Rochester KLUG Keyboard Meetup

Tim Anderson

Tim Anderson

Who am I?

- https://github.com/mcdviii
- http://aberrant.online
- timothy.c.anderson@mykolab.com

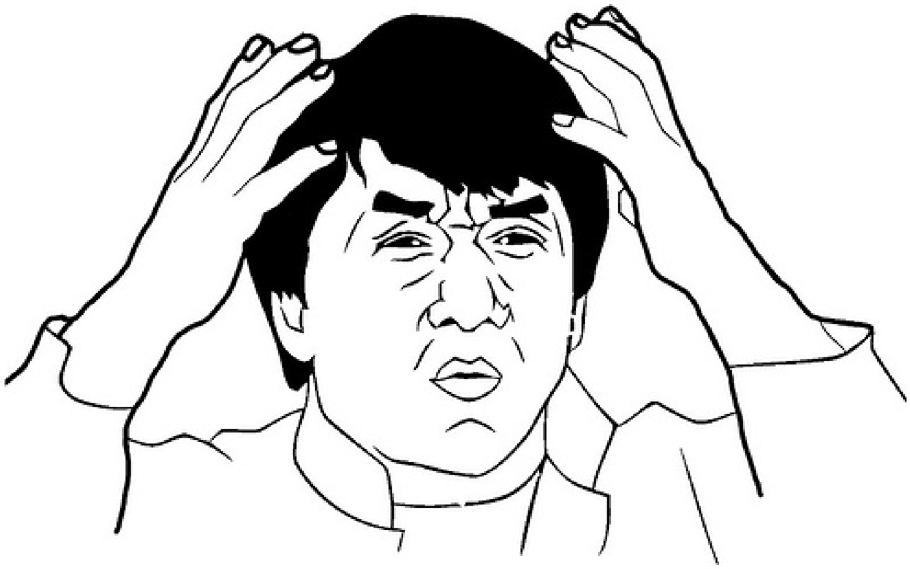
I don't think there's much to tell about myself. I've worked 8 years in manufacturing for a local furniture manufacturer. About 6 of those years I've worked as a machine operator. I own a reasonably sized 3D printer; I tinker with electronics & free software on my free time. You can follow me through the following links. You won't find me on Facebook or Twitter, so don't bother looking for me there.

So, what is this Dactyl Keyboard thing?

- And why should I care?

Parameterized, Split-Hand, Concave, Columnar, Ergonomic Keyboard

- Written in Clojure



Those are a lot of fancy words. What does all of that mean? Well we'll look at the project & see how it compares to a regular keyboard to answer that. But first I'll talk a little bit about the project.

History & Background

yt:uk3A41U0iO4

- https://www.youtube.com/watch?v=uk3A41U0iO4
- Started March 2015
- Made as an attempt at improving on the Kinesis Advantage2? @adereth
- Written in Clojure by using a wrapper arond OpenSCAD

The Dactyl project was originally started in 2015 by a guy named Matt Adereth. I haven't seen what his motivations for starting this were, but I would guess you could find some clues to that on his Twitter page. There are some obvious similarities between this and another keyboard commercially available that I'll show later on, and I think he drew inspiration & motivation from that.

If you're wondering what dactyl means, it's defined by the freeonlinedictionary as: finger, toe or digit.

The really interesting thing about this project is that the mesh was written in Lisp. If you're not familiar with OpenSCAD, it's free software which interprets functions and creates geometry from that. Obviously it uses a lot of math & complex geometry for more complex objects. Matt didn't like the language that OpenSCAD used, but was able to serindipitously find a wrapper to go around the program that would interpret Clojure and convert it into the language that

OpenSCAD understands.

Where is the Project Now?

- Last commit was in 2017
- Left in unfinished state & appears to be abandoned
- Documentation is left unfinished
- OpenSCAD functions don't stay consistant between versions or otherwise don't play nice with lisp wrapper.
- One of the most challenging DIY keyboard builds to take on

So the repository appears to be abandoned since 2017. There are some pull requests left open, and the physical build documentation has been only partly provided by community members. Once you understand the basics of how the electronics work, it's not too complicated, but it reqiures quite a bit of research from dispirate sources.

The documentation for setting up the workflow seems to be fairly complete, but changes between versions of OpenSCAD seem to break significant functionality and make using the workflow with the wrapper difficult. Many of the meshes other than the standard one currently have holes in them because of these issues.

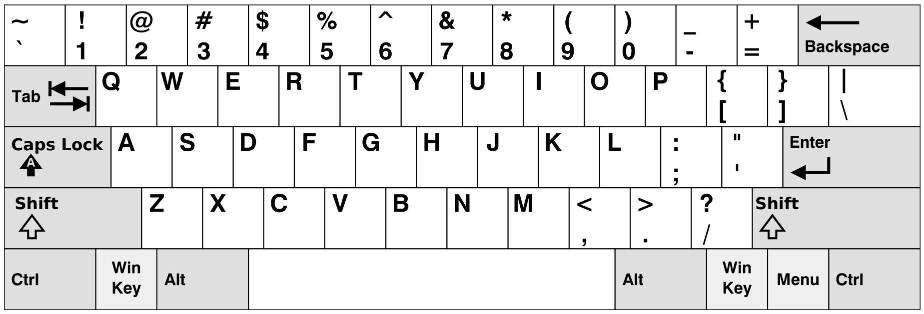
Because of all of these reasons–and the shape of the keyboard & minimal internal clearance for components–in my opinion, this is probably one of the most challenging DIY keyboard projects you could take on.

Why Tho?

- In short, because I can.

During my research into this other ergo keyboards I see a recurring question: "Why would you need/want that.", and I think that it's a fair question. It smacks of iPhone ownership or RGB lighting: Do you really need that? To the outside observer it might look like an expensive, showy & unnecessary gadget. But I think there are real benefits to using a proper ergonomic keyboard, and before I go any further into my build I'd like to show you a couple of examples of why I think ergonomic keyboards should become a standard. I don't experience RSI issues, nor have I ever. I'm just a guy who thought building my own keyboard would be both a fun & educational project, with a very practical benefit of protecting myself from injury in the future. I take the usefulness of my hands pretty seriously & I want to be proactive with protecting myself from any unnecessary risks. I use a computer a lot, and I would like to continue using a computer a lot with as few negative affects as possible. I had the means to build my own badass keyboard, so I did.

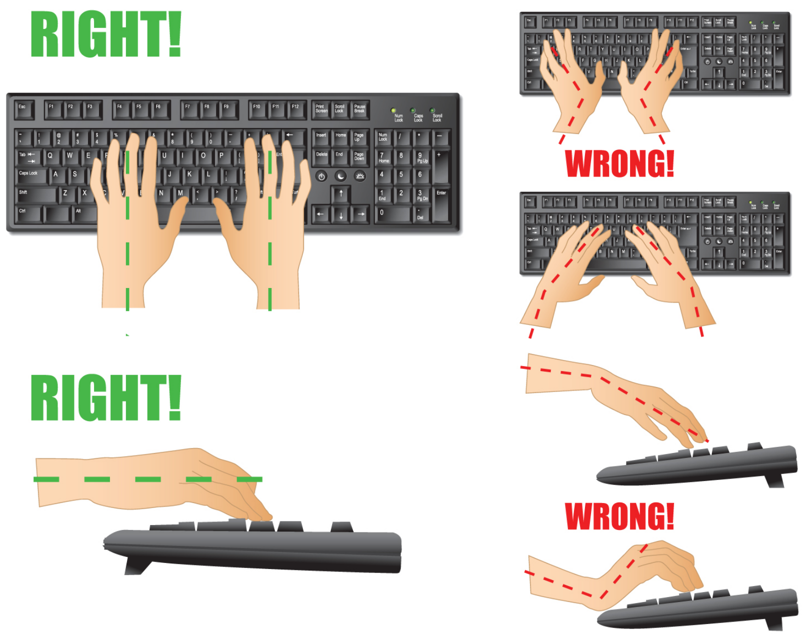
So, let's look at a regular keyboard



- Single board is cost effective & efficient, but can encourage unhealthy posture
- Offset rows are holdovers from typwriters

So this is a regular keyboard, or what would be a regular keyboard if it were an actual image of one. The two things I want to point out here is the width of the design & the offset rows. The offset rows are holdovers from the days of typewriters. The letters were offset like that to allow for the armatures to clear each other when striking the paper. In terms of today's application this is unnecessary and bad. The offset rows no longer serve any practical purpose. In terms of RSI injury, it's bad because you're reaching for those keys that could be closer to your fingers if they were placed in a straght vertical columns, which would reduce the amount of reaching you would need to do while typing.

Good Habits Vs. Bad



The reason I've included this is to demonstrate how a regular keyboard can encourage a certain posture, and hopefully you can see how the designs of these other ergonomic keyboards I'm about to show you are informed by these issues. I think you'll see how they attempt to mitigate some of these habits through better design. Most of these are other open source keyboards that you can build yourself.

Notable Mentions

So the original Dactyl design is what I chose to build, and I'll go into that process after this sectior. But first I'd like to quickly show you some other options that I've seen mention of while digging into this topic.

Kinesis Advantage2

- Advantages: Widely considered one of the best commercially available ergo keyboards for poeple suffering from RSI
- Disadvanteges: $320.00 USD, some reviews describe feeling 'cheap'
- Open Source: No

As I first started to really focus on learning tools like bash & emacs, I started hearing allusions to RSI, & more than a few references to this keyboard. It's probably the best commercial ergo keyboard you can get–from what I've heard–but it comes at a price. Also, I think it was a review by Linus Tech Tips where he describes the feel of the plastic being light & inexpensive. Not something you want to hear when you drop over $300 on something.

ErgoDox (EZ)

- Advantages: Lots of support if building from scratch, also may be a better product than Advantage2
- Disadvantages: Also about $300 if purchased commercially (depending on the options)
- Open Source: Both commercially available & open source

One of the most popular ergo keyboards I've seen. It's garnered the attention & a review from Linus Tech Tips, and has a long open source history. As far as I can tell, the Ergodox project kickstarted the idea of open source keyboard hardware. You see references to it everywhere, and its popularity in the ergo keyboard space is unavoidable. Almost any split, open hardware design has borrowed something from the ErgoDox. If you go the DIY route, you can find blank PCBs somewhat readily, and the rest of the parts can be easily sourced yourself.

Let's Split

- Advantages: Split-hand, simple, affordable design (~$100 USD)
- Disadvantages: Not concave
- Open Source: Yes

This was my first introduction to the idea of an open source keyboard. I was browsing thingiverse (A place where people share their 3D print designs), and someone had shared a brakcket they used to attach this keyboard to their laptop. This is probably the cheapest & simpliest columnar, split design project I've seen.

Atreus

- Advantages: Small, single-board form factor
- Disadvantages: Not a split design?
- Open Source: Yes

This is another keyboard I saw referenced a lot. It's a single board, but it's small (the website shows it fitting in someone's jeans pocket). The single board design could be an advantage or disadvantage depending on what you're using it for. It's potentially less comfortable to use, but I see it being easy to grab & go if you're using it in a mobile set-up.

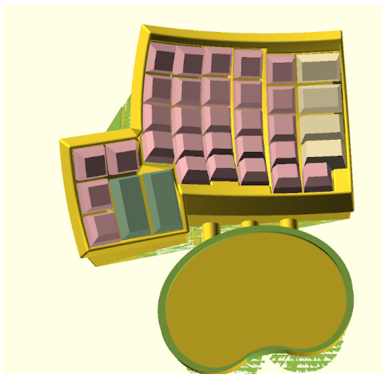
Signum 3 (Troy Fletcher)

- Advantages: Very simple design (solder on components, nothing else to worry about)
- Disadvantages: PCB is $80, no case for protection
- Open Source: Yes?

The guy who makes this is a freelance programmer based in Kentucky. He has a youtube channel that's pretty interesting if you're into vlogs. The notable differences between this & Atreus on the previous slide are the thumb cluster positions and the exposed PCB. It does look like the entire board is covered with a solder mask, but it won't be as protected if dropped. It looks like there is a repo made for this on github, but there's not much in it. You may need to contact the author for product designs if you need them.

Other Dactyl Variations

Dactyl Ergodox



- Advantages: Reuse your Ergodox keycaps
- Disadvantages: Incomplete design
- Open Source: Yes

This was a pull request by Joe Devivo (One of the writers of the earlier build guides). He was attempting to add some changes to fit the Ergodox keycaps, but the latest update to the pull was from 2017. /u/chrystalhand on Reddit has apparently made more updates to the design and is trying to market it on Reddit and OhKeycaps.com. I don't know if he's made the source available.

Lightcycle Dactyl

- Advantages: Slightly smaller footprint (fewer keys)
- Disadvantages: STL files for 3D printing appear to need more repair than basic model
- Open Source: Yes

In the main repository you'll find this as an option along with the 'cherry' option for the same version. The LightCycle version of the Dactyl has fewer thumb cluster switch positions and one less row of keys. It was originally designed to match with the Matias ALPS-inspired mechanical keyswitches. The 'cherry' version of this is the same design, but are meant to be fit with Cherry MX mechanical switches.

Dactyl Manuform

- Advantages: Thumb clusters are brought down to a more natural position, Case is larger making wiring less tedious
- Disadvantages: Must be wired by hand, all of the challenges that come with original Dactyl
- Open Source: Yes

I think this is probably the best designed of all of the options I've seen. The Manuform retains all of the features you would look for in the original, but lowers the thumb clusters so that your hands can remain in a more natural position. Other notable differences between this & the original Dactyl are the use of DSA keycaps and the use of 2 Pro Micros for the microcontroller. I'll probably build & switch to this in the future.

More Do-it-Yourself Options

```
There is a very nice list of other ergonomic keyboards, with
pictures, on Xah Lee's website.
```

- http://xahlee.info/kbd/diy_keyboards_index.html

Reasons for Choosing Dactyl

- Open Source
- Looked like the most comfortable design
- Kinesis Advantage form factor, but DIY
- Looks hella cool

My Reasons for Building by Hand Instead of Purchasing

- At the time there were none being manufactured
- Sense of self-satisfaction
- I already own a 3D printer: screw paying someone else >$300, I'll just build my own!

One Year Later

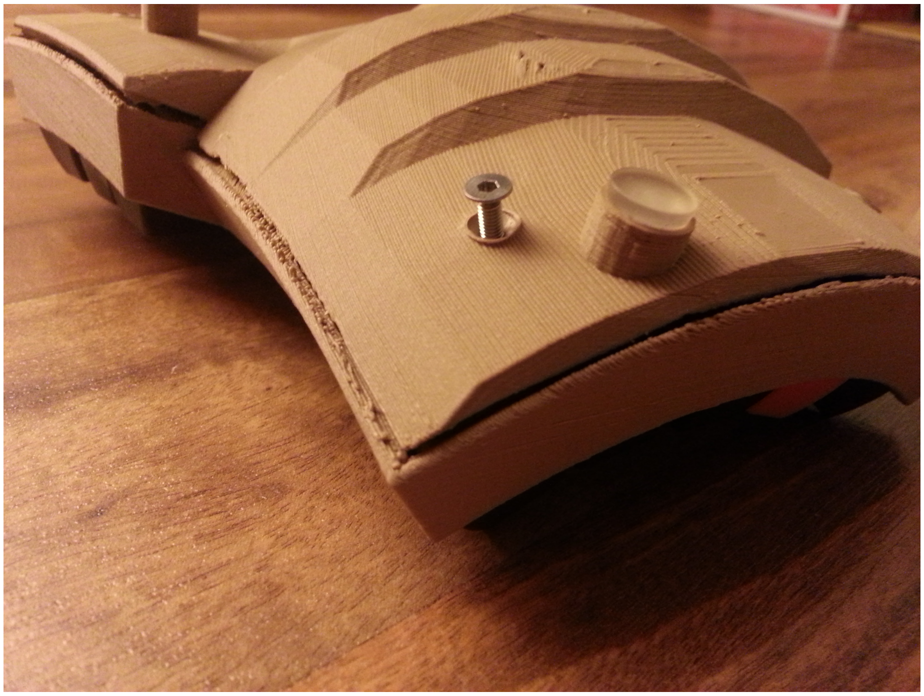
Build Overview

Shell/Case

- The body of the keyboard is 3D printed by me
- There are 4 parts to print, each took 21 hours to complete on a RepRap style Cartesian 3D printer
- Material is PLA infused with wood fibers.

The body of the keyboard is 3D printed in PLA with infused with wood fibers, so at some point I want to try and stain it. There is a top and bottom to each half, and each top and bottom piece took 21 hours each to print. My main challenge here was finding the correct amount of support material to use. For a slicer I used Slic3r, and your options for support material are somewhat limited. In my first attempt I used too high of a resolution and the support was so fine that it was impossible to separate from the print. My second attempt was the bottom left half and I think it turned out pretty good.

Hardware



For hardware connecting the top & bottom halves of the body, there was no documentation for it so I referenced some build videos on YouTube and dug through my collection of computer hardware & screws and found that 3mm female motherboard standoffs that were 6mm tall worked great with these 3mm countersunk screws. I took a soldering iron and heated the screws up while pushing them into the hole in the plastic to create a countersink. Without that countersink there would be clearance issues with keycaps, so the keycaps would hit the head of the screw when pressed down. I also added some 1/4" heatshrink tubing to the standoffs so that they wouldn't cause any shorts with the wiring.

Switches

- Fits Cherry MX switches

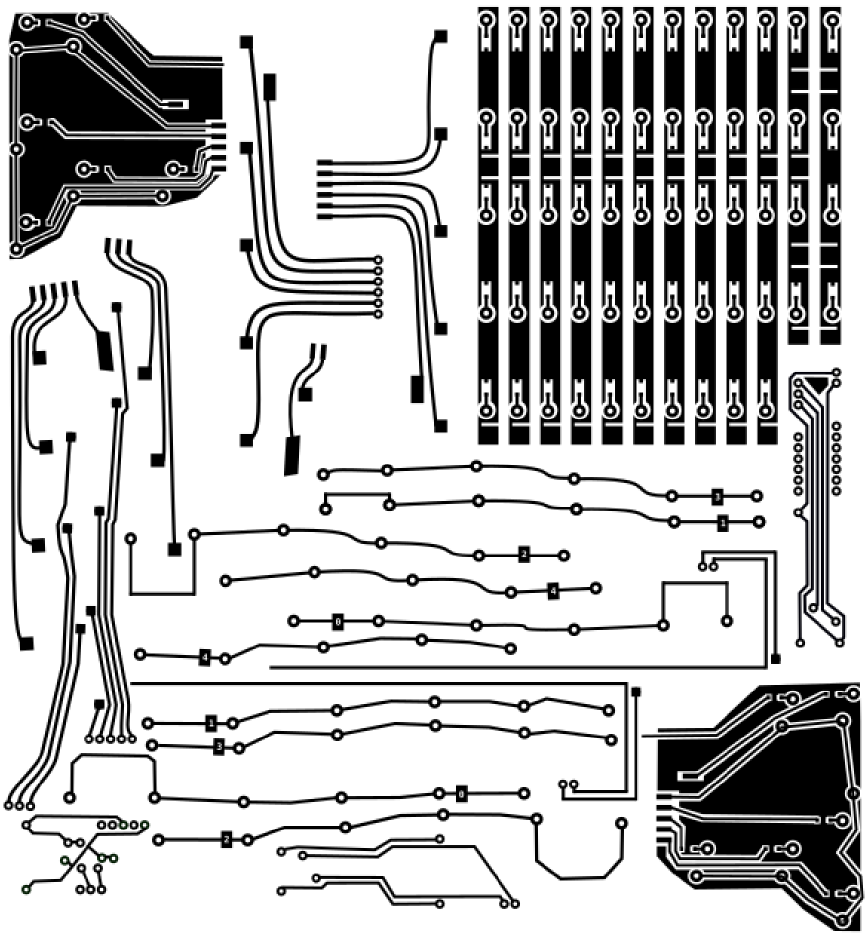
So Cherry MX is the most popular switch manufacturer on the market. Most mechanical keyboards use Cherry MX for key switches. There are several different models available which range from linear to tactile feel, quiet to noisy & clicky. I have 9 different ones to choose from here if anyone wants to test them out. There are knockoffs available on the market as well. Gaterons are a notable example (just be aware that colors/models differ between the two manufacturers).

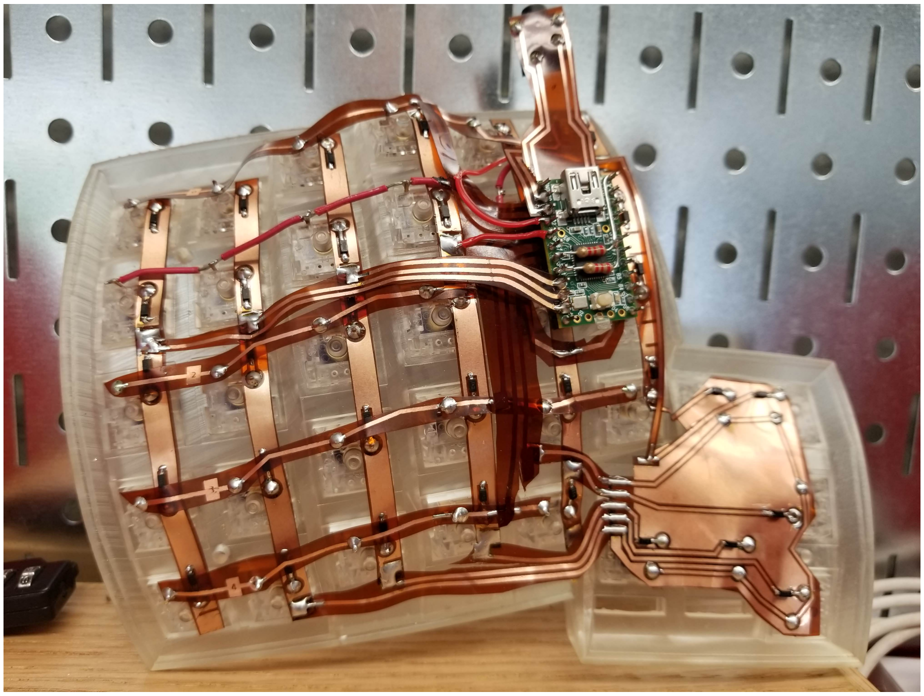
Keycaps

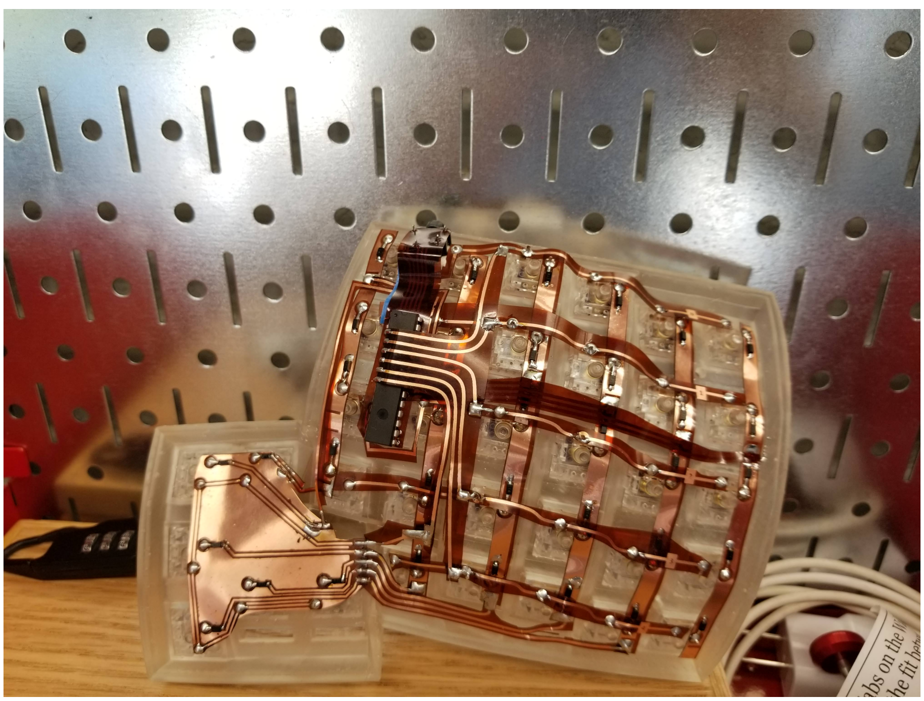

SP DSA
Apple //c
80s Alps
1990 laptop
SP SA
Alps spherical
Cherry
SP DCS
Alps
Tai Hao
OEM
Laser
Apple (AEK)
Apple IIGS

If you look at your OEM keyboard from the side, you'll notice a curve to the tops of the keys. The bottom row of keys also hav a different shape compared to the top row. This is called the keycap profile. SA is what's recommended for the standard Dactyl and what I've used in my build. DSA is what's recommended for the manuform. A quick note on keycap material: You'll see two main types of materials referenced when shopping for keycaps. ABS & PBT. ABS is cheaper, but fades over time. PBT is known for holding up better but is obviously more expensive for that reason.

PCB Design





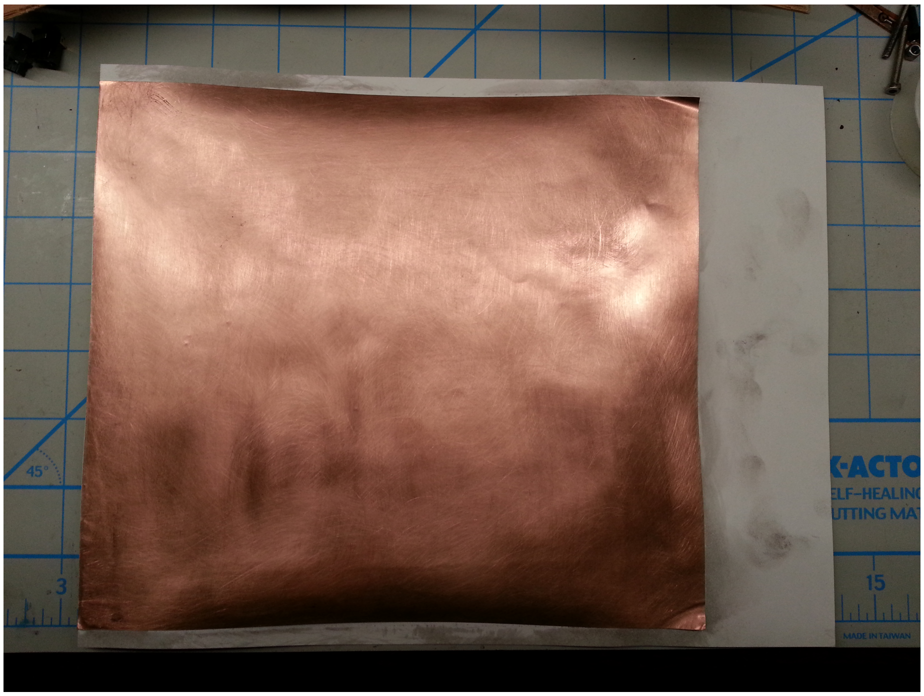


Oh boy. So to start with, I've combined both the left-hand & right-hand PCB designs onto one sheet to save on material. The way this works is there's something called Pyralux you can get that is basically a thin sheet of copper with a kapton tape backing. This is what I've used for mine. We'll get into the etching later. Another thing I'll note about the PCB designs in the original repo is that some of the pieces are flipped the wrong way around, and for some reason they weren't consistantly flipped. I found this out after my first etching attempt and I had cut out all of the individual pieces and was trying to apply them to be soldered. If you were using the toner tranfer method, where you would use a laser printer and iron the paper onto your workpiece to leave a toner deposit in the shape of your circuit, it would make sense to mirror the entire design, but some pieces were flipped one way while others were correctly oriented. I searched imgur.com for completed builds and was able to find 2 very nice photos of a nearly completed build. I used those to cross-reference the design, and I used Inkscape to flip each piece individually.

Xerox Printer

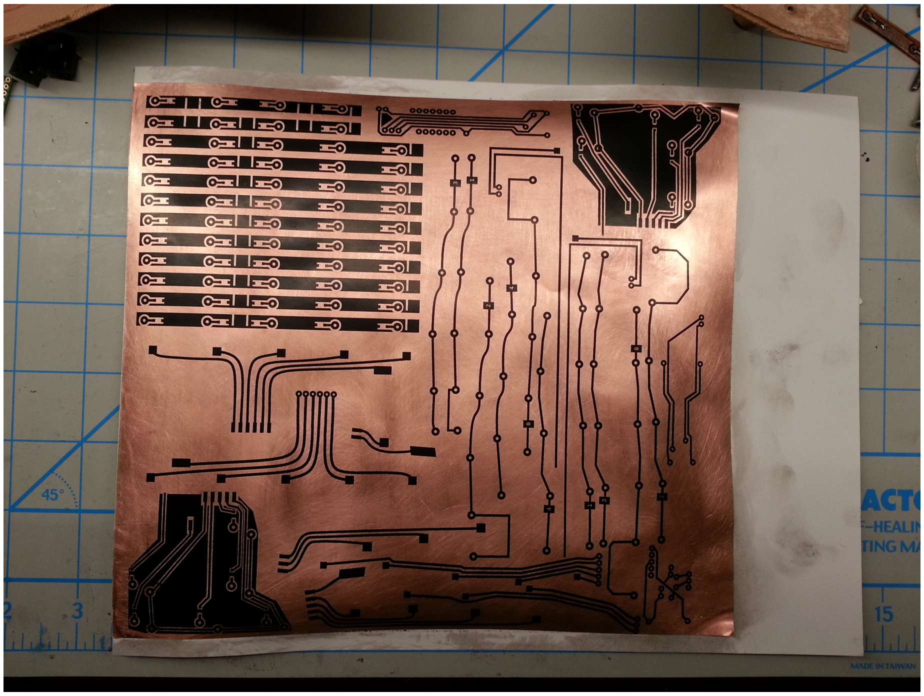
In many tutorials, you'll read references to printing circuit designs onto Pyralux with toner laser printers at Kinkos. This is incorrect. What they're actually referring to are solid ink printers manufactured by Xerox. These commercial printers use a type of wax for ink, and are actually kind of hard to come by. I was able to find a used one on Craigslist that was having paper jamming issues that I picked up for $50 from a guy in Rogers, MN. I found an official service manual on manualslib.com & pinpointed the offending sensors for the paper jamming issue and replaced them with parts sourced from ebay.

Preparing the Copper for the Design

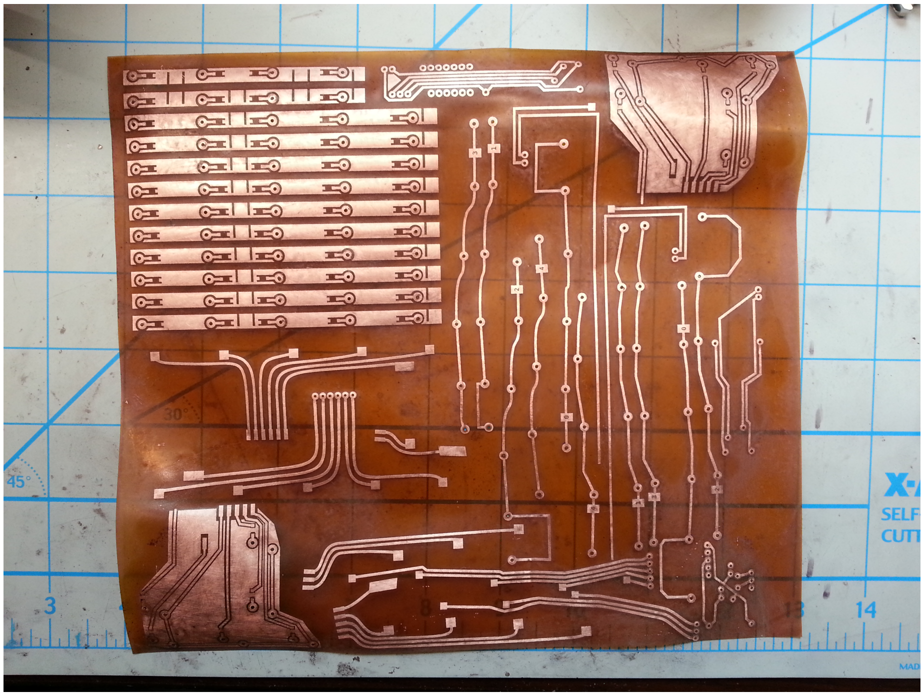


To prepare the copper I scuffed the surface of the copper & rubbed it down with 90% Isoprpyl Alcohol. I also had to use double sided tape to center the Pyralux sheet at the top of a piece of card stock, so that the printer would recognize the sheet as a normal 8 1/2" X 11" piece of paper. The last thing I did was change the print settings on the printer to 'Transparency' so that the printer would apply enough wax to hold the design together.

Final PCB Design



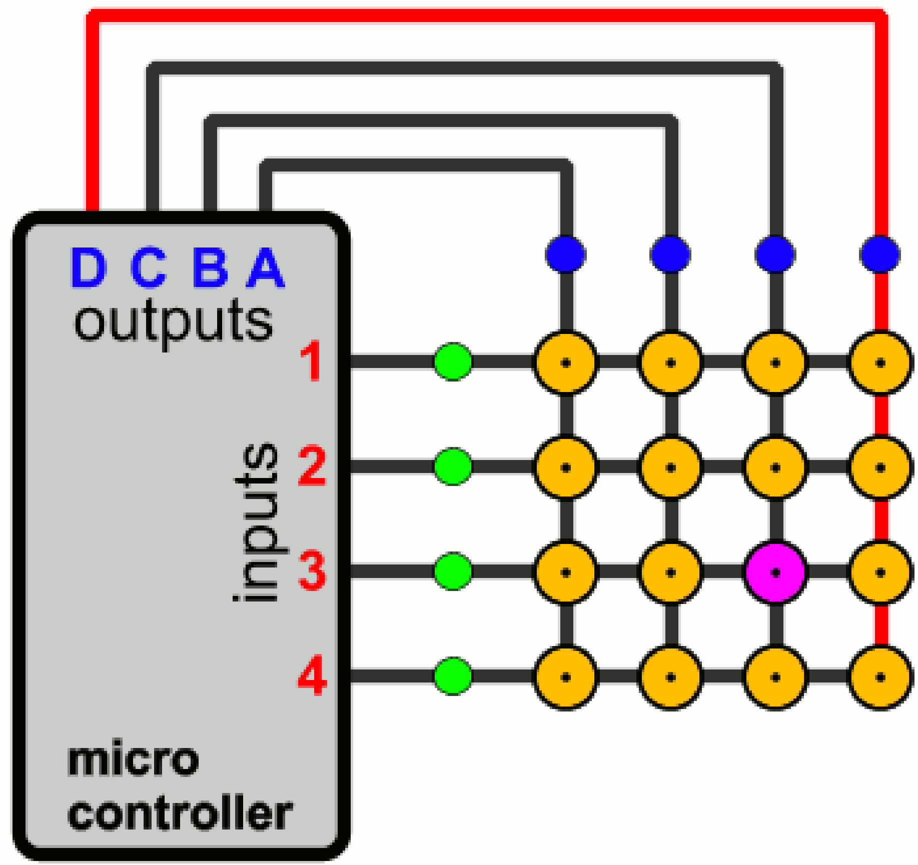
Etching



For etching I used about 1:1 ratio of 3% Hydrogen Peroxide & Acid Magic (Marketed as 'safer' Muriatic Acid, found on Amazon.). This seemed to be the safest and most accessable option over ferric chloride. You can reuse the solution by monitoring the amounts of each ratio in the solution, but you need to be very careful about it because you might end up creating chlorine gas.

Make sure to 'add the acid', you can see the copper lifting here, you'll notice the color of solution turning green with copper at this point, here you see the Kapton tape backing and some remaining copper, after you're done you should put the sheet in a water bath to stop the acidic reaction and clean any off any residues left from the precess. I chose to sand off the wax with 220 grit sandpaper because it was mixing with the solder and I got annoyed by it. You probably want some kind of coating over the copper to prevent corrosion.

How Does it Work?



So there are two ways in which you can wire up your key matrix, which are row-driven & column-driven. A row-driven matrix is one where current travels from the microcontroller, through the switch & diode & then back into the MC via the column. A column-driven matrix is the inverse. While wiring the key matrix, one big challenge was trying to figure out the orientation of the diodes. If you're not familiar with what a diode is, it basically works to only allow current to flow in one direction. The way they're used here is to insure that the input and output sent and recieved by the microcontroller only flows in one direction. The orientation of the diodes is entirely dependent on how you wire your rows & columns and whether you choose to send output through the switch from the rows or the columns. In my case I went with a row-driven matrix, as that seems to be most common, and I read allusions to it being a more efficient option in the QMK firmware. With that decided, I wired the Cathode, or negative end with the stripe facing away from the switch pin on the column. As long as all the diodes are consistantly wired, it really shouldn't matter how you wire your matrix. You can easily flip it in the QMK firmware. I should also point out there is a discrepency in pin position for some of rows between the photos I used for referencing the PCB design and the wiring diagram shown in the guide in the repo. I attribute this to updates in the QMK firmware, but I don't know. It's something to be aware of.

Notes on LEDs

- Use Ohm's Law to match reistors to curret rating on LEDs (Assume 5v)
- For LEDs under keycaps use T-1 LEDs

One challenge I faced was how to choose resistors to match up with the LEDs, and what size LEDs to buy. LEDs need resistors paired with them so that they draw a steady current and don't become overloaded. You won't likely be able to find the current draw unless you purchase them with a documented current draw. I got mine from Digikey.com. I used on online calculator to calculate the correct resistance, but the formula is pretty simple. R=I/V. Almost all USB hubs are going to power your devices at 5V DC, so if you take the current rating for your LED and devide it by 5V you should get the resistance value needed for your LED. I wanted the LEDs to fit into the slot provided by the key switch, so I looked it up and the T-1 LED form factor is what you want to use. This is also the type of LED you would use to backlight your keycaps.

PSA: TRRS != TRS!

file:img/TRRS.JPG

So when I ran my first tests after completing the wiring ane firmware, OSX kept complaining about the USB device drawing too much current. It was selfishly holding power to the USB hub hostage until I unplugged the device. This is why. This is the cable that connects the two halves together so the MC can communicate with the left half. The letters TRS stand for Tip, Ring & Sleeve. The documentation calls for a TRRS cable. Originally I mistook it for a basic 3.5mm stereo audio connector, but as you can see it doesn't have the necessary contact points and was causing a short inside the input jacks. The way I solved this on short notice was by running to Menards and buying 2 sets of cheap earbuds with microphones built in, cutting off the earbud half ande soldering the 2 ends together.I'm still not getting input from the left half, but I think that's more likely due to a short in the right half that I still need to diagnose.

Firmware

So I have a spare Teensy 2.0 here, and I think instead of describing the flashing process I'll just show you. I'll be loading QMK firmware onto this. It's the most popular and widely supported keyboard firmware available that I've found at least. They have a huge list of hardware that the project supports.

Was It Worth It?

Parts:	Cost in USD:
1/3 1kg Spool PLA Fil for Shells	10.00
70ct Key Switches	53.00
Rubber O-Rings for Switches	2.00
Designer Key Caps	80.00
Pyralux (Flexible copper for PCB)	30.00
Electrical Components/MCU	65.00
Total	240.

Resources

- Drop.com (formerly MassDrop)

Crowd sourced, limited manufacturing.

- OhKeycaps.com

Working with members of the reddit mechanical keyboard community to commercialize different Dactyl variations.

- MehKee.com

Seems to be the primary vendor for the Let's Split PCBs `I have no affiliation or experience with the above. YMMV!`

Tutorials

- Etching with Muriatic Acid & Hydrgen Peroxide
- HongKongGhost - Dactyl Hand-Wiring Build Video
- AfrotechMods on resistance
- https://www.instructables.com/id/Making-flexible-PCBs-with-a-laser-jet-printer-or-c/
- https://www.instructables.com/id/DIY-Flexible-Printed-Circuits/

Hardware & Accessories Vendors

- Digikey.com - Electrical components
- Sparkfun.com - Hobby-grade electrical components
- McMaster-Carr - Hardware supplier (price-compare Ebay first)
- KBDFans.com - Lots of keyboard components, ships from China
- AliExpress.com - China's Amazon.com, lots of hardware including keyboard supplies
- PimpMyKeyboard.com - Another keyboard component vendor