PCB Design Workshop - Tufts IEEE Keyboard PCB

Remy Ren

March 2023

1 Before You Attend: DOWNLOAD KiCad 7.0

`https://www.kicad.org/download/`

1. Do this before coming to the workshop as it's a 1 GB download for just the installer. KiCad is neat because it is **OPEN SOURCE AND FREE!**
2. Other PCB design tools exist, such as Altinum, EAGLE, EasyEDA, etc. but KiCad is a great starter.

Optional: Git/Github Primer

Git/Github is a great solution for keeping track of files (version history), storing files, and collaboration.

If enough people want, I'll go over a really basic primer on Git/Github using a command line interface (Terminal, Windows PowerShell, MobaXterm, etc.).

If you want to follow along or do this in your own time, the primer I'll be giving can be found here:

`https://bit.ly/Git-GithubPrimer`

2 Starting Interface

1. Create a new folder in a place of your choice to put your project in. If you used Git/Github, make this new folder within where you cloned your repository.
2. In KiCad, click *Create new blank project* (`CTRL+N`), name it and save it to the folder you just made. We start with the *Schematic Editor* (`CTRL+E`) first.

3. We'll need additional footprint libraries to use keyboard switches and USB-C ports. Either download the ZIP to each repository or add them as submodules in your repository:

   `https://github.com/ai03-2725/MX_Alps_Hybrid.git`
   `https://github.com/ai03-2725/Type-C.pretty.git`
   `https://github.com/ai03-2725/random-keyboard-parts.pretty.git`

4. In *Preferences* → *Manage Symbol Libraries* and add the three libraries by clicking on the folder icon and adding the `.lib` file in the folders (may be nested within *Schematic Folder*). Then, in *Preferences* → *Manage Footprint Libraries*, select all folders which have a `.pretty` at the end of its name.

3 Building the Schematic: Microcontroller Unit

1. We will be using the Atmega32U4 as our microcontroller of choice.
2. To add the component, press (A) or goto the right sidebar and select the op-amp icon. Give the interface some time to load!
3. Search `atmega32u4` and select *ATmega32U4-A* (not *-M* as its pins are on the bottom of the package). Place it on the schematic by clicking within the space.
   (a) The datasheet for this MCU: `http://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf`
   (b) We'll be following this schematic to layout components for the MCU:

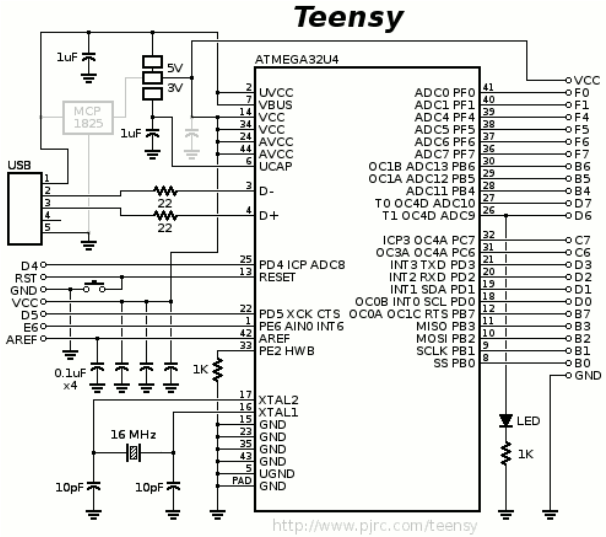


Figure 1: PJRC Teensy Schematic for the ATmega32U4-A

4. Let's first add power/ground to the MCU. Press (P) or select the ground icon on the right sidebar, searching for: `5v`. Select it and place it above the top of the MCU.

5. `UVCC, VCC, AVCC` all need to be connected to this +5V. To place wires and connect them, press (W) or the gray line icon on the right sidebar. Add another +5V component for `VCC`.

6. To place the ground, again press (P), search for `GND` and place it at the bottom of the MCU and cwonnect `UGND, GND` to `GND`.

7. The `HWB/PE2` requires a pull-down resistor (needs to be connected to ground). Hit (A) and search for `R` or `R_Small` (preferred). Add another ground and connect the resistor to it.

   To exit any current action mode, press (ESC). If you want to rotate the component, press (R) while it is selected. To move a component, select and press (M).

8. Double-click on the `R` or `R_Small` to edit its value and set it to **10k** (discrepancy with Teensy schematic).

9. `UCAP` is the voltage regulator for the internal USB pins. It requires a $1\mu F$ (type 1u) capacitor according to the datasheet. Add a `C_Small` to the pin and connect it to a new ground component.

10. `AREF` which is responsible for analog signals doesn't need to be wired, so mouse over the pin and press (Q) to mark it with a cross.

11. Next, we'll work with labels. These allow you to connect components together by name association instead of wires. There are two labels within KiCad: global or net class directive. Using either is OK in this demo, but if you were to use a multi-level/layer PCB, global is the preferred option.

12. To add a global label, press `CTRL+L` or click on the "tag" looking icon with an A inside.

13. Add a global label to `XTAL1` naming it `XTAL1`, then do the same for `XTAL2`. To flip a component, press (X).

14. Add a global label the same way for `D+, D-`.

15. Next, we'll be working to add an external clock for the MCU. Elsewhere on the board, place a global label for `XTAL1, XTAL2`.

16. The circuit for the external clock uses a 16MHz crystal oscillator (search `Crystal_GND24_small` and two 22pF capacitors. Connect them to the global labels like this:

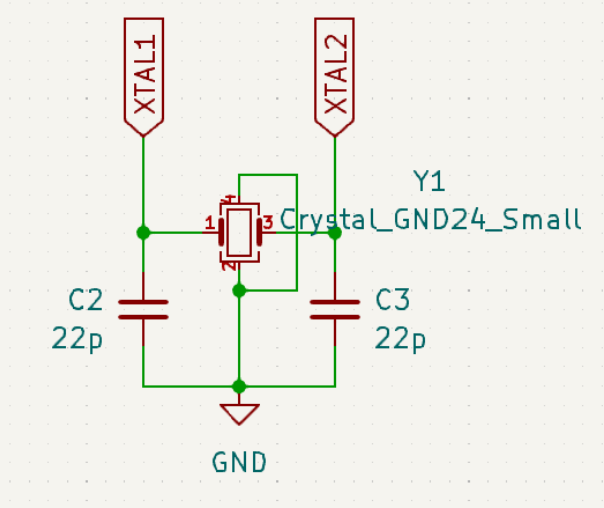


Figure 2: Schematic for XTAL

17. We need to add a set of decoupling capacitors for the MCU. The datasheet recommends a $0.1\mu F$ for each VCC pin (`2xUVCC, 2xVCC, AVCC`). One of the capacitors is shared between `UVCC` and `VBUS` instead of adding an additional one. We can make our schematic neater by using a +5V component to connect the capacitors like in the following:

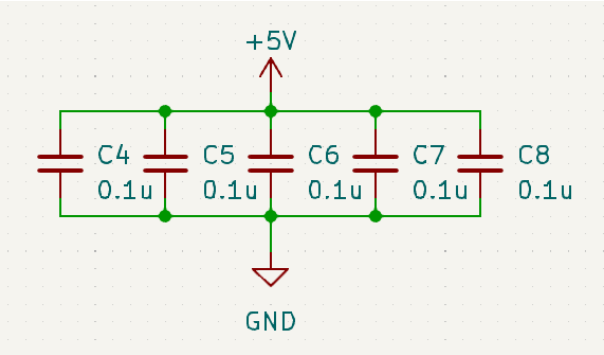


Figure 3: Schematic for Coupling Capacitors

Press (CTRL+C) and (CTRL+V) to copy/paste a component instead of entering it multiple times.

18. We will add a set of ISP header pins to access the MCU in event of catastrophic failure. This will allow us to reprogram the MCU with the Serial Peripheral Interface (SPI) and the ISP programming mode.

    (a) First, add a global label like we did earlier for `MISO, MOSI, SCK`. Connect them like this to the MCU:

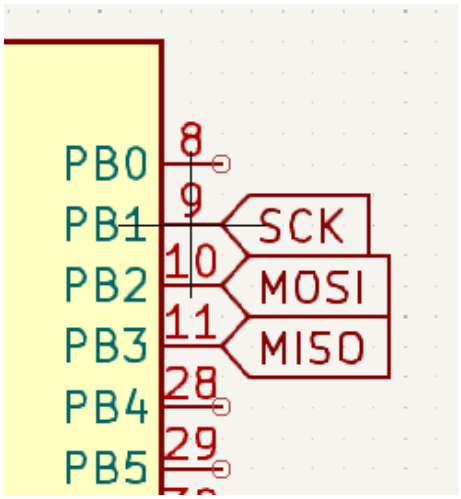


Figure 4: Global Labels for ISP to MCU

    (b) Add a `AVR-ISP-6` to the schematic. Connect a +5V to `VCC`, `GND` to GND. Create global labels for `MISO, MOSI, SCK, RESET` and connect them to their corresponding pins on the header.

19. Add a reset circuit to the schematic like so (`SW_Push` and 10k res.):

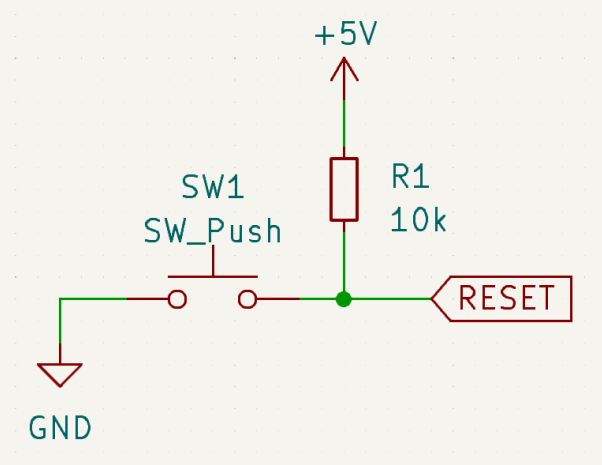


Figure 5: Basic Reset Schematic

Place a global label for reset likewise on the MCU.

20. Add a USB connector like so:

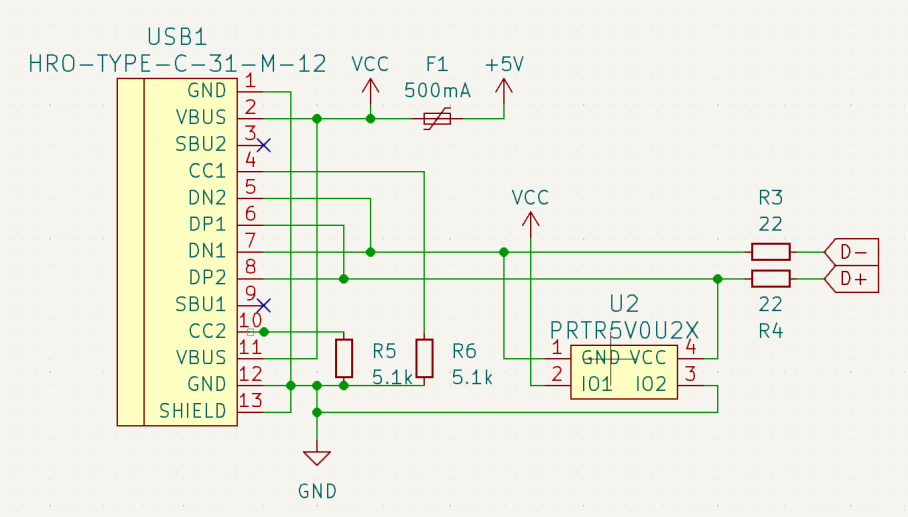


Figure 6: USB Connector Schematic

Note that `F1` is a `Polyfuse_Small` and the `PRTR5V0U2X` (ESD protection) is from aio3's submodules, not the one from KiCad's default libraries.

For more details as to why these components were chosen, see: Masterzen's Guide Pt. 1

21. Congratulations! You're now done with the MCU! You should have a schematic like this:

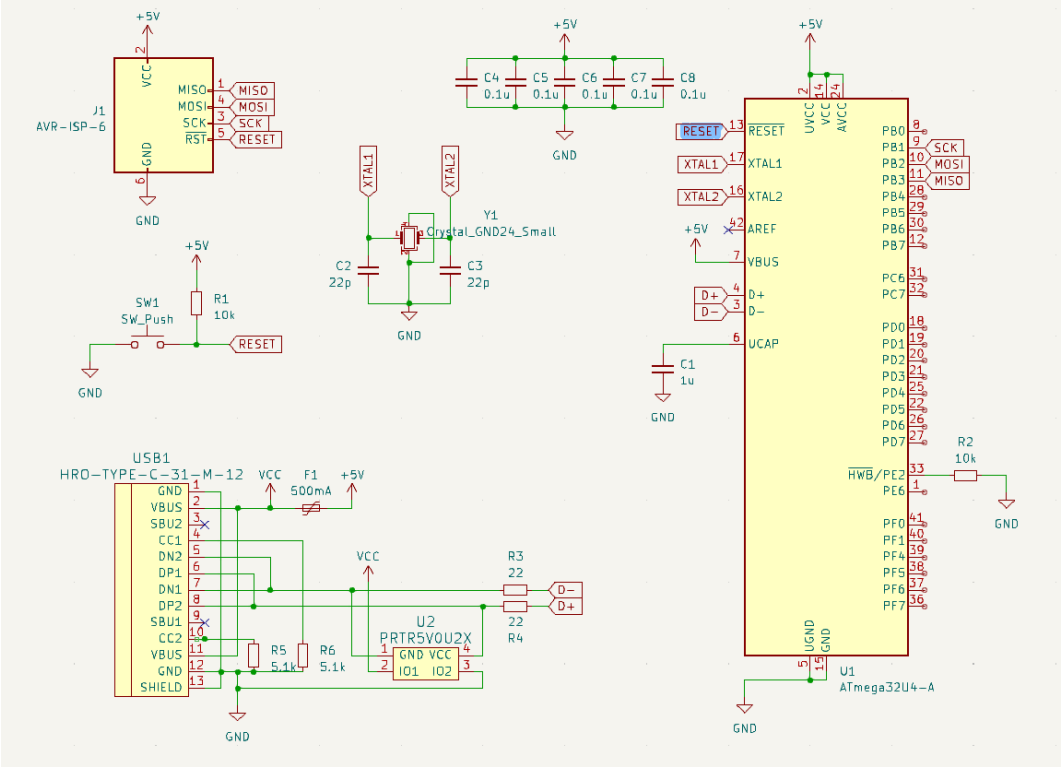


Figure 7: Current MCU Schematic

Whew! That was a lot! But we're not done just yet!

4 Building the Schematic: Switch Matrices

For this workshop, we'll just be constructing a simple 2x2 macropad. If you're looking to create an entire keyboard, the process is not very different, just a bit more time consuming. If you wish to create an entire keyboard, follow: Masterzen's Guide Pt. 2

1. To make a basic cell of the matrix, create this circuit. We will not be assigning values yet:

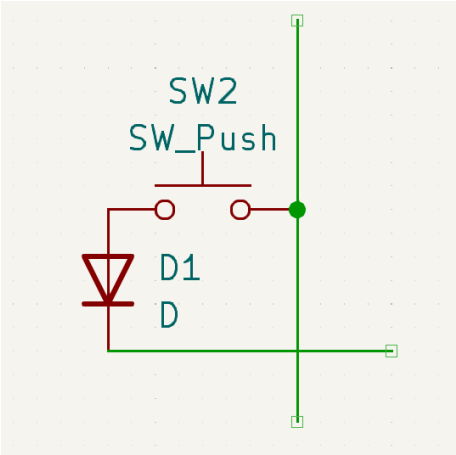


Figure 8: Keyboard Matrix Cell

To terminate wires without connecting it to a component, press (K).

2. Now copy the cell and paste into an arrangement like this:

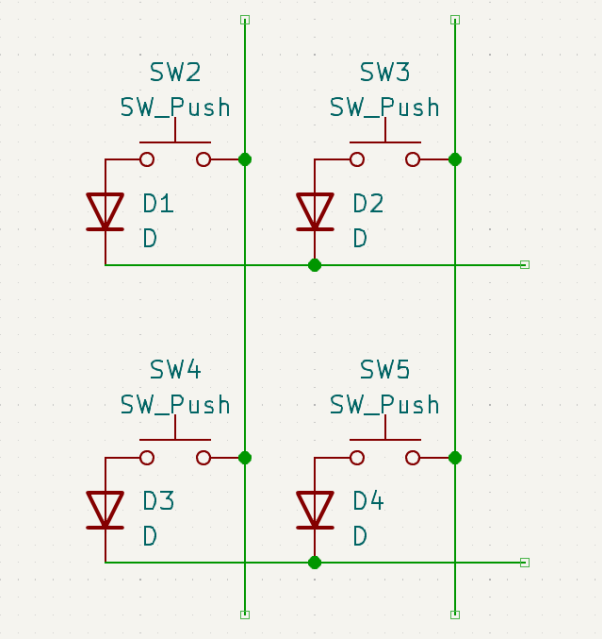


Figure 9: Keyboard Matrix

3. Add the global labels `col0, col1, row0, row1` to the matrix as shown.

You will need to delete the extraneous wires:

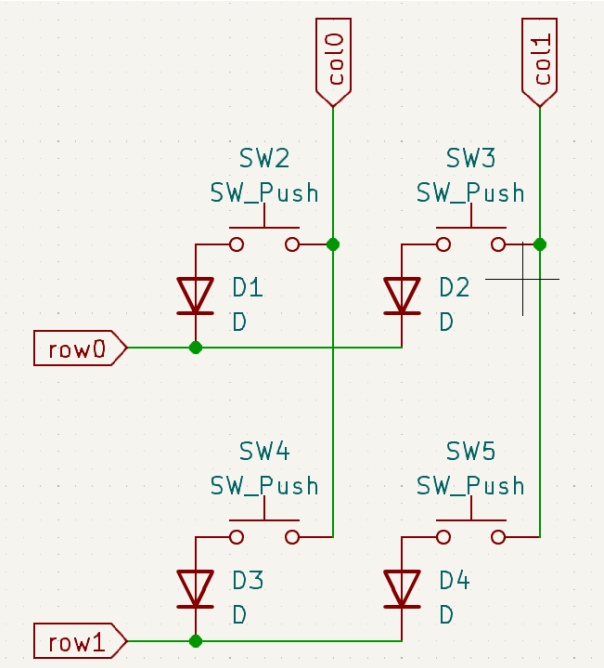


Figure 10: Keyboard Matrix Labeled

4. Attach corresponding global labels `col0, col1, row0, row1` to the MCU pins `PB4, PB5, PB6, PD7`. Note, the last one is at `PD7`. For all the other unused pins, you can mark them with a cross via (Q). Super congrats! You're now done with the schematic... but there's more afterwards!

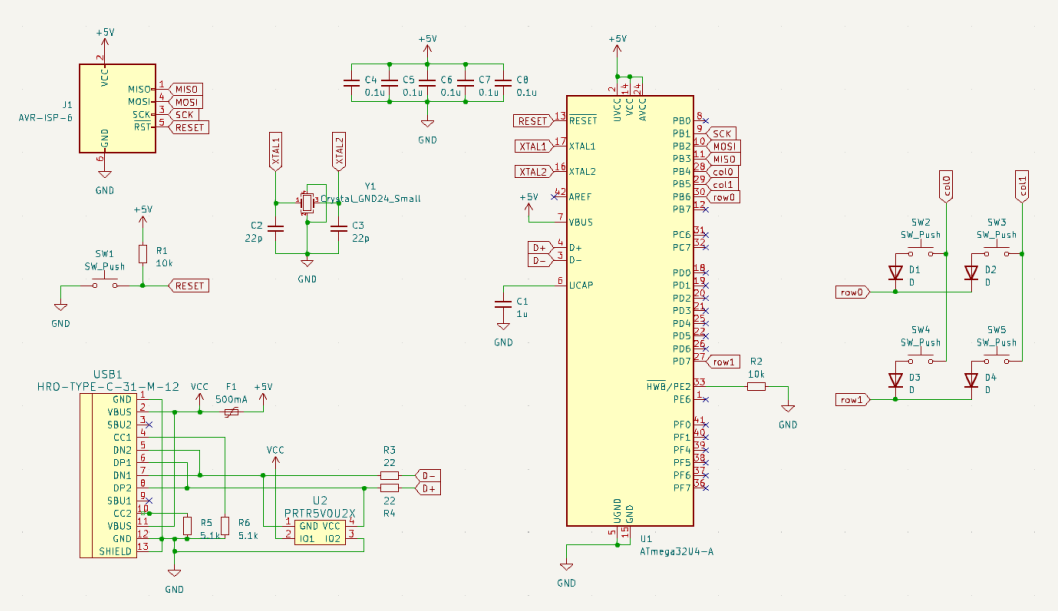
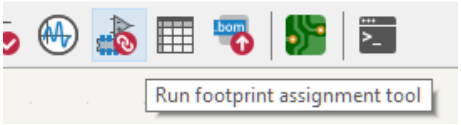


Figure 11: Completed Keyboard Schematic

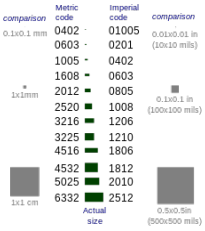
5 Assigning Footprints

KiCad now knows about the circuit you laid down! However, to turn this into a PCB, it doesn't yet know how to translate the components into their actual physical counterparts. To do this, we need to assign the footprints.

1. To start assigning footprints, use *Run footprint assignment tool* in either *Tools* or here on the hotbar:



For context, most of the text in footprint names indicate dimensions:



2. Assign the footprints according to this table:

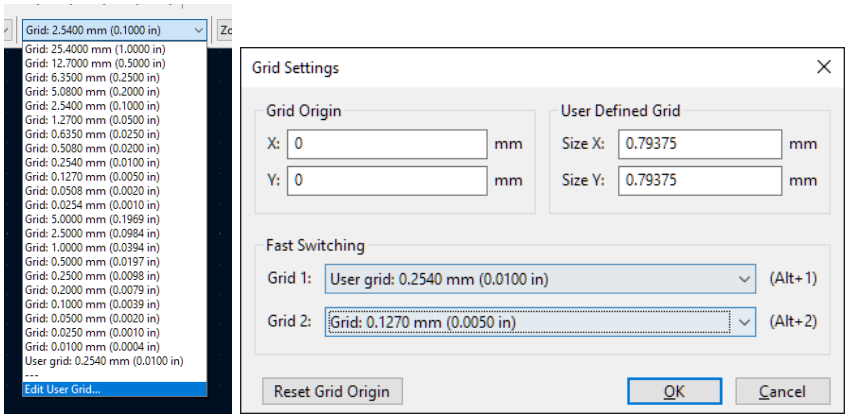
Components	Footprint
Capacitors	`Capacitor_SMD:C_0805_2012Metric_Pad1.15x1.40mm_HandSolder`
Diodes	`Diode_SMD:D_SOD-123`
Fuse	`Fuse:Fuse_1206_3216Metric_Pad1.42x1.75mm_HandSolder`
ISP Header	`random-keyboard-parts:Reset_Pretty-Mask`
Resistors	`Resistor_SMD:R_0805_2012Metric_Pad1.20x1.40mm_HandSolder`
Reset Push Switch (`SW1`)	`Button_Switch_SMD:SW_SPST_SKQG_WithStem`
Keyboard Switches (`SW...`)	`MX_Alps_Hybrid:MX-1U-NoLED`
`ATmega32U4-A`	`Package_QFP:TQFP-44_10x10mm_P0.8mm`
`PRTR5V0U2X`	`random-keyboard-parts:SOT143B`
USB Type-C Connector (`HRO...`)	`Type-C:HRO-TYPE-C-31-M-12-HandSoldering`
Crystal Resonator	`Crystal:Crystal_SMD_3225-4Pin_3.2x2.5mm_HandSoldering`

Once done, press *Apply, Save Schematic & Continue*. If you haven't already, save your project!

6 Creating the PCB: Arrangements

Navigate to the *PCB Editor* (CTRL+P) in the main menu window. Now, press *Update PCB with changes made to schematic* (F8). This will spit everything out in the PCB schematic in one big cluster. To create the PCB, we'll first make various arrangements.

1. We'll want to set custom grid snapping to make placing components easy (Size X/Y: 0.79375mm, Grid 1: User grid, Grid 2: 0.1270 mm (0.0050 in)



2. While in the PCB Editor, some hotkeys function may change: (M) move, (R) rotate, (X) create traces, (V) create a via ("Through-hole", (CTRL+F) find comp.

3. The blue lines aka “rats nest” represents all traces from the schematic. To make the actual traces, press (X). If a trace can’t connect, it’s likely due to KiCad trying to avoid an short.

4. By default, KiCad has two layer PCBs, where there’s a front (F.Cu) and back copper (B.Cu) layer. Masterzen’s Guide Pt.2 contains a concise summary of these layers:

Components	Type	Usage
F./B.Cu	Technical Pair	Denotes copper trace layers
F./B.Adhesive	Tech. Pair	Denotes adhesive layers to hold SMDs to PCB before soldering
F./B.Paste	Tech. Pair	Defines mask to place solder for reflow soldering
F./B.Silkscreen	Tech. Pair	Denotes where components are drawn
F./B.Mask	Tech. Pair	Defines areas w/no solder mask e.g. component pads
F./B.Courtyard	Tech. Pair	Shows how much physical space a component takes
F./B.Fab	Tech. Pair	Documentation for manufacturing
Edge.Cuts	Independent	Defines a safe margin for PCB edge cuts
User.Drawings/Comments	Optional	Can be used for instructions, drawings. Some footprints may u
User.Eco1/2	Optional	User specific, never used in footprints

5. Next, drag and select all the SMD component footprints to the back copper layer. To do this, press (F) after selecting them.

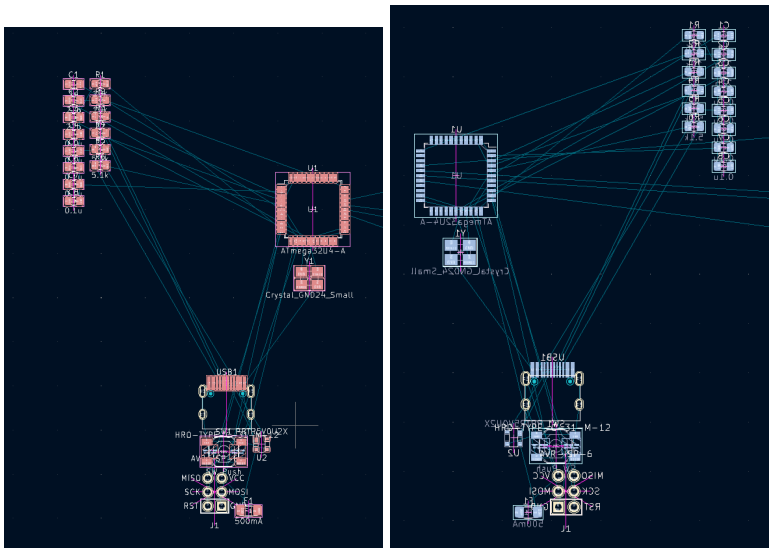


Figure 12: flip

6. While in Grid 1 (ALT+1), move the keyboard switches in this arrangement (overlapping outlines):

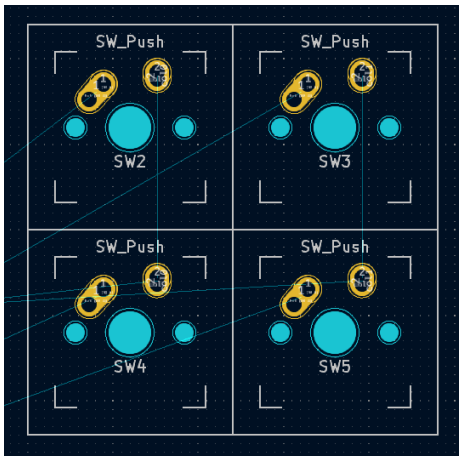


Figure 13: Arranging Switches

Make sure the ordering matches the schematic's (SW2, SW3, SW4, SW5).

7. While still in Grid 1, move the diodes to these positions:

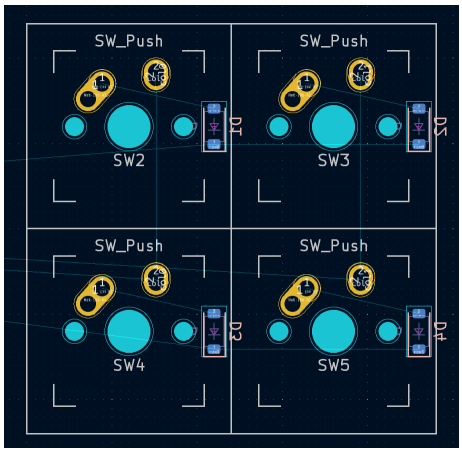


Figure 14: Note the order of the diodes follow the schematic (flipped).

8. If you zoom into components enough, you can also see which pins or labels they connect to. Move the capacitors for the MCU so that they are close to the pins they connect to and the rat traces are neater. Note, these are tentative layouts. When routing (placing traces), components may need to be moved to avoid conflicts.

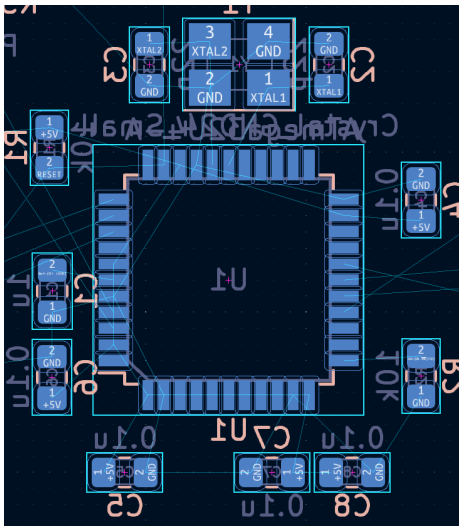


Figure 15: Components do not have to exactly match this arrangement.

9. For now, not all the components have to be arranged. We'll draw out the edge cut out line next. Select Edge.Cuts on the right sidebar.

10. First, round one of the corners of the switches using the arc tool on the right sidebar (CTRL+SHIFT+A):

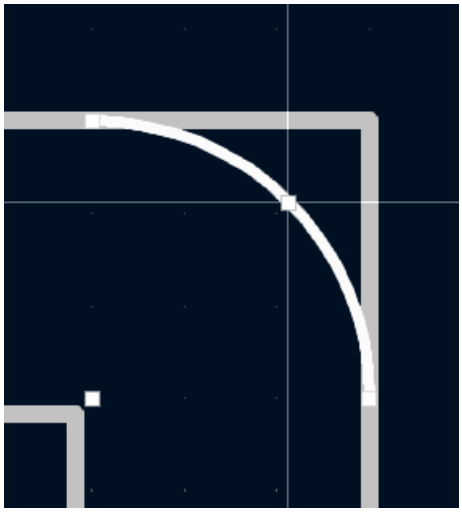


Figure 16: Note the order of the diodes follow the schematic (flipped).

Create the rest of the lines using straight lines. You can copy the lines

and overlay them for a longer edge. If you have trouble seeing the outline from Edge.Cuts, you may have to recolor the layer. This can be done by double-clicking the color, where it might prompt you to create a new theme. In preferences, go to *PCB Editor* → *Colors*, create a new theme and edit Edge.Cuts to a color you like.

You can also hide the User.Drawings layer to get a better view. Make sure that your lines in Edge.Cuts are all properly aligned and make a shape with no gaps.

11. Afterwards, position the remaining components within the area defined by Edge.Cuts. Try to keep the components related to the USB-C connector parallel as we will have to route a differential pair later. You should end up with a result roughly like this:

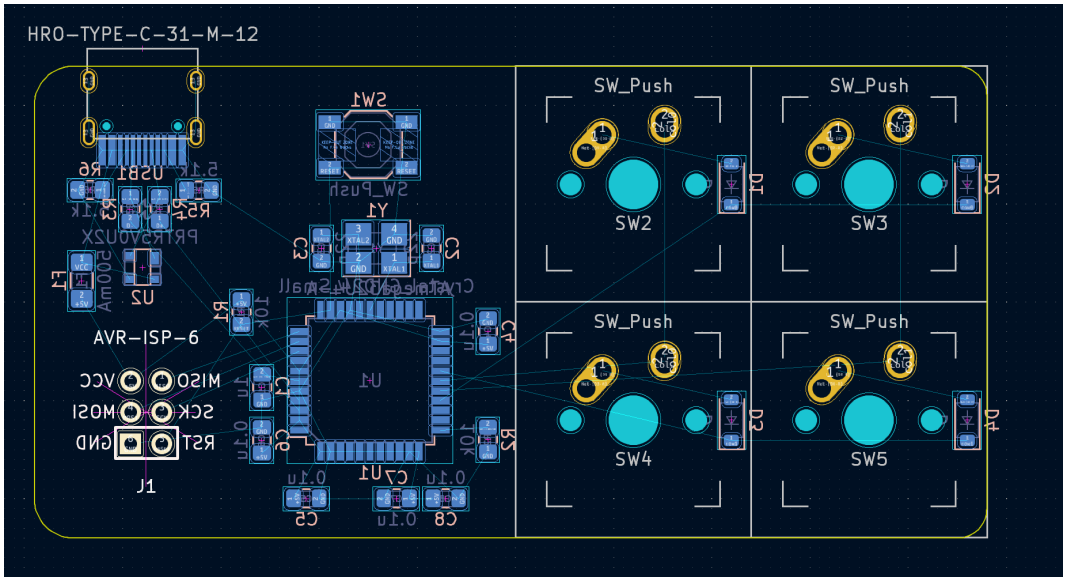


Figure 17: Tentative layout, may change in routing.

12. At this stage, we can take a good look at our progress, but now in 3D! Go to *View* → *3D Viewer* (ALT+3) to see the model.

13. You may have noticed that the USB-C connector only has its pads and doesn't have a model for the actual connector. To get the 3D model, double click the component in the PCB Editor, then the *Footprint Properties* window should appear.

14. Click the tab *3D Models* and click on the "+" symbol.

15. Navigate to where you downloaded ai03's PCB footprints and find the folder "Type-C.pretty" and select the `.step` file. If you have SolidWorks installed, this might show up as a CAD file.

16. Apply the following transformation:
Rotation: X = -90.00°, *Offset:* X = -4.45mm

17. WE'RE ALMOST THERE! Now time for the final stage, ROUTING!

7 Creating the PCB: Routing

Routing is where we'll connect components via traces. The hotkeys of interest are: (X) make a trace, (V) make a via. Vias allow you to connect traces on one layer to the other (front/back) in series by creating a copper-coated hole.

You will want to minimize the number of vias you use as each one will add inductance to the trace. In addition, for the differential pair traces, it's generally recommended to not use vias in their routes.

(Interested in routing fundamentals? See: Masterzen's Guide Pt.3)

1. Make sure you're on one of the copper layers (F/B.Cu) first. You may also want to switch to Grid 2 for routing.

2. First, add these traces to the USB-C connector:

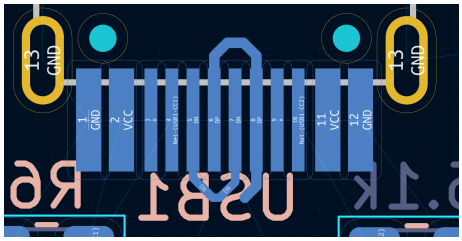


Figure 18: Note that the end of the traces on the left is triangular.

3. A helpful tip, if you press (U) while selecting a trace, then press it a few more times, it will select the entire trace if you need to delete it.

4. Now, we want to route the differential pair. Find this in *Route* our press (6). You might have an error appear about the names of the nets. To fix this, add global labels in the Schematic Editor to the data lines:

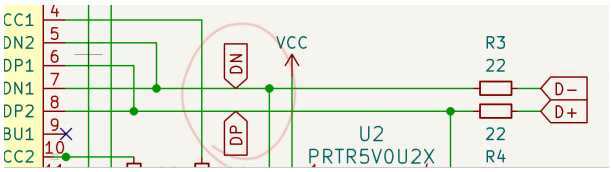


Figure 19: Placing DP, DN on the Schematic Editor

5. Make sure to update the PCB from the schematic afterwards (F8).

6. There are a couple of rules when it comes to routing differential pairs. Due to time constraints we will unfortunately we not be following those rules to a tee (see Masterzen's Guide Pt.3 if you're interested). Do what you can to get the differential pairs to properly route:

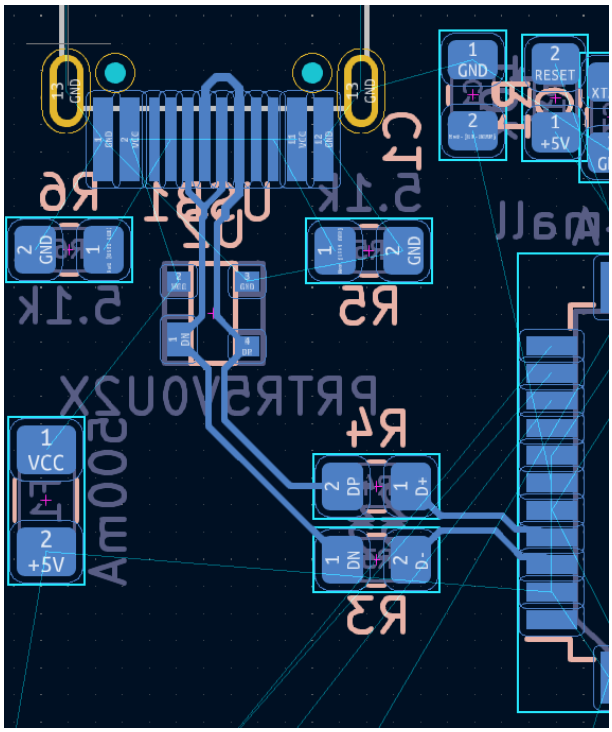


Figure 20: Example of differential pair routing. Note the attempt at symmetry.

7. Next, route the switches. Unfortunately, the default library in KiCad does not indicate which way the `SW_Push` opens or closes. If you cannot route the vias (through-holes) from the switches to the diodes, you may need to flip its corresponding `SW_Push` in the schematic editor.

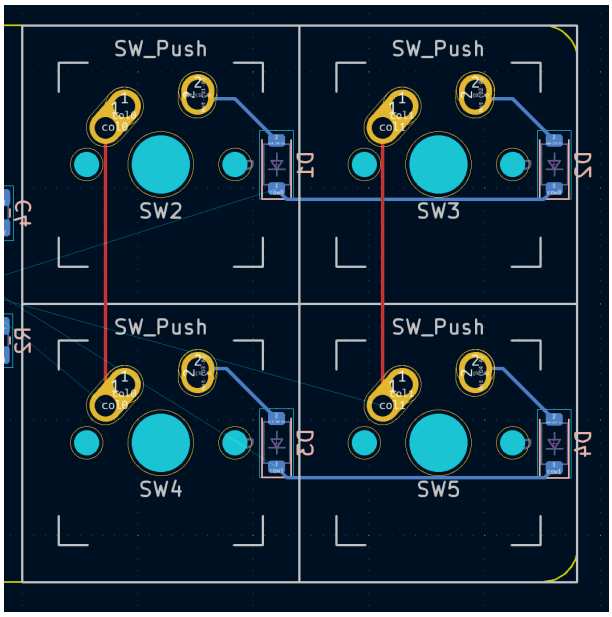


Figure 21: Switch Matrix Routing

8. Note that routing is will take a lot of time! It's really more like a puzzle so it is a skill that takes practice. Don't worry if you end up with situations that are impossible to continue, just keep trying!

9. Once you think you're done with routing, that everything is connected to each other, click on: *Show the design rules checker window*. Then click on: *Run PRC*

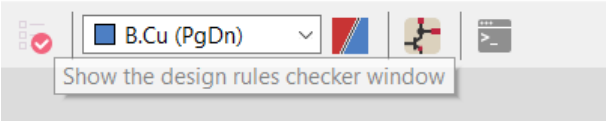


Figure 22: Note the order of the diodes follow the schematic (flipped).

10. Fix any unconnected errors and run the checker again. There are some odd errors with the USB-C tolerances and the vias on the switches. You can ignore those errors for now, but you may need to check/fix these issues if you wish to put this PCB to fabrication.

11. If we run into time issues, you can download my PCB that has the traces finished to move onto the next part.

Note that we are missing some additional components that are used for protecting the circuit and ensuring a stable output. Again, please check out Masterzen's Guide Pt.3 if you are interested.

12. You should end up a result somewhat like this:

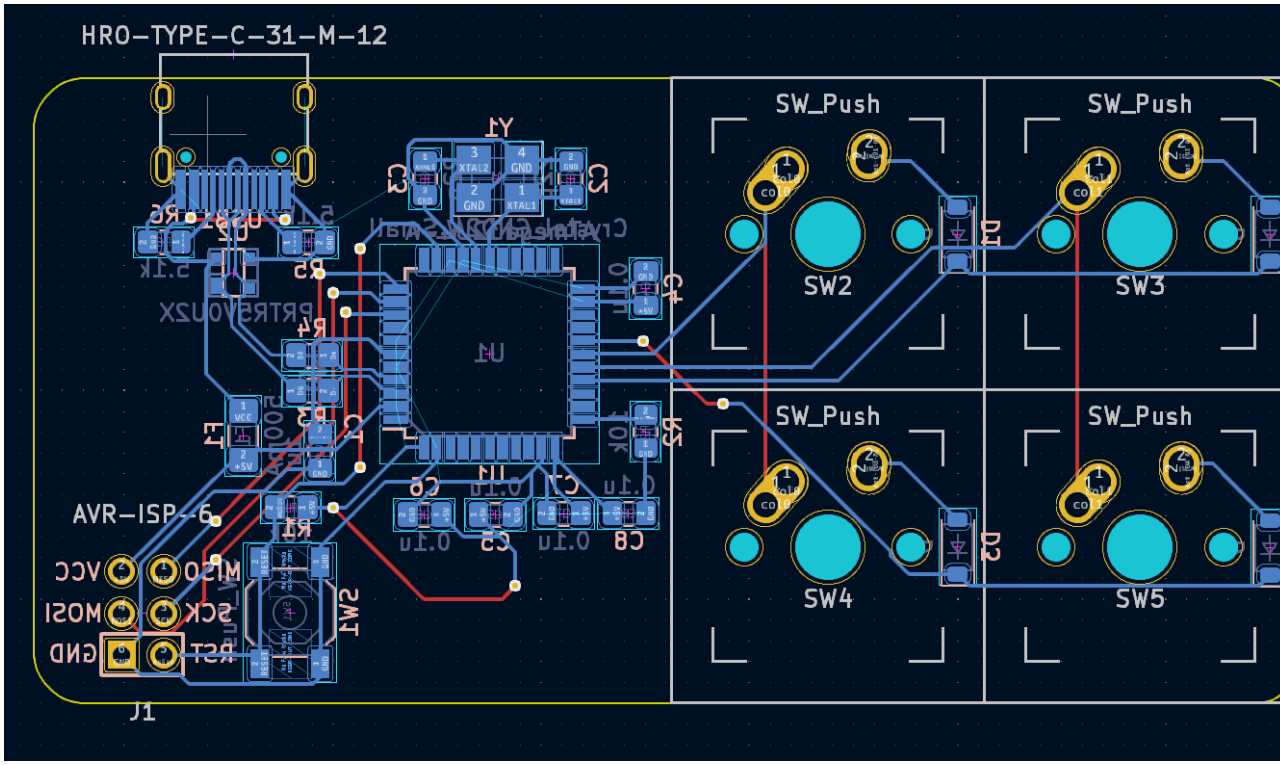


Figure 23: Example Complete Routing (might be with errors).

8 Creating the PCB: Ground Fills and Extras

1. Next, we need to create a ground fill. There are a couple of reasons to do this, especially for high-speed circuits. For actual reasons, again check out Masterzen's Guide Pt.3.
2. To do add a filled zone press (CTRL+SHIFT+Z) or its corresponding icon in the right sidebar ("blue tetris block with a gray line").
3. Grid 1 would probably be the best option to outline the circuit area. Note that this is likely not the best way to make the ground fill for this circuit. In Masterzen's Guide Pt.3, it goes much more into detail about methodologies behind why or how to create a proper ground fill.

4. Make sure that in the new window, “Copper Zone Properties”, that only on B.Cu is selected. Then click on the GND option and click OK.

5. Make a rectangle that outlines the left half of the board then press (B) to fill the entire zone:

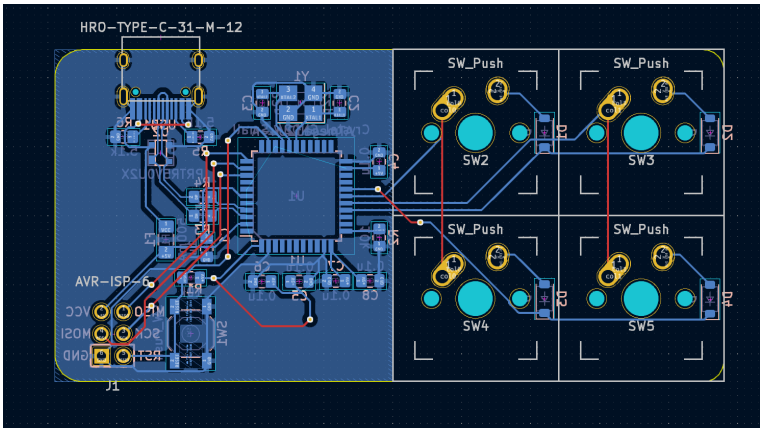


Figure 24: The filled zones are all in a transparent blue (B.Cu)

6. Check if there are any errors with the board by running DRC again. If you run into a thermal relief error, you may just need to add a standalone via someplace on the board.

7. No errors anymore? CONGRATULATIONS! Pat yourself on the back because you have finished creating a PCB! Be sure to iterate and improve the design or retry routing if you are unsatisfied with the result.

9 Final Steps: Gerber Files and Fabrication

1. To be continued!

2. For actual fabrication, I recommend Masterzen’s Guide Pt.4. After fabricating the PCB though, it seems that there hasn’t been much progress on their guide. Note, from the comments, there also may be additional errors such as the confusion between a standalone crystal or crystal oscillator unit.

10 References

This workshop wouldn’t have been possible without these guides!

Masterzen's Guide - Incredible resource with more EE aspects to the design.
aio3's Guide - Great primer, uses some simpler components, but useful nontheless.