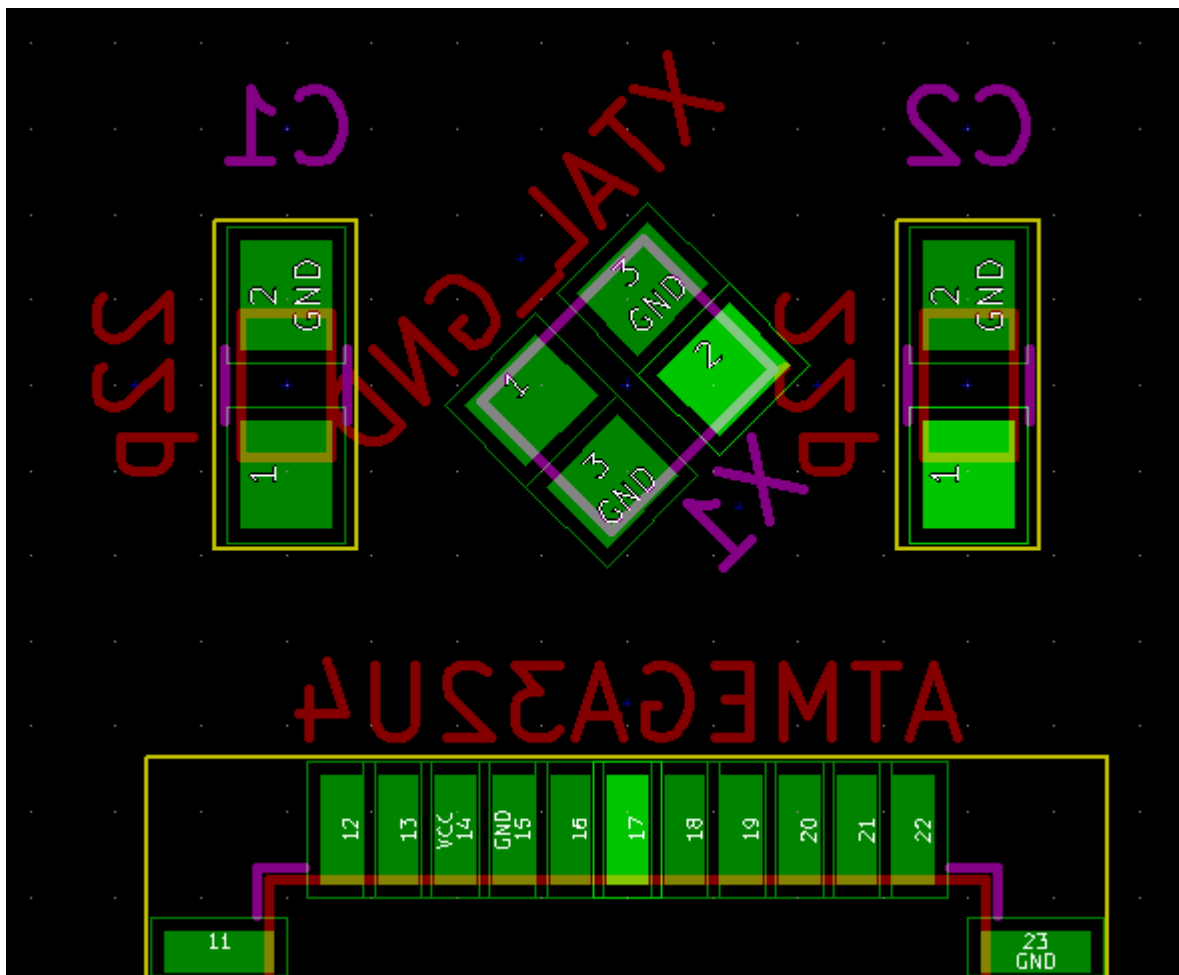
Let's move our microcontroller next to the switches, like so:



Now the most important part of PCB design: the crystal. We need to make sure the traces to the crystal are as short as possible and that they are roughly the same length. An easy way to tell what pads are supposed to connect to what pads is to use the "highlight net" tool on the right. You use the tool and simply click on a pad, and it and the pads it connects to are highlighted. For this example, I put the crystal above the microcontroller and rotated it by 45 degrees:

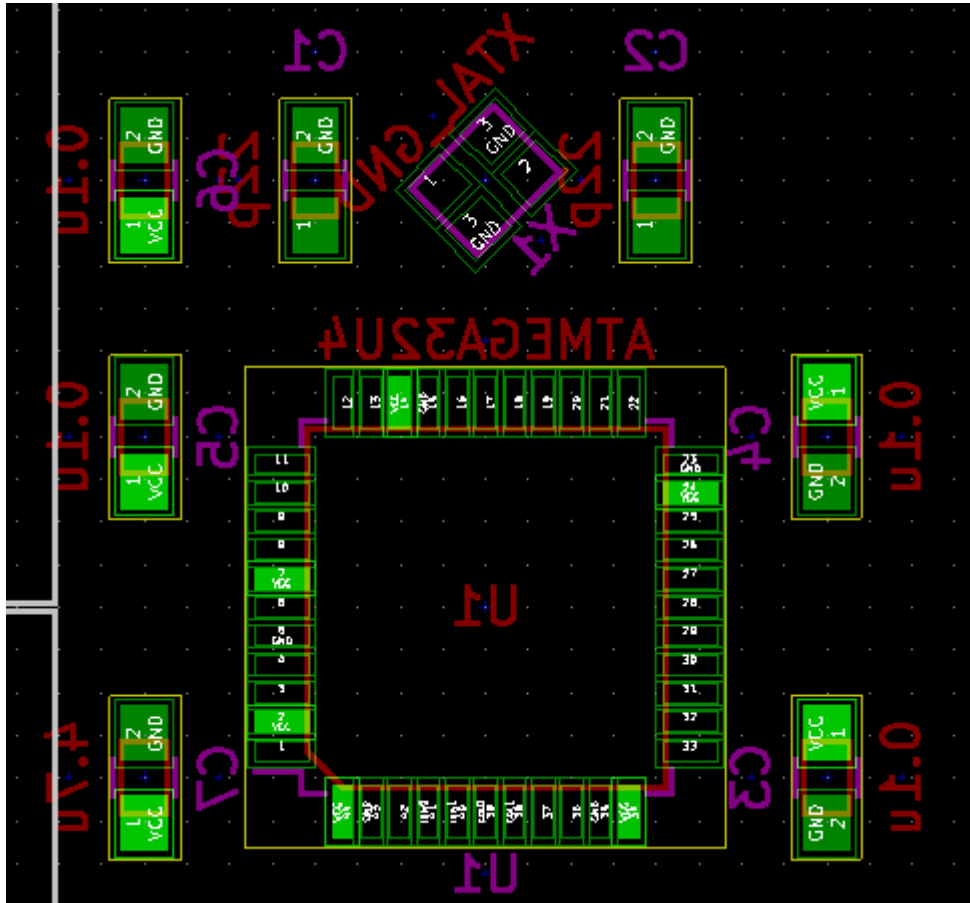


Then place the 2 decoupling capacitors next to their respective pads:



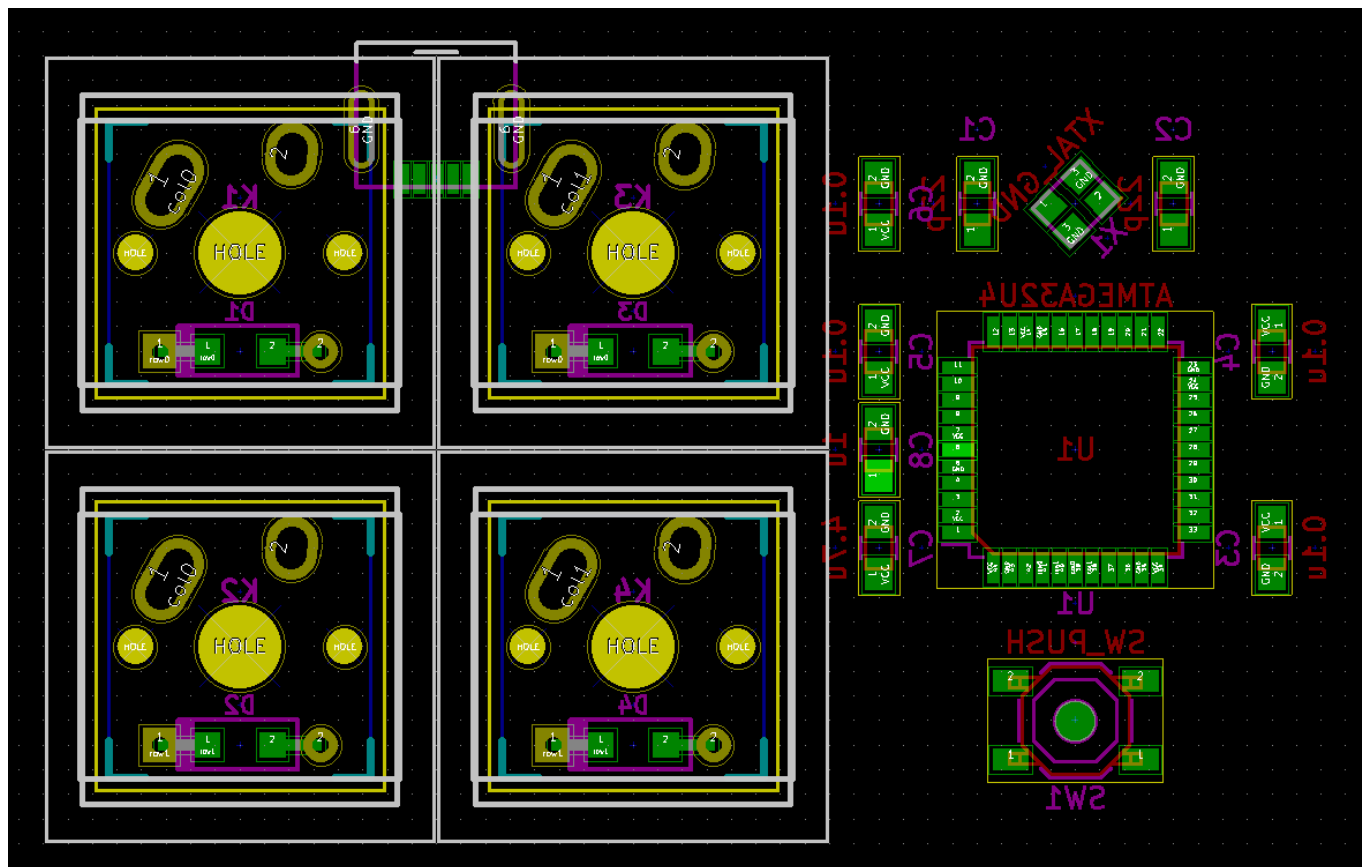
You don't need to worry about connecting ground to any other grounds, since we will be putting a ground plane under all the components. More on that later.

Next, we want to place the decoupling capacitors for VCC. Place a 0.1uF capacitor next to each VCC and AVCC and the 4.7uF capacitor next to UVCC. At this point, I discovered that the microcontroller was a little too close to the switches, so I move it out a little bit:

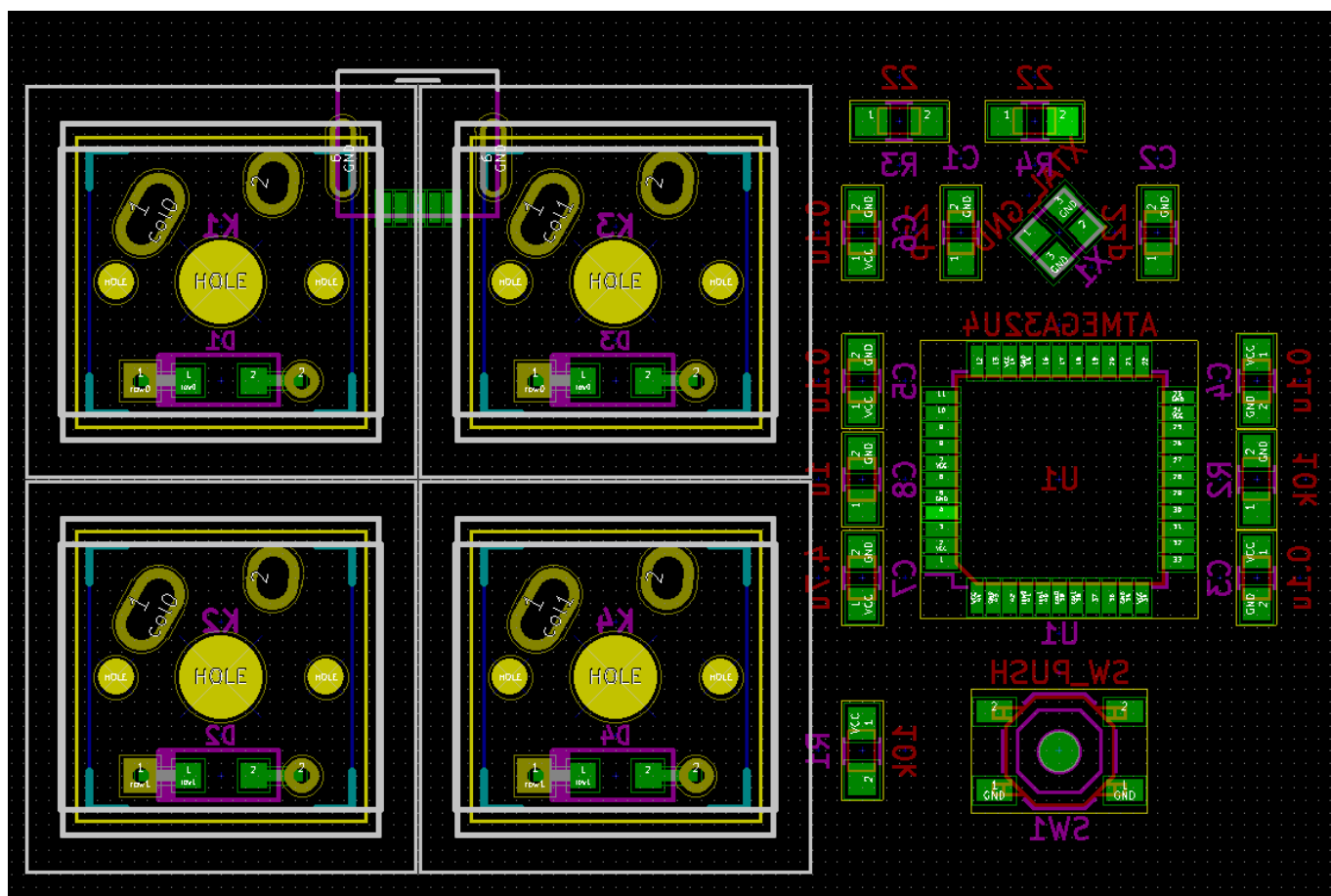


Let's put our last capacitor on, which will fit nicely between C5 and C7:





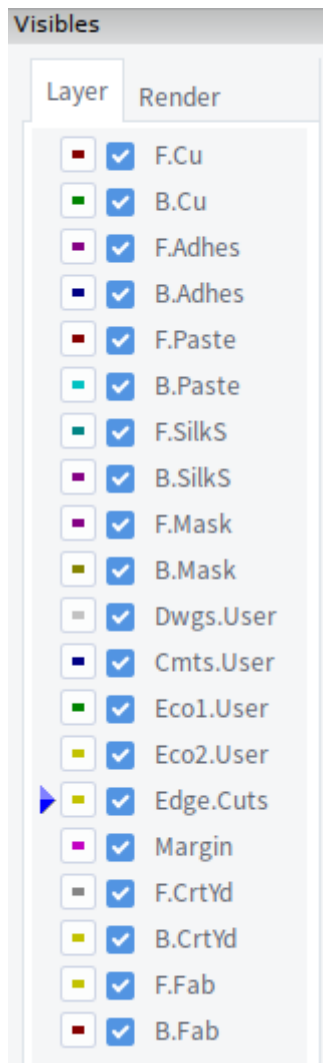
And our last components, the resistors. Place them in a way such that routing traces later will be easier. Here's how I did mine:



Note that at this point, my grid sizes are at 0.0234375" in each dimension to allow for finer positioning.

Edge Cuts

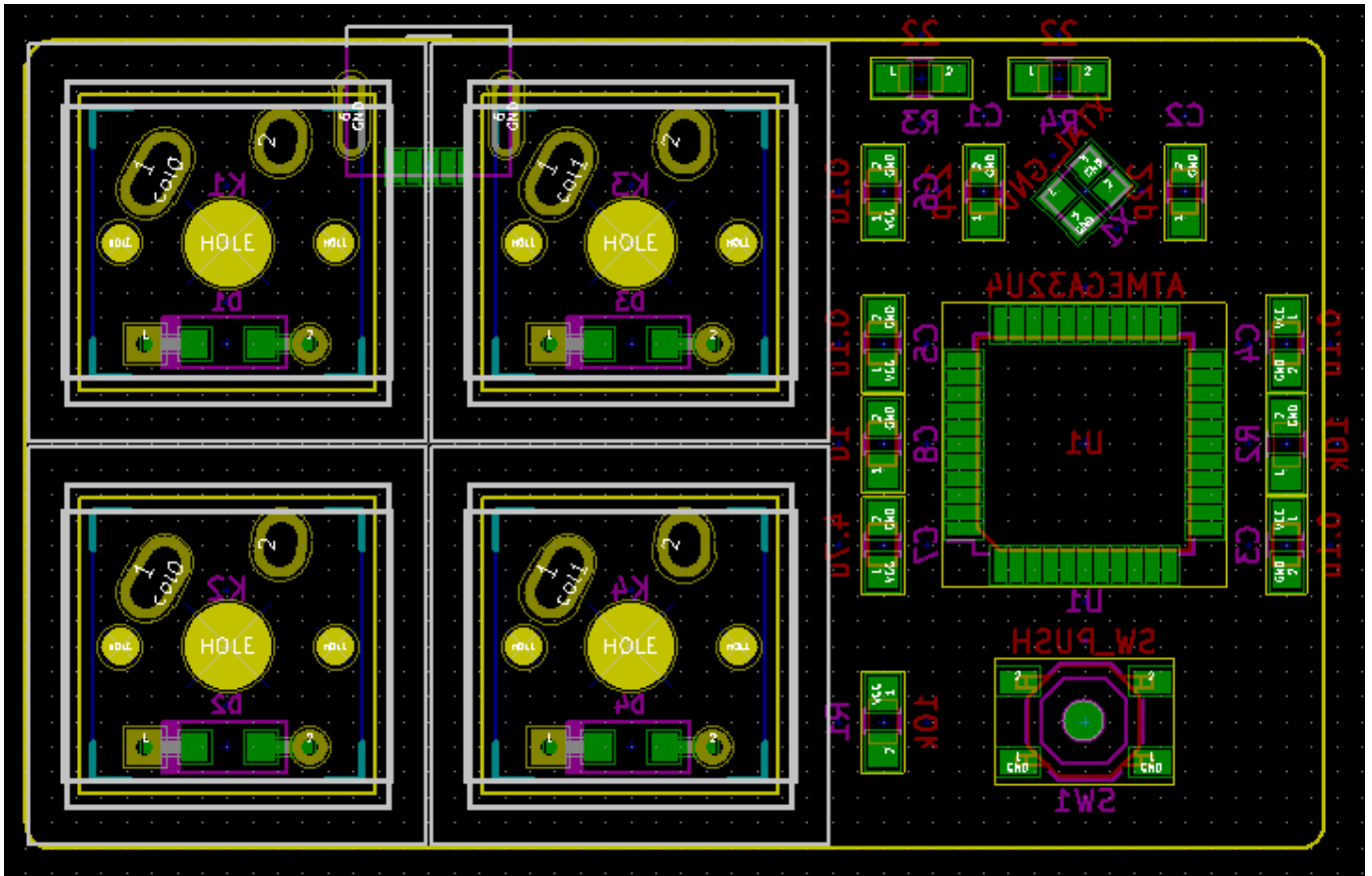
Now let's draw out the outline for our board! Go to the "Layer" tab on the right and click next to "Edge.Cuts" to move the blue arrow down to it, effectively selecting it as the layer we're going to draw on:



Use the drawing tools on the right to draw an outline for the PCB:



Here's how I cut mine:



Ground Plane

We want to put a ground plane in the PCB. Essentially, a ground plane is just one big chunk of copper that's connected to ground on both sides of the PCB. It's useful when we have a lot of components that are connected to ground, like in our PCB. To do this, we want to use the zone tool:



Make sure the blue arrow is back on F.Cu in the Layers tab. Then, select the zone tool and click on one of the corners of the edge cuts. A dialog will pop up asking you which net the zone should be associated with. Select GND and hit "OK":

×

Copper Zone Properties

Layer:

F.Cu

B.Cu

Net:

<no net>

GND

VCC

/row0

/row1

Net-(D1-Pad2)

Net-(D2-Pad2)

Net-(D3-Pad2)

Net Filtering

Display:

Show all (pad count) ▾

Hidden net filter:

Net-*

Visible net filter:

*

Apply Filters

Settings

Clearance (mm):

0.508

Pad connection:

Thermal relief ▾

Priority level:

0 ▴ ▾

Outline slope:

Arbitrary ▾

Minimum width (mm):

0.254

Thermal Reliefs

Antipad clearance (mm):

0.508

Fill mode:

Polygon ▾

Outline style:

Hatched ▾

Corner smoothing:

None ▾

Spoke width (mm):

0.508

Segments / 360 deg:

16 ▾

Chamfer distance (mm):

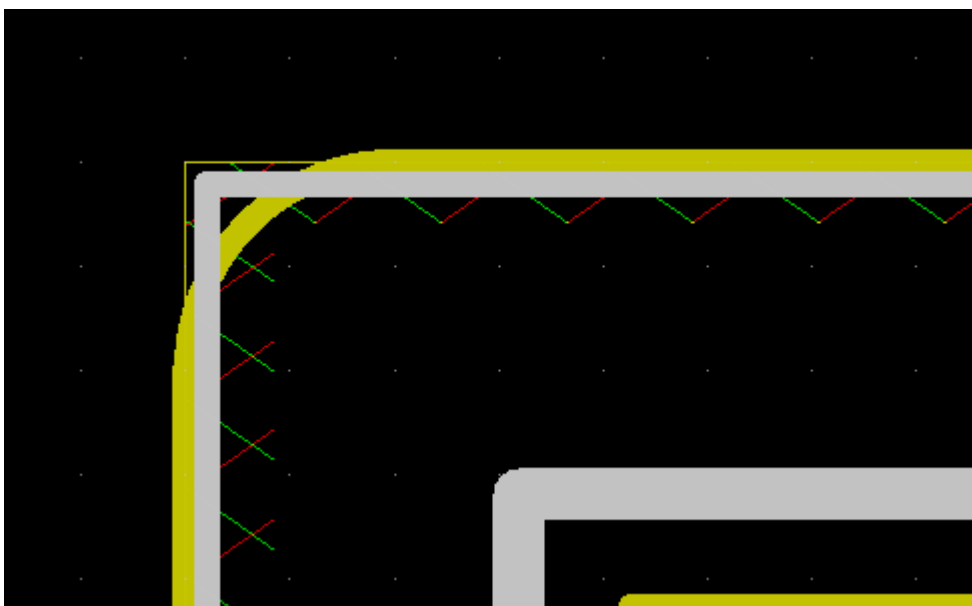
0

Export Settings to Other Zones

Cancel

OK

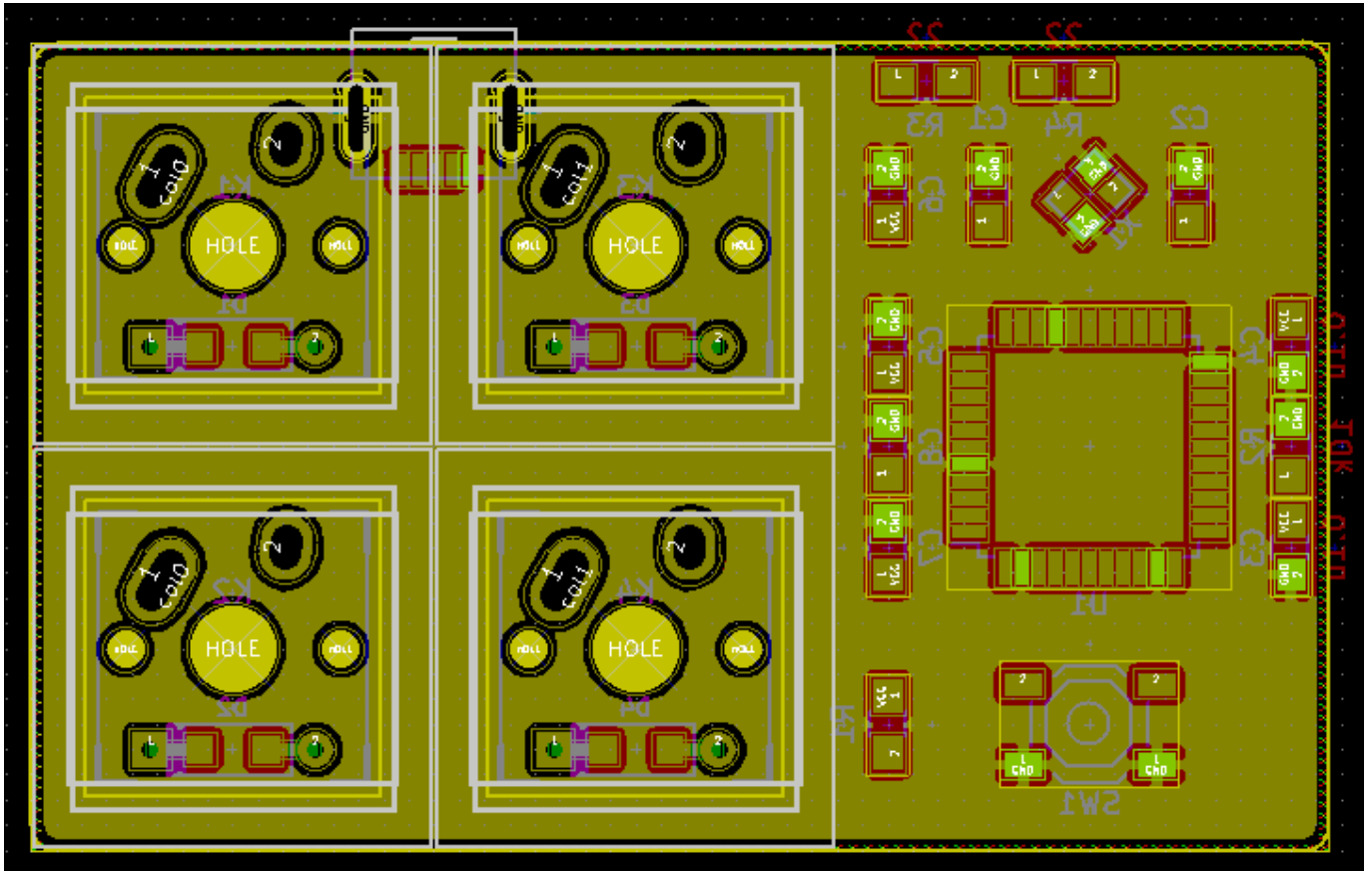
Now, draw a border where you put the edge cuts. When you get back to the starting point, double click. You will see a red hatch pattern around your PCB. Now right click on the edge of the zone and select Zones > Duplicate Zone onto Layer. The same dialog will pop up, but this time, select B.Cu on the left and hit "OK". Now your PCB will have both a red and green hatch pattern around it:



Now right click and select each zone, and for each zone, select Zone > Fill Zone. Make sure the option to show filled zones is selected on the left:



And your PCB should now look something like this:



Now choose the option to hide filled zones. We don't want them while routing.

Routing

Now we want to do some routing. To route, we want to use the "add tracks and vias" tool:



I would also recommend setting your grid size to 0.25mm for this part.

Let's get some terminology out of the way first:

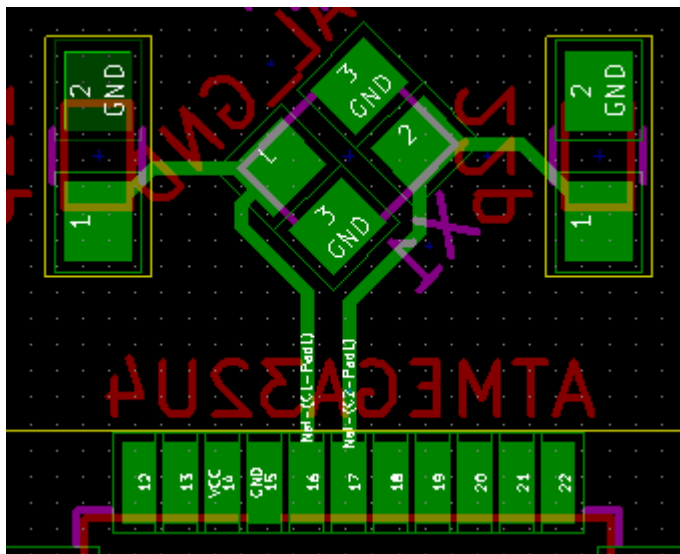
- A **trace** is a physical electrical copper connection. Like a wire.
- A **via** is a hole that goes through both sides of the PCB. This is useful because when routing

traces, we'll run into collisions where we can't route a trace through another. A via lets us hop to the other side of the PCB and continue the trace on the other side.

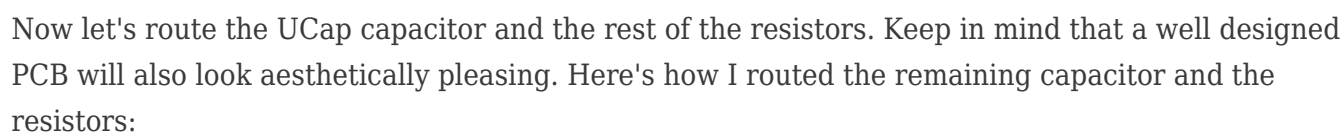
While routing, you can press v to switch layers and create a via.

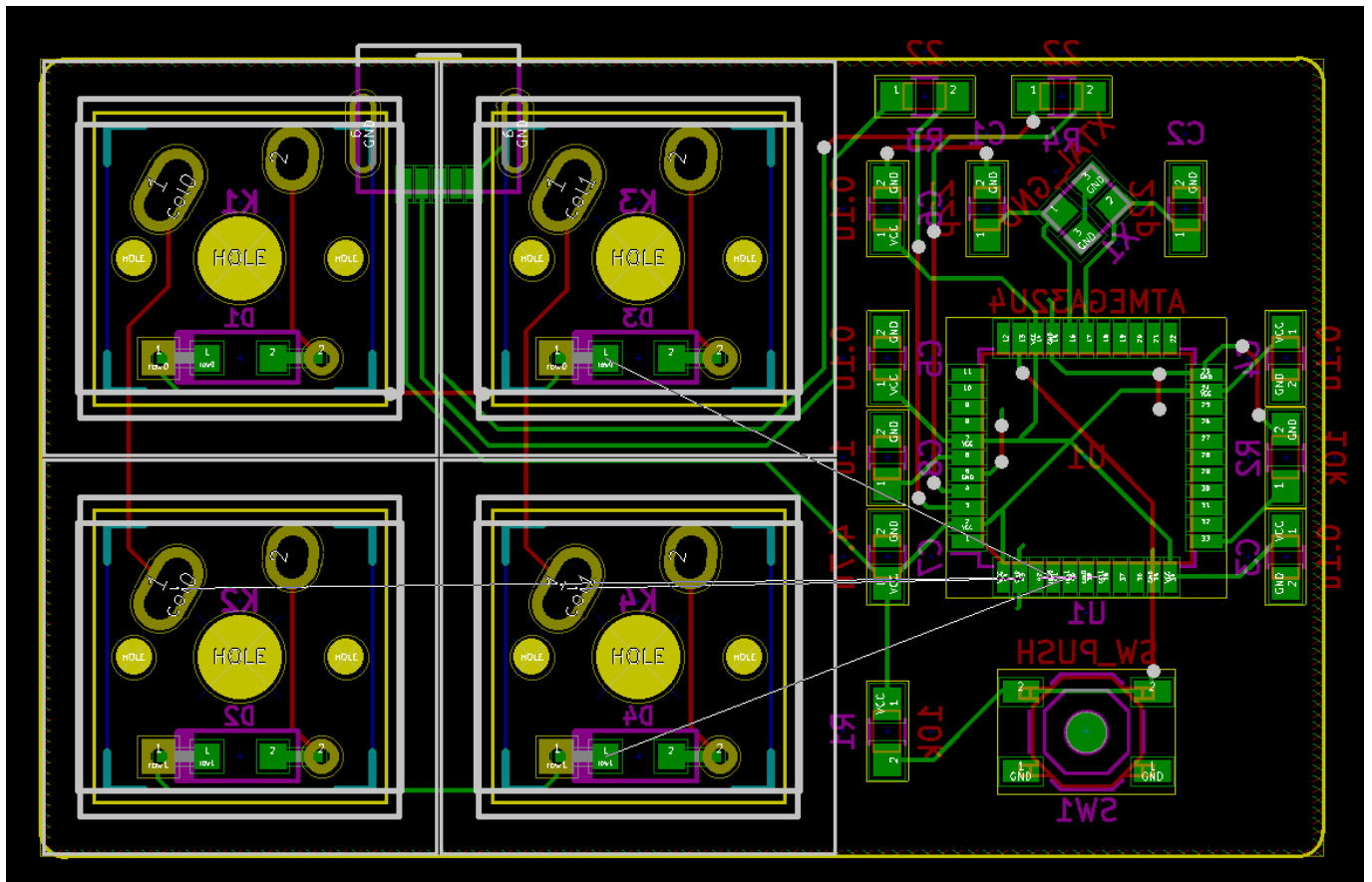
Now that we've gotten that out of the way, let's set some ground rules. Well, one ground rule. Namely, no vias between the crystal and the controller. Vias can potentially lead to a small amount of capacitance that can actually affect our crystal operation, so that's a big no no.

Now let's start routing. The first thing you want to route is always the crystal and the decoupling capacitors next to it. Luckily for us, this is pretty simple. To get started, make sure you have the B.Cu layer selected, since that's where most of our components are. Here's how I did mine:

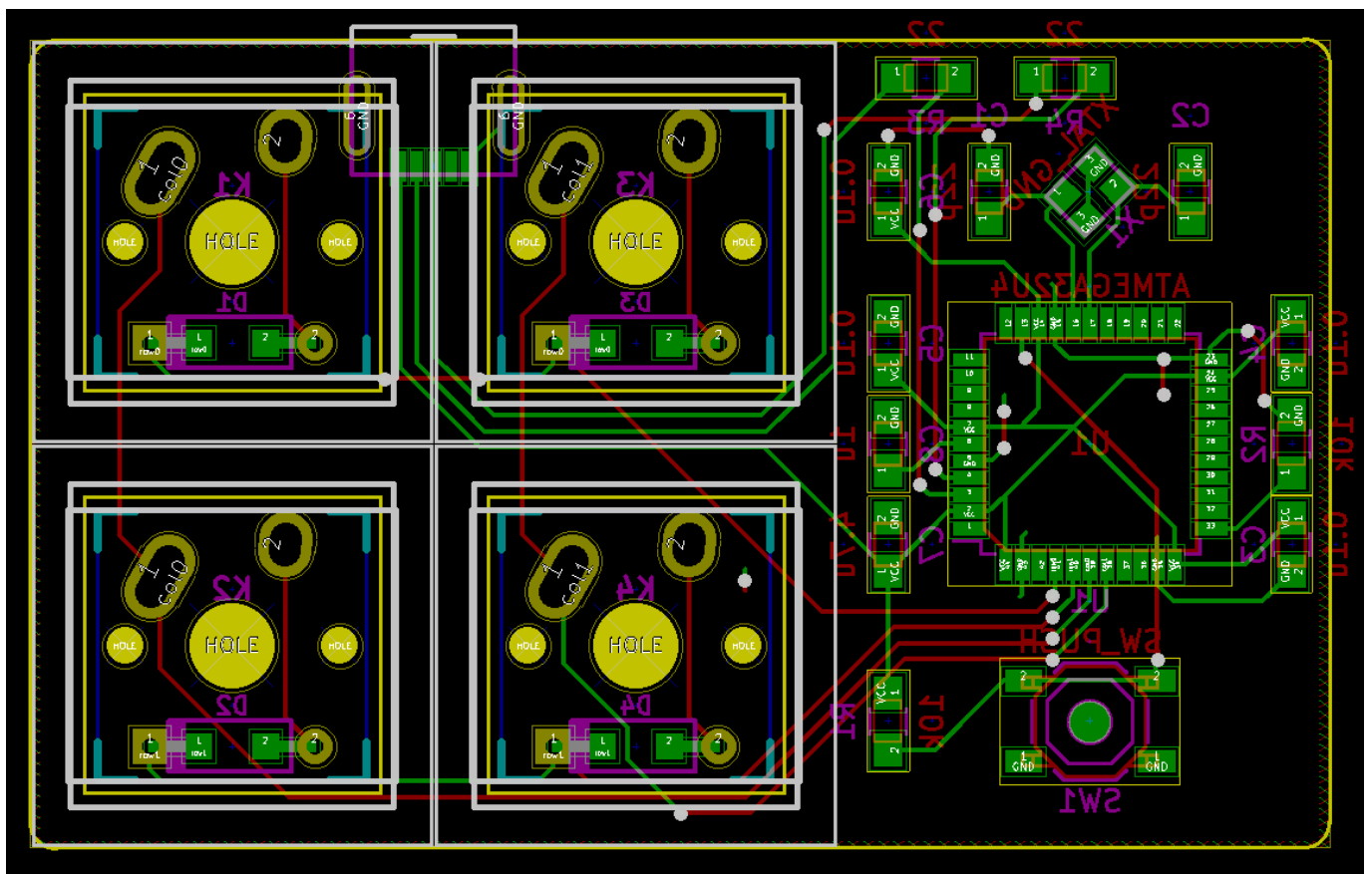


Next, let's route the VCC lines. Make sure you route the decoupling capacitors to their appropriate VCC pins on the microcontroller. And don't forget to route to the USB port!

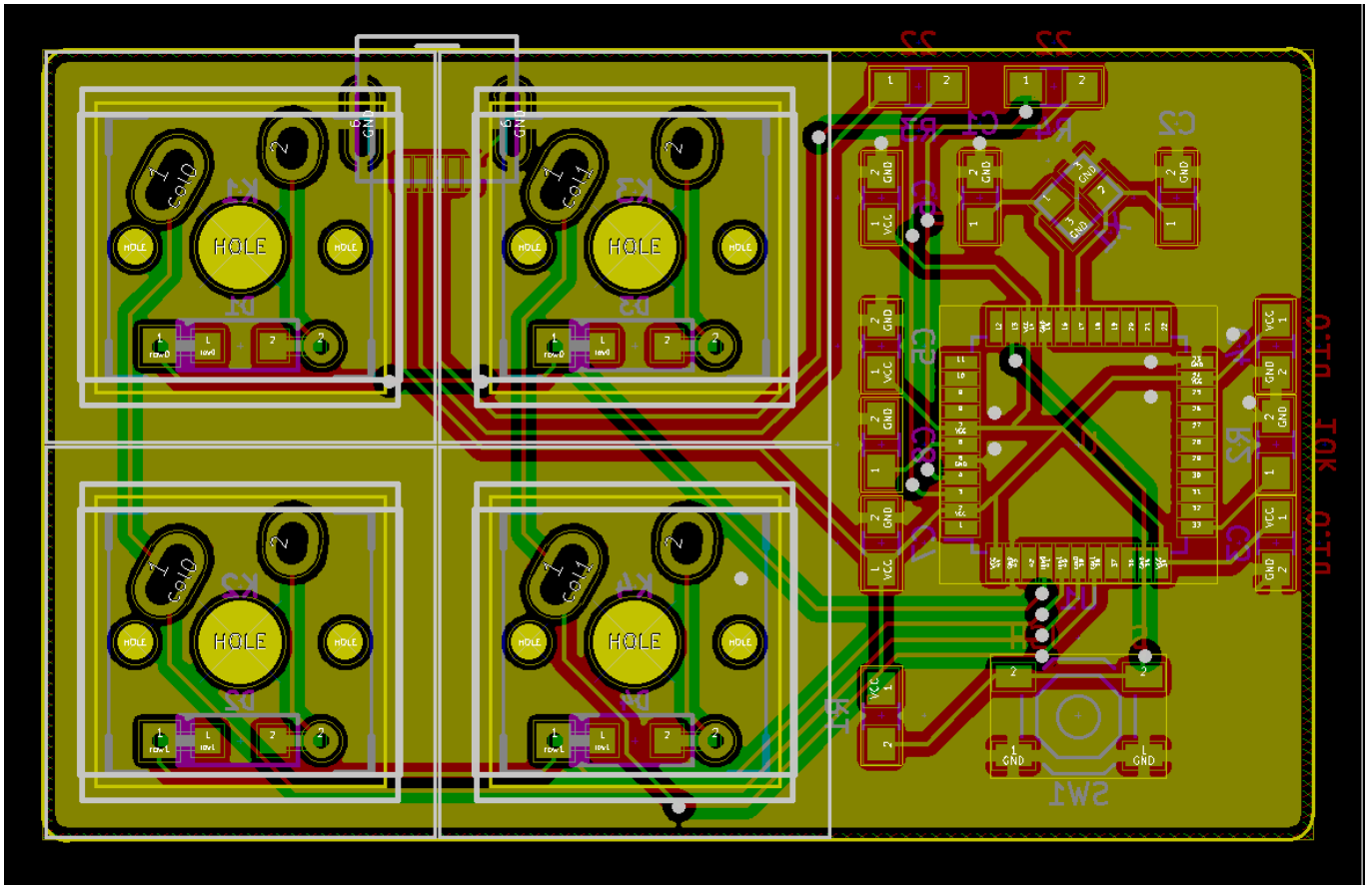




And finally, we'll route the controller to the rows and columns:



Turn on your zones, fill in any missing ground planes, press b to update the zones, and voila:



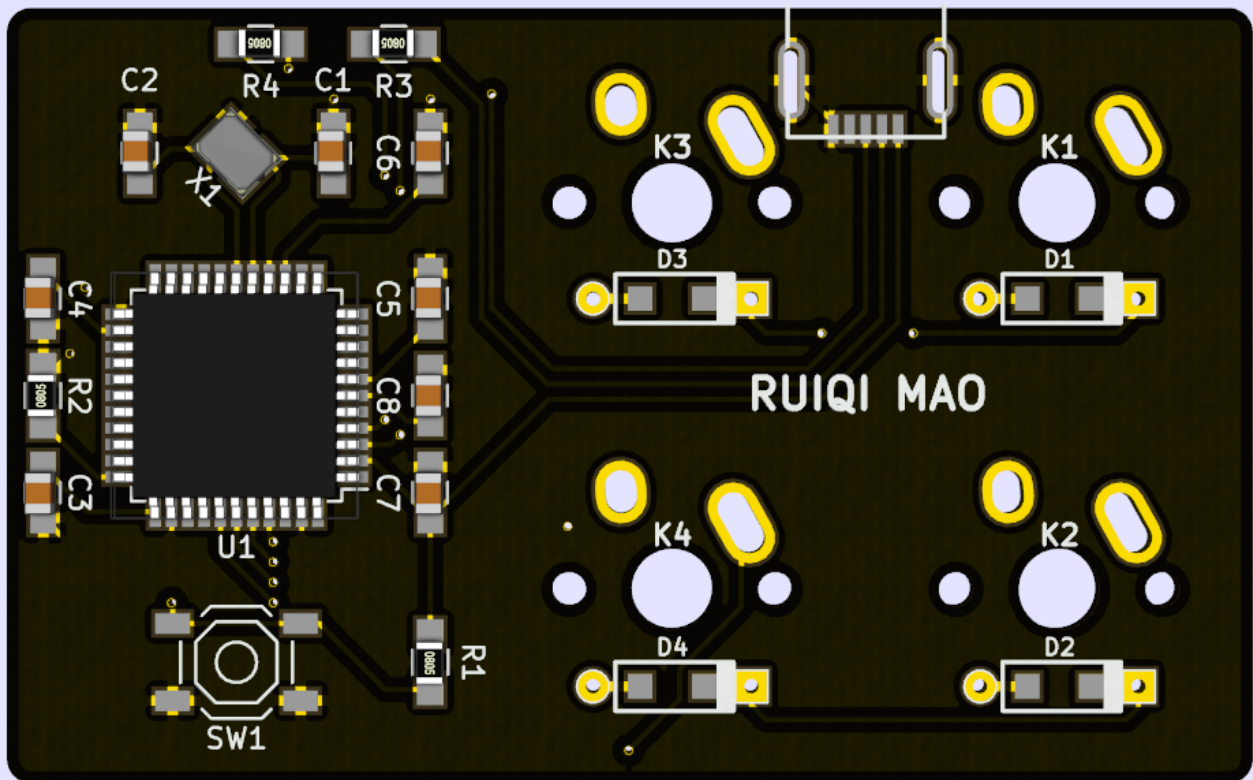
Double check that all connections have been made using the DRC (design rule check) tool:



Click "Start DRC" and make sure there are no problems. Then click "List Unconnected" and check for unconnected nets. If both sections are blank, then...

Your PCB is finished!

Go to View > 3D View and turn on Preference > Realistic Mode. Turn up all the settings you want, set your background color, and you'll have a nice render of your PCB! You can even select the Text tool and the B.Silks layer to put your name on your PCB:



Mounting Holes

Now, with a normal PCB, you would want to add some mounting holes. The way you would add those would be by creating custom footprints with NPTH (non-plated through hole) pads and adding them to your PCB. This is out of scope for this guide, but this is something that is very simple to Google!

Production

Now that you've finished designing your PCB, you want to get it made, right? Let's get you started on that.

Gerber Files

First thing is to generate our "gerbers", which are essentially files that tell the PCB manufacturer what is on each layer. Click on the "plot" icon:



Specify a directory to put your gerber files into and make sure the F.Cu, B.Cu, B.Paste, F.Paste, B.SilkS, F.SilkS, B.Mask, F.Mask, and Edge.Cuts layers are selected. Check the "Use Protel filename extensions" option, and make sure the format is "4.6 (unit mm)":

✕

Drill Files Generation

Output directory:

gerber/

Browse

Drill Units:

☐ Millimeters

☒ Inches

Zeros Format

☒ Decimal format

☐ Suppress leading zeros

☐ Suppress trailing zeros

☐ Keep zeros

Precision

2:4

Drill Map File Format:

☐ HPGL

☒ PostScript

☐ Gerber

☐ DXF

☐ SVG

☐ PDF

Drill File Options:

☐ Mirror y axis

☒ Minimal header

☐ Merge PTH and NPTH holes into one file

Drill Origin:

☒ Absolute

☐ Auxiliary axis

Info:

Default Vias Drill:

Use Netclasses values

Micro Vias Drill:

Use Netclasses values

Holes Count:

Plated Pads: 18

Not Plated Pads: 12

Through Vias: 24

Micro Vias: 0

Buried Vias: 0

Drill File

Map File

Report File

Close

Messages:

Now close the dialogs.

Put all the files you just generated into a zip file:

gerber.zip				
Location: /				
Name	Size	Type	Modified	
example.drl	948 bytes	unknown	14 January 2017, 01:06	
example-B.Cu.gbl	95.9 kB	unknown	14 January 2017, 01:04	
example-B.Mask.gbs	4.1 kB	unknown	14 January 2017, 01:04	
example-B.Paste.gbp	3.4 kB	unknown	14 January 2017, 01:04	
example-B.SilkS.gbo	20.4 kB	unknown	14 January 2017, 01:04	
example-Edge.Cuts.gm1	790 bytes	unknown	14 January 2017, 01:04	
example-F.Cu.gtl	42.6 kB	unknown	14 January 2017, 01:04	
example-F.Mask.gts	1.1 kB	unknown	14 January 2017, 01:04	
example-F.Paste.gtp	271 bytes	unknown	14 January 2017, 01:04	
example-F.SilkS.gto	1.9 kB	unknown	14 January 2017, 01:04	
example-NPTH.drl	245 bytes	unknown	14 January 2017, 01:06	

Upload your files onto <http://www.gerber-viewer.com/> and make sure all the layers look good. If so, you're ready to send your PCB off to the manufacturer!

Manufacturer

Now, there are a lot of options here. I've personally used [PCBWay](#) (referral link) to great success, but there are plenty of other cheap PCB prototyping services, such as [EasyEDA](#), [OSH Park](#) and [DirtyPCBs](#).

All of these services simply involve choosing some options for how your PCBs will be manufactured (default settings are fine for all of them), then uploading the zip file you just created. If you want to change the color of your PCB, then the option you will want to look for is "solder mask color". The color of the text on your PCB will be "silkscreen color".

Once you receive your PCBs, you can simply use some solder paste and a hot air rework station to put everything together! If you're unsure of how to do this, there are plenty of resources online that can teach you the basics of SMD soldering.

Components

But where do you get the components from? I highly recommend [DigiKey](#) for components. Here's a

list of all the components used in this guide:

- **Microcontroller:**
<https://www.digikey.com/product-detail/en/microchip-technology/ATMEGA32U4-AUR/ATMEGA32U4-AURCT-ND/3440960>
- **Crystal:**
<https://www.digikey.com/product-detail/en/epson/FA-238-16.0000MB-C3/SER3686CT-ND/2403459>
- **22pF Capacitors:**
<https://www.digikey.com/product-detail/en/kemet/C0805C220J5GACTU/399-1113-1-ND/411388>
- **0.1uF Capacitors:**
<https://www.digikey.com/product-detail/en/yageo/CC0805ZRY5V9BB104/311-1361-1-ND/2103145>
- **1uF Capacitors:**
<https://www.digikey.com/product-detail/en/yageo/CC0805KKX7R7BB105/311-1365-1-ND/2103149>
- **4.7uF Capacitors:**
<https://www.digikey.com/product-detail/en/murata-electronics-north-america/GRM21BR61E475KA12L/490-3335-1-ND/702876>
- **22 Ohm Resistors:**
<https://www.digikey.com/product-detail/en/stackpole-electronics-inc/RMCF0805JT22R0/RMCF0805JT22R0CT-ND/1942533>
- **10k Ohm Resistors:**
<https://www.digikey.com/product-detail/en/stackpole-electronics-inc/RMCF0805JT10K0/RMCF0805JT10K0CT-ND/1942577>
- **Diodes:**
<https://www.digikey.com/product-detail/en/micro-commercial-co/1N4148W-TP/1N4148WTPM5CT-ND/717311>
- **Button:**
<https://www.digikey.com/product-detail/en/e-switch/TL3342F160QG-TR/EG2531CT-ND/379004>
- **Mini USB Connector:**
<https://www.digikey.com/product-detail/en/hirose-electric-co-ltd/UX60SC-MB-5S8/H11589CT-ND/1949225>

Other good places to get electronic components include [Mouser](#) and [element14](#).

Good luck and have fun building your own PCBs!