Diego Rivas
ME290U – Text Entry Device

What text entry technique was used? Why?

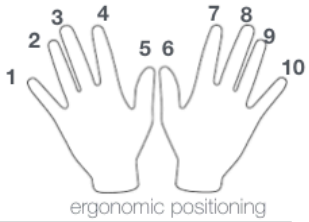
For this design I chose the chording technique, where single keys and dual key combinations gave different output characters. I chose this technique firstly because I believe it is quicker than other techniques that could be used (such as pressing the same key multiple times) and also simpler to rememeber. In my experience using previous devices like this, and in prototyping this device, I realized that I quickly got a hand for the different chords that produced the different characters through muscle memory. In summary, I chose this technique because of its efficiency in time and ease of use.

How was character recognition implemented?

My code went through a loop that listed out all key combinations and circumstances for which a certain character should be printed out. Prior to designing my code, I looked at the algorithm what I wanted to achieve, and also built a key map for easier reference:

Chording: key combinations trigger specific letters

Special Consideration: give a small delay for users to trigger all keys



ergonomic positioning

Pattern Design

KEY PRESS	1 (D0)	2 (D1)	3 (D2)	4 (D3)	5 (D4)	6 (D8)	7 (D9)	8 (D10)	9 (D11)	10 (D12)	Letter
SINGLE	■										E
		■									T
			■								A
				■							O
					■						SPACE
						■					I
							■				N
								■			S
									■		H
										■	R
ADJACENT PAIRS	■	■									D
			■	■							L
					■	■					C
							■	■			U
									■	■	M
		■	■								W
				■	■						F
						■	■				G
								■	■		Y
SPACE + KEY	■				■						P
		■			■						B
			■		■						V
					■		■				K
					■			■			J
					■				■		X
END KEYS	■			■							Q
						■				■	Z

Furthermore, I assigned specific key locations to finger number based on hand placement (this will be explained further in the next section).

Furthermore, then I listed out my algorithm in the form:

```
If(sensor combination is read high)
        Debounce sensor
        If (it is still high)
                Print character onto serial terminal
end
```

Effectively, what I then ended up doing was writing code such as this sample:

```
//printing 'e'
        if(digitalRead(keys[0]) == LOW && digitalRead(keys[1]) == HIGH &&
         digitalRead(keys[4]) == HIGH && digitalRead(keys[3]) == HIGH ){
          delay(100);
                if(digitalRead(keys[0]) == LOW &&  digitalRead(keys[1]) == HIGH &&
                digitalRead(keys[4]) == HIGH && digitalRead(keys[3]) ) {
                Serial.print(letters[0]);
                }
       }
```

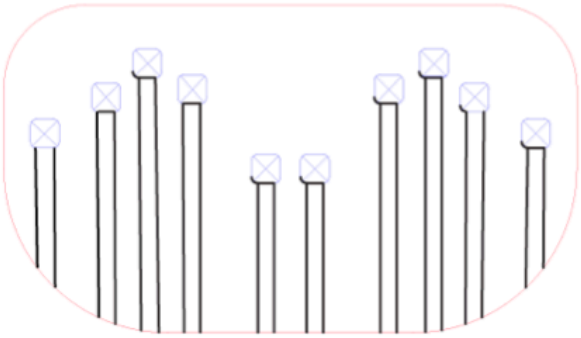
At first, I was concerned about to things: this delay method being to inefficient for sensor debounce, and the chording not having enough delay for it to detect users pressed multiple keys and not several single ones. However, after some empirical testing I determined the optimum value for the delay, and how this approach served both purposes fine for my keyboard. Future work for a more refined prototype could include building an external function that debounces the buttons to more precision if need be.

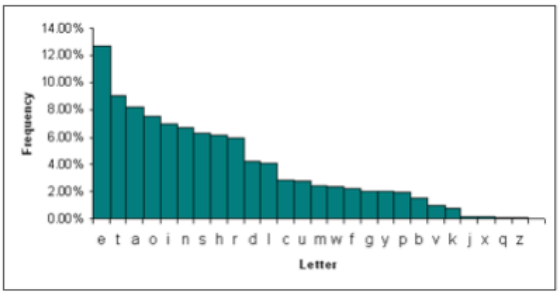
How was the physical device constructed?

Before designing it, I started sketching different arrangements of keys that would make sense. My finalized design included hand prints for the user to place their hand and use one finger per key (as I demonstrate in the video). This is not only ergonomically better after one gets used to it, but it is also quicker in typing combinations.

I then modeled the device in Solidworks and exported a 2D drawing for Adobe Illustrator to laser cut two separate layers.

letter selection based on statistic frequency of letters typed

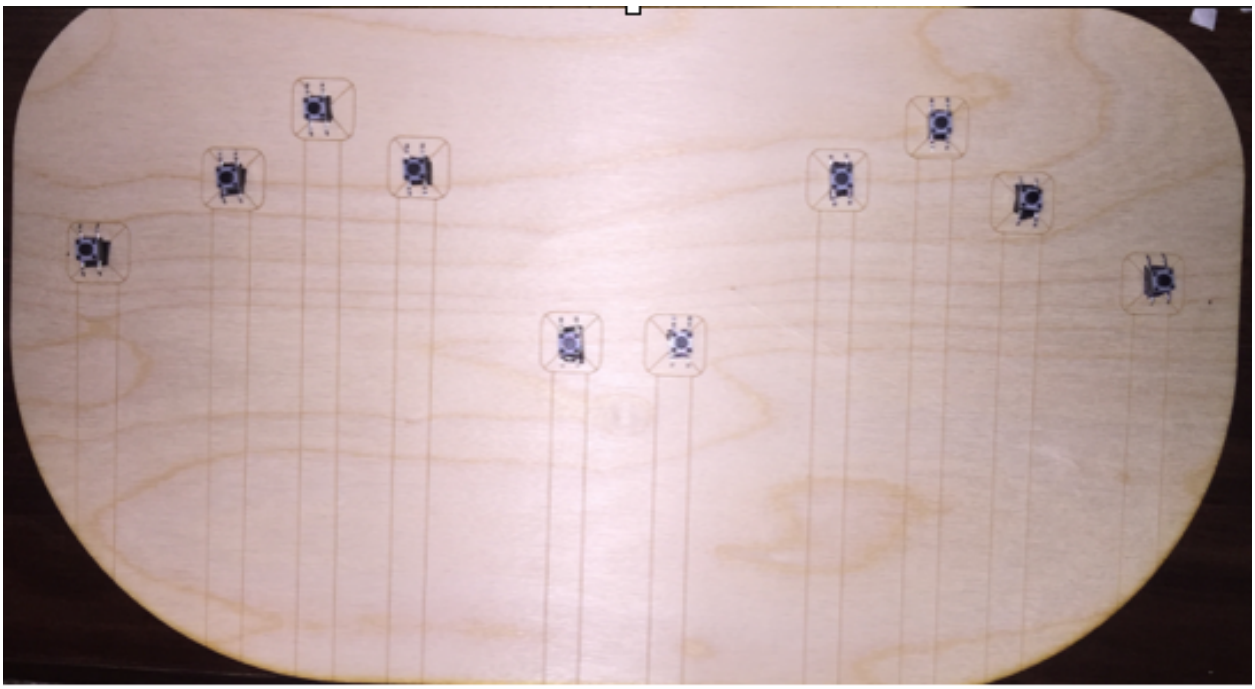




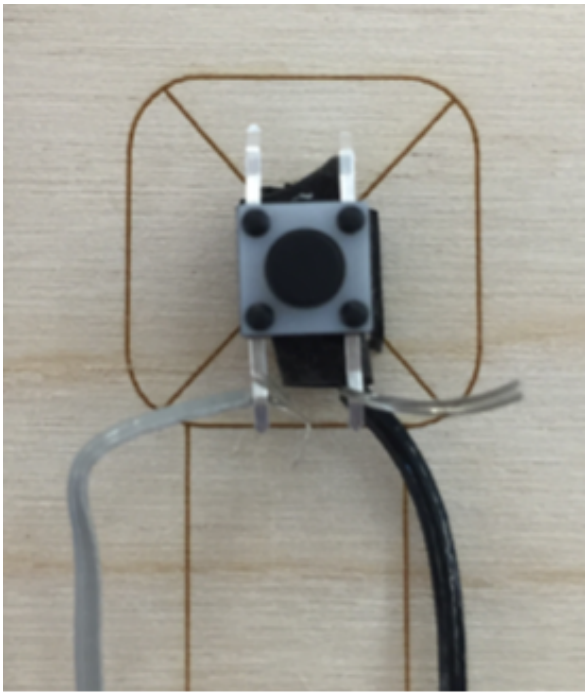
letter chording done through color combinations

The upper layer functioned as the aesthetic portion, not only including the hand raster but also a dual key combination legend, which was correlated to the colors painted on the different keys.

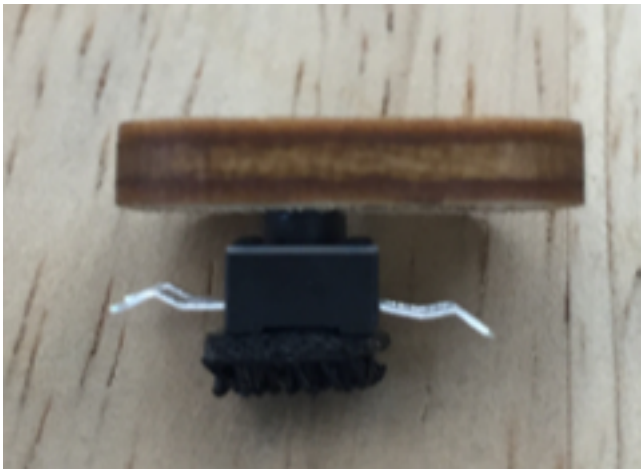
The bottom layer functioned to align the buttons appropriately and also setting guides for how the cables would be routed afterwards.



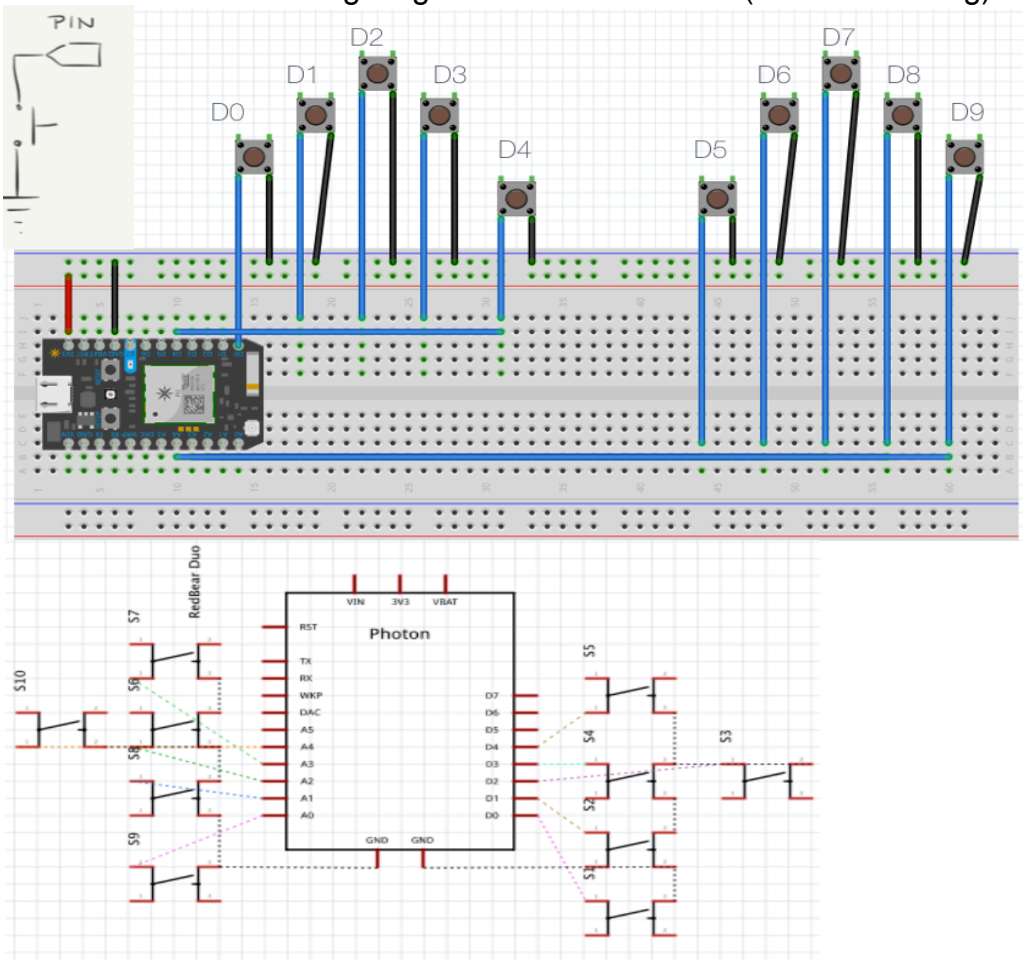
The sensor alignment grid looked something as follows, in aligning the sensor at the center of the square.



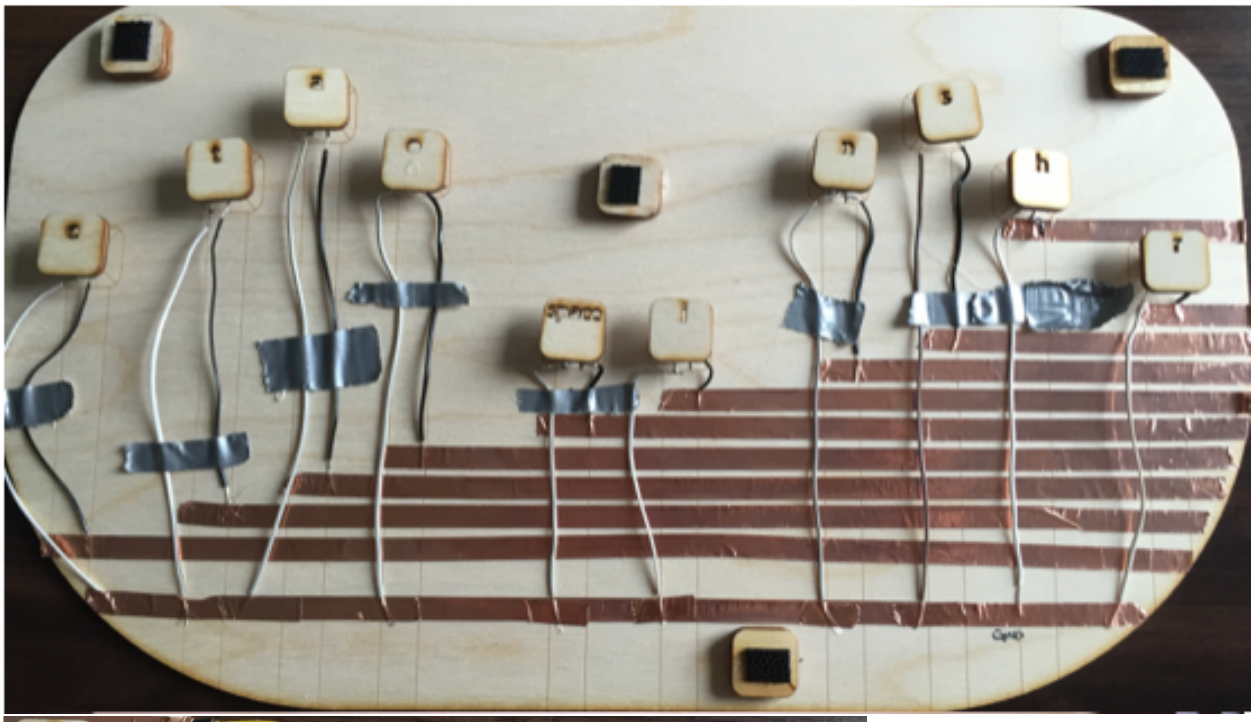
The key was then glued on the button switch and the switch was attached using Velcro to the bottom layer, to allow the switch to pivot and not be rigidly set in case the user pressed the keys from different angles.

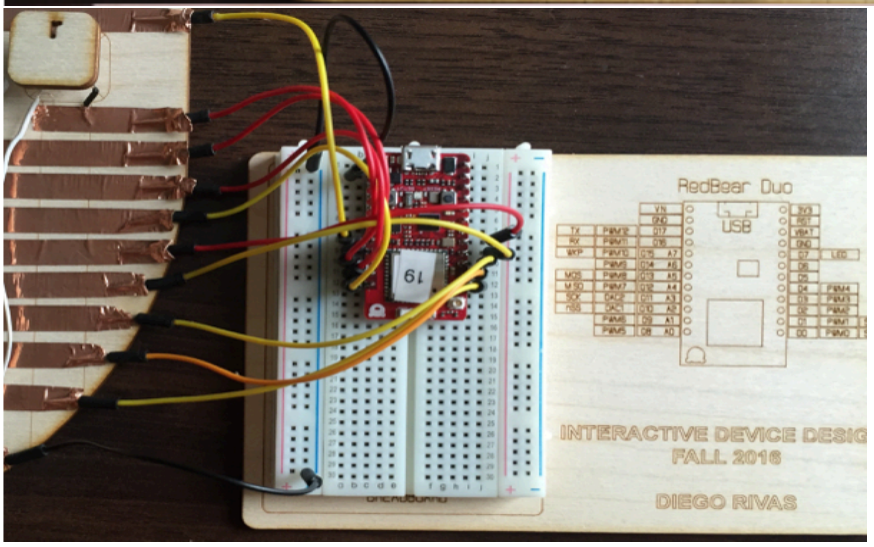


Lastly, the bottom and top layer were also attached using Velcro to access wire routing if needed. The first wiring diagram made looked like (made in Fritzing):



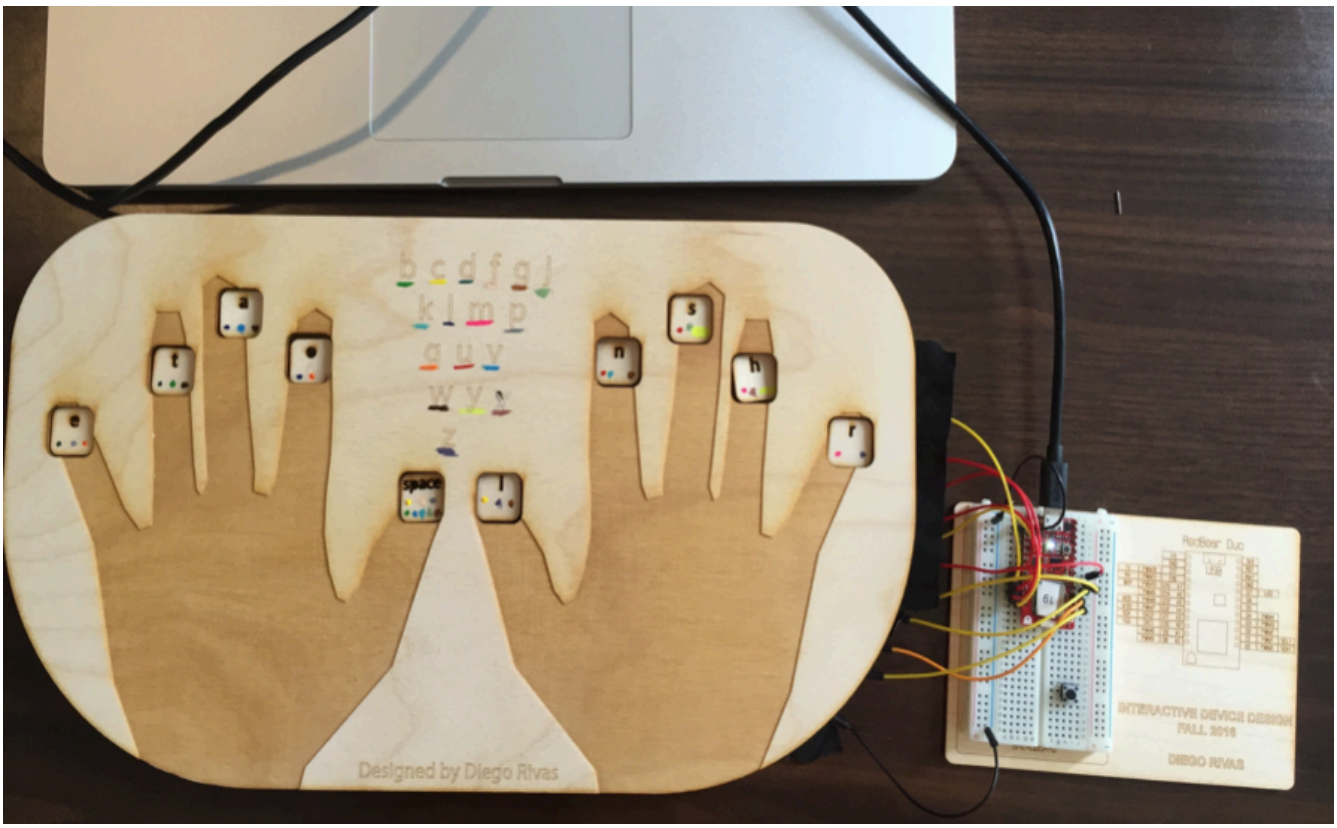
However, in building the actual device one ground like was used to join all the sensor GND terminals, and each switch had its voltage terminal using the internal pullup resistors in the RedBear Duo.





As shown in the images above, the copper lines were used to organize wire better in a perpendicular manner, and the common ground is the bottom most copper line. Each copper line guided to one respective wire input to the Duo. The duct tape shown was placed momentarily to align the wires when soldering them to the switch terminals.

The final device connected looks as follows:



Reflection

This assignment was good as a first step to learn hardware and software design interaction. I learned that the first part to this process is troubleshoot and make the software work, then once that is polished the enclosure should be designed to fit the user needs. I think the assignment was neither easy nor hard, it had a learning curve in designing the code properly but the enclosure design was overall easy.

Video

https://youtu.be/D0llgkKWGEg