CyberJewelry

Saniya Singh, Madi Davis & Shize Che

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*— Wearable tech has become ubiquitous in the last few years, but most of the widely available options are focused on utility: collecting health/fitness data or accessing smartphone features. These devices have extremely limited customization and are lacking sufficient avenues for self-expression. Our system hopes to provide a solution to this in the form of wirelessly customizable digital jewelry.**

***Index Terms*— BLE, DotStar, UART**

I. Introduction

In the last decade, wearable technology has become increasingly ubiquitous in our society. Most of the widely available wearable tech options are focused on providing some utility such as collecting health data (ie: Fitbit, Oura Ring) or increasing access to smartphone functions (ie: Apple Watch). These devices all have limited customization options which do not provide sufficient avenues for the wearer to express their individuality. Due to these limitations, there exists a market for digital wearable technology that exists purely for the sake of style and self-expression. These bold electronic accessories used to be reserved for the realms of festival wear and costuming, but recently they have been slowly entering the mainstream fashion consciousness. To draw inspiration for our device, we looked to the electronic jewelry options that are currently available. The festival wear company Neon Cowboys has a few electronic accessories in this area such as their 'LED Face Jewelry' and 'Light-up Bolo Tie'; the former utilizes single-color LED strips in a few shapes and color options that are worn directly on the skin using an adhesive and powered by an external non-rechargeable CR2032 coin battery, and the latter uses EL wire powered by a large rechargeable in-line battery pack. We also found an open-source project called HALO-90 that allows users to create a sound-responsive earring with a ring of single-color LEDs controlled by an STM8 and powered by a non-rechargeable CR2032 coin battery. This device is not commercially available. None of these devices have color changing options or wireless control.

With our project, we aim to design a set of digital earrings that are wirelessly customizable via Bluetooth and an iOS mobile application. This device would have a variety of use cases from daily jewelry to clubwear for people who want an elevated accessory that provides near infinite customization options. There are very few competing technologies in this area, and no commercially available products that provide the features we intend to implement with our system.

II. Use-Case Requirements

From the description of our system and the needs of our user base, we proceed by introducing the use case requirements that will guide our design process.

A. Physical Design

a. Weight

We want our users to be comfortable with the weight of the device that will be hanging from their earlobe. Looking at weight constraints for heavier earrings, we've decided on a maximum weight constraint of 20 g per earring to ensure user comfort.

b. Temperature

We want our users to be comfortable with the temperature of the device because it will be in close contact with the user's skin. Looking at data for human temperature sensitivity, we've decided on a maximum temperature constraint of 40° C.

c. Material

We want the material that is used for the casing of the device and the earring post to be skin-safe and hypoallergenic. For this, we will be using surgical grade steel for the portions of the earring that will be in direct contact with the skin.

B. Functionality

a. Update Speed

We want the user to be able to quickly modify the pattern on the earring without a lot of lagging after pushing the new design to the earring.

b. Setup Time

We want users to have a quick and painless way to set up the device. We are using BLE to make the device more comfortable to wear, but this means the user's will need to pair to the device from their iOS device. We would like the general setup time for the device to come in under 90 seconds for a new user.

c. Battery Life

We want the device to be usable for general eventwear so we would like to achieve a minimum battery life of 3 hours for the DotStar Matrix and 45 minutes for the LCD screen versions of the device.

III. Concept of Operation and System Architecture

A. CONCEPT OF OPERATION

Our product consists of three primary subsystems: hardware design, firmware and the mobile application interface . System composition and Interactions are outlined in Fig . 1.

We designed the concept of operation as follows. The user initiates interaction with the system via the iOS mobile application. Upon launching the application, if a device is not currently connected to the user's mobile, a function to scan for peripherals will be executed via the custom CoreBluetoothBased BLE module. After turning on the

earring device, the peripheral will begin transmitting advertising data containing identifiable information via bluetooth over an advertising interval of 20ms. The mobile application utilizes a filtering algorithm to detect UART-enabled "Cyber Jewellery" devices and present them as a list of connectable wearables on the mobile interface. After a user selects and connects to a device, the application will survey the device's available characteristics and services, including peripheral address and battery life, and store relevant information locally as model classes. After establishing a connection to the device, the application will launch the Device Info/Home page, which will provide details about the connected device's name, brightness, current design and battery life. A peripheral delegate in the BLE module will asynchronously prompt updates to the displayed battery and brightness properties by observing changes to corresponduy87A

A. BATTERY LIFE

As outlined in our design requirements, our earring must be lightweight and have durable battery life. The weight and battery life are two competing factors because more durable batteries carry heavier weight. We plan to balance these two competing factors in designing our firmware and making our choice of batteries. We want the firmware to configure the hardware to consume as little energy as possible. Consideration of the following criteria allowed us to develop a list of design guidelines.

LED Matrix Brightness

The brightness of the LED matrix is the dominating factor of the total current draw of the hardware. The total current draw of the hardware is critical to the battery life. Without careful consideration, the LED matrix can draw as much as 2.5A current, which not only raises the temperature too high but also shortens the battery life to less than 15 minutes. On the other hand, a low brightness value provides less color choices and can have the LED light invisible in a bright surrounding. After experimenting with different brightness values, we decided to set the maximum brightness to 2, which is equivalent to limiting the RGB values to below 64 (each RGB value is 255 maximum).

By limiting the brightness to 2, the average current draw of the LED matrix is 100mA, which results in an overall average current draw of 120mA. Using 2 as the maximum brightness value results in 64^3 different color choices, and we noticed it's enough for the user to create usual colors such as yellow, aqua, and magenta. In consideration of brightness, we traded the number of color choices for battery life by limiting the current draw of the LED matrix.

BLE Polling VS Interrupt

The BLE data can be received via either polling or interrupt scheme. To further increase the battery life, we decided to use interrupt. By using the interrupt scheme, it releases the microcontroller from executing the BLE data receive function on every iteration of the loop and thus further reduces the energy consumption. To enable a hardware triggered interrupt, we plan to have users click on a push button to let the microcontroller enter the data receiving interrupt handler.

Choice of Battery

Since we want the earring to be rechargeable, we narrowed our scope of battery choice to lithium batteries. Choosing a battery is essentially a tradeoff between weight, size, and battery life, and we want to find the balance between them. We researched the 3.7v lithium batteries provided by Adafruit and found three choices. The metrics of each battery is summarized into table 1.

Table I. Battery Options

Capacity	**Design Consideration**	
	Dimensions	***Weight***
150mAh	19.75mm x 26.02mm x 3.8mm	4.65g
350mAh	32.5mm x 25mm x 5mm	7g
500mAh	29mm x 36mm x 4.75mm	10.5g

From the capacity column of table 1, we can infer the battery life achievable by each of these three batteries. We designed the overall system current draw to be no greater than 120mA, so both 350mAh and 500mAh batteries satisfy the three hour threshold. However, using the 500mAh battery will cause a total weight of more than 20g and the size of it doesn't fit well with the dimensions of Feather M0, so we choose the 350mAh battery as the default battery. Additionally, considering some users may prefer a lighter and more elegant earring and are willing to accept a shorter battery life, we are actively working on a different design that uses the 150mAh battery.

IV. System Implementation

A. Hardware

We built two versions of hardware, the first version serves as our MVP and the second version has additionally a power switch and an interrupt push button included.

Version 1

In this first version, we locate the LED matrix on the front and the battery on the back. The Feather M0 is located in the middle. To power and drive the LED matrix, there are four connections we need to make: 3v, GND, SPI clock pin, and SPI data pin. To minimize the thickness, the connections are made by soldering the pins closely together. Fig 5 shows the locations of the Feather M0 and the LED Matrix.

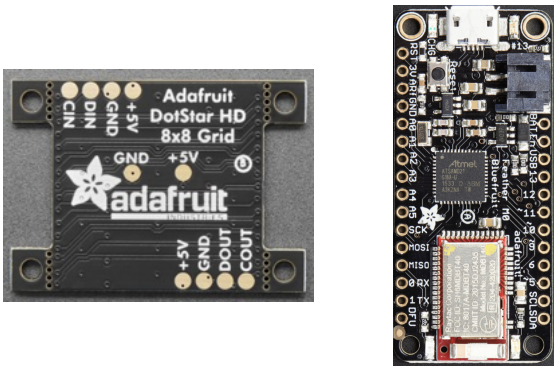


Fig. 5. Feather M0 and Matrix LED pin locations

To stabilize the LED matrix, instead of using LED matrix pins on one single side, we use pins from both sides. Specifically, we connect the +5v on LED matrix to the 3v pin on the Feather M0, and we connect the GND pin on the bottom to the GND on Feather M0. We also connect the CIN, and DIN pins to Feather M0 pin 5 and 6, and use pin 5 and 6 as the SPI clock and data pin. By using these pin connections, the matrix has a stable connection to the Feather board. Fig 6 shows the actual physical connections.

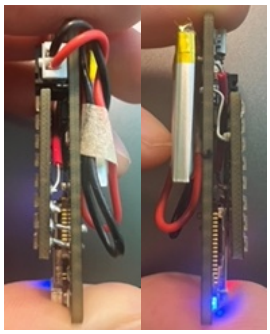


Fig. 6. Soldered Connections on Both Sides

There is a discrepancy between the voltage provided by the Feather and voltage required by the LED matrix. The LED matrix requires 5v on all pins. However, in practice, we were able to get it working perfectly using 3v power supply and 3.3v on SPI clock and data. The soldered connections are demonstrated in Fig 6.

As demonstrated in Fig 6, the battery connects to the Feather M0 to supply power to the system. We are using a 3.7v 350mAh lithium battery. The battery is rechargeable and can be easily charged by simply plugging the Feather M0 to a micro USB. The overall dimension of the earring is 51mm x 25.4mm with a thickness of 1 cm including the battery.

Version 2

In this second version of the hardware, we added in two more functionalities on top of the first version: power switch, and interrupt push button. The first version has the battery connected directly to the system, so it lacks a way for users to turn it on and off. In this second version, we introduce a switch in between so that users can power it off when not using. The switch and battery are connected on a solder board. In addition, as discussed in the battery section of tradeoff studies, we plan to use an interrupt scheme for receiving BLE data, hence it's necessary to include a push button to trigger the interrupt. We plan to connect the push button with battery and pin on the same solder board that connects the switch and the battery. Both the switch and push button will be taped to the back of the Feather M0.

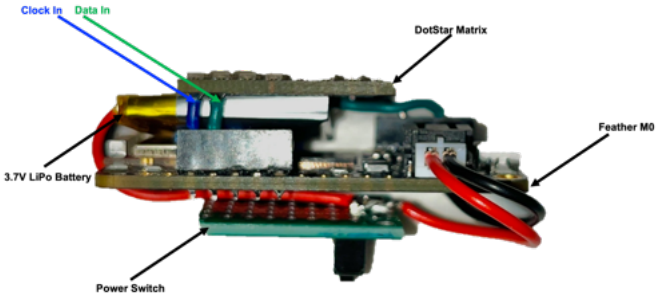


Fig . 7. Diagram of Hardware Version 2

The second version also has a new design for connection. Instead of connecting the two components using soldered wires, we use a set of short female headers to build the connection. Using short female headers increases the thickness but has more reliability. The SPI clock and data pins on the second version are pin 5 and 6. See Fig 7, 8 and 9 the newly modified connections of hardware version 2.

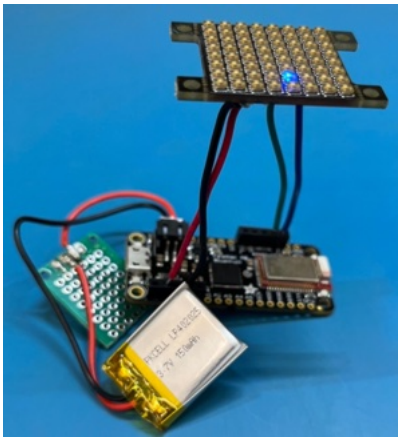


Fig . 8. Testing with Hardware Version 2

By shortening the wires to suitable length, we can hide the battery in between the Feather M0 board and the LED matrix. Fig 8 shows the front and side views of a fully integrated hardware version 2.

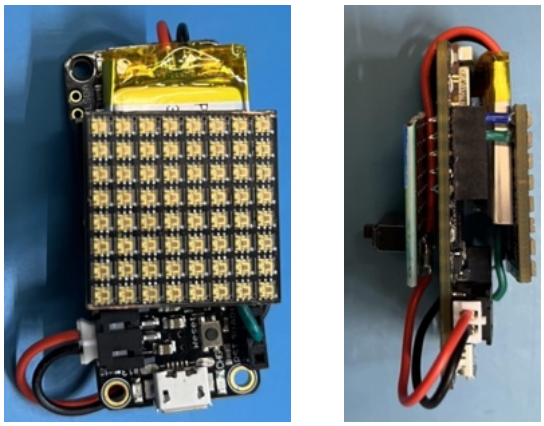


Fig. 9. Front and Side Views of Hardware Version 2

B. *Firmware*

The firmware is coded using Arduino. We used Adafruit's BLE and DotStarMatrix libraries to interface with the BLE module and LED matrix. Similar to hardware, we implemented two versions of the firmware. The first version is for interacting with our customized software, and the second version is for using Adafruit's Bluefruit Connect App. The two versions differ mainly in how LED pixels are updated and the implementation of moving patterns.

Version 1

The first version of the firmware is designed for interacting with our customized iOS App. In our customized iOS App, BLE data is sent using hex strings. For example, if the RGB values of a pixel are 255, 255, 255, the App will send the string "0xff0xff0xff" (each RGB value is 1 byte). The firmware receives and parses this string into its corresponding RGB values and sets the pixel value.

In this version, the LED matrix is updated only after receiving the entire RGB values for all 64 pixels. This is in correspondence with our software design where users first design the whole matrix and then click send. As a result, users won't be able to see single pixel updates as they set single pixel color on the App.

Independent moving pattern is implemented in this version. To design a moving pattern, users first create 5 designs in the App. These 5 designs are used as "frames" and are sent to the firmware after the design is finalized. The data for all five frames are sent sequentially (frame1 followed by frame2 and so on). After receiving the data, the firmware processes them into RGB values, chunks it into 5 frames, and saves them. Then the firmware displays the saved frames sequentially (frame1 followed by frame2 and so on) with 500ms delay in between. This moving pattern gives users the flexibility to create any 5-frame moving pattern they want.

Version 2

The second version of the firmware is designed for interacting with Adafruit's Bluefruit Connect App. This App is freely available on the App Store, so anyone who comes to our demo can download the App and try interacting with our earring on their phone.

Adafruit's App works differently, and every BLE data it sends starts with a command character defining what it expects the firmware to do. Table 2 summarizes the command characters and the data following them. For our application specifically, some commands are unnecessary, and they are "not implemented" in the table.

Table II. Adafruit BLE Commands

Command Character	Data Following the Command Character	Expected Firmware Behavior
'V'	Not Implemented	n.a.
'S'	Not implemented	n.a.
'C'	uint8_t r, uint_t g, uint8_t b	Set RGB values of all pixels to {r, g, b}
'B'	Not implemented	n.a.
'P'	uint8_t x, uint8_t y, uint8_t r, uint8_t g, uint8_t b	Set RGB value of pixel (x, y) to {r, g, b}

In correspondence with Adafruit's Bluefruit Connect App, the firmware reads the command character and data provided in table 2 and performs expected behavior. If the command is successfully executed, the firmware sends "OK" to the App. Using the 'P' (set pixel) command, users are able to see single pixel updates as they design the pattern.

This version of firmware implements dependent moving patterns. Dependent moving patterns can have the current pattern move up/down/left/right and rotate clockwise/counterclockwise. To implement dependent moving patterns, we extended the command characters to let users define the way they want the pattern to move. Table 3 summarizes the extended command characters for moving pattern definition.

Table III. Moving Pattern Commands

Command Character	Moving Pattern Definition
'^'	Move Up
'-'	Move Down
'<'	Move Left
'>'	Move Right
'l'	Rotate Left
'r'	Rotate Right
'.'	Stay Still

The moving pattern commands are sent via the UART interface provided by the Adafruit App for users to move the

current pattern. The firmware computes the next frame based on the current frame and delays for 500ms to display the next frame. To avoid unnecessary usage of dynamic memory, all next frames are computed in-place.

C. *Mobile Application*

The Mobile Application interface can be segmented into three user flows, as documented in Appendix A: User Setup and Design Selection. Design Editing and Design Upload & Save.

User Flow 1: User Setup - After Launching the application, users will be directed to a "Bluetooth Connection" page. Landing on this page will direct an instance of CoreBluetooth's CBCentralManager to scan for nearby peripheral devices. Detected devices will appear on the bottom of the screen as a list of card style buttons detailing the device name and type (LED or LCD). When a user selects their desired device from the available options, "Device Home" will be loaded in the application.
This Home Screen will display a summary of device information including an image of the device, device name, the current battery life, the current uploaded design and a slider control top adjust the overall display brightness. A button in the top right hand corner of the screen will enable the user to toggle the device's bluetooth on and off. The home page will be the same for both types of device. Using the toolbar at the bottom of the screen allows the user to Navigate to the "Design Library" where they will be able to choose an existing design to upload to the device from a scrollable list of options, via a search bar. Alternatively Users may choose to create a new design by selecting the button in the top right corner of the screen view. Both options will prompt the application to navigate to the "Design Edit" page.

User Flow 2: Design Creation - The type of peripheral device connected will dictate the "Design Edit" display and subsequent functionalities.
If the device features an LCD Screen Display, the "Design Edit" page will consist of a preview of the design rendered on stylised representation of the device. A control menu at the bottom of the screen will provide users with two options to customize their design. Selecting the "Visual" Tab of the menu will allow the user to upload files in the format of .jpg, .png and .gif. Selecting multiple files will display each visual as a slideshow, with a default display time of 10 seconds per image and a default transition length of 500ms. Controls in the "Motion" tab will allow the user to adjust slideshow parameters. The"Transition Length"slider will change the transition time between visuals (from 0 to 1000ms) and the "Event Length" slider will alter the amount of time each visual is displayed (from 0 to 30 seconds).
If the device has an LED Screen Display, the "Design Edit" Screen will feature an touch-interactive model of the 8x8 matrix in the center of the page. The user can tap on one or more of the 64 graphical representations to edit display properties of the corresponding LED units on the device. Selecting the "Hue" tab from the control menu allows the color and brightness of the LED(s) to be adjusted using 2 separate slider controls. Selecting the "Motion" tab allows the user to animate a single or group of LED(s). Animation can be customized by selecting custom preset animations (E.g. blink, fade, pulse etc.) or setting custom values for the Event Length (length of time an LED is on/off) and Transition length via slider widgets.
Selecting the Reset button in the top left hand corner of both views will; reset the design back to its last saved state.

User Flow 3: Design Upload and Save
Selecting the button in the top right corner of the "Design Edit Page" will save the current design and upload the device to the design.

V. Testing, Verification and Validation

A. HARDWARE DESIGN & FIRMWARE

STM32WB55RG System

Before switching to using the Adafruit Feather M0 board for our system implementation, we spent a significant amount of time working with the STM32 interface and developing a custom PCB for that design. We selected the STM32WB55 model microcontroller and began developing our firmware with the P-Nucleo development board. This microcontroller was selected for versatility and computation power as well as the wireless and BLE capabilities. The plan was to develop basic firmware and create a matrix control library with the P-Nucleo board, then transfer the firmware code to the STM32WB microcontroller on a PCB with the same footprint as the LED matrix. With the PCB, our device would have a much sleeker form factor. Using the development board as a reference, we designed the following schematic for our custom PCB.

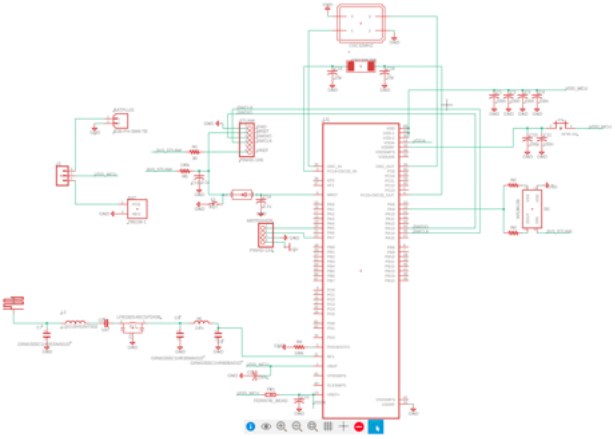


Fig . 10. Overview of PCB Schematic

The design intentionally accounts for various possible modifications to the hardware circuit; there is a battery bracket for a mounted CR2032 coin battery and an optional jumper for Li-Po batteries. This schematic also includes special circuits to specifically accommodate the STLINK programming interface for the STM32 as shown in Fig. 11.

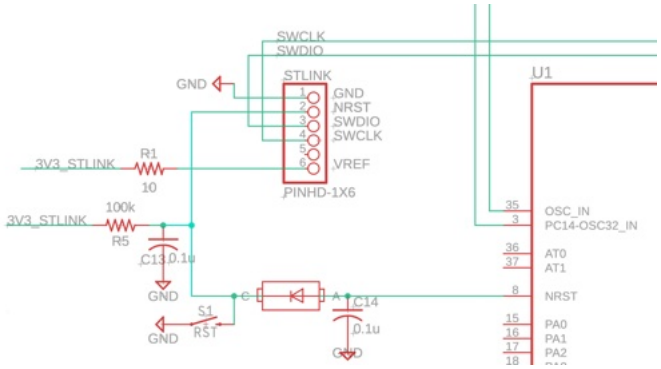


Fig . 11. STLINK Circuitry for Custom PCB

The most vital aspect of designing our PCB was to correctly implement the RF circuitry that supported the STM32WB microcontroller's wireless and Bluetooth capabilities.

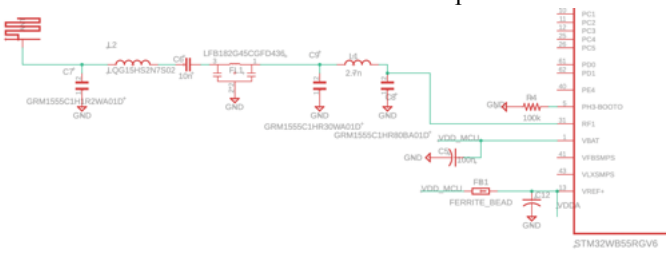


Fig . 12. RF Circuitry for PCB

Several components for the RF circuit were designed to exactly match the inductor, capacitors and bandpass filter used on the development board to ensure correct operation. The board was also equipped with an onboard trace antenna (see Fig. 13) for which there are no existing CAD components. Several other components required for the operation of our microcontroller did not have publicly available schematic symbols or 2D models. Due to this, we designed a library of custom CAD components, including a Schottky diode, both oscillators and the RF trace antenna.

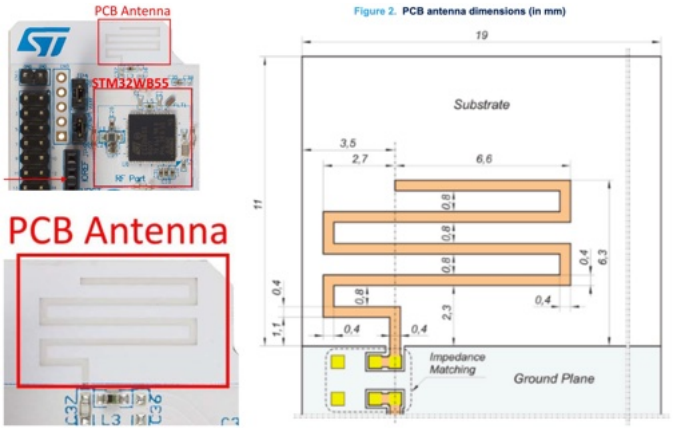


Fig . 13. Trace Antenna on the P-Nucleo Board [1][2]

A few iterations of the PCB antenna were developed, each using the design specifications shown in Fig. 13.

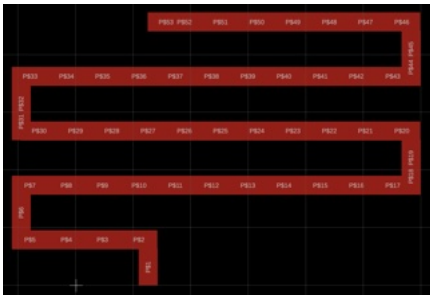


Fig . 14. Trace Antenna, First Version

In the first version (Fig. 14), we approximated the design specs of the trace antenna using PCB pads in Fusion 360.

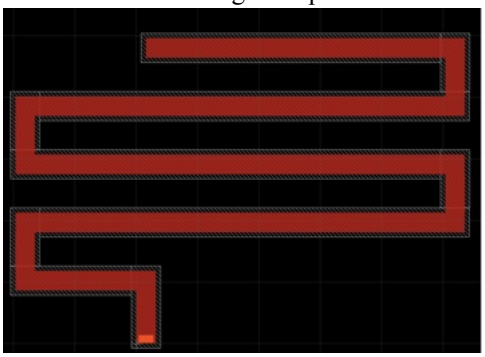


Fig . 15. Trace Antenna, Final Version

In the final version (Fig. 15), the antenna is designed with the exact measurements shown in Fig. 13 and connected to the impedance matching circuitry via a single connective pad. Once completed, the antenna was added to the schematic and layout for our custom PCB. The final component and schematic symbol for the trace antenna are shown in Figures 16 and 17.

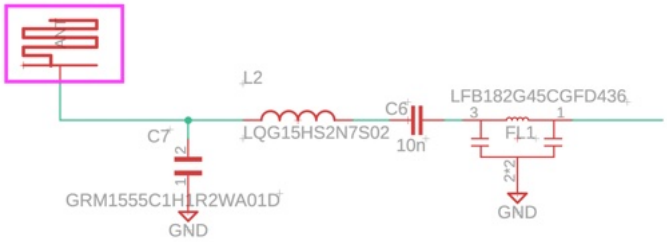


Fig . 16. Antenna and RF Circuitry for PCB

Fig . 17. Custom antenna component

Many other components were designed for our custom PCB in this fashion. Additional details about the custom design of one of the oscillators is provided.

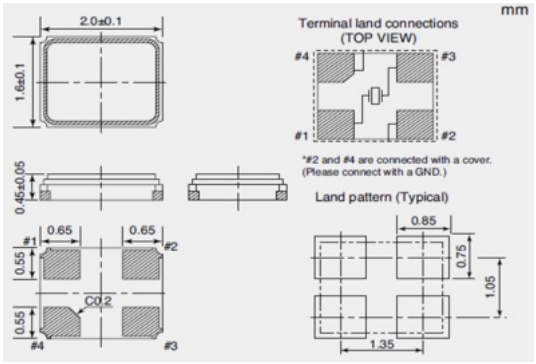


Fig . 18. Specification for NX2016_32M oscillator[3]

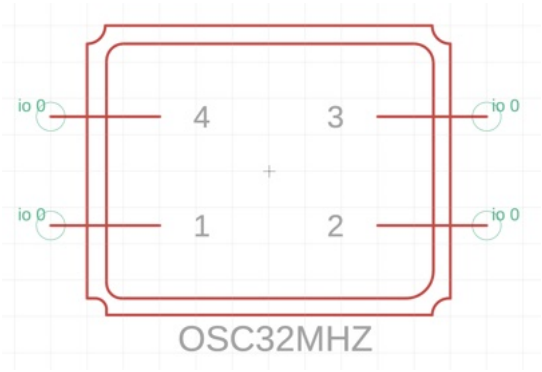


Fig . 19. Schematic symbol for oscillator

Fig . 20. 2D Layout (left) for NX2016_32M

Fig . 21. Library component (right) for NX2016_32M

Unfortunately, due to supply chain issues, we were unable to procure some of the circuit components required to fabricate the PCB for our system. Since the components missing were essential to the RF circuitry, we decided to shift our focus and begin developing with the Feather M0 board.

Feather M0 System

Results for Design Specification A1 - Display Update Latency

We measure the display update latency using the Saleae logic analyzer. The latency between data reception and display is caused by data parsing. Therefore, we wrap the data parsing section of the code with digitalWrite(pin, HIGH) and digitalWrite(pin, LOW) to indicate the time elapsed executing them during a design upload. The following Fig 20 shows the timing result.

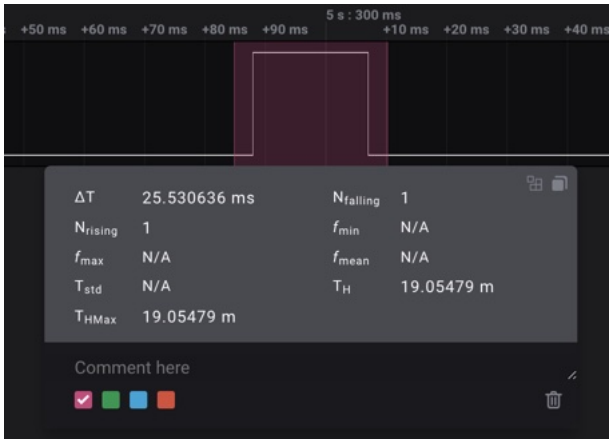


Fig. 20. Display Update Latency Timing Result

As shown in Fig 20, the latency between data reception and display update is 25.53ms, which is faster than the 500 ms display update latency requirement.

Design Specification A2 - Battery Life

We measure the total battery life in two ways. The first way is a theoretical estimation using measured current draw. As in the tradeoff studies section, we designed our system to have an average current draw of 120mAh. Using a fully charged 350mAh battery, the battery life should be 2.91 hours.

We also conducted a battery life experiment where we put the LED matrix into an average power state—half pixels display white and half pixels display blue. We chose this state because white is the color that consumes the most energy and blue is a color that consumes relatively less energy. Then we let the system display a rotation moving pattern, which we assume is a typical use case. Under such operating conditions, the LED matrix kept lighting for more than 3 hours. However, we noticed the color starts to dim after 2 hours and 30 minutes. This is expected because a lithium battery's voltage starts dropping when it approaches the end of battery life. Fig 21 shows the voltage of a 3.7v lithium battery tested using a 18Ω resistor. We noticed the same behavior in our battery life testing.

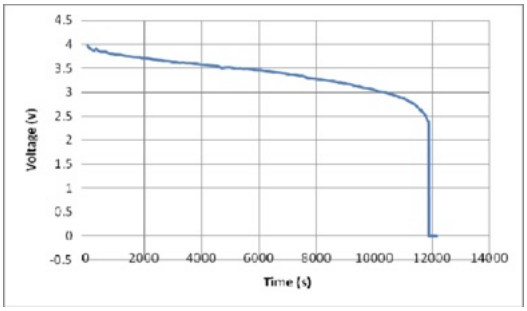


Fig. 21. Lithium Battery Voltage Decays with Time

Design Specification A3 - Temperature
To measure a representative operating temperature, we use the same operation condition as in the battery life test. After the system operated for 1 hour, we measured the temperature using a thermometer. The measured temperature is 38.3 ℃.

B. MOBILE APPLICATION INTERFACE

Results for Design Specification B1 - Pairing Time
To evaluate the ability of our system to effectively discover and connect to the Adafruit Feather Device, we developed a test module to stress evaluate the ability of our BLE module to scan for and connect to peripherals. We ran 20 trials on three different Feather M0 peripherals to check if our design requirement of a minimum discovery window of 5000ms and connection window of 50ms was met. We found that on all trials, the peripheral was discovered within 500ms after the startScan() function was invoked. Additionally, the connection design requirement was also met 100% of the time.

Results for Design Specification B2 - BLE Connection Range
We evaluated the connection range of the device by testing the ability of the application to discover and connect to the feather M0 from an increasing range of horizontal distances, up to our specified maximum of 5m.

TABLE IV. BLE CONNECTION TEST RESULTS

Distance (m)	Discovered?	Connected?	RSSI (db)
0	Yes	Yes	-30
0.5	Yes	Yes	-57
1	Yes	Yes	-82
1.5	Yes	Yes	-77
2	Yes	Yes	-92
2.5	Yes	Yes	-84
3	Yes	Yes	-88
3.5	Yes	Yes	-87
4	Yes	Yes	-96
4.5	Yes	Yes	-93
5	Yes	Yes	-85

The results displayed in Table 4 demonstrate that the system sufficiently meets the specified BLE connection range of 5m. Moreover, the recorded RSSI values indicate the BLE signal power strength is largest within a range of 0 to 1.5, which suffices for the device's intended use of the variable. To further explore the connectivity range beyond the results displayed above, we ran additional testing to explore the maximum range of connection. We found that the mobile application could successfully detect and connect to the feather device up to 20 m ± 0.5m, although the pairing time increased with distance, for distances greater than 15m.

C. USE-CASE REQUIREMENTS

Weight
The weight is measured using a weight scale. The measured weight is 20g

Material
The components that can contact users' skin are the LED matrix, the Feather M0 board, or the battery, all of which are skin-safe. Additionally, the maximum voltage in the entire is 3.7v, which is also safe.

Sturdiness
We conducted preliminary sturdiness tests on both versions of hardware by shaking them in the air, dropping them on tables, and exerting pressure on them, and the hardwares still functions correctly.

VI. PROJECT MANAGEMENT

A. *Schedule*

The schedule was changed due to the change of hardware design. We put together hardware and firmware version 1 before the final presentation. After the final presentation, we are putting together the software with hardware and firmware version 2. At the end of the semester, we are all working on the version 2 hardware and software together. Fig 22 shows the updated schedule after the design report.

B. *Team Member Responsibilities*

Shize Che is responsible for developing the firmware for both of the systems that were tested (STM32 and Feather M0). This includes interfacing with the LED matrix and transmitting serial data to control the patterns on the device. He is also responsible for assembling the initial hardware versions for the earring device.

Madi Davis is primarily responsible for the Mobile Application Interface section of the project. This will entail designing the application functionality and UI/UX, front-end development and implementing modules for data storage and BLE Communication. She is also responsible for overseeing integration and working on the first iteration of the CAD model for the device enclosure

Saniya Singh is responsible for the PCB, as well as some firmware and BLE-related software. Developing the custom PCB for the STM32 version of the system involved creating a custom library of components. She is also responsible for assembling the final hardware design for the earring.

C. Bill of Materials and Budget

Part	Manufacturer	#	Cost
Nucleo Dev Board for STM32WB55	Digikey	2	$83
LCD Screen	Digikey	1	$38
DotStar LED Pixel Matrix	Adafruit	5	$150
Feather M0 Board	Adafruit	4	$120
Switch	Adafruit	5	$10
Short Female Header	Adafruit	5	$10
150mAh 3.7v Lithium Battery	Adafruit	2	$9
350mAh 3.7v Lithium Battery	Adafruit	2	$11
500mAh 5v Lithium Battery	Adafruit	2	$20

D. Risk Management

Two risks have happened since the design report. First, we weren't able to design our own PCB and second, we experienced a significant challenge and delay in integrating the hardware and software. We managed the PCB risk by redesigning the hardware using a commercially available microcontroller board. By using the commercially available Feather M0 board, we comprised the size and weight of the earring and mitigated the developing difficulty of the PCB. Also, there were misconceptions about how the data is transferred from the software to the hardware which caused the delay on system integration. We successfully managed this risk by having the team members communicate and work together more frequently towards the end of the semester.

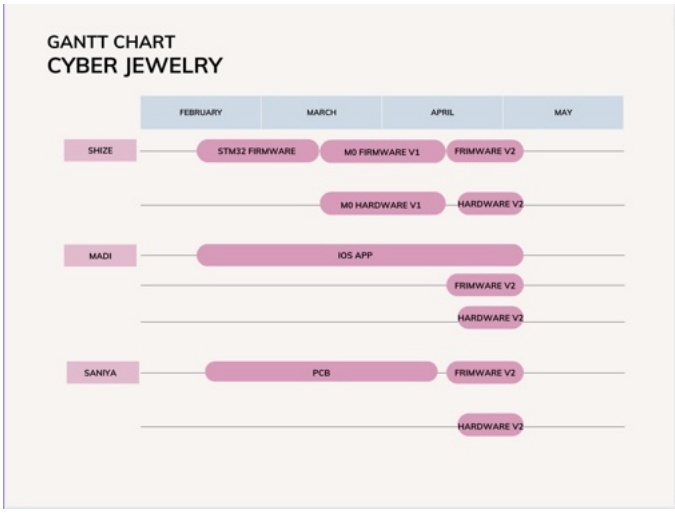


Fig. 22. Schedule Update After Design Report

VII. Ethical Issues

There are two main ethical concerns in this project. First, people other than the user may find and connect to the device. Second, users may display hostile patterns that are inappropriate for certain groups.

We specifically addressed the first ethical concern in firmware. To enable the display functionality of the earring, the user has to first send a password. The password is provided by us and only known to the user. Without sending the correct password first, the firmware will ignore any incoming data.

It's harder to address the second ethical problem. A way we thought about is to include a machine learning model in the iOS App to recognize inappropriate patterns and avoid sending it to the earring.

VIII. Related Work

There's a handful of other projects we found that provided similar capabilities to what we want to achieve with our device. The first and the simplest is the HALO-90 open-source earrings project. The HALO-90 earrings use a ring of single-color LEDs on a custom PCB for their design. The PCB contains a STM8L series microcontroller and an onboard battery. The device must be programmed over a wired connection and the earrings react to music. There is one more open source project that we found relevant to our work as it uses the same LED matrix component for a similar application. The project is titled DotStar Fortune Necklace: it is a digital necklace that uses another Adafruit component for Bluetooth connectivity and a large 3.7 V Lithium Polymer battery that are encased in a large rectangle that hangs at the back of the neck while the matrices acts as the necklace pendant and hangs at the front connected by a wire.

IX. Summary

The project met all the design and use-case requirements.

However, due to the fact that we replaced the PCB with Feather M0, we compromised the size and weight. The size was not originally in the design requirements but having a PCB that fits together with the LED matrix creates a more graceful design.

A. Future work

There's potentially more work to be done on this project. Specifically, Shize plans to continue expanding the project by using our custom PCB design with the STM32WB55. The STM32WB55 microcontroller is faster in both BLE and serial communications. Also, we didn't get a chance to try using the LCD screen, and Shize plans to look into the possibility of implementing the screen.

B. Lessons Learned

We were more on the optimistic side of building a Bluetooth PCB. As discussed in the test and validation section, designing a Bluetooth PCB is harder than a regular PCB and requires more experience than we have. Furthermore, our project experienced a delay in integration, which was caused by miscommunication between team members. A lesson from it is we should always communicate more with this team and make sure we settle on how the system is integrated before we proceed to working on our own parts.

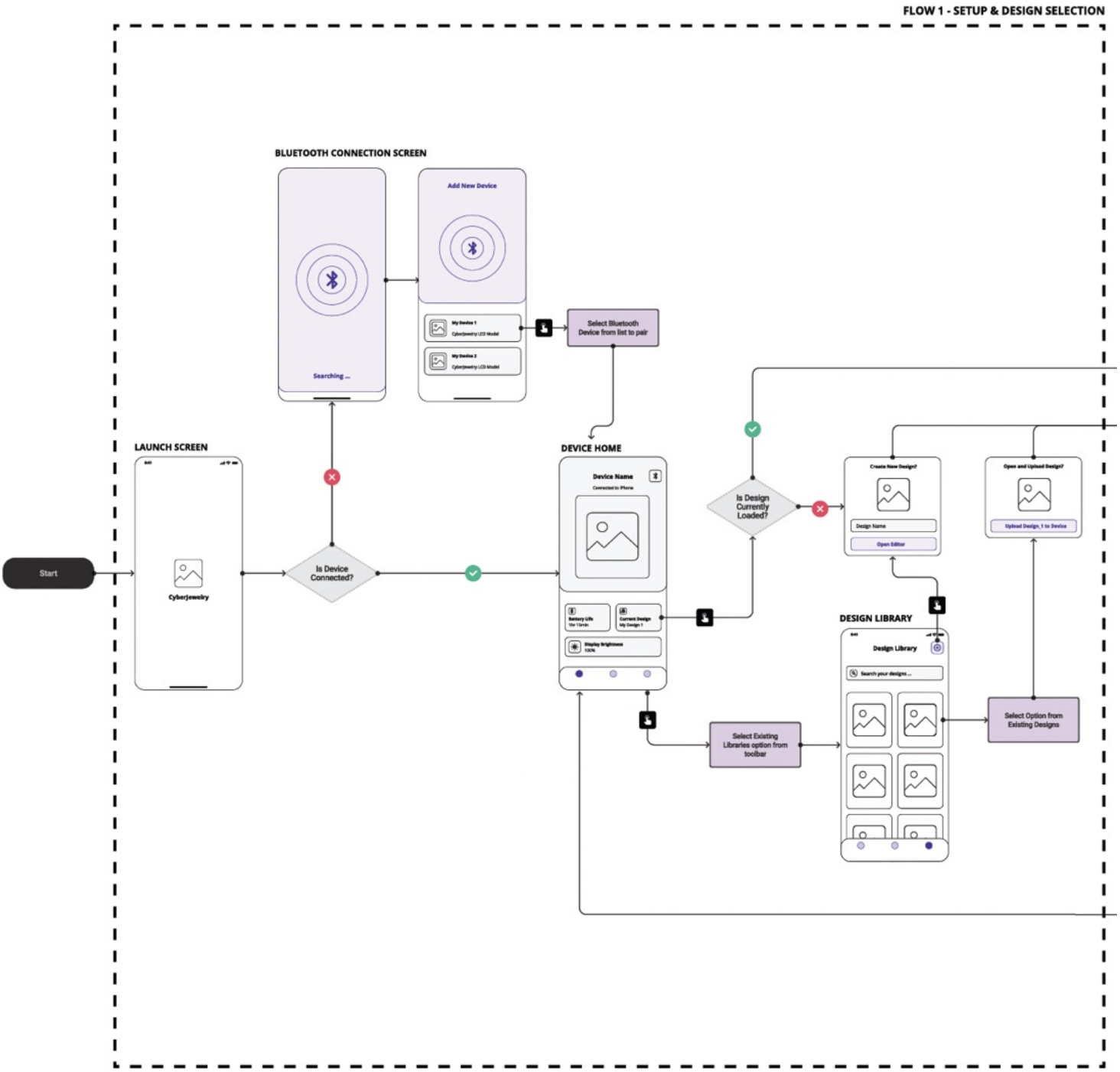
Glossary of Acronyms

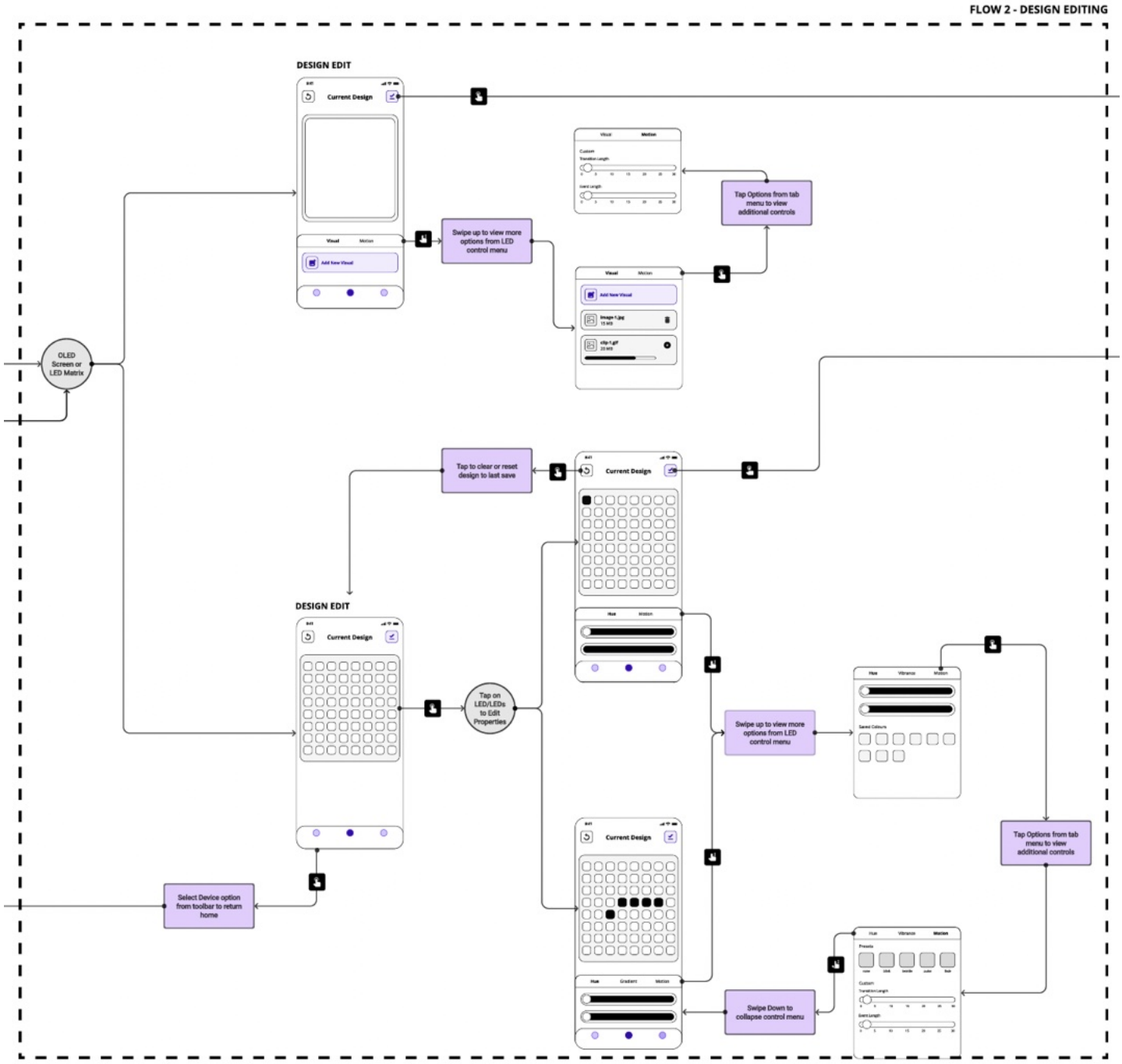
BLE – Bluetooth Low Energy
CB - CoreBluetooth
GATT - Generic Attribute Profile
PCB - Printed Circuit Board
UART - Universal Asynchronous Receiver/Transmitter

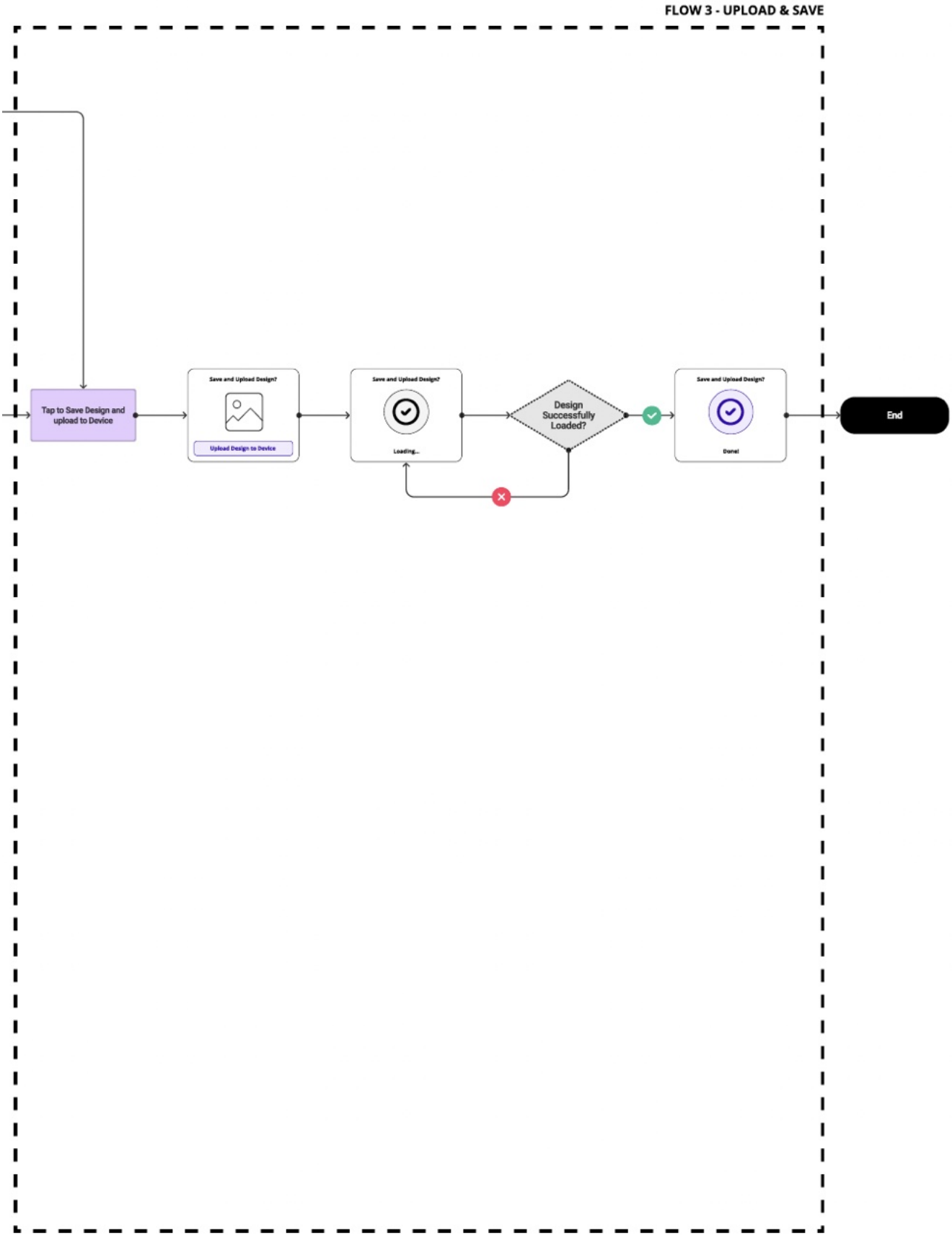
References

[1] STMicroelectronics, "Bluetooth® Low Energy and 802.15.4 Nucleo pack based on STM32WB Series microcontrollers," UM2435 User Manual, Jan. 2018 [Revised Apr. 2019].
[2] STMicroelectronics, "Low cost PCB antenna for 2.4 GHz radio: meander design for STM32WB Series," AN5129 Application note User Manual, Jan. 2018 [Revised Apr. 2019].
[3] NDK, "Specification of Quartz Crystal Units," NX2016SA-32M-EXS00A-CS06465 datasheet, Mar. 2018.
[4] P. Taylor et.al, "Battery Power Comparison to Charge Medical Devices in Developing Countries", in Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2009.
[5] u/Cummins_B, et al. "CoreBluetooth with SwiftUI." Reddit, 2021, https://www.reddit.com/r/SwiftUI/comments/za3o2i/corebluetooth_with_swiftui/.
[6] Novichkov, Artem. "Bluetooth and SwiftUI." Artem Novichkov, 2021, https://blog.artemnovichkov.com/bluetooth-and-swiftui.
[7] Ito, Kazuya. "CoreBluetoothViewModel.swift." GitHub, 2021, https://github.com/Kazuya-Ito-B/iOS-SwiftUI-BLE-Project/blob/main/SwiftUI-BLE-Project/ViewModel/CoreBluetooth/CoreBluetoothViewModel.swift.
[8] Belanger, Jordan. "CBUUIDConvertible.swift." GitHub, 2017, https://github.com/jordanebelanger/SwiftyBluetooth/blob/master/Sources/CBUUIDConvertible.swift.
[9] Mazzini, Federico. "CBTransport.swift." GitHub, 2019, https://github.com/federicomazzini/BLETemplate/blob/master/BLEThing/BLE/Provider/CBTransport.swift.
[10] Nordic Semiconductor. "CBMPeripheral." IOS-CoreBluetooth-Mock, 2021, https://github.com/NordicSemiconductor/IOS-CoreBluetooth-Mock/blob/main/CoreBluetoothMock/Documentation.docc/CBMPeripheral.md.
[11] Appcoda. "Core Bluetooth Programming Guide for iOS." Appcoda, 2021, https://www.appcoda.com/core-bluetooth/.
[12] Adafruit. "Build a Bluetooth App Using Swift 5." Adafruit, 2021, https://cdn-learn.adafruit.com/downloads/pdf/build-a-bluetooth-app-using-swift-5.pdf.

APPENDIX I - MOBILE APPLICATION WIREFLOW DIAGRAMS


FLOW 1 - SETUP & DESIGN SELECTION
BLUETOOTH CONNECTION SCREEN
Add New Device
Searching ...
Select Bluetooth Device from list to pair
LAUNCH SCREEN
DEVICE HOME
Start
Cyberjewelry
Is Device Connected?
Device Name
Is Design Currently Loaded?
Create New Design?
Open and Upload Design?
Design Name
Open Editor
Upload Design_1 to Device
DESIGN LIBRARY
Design Library
Search your designs ...
Select Existing Libraries option from toolbar
Select Option from Existing Designs


FLOW 2 - DESIGN EDITING
DESIGN EDIT
Current Design
Visual
Motion
Add New Visual
Swipe up to view more options from LED control menu
Tap Options from tab menu to view additional controls
OLED Screen or LED Matrix
Tap to clear or reset design to last save
Current Design
DESIGN EDIT
Current Design
Tap on LED/LEDs to Edit Properties
Hue
Vibrance
Motion
Swipe up to view more options from LED control menu
Tap Options from tab menu to view additional controls
Select Device option from toolbar to return home
Current Design
Hue
Gradient
Motion
Swipe Down to collapse control menu


FLOW 3 - UPLOAD & SAVE
Tap to Save Design and upload to Device
Save and Upload Design?
Upload Design to Device
Save and Upload Design?
Loading...
Design Successfully Loaded?
Save and Upload Design?
Done!
End