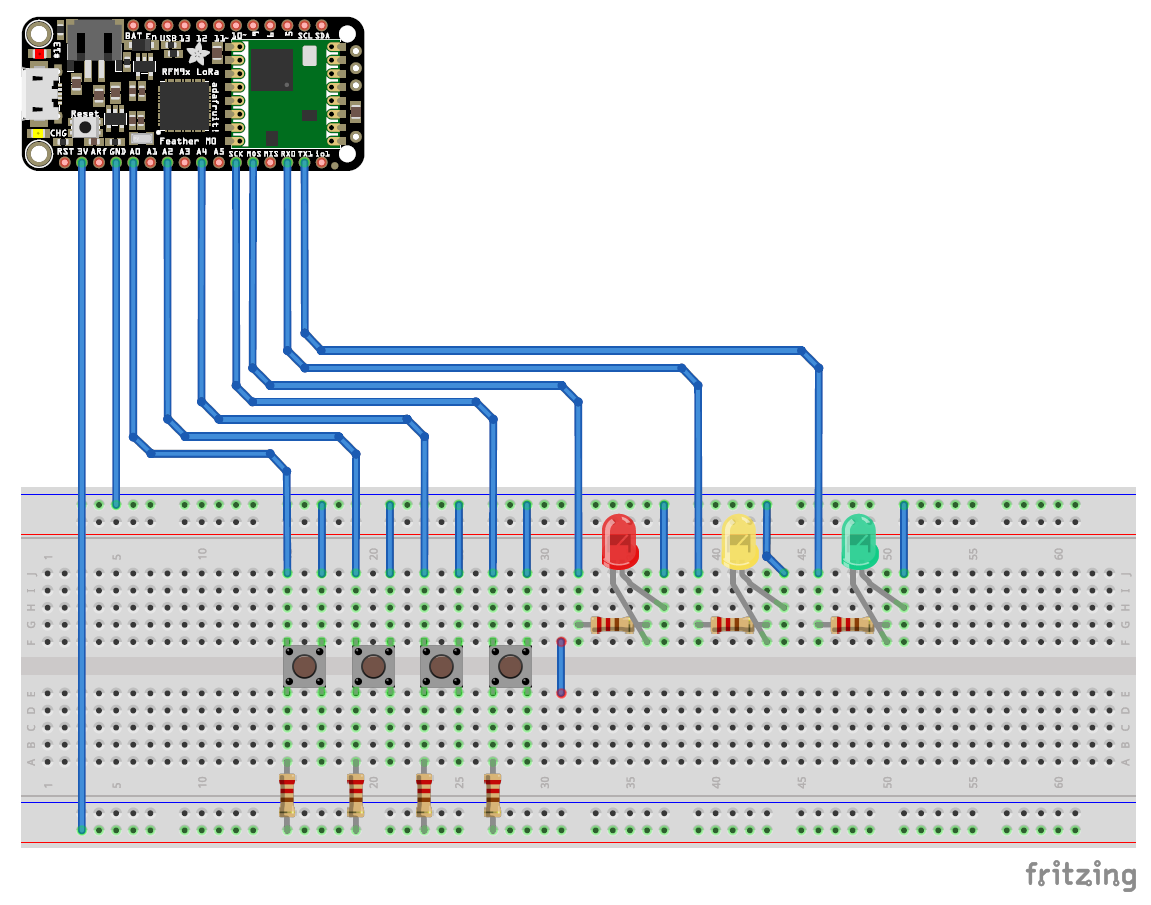
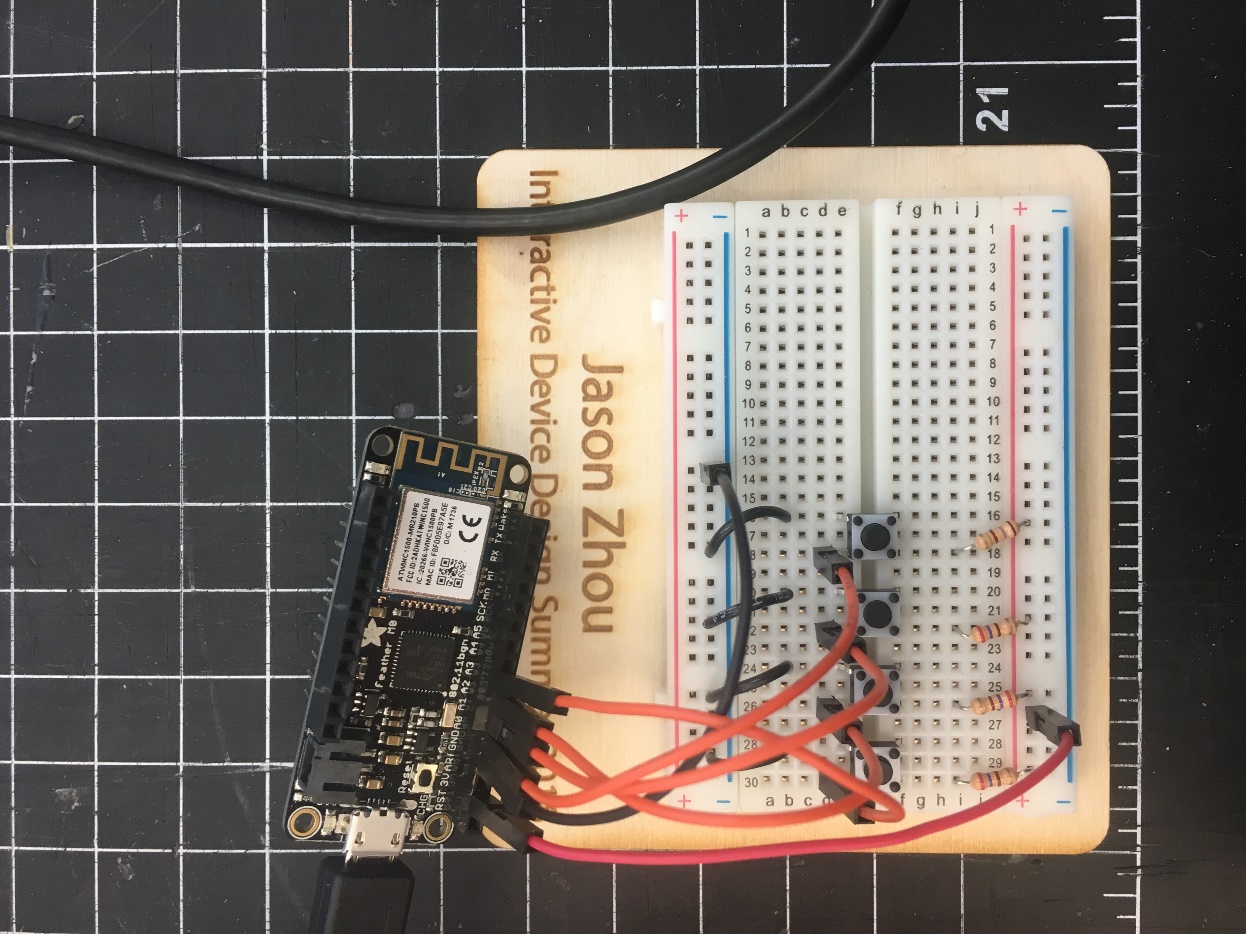
**Interactive Device Design Summer 2018**

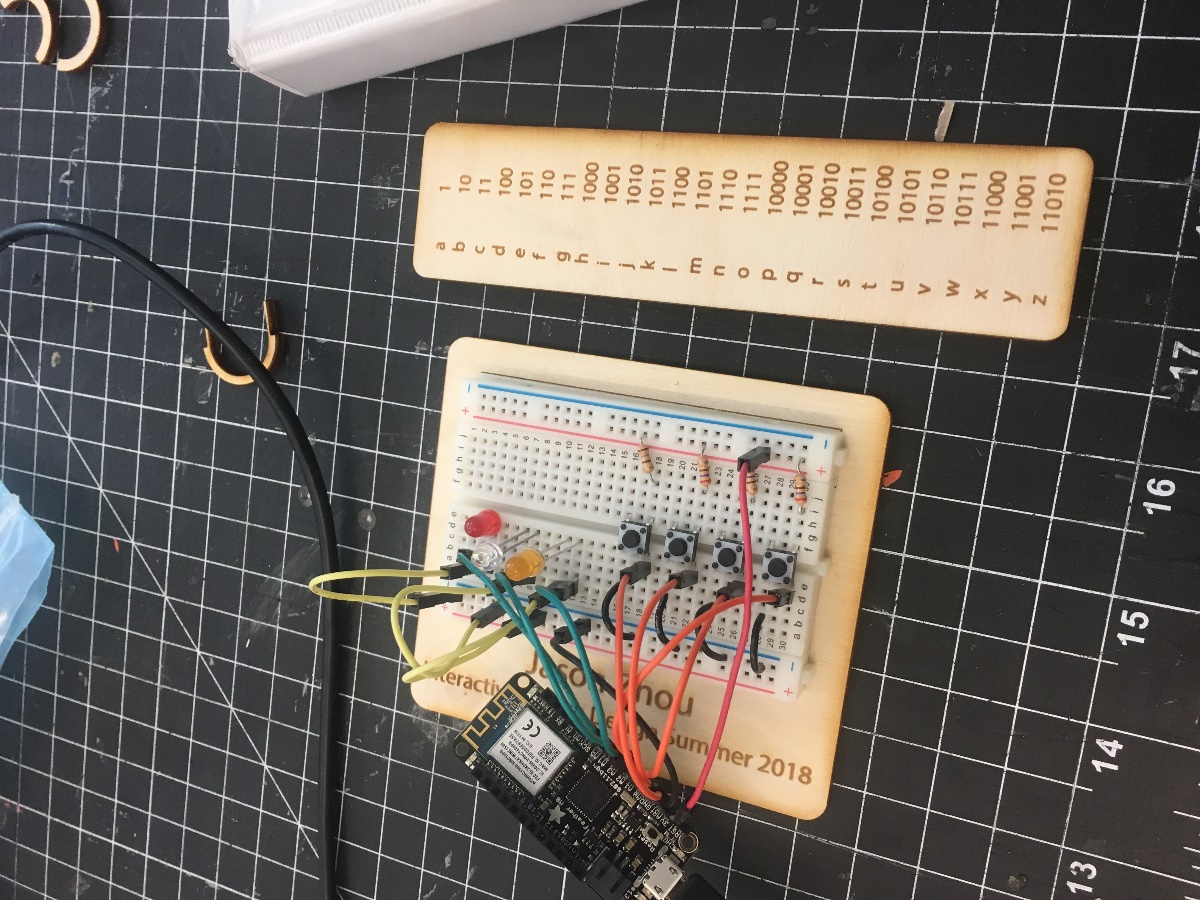
**Homework 2: Text Entry Device**

**Jason Zhou & Cameron Riley**

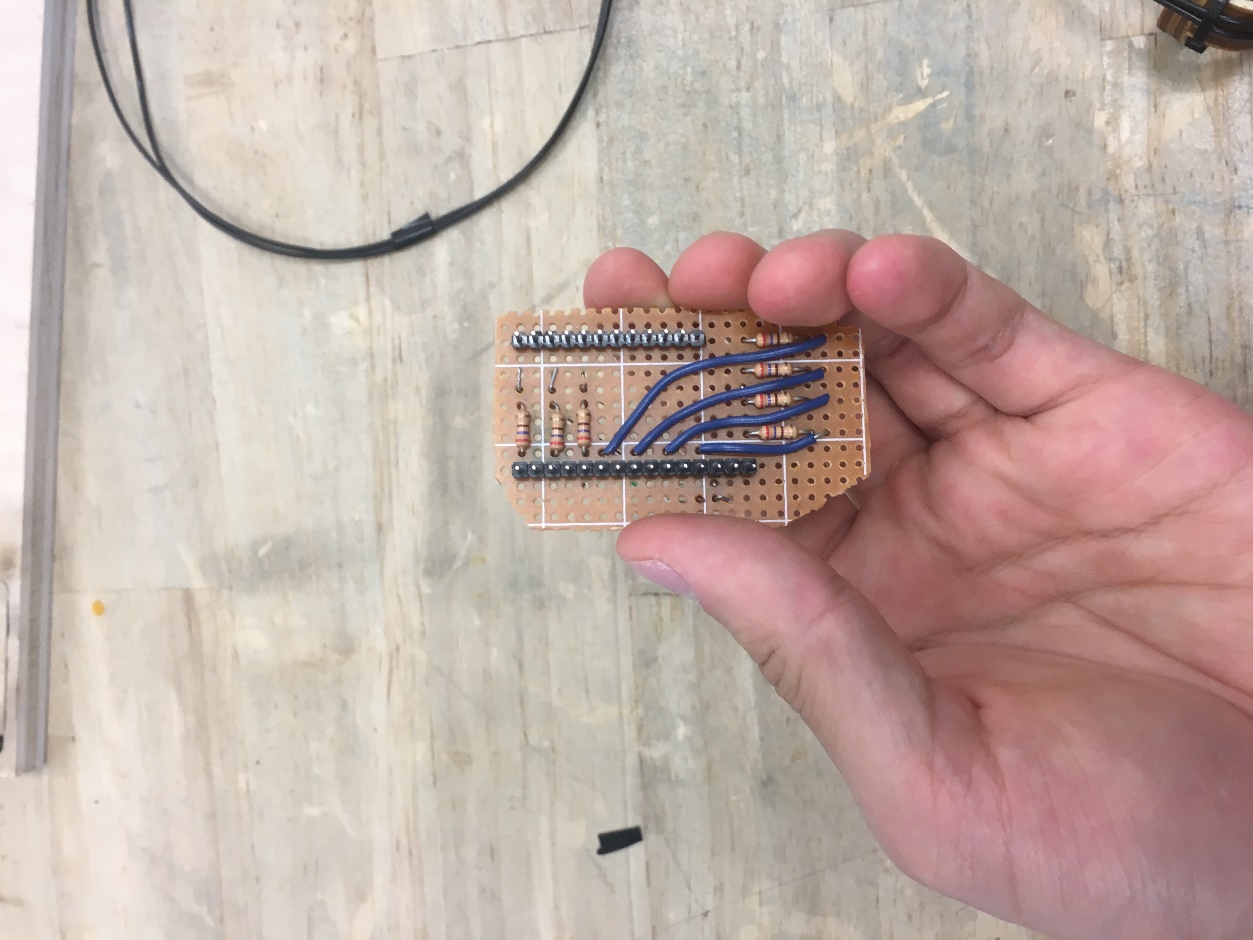
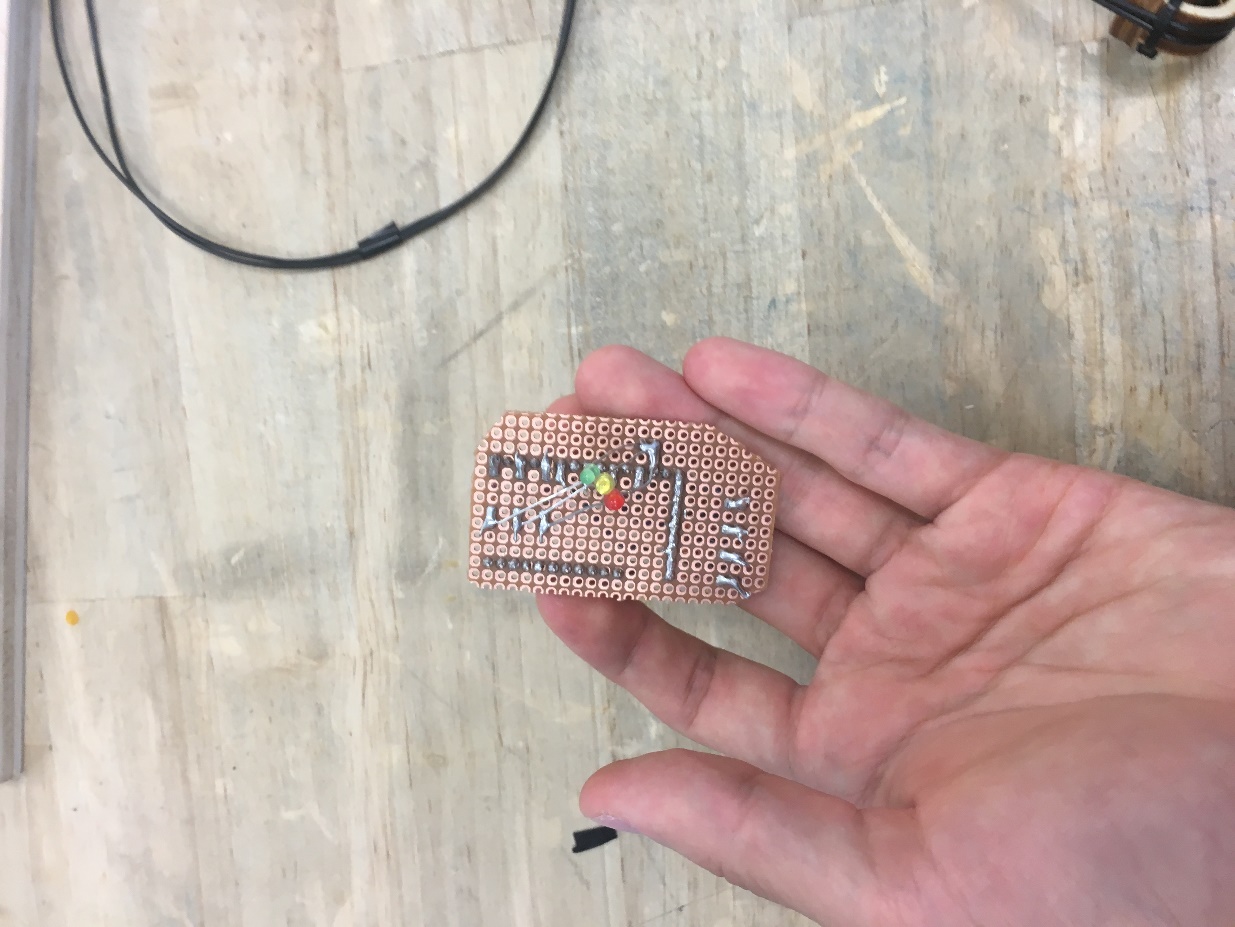
**Wiring Diagram & Project Images:**

**Initial breadboarding work of button mechanism:**

**Final breadboarding work including instruction tablet and LED indicators:**

**I**

**Front and back of perfboard circuit**

****

**Wrist strap work in progress:**

****

****

**Fingertip button mechanism work in progress:**

****

****

**Final Device Pictures:**

**Github URL:**

<https://github.com/IDD-su18/hw2-jasonxzhou>

**Writeup:**

The code/project utilizes 7 microcontroller pins total, with 4 as input and 3 as output. The 4 input pins are connected using pull-up resistors to the 4 buttons that correspond to the user’s finger. As each button is pressed, a change in voltage for that respective pin is detected. The index and middle finger buttons are mapped to the binary bits, the fourth finger is mapped to the enter/capslock key, and the fifth finger button is for entering a space.

The code initializes an integer variable called “key” to the value 0; this variable stores the letter to be inputted. Note that the value 0 does not correspond to any letter. The code now awaits input; if the index button is pressed, “key” is left-shifted by 1 bit (analogous to adding a 0 to the right of the current number). Similarly, if the middle button is pressed, “key” is left-shifted by 1 bit and incremented by 1 (analogous to adding a 1 to the right of the current number). The code will await user input until either the enter button is pressed, in which case the value “key” is converted to the corresponding letter (using a hardcoded translation function), or until the space button is pressed, in which case a space char is outputted and “key” is simply reset to 0. The capslock function is implemented by checking if “key” is 0 when the enter button is pressed (i.e. if a user simply presses enter without pressing any of the binary bit keys). Lastly, the code handles two types of input errors. If more than 5 bits are entered (even prior to the user pressing the enter key), the code will reset “key” to 0 and prompt the user to try again. If 5 bits are entered but the pattern does not match one of the hardcoded translations (as not all 5 bit numbers are used/mapped), then the code will prompt the user to try again as well.

The three LEDs (green, yellow, and red) indicate the device’s state. The red LED toggles on/off according whether capslock is engaged or not; the green flashes to indicate a successful character entry, and the yellow flashes to indicate that an error has occurred.

**What text entry technique you chose and why:**

**How you implemented character recognition (i.e., how your code works):**

**How you constructed the physical device:**

**Reflection:**

**Link to video:**

**Link to your code**

<https://github.com/IDD-su18/hw2-jasonxzhou/blob/master/hw2/hw2.ino>