ECE 445 Project Final Report

Shower Music Controller

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

Date: 6 May 2026

Project #:17

TA: Eric Tang

Professor: Craig Schultz

Abstract

The purpose of this project was to design and build a shower music controller that allows users to control music without needing to bring their phone into the shower. The device connects to a phone companion app through Bluetooth, and the app connects to Spotify to retrieve a user's playlists and songs. The physical device uses an ESP32-S3 microcontroller, a TFT display, waterproof buttons, and 4 AA batteries. The components will be placed inside a 3D-printed enclosure. The user can navigate playlists, select songs, and control playback using physical buttons.

Contents

1. Introduction

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 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.

The high-level objective of this project is to create a reliable shower music controller that allows a user to control their music in a shower environment. The controller should allow the user to successfully perform different playback actions (Play/Pause, Next/Prev Track, etc) on the controller with a maximum 200 milliseconds of delay. The controller should also be able to connect and remain connected through Bluetooth to the phone companion app, and all relevant information from the Spotify API is displayed on the screen. The controller should also remain functional after 10 minutes of exposure to shower spray/steam. Finally, the controller should operate for at least 6 hours of active use on a full charge.

2. Design

2.1 Block Diagram

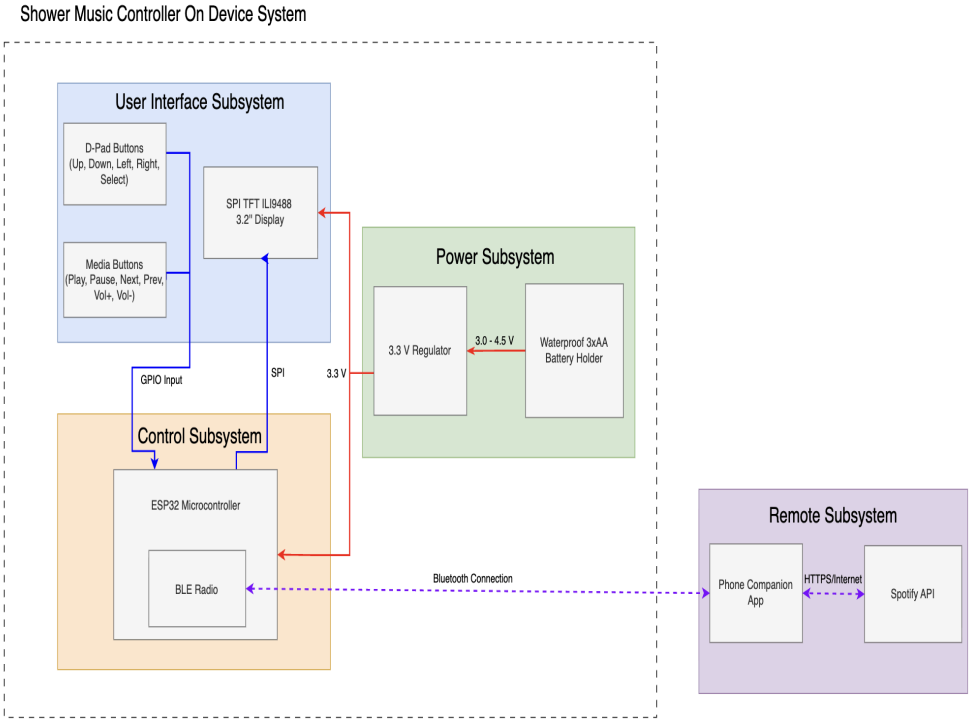

Figure 1. Block Diagram

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 Power Subsystem

The power subsystem is responsible for supplying stable electrical power to the electronics in the device. The system uses four AA batteries inside a battery holder with a built-in power switch as the primary energy source. 4 AA batteries will be able to provide a voltage range of 4.5-6 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.

Originally, in our design, we were going to use 3 AA batteries, which would be able to supply a maximum of 4.5 V. However, during our testing, we discovered brownouts on our device, meaning that the batteries were drawing too much current. Due to this, we explored many different solutions, like using a buck boost converter or a LIPO battery source, but we ended up choosing to

use 4 AA batteries instead of 3. After making the change, our device was able to be powered continuously and had no brownouts.

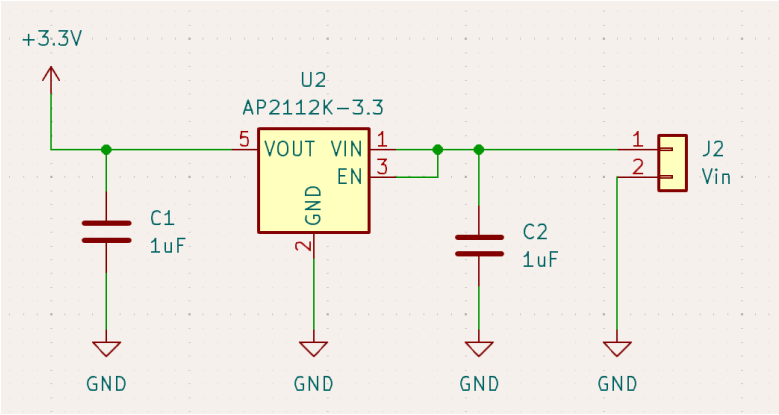

Figure 2. Power Subsystem Circuit

Figure 2 is the circuit diagram of our power subsystem. In this diagram, the Vin power supply, which comes from the AA batteries, is fed into the AP2112k-3.3 regulator. This regulator outputs a regulated 3.3 V rail for the microcontroller and display. 2 1uF capacitors are placed at the input and output terminals of the regulator to provide decoupling.

2.3 Control Subsystem

The control subsystem is the main processing unit of the device, and it is composed primarily of the ESP32-S3-WROOM microcontroller.

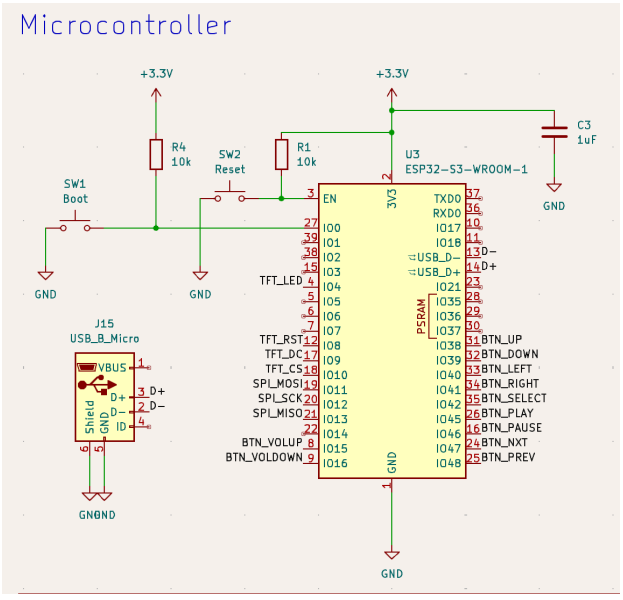

Figure 3. Control Subsystem Circuit

The microcontroller will update the display through the SPI interface, which is connected to the IO pins 8-13 on the circuit diagram. The microcontroller will also read the GPIO signals from the control and playback buttons. We will also need a micro-USB input and a boot and reset button in order to flash code onto the chip. This microcontroller also includes a built-in Bluetooth module, which will be used to wirelessly connect to the remote system phone companion app.

2.4 User Interface Subsystem

The user interface subsystem allows the user to view music information and control the device. This subsystem consists of a 3.2-inch SPI TFT ILI9341 display and waterproof push buttons located on the front of the enclosure. The display is connected to the ESP32-S3 through SPI, and it shows the main menu, playlists, tracks, and now playing screens.

The device has physical buttons rather than a touch screen because the buttons are easier to use with wet hands and better for a water-based environment. The navigation buttons help the user move through the different screens and choose specific songs/playlists. Then the media buttons allow the user to play, pause, skip, and control playback. When clicked, each button sends a unique GPIO signal to the ESP32-S3. The microcontroller then updates the screen or sends a Bluetooth command to the companion application.

Figure 4. Screen Menu UI

The display uses a simple and clean menu system, which is shown in Figure 4. First, the user starts at the main menu and can choose to select either Now Playing or Playlists. When a playlist is selected, the screen updates and displays the tracks within the playlist, which are received from the phone app. The screen updates accordingly once a track is selected and the Now Playing screen is shown, displaying the song name, artist, and progress bar. The progress bar is updated every

second. This is done by parsing the song duration and elapsed time transmitted over BLE from the companion app and incrementing a local timer on the microcontroller.

2.5 Remote Subsystem

The remote subsystem consists of an iOS app that connects to the controller via Bluetooth and the Spotify Web API. The app contains two buttons, one to connect to the user's Spotify account and the other to connect to the controller. Figure 5 shows a screenshot of the app.

Once the user clicks the "Login with Spotify" button, they are prompted to log in to their Spotify account. Additionally, the user is notified of the data that is being collected by the app through Spotify, and they must agree to share this information before proceeding. Once Spotify is connected, the user can connect to the controller by selecting the "Connect to Device" button. The app will initiate a BLE scan and display all discovered devices on the screen, from which the controller can be selected.

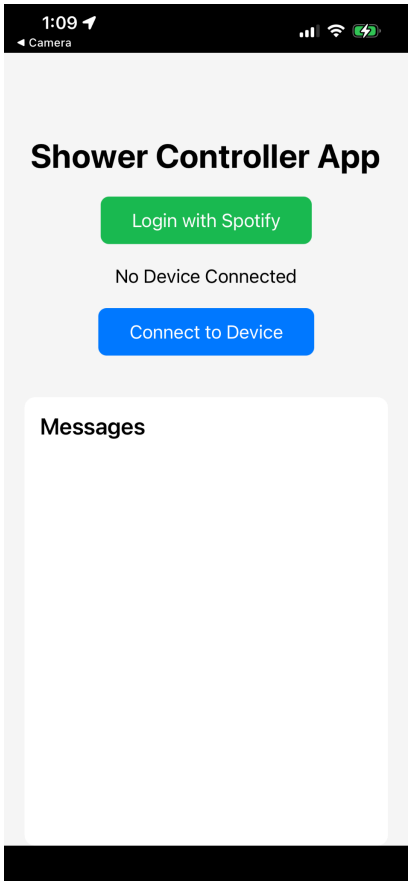

Figure 5. Companion App

After the app is connected to Spotify and the controller, it will begin listening for commands sent by the controller. These commands can be split into two categories: playback and data. Playback commands involve changing the playback state, which includes play, pause, skip, and previous. When these commands are received, the app will send the corresponding request to the API to update the playback state. Data commands request data from Spotify, either songs or playlists, to be sent back to the controller to be displayed. The app sends this data in chunks of five due to limitations on the amount of data that can be sent at once over a BLE connection. To accommodate this, all of the user's playlist data is stored in the app's memory on startup, and when a playlist is selected, all of the songs in that playlist are stored in memory. By taking this approach, it is very easy to send the data in chunks of five when it is requested.

2.6 Physical Design

Figure 6. Final Device Image

As shown in Figure 6, the physical design of our device is primarily based on a 3D-Printed enclosure, which will hold our different subsystems, like the PCB. The enclosure will be approximately 5 inches wide, 6.5 inches in height, and 2 inches in depth. The front side of the device will include the SPI ILI9341 display positioned near the top of the enclosure and an array of playback and menu navigation buttons below it. The back side of the device includes a battery holder compartment that can be removed with screws. It also includes a velcro strip that allows the device to be attached to the shower wall. In order to help waterproof the device, the screen is covered with an acrylic sheet and sealed with silicone sealant on the sides. We used waterproof push buttons, which attach to the enclosure using a screw and bolt from the inside, to ensure that no water gets into the enclosure from the shower.

3. Design Verification

3.1 Power Testing

The power subsystem was tested to verify that the device could accept the right amount of power from the 4 AA battery pack. First, the input pins of the battery were checked using a digital multimeter to make sure that it was between the required range of 4.5 to 6V. The top pictures in Figure 6 show the input pins being tested and the output voltage of 5.85, which is well within the range. Next, the 3.3V power line was tested by checking the 3.3V pin on the ESP32 Microcontroller. In the device, it is required that the voltage regulator must continuously supply 3.3 V ± 0.2 V to the system. Figure 7 also shows this being confirmed on the bottom images, where the multimeter shows 3.29 V, which is within the required range. These tests ensure that the power subsystem is working as intended.

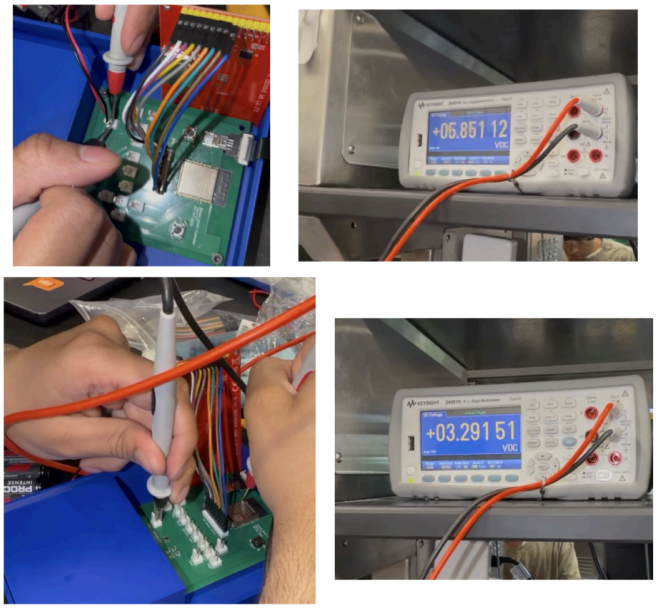

Figure 7. Power Subsystem Testing

3.2 Latency Testing

To test the latency between the companion app and the controller, logs with timestamps were set up in the mobile app to check how much time it takes from when the app receives the “Play Song” command to when it sends the song data back to the controller. In Figure 8, it is shown that it takes 5 milliseconds between these two events, which is well below the threshold of 100 milliseconds set in the requirements. The second latency test that was needed involved the navigation buttons on the controller, and it was observed that these buttons operated with zero delay, satisfying the requirements.

```
LOG  [18:40:35.908] 1. RX FROM ESP32 | PLAY_TRACK:0
LOG  [18:40:35.913] 2. TX TO ESP32 | Sent Now Playing: Skrilla
```

Figure 8. Companion App Logs

3.3 Waterproof Testing

The waterproofing design was tested to verify that the controller could remain protected in a shower environment. The goal was for the device to receive an IP64 rating, meaning the enclosure should protect the electronics from water splashing from any direction. To ensure that the front of the device was protected, waterproof buttons were screwed from the inside of the device. These buttons were tested by pouring water on the top of the buttons and making sure that no water got inside. An acrylic sheet was used to cover the screen and was sealed using a silicone sealant on the edges. This was also tested by pouring water over the edges of the sealant to make sure that no water would go through. Oil-based paint pens were used to write the button logos on the buttons to make sure that the user could know what each button was. For the sides of the enclosure, gorilla weather-proof tape was used to cover the edges of the enclosure. This was also tested by pouring water over the edges. These waterproofing techniques can all be seen in Figure 9.

Figure 9. Waterproofing Device

An issue we encountered was being able to waterproof the back of our device. Originally, a waterproof battery holder was purchased that allowed the device to hold 3 AA batteries. However, it was necessary to upgrade the power system to use 4 AA batteries. It was not possible to acquire a

new waterproof holder in time for the project, so the back of the case was not fully waterproof. As seen in Figure 7, the edges of the battery holder were covered in tape, but the opening where the power switch was located was not covered. With the original waterproof holder, this would not have been an issue. In regards to the IP64 rating, we tested the full device waterproofing by splashing water onto the front and side of the enclosure, but for safety did not test the back.

4. Costs

4.1 Cost Analysis

The total cost for this project can be split into the labor and parts costs individually. The total cost of labor is estimated using the average annual earnings of an Electrical and Computer Engineering graduate from UIUC, which is around $105,000 per year, which comes out to be around $50.48 per hour. As each team member works an average of 10 hours per week for 16 weeks, and applying the standard 2.5x multiplier for overhead and benefits, our cost of labor works out to a total of $60,576.00 for all three team members and $20,192.00 per member :

$$\$50.48 * 2.5 * (16 \text{ weeks} * 10 \text{ hours/week}) = \$20,192.00$$

We have 3 members in our group:

$$3 \text{ members} * \$20,192 = \$60,576.00$$

Then, for the parts, covering all electronic components, custom PCB fabrication, 3D printing, and mechanical hardware required to build the Shower Music Controller. The parts will come out to be around $81.22

4.2 Parts

Table 1

Description	Manufacturer	Part Number	Qty	Unit Cost	Total Cost
ESP32-S3-WROOM	Espressif	ESP32-S3-WROOM-1-N16	1	$6.50	$6.50
1uF 0603 Capacitor	YAGEO	C0603C105K9PAC7867	3	$0.10	$0.30
10KΩ 1% Resistor	Stackpole	RMCF0805JG10K0	1	$0.10	$0.10
5.1KΩ Resistor	YAGEO	CFR-25JT-52-5K1	2	$0.10	$0.20
3.2” SPI TFT Display (ILI9341)	Hosyond	ILI9341-3.2-SPI	1	$16.99	$16.99

Tactile Push Buttons (D-Pad & media)	Twidec	PBS-33B-BK-X	2	$9.99	$19.98
AP2112k-3.3 Voltage Regulator	Diodes Inc.	AP2112K-3.3TRG1	1	$0.22	$0.22
AA Battery Holder (3xAA) Waterproof, On/Off Switch	Jex Electronics	BHWAA3	1	$4.99	$4.99
Silicone Sealant	KPPTYTY	N/A	1	$7.97	$7.97
Acrylic Sheet	Art3D	B09Y2ZGRRB	1	$4.99	$4.99
Velcro Strip	VELCRO	B09V5FV7H8	1	$5.53	$5.53
3D Printing Filament (PLA)	OVERTURE	N/A	1	$13.98	$13.98
AA Batteries (Energizer, 3-Pack)	Energizer	E91BP-3	2	$4.99	$9.98
Boot/Reset Button	C&K	PTS645SL43SMTR92 LFS	2	$0.36	$0.72
Micro-USB Port	Molex	WM11263TR-ND	1	$0.87	$0.87
Total					**$93.32**

5. Conclusion

5.1 Accomplishments

The final product operated as intended and was able to meet the high-level requirements along with each of the subsystem requirements. The device is able to connect via Bluetooth to the companion app, connect to a user's Spotify account, display the user's playlists and songs, allow the user to select any song from their playlist, and be able to control their music using different playback buttons. The device can also successfully withstand water splashes from the front and the side and can last on battery for over 6 hours.

5.2 Uncertainties

For our device, the main uncertainty is the complete waterproofing of the rear panel. The original design had a waterproof 3xAA battery holder enclosure, but we had a mid-project switch to 4xAA batteries, and we were unable to get a waterproofed one, similar to the original 3xAA waterproofed holder, in the given time frame. So we used a non-waterproof replacement and taped the edges as a temporary measure. Although this is the case, the opening around the power switch remains exposed to the environment. The front and sides were tested by direct water splash with no leaks observed, but the back was not subjected to the same test out of caution for electronics. Then, the long-term durability of the silicone sealant around the acrylic screen covered under repeated seam and humidity exposure has not been fully tested as well.

5.3 Ethical Considerations

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.

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.

5.4 Future Work

For future work of the Shower Music Controller, there are several improvements that would strengthen the device. The most important change would be to use a waterproofed 4xAA battery holder to close the gap on the rear panel. For the power itself, we can replace the AA battery stack with a LiPo cell paired with a buck-boost converter. This would provide more stable voltage regulation across the discharge curve and allow for a slimmer and compact enclosure. For the PCB, the micro-USB should be repositioned to the edge of it so that it would be more accessible to the cable and be more convenient in general. On the User Interface side of things, we could redesign the screens and overall aesthetics to be more modern. Also, we could make the device compatible with Apple Music and Android devices as well, so that we can broaden the user base beyond our current iOS and Spotify-only implementation.

References

[1] Espressif Systems, ESP32-S3-WROOM-1/WROOM-1U Datasheet, Espressif, 2023. [Online]. Available: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[2] LCD Wiki, 3.2inch SPI Module ILI9341 SKU: MSP3218, LCD Wiki. [Online]. Available: https://www.lcdwiki.com/3.2inch_SPI_Module_ILI9341_SKU:MSP3218

[3] Twidec, Waterproof Momentary Pre-soldered Pushbutton Switch PBS-33B-BK-X, Amazon. [Online]. Available: https://www.amazon.com/Twidec-Waterproof-Momentary-Pre-soldered-PBS-33B-BK-X/dp/B08JHW8BPV

[4] IEEE, IEEE Code of Ethics, IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8

[5] Association for Computing Machinery, ACM Code of Ethics and Professional Conduct, ACM, 2018. [Online]. Available: https://www.acm.org/code-of-ethics

[6] Nature's Generator, AA Battery Voltage: What You Should Know, Nature's Generator Blog. [Online]. Available: https://naturesgenerator.com/blogs/news/aa-battery-voltage

[7] Diodes Incorporated, *AP2112K-3.3 Low Dropout Regulator Datasheet*, 2021. [Online]. Available: https://www.diodes.com/assets/Datasheets/AP2112K.pdf.

[8] Yageo, *CFR-25JT-52-5K1 5.1kΩ Through Hole Resistor*, Digi-Key Electronics. [Online]. Available: https://www.digikey.com/en/products/detail/yageo/CFR-25JT-52-5K1/16674047

[9] GCT (Global Connector Technology), *USB4085-GF-A USB Type-C Receptacle Connector*, Digi-Key Electronics. [Online]. Available: https://www.digikey.com/en/products/detail/gct/USB4085-GF-A/9859662

[10] C&K, *PTS645SL43SMTR92-LFS Tactile Switch*, Digi-Key Electronics. [Online]. Available: https://www.digikey.com/en/products/detail/c-k/PTS645SL43SMTR92-LFS/3861373

[11] Twidec, *Waterproof Momentary Pushbutton Switch PBS-33B-BK-X*, Amazon. [Online]. Available: https://www.amazon.com/Twidec-Waterproof-Momentary-Pre-soldered-PBS-33B-BK-X/dp/B08JHW8BPV

[12] Jex Electronics, *Waterproof 3x AA Battery Holder with Switch*, Amazon. [Online]. Available: https://www.amazon.com/Jex-Electronics-Waterproof-Battery-Holder/dp/B0CXJZD4YY

[13] HiLetgo, *3.2" TFT LCD Display Module (ILI9488 SPI)*, Amazon. [Online]. Available: https://www.amazon.com/dp/B0B1M9S9V6

[14] Generic Manufacturer, *Shower Caddy Suction Cup Replacement Connectors*, Amazon. [Online]. Available: https://www.amazon.com/Shower-Caddy-Connectors-Replacement-Compatible/dp/B0DGT6X5TX

Appendix

Power Subsystem	
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
Must include an on/off switch	1. Toggle the built-in battery holder switch to the "OFF" position. 2. Measure the voltage at the input of the LDO regulator using a digital multimeter. 3. Toggle the switch to "ON" and verify voltage presence. 4. Verify the ESP32/Screen powers down completely when "OFF". 5. **To pass:** "OFF" state must measure 0.0 V ± 0.01 V at the regulator input, and "ON" state must measure >3.3 V

Control Subsystem	
Requirements	**Verification**
Must read button inputs and update the display within ≤ 200 ms	1. Record the timestamp of a button press and the time when the screen updates 2. Repeat for 5 consecutive presses. 3. **To pass:** All 5 measurements must be ≤ 200 ms
Must maintain a BLE connection to the phone during use	1. Pair the ESP32 with the companion phone app. 2. Simulate active use by toggling D-Pad buttons once per minute. 3. Monitor the Arduino serial monitor for "BLE Disconnected" flags or app-side dropouts. 4. **To pass:** 0 disconnected signals over a 15 min period

Must send commands to the phone companion app and receive updates within ≤ 400 ms	1. Press the play button on the device 2. Record the timestamp of the outgoing BLE command packer from the serial monitor 3. Record the timestamp when the ESP32 receives the corresponding update from the phone and calculate the difference 4. Repeat for different buttons on the device 5. **To pass:** Average of all differences must be less than 400 ms

User Interface Subsystem	
Requirements	**Verification**
Must register and transmit unique GPIO signals for each button input	1. Connect a digital multimeter to the GPIO output of the board 2. Press each button one by one 3. Verify that the corresponding pin goes to the correct logic state 4. **To pass:** All buttons must produce the correct logic transition when pressed
Must update display within ≤ 200 milliseconds of a button input	1. Use a smartphone camera to record the screen while pressing an input button 2. Review the recording and confirm the screen updated within 200 ms of the button press 3. Repeat for 5 button presses 4. **To pass:** Screen updates within 200 ms for all 5 button inputs

Remote Subsystem	
Requirements	**Verification**
Must translate a BLE command to a Spotify API request and execute the action within 200 milliseconds.	1. Open the Spotify Desktop app on a PC to visually monitor playback status. 2. Simultaneously, press "Skip" on the ESP32 hardware. 3. Using a stopwatch, measure the time from the physical button press until the track changes on the Spotify Desktop app. 4. Repeat for 10 trials to account for network variability. 5. **To pass:** Average latency must be less than 200 ms.

Must successfully execute play/pause, skip song, previous song, add to queue, shuffle, and selection.	1. Trigger each command by pressing corresponding button on device 2. Verify corresponding change on Spotify 3. **To pass:** All functions must execute correctly on Spotify
Must fetch and send all users playlist and song information to the microcontroller	1. Add a log to show which playlists and their songs are being sent to the microcontroller 2. Double check users Spotify account to ensure that all playlists are being sent and displayed on the screen 3. **To pass:** All users playlists and songs must be displayed correctly on device screen