ECHOPOINT

Contact: Andrew D. Marques

Website: www.AndrewDMarques.com

Email: _____
_____

Address: _____
_____

Phone: _____

Version: 2
Updated: 4/24/2025

INTRODUCTION

This manual serves as a comprehensive guide for using and maintaining EchoPoint. EchoPoint is an audio playback board. One of EchoPoint's defining features is that it can be battery operated, or powered using a 6V DC source. Using the supplied 6V 6Ah LiFePO4 battery, an expected battery life is 107 active hours, 6,400 button presses, or 3-5 months in a museum setting before charging.Batteries recharge in about 5 hours with the supplied battery charger.

Core features include:

1. **Battery Operated** - external power source not required
2. **Deep Sleep Mode** - device enters deep sleep after 1 minute of inactivity*
3. **Volume Control** - adaptable to most spaces
4. **Customizable Audio** - any mp3 file < 1GB large

*If your audio files are longer than one minute, this can be adjusted to not crop the audio files.

Primary contributors include:

Individual	Contribution
Andrew D. Marques	Electronics, software development, hardware, and initial prototyping

TABLE OF CONTENTS

QUICKSTART GUIDE

Operating Steps:

1. Press any of the buttons to wake up the device and begin hearing audio.
2. Continue to press the other buttons to sample audio files, or press the same button to replay the previous message.

Recharge Batteries:

1. Connect the 2A 7.2V lithium battery charger to the wall outlet.
    a. The LED indicator should be green.
2. Toggle the switch in the back of the EchoPoint from "Audio" to "Silence/Charging"
3. Connect the charger to the EchoPoint using the charging port.
    a. The LED indicator should now be red.
4. When the EchoPoint is fully charged the LED indicator on the battery charger should be green.
5. Once fully charged, the battery charger can be disconnected, toggle switch change to "Audio" and the EchoPoint can be operated once again.

Change Volume:

1. It is recommended to change the audio file volume using software like Audacity:
    a. Select just the bottom waveform and Effect → Volume and Compression → Amplify.
2. If you want to change the audio on the amplifier, follow the steps below to open the EchoPoint and use the diagram to locate the amplifier potentiometer to change the playback volume using a phillips screwdriver. Turning to the right increases the volume and turning to the left decreases volume.

Change Audio Programs:

1. Disconnect the fuse.
2. Remove the MicroSD card from the back of the electronics box.
3. Rename the new audio files before transfer:
    a. Ensure that the audio files are (1) mp3 audio format and (2) contain four "0" before them (i.e. 00001.mp3).
4. Connect the MicroSD card to your computer using an adapter if necessary.
5. Remove all old audio files.
6. Upload the new audio files with the correct naming scheme.
7. Eject the MicroSD card from your computer.
8. Place the MicroSD card card back into the EchoPoint.
9. Reconnect the fuse.
10. Test if the audio files need the volume adjusted

a. If possible, adjust the audio file volume using software like Audacity rather than opening the unit and changing the potentiometer dial.

Opening the Electronics Box

1. Unscrew the fuse from the electronics box before beginning work.
2. Unscrew the 6 phillips screws on the underside of the electronics box.
3. Carefully lift the plaque and top of the electronics box off of the base of the stand.
    a. NOTE: There are red and brown wires that must be carefully disconnected to allow full separation of the two halves (speaker to amplifier wires).
4. After the speaker has been disconnected, the top half can be set on a table. All electronics remain in the top half of the stand except for the speaker.

Closing the Electronics Box

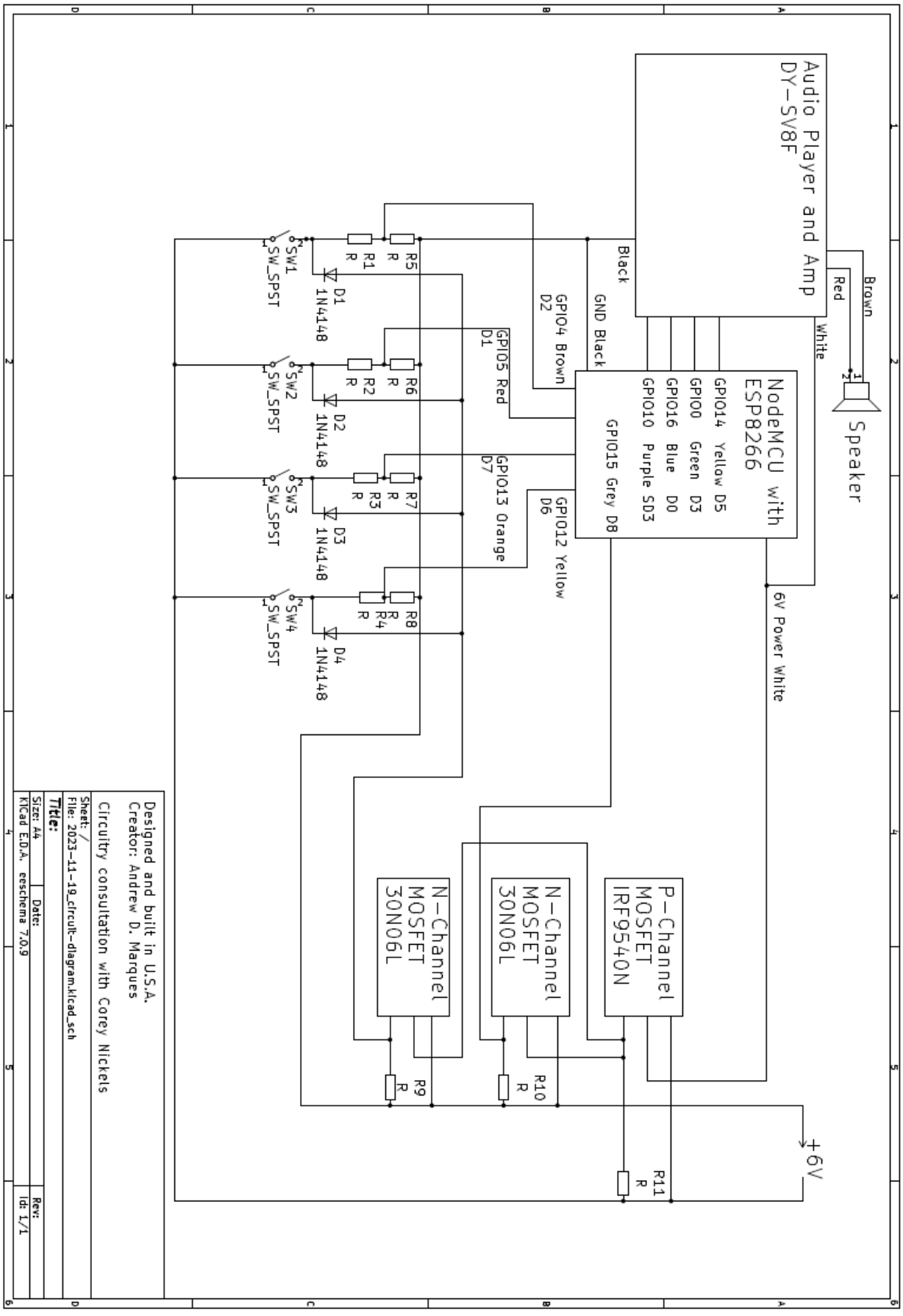
1. Check that the fuse is disconnected before beginning work. If it is not disconnected, unscrew the fuse from the box before work.
2. Position the top half of the stand carefully on the bottom half.
3. Reconnect the dupont connectors (red to red and brown to brown).
4. Position the top half of the stand so that the pegs from the lower half fit in and ensure that no wires are pinched between the two halves.
5. Screw the 6 phillips screws to the bottom of the stand to secure it.
6. Reconnect the fuse and test that the stand is working.

Long-Term Storage:

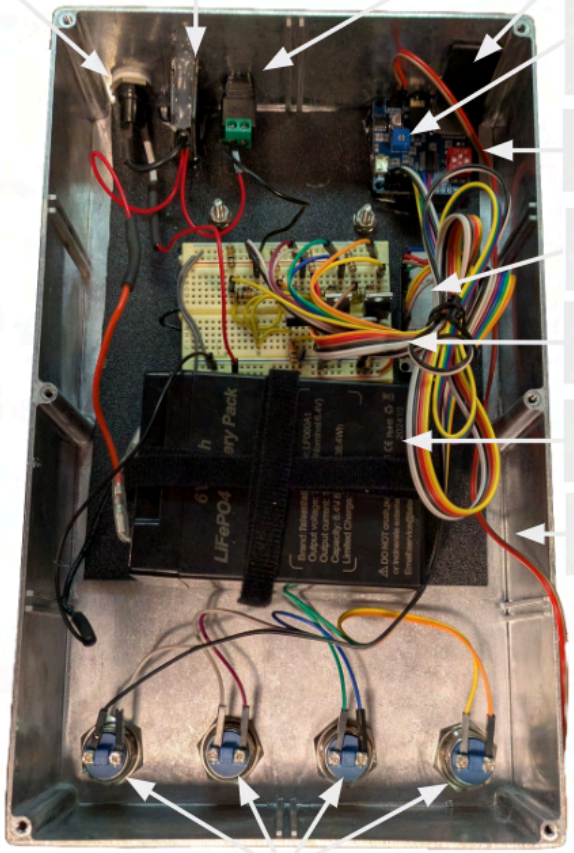
1. EchoPoint should be stored in dry cool spaces without rodents. This device should be thought of like a computer, so make sure that it is stored appropriately.
2. Check the integrity of the wiring connections.
3. Consider labeling the box with the combination code so that it can be opened in the future.

DIAGRAMS

Above shows the controls for this EchoPoint including the fuse, toggle switch for audio or charging, charging port for 7.2V 2A lithium battery charger, and MicroSD ejector slot.


Audio Player and Amp
DY-SV8F
Brown
Red
Speaker
White
Black
GND Black
GPIO4 Brown
D2
GPIO5 Red
D1
GPIO13 Orange
D7
GPIO12 Yellow
D6
NodeMCU with
ESP8266
GPIO14 Yellow D5
GPIO0 Green D3
GPIO16 Blue D0
GPIO10 Purple SD3
GPIO15 Grey D8
6V Power White
SW1
SW2
SW3
SW4
SW_SPST
R1
R2
R3
R4
R5
R6
R7
R8
R
D1
D2
D3
D4
1N4148
P-Channel
MOSFET
IRF9540N
N-Channel
MOSFET
30N06L
N-Channel
MOSFET
30N06L
R9
R10
R11
+6V
Designed and built in U.S.A.
Creator: Andrew D. Marques
Circuitry consultation with Corey Nickels
Sheet: /
File: 2023-11-19_circuit-diagram.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. eeschema 7.0.9
Id: 1/1

Main Internal Components

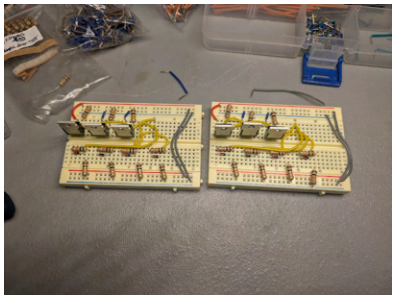


BUILD DOCUMENTATION

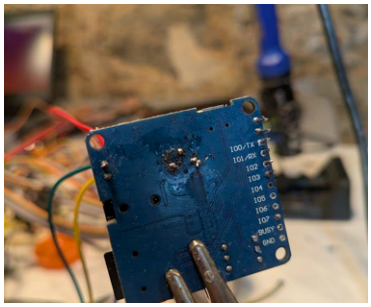
3/4/2025
Meeting at the Boyertown Museum of Historic Vehicles with description of project described here.

3/8/2025
Initial designing, planning, and ordering

3/22/2025
Assembling electronics (pictured are EchoPoint 16 and EchoPoint 17 breadboards).



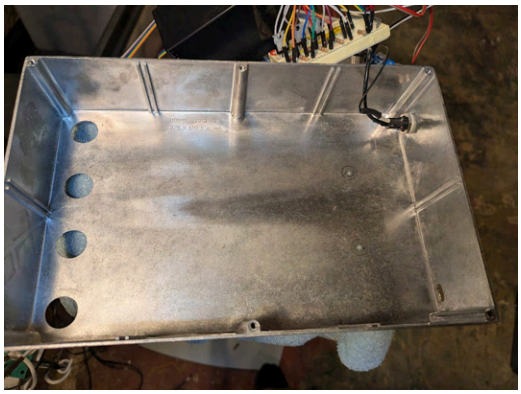
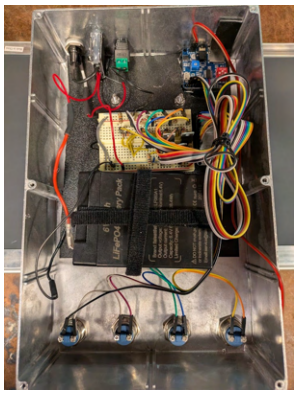
3/28/2025
Failed attempt of connecting volume potentiometer to dial for outside of electronics box control. Results were poor sound quality. Further manipulation resulted in complete destruction of the audio board.



3/29/2025
Executing design plan for mounting hardware to the electronics box.

4/9/2025
Assembly and mock up without working electronics. Emailed with Kendra and approved designs.


4/12/2025
Completed assembly of electronics and mounting components.

4/18/2025
Begin compiling notes into project documentation.

4/24/2025
Completed compiling notes of project description.

4/25/2025
EchoPoint delivered to Boyertown Museum of Historic Vehicles.

KEY SERVICEABLE MATERIALS

- Metal case https://www.amazon.com/BUD-Industries-CN-5714-Aluminum-Enclosure/
- Fuse https://www.amazon.com/dp/B0C65MFNPY?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_2&th=1
- Toggle Switch https://www.amazon.com/dp/B075RC6TFB?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_45&th=1
- Velcro https://www.amazon.com/dp/B00O2A3P88?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_1&th=1
- Screws https://www.fastenere.com/8-32-slotted-flat-head-machine-screws-stainless-steel-18-8?variant_id=4541&device=c&network=g&keyword=&creative=748025735257&placement=&gad_source=1&gad_campaignid=22475900771&gbraid=0AAAADDwjuWnTd3Ju2MaTJN7UBNVXgrwN&gclid=CjwKCAjwwqfABhBcEiwAZJjC3ioPGiAZEF5EFfb9SvJohh52zTKdAHTDXMbvxQ-xXeD735PqyNALvxoCEWcQAvD_BwE
- Lock nuts https://www.fastenere.com/standard-thin-nylon-insert-hex-jam-lock-nuts-stainless-steel-18-8?utm_source=google&utm_medium=paid&utm_campaign=20927741921&utm_content=&utm_term=&gadid=&gad_source=1&gad_campaignid=20931129838&gbraid=0AAAADDwjuXHrSKRow5HdokUy85EOoDl2&gclid=CjwKCAjwwqfABhBcEiwAZJjC3pxHnrnRZoR0621qOBvCHhl8eawS5pTaTe6i9nR2lJg5IWXNjpObshoC1iIQAvD_BwE
- Washers https://www.amazon.com/dp/B07ZZ5NJJ9?ref_=ppx_hzsearch_conn_dt_b_fed_asin_title_2&th=1
- Metal speaker grill https://www.amazon.com/gp/product/B0B1LFCDSH/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&th=1
- Speaker
- Dupont wires https://www.amazon.com/dp/B07GD1TH2K/ref=redir_mobile_desktop?_encoding=UTF8&ref_=ya_aw_od_pi&th=1
- Breadboard https://www.amazon.com/dp/B00B8861R4?psc=1&ref=ppx_yo2ov_dt_b_product_details
- 10K Ohm resistor https://www.amazon.com/gp/product/B08FHPKF9V/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
- Diodes https://www.amazon.com/gp/product/B00N1ZKU7E/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
- MOSFET Transistor 30NO6L https://www.amazon.com/gp/product/B07WHSD3GJ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
- MOSFET Transistor IRF9540N https://www.amazon.com/gp/product/B082J3F8HJ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1
- Playback Module DV-SV8F https://www.amazon.com/dp/B0BL9SDJPW?ref=ppx_yo2ov_dt_b_product_details&th=1
- 6V 6Ah LiFePO4 Battery Pack LF060A1 https://www.amazon.com/dp/B07P8LCSFJ?psc=1&ref=ppx_yo2ov_dt_b_product_details
- Plastic
- ABS Plastic https://www.amazon.com/gp/product/B08R9W4NZW/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&th=1
- 19mm metal push buttons https://www.amazon.com/gp/product/B09BKXXRD4/ref=sw_img_1?smid=ALOZL3MQGX35O&th=1

OPERATING SYSTEM CODE

NodeMCU code: **echopoint_v5.42.ino** is run on the NodeMCU microchip. It communicates with the power transistor circuitry to cut power to itself and the amplifier after 1 minute of inactivity. It also communicates with the audio amplifier and sound board to select which file to play.

```
// Define button pin constants
const int button1Pin = 4;
const int button2Pin = 5;
const int button3Pin = 13;
const int button4Pin = 12;

// Define the output pins for audio
const int audioPin1 = 14;
const int audioPin2 = 0; //trying to fix this, testing 3,1,and now 0, 0 works
const int audioPin3 = 16;
const int audioPin4 = 10;


// Define the pin for signal
const int signalPin = 15;

// Variable to keep track of the last time a button was pressed
unsigned long lastButtonPressTime = 0;

// Variable to check if it's the first time the loop is running
bool firstRun = true;

void setup() {
  // Initialize the signal pin as an output and set to HIGH
  pinMode(signalPin, OUTPUT);
  digitalWrite(signalPin, HIGH);

  // Initialize button pins as inputs
  pinMode(button1Pin, INPUT);
  pinMode(button2Pin, INPUT);
  pinMode(button3Pin, INPUT);
  pinMode(button4Pin, INPUT);

  // Initialize audio pins as outputs
  pinMode(audioPin1, OUTPUT);
  pinMode(audioPin2, OUTPUT);
  pinMode(audioPin3, OUTPUT);
  pinMode(audioPin4, OUTPUT);

  // Start serial communication at 9600 baud rate
  Serial.begin(9600);

  // Initialize the last button press time to the current time
  lastButtonPressTime = millis();
}

void loop() {
  // Read the state of each button pin
  int button1State = digitalRead(button1Pin);
  int button2State = digitalRead(button2Pin);
  int button3State = digitalRead(button3Pin);
  int button4State = digitalRead(button4Pin);

  // If it's the first run of the loop, pause to allow sound board to boot up.
  if (firstRun) {
//    delay(10);
//    int button1State = digitalRead(button1Pin);
//    int button2State = digitalRead(button2Pin);
//    int button3State = digitalRead(button3Pin);
//    int button4State = digitalRead(button4Pin);

    digitalWrite(audioPin1, HIGH);
    digitalWrite(audioPin2, HIGH);
    digitalWrite(audioPin3, HIGH);
    digitalWrite(audioPin4, HIGH);

    delay(500); // Delay on the first run only
    Serial.println("first loop");
    Serial.println(button1State);
    Serial.println(button2State);
    Serial.println(button3State);
    Serial.println(button4State);
    firstRun = false; // Set the first run to false so the delay doesn't happen again
    // Handle the state of audioPin1 based on button1Pin
    if (button1State == HIGH) {
      digitalWrite(audioPin1, LOW);
      Serial.println("Audio pin 1 set to LOW because button 1 is HIGH.");
      delay(100); // Allow the nodeMCU to send the low signal.
      digitalWrite(audioPin1, HIGH);
    } else if (button2State == HIGH) {
      digitalWrite(audioPin2, LOW);
```

```
      Serial.println("Audio pin 2 set to LOW because button 2 is HIGH.");
      delay(100); // Allow the nodeMCU to send the low signal.
      digitalWrite(audioPin2, HIGH);
    } else if (button3State == HIGH) {
      digitalWrite(audioPin3, LOW);
      Serial.println("Audio pin 3 set to LOW because button 3 is HIGH.");
      delay(100); // Allow the nodeMCU to send the low signal.
      digitalWrite(audioPin3, HIGH);
    } else if (button4State == HIGH) {
      digitalWrite(audioPin4, LOW);
      Serial.println("Audio pin 4 set to LOW because button 4 is HIGH.");
      delay(100); // Allow the nodeMCU to send the low signal.
      digitalWrite(audioPin4, HIGH);
    }
  } else {
    // Handle the state of audioPin1 based on button1Pin
    if (button1State == HIGH) {
      digitalWrite(audioPin1, LOW);
      Serial.println("Audio pin 1 set to LOW because button 1 is HIGH.");
      lastButtonPressTime = millis();
    } else {
      digitalWrite(audioPin1, HIGH);
  //    Serial.println("Audio pin 1 set to HIGH because button 1 is LOW.");
    }

    // Handle the state of audioPin2 based on button2Pin
    if (button2State == HIGH) {
      digitalWrite(audioPin2, LOW);
      Serial.println("Audio pin 2 set to LOW because button 2 is HIGH.");
      lastButtonPressTime = millis();
    } else {
      digitalWrite(audioPin2, HIGH);
//      Serial.println("Audio pin 2 set to HIGH because button 2 is LOW.");
    }

    // Handle the state of audioPin3 based on button3Pin
    if (button3State == HIGH) {
      digitalWrite(audioPin3, LOW);
      Serial.println("Audio pin 3 set to LOW because button 3 is HIGH.");
      lastButtonPressTime = millis();
    } else {
      digitalWrite(audioPin3, HIGH);
  //    Serial.println("Audio pin 3 set to HIGH because button 3 is LOW.");
    }

    // Handle the state of audioPin4 based on button4Pin
    if (button4State == HIGH) {
      digitalWrite(audioPin4, LOW);
      Serial.println("Audio pin 4 set to LOW because button 4 is HIGH.");
      lastButtonPressTime = millis();
    } else {
      digitalWrite(audioPin4, HIGH);
  //    Serial.println("Audio pin 4 set to HIGH because button 4 is LOW.");

    }

    // Check if 60 seconds have passed since the last button press
    if (millis() - lastButtonPressTime > 60000) {
      digitalWrite(signalPin, LOW); // Set signal pin LOW after 60 seconds of inactivity
    } else {
      digitalWrite(signalPin, HIGH); // Otherwise keep it HIGH
    }
  }


  // Small delay to prevent flooding the serial output
  delay(50);
}
```

TROUBLESHOOTING: GENERAL

Your EchoPoint is not working correctly? Start with this checklist!

1. Check that the switch is set to audio not charge.
2. Check that the fuse is not blown.
3. Check that the MicroSD card is seated correctly in the player (eject, remove, reinstert until there is a click).
4. Confirm using your computer that the MicroSD card has the correct files and they are in the correct playable order.
5. Check that the battery is charged.
6. If issues persist then, continue to troubleshoot specifics.

TROUBLESHOOTING: SPECIFIC

No sounds

Problem: There is no power to the device or any of its components.

- Solution:
    - Check that the toggle switch is set to "Audio" and not "Charge."
    - Check that the microSD card slot is seated correctly in the amplifier (oriented with writing side down).
    - Check that the files on the microSD card are formatted correctly (00001.mp3 format).
    - Check that the files on the microSD card are not corrupted and can play on your computer.
    - Charge or replace battery with a fully charged battery.
    - Check that all connections are secure.
        - Reference the electrical diagrams to make sure the circuit is connected correctly.
    - Check that the fuse is not blown.
        - Note: if the fuse is blown, it is important to investigate further to determine what caused the fuse to blow. Do not use the EchoPoint until this is determined.

Audio too quiet:

- Problem: The audio is too quiet.
- Solution:
    - Change the audio amplifier dial located on the amplifier board within the unit.
    - The selected audio file might be digitally too quiet if it is a new recording being added, use software like Audacity to increase the digital volume of the file.

DISCLAIMER

The EchoPoint device described in this manual is provided "as is" with limited warranty at the discretion of the creator (Andrew D. Marques). This discretionary limited warranty includes but is not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. In addition, the user may return the device at any time. Replacement of the device can be discussed at the discretion of the creator. By using this device, the user agrees that Andrew D. Marques shall not be held liable for any damage, harm, or loss arising from the use or misuse of the device. The user assumes full responsibility for ensuring the proper and safe use of the EchoPoint device.

ACKNOWLEDGEMENTS

The success of this project was made possible by the assistance of numerous friends, whose contributions I would like to acknowledge. I extend a special thank you to Carter Merenstein, Corey Nickels, and Prakrati Nickels, for their invaluable troubleshooting help with conceptualization and programming.

Additionally, I owe a debt of gratitude to my wife Caitlyn Mierau-Marques, whose unwavering support was critical to the project's success. Caitlyn was endlessly patient and supportive, providing encouragement throughout the countless nights and many hours spent grinding away using the dremel or on the computer programming.

Their insights, feedback, and support were vital to the completion of this work.