IDD HOMEWORK 2: TEXT INPUT DEVICE

BrailleKey

Designing a travel-sized text input device for the visually impaired

Implemented by Michelle Chang

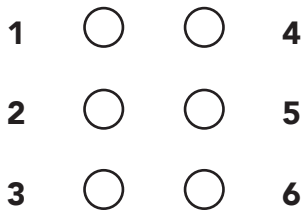
Introducing, BrailleKey

GitHub Repository: https://github.com/mimilei/hw2_mimilei_braillekey

Demo Video: https://youtu.be/ll9pNwTxQsE

BrailleKey is a physical text-entry interface for braille. The text-entry technique employed mimics the existing set of English alphabet characters used by the visually impaired, allowing users to form alphanumeric characters via the six dots that define the braille code system. It is in part based on a number of existing braille keyboards that involve users pressing keys simultaneously on a board of spread-out switches (the appearance is similar to a QUERTY keyboard, but with only four keys per hand).

Each braille character is made up of a specific pattern of six dots. Each dot can either be filled or empty. The dots are numbered as follows:



Accordingly, BrailleKey has a button for each dot. The user's right hand controls dots four through six, and the user's left hand controls dots one through three. BrailleKey also has additional space key (right hand) and newline keys (left hand), which are controlled by the user's thumbs.

I opted to implement this system because I wanted to explore systems of input for less mainstream target users. This was partly inspired by the videos we watched on the first day of class, detailing the many problems in the lives of those with health issues, and the emphasis the instructors have put on designing devices to make the lives of others easier. In brainstorming for this project, I realized that I know next to nothing about how visually impaired people interact with computers and mobile phones, and wanted to learn more about creating a text input device that is both familiar to and convenient for this user group.

The Code

The code powering BrailleKey is quite simple. It assigns eight of the Duo's ports to eight different keys: braille dots one through six, the space key, and the newline key.

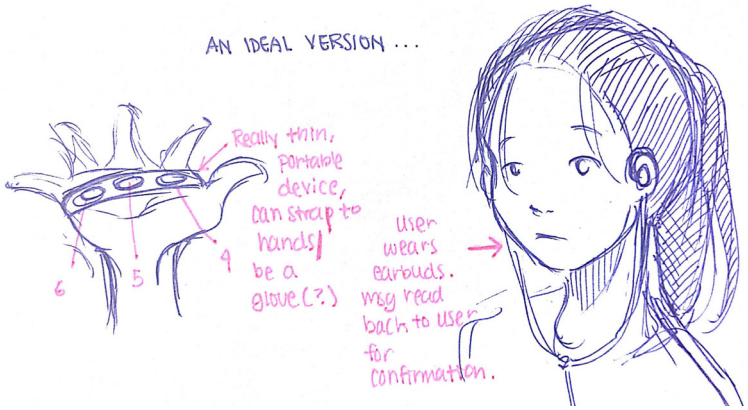
In setup():
The speed of the serial is set to 9600 and all ports corresponding to a key are specified as INPUT_PULLDOWN for their pinMode.

In loop():
In braille, every letter of the English alphabet is assigned a certain pattern of dots (there are six total). The input system employed by BrailleKey requires the user to press the specific pattern of dots for a character simultaneously (or as simultaneously as possible). In every iteration of the loop, the digital value of every single key is read and saved. Through a series of if statements, the loop compares the pattern of dots entered by the user with the existing dot patterns in the braille system. If there is a match, then the corresponding letter is printed out to Serial. Similarly, if the system detects that the space key or the newline key have been pressed (that is, its value is HIGH), a space or newline character is printed to Serial. If the user's input pattern is not recognized by the program, then nothing happens. The loop implements a delay of 200 milliseconds between iterations.

Designing the Physical Device

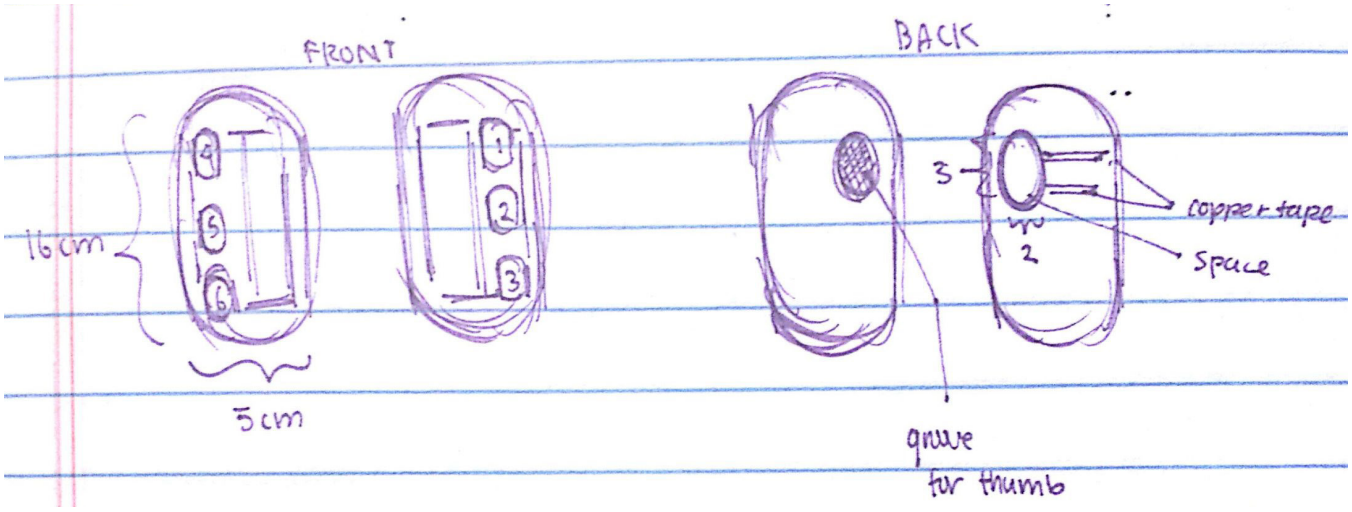
BrailleKey's physical design is crafted to feel somewhat natural and to suggest the potential for portability. While many visually impaired people today do use the classic QWERTY keyboard, with the full set of alphanumeric characters, the classic braille code does have the advantage of using less keys. This makes classic braille an interesting option for a portable text input system, possibly for mobile devices. It also has the advantage of being familiar to the blind, since most visually impaired people know how to read braille.

Since a physical device for the visually impaired should be as haptically distinct as possible, I envisioned an interface that can be held in each hand and easily reached by the fingers. Such a system is small, fairly unobtrusive, and using current technology and very thin switches, could likely be reduced to the thickness of a piece of paper. Although the prototype is wired to the microcontroller, a more polished version could potentially communicate wirelessly.

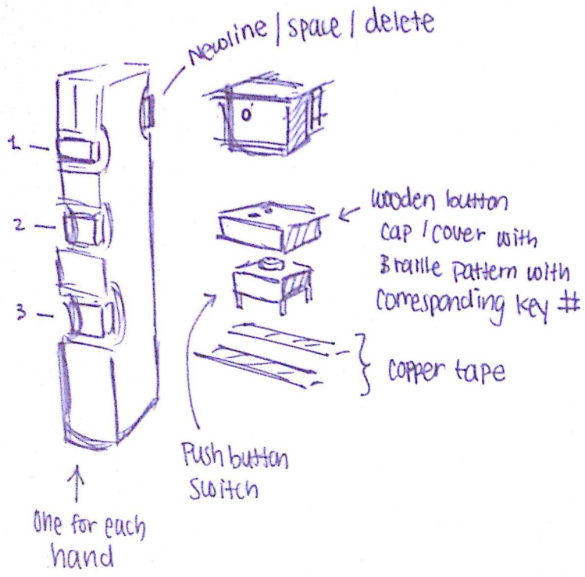


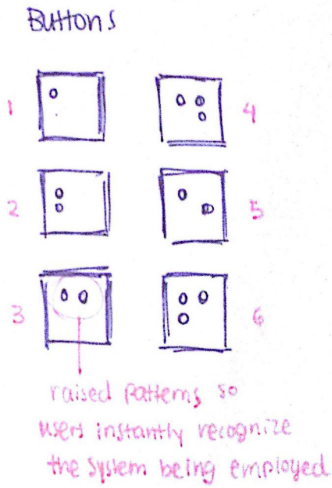
Brainstorming: Design Iterations

During the planning process, BrailleKey was first conceptualized as two wooden boards with three buttons (for the six dots) on each. They were to be mounted on Adafruit protoboards. However, after some thought, I came to the conclusion that such an arrangement would be rather clunky and defeat the purpose of creating a lightweight, portable keyboard.



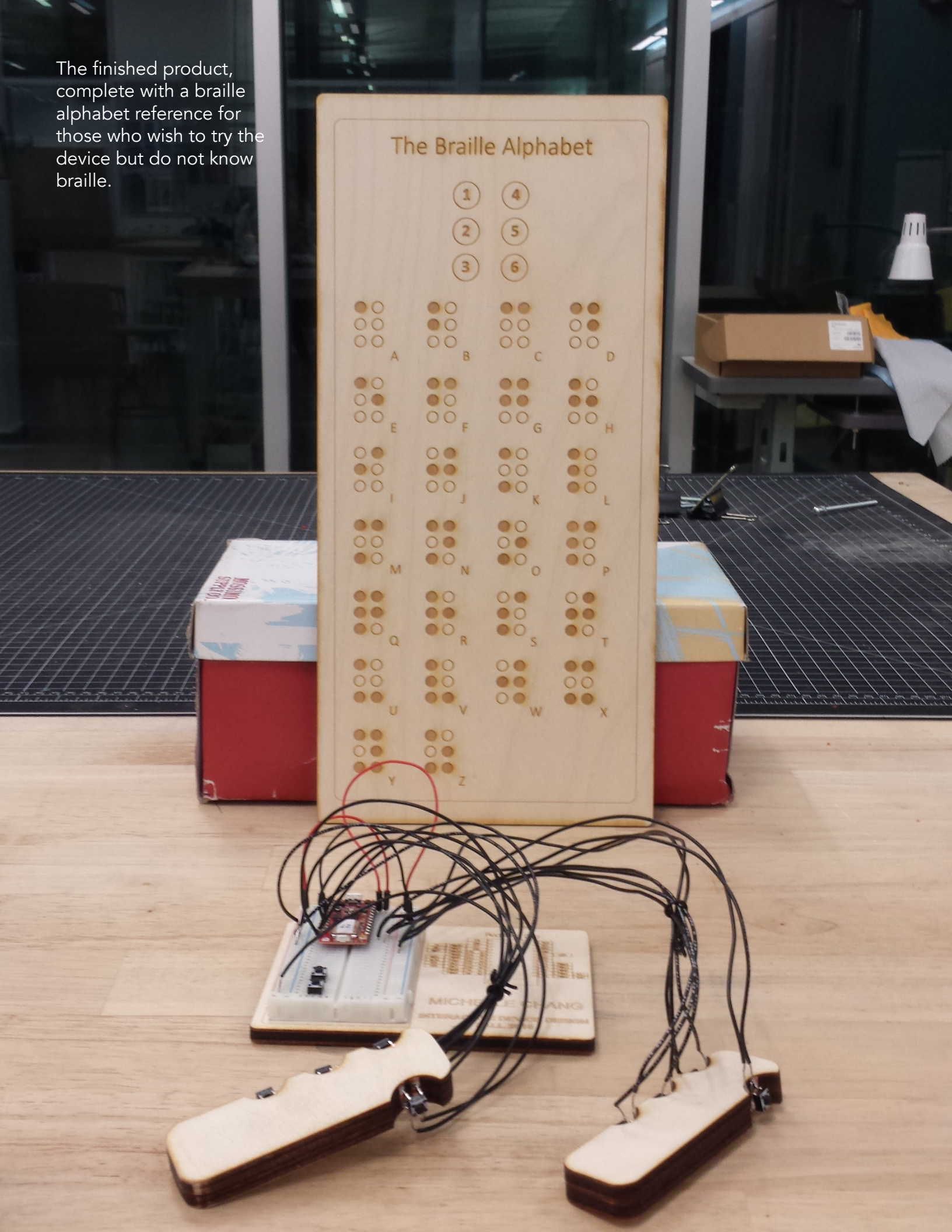
Next, I tried a more compact design that resembles Wii controllers or the handles of a bicycle. The user's fingers should be able to curl naturally around the device so that the curve of the fingers fall comfortably on top of the keys. The base of the user's thumb should be able to rest comfortably on the top edge of the handles, so that all users have to do to press the newline or space keys is to straighten the thumb so that the base of the bone hits the key.



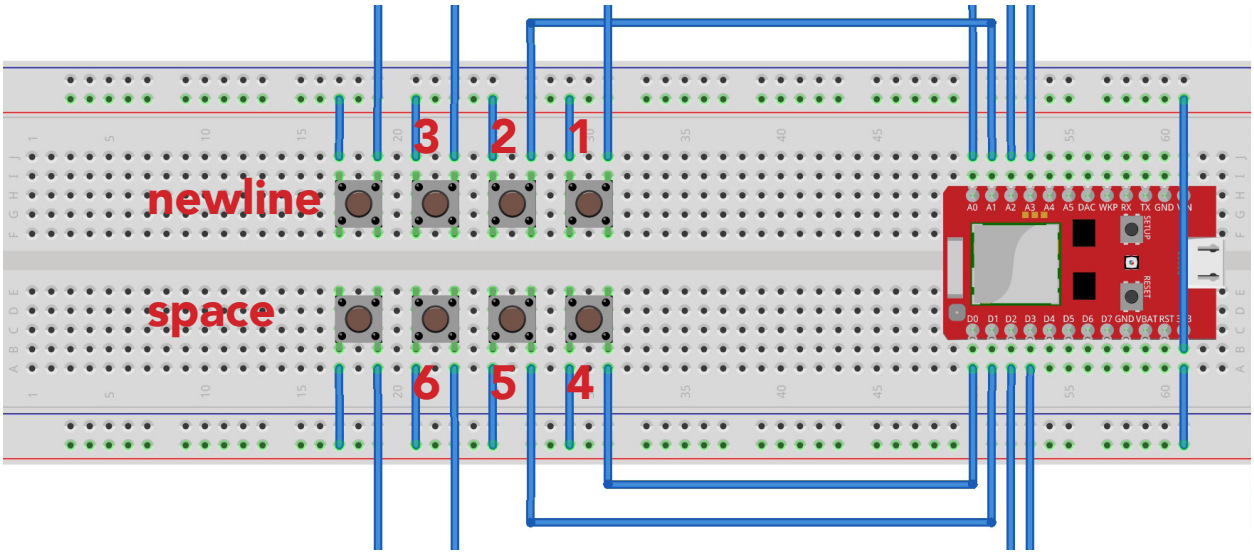


The hand grips in the final prototype consist of two laser-cut pieces of wood glued together, to create greater thickness. Much of the final prototype resembles the above diagram, except for the fact that there are no button caps. This is because, given the size at which the handles were cut at and the depth of the grooves in the handles, button caps would protrude unnaturally far from the main body of the handles and likely act as more of an obstruction than an aid. Input is read via push-button switches soldered to copper tape and attached to the main breadboard by wiring.

The finished product, complete with a braille alphabet reference for those who wish to try the device but do not know braille.



The following is a depiction of the electronics when set on a breadboard:



Reflection & Future Improvements

I learned a lot (and had a ton of fun) doing this assignment. I was actually extremely nervous and worried before beginning because I've never built a "complete" device by myself before, and had no exact mental model or experience to draw upon regarding messing with switches and bending them to my will. Some new things I learned and encountered include preparing wires to for prototyping (twisting, tinning), using a multimeter to test for good connections, and designing and crafting a device for handheld operation. This project was great practice for refreshing my soldering and electrical skills.

I've learned a lot about how to improve my design from this first prototype. (There are actually more problems with my idea than I initially expected!) Although I tried to create a device that felt natural, it in fact feels rather awkward in practice. As it turns out, pressing buttons with just the curve of your finger is rather difficult and feels strange. The sizing of the wooden grips probably need to be tailored to the user. I found that I would prefer them a little bigger and more spread out, as my hands felt rather cramped after using the device for a while. In practice, there would probably be different devices for different hand sizes (rather like clothing). My roommate, whose hands are smaller than mine, tried the device and found it just right.

Another important thing I learned is that the method of text input I chose (pressing the right pattern of buttons simultaneously) requires a lot of coordination and concentration (most people don't have a lot of practice using both their index and ring fingers at the same time). Part of this may be due to the lack of key covers on top of the switches. Key covers increase the surface area accessible to the user, so that they have to put less effort into key presses. However, much of the issue lies in the fact that key presses must be purposeful and backed with sufficient force. Just pressing keys casually often leads to the system misinterpreting the user's intentions because the force is often too light in certain areas. Because users must not only press more than one key simultaneously, but also

put the same amount of effort into each finger, the system puts more strain on users overall than the classic QWERTY keyboard. More sensitive switches might help with this.

Some great ways to expand on this device include:

1. Adding a delete function.
2. Adding a text-to-voice feature. After text is entered, the system will read the message back to the user so they can make corrections as needed. Without haptic feedback, the only other way for the visually impaired to easily correct their input is to hear it read back to them.
3. Redesign the grips to fit more naturally with the human hand.

Overall, I thought this assignment was just the right amount of challenging and enjoyable. I picked up a lot of new information regarding technology for the visually impaired in the midst of the Mobile Revolution. I found Stanford's braille keyboard for the iPad to be quite interesting. The assignment felt difficult and worrying at times, but not so much so that I felt like falling into despair!