ECE 445 Project Proposal

Shower Music Controller

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

Date: 2 February 2026

Project #:17

TA: Eric Tang

Professor: Craig Schultz

1. Introduction

1.1 Problem

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.

1.2 Solution

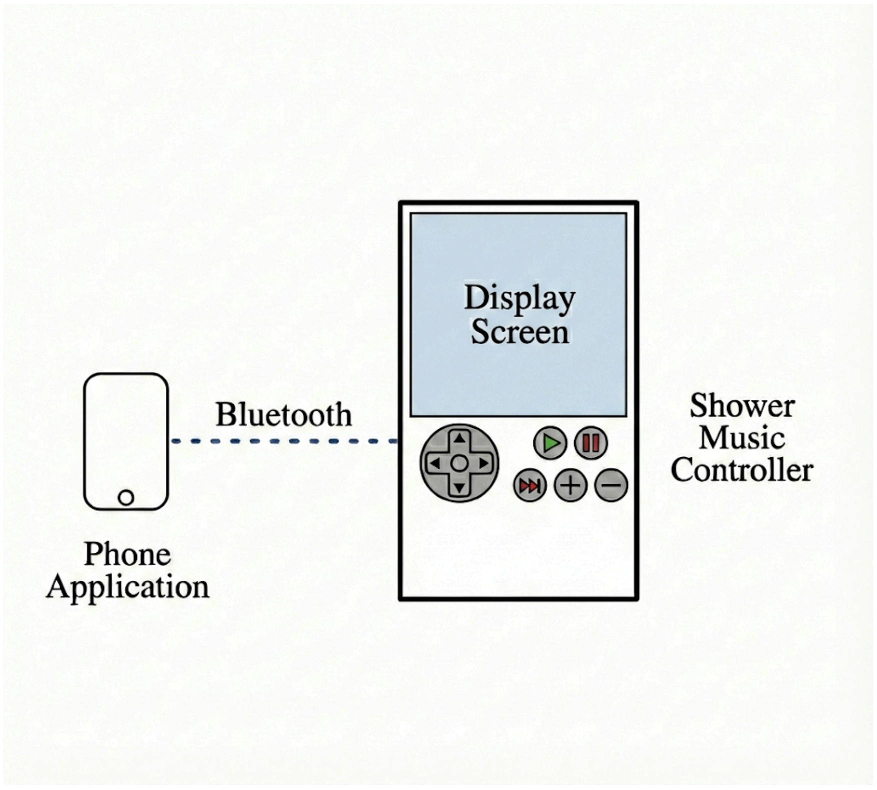
Our solution is a shower music controller that can be stuck to the wall and connects to an app on a phone via Bluetooth.

The controller will have a screen to display the user’s playlists and songs, along with D-pad buttons to navigate the screen, which will provide more reliability than a touch screen. Additionally, the device will have buttons to play, pause, skip, and control volume to allow for full control of the music. The device will also be powered using AA batteries enclosed in a waterproof enclosure to prevent any open ports into the device.

In order to waterproof our device, we will first 3D print an enclosure for the microcontroller, screen, batteries, and buttons. This enclosure will not be waterproof, and instead, we will create a custom plastic container that will be molded to cover the screen and other water-sensitive components. This container will be the primary component that is attached to the shower walls using suction, and our device will be able to be inserted into this container. The buttons will be exposed, but to prevent water damage, they will be covered using a silicone membrane.

The mobile app will connect to the controller using Bluetooth and call the Spotify API to transmit and retrieve all the information necessary to operate the controller. The user can connect their Spotify account in the app to have access to all of their playlists and songs on the controller.

1.3 Visual Aid

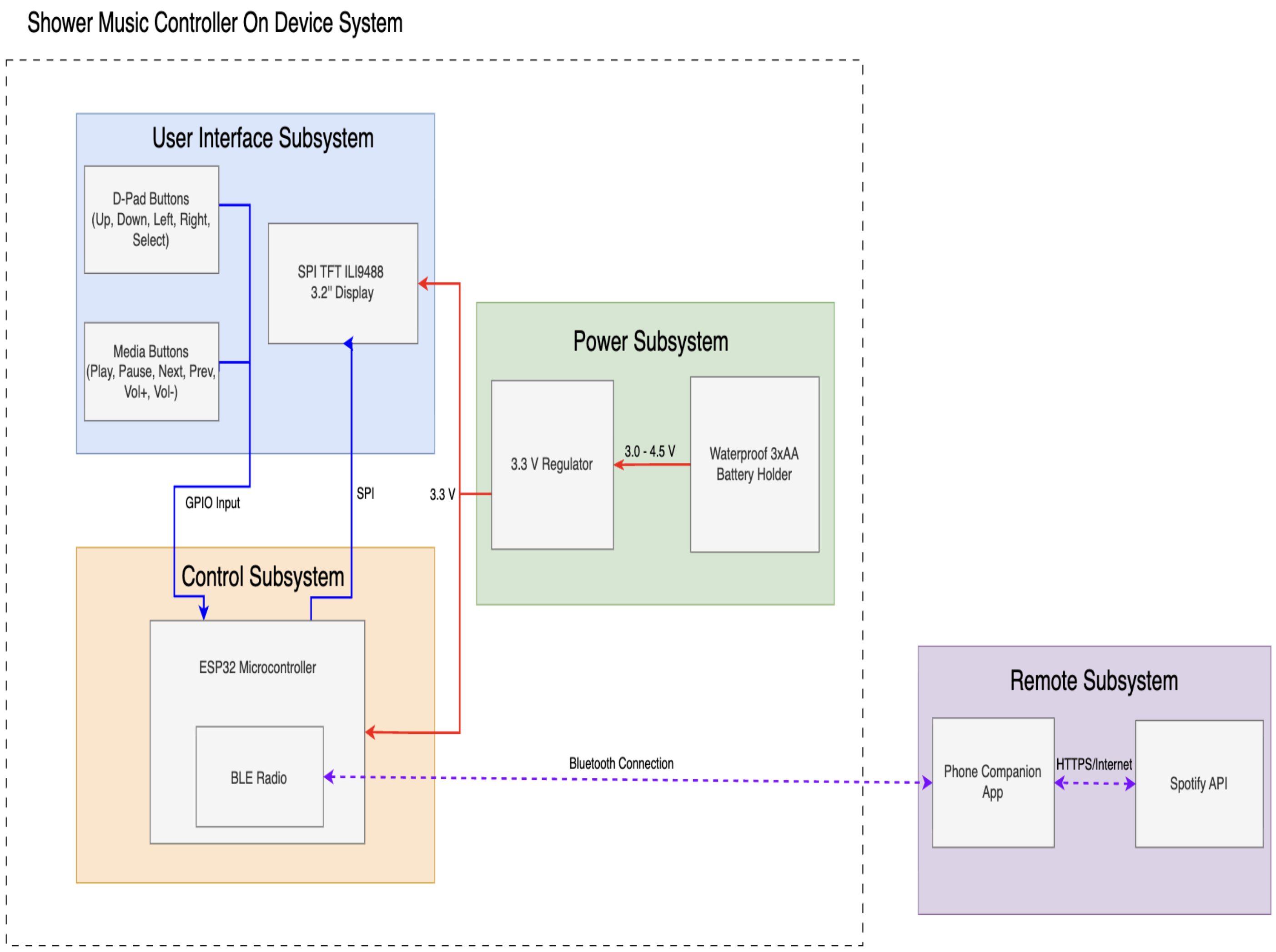


1.4 High-level Requirements List

- The user can successfully perform different playback actions on the controller with a maximum 1 second of delay: Play/Pause, Next Track, Volume Up/Down
- The controller can connect and remain connected through Bluetooth to the phone companion app, and all relevant information from the Spotify API is displayed on the screen
- Controller remains functional after 5 minutes of exposure to shower spray/steam
- Controller operates for at least 2 hours of active use on a full charge

2. Design

2.1 Block Diagram



2.2 Subsystem Overview

Our design is made up of 4 different subsystems, 3 of which are on the physical device, including the power, control, and user interface subsystems, and the last, which is a remote system based on a phone companion app.

2.2.1 Power Subsystem

The power subsystem is responsible for providing a stable power supply to the device electronics. The main power source will be 3 AA batteries, which are stored in a waterproof battery holder with a built-in switch. The battery voltage is regulated by a voltage regulator to 3.3V, which is supplied to both the control and user interface subsystems. This ensures reliable operation of the ESP32 microcontroller and the display regardless of battery voltage variation.

2.2.2 Control Subsystem

The control subsystem is the main processing unit of the device, and it is composed primarily of the ESP32-S3-WROOM microcontroller. The microcontroller will read inputs from the D-Pad and the playback buttons and update the display through the SPI interface. This microcontroller also includes a built-in Bluetooth module, which will be used to wirelessly connect to the remote system phone companion app.

2.2.3 User Interface Subsystem

The user interface subsystem contains the SPI TFT display and the physical buttons for menu navigation and playback control. The buttons send a GPIO signal to the microcontroller on the control subsystem, and the display receives graphical data from the microcontroller via SPI. This subsystem allows the user to navigate menus and control music playback while providing visual feedback.

2.2.4 Remote Subsystem

The remote subsystem contains the phone companion app, which connects to the Spotify API via HTTPS/Internet protocol. The phone app communicates over Bluetooth to the microcontroller, and its primary use is to allow the device to use and access the user's music library.

2.3 Subsystem Requirements

2.3.1 Power Subsystem

The power subsystem is responsible for supplying stable electrical power to the electronics in the device. The system uses three AA batteries inside a waterproof battery holder with a built-in power switch as the primary energy source. 3 AA batteries will be able to provide a voltage range of 3-4.5 V, depending on the battery state. 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.

Interfaces:

- Input: 3.0-4.5 volts from 3 AA Battery holder

- Output: regulated 3.3V to the control and user interface subsystems at ≥ 500 mA

Requirements:
- Must accept an input voltage between 3.0 V and 4.5 V
- Must provide 3.3 V +- 0.2 V to the system
- Must supply at least 500 mA continuously.
- Must include an on/off switch.

2.3.2 Control Subsystem
The Control Subsystem is built around the ESP32 microcontroller and is the main processing unit of the device. It is responsible for reading user inputs from the D-pad and media control buttons. It will run the user interface logic by receiving GPIO signals from the buttons and updating the graphical display using the SPI interface. In addition, the ESP32 includes an integrated Bluetooth Low Energy (BLE) radio. This allows the device to communicate wirelessly with the phone companion app. The subsystem will be able to communicate with the phone and receive playback status and user playlist information from Spotify.

Interfaces:
- Input: 3.3 V from power subsystem, button signals via GPIO
- Output: Display data via SPI
- Wireless BLE link to the phone companion app

Requirements:
- Must read button inputs and update the display within ≤ 400 ms
- Must maintain a BLE connection to the phone during use
- Must send commands and receive updates within ≤ 2 seconds

2.3.3 User Interface Subsystem
The User Interface Subsystem contains the physical and visual interfaces for the device. It consists of a 3.2’’ SPI TFT display, which will show menus, playlists, and playback information. This subsystem will also include a D-pad for navigation and media control buttons for playback. The buttons will allow the user to navigate the menus on the screen and to perform common playback controls while listening to music.

Interfaces:
- Input: 3.3 V from power subsystem, display data from the microcontroller via SPI
- Output: Button signals to the microcontroller via GPIO

Requirements:

- Must register button presses and update the display within ≤400 ms.
- Must support navigation (up/down/left/right/select) and playback controls.

2.3.4 Remote Subsystem

The Remote Subsystem consists of the phone companion app and the Spotify Web API. The phone app acts as the bridge between the device and Spotify as it handles the user authentication and playback for music control. The app communicates with the device using Bluetooth and translates the button commands into the Spotify API which is done through HTTPS/Internet. The app will also communicate back to the device so that the display can be updated.

Interfaces:
- BLE connection to the Control Subsystem
- HTTPS/Internet connection to the Spotify Web API

Requirements:
- Must maintain a BLE connection with the device during operation
- Must execute playback commands within ≤ 2 seconds
- Must support play/pause, skip, volume control, and selection.
- Must maintain consistent communication with Spotify during operation.

2.4 Tolerance Analysis

A risk for this device is that it might not be able to have enough battery power to last the required runtime more than 2 hours of operation. We want to make sure that we do not underestimate the total current draw which would lead to a short battery life and burnouts. Below we will highlight the calculations required to figure out the battery runtime for our device.

Known Values:
- Battery Pack
  - Capacity(C): 2850 mAh
  - Nominal Voltage: 1.5 V per AA = 4.5 V total
- Voltage Regulator
  - Output voltage: 3.3 V
  - Max output current: 600 mA
  - Dropout voltage (max): 0.4 V at 600 mA
- ESP32-S3-WROOM Microcontroller
  - Receiving current: 95-97 mA
  - Transmitting current: 285-355 mA
- 3.2” SPI TFT (ILI9488) display

    - Working current: 90 mA

Using these known values we can calculate the total system current for 3 different use cases.

Case 1: Showing UI and maintaining wireless connection
- 97 mA from the ESP32 and 90 mA from the display. This means that the total current is 187 mA.

Case 2: ESP32 transmitting at high current with a high power radio state with display running
- 355 mA from the ESP32 and 90 mA from the display. This means that the total current is 445 mA.

Using these total current values, we can estimate the battery runtime of the device using: Battery life (hours) = Battery capacity (mAh) / Current draw (mA)

The battery capacity is 2850 mAh, which represents how much charge the AA battery pack can supply over time.

For Case 1 (normal UI and wireless connection):
- Total current = 187 mA
- Battery life = 2850 mAh / 187 mA = 15.24 hours

For Case 2 (high radio transmit power with display running):
- Total current = 445 mA
- Battery life = 2850 mAh / 445 mA = 6.40 hours

Even in the worst-case scenario, the device can operate for over 6 hours, which is well above the required 2-hour runtime.

Next, we consider the voltage regulation limit. The AP2112K-3.3 regulator requires the input voltage to be at least: 3.3 V + 0.4 V = 3.7 V

Since the battery pack uses 3 AA batteries in series, the minimum average voltage per battery is: 3.7 V / 3 = 1.23 V per battery

If the battery voltage drops near this level, the regulator may no longer maintain 3.3 V, which can cause the ESP32 to reset or the display to become unstable. This shows that the main risk is voltage drop under high current, not total battery capacity. However, since the calculated runtime exceeds the requirement, the design is feasible for the intended use.

3. Ethics and Safety

3.1 IEEE Code of Ethics I.1: Safety, Health, and Welfare

Operating electronics in a high-humidity and wet environment poses a risk of electrical shock and device failure. To prioritize user safety, we will use a non-conductive 3D printed enclosure with a silicone membrane to seal the physical buttons themselves. Our design will aim to reach an IP64 rating, which will make sure that the device is protected from any water splashing in any direction. The lithium polymer battery that we will use has the risk of overheating, fire, or leakage in a humid shower environment. We will ensure to implement proper battery protection circuitry, including overcurrent, overvoltage, and thermal protection. Then we will also make sure the charging mechanism is performed in safe conditions, as the users will be instructed to charge the device outside the bathroom environment. During our development process, we will conduct these waterproofing tests in a controlled environment using simulated environmental conditions before actually being used in an actual shower. The device will include clear user warnings about its limitations that it is designed for shower “splashes” and steam exposure.

3.2 IEEE Code of Ethics I.5: Honest Disclosure

We will be transparent and realistic with our users by stating the limitations of our device. As we are aiming for an IP64 rating, we will inform the users that the device is intended only for water splashing and steam, and not for total submersion or direct water jets. We will make sure to communicate potential performance variances, such as certain waterproofing limitations or Bluetooth signal connections in certain scenarios, to the user in a technical manual.

3.3 ACM Code of Ethics 1.6: Respecting Privacy

As our device will interact with the Spotify Web API, which requires access to user account information, we will collect minimal data. In order to have high standards of privacy, our device will only request the minimum information needed to perform the functions that we described our device would be able to perform, such as control playback and retrieve playlist data. We will also have secure authentication and make sure that the user’s Spotify password is never stored or any other personal information. The companion app will clearly request only needed permissions and provide the users with transparent details of what data is accessed and how it will be used.

3.4 Societal and Accessibility Impact

Our project will aim to contribute to a quality of life improvement for users who enjoy listening to music but face the difficulty of controlling the playback of the music while staying in the shower, with wet hands, and non-waterproof phones. Our device will provide a more accessible interface rather than attempting to use a smartphone in wet conditions for which it was not made, which can cause inconveniences. The device will be able to provide a better interface for these conditions with the physical buttons and dedicated controls required for the playback. By creating a dedicated peripheral for humid and wet environments, we help users avoid the risk of damaging expensive primary smartphones and provide a more convenient option rather than stepping out in the middle of a shower.