UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Shower Music Controller - Team 17

Electrical & Computer Engineering

Team: Shalin Joshi (shalinj2), Varnith Aleti (valet3), Amar Patel (amarcp2)

4/30/2026

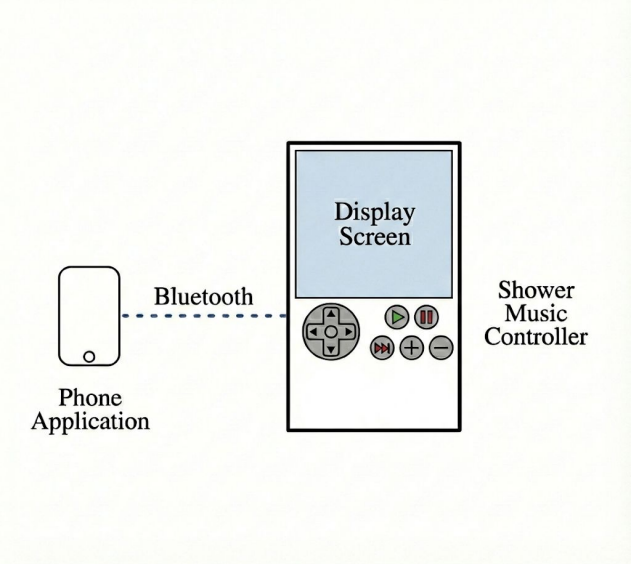
Many people enjoy listening to music in the shower to boost their mood or relax after a long day

However, bringing a phone into the shower to control the music can cause damage from water and steam

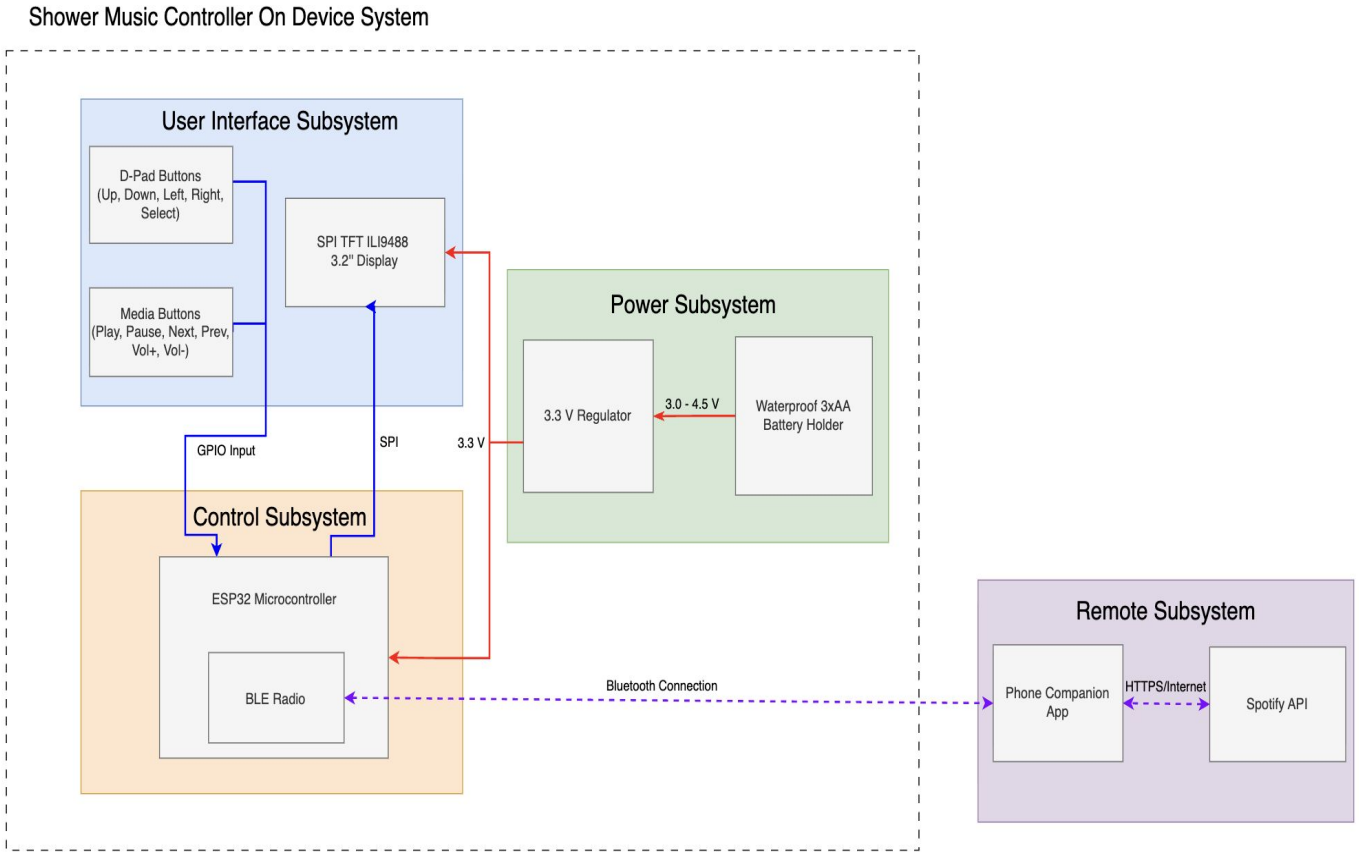
- Existing solutions to this problem involve placing the phone into a protective, waterproof case, but these can be inconvenient because they inhibit the use of the touch screen and often are not very durable.
- Additionally, trying to operate a phone with wet hands is difficult and raises the risk of dropping and damaging it.

Our solution is a music controller device that:

- Can be stuck to a shower wall
- Has a screen and UI which will display your playlists and songs from Spotify
- Has playback control and navigation buttons
- Bluetooth connectivity to a phone companion app
- Water resistant to splashes and steam

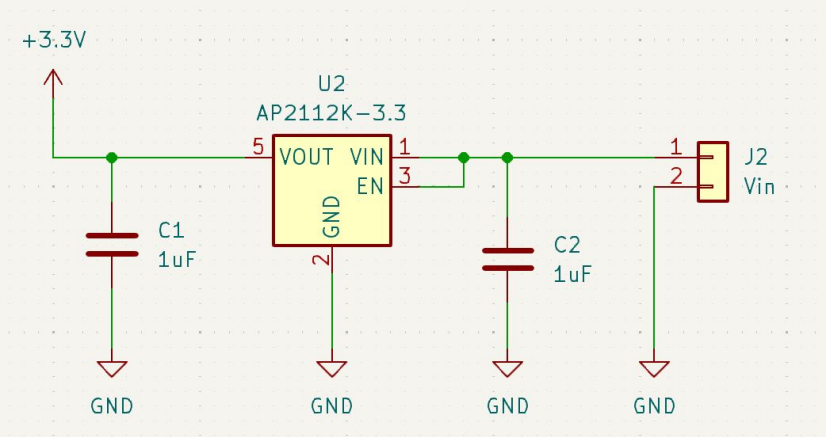


Block Diagram



The power subsystem is responsible for supplying stable electrical power to the electronics in the device

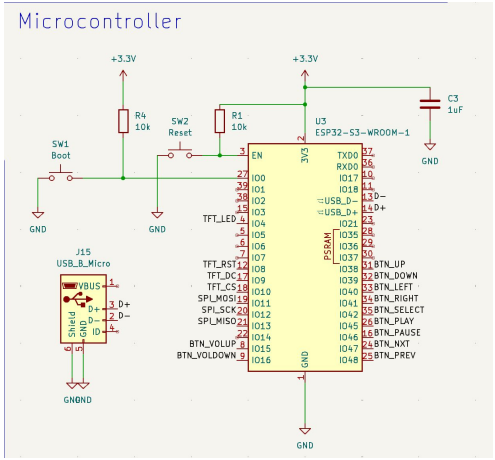
- Powered using 4 AA batteries (6V)
- The voltage will be regulated to 3.3 V using an LDO 3.3V 600MA regulator, which is the required voltage to power the microcontroller and screen.



Control and User Interface Subsystem Design

The control and user interface subsystem work together to process and display user interactions

- **ESP32 Microcontroller:** ESP32-S3 runs all system logic, manages screen states, and handles communication
- **TFT Display (SPI):** 3.2” SPI TFT (ILI9341) displays all system information
- **Waterproof Button Inputs (GPIO):** Reads D-pad and media buttons to navigate menus and control playback
- **Micro-USB**: Used to program microcontroller



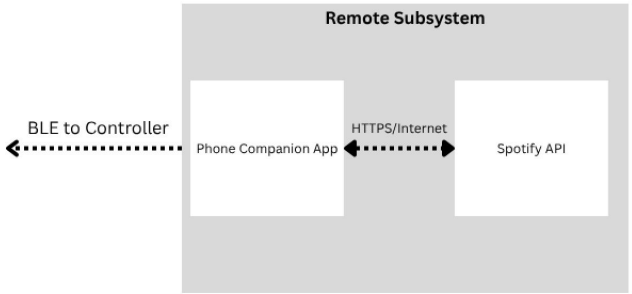
3.2” SPI Module ILI9341

Waterproof Push Button

Remote Subsystem Design

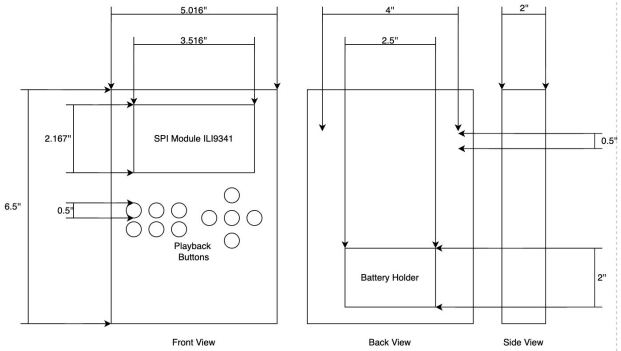
The remote subsystem consists of an iOS app that controls the connection between the controller and Spotify

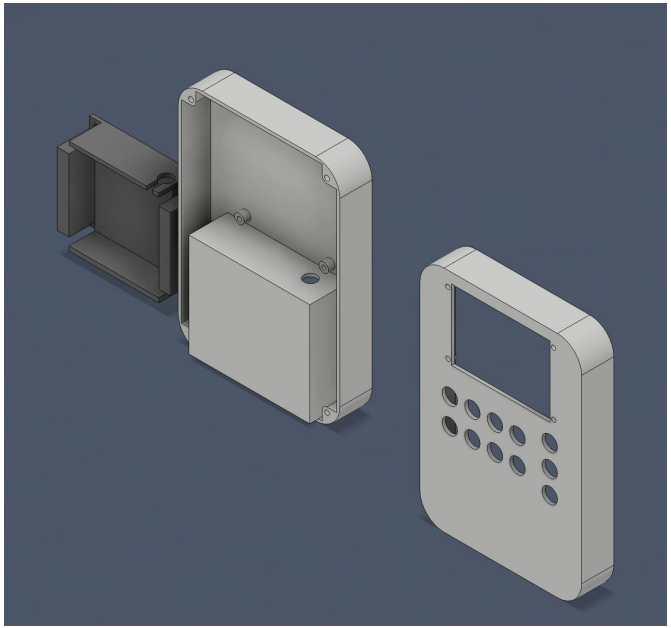
- **User Authentication:** Login to Spotify and asks for data sharing permissions
- **Bluetooth to Device:** Scans for Bluetooth devices to find and connect to the controller
- **Data Exchange:** Passes commands from the controller to Spotify and sends data from Spotify back to the controller
- **Playback Execution:** Makes API calls to Spotify to execute playback actions such as play, pause, skip, etc

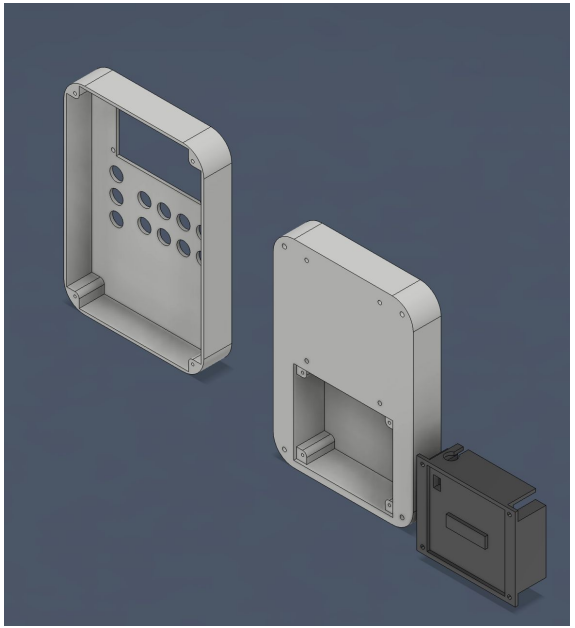


Physical Design

The physical design of the device will be primarily a 3D printed enclosure which holds the subsystems

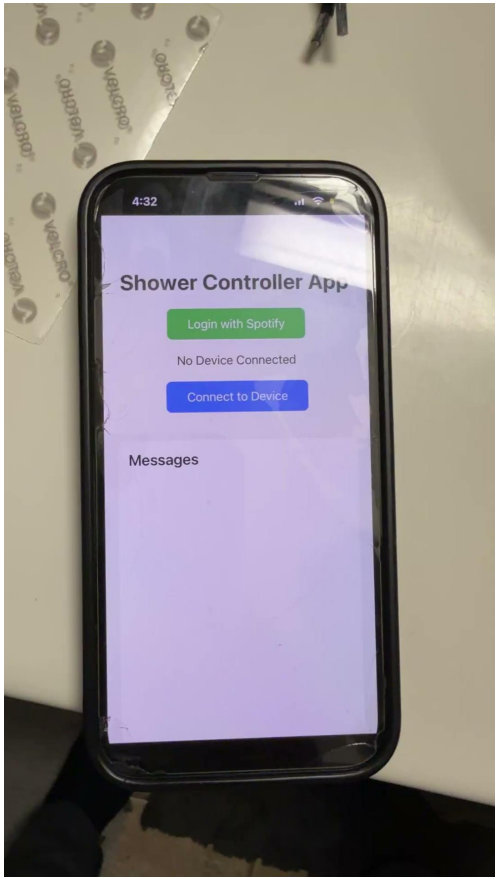




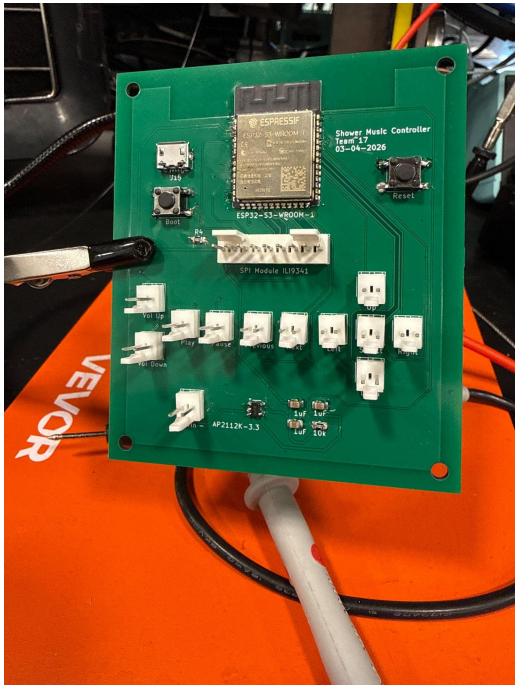


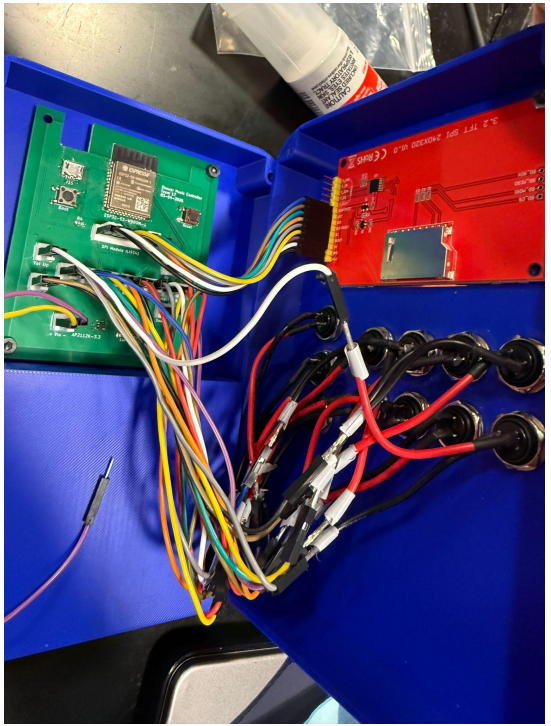
Final Product

Demo Video


4:32
Shower Controller App
Login with Spotify
No Device Connected
Connect to Device
Messages

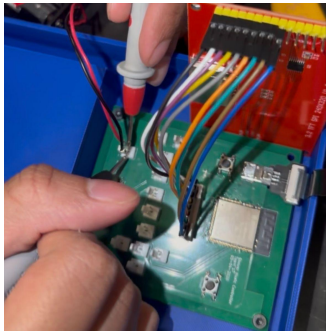
PCB Implementation

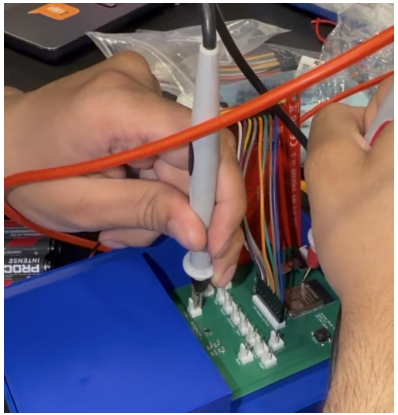




Power Testing

Requirements	Verification
Must accept an input voltage between 4.5 V and 6 V from the 4 AA batteries	1. Connect a variable DC power supply to the input terminals of the power subsystem. 2. Connect the battery connector to the PCB 3. Use a digital multimeter to measure the voltage at the voltage input pins 4. **To pass:** At each point, the measured value must be in between 4.5 - 6 V
Voltage regulator must continuously supply 3.3 V ± 0.2 V to the system	1. Connect the battery connector to the PCB 2. Use a digital multimeter to measure the output voltage of the regulator 3. **To pass:** All measured output voltages on the 3.3 V rail must remain at 3.3 V





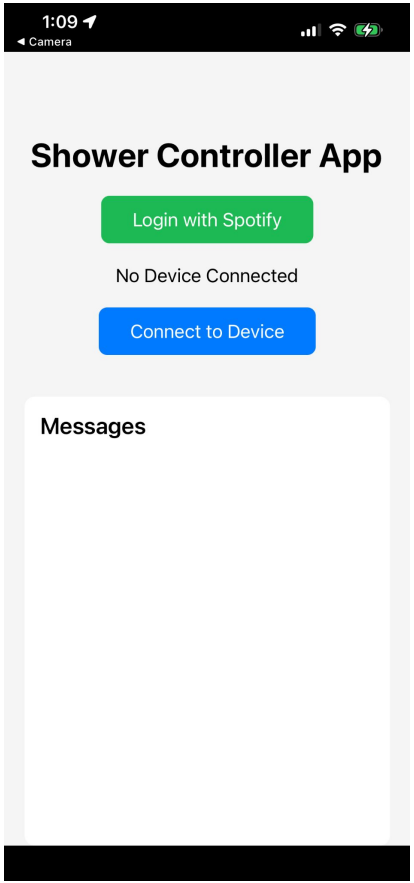
User Interface Implementation & ESP Programming

- **Microcontroller**
  - ESP32-S3-WROOM-1: runs system logic and manages peripherals
  - TFT_eSPI: draw UI (text, shapes, screen rendering)
- **BLE Communication**
  - Sends commands:
    - Playback: PLAY, PAUSE, NEXT, PREV, SHUFFLE, QUEUE_TRACK:x
    - Data: GET_PL_PAGE:x, LOAD_PLAYLIST:, GET_PAGE:x, PLAY_TRACK:x
  - Receives data:
    - P: Playlists
    - T: Tracks
    - N: Now Playing (song|artist|progress|duration)
- **State Machine**
  - Manages screen states and navigation
- **Real-Time Updates**
  - Progress bar updates every second while playing

UI Screen Flow

Remote Subsystem Implementation

- **App Startup**
  - Once user logs in to Spotify, sends authentication token to the API which returns an access token
- **Main Event Loop**
  - Once message is received, decode Base64 to string
  - Interpret command by reading start of string and take appropriate action
- **Sending Data to Controller**
  - Saves all playlists in memory on startup
  - Once playlist is selected, save all songs in that playlist in memory
  - Send playlists and songs in chunks of 5
  - All data is compacted and encoded back to Base64 before being sent
- **Playback**
  - Verifies Spotify access token exists and executes fetch request to the API

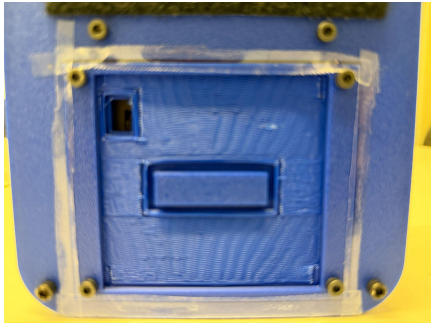


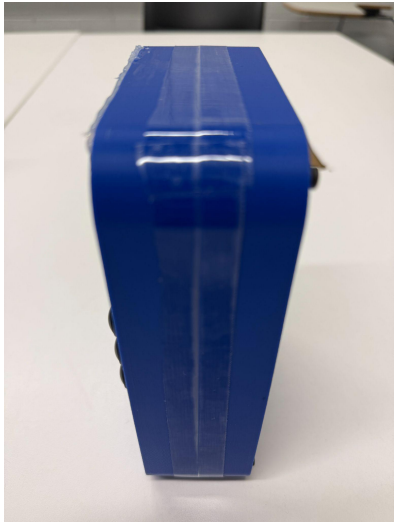
Latency Testing

Requirements	Verification	Results
Must read button inputs and update the display within ≤ 100 ms	1. Record the timestamp of a button press and the time when the screen updates 2. Visually screen must update instantly based on navigation button presses	Up, down, select, back, and now playing button all update screen with no delay
Must send commands to the phone companion app and receive updates on device within ≤ 100 ms	1. Press the play button on the device 2. Record the timestamp of when companion app receives button input 3. Record the timestamp of when companion app sends data back to ESP32	Logs on companion app terminal show that app receives button press and sends data back to ESP32 all within 100 ms

Waterproofing

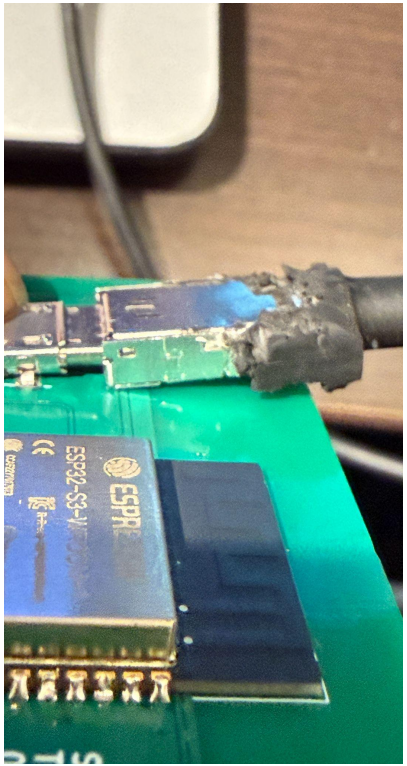
- **Waterproof Buttons:** Screwed from the inside and prevent water from getting in the holes. Tested by pouring water on top of buttons and making sure no water gets inside
- **Acrylic Sheet and Silicone Sealant:** Acrylic sheet to cover the screen and sealed using a silicone sealant on the edges
- **Oil Based Paint Pen:** Water resistant pen to write button logos on buttons
- **Enclosure Edges:** Put Gorilla weather-proof tape around edges of the enclosure
- **Battery Holder:** Waterproof battery compartment. Temporary solution to put tape around edges
- **IP64 rating**: Makes sure that the device is protected from any water splashing in any direction.





Challenges

- **PCB Delays:** PCB round 2 got delayed so had to breadboard whole design to test
- **Micro-USB Port:** Micro-USB port was too far from edge of PCB board
- **Switching from 3 AA to 4 AA:** Had to switch to a 4 AA battery holder which wasn't waterproof
- **Progress Bar:**
  - Had to sync with song timing
  - Parsed the song duration from BLE data, started timer and updated the UI each second to match playback



Conclusion

Reflection:

- Learned importance of early hardware testing + iteration instead of waiting for final PCB
- Start with power system validation first to catch brownout issues earlier
- Learned to adapt quickly when plans fail (Switching buttons, battery holder)

Future Improvements

- Improve waterproofing
- Design custom waterproof battery holder
- Optimize power consumption by using LIPO battery and also use buck boost convertor
- Enchance UI to look more modern
- Add other music providers (Apple Music) and add Android compatibility
- Make enclosure more compact