NotherButton Final Report

Anders Pitman

Overview

The purpose of this individual study is for me to gain experience building a wearable Internet of Things (IoT) device that could be useful in the real world.

The NotherButton is envisioned as a wearable Bluetooth device which aims to add additional hardware IO capabilities to smartphones. The primary focus is to add buttons the user can

configure to control different functionality of their phone, such as media pause/play/skip, volume, screen brightness, etc. Other novel uses could be implemented. Imagine, for example, programming a button to interpret Morse code for sending text messages. This device could also have applications in improving device usability/accessibility for those with disabilities, by programming buttons to perform tasks that are difficult for them to accomplish using a touch screen.

The scope of this semester project was to create a proof-of-concept prototype of the NotherButton.

Platform

Hardware

The NotherButton is built on the Nordic nRF52 Bluetooth LE platform [0]. Specifically, I used a Redbear Labs MB-N2 bluetooth low energy (BLE) module [1]. There are two primary advantages to using the nRF52. First, it includes not only a BLE radio, but also a powerful embedded ARM Cortex-M4 processor [2]. This greatly simplifies the hardware, as I didn't have to worry about getting a primary board talking to a separate BLE board. The second advantage is that the nRF52 is well supported by the open source Zephyr RTOS (see below), which I wanted to use for the project.

Software

As mentioned above, one of my goals for this project was to learn how to use the open source Zephyr real-time operating system (RTOS) [3]. Nordic provides their own SDK for the nRF52, but it is not open source, and since Nordic also supports the use of Zephyr it seemed like a great choice.

NotherButton PCB

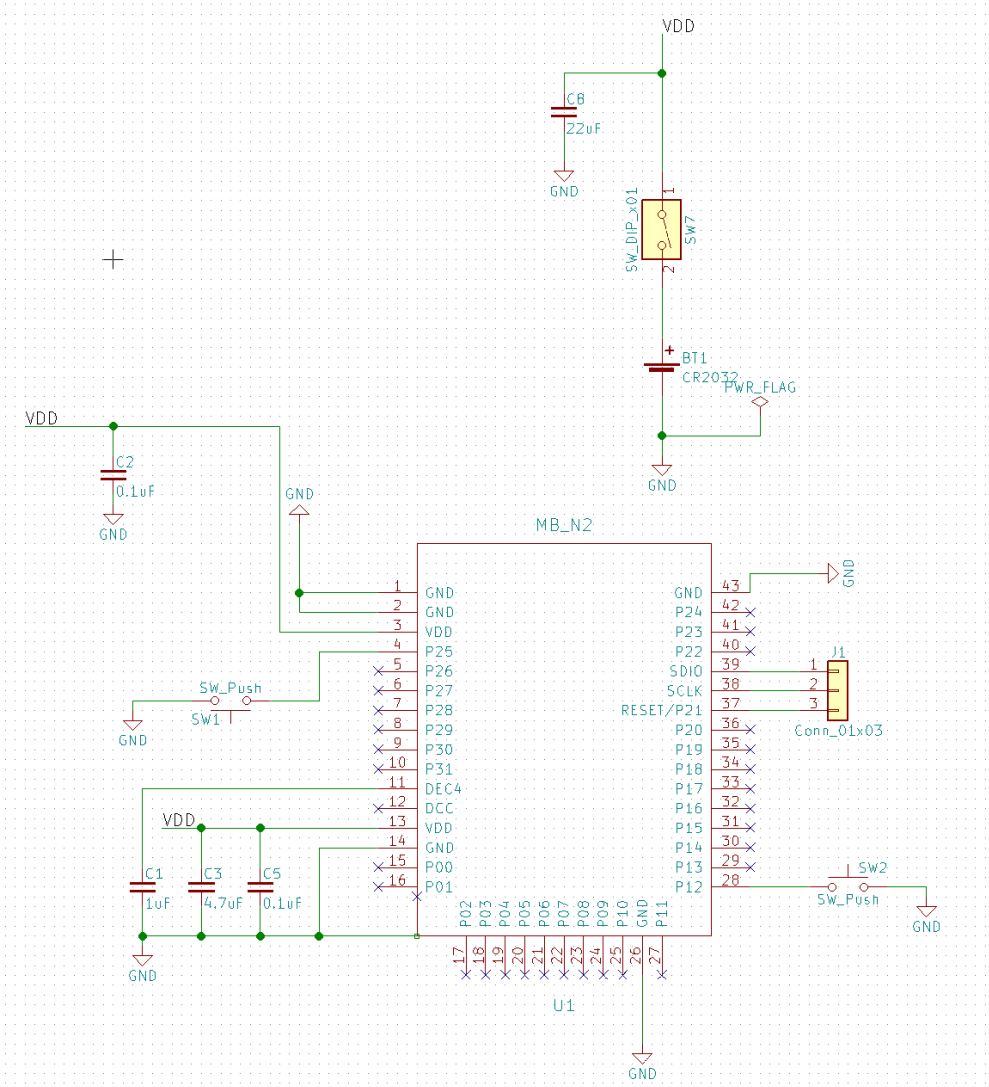
For all of the PCB design work, I used the excellent KiCad [4]. This really is a fantastic piece of software. I have never designed a PCB before, but was able to get up to speed fairly quickly.

Schematic

For the circuit design, I started with the Redbear MB-N2 reference circuit [5, pg 4]. I tried to strip it down as much as possible. This was a tricky process since I have very limited electronics experience. The Redbear forums were very helpful. They linked me to the nRF52 product

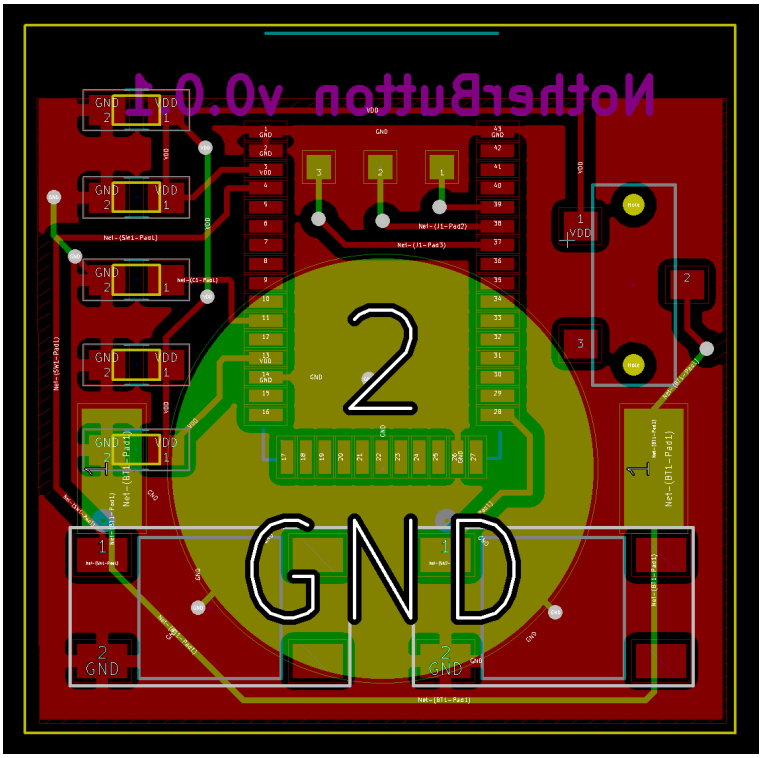
datasheet [6]. I eventually decided that a lot of the circuitry would not be necessary for my use case, which resulted in the simplified circuit shown below.

The overall design is rather simple. Just the basic circuitry required for running the SOC, a holder for a CR2032 battery, a power switch, and 2 push buttons. The BLE module provides everything else, including an antenna.



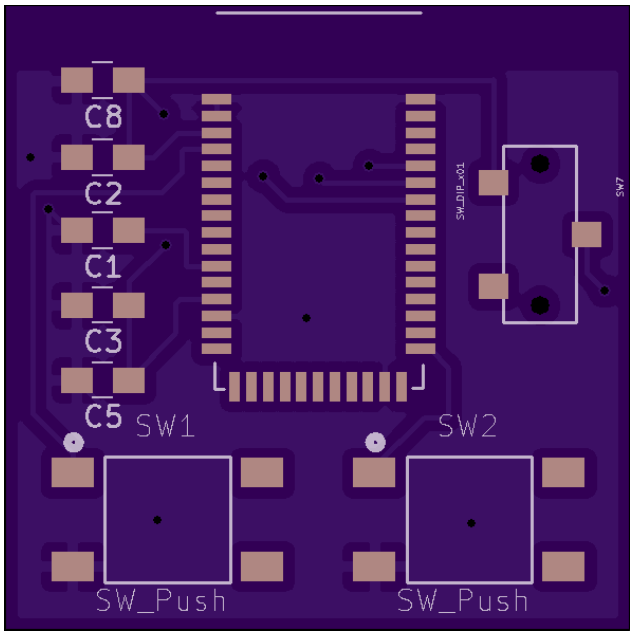
Layout

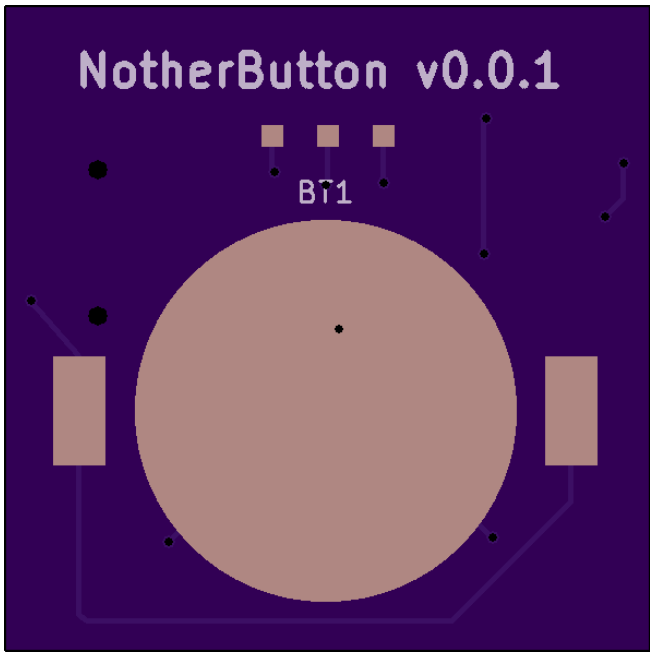
The layout turned out to be more difficult for me than the schematic. I just about drove myself crazy trying to optimize to reduce wire crossings. It's amazing how many degrees of freedom just two layers give you. The end result is shown in the figure below.



Ordering

I used the Oshpark service [7] for ordering the prototype PCBs. This worked out quite well, and had a fairly fast turnaround. I would definitely use them again. Below are images of the renders they sent me when I submitted my order.

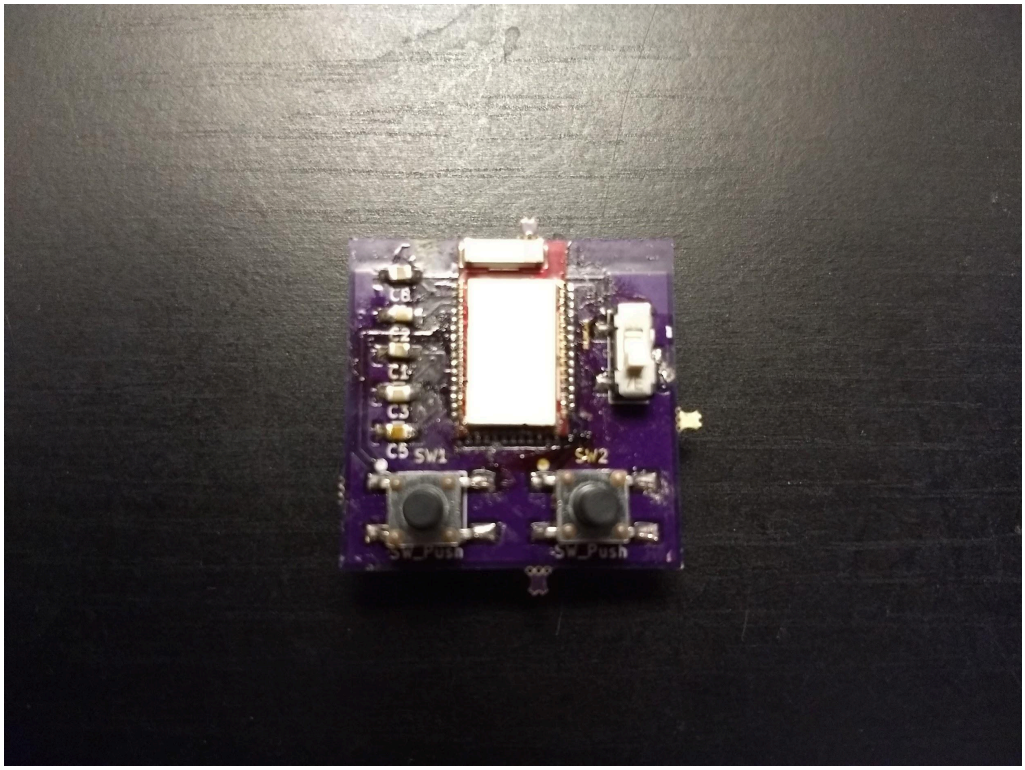


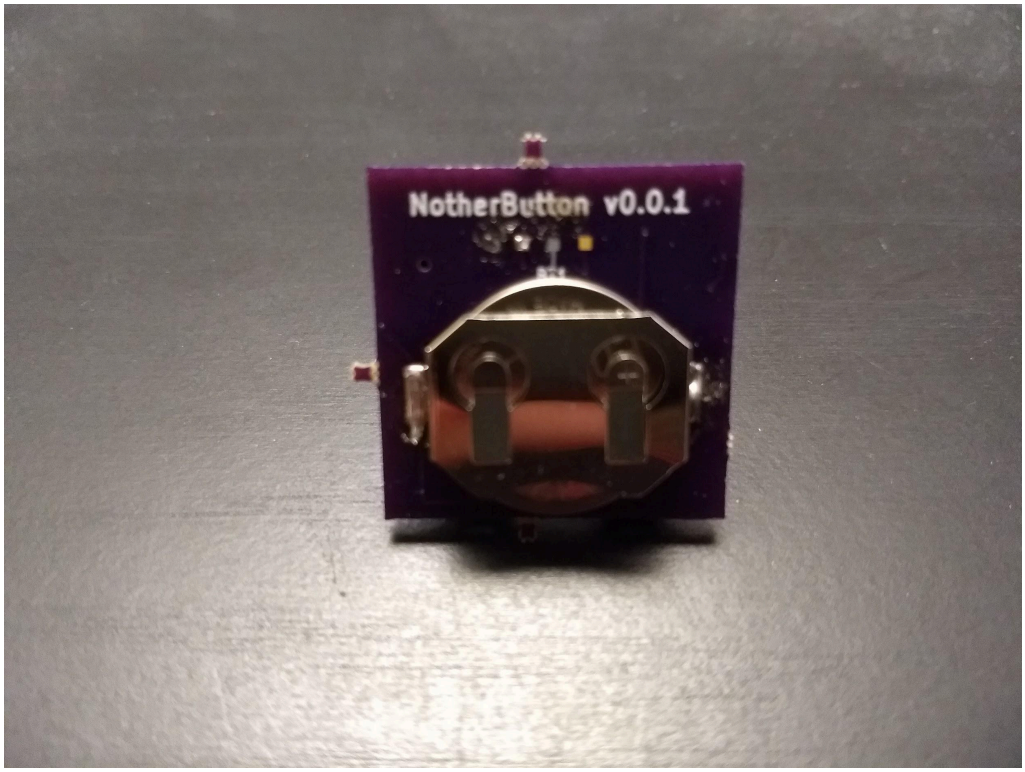


Assembly

I sourced all of the other components, which mostly consisted of resistors, capacitors, buttons, battery holders, and switches, from Digi-Key. I hand-soldered the board. The trickiest part of this was handling the fine pitch on the BLE module, but I got fairly good at drag-soldering [13]. The final result is shown below. Unfortunately, after all was said and done, the board didn't work. I couldn't even get it to flash. The first mistake I realized was that the programming pads on the back of the board are way to small. This made it hard to debug. In the process of trying to solder wires onto the pads, I destroyed one of them. This left me with no great way to debug the board. I had another BLE module and tried to assemble another one (Oshpark sends 3 samples per order), but completely messed up the soldering on that one. Overall, my error is obvious. I should have prototyped my circuit design on a breadboard before designing a custom PCB. Trying to shortcut the process and rely on luck was not an effective strategy.

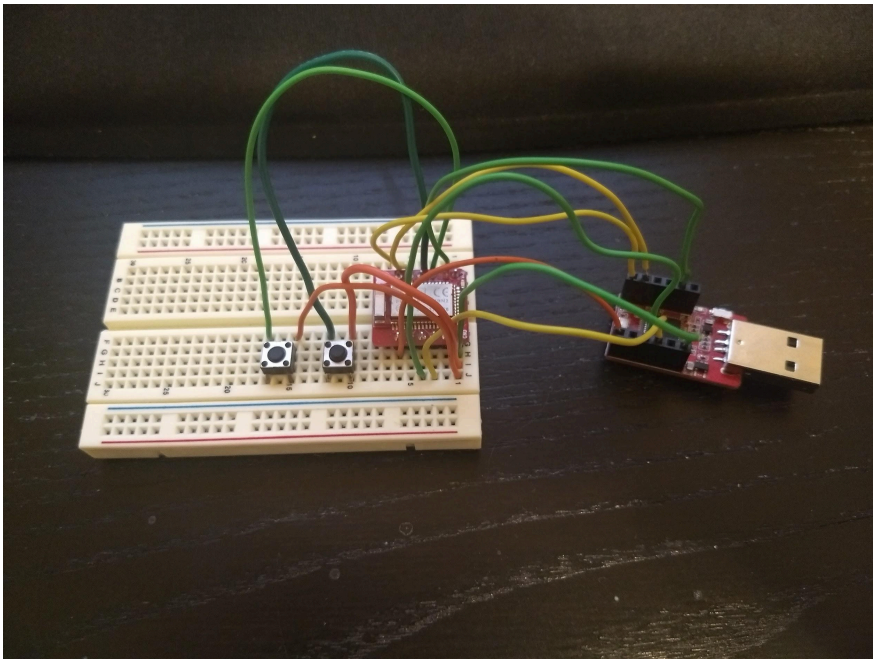
If I were to do it over again, I think I would have picked a different bluetooth module than the Redbear MB-N2. The problem is that the breadboard-able version they provide already has the extra circuitry, which their bare model does not. What I really need is a module where the bare module and the breadboard breakout version have identical circuitry, that way I can get it working on the breadboard and be more confident about transitioning to a custom PCB.


C8
C4
C3
C5
SW1
SW2
SW_Push
SW_Push


NotherButton v0.0.1

Breadboard Prototype

In the end, in order to at least get some sort of working prototype, it was decided to implement a breadboard version using the Redbear Nano2 kit [14]. This kit is based around the MB-N2 module, but already includes the reference circuitry. It also is programmable with Zephyr out of the box. So I moved the prototype to a breadboard, shown below. This worked great. Even though it's not a complete proof-of-concept, it's pretty close to the circuitry that would be used in the final version.



NotherButton Software

Embedded

Zephyr has a very nice build system. All of the application code is written in C. I was able to combine two of their example apps, one for using a button and one for a mock BLE heart rate monitor (HRM), in order to implement the NotherButton bluetooth device. The trickiest part was defining a custom BLE service, but mostly it just boiled down to using new UUIDs. I also had a bit of trouble getting the GPIOs correctly configured, but overall it wasn't too bad.

The application on the embedded side simply triggers interrupts when the buttons are pressed, and sends the button state over the bluetooth link.

Phone

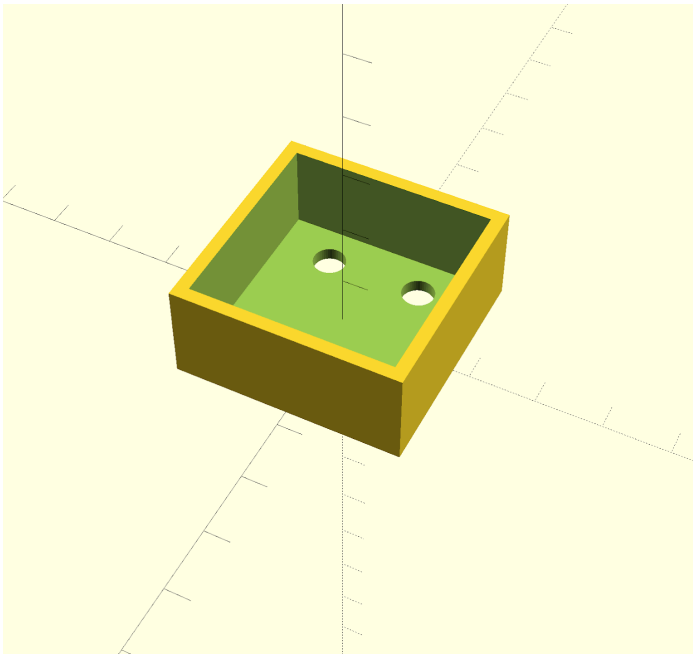
I wanted to use this as an opportunity to try out React Native [8]. React [9] is a JavaScript library for making user interfaces for web applications. It was created by Facebook and is very popular. I have some experience using it and really like it. React Native is Facebook's version of React for writing mobile applications. It lets you write your app in JavaScript, but the really cool thing is that it actually compiles down to native code for the UI. This gives a native look and feel and better performance than simply using a webview on the phone. It also lets you write apps for iOS and Android that share the same codebase. Although I focused on Android for the prototype, I was careful to choose libraries that also work for iOS. In theory the app I've developed should work on both. In addition to React Native, I needed two other libraries to implement my prototype. `react-native-ble-plx` [10] let's me control the bluetooth radio on the phone from JavaScript. `react-native-system-setting` [11] provides access to system settings such as screen brightness and volume. These both worked great. I've been incredibly impressed with the React Native ecosystem overall. There were several situations that I would expected things to break and they just worked.

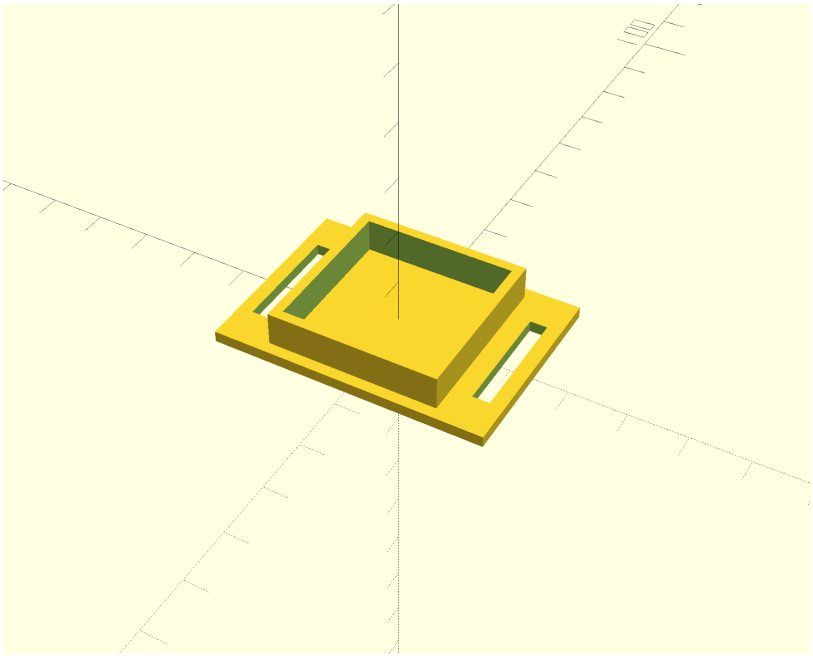
The prototype app I developed is very simple. There are two buttons, which let the user select between screen brightness and volume. Whichever one is selected determines what setting changes when the buttons on the NotherButton board are pressed.

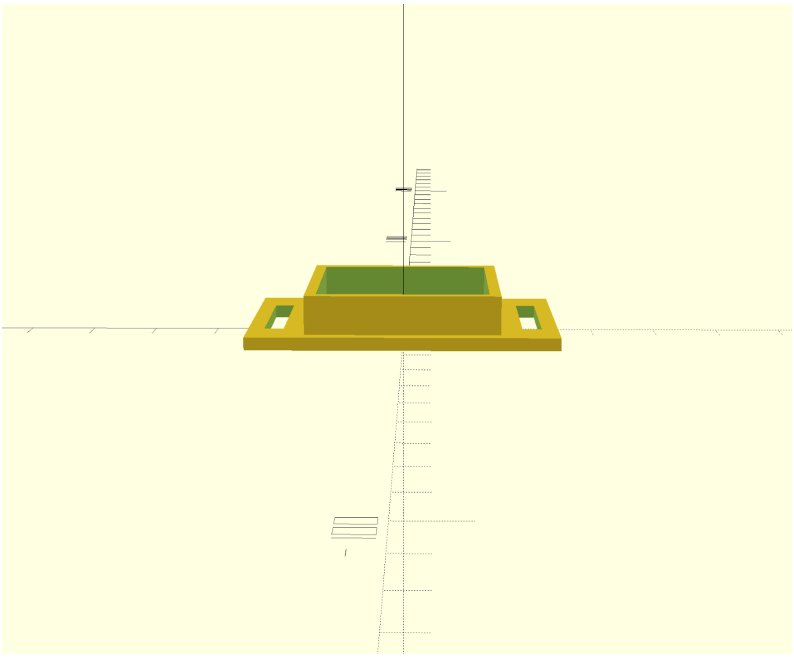
Enclosure

Design

For the enclosure, I designed a simple 3D module using OpenSCAD [12]. This is another incredible piece of open source software. OpenSCAD takes an interesting approach to CAD. Rather than using the mouse to create shapes and build up your model, instead you define your model in code using combinations of primitive shapes and math. So, for example, to create a hollow cube you make a big cube and then subtract a smaller one from inside it to hollow it out. Obviously this workflow is designed to cater to programmers, and I found it very natural. My enclosure is designed to be worn on the wrist. The prototype is a very simple base for holding the PCB, with a lid that goes over the top with holes for the buttons to poke through. The base has slots for a watch strap to go through. I've including some renders below.

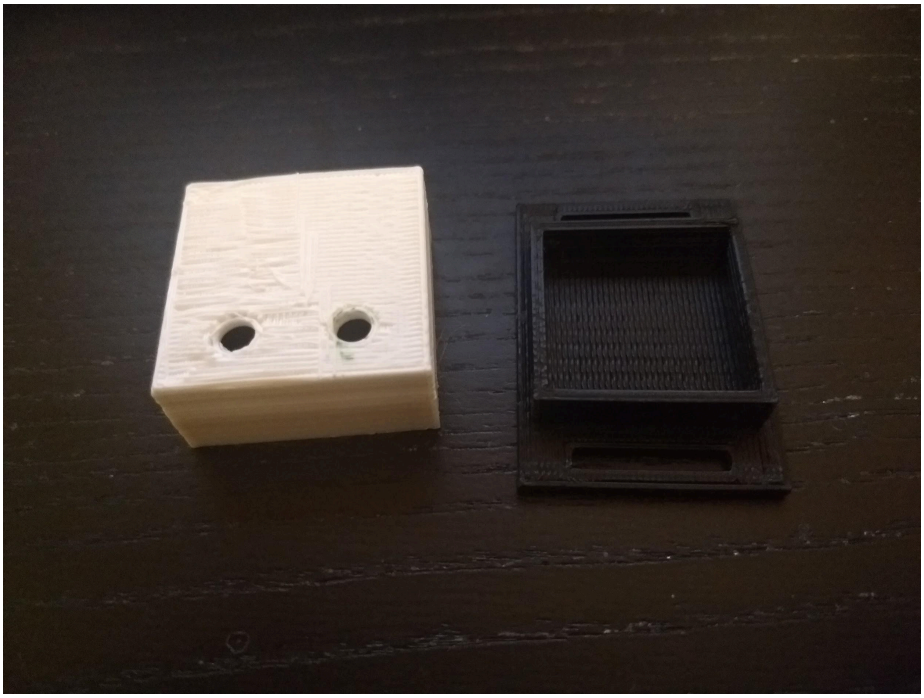


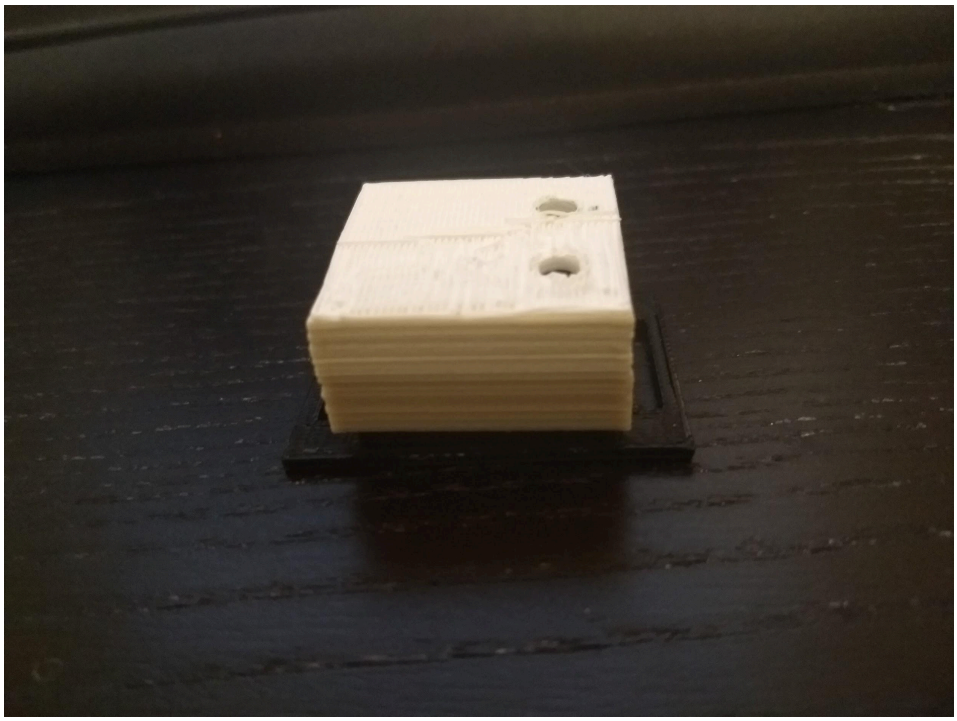


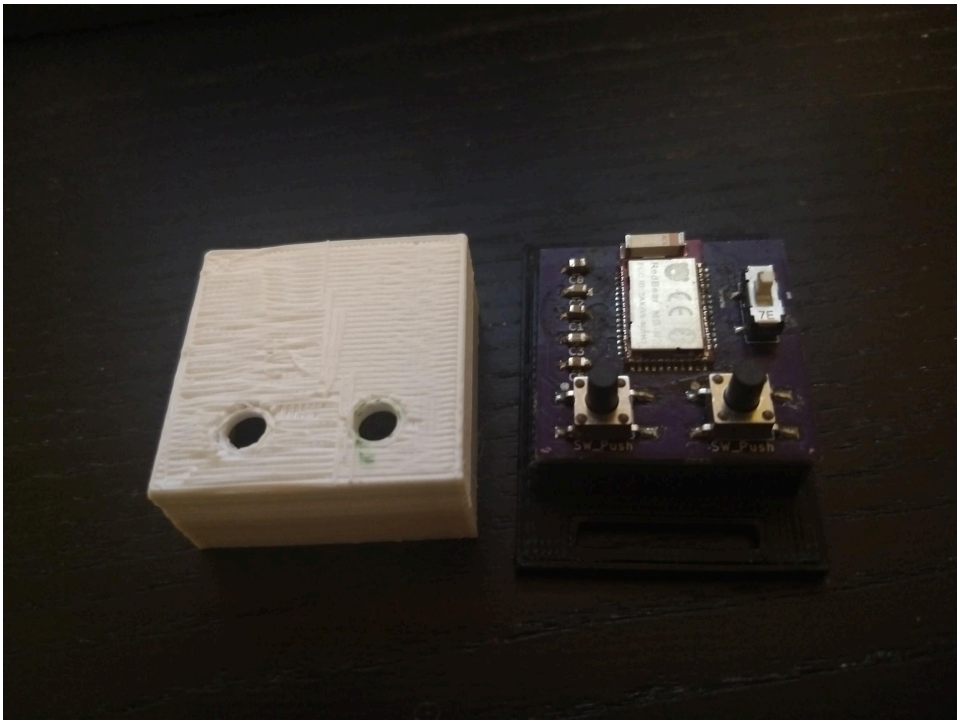


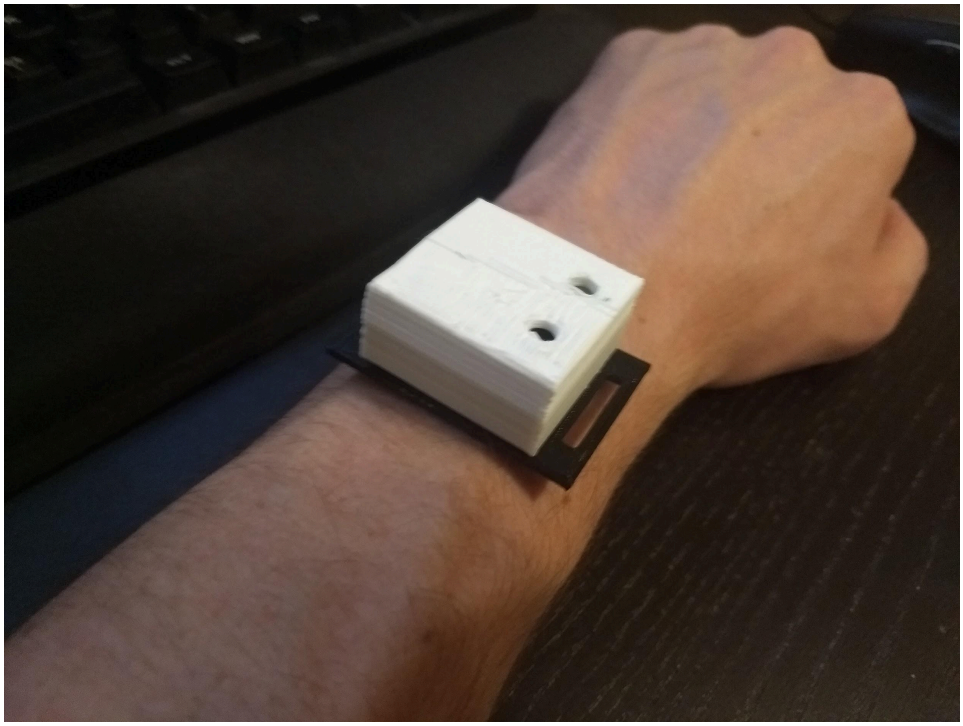
Fabrication

I was able to get the 3D model printed for free from ASU's excellent 3D Print lab [15]. Images of the printed enclosure are below. I think it turned out rather well, although it's not exactly super wearable at the current size. It needs to be optimized. Unfortunately, my tolerances weren't big enough on the first version and the lid barely fits on the base. It doesn't fit around the board at all. I put a new order in right away but I'm not sure if it will be done before the end of the semester.









Conclusion

This project was a fantastic learning experience. Although I really wish the custom PCB had worked, I still feel like I accomplished about 80% of what I set out to build, and 100% of what I

wanted to learn. I feel privileged to be able to get school credit for working on things I'm so intrinsically interested in, and I know the skills I've developed will serve me well in my career.

References

[0] https://www.nordicsemi.com/Products/nRF52-Series-SoC
[1] https://redbear.cc/product/mb-n2.html
[2] https://developer.arm.com/products/processors/cortex-m/cortex-m4
[3] https://www.zephyrproject.org/
[4] http://kicad-pcb.org/
[5] https://github.com/redbear/nRF5x/blob/master/nRF52832/datasheet/MB-N2_Datasheet.pdf
[6] http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf
[7] https://oshpark.com/
[8] https://facebook.github.io/react-native/
[9] https://reactjs.org/
[10] https://github.com/Polidea/react-native-ble-plx
[11] https://github.com/c19354837/react-native-system-setting
[12] http://www.openscad.org/
[13] https://www.youtube.com/watch?v=lo6h8HJ1oV0
[14] https://redbear.cc/product/ble-nano-kit-2.html
[15] http://fse3dprintlab.wikispaces.asu.edu/