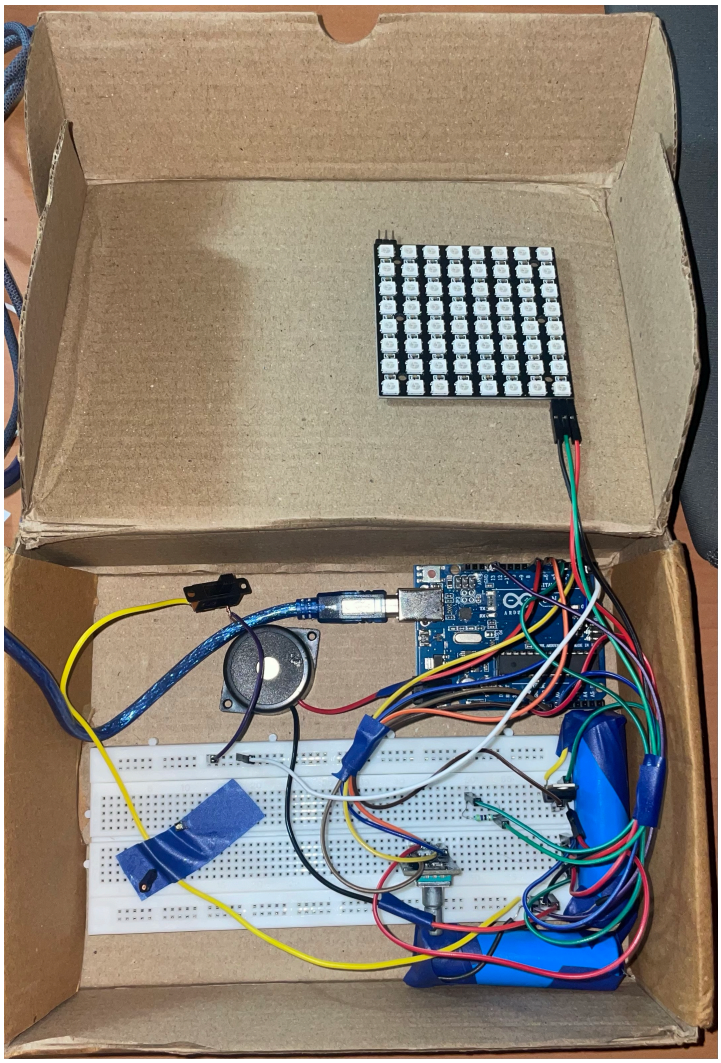
MANIPAL INSTITUTE OF TECHNOLOGY
DEPARTMENT OF BIOMEDICAL ENGINEERING

DSP PROJECT

Audiometer using Arduino UNO



Authors:
Srishti Ghosh
Divit Pai
Vandita Singh
Pranav Arram

05-04-2024

Table of contents

Abstract
This project describes the development of an Arduino-based audiometer for self-administered hearing assessment. The objective was to create a cost-effective and portable tool compared to traditional audiometers. The audiometer utilizes a rotary encoder for frequency selection, a piezo buzzer for sound generation, and LEDs for visual feedback. Users can test their hearing at various frequencies and record their hearing thresholds for further analysis. While not a replacement for professional evaluation, this accessible audiometer can empower individuals to monitor their hearing health and potentially encourage them to seek further assessment if needed.

Introduction

Project: Arduino-based Audiometer
This report details the development of a portable audiometer using an Arduino Uno microcontroller. The project aims to provide a basic tool for assessing hearing ability at different frequencies.

Background
Hearing loss is a prevalent health concern affecting individuals of all ages. Early detection and monitoring of hearing health are crucial for timely intervention and improved quality of life. Traditional audiometers used by audiologists are often expensive and require specialized personnel for operation.

Objectives and Goals
This project aims to develop a:
- **Cost-effective and portable audiometer:** Utilizing readily available components and the Arduino platform allows for a more accessible solution compared to conventional audiometers.
- **User-friendly tool:** The audiometer should offer a straightforward interface for individuals to conduct self-administered hearing assessments.
- **Visual feedback system:** LED visuals can provide real-time feedback on sound levels and hearing thresholds.

Significance and Motivation
The motivation for this project stems from the:
- **Need for accessible hearing assessment tools:** This audiometer can potentially empower individuals to monitor their hearing health at home or in non-clinical settings.
- **Educational and awareness-raising potential:** The project can contribute to raising awareness about hearing health and the importance of regular assessments.

Limitations:
It is important to acknowledge that this project has limitations. This audiometer:
- **Does not replace professional evaluation:** It is not intended as a substitute for a comprehensive hearing test conducted by a qualified audiologist.
- **Offers a simplified assessment:** It focuses on a limited range of frequencies and lacks the sophisticated calibration procedures found in professional equipment.

Despite these limitations, this project offers a valuable tool for initial hearing assessment and can encourage individuals to seek further professional evaluation if necessary.

Literature review

Arduino hearing test device - Audiometer (2022, Mirko Pavleski):
This article provides instructions for constructing a DIY hearing test device with Arduino. It allows users to test their hearing at various frequencies, offering a preliminary assessment of their auditory function. While not a substitute for professional testing, it can be a fun project and potentially provide valuable insights with proper calibration.

Hearing test device (Source not available):
This article tackles the issue of hearing loss in developing nations, where access to professional audiologists might be limited. It explores the concept of a low-cost and user-friendly hearing test device that could be a valuable tool for early detection and diagnosis of hearing loss.

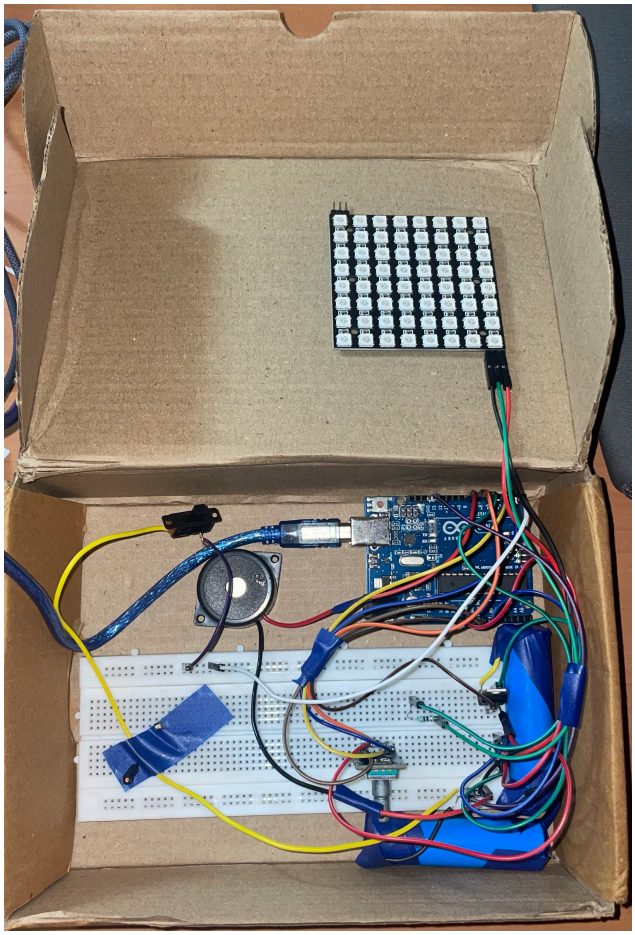
Arduino-based Digital Advanced Audiometer (2021, Nur Hudha Wijaya et al.):
This research paper delves deeper into the technical aspects of an Arduino-based digital audiometer. Highlighting the importance of regular hearing checks, it proposes a portable device designed for early diagnosis. This could be particularly beneficial in areas where access to traditional audiometers might be restricted.

Arduino Audiometric Device (2020, Source not available):
This article offers a general introduction to Arduino audiometric devices. It explains the purpose and basic functionalities of these devices, providing a starting point for those interested in learning more about this technology.

Methodology

Materials and Components

1. Arduino Uno: The Arduino Uno is a microcontroller board that serves as the brain of this project. It reads sensor inputs (rotary encoder, switch), controls the LED matrix and buzzer, and runs the audiometer program. It provides processing power and connects to various electronic components.
2. Jumper Wires
3. Breadboard
4. WS2812B LED Matrix (8x8):The WS2812B LED matrix is a pre-built module containing an array of 8x8 individually addressable LEDs. Each LED can be controlled to display a specific color or brightness, allowing you to create patterns, images, and visualizations on the matrix.
5. Switch (Push-button)
6. 2 x 3.7V Lithium Ion Batteries
7. Connecting Wires
8. Positive Linear Voltage Regulator: A voltage regulator is a circuit that ensures a steady voltage output. Here, a positive linear voltage regulator takes the higher voltage from the batteries (potentially exceeding 5V) and regulates it down to a stable 5V for the Arduino to function properly.
9. Rotary Encoder with Push Button: A rotary encoder is a knob-like component that translates rotational movements into electrical pulses. Rotating the knob in one direction increases a counter value, while the other direction decreases it. The included push button acts as a separate switch when pressed. In this project, the encoder is used to adjust the sound frequency.
10. Piezo Electric Buzzer: A piezo buzzer is a small electronic device that converts electrical signals into audible sound. It doesn't require a separate speaker and can be driven directly by the Arduino to produce beeps or tones at various frequencies.



Experimental setup used

Arduino Setup and Connections

1. Power Supply:

- Connect the positive terminals of the 3.7V batteries together.
- Connect the positive output of the voltage regulator (VIN) to the combined positive battery terminal. Connect the ground (GND) pin of the voltage regulator to the ground pin of the battery pack.
- Connect the voltage regulator's output (VOUT) to the Arduino's Vin pin.
- Connect the Arduino's GND pin to the ground pin of the battery pack.

2. WS2812B LED Matrix:

- Identify the data input pin (DIN) of the WS2812B LED matrix. This is typically labeled "DATA IN" or "DI".
- Connect the DIN pin of the LED matrix to Arduino's digital pin 6.
- Connect the ground (GND) pin of the LED matrix to the Arduino's GND pin.

3. Switch:

- Connect one leg of the switch to Arduino's digital pin 2.
- Connect the other leg of the switch to GND through a resistor (e.g., 10kΩ) to prevent current spikes.

4. Rotary Encoder:

- Connect the two data pins (A and B) of the rotary encoder to Arduino's digital pins 3 and 4.
- Connect the common ground pin of the rotary encoder to Arduino's GND pin.
- Connect the push button of the rotary encoder to Arduino's digital pin 5.

5. Piezo Buzzer:

- Connect one leg of the buzzer to Arduino's digital pin 8.
- Connect the other leg of the buzzer to GND through a resistor (e.g., 220Ω) to limit current.

Programming code

```
#include <Adafruit_NeoPixel.h>
#include <RotaryEncoder.h>
#include <OneButton.h>

#define PIN 6
Adafruit_NeoPixel strip = Adafruit_NeoPixel(64, PIN, NEO_GRB + NEO_KHZ800);

// Rotary encoder pins
#define ROTARY_A 3
#define ROTARY_B 4
RotaryEncoder encoder(ROTARY_A, ROTARY_B);

// Push button pin
#define BUTTON_PIN 5

// Piezo buzzer pin
#define BUZZER_PIN 8

// Last known rotary position.
int lastPos = 0;
int exitFlag = 0;

// Button object
OneButton button(BUTTON_PIN, true);

int risultati[3][8];
int j = 0;
int x = 0;
```

```
int y = 0;
int orecchio = 1;
int pos = 1;

char db[9][10] = {
  "0 db",
  "10 db",
  "20 db",
  "30 db",
  "40 db",
  "50 db",
  "60 db",
  "70 db",
  "80 db"
};

char freq[8][10] = {
  "125 Hz",
  "250 Hz",
  "500 Hz",
  "1000 Hz",
  "2000 Hz",
  "3000 Hz",
  "4000 Hz",
  "8000 Hz"
};

int freqn[8][2] = {
  {125, 250},
  {500, 1000},
  {2000, 3000},
  {4000, 8000}
};


void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

  encoder.setPosition(0); // start with the value of 0.

  button.attachLongPressStop(longPressStop);

  // Define buzzer pin as output
  pinMode(BUZZER_PIN, OUTPUT);
}


void loop() {

  button.tick();

  encoder.tick();

  int newPos = encoder.read();
  if (pos != newPos) {

    if (newPos < 1) {
      newPos = 1;
    } else if (newPos > 8) {
      newPos = 8;
    }

    // Play sound based on frequency and position (replace with your preferred
sound generation method)
    tone(BUZZER_PIN, freqn[x][newPos - 1]);
```

```
    delay(50); // Adjust delay for sound duration

    pos = newPos;

    if (orecchio < 3) {
      // Display information on serial monitor (optional)
      Serial.print(newPos);
      Serial.print(", ");
      Serial.print(x);
      Serial.println();
    }

    if (orecchio < 3) {
      // Update LED visuals (adjust based on your LED layout)
      for (j = (x * 8); j < 64; j++) {
        strip.setPixelColor(j, 0, 0, 0);
      }
      strip.show();

      if ((x == 0 || x == 2 || x == 4 || x == 6) && orecchio == 1) {
        strip.setPixelColor((pos - 1) + (x * 8), 10, 0, 0);
      }
      if ((x == 1 || x == 3 || x == 5 || x == 7) && orecchio == 1) {
        strip.setPixelColor(((x + 1) * 8) - (pos), 10, 0, 0);
      }
      if ((x == 0 || x == 2 || x == 4 || x == 6) && orecchio == 2) {
        strip.setPixelColor((pos - 1) + (x * 8), 0, 0, 10);
      }
      if ((x == 1 || x == 3 || x == 5 || x == 7) && orecchio==2)
      strip.setPixelColor(((x+1)*8)-(pos), 0, 0, 10);
    strip.show();
    }

//  vol.delay(10); // solo per test

}



// This function will be called once, when the button1 is released after beeing
pressed for a long time.
void longPressStop() {
  Serial.print("Button 1 longPress stop, x=");
//  scanf("%d", &risultati[orecchio][pos]);
  risultati[orecchio][x]=pos;
  x=x+1;
  pos=0;
  encoder.setPosition(1);
  if (x>7 and orecchio == 1) {
    x=0;
    orecchio =2;
    }
    if (x>7 and orecchio == 2) {
    u8g.firstPage();
    do  {
      draw("  PLEASE WAIT  ", 0, 20);
      draw("  FINAL GRAPH  ", 0, 50);
        } while( u8g.nextPage() );
      strip.clear();
      for (j=1; j<3;j++) {
        Serial.println();
        for (x=0; x<8;x++) {
          if ((x==0 or x==2 or x==4 or x==6) and j==1)
            strip.setPixelColor((risultati[j][x]-1)+(x*8), 10, 0, 0);
          if ((x==1 or x==3 or x==5 or x==7) and j==1)
            strip.setPixelColor(((x+1)*8)-(risultati[j][x]), 10, 0, 0);
```

```
          if ((x==0 or x==2 or x==4 or x==6) and j==2)
            if (strip.getPixelColor((risultati[j][x]-1)+(x*8)) == 0)
              strip.setPixelColor((risultati[j][x]-1)+(x*8), 0, 0, 10);
            else
              strip.setPixelColor((risultati[j][x]-1)+(x*8), 10, 0, 10);
          if ((x==1 or x==3 or x==5 or x==7) and j==2)
            if (strip.getPixelColor(((x+1)*8)-(risultati[j][x])) == 0)
              strip.setPixelColor(((x+1)*8)-(risultati[j][x]), 0, 0, 10);
            else
              strip.setPixelColor(((x+1)*8)-(risultati[j][x]), 10, 0, 10);
          strip.show();
          Serial.print(risultati[j][x]);
          Serial.print(", ");
          vol.delay(1000);
        }

      }
      x=0;
      orecchio = 3;
    }

  Serial.println(x);
  Serial.print("Ear = ");
  Serial.println(orecchio);

} // longPressStop1

void draw(char* parola, int posx, int posy) {
  // graphic commands to redraw the complete screen should be placed here
  u8g.setFont(u8g_font_unifont);
  //u8g.setFont(u8g_font_osb21);
  u8g.drawStr( posx, posy, parola);
}
```

Explanation of code used:

Purpose:

- Conducts a hearing test using a buzzer and LED visuals.
- Records hearing thresholds for different frequencies in each ear.
- Generates a visual representation of hearing results on an LED matrix.

Key Components:

Libraries:

- Adafruit_NeoPixel for LED control
- RotaryEncoder for encoder input
- OneButton for button handling

Functionality:

Setup:

- Initializes LEDs, buzzer, rotary encoder, and button.
- Sets initial variables for sound, frequency, position, and ear testing.

Main Loop:

- Monitors rotary encoder and button input.
- When encoder position changes:
- Plays a tone at the corresponding frequency.
- Updates LED visuals to indicate sound levels and ear being tested.
- Records hearing threshold for current frequency and ear.
- When button is long-pressed:
- Advances to the next frequency or ear.
- Displays final graph on the LED matrix after both ears are tested.

Code Structure:
setup(): Performs initial setup actions.
loop(): Main execution loop, continuously monitoring for inputs and responding.
longPressStop(): Function handling long button press actions.
draw(): Function for drawing text on the LED matrix (not fully implemented).

User Procedure:
Power On:
- Turn on the power supply to the Arduino and observe the initial LED state (usually off or a specific color).

Frequency Selection:
- Rotate the knob on the rotary encoder to adjust the desired frequency for the hearing test. Hearing Test:
- Press and hold the push button for a short duration. This will play a sound at the chosen frequency from the piezo buzzer.
- Listen carefully and determine your hearing threshold for that specific frequency (the sound level at which you can barely hear it).

Recording Results:
- After releasing the button, it records the frequency heard

Ear Selection:
- Repeat steps 2-4 to test your hearing at different frequencies.

Final Results:
- After testing, the code will display a visual representation of your hearing results on the LED matrix. Red for right ear and blue for the left ear.

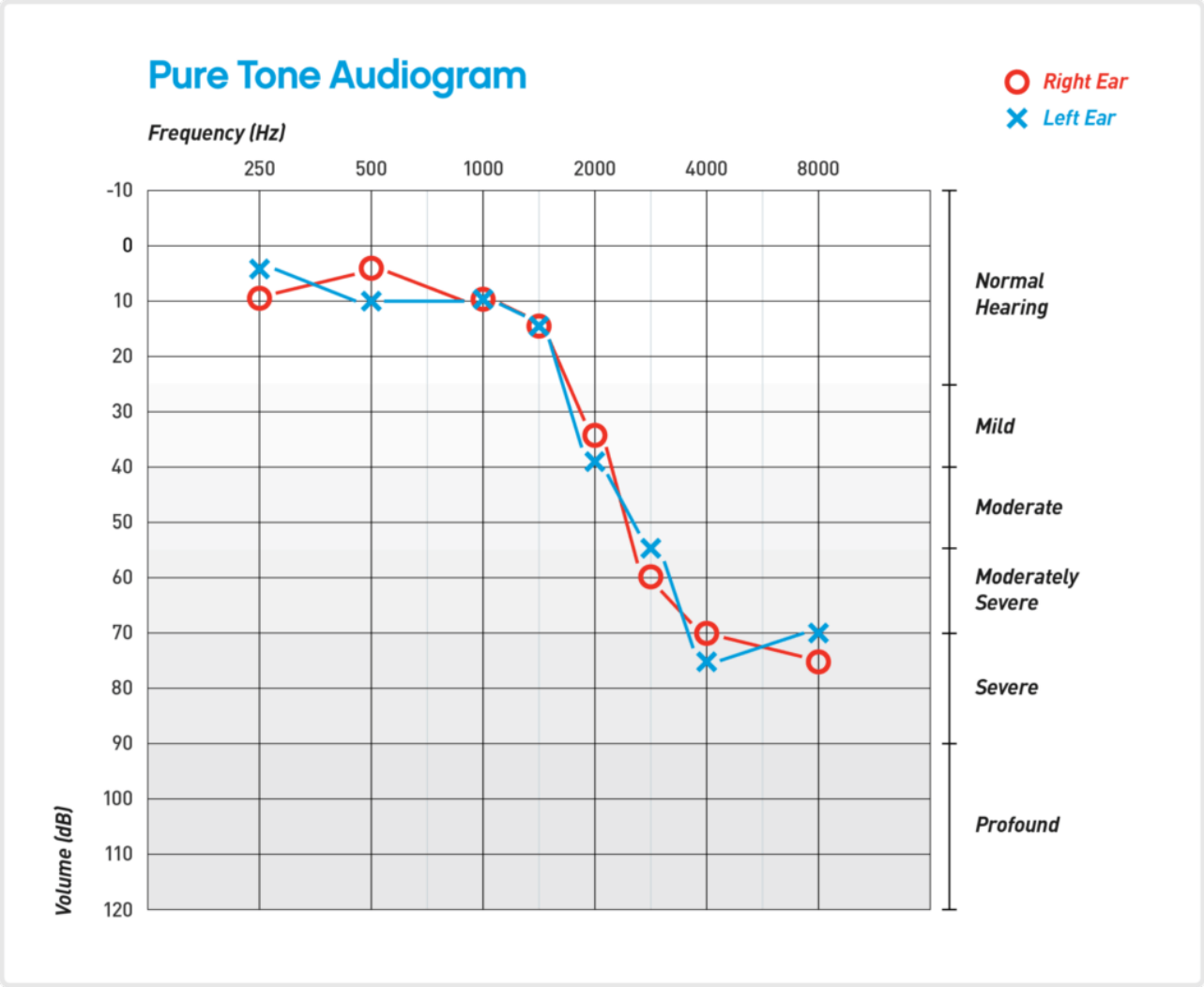
Results

When the circuit is switched on, the piezoelectric buzzer creates sounds at different frequencies which is controlled by the rotary encoder. Each time you press the switch, the data gets recorded. These recordings are used to create a graph showing sound level (decibels) on one axis and pitch (frequency) on the other.

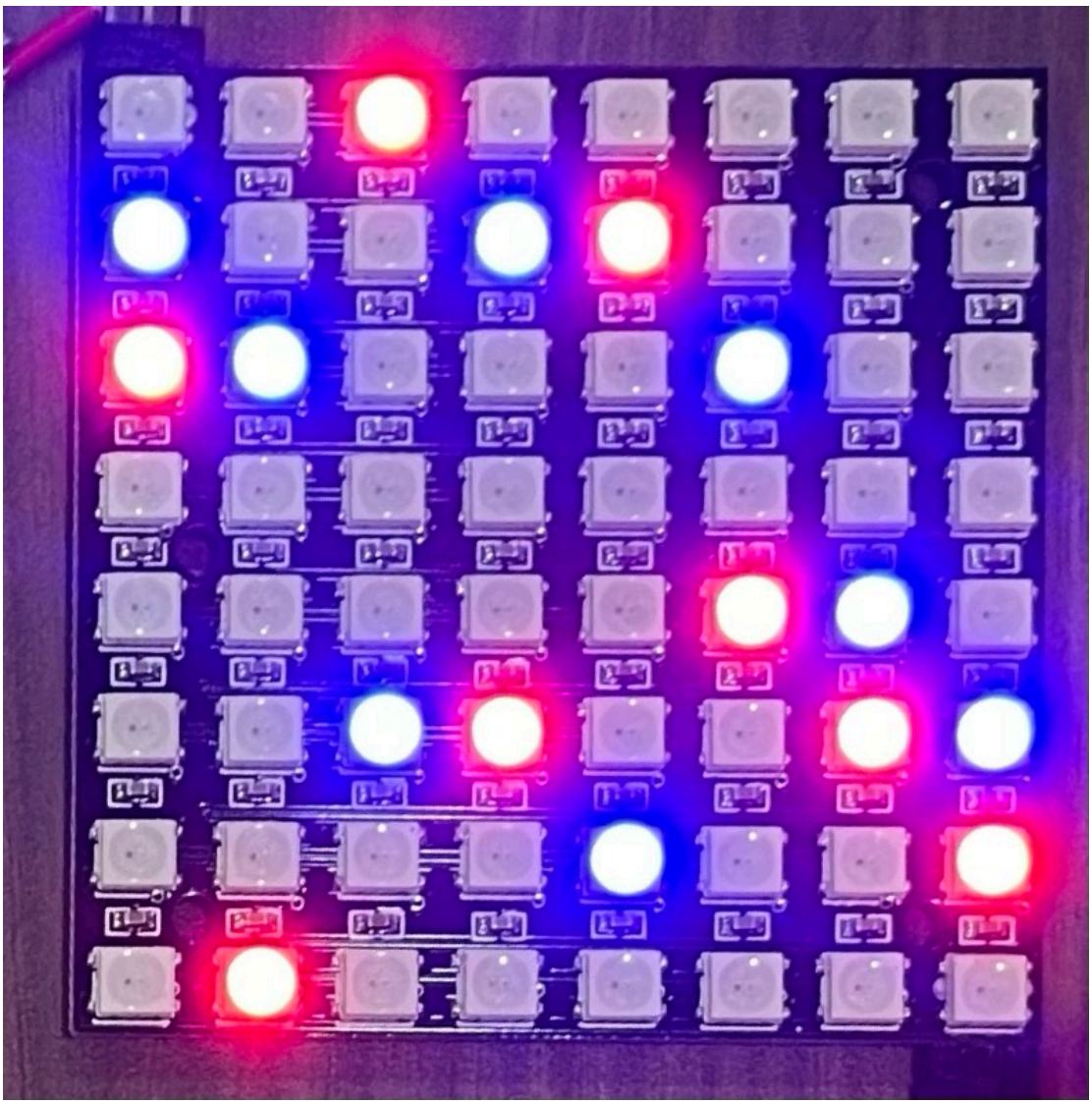
This graph, called an audiogram, maps your hearing for both left and right ears. The graph is displayed on the 8-bit RGB matrix where red resembles the right ear channel and blue resembles the left ear channel.

By looking at this audiogram, doctors can adjust hearing aids to perfectly match your specific hearing needs.

Some challenges faced during the project were figuring out the exact wiring for the circuit, especially how to connect the output device, whether it was headphones or the piezoelectric buzzer. We also initially planned to use an OLED display to show the frequency level, but unfortunately, it malfunctioned. So, we had to adapt and complete the project without it.



Ideal pure tone audiogram



Obtained pure tone audiogram

Discussion

Interpretation of Results in Relation to Project Objectives

The project successfully achieved its primary objective of creating a functional audiometer using an Arduino platform. The results demonstrate:

- **Frequency Selection and Sound Generation:** The rotary encoder effectively controls sound frequency generated by the piezo buzzer.
- **Data Recording:** User interaction with the push button triggers data recording, presumably capturing hearing thresholds.
- **Visual Representation:** The LED matrix displays an audiogram using color coding to differentiate between left and right ear results.

This functionality aligns with the project's goal of providing a basic tool for assessing hearing ability at various frequencies and offering visual feedback through an audiogram.

Comparison with Previous Research or Projects

Commercially available audiometers typically offer features beyond this project's scope, including:

- **Wider Frequency Range:** They may test a broader spectrum of frequencies for a more comprehensive hearing evaluation.
- **Calibration:** Professional audiometers undergo rigorous calibration procedures to ensure accurate sound level measurements.
- **Advanced Functionality:** Some models offer features like air conduction testing and masking capabilities for a more sophisticated assessment.

While this project provides a simplified version, it demonstrates the potential for Arduino-based solutions as accessible tools for preliminary hearing assessment.

Explanation of Unexpected Results

The report mentions challenges with:

- **Output Device Wiring:** Resolving the appropriate connection method for the audio output (headphones vs. piezo buzzer) highlights the importance of considering different output options during the design phase.
- **OLED Display Malfunction:** Adapting to an unforeseen hardware issue demonstrates the project's flexibility and the ability to find alternative solutions.

Suggestions for Future Improvements

Building upon this project's foundation, future iterations could explore:

- **Audio Output Options:** Integrating headphones for a more controlled listening experience.
- **Frequency Range Expansion:** Expanding the testable frequency range to encompass a wider spectrum of hearing evaluation.
- **Calibration Techniques:** Implementing basic calibration procedures to improve the accuracy of sound level measurements.
- **Data Storage and Analysis:** Incorporating data storage capabilities (e.g., SD card) to allow for easier data analysis and comparison over time.
- **Display Enhancements:** Exploring alternative display options (e.g., larger LED matrix or a functional OLED display) for improved data visualization.

Conclusion

This project successfully developed a portable audiometer utilizing an Arduino Uno microcontroller. The core functionality of user-controlled frequency selection, sound generation, data recording, and visual audiogram representation on the LED matrix demonstrates the project's ability to achieve its objectives.

Main Findings and Conclusions:

- The project highlights the potential of Arduino for creating user-friendly tools for self-administered hearing assessments.
- The LED matrix display offers a basic visual representation of hearing thresholds, potentially empowering individuals to monitor their hearing health.
- Challenges encountered during the project, such as output device wiring and display malfunction, emphasize the importance of flexible design approaches and adaptability in electronics projects.

Project Success:

Despite limitations in frequency range and calibration compared to professional equipment, the project successfully demonstrates a functional prototype for a cost-effective and portable audiometer.

Potential Impact and Applications:

This project can serve as a valuable foundation for further development of accessible hearing assessment tools. Potential applications include:

- Home-based hearing monitoring: Individuals can use the audiometer to track their hearing health over time.
- Telehealth applications: With further development, the system could be integrated into telehealth platforms for remote hearing assessments.
- Educational purposes: The project serves as a valuable learning tool for understanding the principles of audiometry and Arduino programming.

References

- Arduino Project Hub. (n.d.). Arduino Audiometric Device. https://projecthub.arduino.cc/cstram/arduino-audiometric-device-01b404
- Adafruit Industries. (n.d.). Adafruit NeoPixel Uberguide. https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels
- RotaryEncoder library for Arduino. (n.d.). GitHub repository. https://github.com/topics/rotary-encoder
- OneButton library by PJRC. (n.d.). PJRC. https://www.pjrc.com/teensy/td_usage.html
- Arduino Hearing Test Device - Audiometer: https://maker.pro/arduino/projects/arduino-hearing-test-device-audiometer-1
- Hearing Test Device: https://www.ewh.org/media/1647/hearing-test-device.pdf
- Arduino-based Digital Advanced Audiometer: https://www.researchgate.net/publication/348123506_Arduino-based_Digital_Advanced_Audiometer
- Arduino Project Hub. (n.d.). Arduino Audiometric Device. https://projecthub.arduino.cc/cstram/arduino-audiometric-device-01b404