My Keyboard:

My interest in keyboards and input devices started when I wanted a global hotkey to skip forward or backward in Spotify. The shortcut is there, ctrl + shift + right/left arrow, but it only works inside the app. I loved the utility of the media keys built into my laptop, and wanted one for that function that I could use anywhere in my system. So, I wrote an autohotkey script to make that feature myself. It didn’t have any buttons which could activate it though, but I have a Launchpad that I use for music production. I wanted a way to connect that to my script.

I found that the audio routing program Voicemeeter, which I had already used to make a makeshift soundboard, has a function to turn midi signals into keystrokes. I upgraded to Voicemeeter Potato, a more extensive version of the software, and started setting it up. I ended up programming my Launchpad to be a sound/media control mixer, controlling the volume and some different effects of different programs on my computer. I connected it to autohotkey using combinations of F13-F24 keys, which are recognized by my computer but don’t have any other function or ways to trigger them. I also programmed and added a few extra slots for copying and pasting, so I could copy/paste multiple things in parallel.

But, I wanted a smaller solution. Along the way, I learned of macropads, small clusters of keys or other various input devices. I loved the idea of having volume knobs, coding firmware, and soldering, so I bought the Keebio BDN9. I built this 7 key 2 knob macropad with 78g zilents, lubed with dielectric grease because I had it on hand. I put some DSA blank keycaps from KBDfans on it, and a white vinyl wrap on the top plate after the fact. I built a lego bracket to connect it to the side of my laptop, hovering over my mousepad on the right side. It was very helpful to have media and basic volume controls on that, and it showed me the world of mechanical keyboards.

My new macropad felt considerably better than my laptop keyboard, and I liked the idea of building a full board. At this time I had also started browsing keyboard subreddits and watching keyboard youtube videos, so I learned more about the hobby. Initially, I loved the layout of the Keebio Sinc, but didn’t like the case or the fact that it was a split board. I remember seeing one of u/onebigdoor’s posts on r/mk, and handwiring stuck in my brain. The flexibility, diy aspect, and look of handwiring kept me interested. I saw more and more amazing groupbuy boards, but I thought that the sense of ownership would be that much greater if I wired it myself. I sat on the idea for a while to make sure it was an actual interest, and then started on my project.

My first step was finding what I wanted the keyboard to be. I made a mock-up in Tinkercad, a tkl with a removable southpaw numpad, a connection I would figure out later. It also had two knobs instead of the print screen and pause keys and another four (I later changed it to three to give my fingers room to turn them) in the space at the top of the numpad. I made it the numpad removable because they seemed really helpful when working with numbers, which I frequently do, and because I don’t always want the space taken up by a numpad. It was on the left because that seems like the superior place to put it, so I can still have a small distance between the alphas and my mouse and so I can still use my mouse while entering numbers.

I learned more about handwiring and custom keyboards and started designing. I used Tinkercad at first because I had previous experience with it, but I think I could have saved a lot of time had I used Fusion360. I found a full-size plate 2d file and edited it to match the layout I wanted, and started wiring. I’ll talk more about the details of wiring later. In my design, I used rotary encoders as knobs, which take a spot on two rows and three columns to say if they are turned left, right, or pushed down.

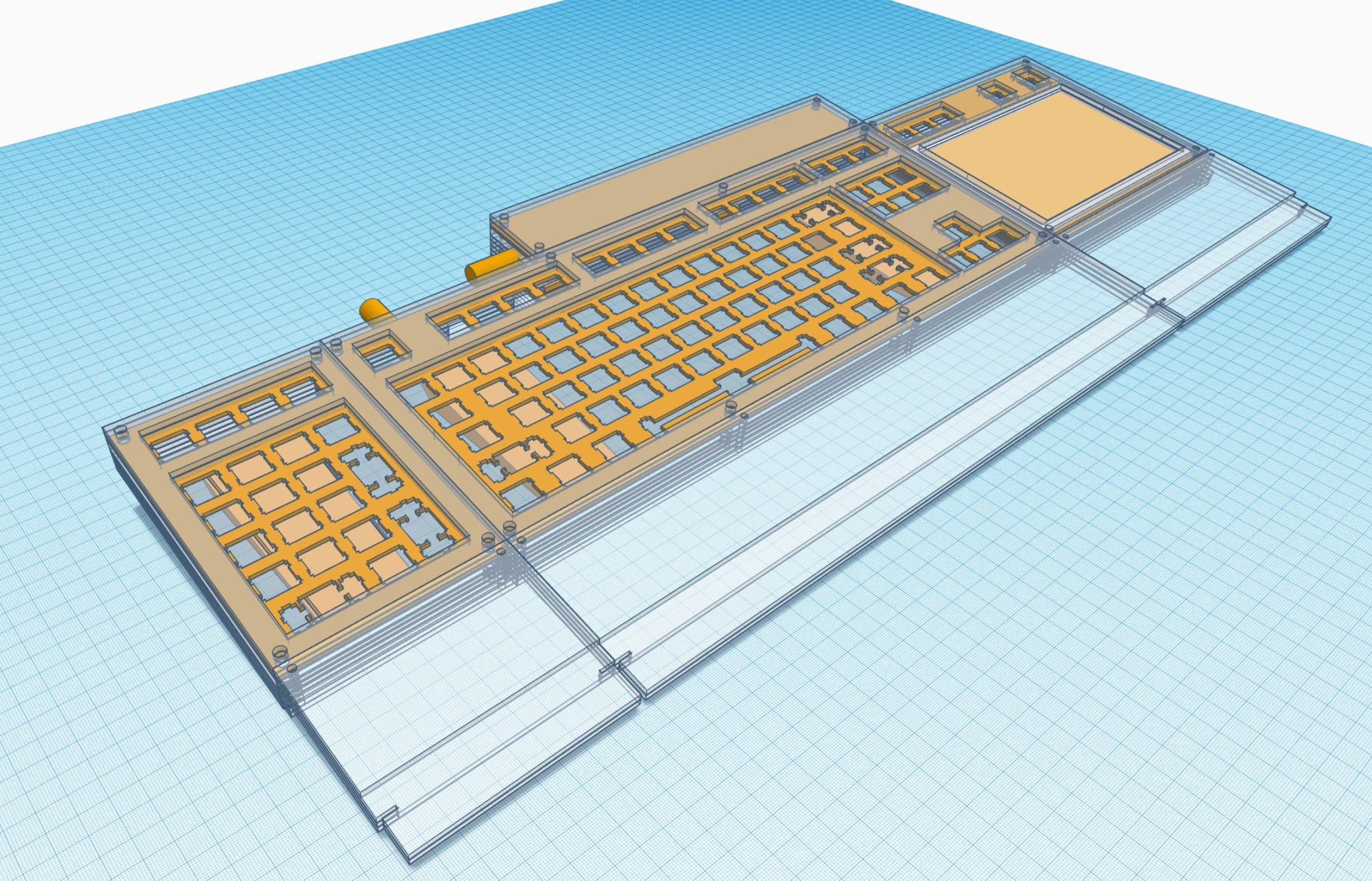
In the 3d space of Tinkercad, I wired up all of the keys in my first design to a model of a microcontroller. I also had some white leds and analogue joysticks in that first design, but I dropped those ideas later. At this stage in the project, the keycaps I wanted on it were GMK Blue Samurai and I wanted the plate and all of the hardware to be brass. I tried rendering the board to see what it would look like:



Before I started designing the case, I wanted to re-evaluate what I wanted in my design. I went back to the principles of the keyboard: an input device. I realized I didn’t want leds in the board, I wanted it to look good enough on its own. I thought it might be more versatile to have it be bluetooth, but I didn’t always want to deal with that aspect of it. Like the numpad, I thought a removable bluetooth module could be a good idea. I liked the look of the HHKB’s bluetooth battery pack at the back, but I didn’t have much room inside the board for a battery, so I planned to put the bluetooth module on the back of the board. I wasn’t sure what I should add next, so I asked some friends. One suggested that a pull-out trackpad could be useful, so I thought the idea over. It solidified the keyboard’s purpose as an input device, and could be a useful tool, so I added it to the design. It had a restive touchpad, two knobs for scrolling, and three keys for clicking.

I got to work finding all the parts I needed. I found everything from lube to spring pins. That was how I planned on connecting the parts of the keyboard, spring pins and contacts, so they could still connect even if they were a little uneven. I’ll list all the parts I used at the end.

The next step was to design a 3d version of the case. I was working with lasercut sheets of a few different materials, so my design was limited by the parts only being able to vary in two dimensions. First, I made accurate 2d files of the switch plates, using <http://www.keyboard-layout-editor.com/> and <http://builder.swillkb.com/>. I then built a few boxes around the plates that could be made up of 2d elements. They were mainly made of horizontal sheets, but had some vertical panels for the spring pins or ports to sit in. I also designed the feet in the phase, using the 8.5-degree typing angle in as much of the feet’s geometry as possible. I was used to typing on a laptop, or a very shallow keyboard, so I thought I would benefit from typing with a wrist rest. My layout is strange enough that I didn’t think I would be able to find a wrist rest sized to it, so I designed one that was based on the Keycult acrylic wrist rest.



To lasercut my case, I needed to draw some 2d files. This was the longest step because I wasn’t careful enough with the dimensions at first. I used adobe illustrator to edit files I made with swillkb to match the designs I made in Tinkercad. The process of turning objects I made up in 3d to 2d was interesting, and I often confused myself. This part of the process was the first that I was working with numbers that actually mattered, so it felt good to get out of working with just concepts. Now that I had accurate dimensions of the case, I could find exactly what other parts I would need for the board. One decision I needed to make was on how thick the standoffs holding the case together should be. I liked the look of the bigger screws, especially when I was still trying to have all the metal be brass, but I didn’t have much room for thick standoffs in the walls of the keyboard. I finalized the parts I wanted and made a few changes to my designs based on them.

I could finally move on to ordering the parts. I didn’t know where I could source my plate from, because all the brass cutting services I found online were very expensive. I called a local shop, and found out that it would be expensive anywhere I went. So, I decided I didn’t want the board to be brass. Luckily, aluminum was much cheaper to get cut. I found silver alternatives to most other brass parts of my keyboard too, so it would match. Looking back, I think aluminum plates sound better, so I don’t regret that choice. The lasercutting service I had in mind to cut all the acrylic and wood, Ponoko, also cuts aluminum, so it worked out well. Beyond not knowing where to get my plate, I didn’t know the exact dimensions of a few parts, like the standoffs, magnets, and knobs. I ordered them, and a caliper to measure those parts accurately. I adjusted my designs to fit those new measurements. I also ordered a few test parts of all of the materials I was planning on using for my case to see how they looked and felt. I also wanted to see how the aluminum bent, because the feet are made of a bent sheet of aluminum.

Another thing I was unsure of was how lasercut acrylic fit together. My school was very helpful here, I was able to go to the middle school and test some cuts with a teacher there. I had bought some acrylic sheets from a local shop that I thought I might have wanted to get all my acrylic from, but they were pretty expensive and didn’t have many types of acrylic in stock. At school, I tested the distance between acrylic parts for them to friction-fit well together, and how my standoffs fit into some different sized and shaped holes. The teacher also 3d printed some little test brackets that let me put rotary encoders in the place of switches. It was around this time that I tried rendering the board again, and I got part way through that process when my computer broke. I had backups to my files for this project, so I didn’t lose much, but it was still a pain starting again on a different computer for a time.

After making some adjustments to my files based on what I learned, I was ready to start ordering parts. I bought around half of the parts to measure them some more before I ordered my case. After they came and I adjusted the files some more, I spend a few weeks combing through the files and confirming the accuracy of them. I also adjusted for kerf, so the laser would cut to leave what I wanted rather than cutting a little bit off the edges of everything. I also designed feet accurate to the new dimensions and what I learned from bending the little piece of aluminum. The feet are very geometric, with a lot of 8.5-degree angles I didn’t want to guess at, so my knowledge of trigonometry helped me here. After many months, I finally ordered the case and the rest of the other parts. I knew it was unlikely that everything about the case would be perfect, and I might have to reorder a few edited parts, but at least I was able to start wiring.

While I was waiting for the case to arrive, I lubed my switches. I chose to use Glorious Pandas for this board, because I like the idea of very tactile tactiles but didn’t like the heavy weight or weird pre-travel of the Zilents I had on my macropad. I used Tribosys 3203, and moderately lubed all the points of contact (including the legs and leaf, I know that’s bad for tactility, but I felt it was worth it to reduce the spring ping). I also started coding my trackpad. The 4-wire touchpad to mouse converter I bought didn’t end up working like I had hoped, so I bought a pro-micro clone to code the trackpad with Arduino myself. I’ll say more about how I coded it later.

When the case arrived, the non-metal pieces hadn’t been popped out of the sheets they had been cut from, with the tape backing still on. It was quite a process to remove all of that. Once all of the pieces were free, I tested how they fit together, with the standoffs, switches, knobs, and interlocking patterns I had designed. There were problems. The 3d printed adapters that that were supposed to let the knobs fit into holes made for switches were too big, the pins connecting the pieces didn’t fit snug, and the usb ports didn’t fit nearly to what I thought they would.

The biggest problem, however, was that the holes for the standoffs were far too small. I didn’t want to spend another 300 dollars and another month to reorder all of the parts, so I decided to work with what I had. I could either widen over 250 holes by hand or grind down 22 standoffs. I chose to grind down the standoffs. I found that with a dremel and a few vice grips, I could make the standoffs fit in my keyboard. It took a bit of time, but I got them all done. They needed to be so thin that there was almost no brass wall left, and you can even see the interior threading from the outside of some.

I popped all of the switches into place, put the magnets where they needed to be, and glued the pins into place. I assembled the board for the first time (without keycaps or any wiring), and it felt good. Now I wanted to get it working. I worked on wiring the keyboard for about two weeks straight before I started debugging the issues.

Each switch is connected through a hot swap socket, which allows me to take the switches out without desoldering. In a handwired keyboard, they don’t work quite as well as in one with a PCB, but they’re still helpful. Anyways, a diode needed to be soldered to one of the sides of each of these sockets. I wrapped the anode of the diode around the end of the socket and added some solder. Once I was through will all of those, I could attach them to a long wire to create a row. I cut a very long piece of thick wire, bent one side into something my drill could hold onto, and twisted until it was straight. This is a very helpful trick that saved me a lot of time and made everything look a lot cleaner. I cut that long wire into a length appropriate for the row it was to become. I marked with a sharpie where along that row the cathode of each diode should connect, and used a dremel to grind away the enameled coating of the wire. If I didn’t do this, I wouldn’t be able to solder to the wire. Then, I soldered the diodes and sockets onto the points I had made for them and put all the sockets onto the backs of all of the switches to create the row.

Next, I soldered the columns. I took another length of straight wire and dremelled into it where I wanted to solder. Unlike the rows, the distance from each socket to the wire was very uneven, so I needed to solder a short thin wire for many of the connections. After I attached every switch to a unique row and column combination, I checked all of the connections with a multimeter. I set it to measure resistance, put the prongs on either side of a connection, and if there wasn’t infinite resistance, the connection was good. I wired the knobs like other switches, just with the rotation measured by one row and two columns. This proved later to be a bit of a weird way to do it, but I got it to work. I extended the rows to the left and right to the removable sections, and used other pins between the parts to connect the columns from the numpad and trackpad to the main board.

Now that I had a complete matrix of all of the keys, I could start connecting the matrix to the microcontroller. This is what the computer connects to, it handles reading which keys have been pressed and sends the appropriate signal up to the computer. I had so many rows and columns that they didn’t fit on the standard Pro Micro microcontroller, I had to use an extended one called a Teensy++ 2.0. I chose a location for this little board, and got to work making the wire runs. Each row and column, some of which were spring pins, needed to connect to one of a handful of safe pins on the microcontroller, and the path from it to the pin wasn’t always a straight shot. I made the wire runs into a neat pattern, and soldered them together.

Next I needed to route that usb signal from the microcontroller to the port in the back of the keyboard. Because I had the trackpad module, and I wanted to consolidate the two usb signals into one cable, I needed to use an internal usb hub. Using this internal component also allowed me to add usb passthrough ports on the back, for something like a mouse or thumb drive. Usb hubs are relatively thick, and fitting one in was a bit of a challenge. I needed to snap the actual usb ports off of the one I chose, and just solder to the pads that female port was connected to. It also only fit in one location, so the wire runs from the ports on the back, the microcontroller, and the spring pins for the trackpad were a bit long. I now had everything inside the main keyboard, numpad, and trackpad wired.

However, I still needed to make the cable. This proved to be very difficult, mostly because I chose to use very small aviator connectors with more pins than is standard. I chose to do this because of the way I wanted the keyboard and the battery to connect. The battery, which was removable and hung off the back, didn’t extend all the way to the left side of the keyboard, to leave the ports accessible. I could have done it other ways, but I decided that I wanted to swap between wired and wireless by disconnecting the first portion of the cable to the computer and plugging it into the battery module. This splitting of the cable required aviator connectors, and I needed 6 pins so that the module was only on while it was connected to the keyboard. I sleeved the coiled cable, to fit the two extra wires that I would need, and cut it up how I needed it to be. Soldering the aviator connectors was very finicky. The fit on the cable was very snug, so I couldn’t easily access the solder joints and it would cause me to twist it too much and break the connections. This was especially difficult on the connector on the coil. There were shorts, broken solder joints, and shorts on the connector housing itself. It took me a week to get the cable sorted out so that all of the connections were good.

Now that I could connect it to the computer, I could start debugging any issues I had. There was a lot. Firstly, I found that I had fried the usb hub, and even though all the connections going through it were good, it wouldn’t do its job combining them into one. So, I rerouted the microcontroller’s usb directly to the main aviator connector.

Now my computer registered that something was connected, but I still needed to program the Teensy. This was also a lot more difficult than I had anticipated. I used qmk to compile the firmware that runs the keyboard, sending power through the rows and reading the columns to find out what key has been pressed. It’s a very powerful and concrete tool, but a real pain to use. It isn’t that bad if you have a keyboard that has already been made, like any you would buy, but for handwired builds it’s a lot more complicated. For qmk to work, you create a specific collection of text files in a specific place and use a unix terminal to compile those files into a hex file your microcontroller can read. However, none of the guides, videos, tools, or even the qmk docs got me something that would compile. After a week or so of trying things, I got it to work. I had to use part of the source code of an online generated keymap combined with files I made to imitate ones for other keyboards in the files structure that a qmk command gave me. I tinkered with the keymap to make something that would work and loaded it onto my keyboard.

Only a few keys worked. This actually ended up being not the worst part of debugging, because I could just go through with my multimeter and check which connections I needed to solder again. Some things worked but then broke later, so there was a time period where I needed to rewire a few things every so often to make my wiring more resilient. The removable parts didn’t work completely, because many of the spring pins didn’t consistently connect. I just wanted to get it to where I could type on it though. Another issue was how the knobs worked. Once I made sure the wiring was ok, I realized that they were sending both commands, for left and right, whether they were turned left or right. The only difference between the two was the order of the two commands. A little bit of qmk magic with the directions triggering one-shot layers which would send the opposite command got them working ok. Because of a few different reasons including polling rate on the microcontroller, processing time for switching layers, and the incredibly short signal that turning a rotary encoder sends, the knobs don’t work incredibly well. They sometimes send the wrong signal if you spin it too fast, or get stuck sending one signal. But they work. Also, the guitar knobs I put on top feel great! Much better than the aluminum-plated plastic ones keyboards normally use. Some stabilizers needed a bit more tuning, but I could do that while they were still inside the keyboard. I fixed some other issues with the keymap and wiring until the main keyboard was working how I wanted it to.

Before moving on to the removable sections, I wanted to make the wireless module. The wiring was relatively simple, I just connect the usb from the keyboard to the input to the wireless adapter and battery pack to the power usb, running one wire up and down the extra two in the cable. I also added a charging port in the back behind the battery pack. I ground the rest of the standoffs I would need for it, assembled the laser-cut layers, and put all the other components in their place. The metal tabs Wiring was fast, but the usb didn’t connect. I didn’t fry anything, because it worked if I just connected everything without my wiring. The cable was the issue again. As I had bent it the opposite way to connect the aviator to the battery module, the two wires I added snapped. It took a few tries to get them reconnected so they wouldn’t break again, but then the keyboard worked wirelessly! Not everything works like it does wired, like F13-F24 don’t get sent so I can’t use my autohotkey scripts. The polling rate is also too slow to use the knobs well at all, too many inputs are missed. There is a feature with my adapter that could throttle these if they were actually being read, but that didn’t fix it.