How to make a replacement keyboard PCB

This is intended to be a rough guide for cloning a keyboard PCB and adding USB functionality. It's main use is for older keyboards who have insufficient key rollover or protocol limitations. Protocol converters or controller replacements are a way easier option in most cases. I only recommend going this route if you can't easily use a converter or if find the rollover to be restrictive and you don't want to do a hand-wired matrix (or can't in the case of plate-free boards).
This guide will follow along making a PCB for an ADDS 1010 keyboard with linear space-invaders. It uses an alien protocol with a 2KRO PCB but has pretty nice switches and caps so it's the perfect candidate.

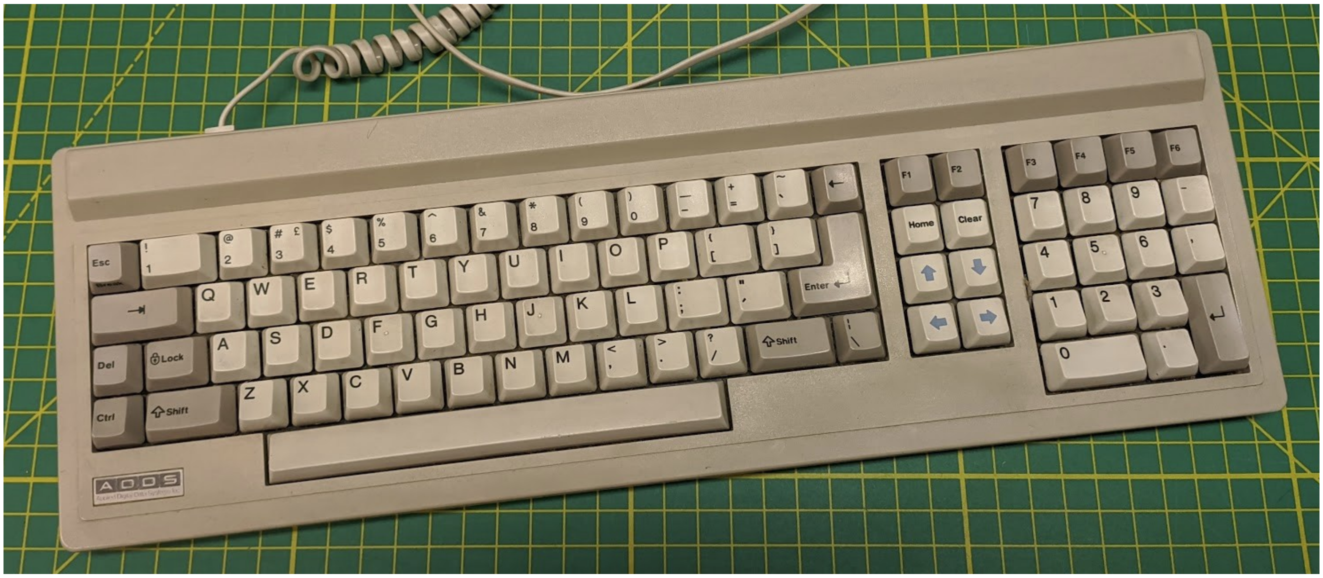
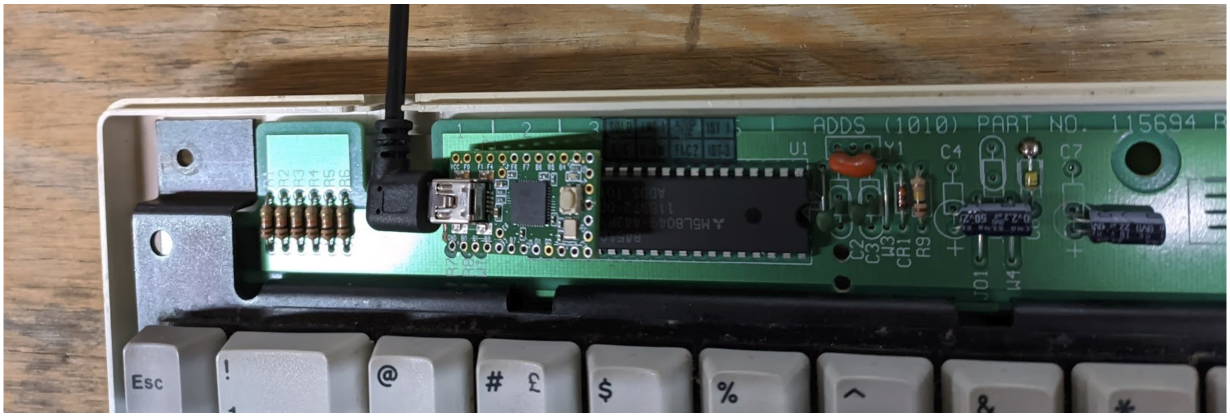


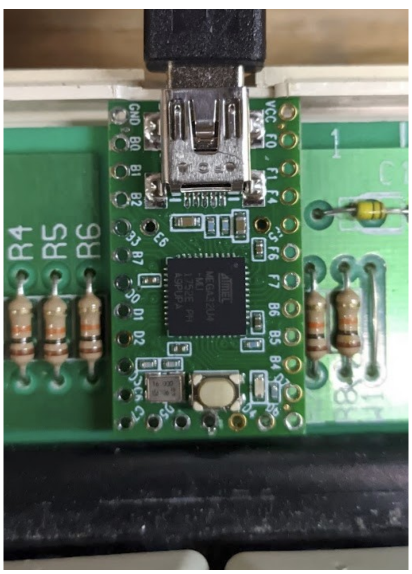
Table of Contents

Planning

Before things are too torn down, spend some time planning where you want to mount the controller and how you're going to make a cable or USB socket.

An option I've used on a few boards are short USB extensions. They're usually pretty easy to mount to the case and can run anywhere inside the keyboard to the controller. You can get whatever female end on them you prefer too.





Also take a look at the room under the switches; we'll need to find a good spot to put the diodes. In my case there's a bit of room between each switch.

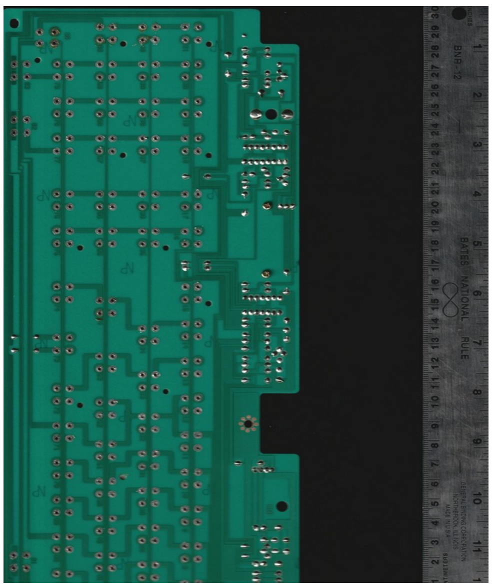
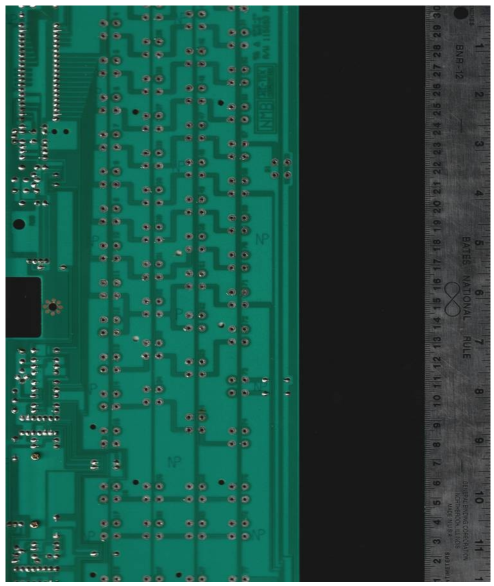
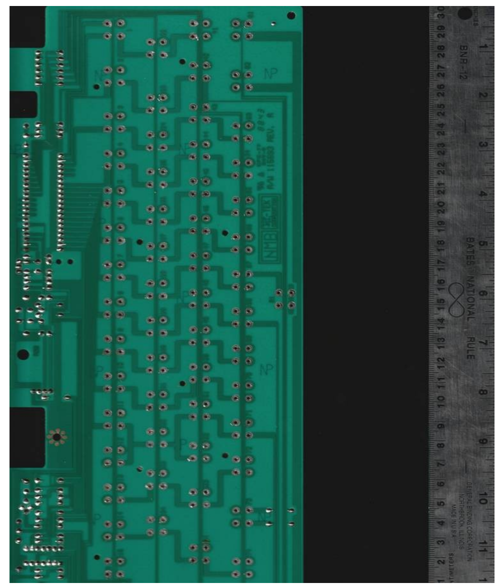
Disassembly

Tear the keyboard down to the PCB. A desoldering gun makes this much quicker. You'll need to be able to lay one side on a flatbed scanner so remove any protruding components and trim any legs relatively flush.

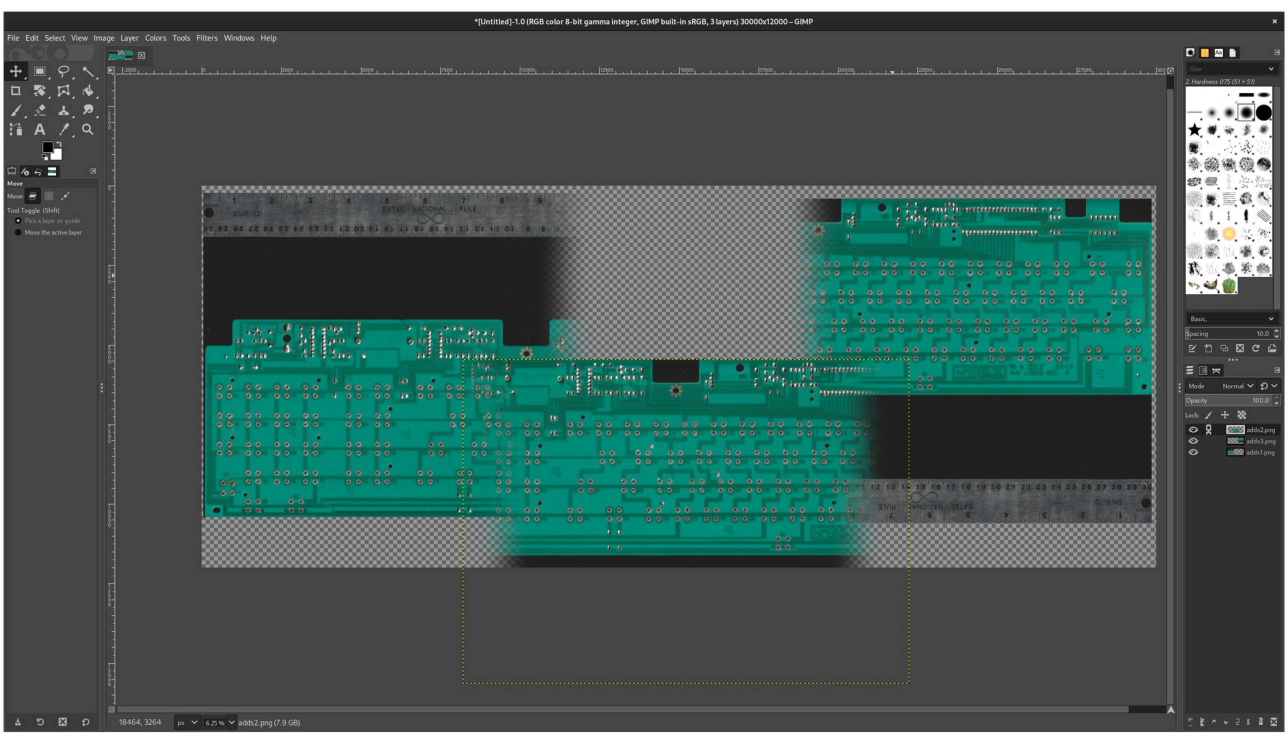
Scanning

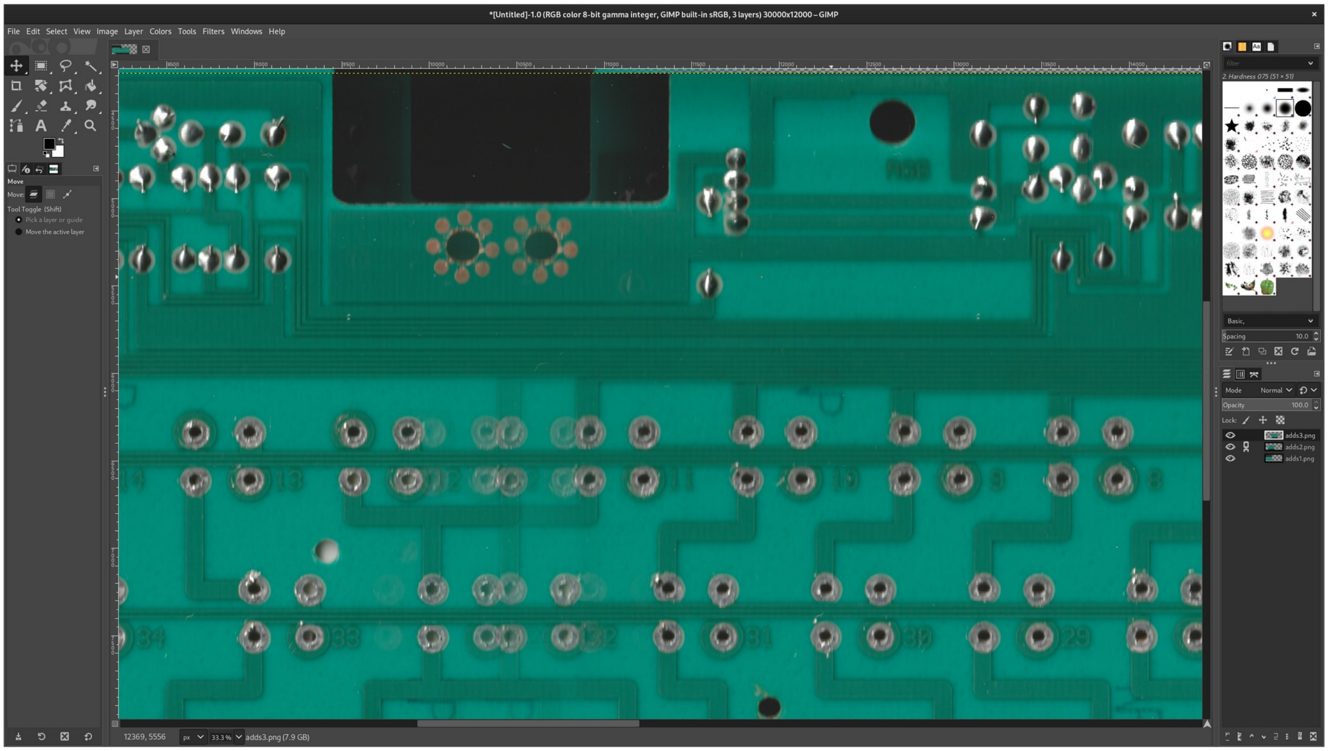
This whole process is a ton easier if you have a flatbed scanner. If you don't, you'll have to get creative with some calipers. This guide will assume you do (or can have it scanned at a print shop if those still exist). It's best if you can get it all in one shot but in my case only part of the PCB fits on the scanner at a time. Get some scans with a lot of overlap so you can be confident in the alignment.

Take a couple maximum resolution scans with a ruler face down next to the PCB so you can set the scaling correctly. These are the three scans that I ended up using:

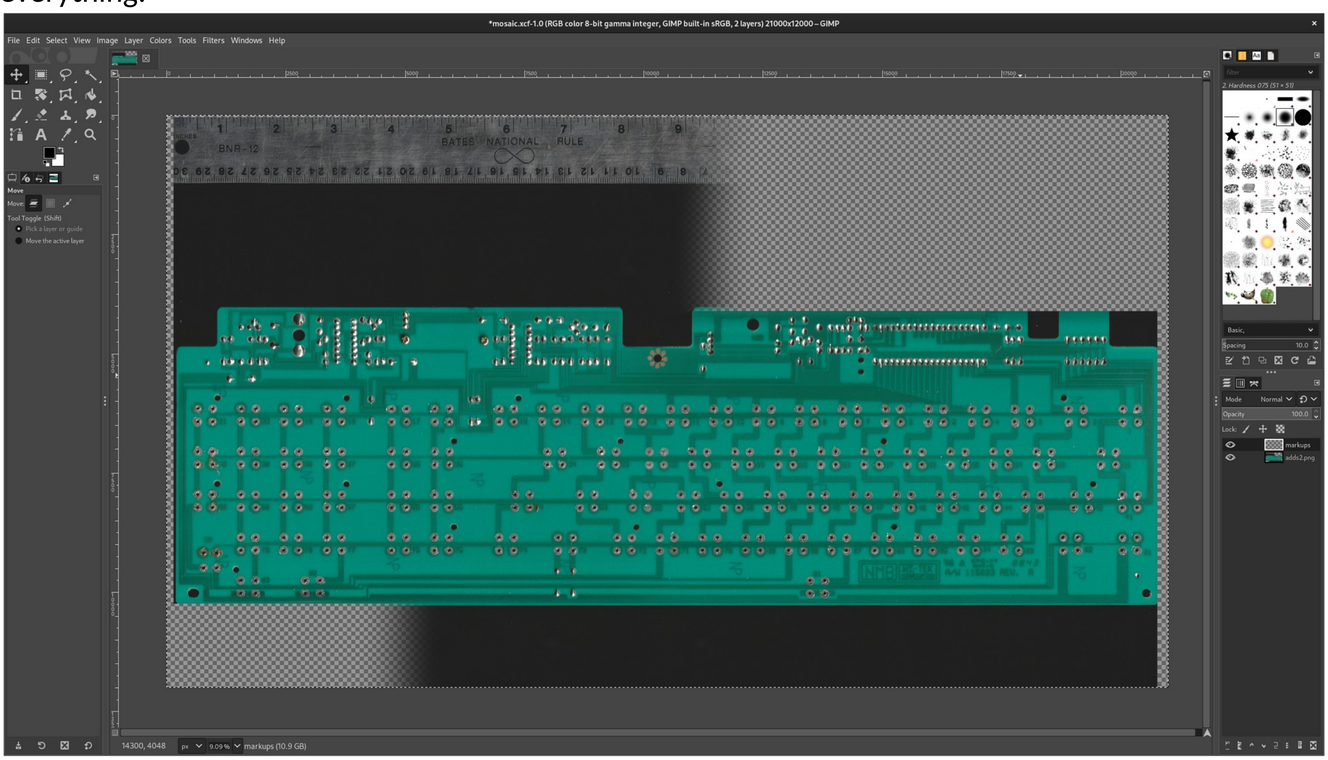




Fire up your photo editor (I'm using GIMP for no particular reason), make a canvas larger than you need, and import your photos. Set the orientation of each correct and blur the edges.
Look for features on the edges that you can use to line them up.





Once they're about as lined up as you can get them, merge the layers, trim the canvas size, and lock everything.



You should now have a mosaic of the entire PCB with the ruler in frame.

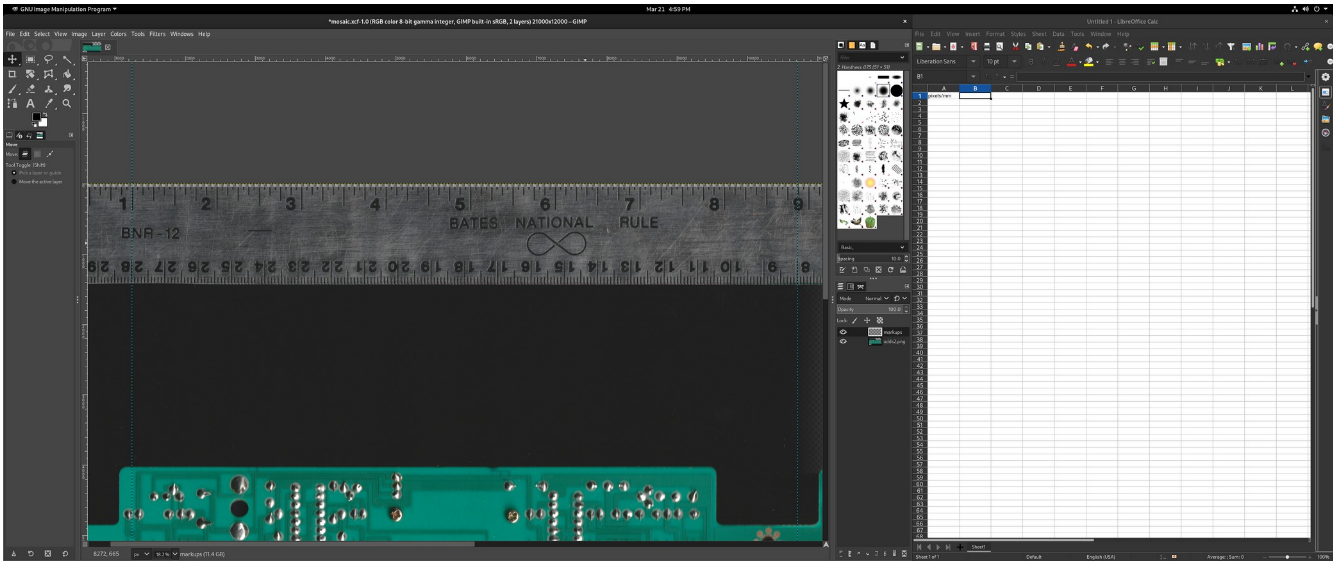
Measuring

There will be a heavy lean on averaging and rounding throughout this process so the end goal is just to ballpark the numbers to within a few pixels of the true values. We'll be using the grid tools to draw vertical and horizontal lines to see the XY locations in pixels.

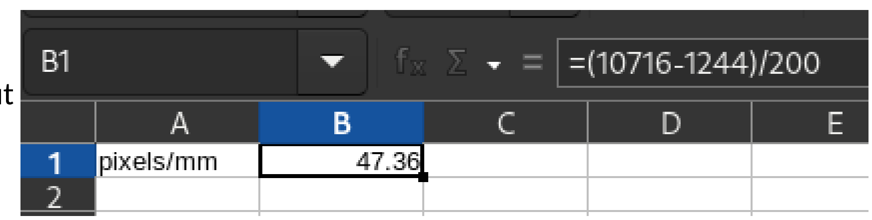
Fire up a spreadsheet program and put on some good music; this will take a while.

Setting the scale

We'll need a way to relate the pixel count of this image to real world dimensions. This is where the ruler comes in. Use the grid tool to get the pixel locations of a large segment on the ruler (larger = reduced error).



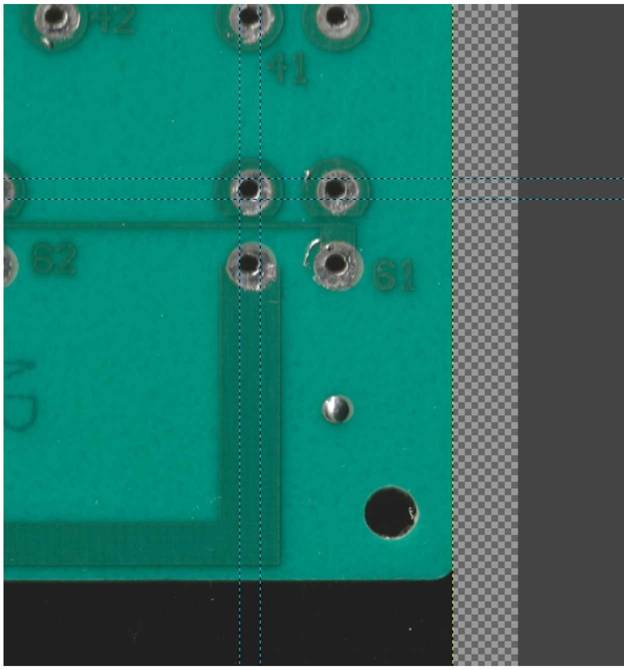
Once you have the X locations, divide their difference by the distance on the ruler. This measurement is going to be used throughout the process so double check the values and make sure it's as close as you can get it.



Defining origin

We'll need to define a datum point that we can use as "home" for all of our dimensioning. This will be 0, 0 on the coordinate grid. Your best bet will be to use a clearly defined pin on a switch on the side of the keyboard. Make careful note of which pin you are measuring here since every other measurement on another switch needs to be done to the same pin. I'm using the top left pin of switch 61 (bottom right on the backside of the PCB).

Put a line tangent to the top, bottom, left, and right side of the PCB hole.



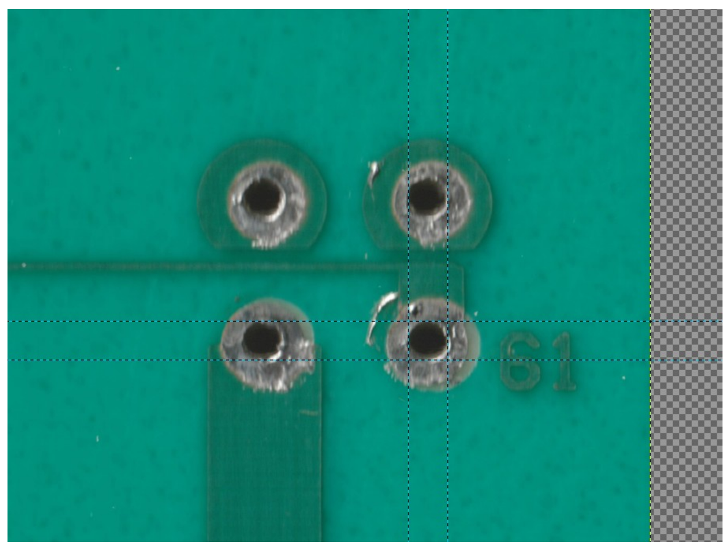
Plug the X and Y values into your spreadsheet like so:

Home x	Home y
19992	8815
20067	8889
20029.5	=AVERAGE(B4:B5)

Taking the average of both the X and both the Y will give you the center X and Y of the hole. This method of measuring round things will be used throughout this guide.

Footprint dimensions

This board uses Hi-Tek ‘space invader’ switches. They’re a bit unique in that they have 4 pins with the left and right side pins being electrically connected (vertically). For this guide I’ll be making the footprint with 4 pins but you get the idea for different pin counts.



Using the ‘home’ switch, take a measurement of the X and Y distance of the pins. I took the XY position for the bottom right pin using the same tangent line averaging and subtracted the origin location.

Take the offset pixel count and divide it by our scaling number to get the pin spacing in mm.

	A	B	C	D
1	pixels/mm	47.36		
2				
3	Home x	Home y		
4	19992	8815		
5	20067	8889		
6	20029.5	8852		
7				
8	**Footprint px**		**Footprint mm**	
9	Width	Height	Width	Height
10	20304	9087		
11	20380	9161		
12	20342	9124		
13	312.5	272	6.60	=B13/B1
14				

3	Home x	Home y
4	19992	8815
5	20067	8889
6	20029.5	8852
7		
8	**Footprint px**	
9	Width	Height
10	20304	9087
11	20380	9161
12	20342	9124
13	312.5	=B12-B6

Switch spacing

Since the row/column spacing on this keyboard is square (and nearly every other keyboard), we can find the rough ‘U’ (key units) dimension in mm by looking at the vertical spacing of the rows.

Using the tangent average measurement of top (Meas 1) and bottom (Meas 2) for the rows’ origin pins, set up the following table. The ‘relative px’ column is each column subtracting the Y location of our origin row. ‘Relative mm’ is that same measurement but divided by our scaling number.

	Row Y	Meas 1	Meas 2	Average	Relative px	Relative mm	
16	Row Y	Meas 1	Meas 2	Average	Relative px	Relative mm	
17	4	6126	6187	6156.5	-2695.5	-56.9151182	
18	3	7017	7089	7053	-1799	-37.9856419	18.9294764
19	2	7923	7986	7954.5	-897.5	-18.9505912	19.0350507
20	1 (home)	8815	8889	8852	0	0	18.9505912
21	0	9720	9792	9756	904	19.0878378	=F21-F20
22							**19.000739**
23							

Our final column is the row to row spacing. Taking the average of those numbers gets us our rough single U measurement in mm. Nearly every keyboard will use somewhere around 19.00-19.05mm. It’s best to sanity check this number by measuring something like 10 switches across to make sure the value is reasonable. In my case it was more like 19.02 when measuring multiple switches over. We’ll be cheating later on so it doesn’t need to be perfect but it helps to get it as close as you can.

Switch X	Meas 1	Meas 2	Average	Relative px	Relative mm	Relative u	Rounded u	Rounded mm	Difference
2	18873	18939	18906	-1123.5	-23.7225507	-1.2485553	-1.25	**-23.75**	-0.03
3	17742	17811	17776.5	-2253	-47.5717905	-2.50377845	-2.5	**-47.5**	0.07
16	5590	5654	5622	-14407.5	-304.212416	-16.0111798	-16	**-304**	0.21
18	3333	3398	3365.5	-16664	-351.858108	-18.5188478	-18.5	**-351.5**	0.36
22	19546	19611	19578.5	-451	-9.52280405	-0.50120021	-0.5	**-9.5**	0.02
23	18194	18265	18229.5	-1800	-38.0067568	-2.00035562	-2	**-38**	0.01
33	9080	9147	9113.5	-10916	-230.489865	-12.1310455	-12.125	**-230.375**	0.11
34	7949	8020	7984.5	-12045	-254.328547	-13.385713	-13.375	**-254.125**	0.20
42	18981	19048	19014.5	-1015	-21.4315878	-1.12797831	-1.125	**-21.375**	0.06
43	17967	18037	18002	-2027.5	-42.8103885	-2.25317834	-2.25	**-42.75**	0.06
52	9751	9814	9782.5	-10247	-216.36402	-11.38758	-11.375	**-216.125**	0.24
53	8625	8699	8662	-11367.5	-240.023226	-12.6328014	-12.625	**-239.875**	0.15
54	7271	7341	7306	-12723.5	-268.654983	-14.139736	-14.125	**-268.375**	0.28
62	18751	18826	18788.5	-1241	-26.2035473	-1.37913407	-1.375	**-26.125**	0.08
63	17511	17586	17548.5	-2481	-52.3859797	-2.75715683	-2.75	**-52.25**	0.14
73	8174	8243	8208.5	-11821	-249.598818	-13.1367799	-13.125	**-249.375**	0.22
74	6932	7006	6969	-13060.5	-275.770693	-14.514247	-14.5	**-275.5**	0.27
80 (Rotated)	737	817	777	-19252.5	-406.513936			**-406.514**	
81	13451	13527	13489	-6540.5	-138.101774	-7.2685144	-7.25	**-137.75**	0.35
82	2868	2942	2905	-17124.5	-361.581503	-19.0306054	-19	**-361**	0.58
83	1514	1595	1554.5	-18475	-390.097128	-20.5314278	=MROUND(G47,0.0625)		-0.59

Now that we know the vertical spacing of our rows, we’ll need to do the same but for the horizontal spacing. The general idea here is that any switch that is not 1u spaced to the switch next to it gets an X measurement using the tangent averaging method. That measurement is converted to key units, rounded to the nearest 1/8th unit, and converted back to mm with an exact 19.00mm key unit spacing.

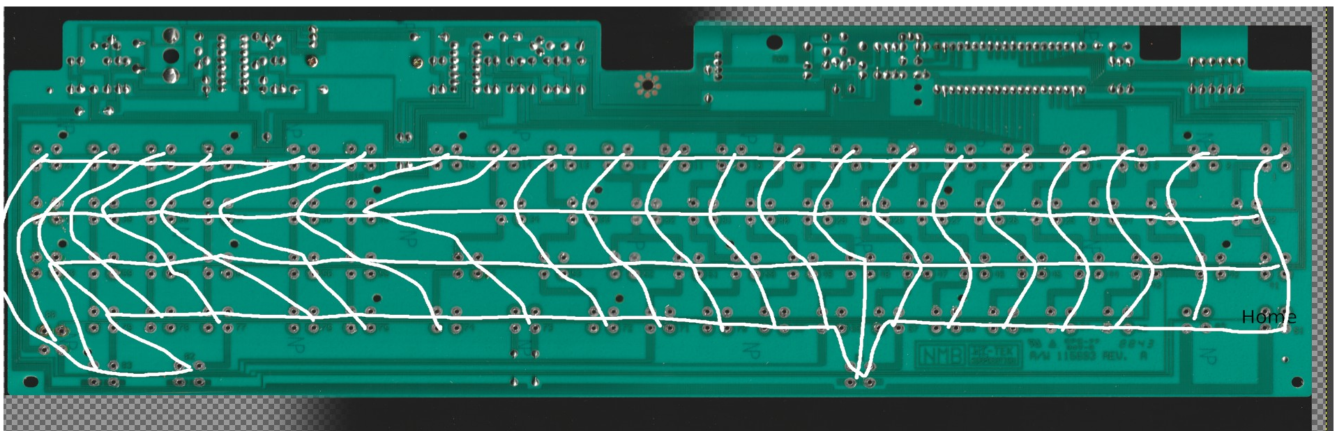
This will round all of our measurements to nice clean units but has the potential to introduce error. The last column is the overall error in mm introduced during this process. Here it's all under about 0.6mm which can be probably be attributed to my mosaic alignment.

Switch 80 is rotated on this PCB so it doesn't play by the same rules. We'll just use the exact XY mm measurements to the nearest switches and cross our fingers.

Board design

Matrix planning

The overall goal of a keyboard matrix is to have a bunch of rows and columns with one and only one switch between each. You'll want to do a bit of doodling to figure out how many you'll need. A square matrix (number of rows=number of cols) is technically the lowest number of pins you'll need but they are generally a pain to layout. Here's my rough plan which manages to fit into 25 pins (21 columns, 4 rows):



*I counted wrong on the first attempt and noticed later while laying out the PCB. You'll notice switch 40 silently disappear and that's why.

Project setup

For this I'll be using KiCAD. A pretty decent (slightly outdated) guide can be found here:

https://github.com/ruiqimao/keyboard-pcb-guide

I'll give a rough overview but a quick read of that might help with how to use some of the tools.

Download these libraries:

https://github.com/tmk/kicad_lib_tmk

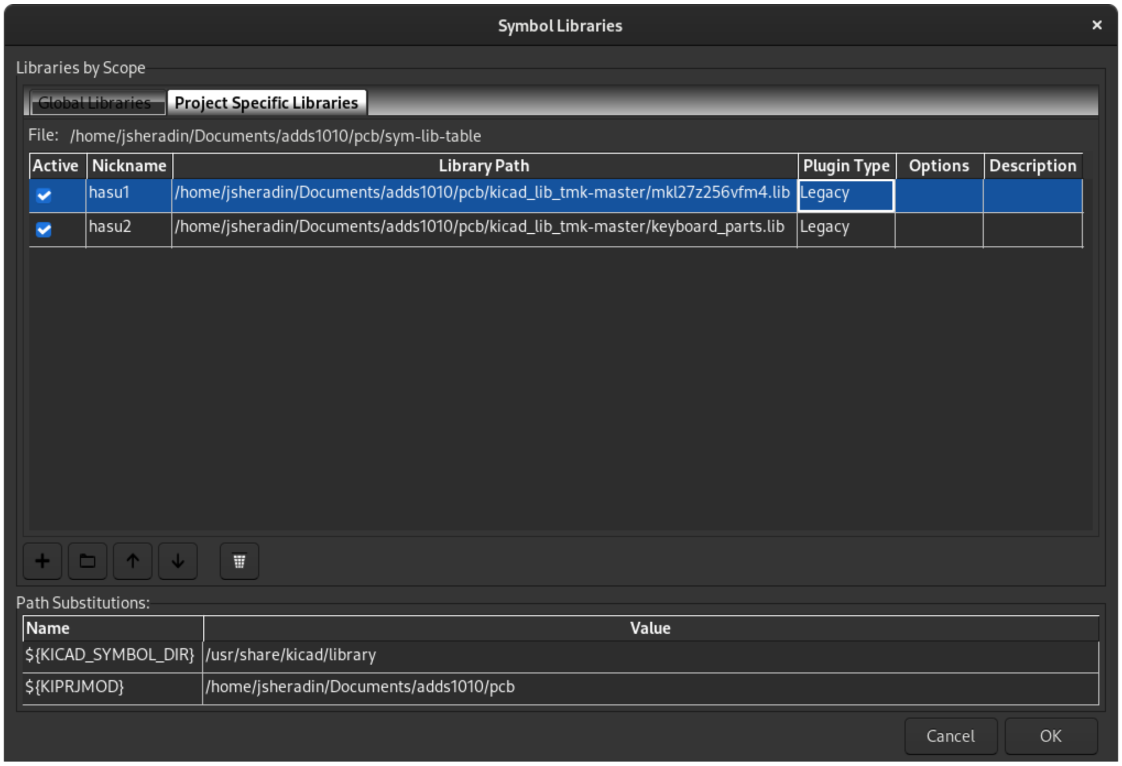
https://github.com/tmk/keyboard_parts.pretty

https://github.com/XenGi/teensy.pretty

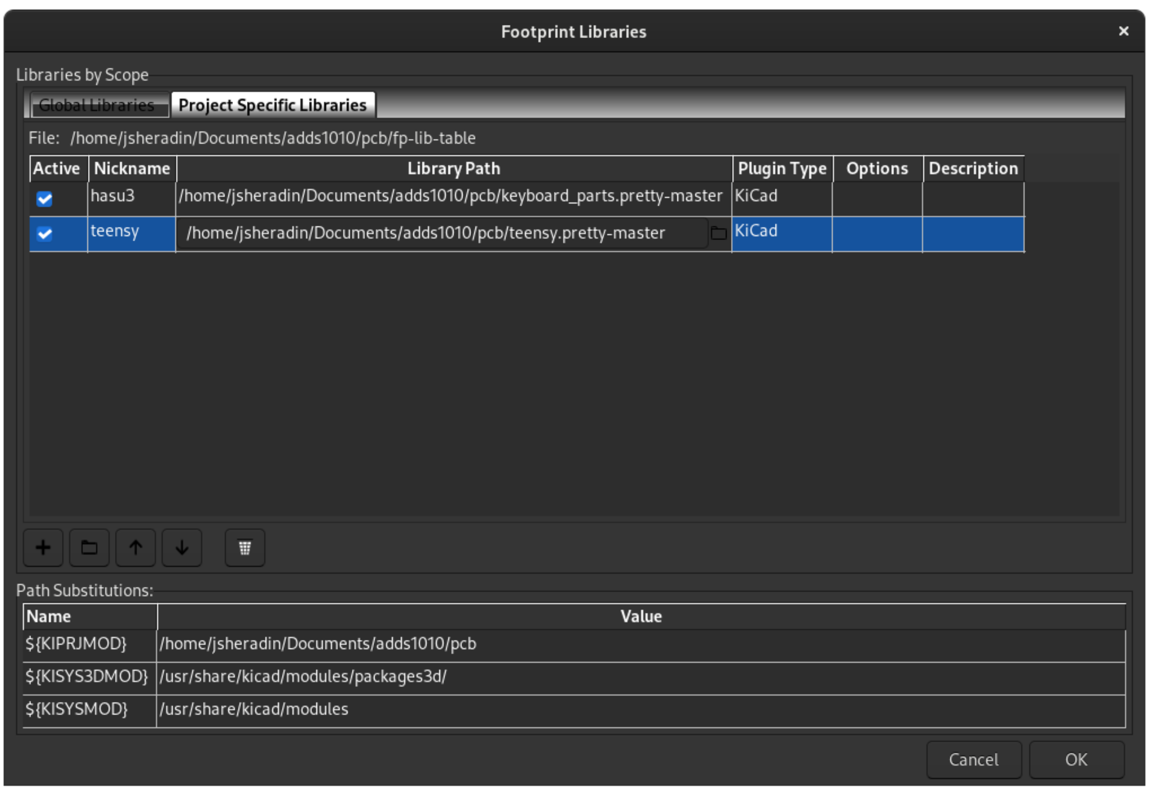
and save them to folders somewhere in the project directory so it's all in one place.

Fire up a new KiCAD project and set the location somewhere in your project directory.

Preferences>Manage symbol libraries, click add in Project Specific tab:

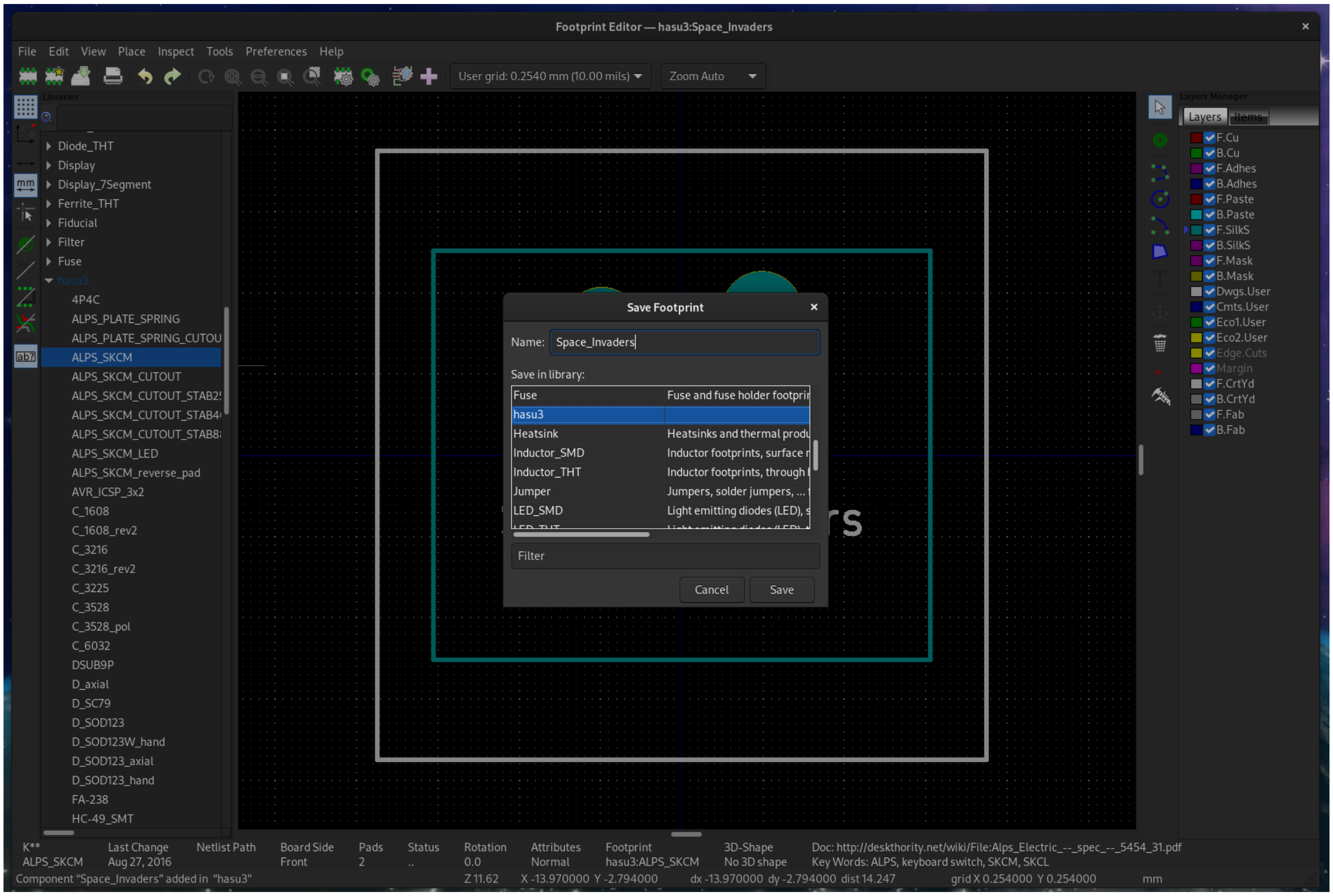


Preferences>Manage footprint libraries, import the rest:



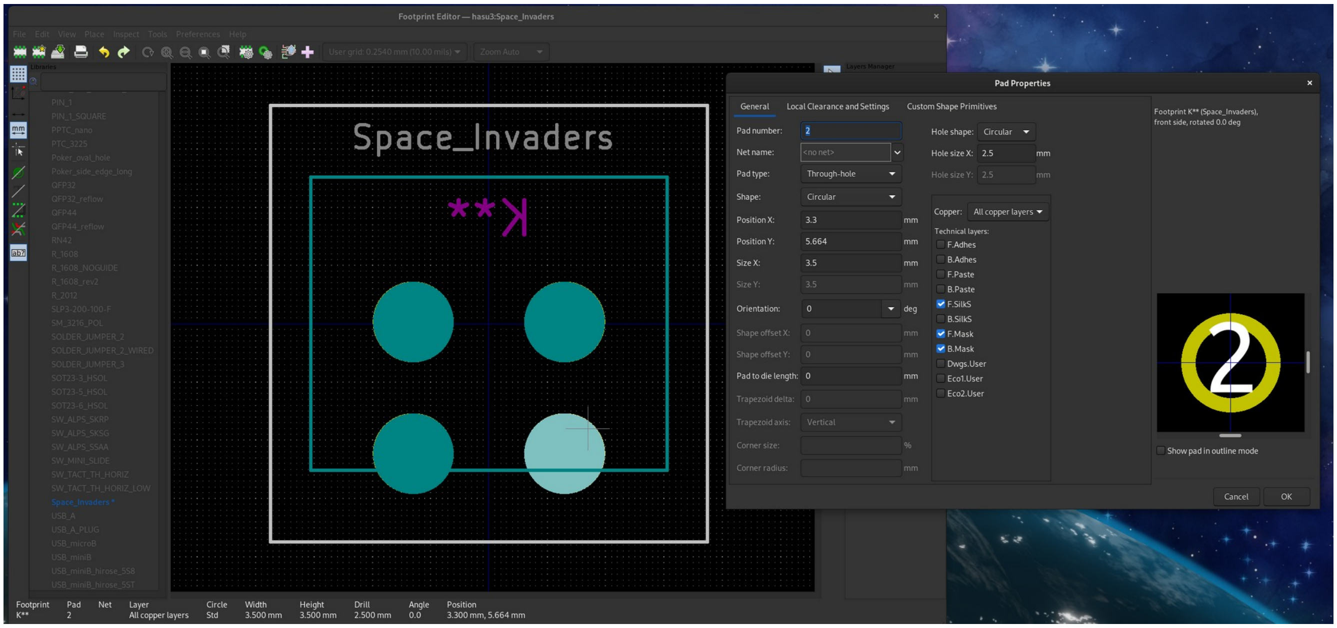
Switch Footprint

This step is only needed if you are working with a switch that you can't find a ready made footprint for. In the main window, open the Footprint Editor. Go to Hasu's folder, right click on a similar looking footprint and save as something descriptive. I chose one of the Alps footprints.

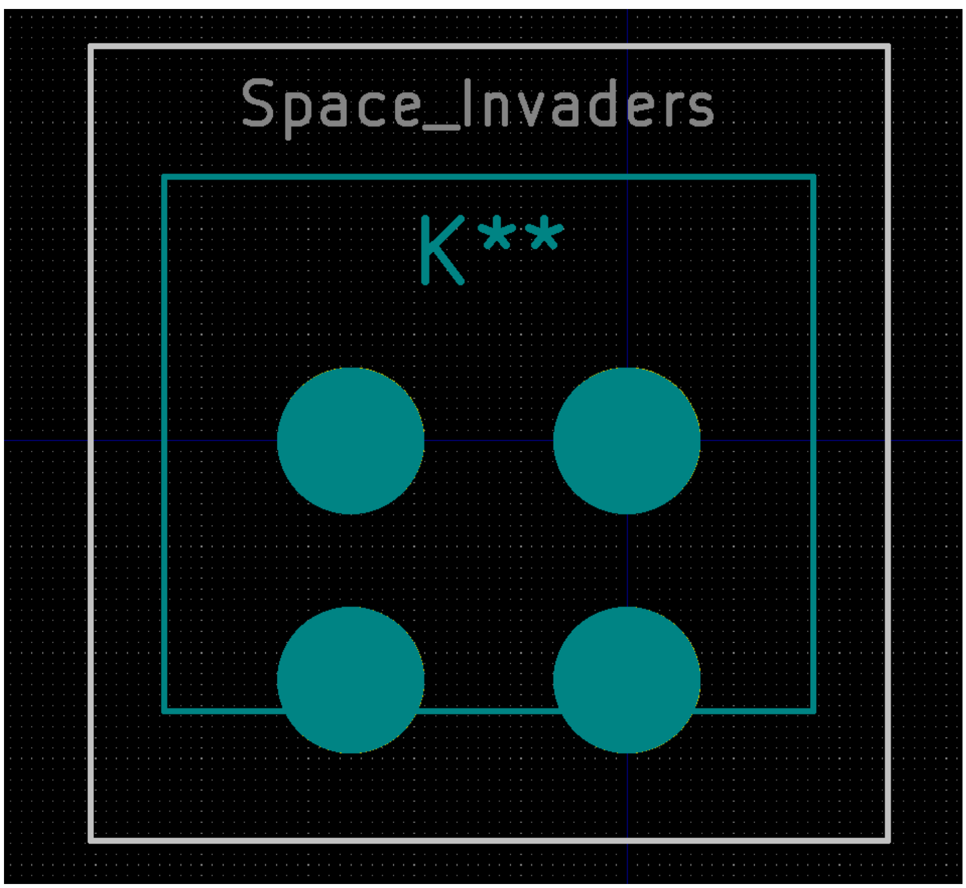


Copy and move the pads into the rough locations that your switch uses. Double click on each and punch in the real measurements. The relative location is all that's important here since our datum point is the pin itself, the silkscreen rectangle is just for looks. Make sure you set the pin number right (should only be a 1 or 2 unless you're messing with hall effect switches).

The cheat for all the measurements occurs here: you can way, way oversize the pads (hole size & pad size). This will take a ton of solder to assemble but it will let the entire PCB sort of float around the pins giving you literal wiggle room on alignment. Make them smaller according to your confidence in measuring.



To make our lives easier, select everything, right click, Move exactly, and punch in the values needed to line up our home pin to 0, 0. Keep in mind this is looking at the front of the PCB and our scans are from the back.



Schematic

Double click your .sch file to open the schematic editor. This is where we'll define our circuit but not actually set up the locations of anything.

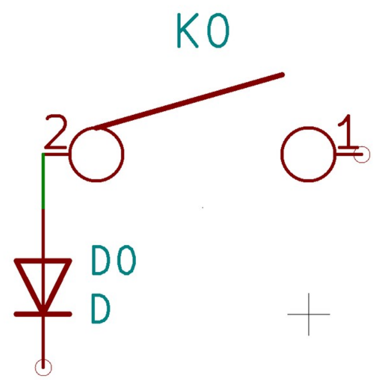
File>Page settings>set it to A2 (or bigger) so we have some room.

Place>Symbol>click somewhere>search for Teensy 2.0 in Hasu's library. Depending on the number of pins you need, you might need to go for a Teensy++.

You could replace the Teensy used throughout this guide with an Arduino Pro Micro (or any other USB capable based board) but I have had bad luck getting the bootloader entry via key combo working on anything but a Teensy. It's really nice to be able to enter programming mode via a key-combo.

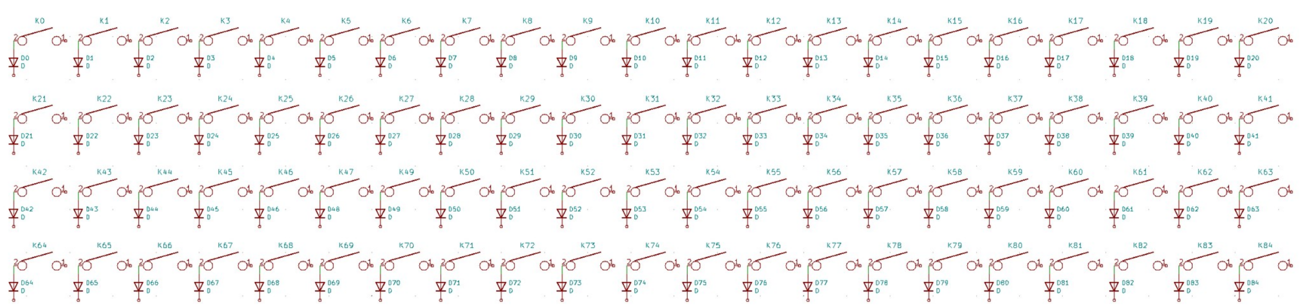
Next we'll need to build our key matrix. It consists of a switch and a diode for every key with columns and rows connecting them.

Use the same place method as for the Teensy but search for KEYSW. Double click the name and set it to K0. Repeat for a component called D. You should end up with this:

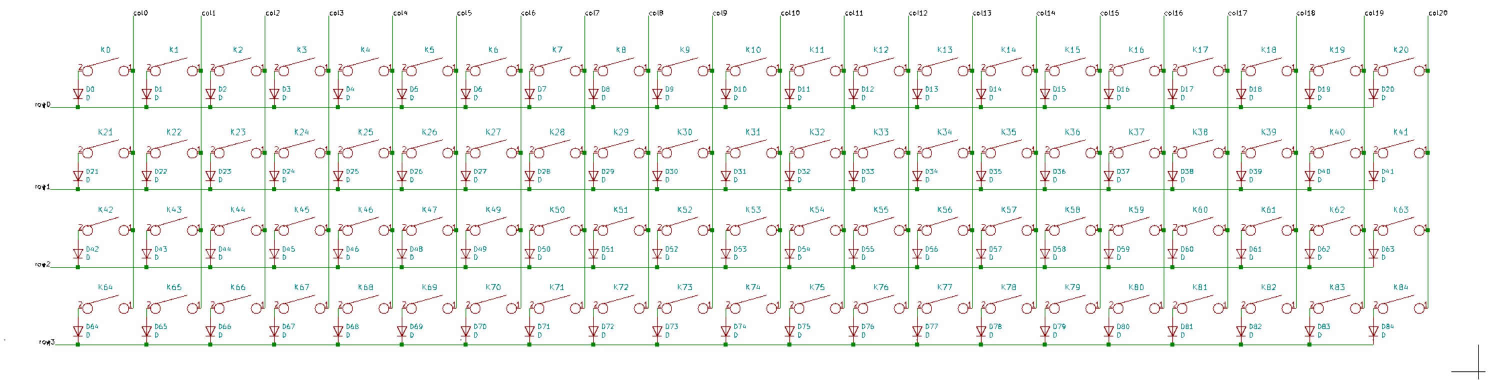


Copy and paste it a bunch of times until you have a row that matches the count of your doodle matrix.

Copy that row a bunch of times until you have the full layout. There's probably an automated way to do this, but click through each and every switch and diode and set their number accordingly.



Draw your rows and columns, add bus labels to each using the button on the right. Start counting at 0 and just call them something consistent.



For now we're not going to connect any of these to the Teensy.

Click 'Assign PCB footprints to schematic symbols' and configure them like so:

Diodes: Hasu's D_SOD123_axial

Switches: Our footprint

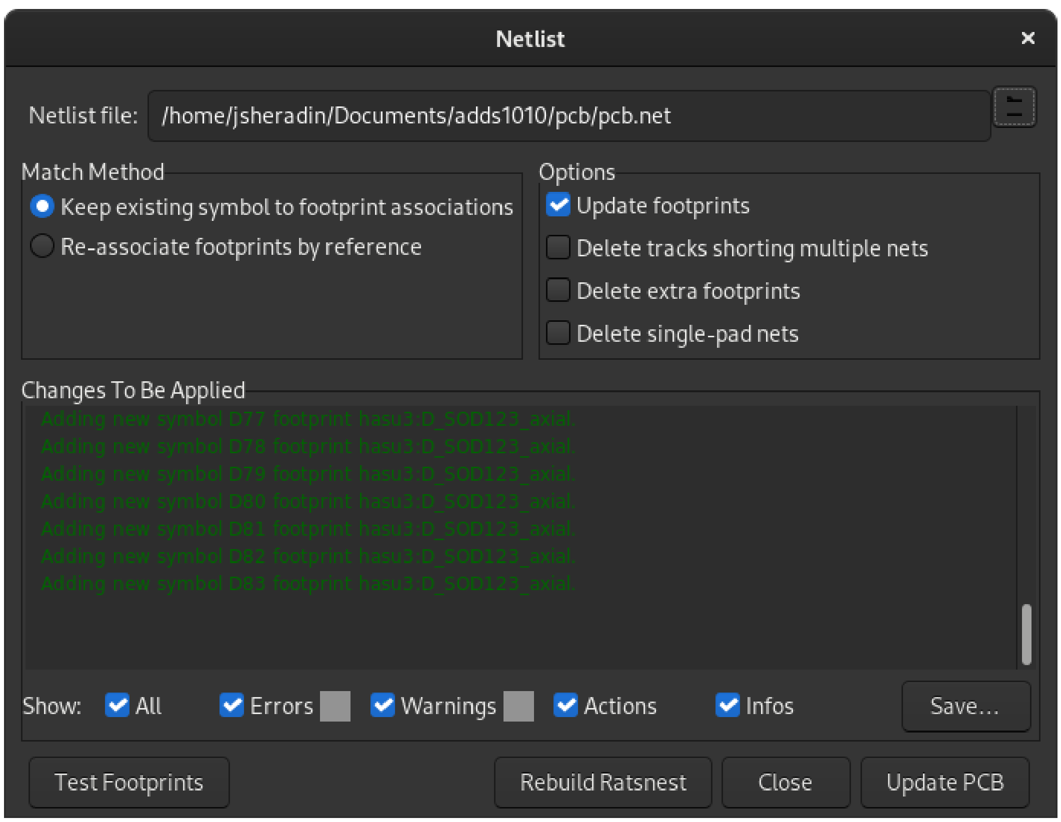
Teensy: XenGi's Teensy 2.0

Click Generate Netlist and save it to the project folder.

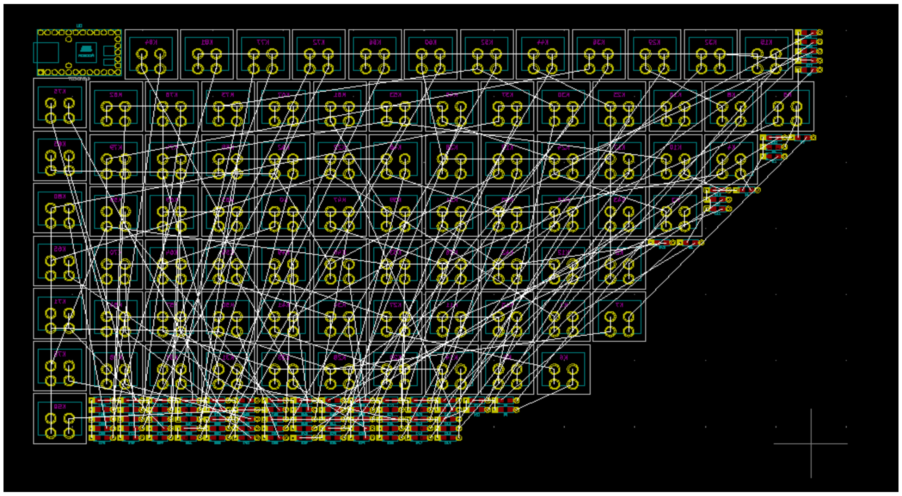
Switches

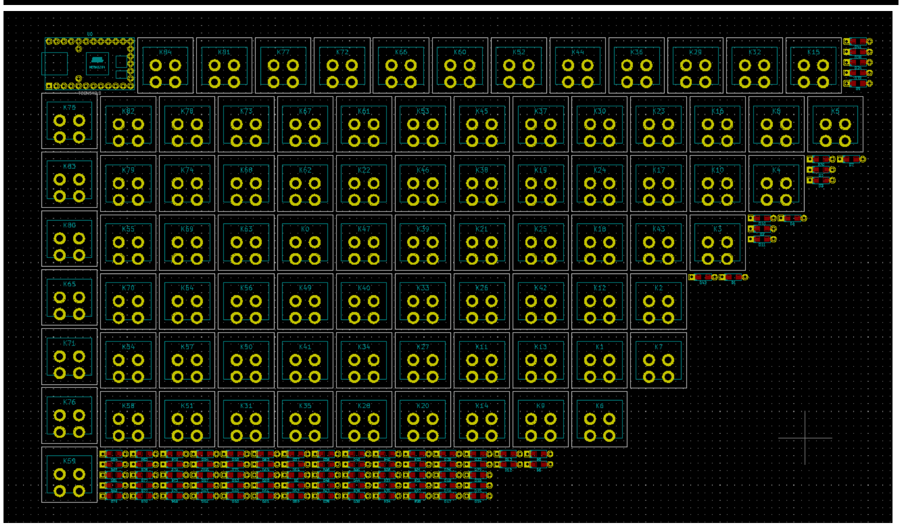
Finally we can get into designing the PCB itself. Go back to the KiCAD main window and double click the .kicad_pcb file to open the PCB editor. Again, go to File>Page settings and change it to A2.

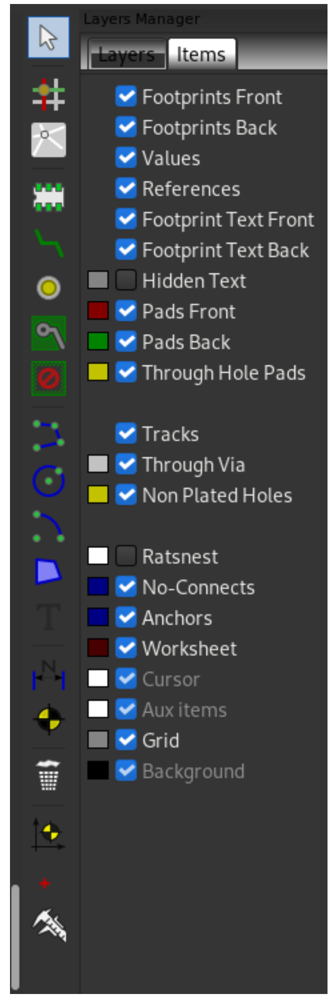
Click load netlist and select the file you just exported. Click Update PCB and then Close



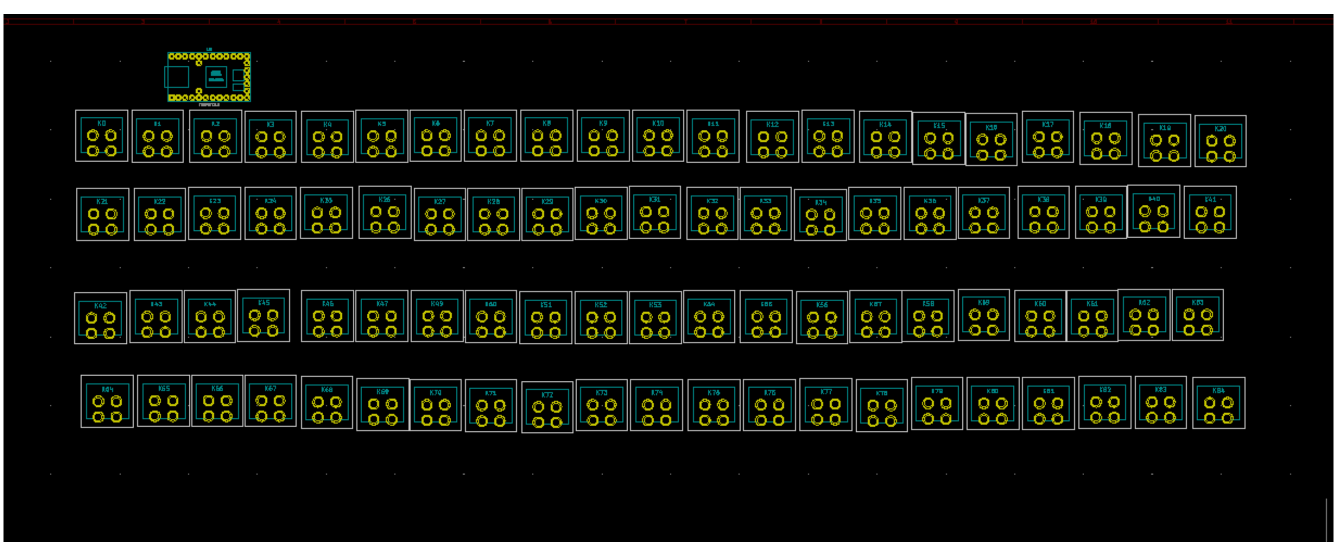
Everything is now plopped down in no particular order. To help make sense of it, go to the menu on the right and disable Ratsnest (note I updated my footprint after realizing the text was flipped).



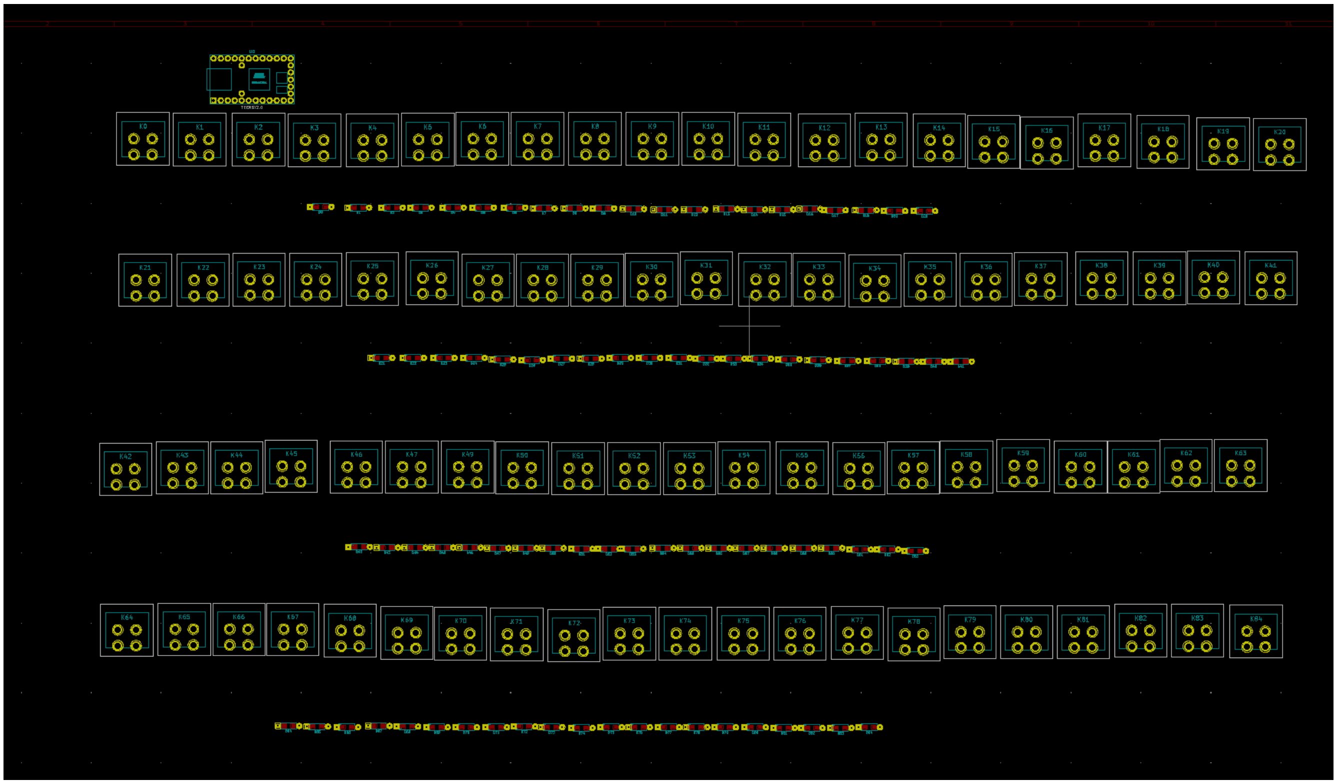




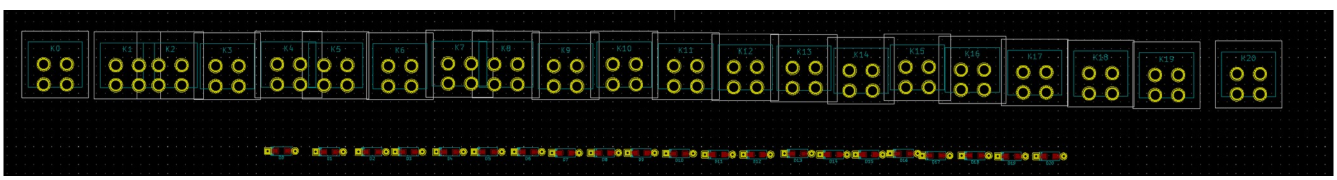
The next part is tedious; I don't know of an easier way. Click and drag all the switch footprints into order, matching your schematic file. Make sure to bang your head on a desk when you realize you skipped 48 when numbering everything (doesn't actually matter so I'm just going to live with it).



Do the same for the diodes:

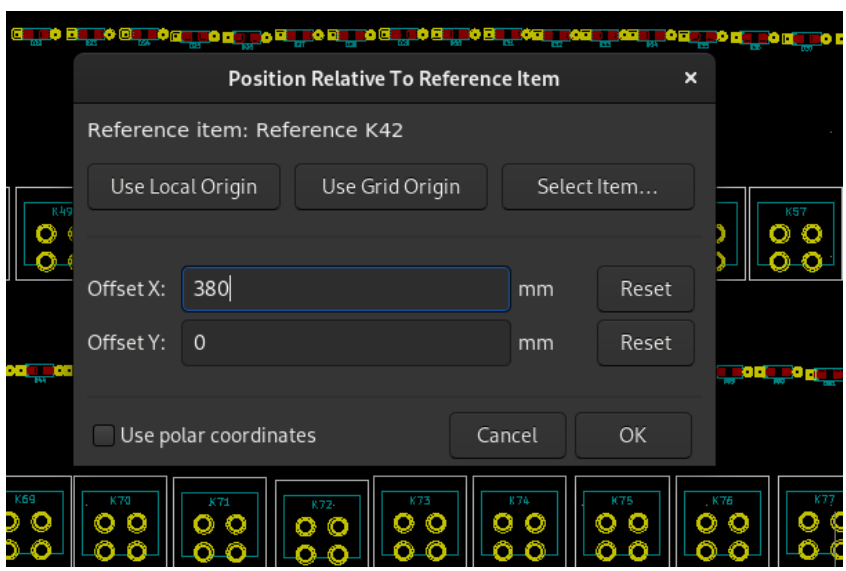


Now drag the switches a bit overlapping to give yourself some room on the ends. Try not to overlap them too much that they change order.

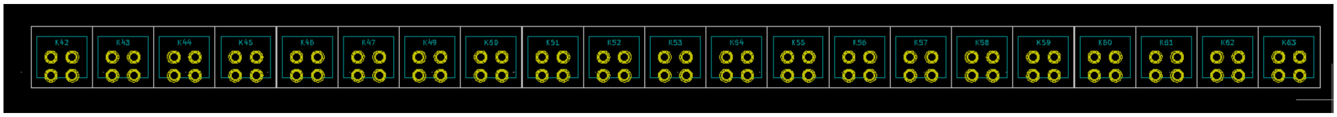


Select the far right switch, hit Ctrl+R, use Select Item and choose the far left switch as the reference. We'll plug in a rounded mm distance to set the overall width. Since it's 21 switches wide and the first doesn't count, we'll do 20*19.00=380mm.

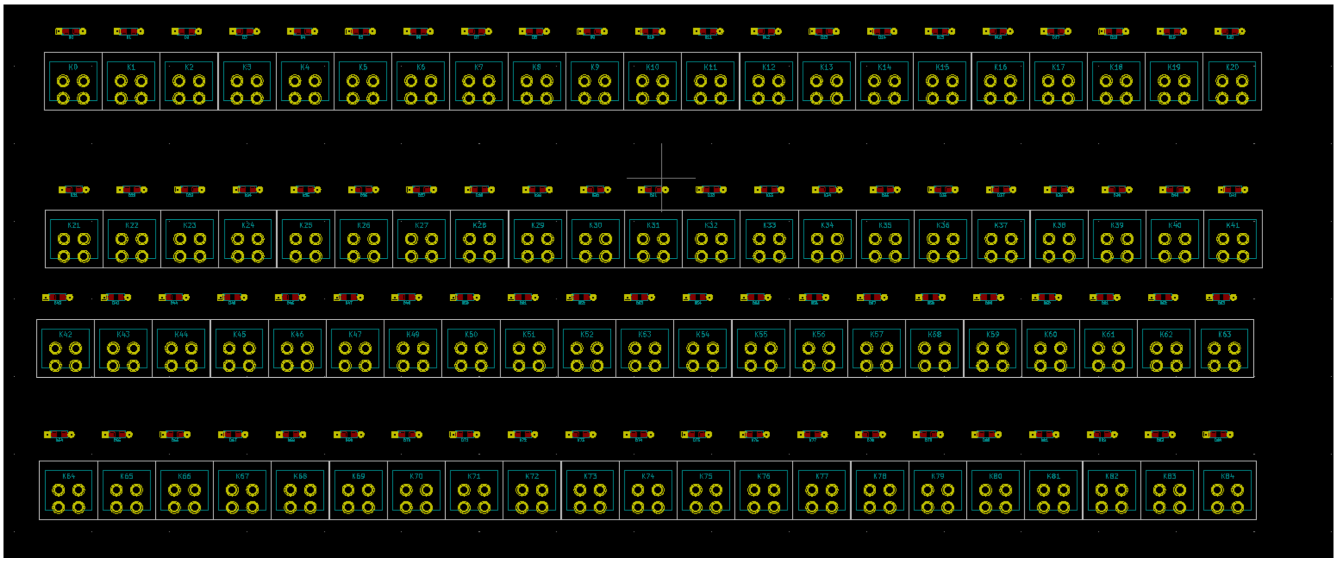
The left and rightmost switches will now have the correct total horizontal distance.



Select them all, right click, Align/Distribute>Align to top. Right click again, Align/Distribute>Distribute Horizontally.

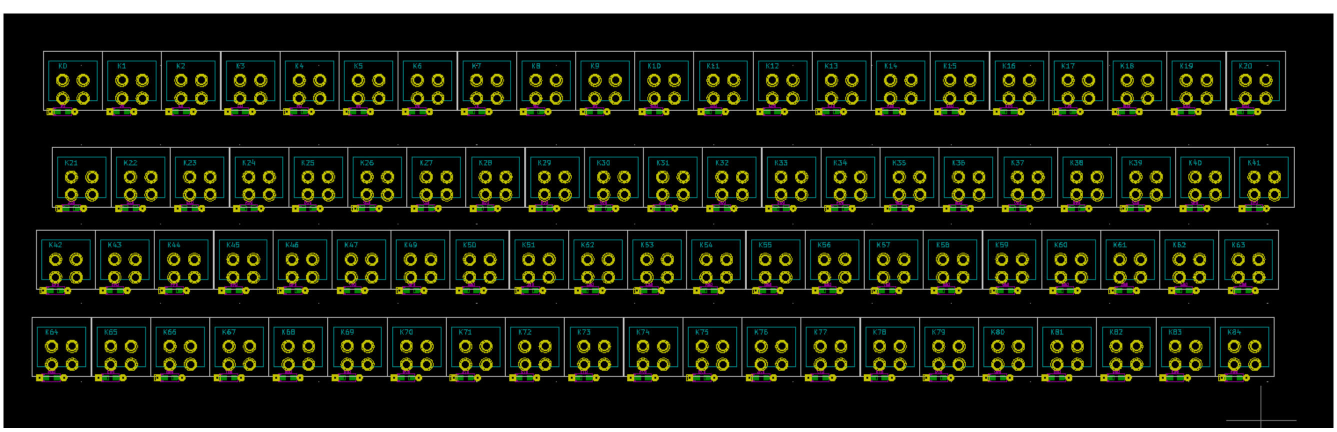


This should plop all the switches in a nice square, perfectly spaced row. Repeat this for every row of switches and diodes.

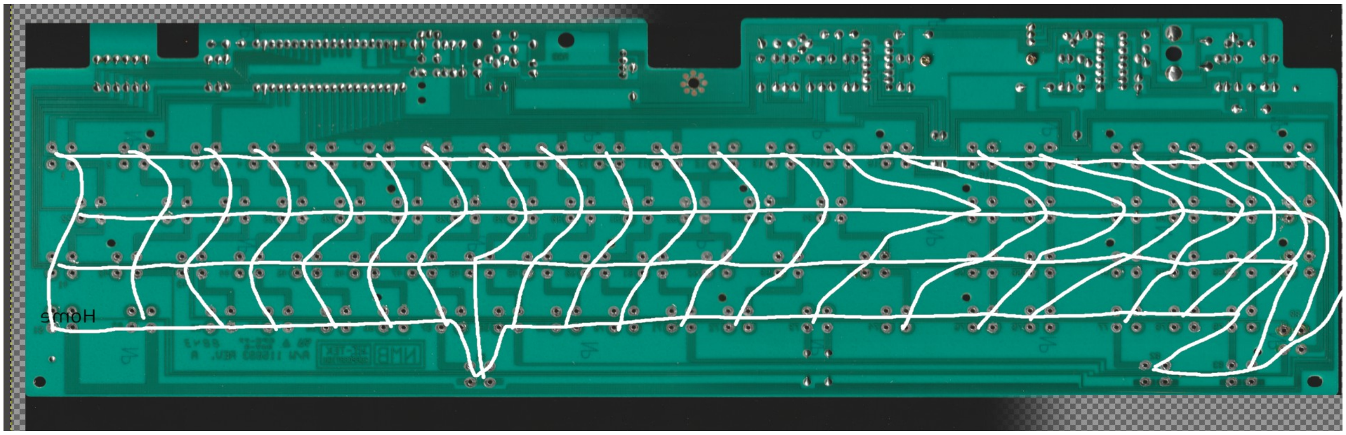


We'll now need to match up the diodes and switches. The groove mentioned during the planning stage is roughly 10.05mm (using some calipers) below our datum point. I like to put diodes on the bottom of the PCB just to make it possible to replace them without having to desolder everything if one randomly dies (has happened to me). To do this, select the entire row of diodes and press F.

Since our datum point is the pin that I measured against, we can just use the switch as the relative position. Select the whole row of diodes, hit Ctrl+R, choose the left most switch as the 'relative to' and punch in 10.05 for the Y position. For X, I chose 0 but you could offset them to the side if that works better.

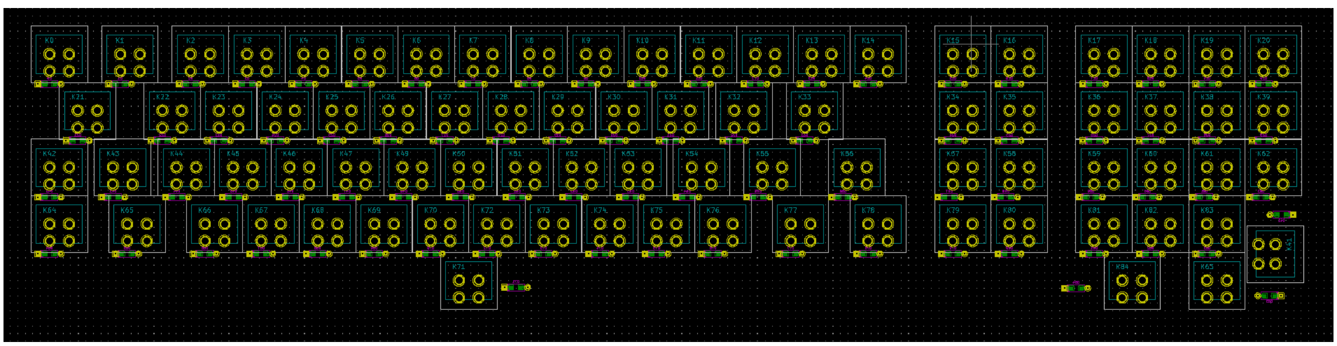


We're slowly approaching something that looks like a keyboard.



Going back to our doodled matrix, it’s finally time to start laying things out to match. Since we’re currently designing from the front of the PCB and the scan/measurements are from the back I’ll mirror it to make it a bit easier to look at. You probably should flip the images from the get-go to avoid the headache.

I’ll start from our home point (switch 64 in our new design), go one row at a time, select groups of switches, and use Ctrl+R. This is where we use the Rounded mm column from our spreadsheet.



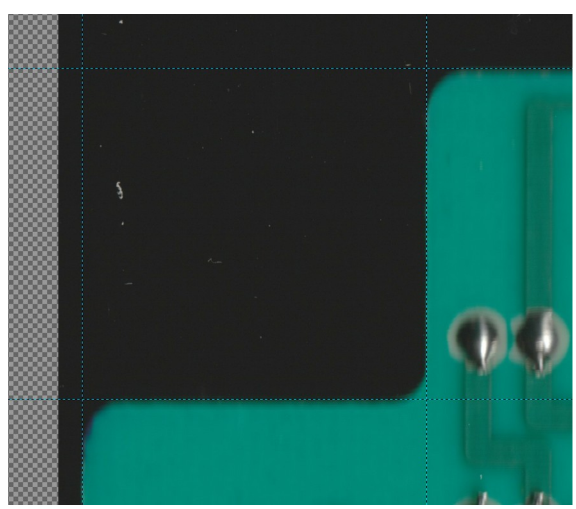
For a few of the diodes I moved them around a bit just to give the edges of the board a bit more room. If everything went according to plan, this should match the original PCB.

PCB Edges

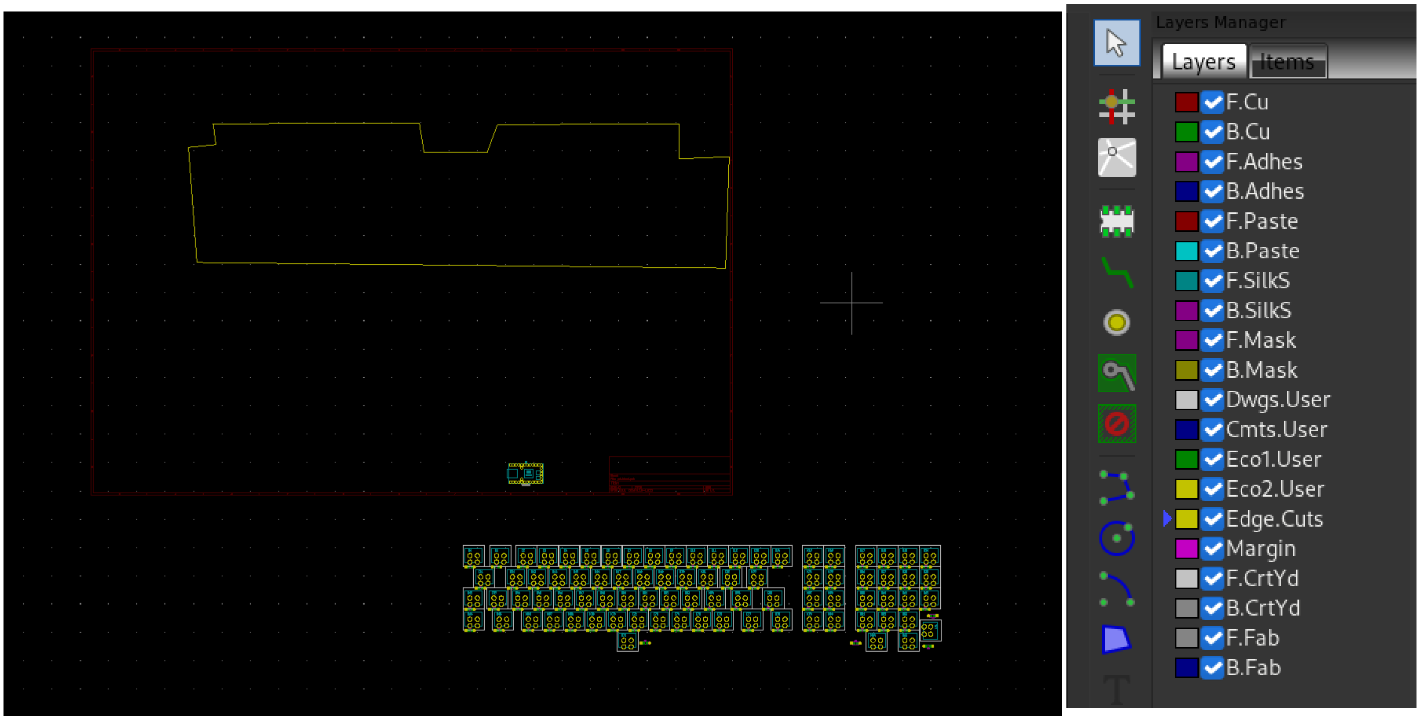
Going back to our image program, do some more pixel counting to figure out the relative locations of the PCB edges.

You get the idea by now.

2	**Edge**	Px	Relative Px	**Relative mm**
3	Top	4020	-4832	**-102.02703**
4	Notch	4846	-4006	**-84.586149**
5	Bottom	10268	1416	**29.8986486**
6				
7	Right	207	-19822.5	**-418.54941**
8	Cutout right	1066	-18963.5	**-400.41174**
9	Notch right	9560	-10469.5	**-221.06208**
0	Notch left	11018	-9011.5	**-190.2766**
1	Cutout left	19757	-272.5	**-5.7538007**
2	Left	20769	739.5	**15.6144426**
3				
4	Controller right	18048	-1981.5	**-41.839105**
5	Controller left	18705	-1324.5	**-27.966639**
6	Controller center		-1653	**-34.902872**

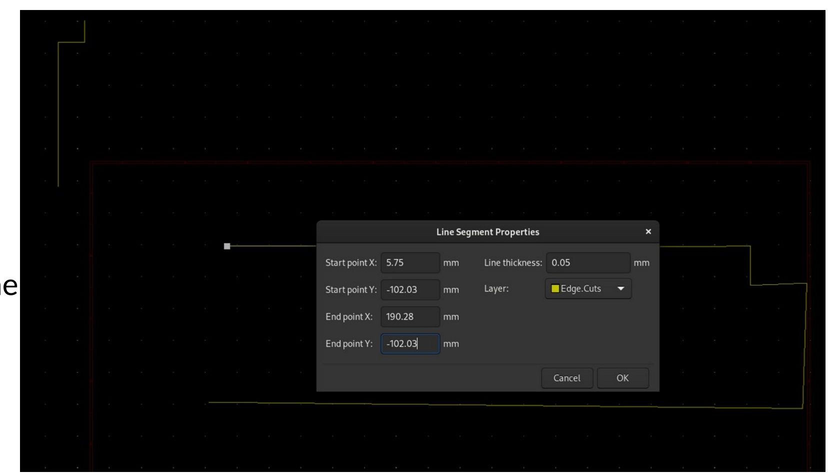


Select all your footprints and drag them out of the way for now.



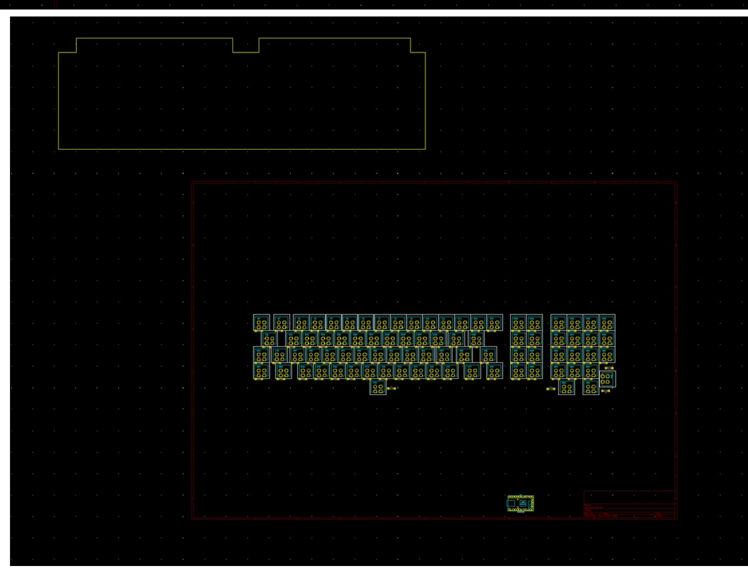
Set ‘Edge cuts’ as your active layer and use the line tool (blue with green dots) to doodle a very rough outline of your PCB.

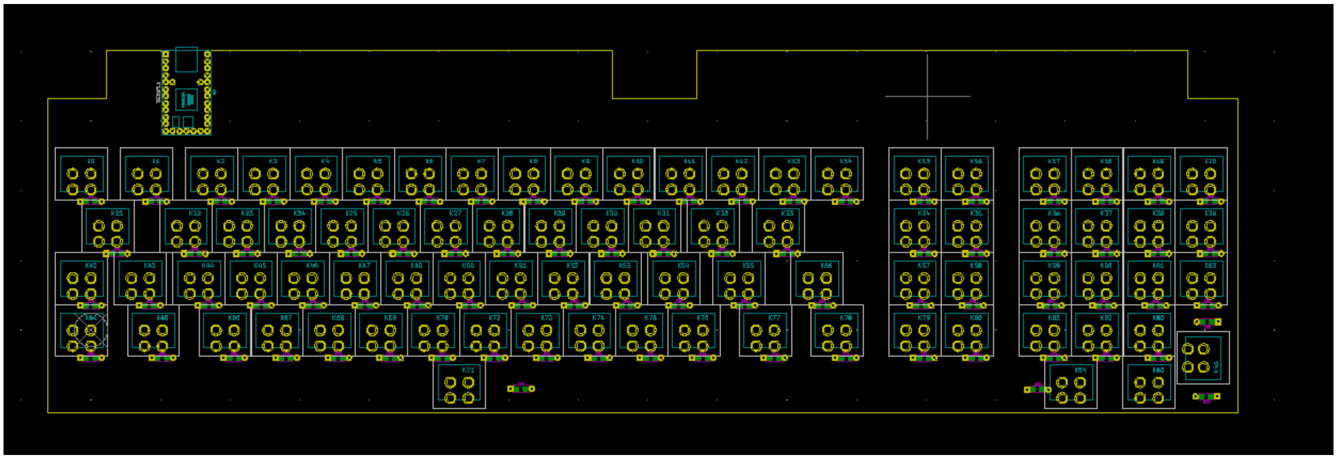
Select each line and one by one punch in the correct coordinates for the endpoints that we have in the spreadsheet. Don’t worry about it being way out in space for now.



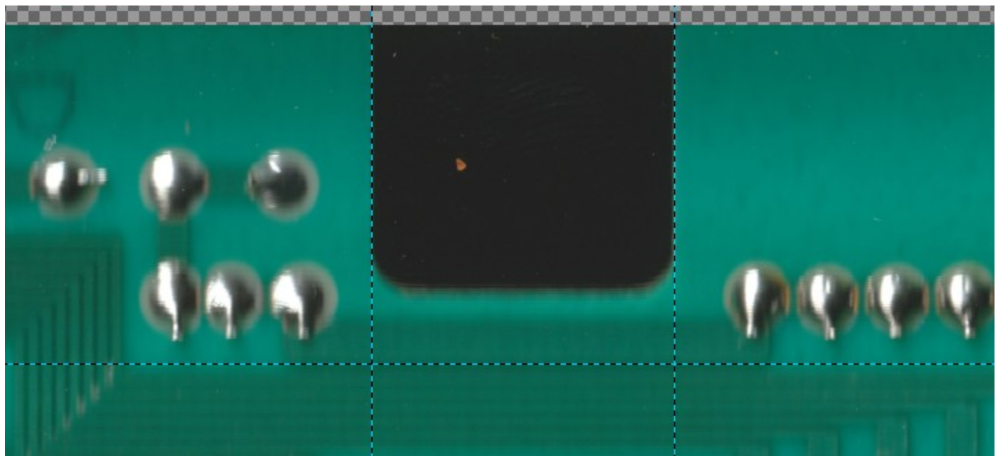
Move your switches and diodes back to the center of the page.

Select all of the lines and use the relative positioning tool to set them to the proper location relative to the home switch. Just punch in 0, 0 at first to see where it’s considering home for the outline (usually a top left corner) and then repeat with the actual relative value.



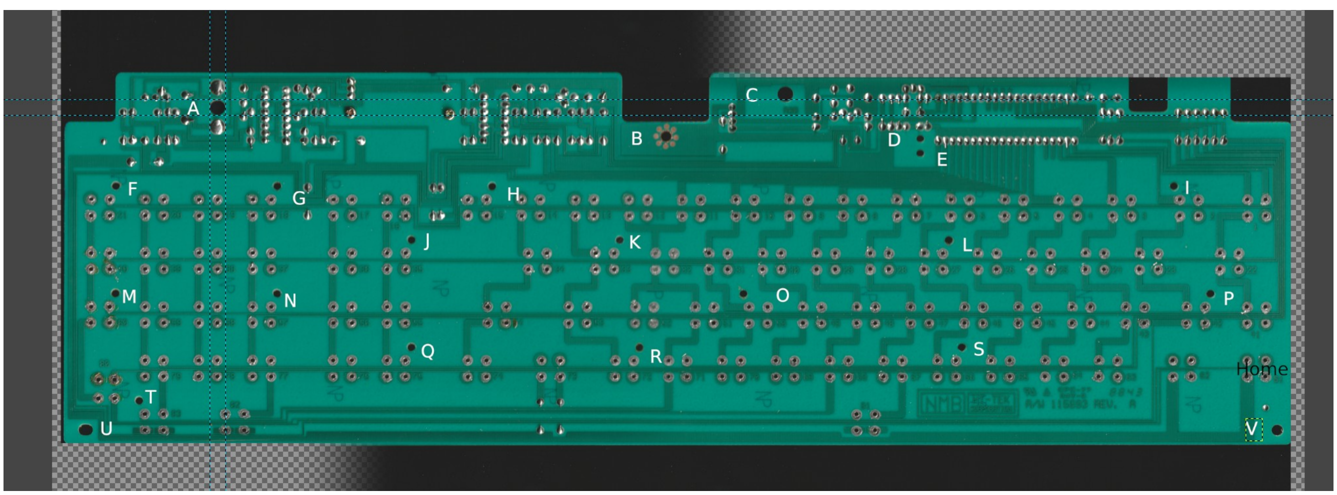


For my chosen controller placement, I measured the sides of the notch for the cable entry and centered it on them.



Holes

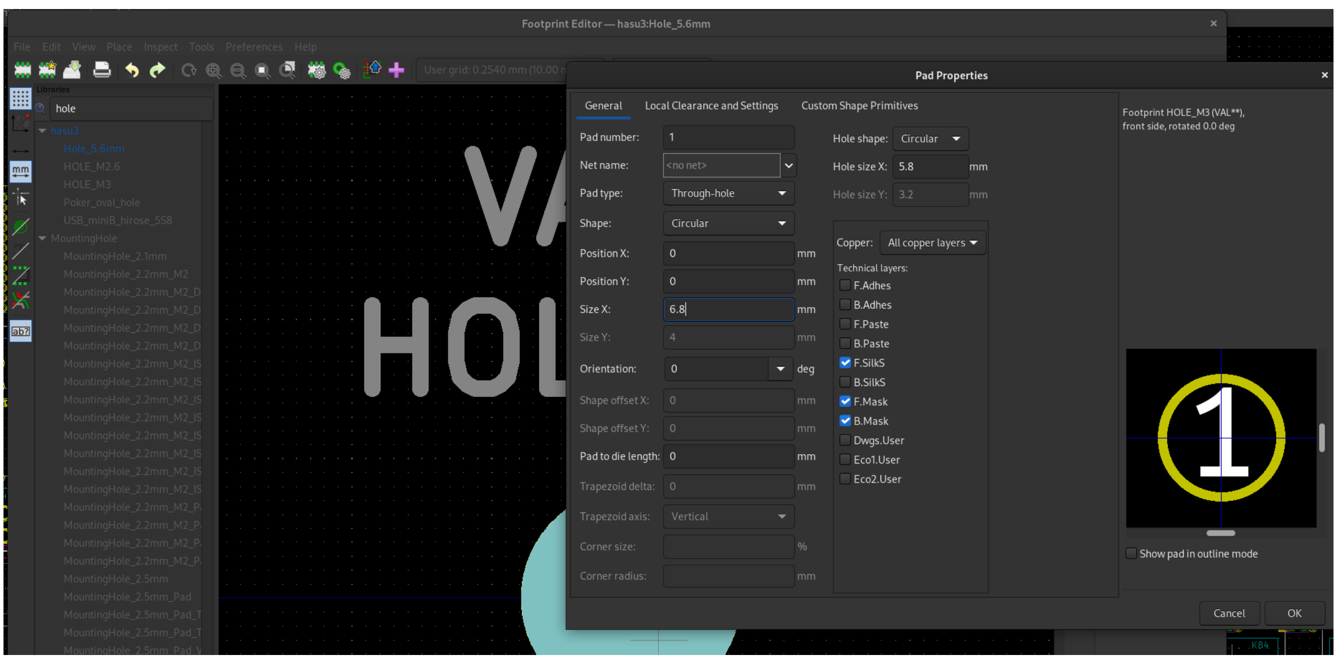
Head back to the image editor and go through and label all of the holes to help keep things sane.



Do the same tangent line measurements and throw the values into a table:

Hole	Top px	Left px	Bottom px	Right px	Top mm	Left mm	Bottom mm	Right mm	X mm	Y mm	Diameter mm
A	4479	2646	4749	2907	94.57	55.87	100.27	61.38	-364.3	-89.5	5.6
B	5010	10205	5192	10387	105.79	215.48	109.63	219.32	-205.5	-79.2	3.8
C	4259	12165	4520	12428	89.93	256.86	95.44	262.42	-163.3	-94.2	5.5
D	5073	14493	5197	14618	107.12	306.02	109.73	308.66	-115.6	-78.5	2.6
E	5313	14493	5445	14618	112.18	306.02	114.97	308.66	-115.6	-73.3	2.7
F	5852	1001	5998	1142	123.56	21.14	126.65	24.11	-400.3	-61.8	3.0
G	5852	3711	5998	3847	123.56	78.36	126.65	81.23	-343.1	-61.8	3.0
H	5852	7317	5998	7445	123.56	154.50	126.65	157.20	-267.1	-61.8	2.9
I	5852	18736	5998	18872	123.56	395.61	126.65	398.48	-25.9	-61.8	3.0
J	6758	5960	6900	6090	142.69	125.84	145.69	128.59	-295.7	-42.7	2.9
K	6758	9450	6900	9582	142.69	199.54	145.69	202.32	-222.0	-42.7	2.9
L	6758	14965	6900	15114	142.69	315.98	145.69	319.13	-105.4	-42.7	3.1
M	7671	998	7805	1127	161.97	21.07	164.80	23.80	-400.5	-23.5	2.8
N	7671	3708	7805	3840	161.97	78.29	164.80	81.08	-343.2	-23.5	2.8
O	7671	11528	7805	11655	161.97	243.41	164.80	246.09	-178.2	-23.5	2.8
P	7671	19355	7805	19487	161.97	408.68	164.80	411.47	-12.8	-23.5	2.8
Q	8558	5955	8702	6087	180.70	125.74	183.74	128.53	-295.8	-4.7	2.9
R	8558	9786	8702	9911	180.70	206.63	183.74	209.27	-215.0	-4.7	2.8
S	8558	15181	8702	15319	180.70	320.54	183.74	323.46	-100.9	-4.7	3.0
T	9456	1396	9595	1526	199.66	29.48	202.60	32.22	-392.1	14.2	2.8
U	9890	445	10142	685	208.83	9.40	214.15	14.46	-411.0	24.6	5.2
V	9925	20444	10120	20642	209.57	431.67	213.68	435.85	10.8	24.7	4.1

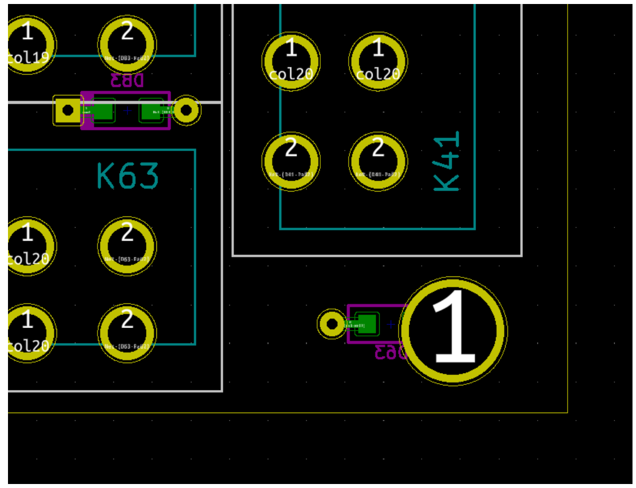
Convert from the pixel count to mm, average left/right and top/bottom and subtract the home location (in mm) to get the coordinates of the center of all the holes. Take the average of the difference between top/bottom and left/right to get the diameters of all the holes. You can usually see what the intended size groups of holes are. Color coding makes it a lot easier.

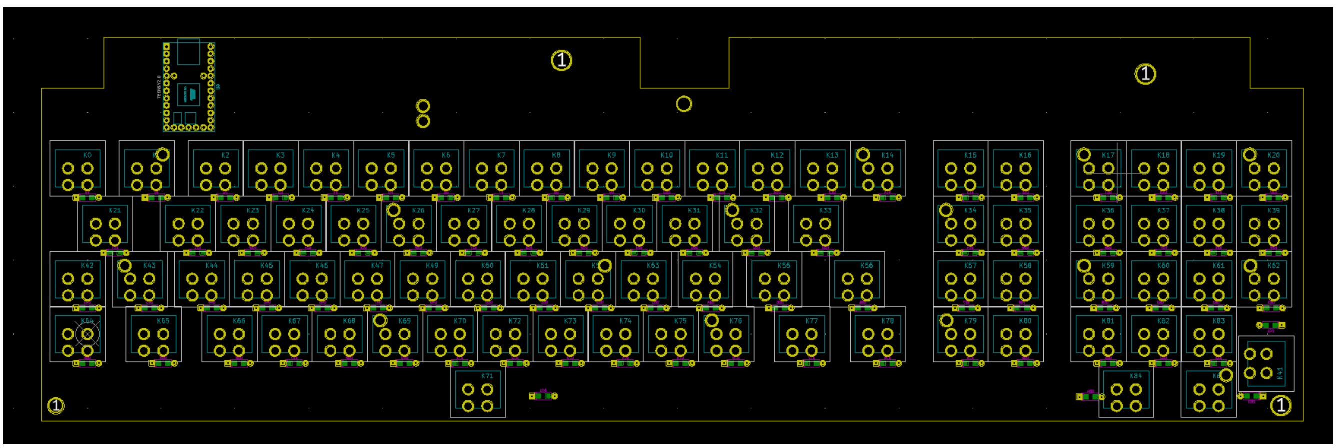


Hop back into the footprint editor and make a copy of HOLE_M3. Change it's name to reflect the size. Set the hole size to match the spreadsheet (pad proportionately to your confidence). Repeat for each of your hole size groups.

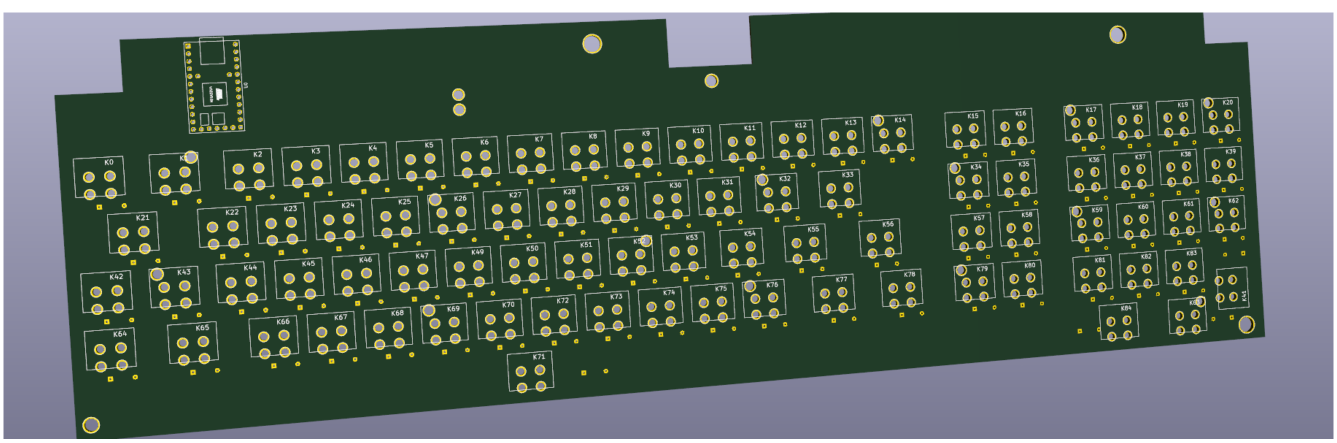
Go back to the PCB editor, hit ‘O’ and click somewhere. Select your desire hole size, place it somewhere, select it, hit Ctrl+R, punch in the XY from the spreadsheet, rinse and repeat for all the holes.

Once they’re all placed, go through and double check that they’re not interfering with anything. In my case I had a diode in the way of one so I found a different spot to put the diode.





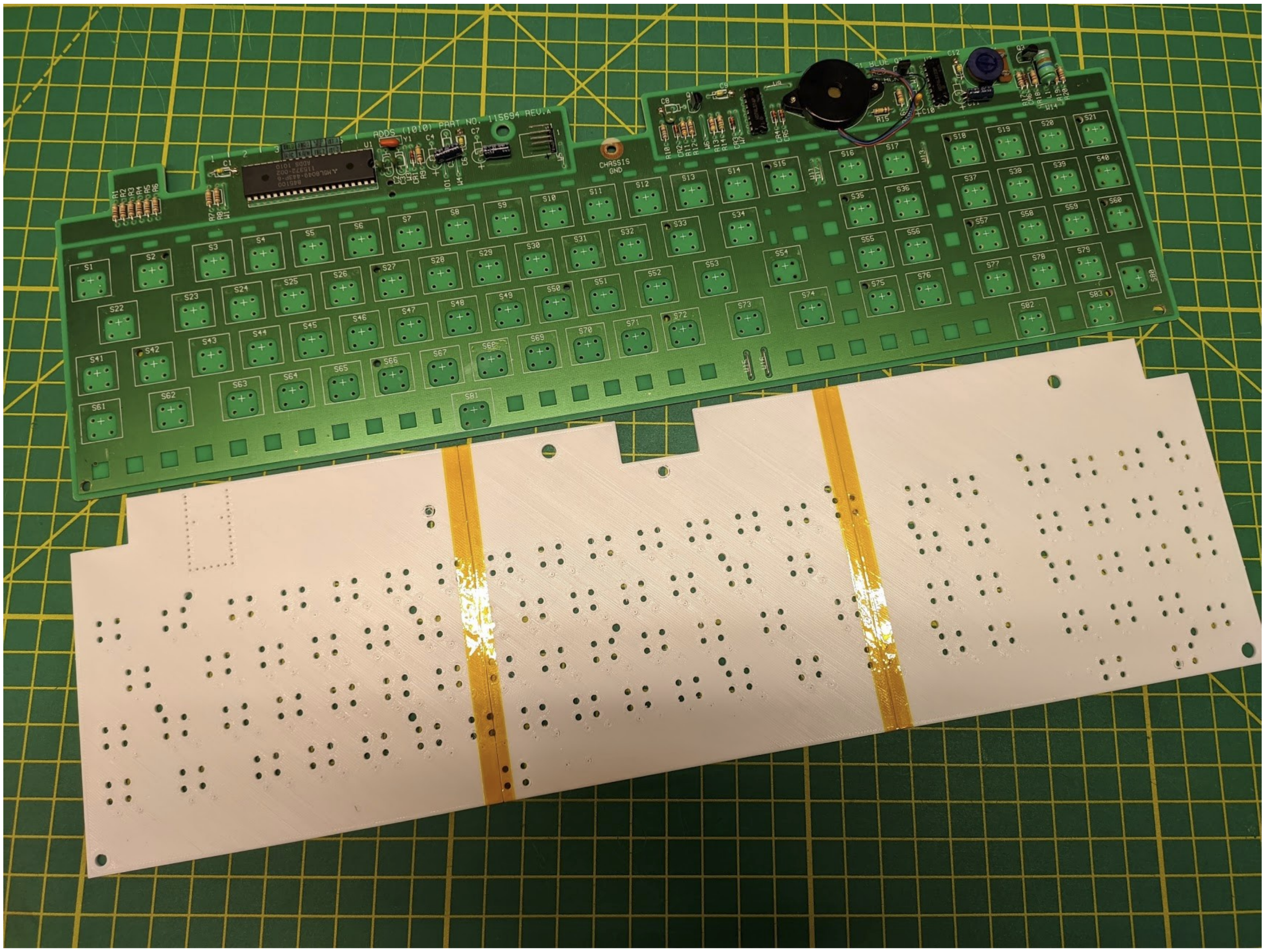
Tada! That’s hopefully the completed layout. You can pop into the 3D viewer to get a glimpse of it.



Double checking

It's a good idea to double check the positions of everything before we go through and make all the traces. You can export layers to SVG/PNG and print them on paper. Since I have a couple 3D printers, I might as well use them.

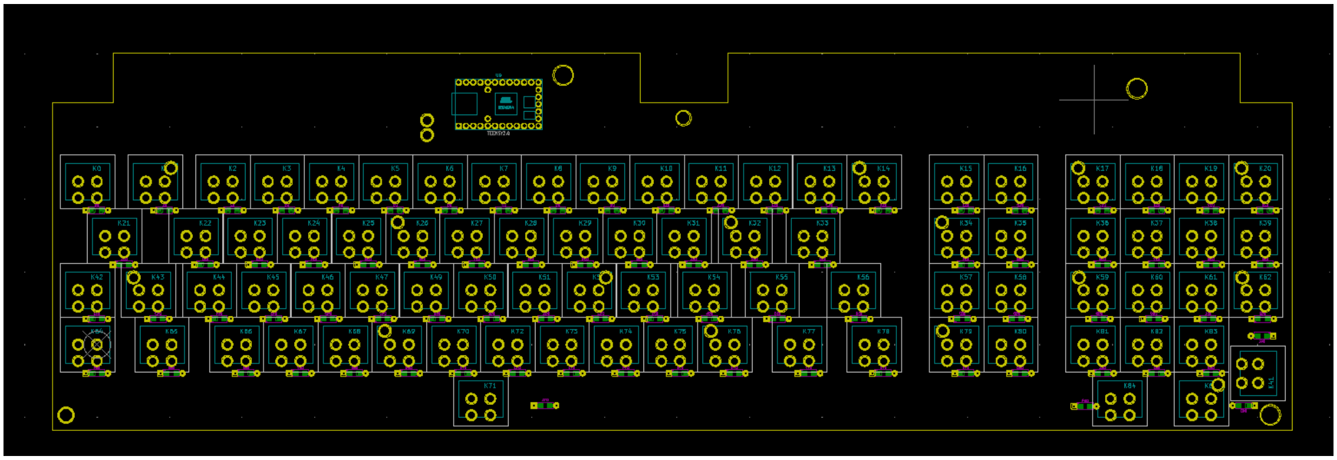
I exported the PCB as a STEP file under File>Export, loaded up the Step file into Autodesk Inventor, exported it as STL in 3 pieces so it would fit on my printer flat, threw the STLs at Cura, and printed. Those segments got taped together and I have a decent sanity checking tool.



Everything fit pretty well! All the holes lined up OK and it fit onto the switches/plate just fine. Going back to my switch-to-switch estimation of somewhere between 19-19.05, the holes furthest away from the origin were indeed a bit tight (still fit though). I'm going to just bump over the arrow cluster by 0.4mm and the numpad by 0.6mm to get it a bit closer. Since the holes are decently oversized it fit fine even without this.

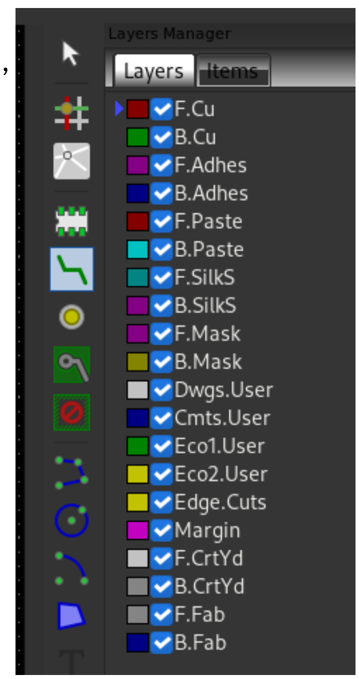
I'm also not too happy with the Teensy location now that I see it so I'm going to end up going with the extension cable route.

With those changes made, this is my final positioning:



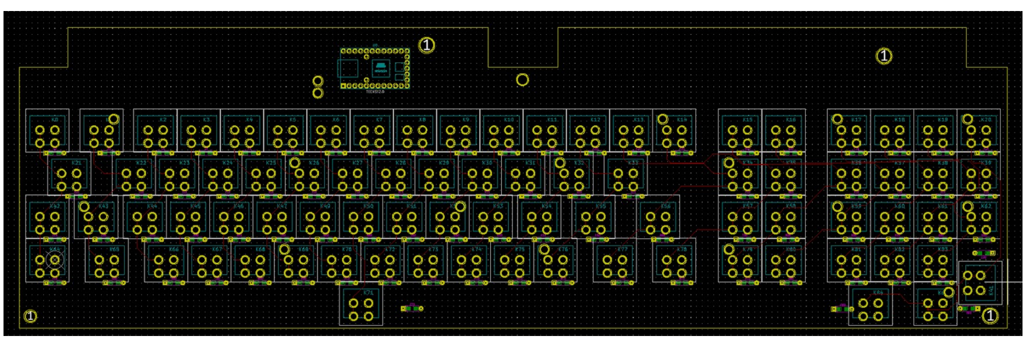
Highlight everything (might need to hide the Edge.Cuts layer), right click, and lock.

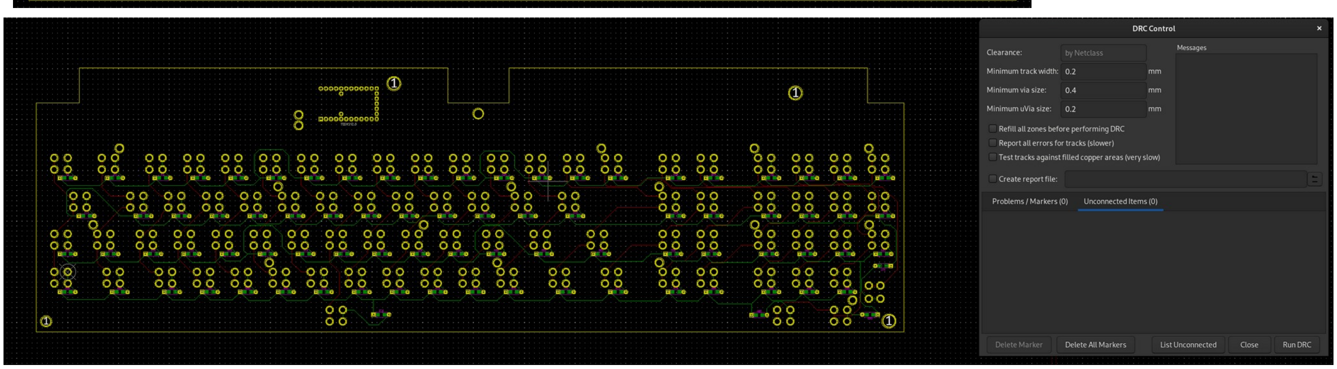
Traces



Now that we know nothing is going to be moving around it's time to start adding traces. Select F.Cu for the front of the PCB or B.Cu for the back. Use the trace tool, click a pad to highlight where it can connect, and get started. In general, try to space things out as much as you can. Give screw holes a wide berth since they might nick nearby traces if they don't have nylon washers.

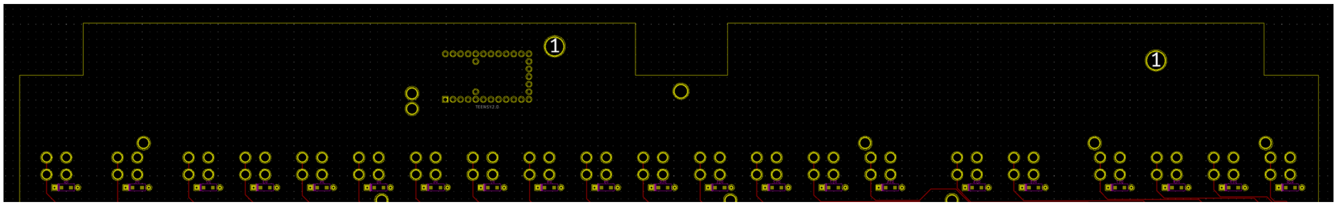
Don't worry about connecting to the Teensy just yet, focus on connecting all the columns together. Once that's done, change layers and start connecting all the diodes and rows.

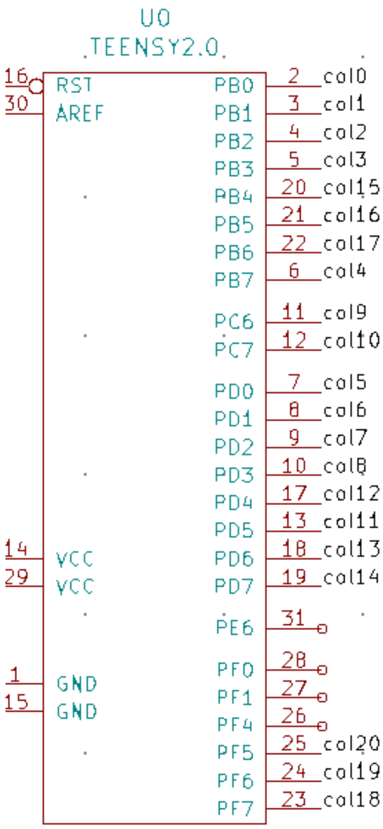




Once everything is connected, use the Design Rules Check tool to make sure you didn't miss anything.

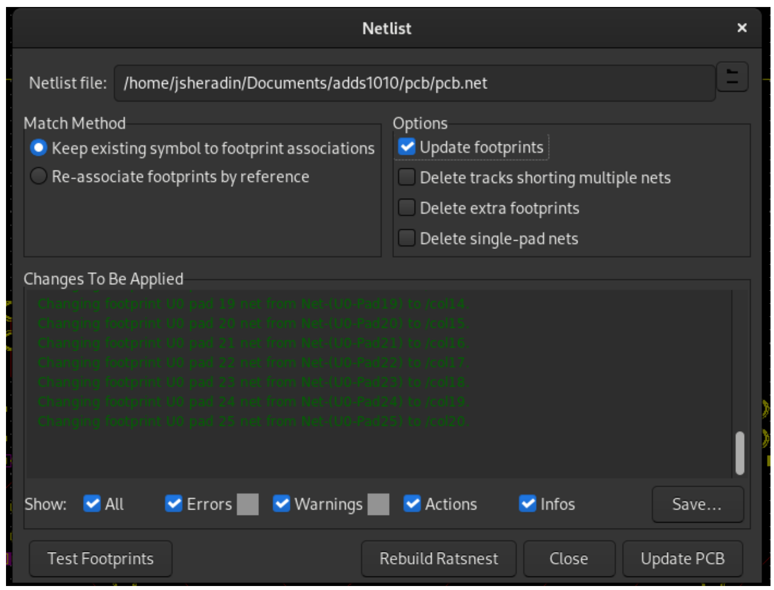
Now take a look at how things will need to wrap around the Teensy to avoid crossing. Reference the pin labeling on the Teensy since the footprint isn't labelled and the order is slightly random.



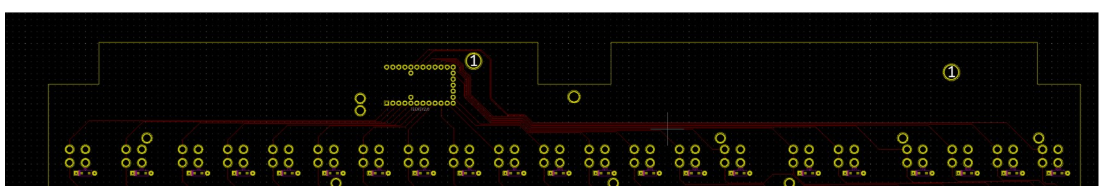


Once you have a good idea of how they need to be routed, head back to the schematic and label all your pins to match the column names.

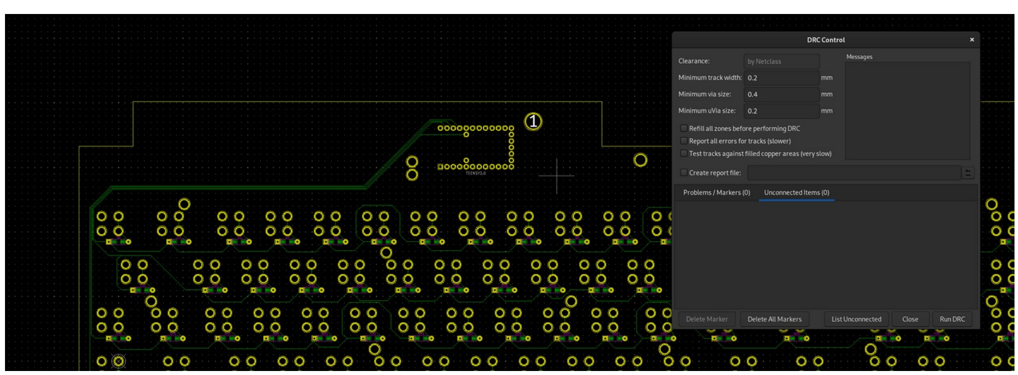
Export the netlist again, and reload it in the PCB editor. Make sure you don't have any of the 'Delete…' options selected since our holes are considered extra.



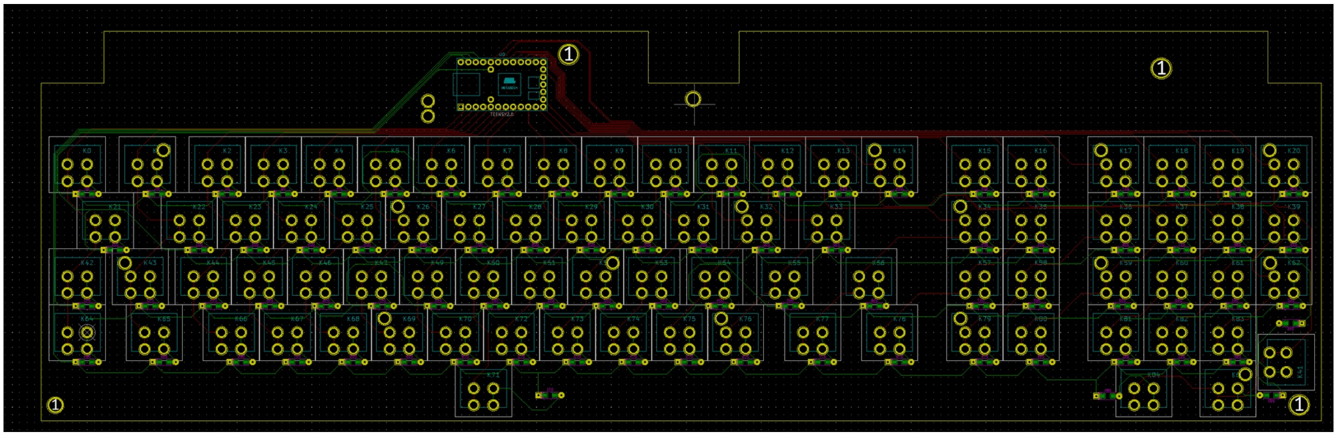
Go back to the PCB editor and connect up all your columns to match.

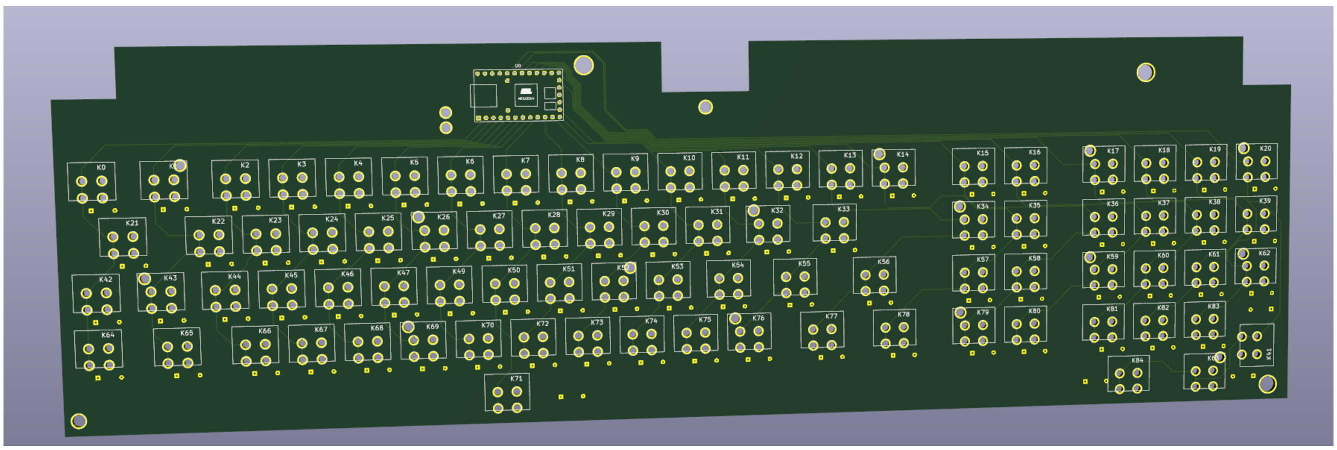


Repeat the process for your rows. Run a final DRC and make sure it's good to go.

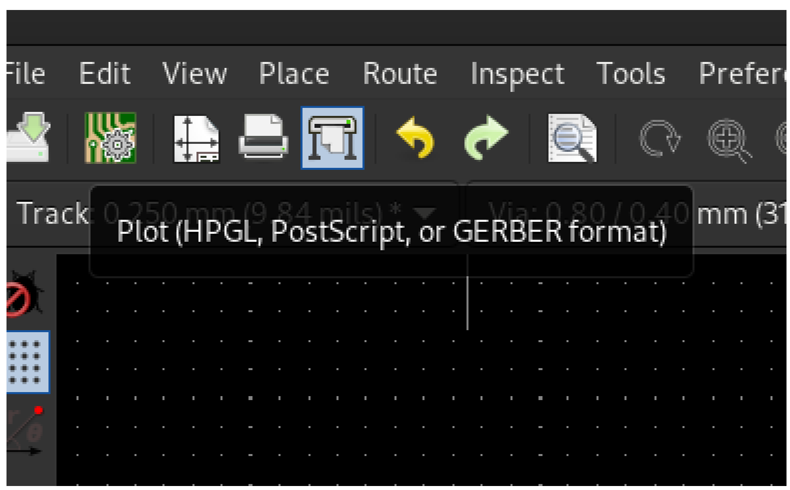


Ordering



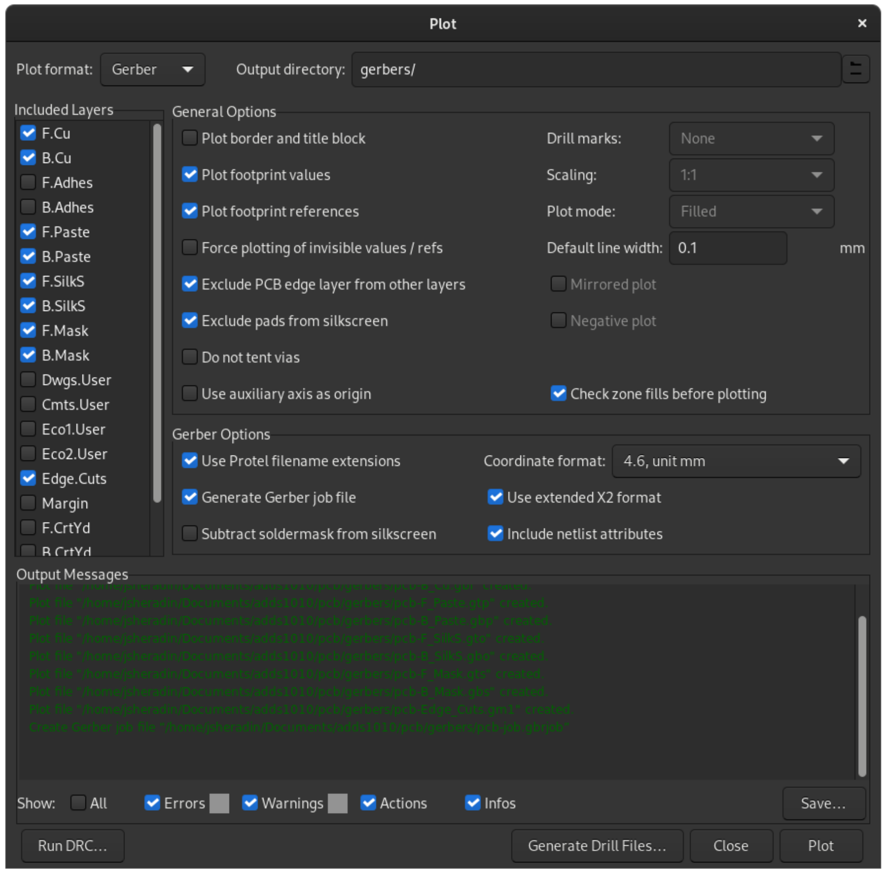


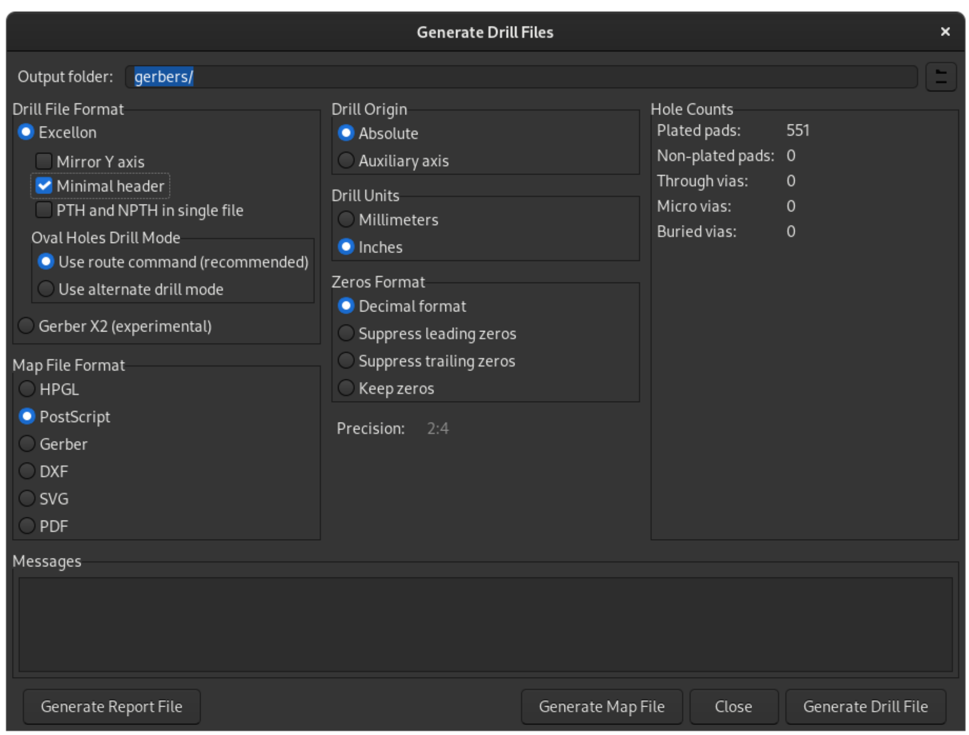
Now that we have our final design, give it a good looking over and make sure there's nothing about it you want to change. If it's all good, click on the plot tool and set your settings to match mine:



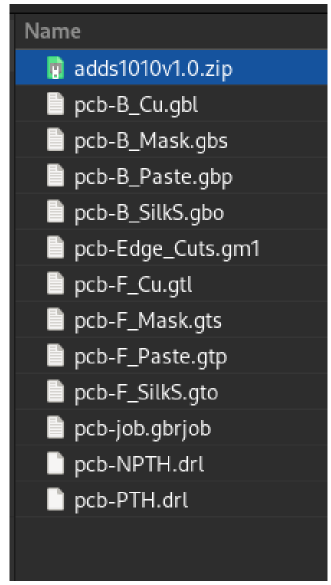
Choose your output directory and click Plot.

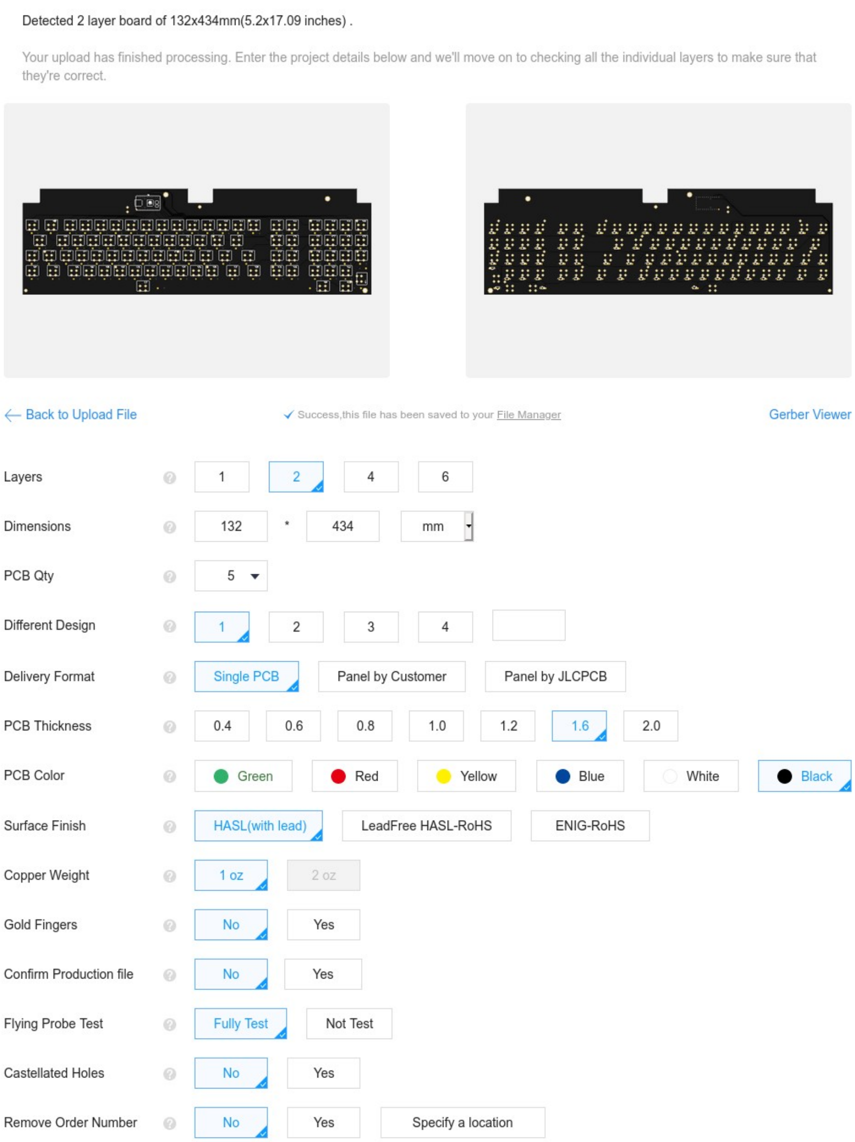
Click on ‘Generate Drill Files’, match my settings, and click ‘Generate drill file’.





Go to the output directory and zip up all the files. Head over to your favorite PCB fab house website (I've had good luck with https://jlcpcb.com/) and upload that zip file containing your Gerbers. Choose whatever color you want and place the order!

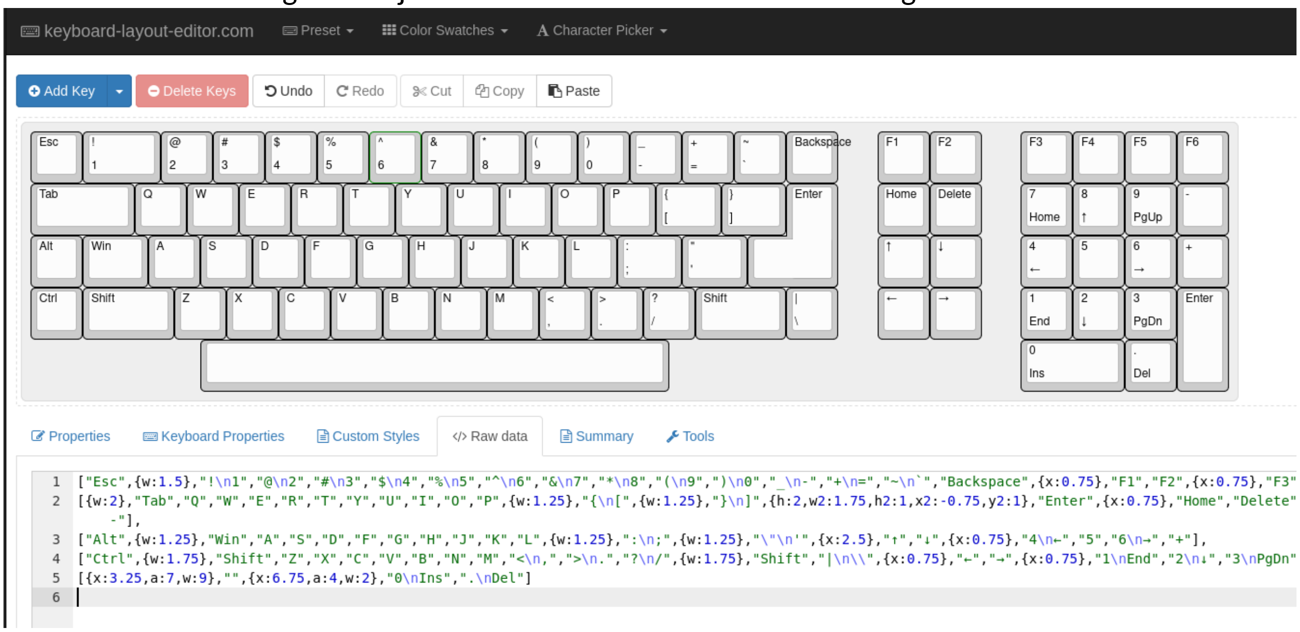




Firmware

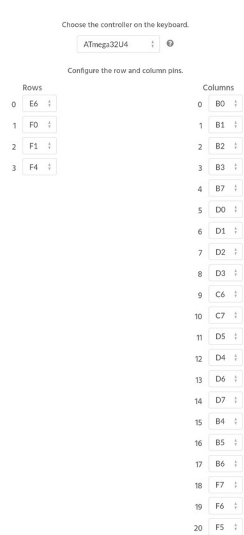
You could flash this with a variety of firmwares. Easiest to set up is probably Soarer's Controller but for this keyboard I'll be flashing it with QMK. Quickest way to start is with http://www.keyboard-layout-editor.com/

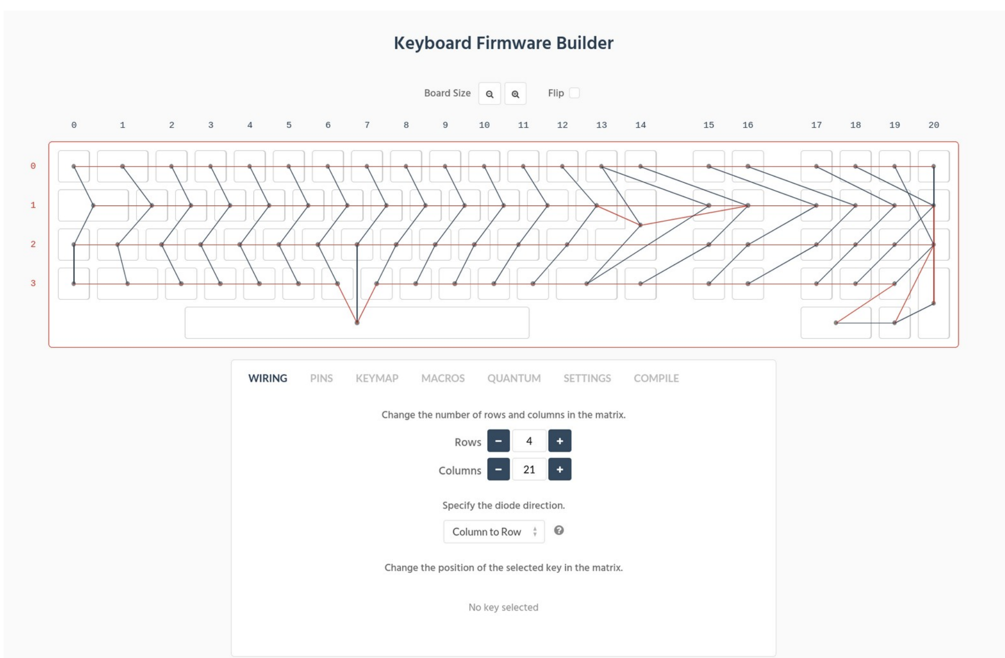
Pick one of the preset layouts and edit it up to match the keyboard. Go to the Raw Data tab and copy the text. Save the configuration json file with the Download button for good measure.



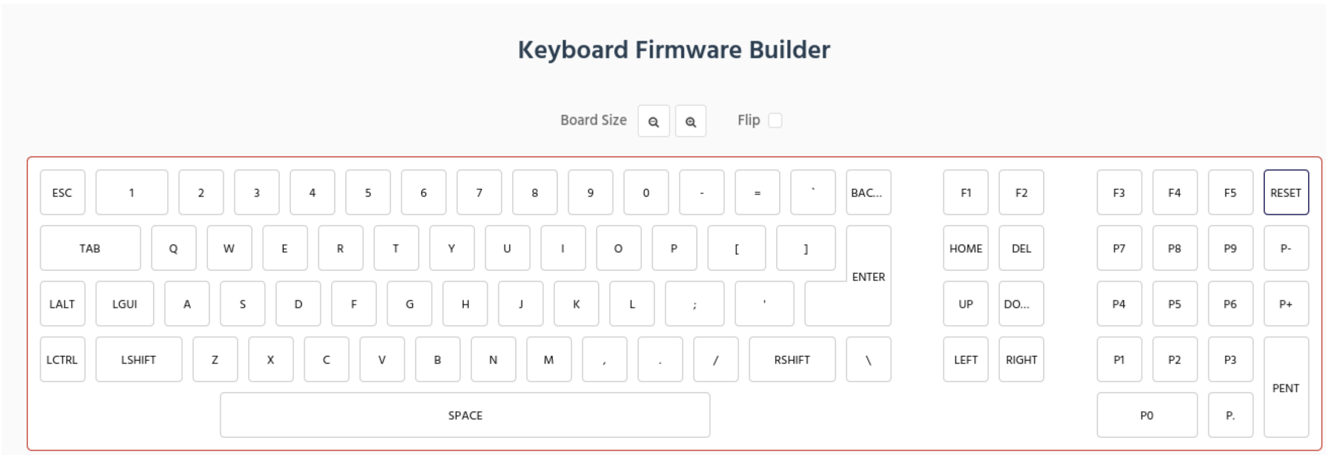
Hop over to https://kbfirmware.com/, paste the text, and click import. In the Wiring tab, configure it to match the matrix you set up.

In the Pins tab, again set them to whatever we set up in the schematic.



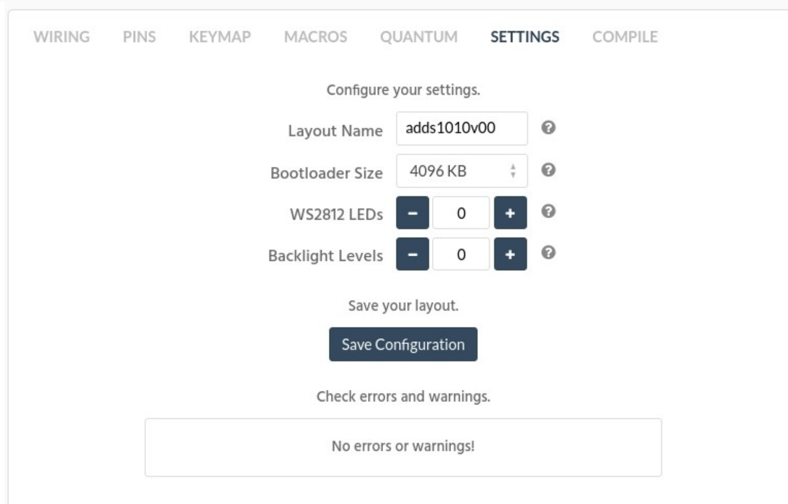


Go to the Keymap tab and set it as you desire (this is just my preliminary map). Make sure you set a key or key-combo as the 'soft reset' key so you can enter bootloader/programming mode without needing to push the physical button on the Teensy.

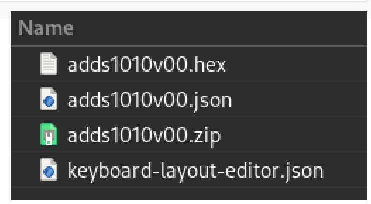


Go to the Settings tab, make sure there's no errors/warnings, name it something descriptive, and save the configuration.

Go to the compile tab and download the hex file. It's a good idea to get the source code too in case you need it later or if this website eventually goes down. In the end, all you need is the hex file but it can't hurt to have the rest.



To program the Teensy for the first time you'll need to push the programming mode button. Use Teensy Loader available here: https://www.pjrc.com/teensy/loader.html Select the .hex file and flash it. Your controller is now good to go.



If you ever want to change the layout, just go back to https://kbfirmware.com/ and upload the json you downloaded (from kbfirmware, not the one from keyboard-layout-editor). Edit it however you want and download the new hex file. Use the key-combo to enter programming mode and flash with Teensy Loader same as before.

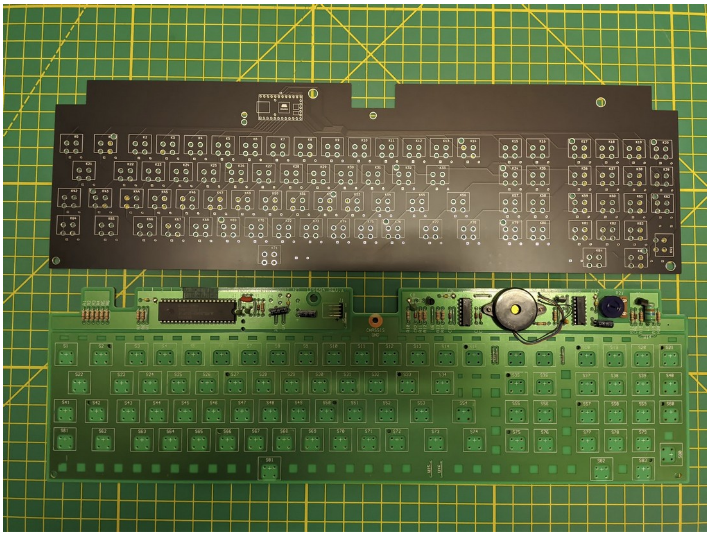
Assembly

While you're waiting on the PCBs to ship, take the time to clean up the rest of the keyboard. I tore all the switches apart and ran everything through an ultrasonic cleaner.

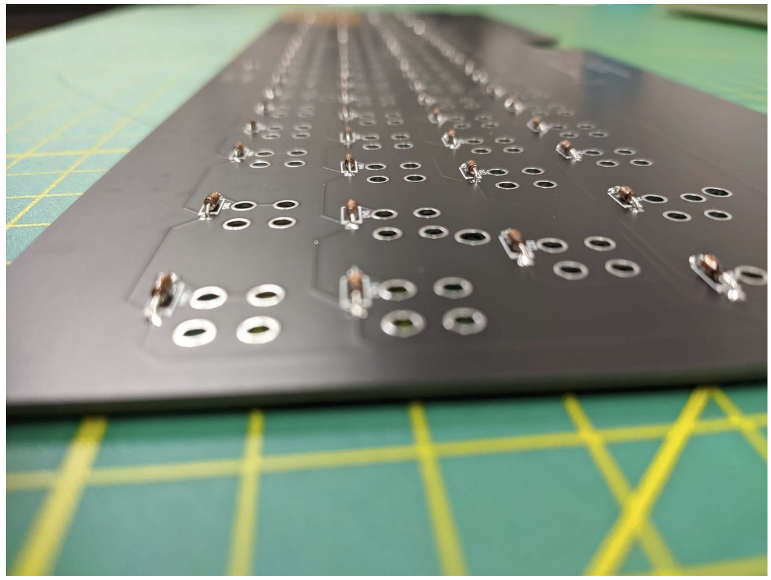
The plate had a little corrosion on it so I sanded it back and hit it with some paint/primer combo.

Reassembled everything with a little 205g0 and checked them individually with a multimeter.

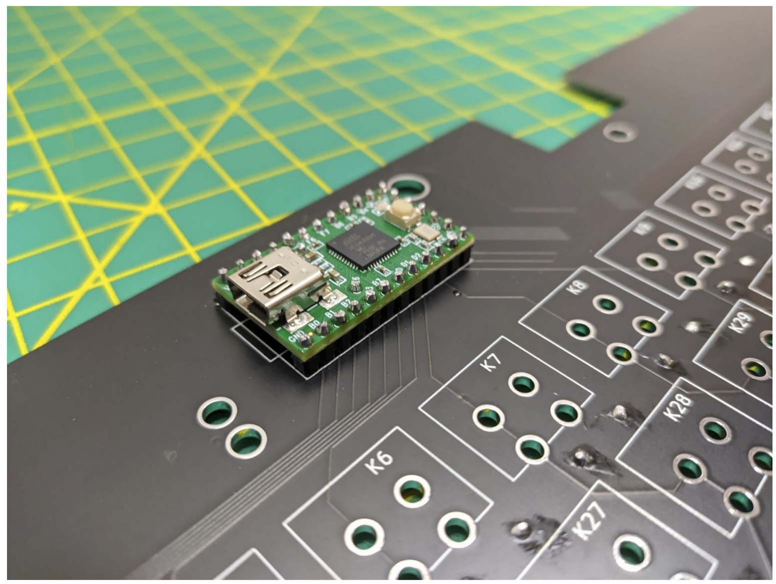
Once the PCB arrives do another sanity check with your fingers crossed since it's too late to do much anyways.



Load it up with diodes (1N4148 switching diodes), make sure to match the line on the diode to the line on the PCB.

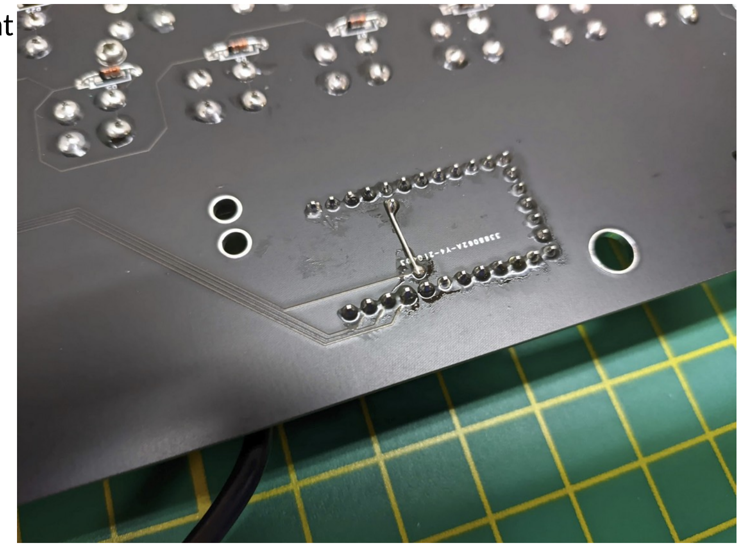


Solder in the Teensy using some pin headers and clip off the excess.



This is my plug solution. The case was messaged a bit to make room for the female end of the cable which was glued to the PCB.

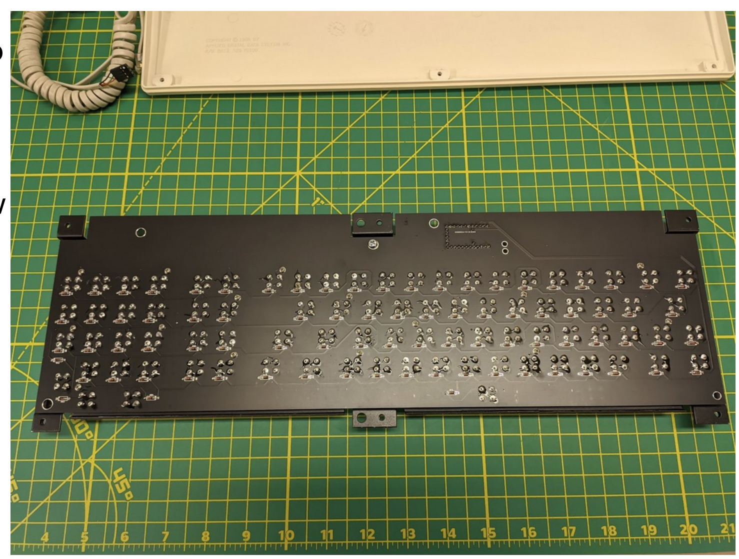
Aaaaand I got burned by an incorrect footprint :c For whatever reason PE6 and AREF pins were swapped so a minor bodge was necessary. My fault for not double checking. **Never trust footprints!**



Although not strictly necessary, I like to desolder the LED on the Teensy for peace of mind that it's not interfering with the matrix scanning. Notice that my bodge required removal of the AREF pin (wasn't doing anything anyways).

Put in the screws that were holding the PCB to the plate if there were any.

Go through and solder in all your switches. This can take a lot of solder depending on how much you padded your hole size.



Give your keyboard a test and then close it all up.

Tada! You now have a fully modern keyboard in terms of functionality with the look and feel of a vintage one.

This document was typed on this keyboard. Hopefully it's of use to somebody.

Thanks for reading,

- jsheradin