Jonya Chen
CS 294-98
Interactive Device Design
HW2
September 9, 2016

Tap Code Text Entry Device

For my implementation, I decided to use tap codes to encode a user's message.

In this homework assignment, my goal was to implement a text entry device that would allow a user to type our any message he/she wanted using the lower-case alphabet and a space character. The only restriction was that all inputs had to be discrete switches and we could only use a maximum of 10 total switches. Typed characters are to be sent over a serial communication and displayed on the PC's serial terminal. For my implementation, I decided to use tap codes to encode a user's message. As a result, I needed a push button to represent both the row and column indices, as well as a button for "ENTER" (when the user is done inputting a character) and a "SPACE" bar to the space character. Through the rest of the documentation, I will detail how the code works, how the physical device was constructed, and my overall thoughts on the project.

Tap Codes

Tap code is a way to encode text messages on a letter-by-letter basis in a very simple way. The coding method has been commonly used by prisoners to communication with each other, usually by tapping call walls, metal bars, or pipes.
The tap code is based on a Polybius square, with a 5x5 grid representing all the letters of the Latin alphabet, except for K, which is represented by C. The tap code table can be seen on the following page (Figure 1).

TAP CODE	1	2	3	4	5
1	A	B	C/K	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figure 1: Tap Code Key

Each letter can be communicated by only using 2 numbers – the first designating the row and the second designating the column.

Each letter can be communicated by only using two numbers – the first designating the row and the second designating the column.
I chose to use the tap code implementation because of simplicity and ease of use. Using only two buttons, "ROW" and "COL", a user can quickly specify the desired row and column to point to a character. In comparison, an encoding method like Morse code would be harder to send because it would need to require the ability to create two differently sounding "taps" and it's also difficult to remember each letter's code. Tap code can be more easily encoded and decoded.
In addition, I also included another push button for the "ENTER" option so that the user can signal when the row and column buttons have been pressed the desired amount of times. Then, the character can be decoded from the number of presses of the "row" and "col" button. Since our design criteria also specified that the user would need to be able to type the space character, I added in a fourth push button, "SPACE", to the design.

Implementation

Hardware

For the hardware portion of my text entry device, the wiring was quite simple. Since I only required four push up buttons total, I just had to ground each of them and connect the other end of each button to an input pin on the Red Bear Duo. Seeing as the Duo accommodates for an input pull up resistor, there was no need to hardwire in a physical resistor in my circuit. As a result, the only components on my breadboard would be the push buttons, which I later ended up soldering them to their own wires off the board.
In Figure 2 below is a breadboard wiring diagram for the circuit:

I ended up designing a little box that neatly stores all wires and connections under the lid so that the user won't have to view them when using the product.

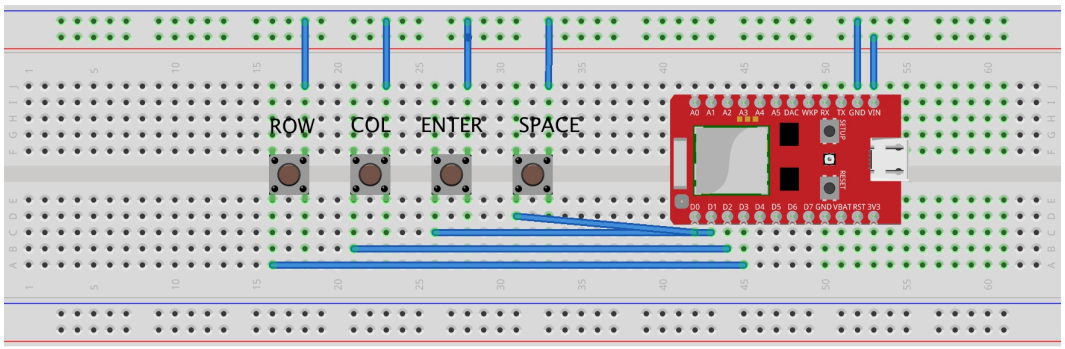

Figure 2: Breadboard View of Circuit

Software

In terms of software implementation, most of the code is spent on debouncing the push buttons. Since switches don't change instantaneously from on to off or vice versa, we must account for this discrepancy and confirm that the button is actually on/off before performing the respective action. The method I used has a minimum delay between presses to debounce the circuit and therefore ignore noise. As a result, my code had to keep track of the current and previous reading of each button, as well as the last time the button was pressed.

Other than the debouncing, the rest of the code is fairly straightforward. I could read in the value of each push button by using digitalRead (and also making sure to set each one as a button using internal pull-up resistor). Furthermore, I had to include a nested switch case statement for when the user pressed "ENTER" to correctly decode the character. This was done according to the tap code key in Figure 1.

My code is also pretty heavily commented so hopefully it's easy to understand and walk through.

The Device

When designing my device's enclosure, I wanted it to be clean and simple. A user should be able to look at my device and know how to use it with little to no confusion. As a result, I decided to create a separate enclosure from my microcontroller/breadboard so that a user wouldn't have to deal with seeing any wires or unnecessary parts.

I ended up designing a little box that neatly stores all wires and connections under the lid so that the user won't have to view them when using the product. I used the laser cutter to create the walls of

the box, as well as engrave the key on a panel. Therefore, a user could easily look at the key and determine which buttons to press. I also laser cut four labels and hot glued them on to each push button, which makes them distinguishable and very easy to press.

I also soldered each of the push buttons to my own created wires, attached them to a ribbon wire, and used a female header to plug the wires into the breadboard. Overall, the design was simple but clean so that users of all ages and backgrounds can use the product.

Below are a few images of the final device (Figures 3-5):

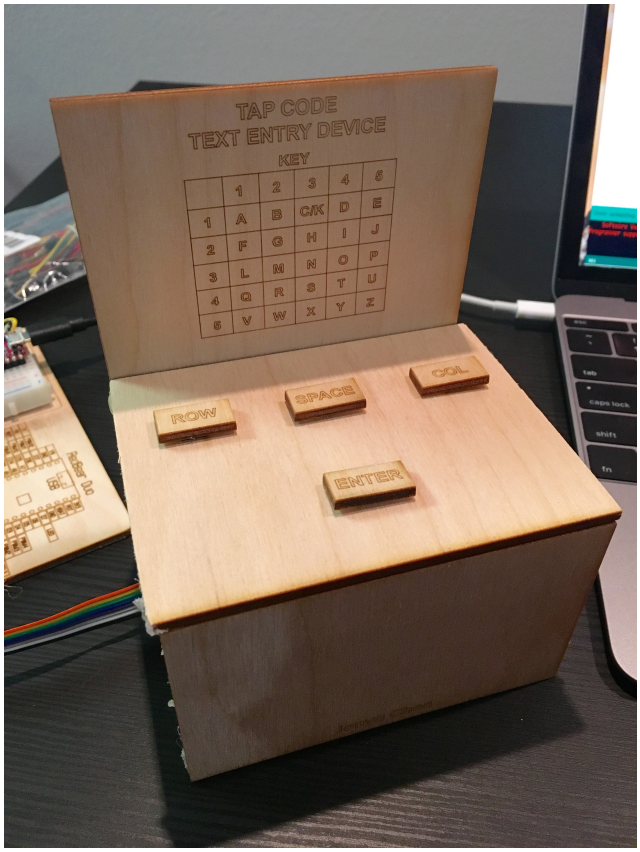

Figure 3: Overall view of final product

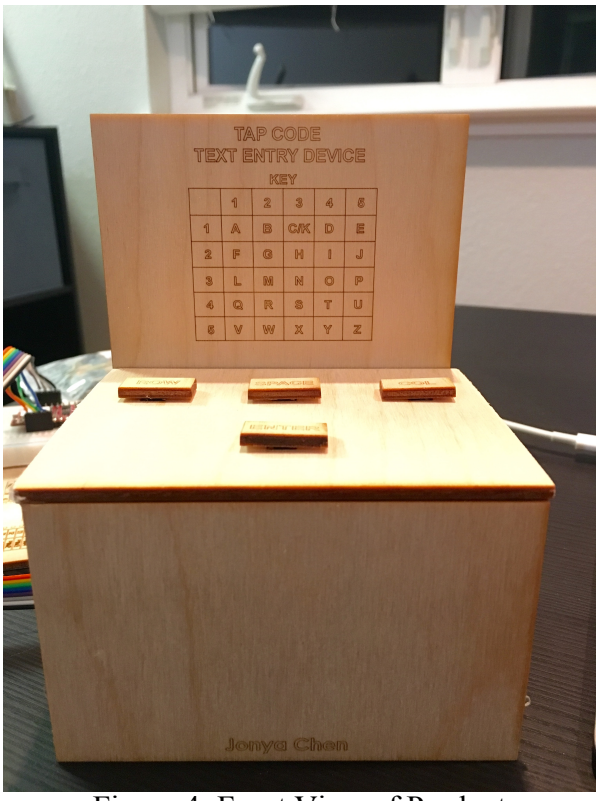

Figure 4: Front View of Product

Figure 5: Top View of Product

Conclusion

Overall, I enjoyed how open-ended the assignment was and allowed us to pretty much design whatever we wanted with little restrictions. As a result, we were forced to plan out our design before implementing everything. I also liked how I could breadboard the whole circuit, code it up and make sure that it works all fully before figuring out how to enclose to design the physical product. This process left room for multiple iterations and ways to make the product better as I worked.

In terms of level of difficulty, I would say it's about just right. The coding wasn't all too challenging, but that might have been because of the design idea that I chose. What I found surprising was how long I actually spent working on building the physical device. What started out as a simple design ended up actually taking a while to build because of how precise everything had to be and how clean I wanted my wires to look. I had a bit of trouble at first because I had my box overlaying on top of my microcontroller but then I realized this made my wires too confusing and the box wouldn't stay flat on the surface so I ended up changing the design at the end. This iterative design process was interesting to deal with but worked out in the long run as my final product was much better than the one I had in my first run through.

Links

GitHub repository URL: https://github.com/jonyachen/HW2_jonyachen_TapCodeTextEntryDevice
YouTube demo video URL: https://youtu.be/68uQESQxRxc