INTRODUCTION

In this project, a device which senses the audio signal in the environment and visualizes its frequency spectrum on an LCD screen was developed. It involves processing the sensor data with Fast Fourier Transform etc. and visualize the data in a variety of ways. The designed using scenario for this device is to display the frequency spectrum of music played out in the environment. With this device, it allows people to perceive music in both acoustical and visible ways. The device is intended to be used by any individual who loves music or be used at music venues such as concerts and clubs to present a more attractive way of appreciating music. The Github link of this project is included in the Appendix.

LITERATURE REVIEW

There are some similar existing projects which display the frequency spectrum of an audio signal fed from headphone output or line output of the music system. For example, in John's project, the input signal is not measured by a sensor, which means it does not include the process of dealing with noise or interference in the physical circuit (John3904 & Instructables, 2020). Also, in the project in this report, the device is designed to sense and react to an open environment, which gives it broader applications. The device can be put in any indoor or outdoor environment to visualize the music spectrum, or even used in noise detection because it can show the frequency band and magnitude of noises.

METHODS

Components List

- Sound sensor module V1.6 (Zuo, no date): This is a microphone module which integrates an L358 amplifier and an electret microphone. It outputs an analog signal that is amplified to a moderate value to be processed by Arduino. Only one data pin is required to connect it to Arduino.

Figure 1. Microphone module

- LCD module MAX7219 (MAX7219, 2021): A MAX7219 module includes an 8×8 dot matrix display and a MAX7219 LED display driver. Four chained modules which has a 32×8 dot matrix display as shown in Figure 2 was used in this project in order to display a wider band of frequency. The MAX7219 communicates via the SPI interface, so 3 data pins are required to connect to Arduino.

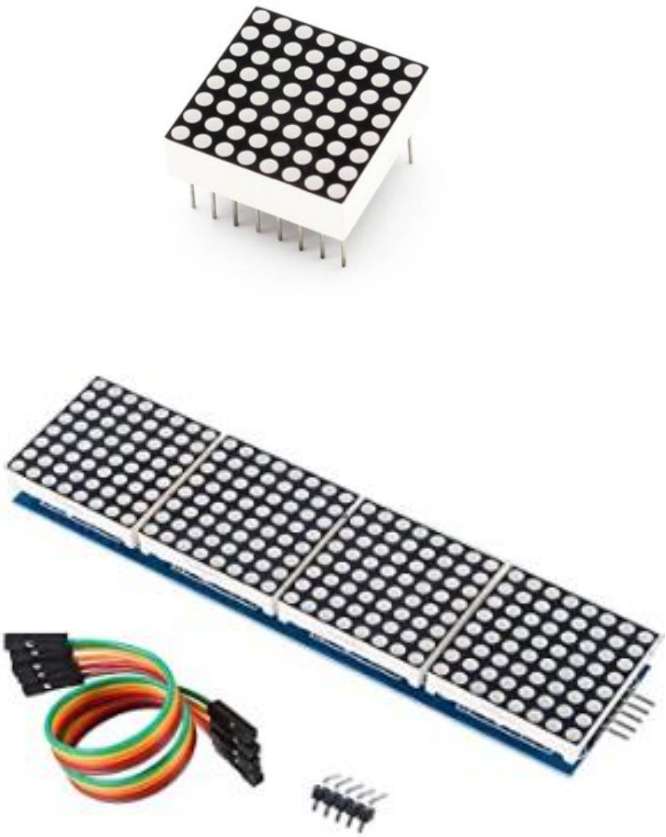

Figure 2. LCD module with a 32×8 dot matrix display

- Press button: This is used to change the display mode of LCD when pressed by the user.
- Arduino Uno: It is used as the microcontroller board which is responsible for receiving data from the sound sensor, carrying out data processing and sending data to the LCD display.

Circuit Connection

The whole circuit was connected in a layout shown in Figure 3. Three digital pins on Arduino were assigned to the LCD module for the clock pin, input data pin and chip select pin. One digital pin was assigned for the button which controls the display mode. One analog pin was connected to the microphone module. Two USB cables were used to power the circuit, instead of powering the LCD module with Arduino, it was powered separately with the Vcc and Vdd of another USB cable, as shown in Figure 4. This was to eliminate the interference between LCD and microphone signal transmitted via same power line, which will be explained more detailly in Result and Discussion part.

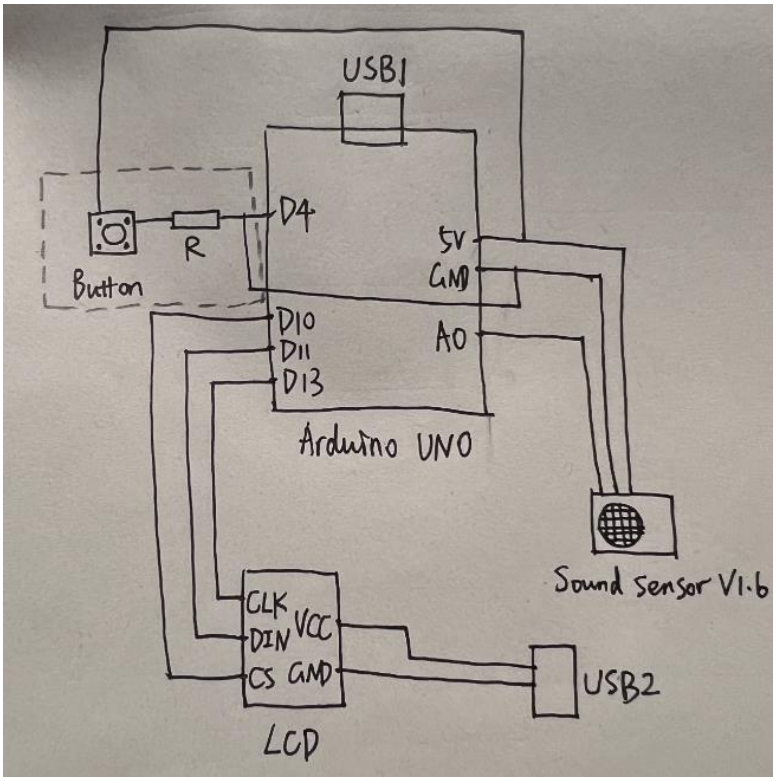

Figure 3. Circuit connection layout

Figure 4. USB cable used to power LCD module

The button circuit which is in the dotted square in Figure 3 was soldered on a small PCB to reduce the wires and make the system robust. A 10k Ohm resistor was used here to prevent short circuit between HIGH and GND in Arduino.

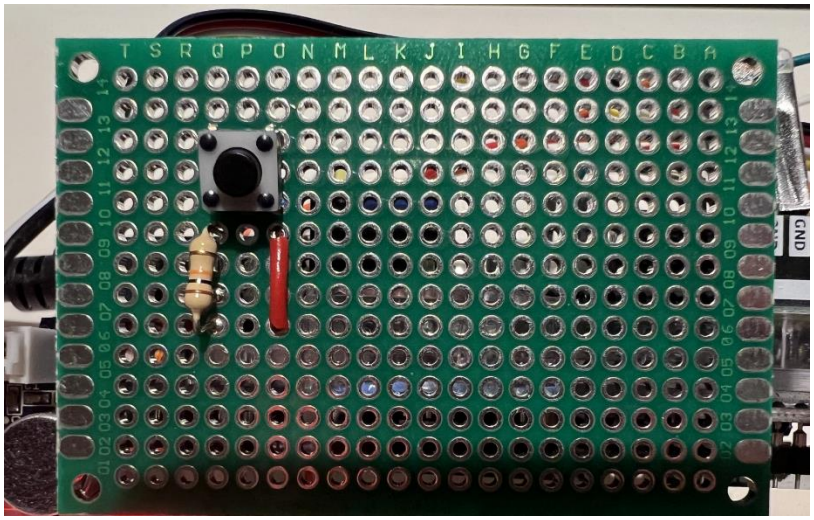

Figure 5. Soldered button circuit on a PCB

Enclosure

The enclosure was made with cardboard. As shown in Figure 6, all the components are enclosed inside and leaves only two USB cables out for powering. On the top leaves two holes for the button and microphone. The position of each component is fixed with

glue (Figure 7) and the overall device is robust.

Figure 6. The enclosure appearance

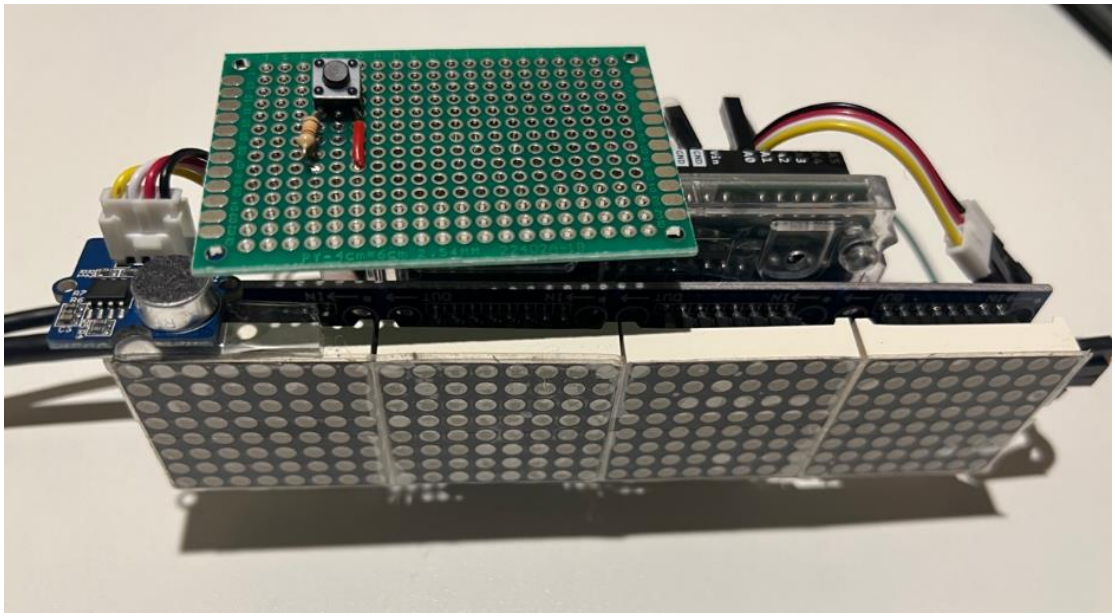

Figure 7. Device inside the enclosure

Data Flow

Analog-to-Digital Converter

The analog data output by microphone needs to be converted to digital so that Fourier Transform can be carried out later. After receiving the data with an analog pin on Arduino, through the code ‘analogRead(pinAdc);’, the analog data was sampled with

the on-board ADC of Arduino. The default sampling frequency of Arduino ADC is about 10 kHz (Arduino Reference, 2019). In this project the sampling frequency was set to be around 38.5 kHz by configuring the ADC prescaler to 32, using the code in Figure 8 (Gammon, 2015). The reason why this frequency was used is that the frequency of audio signal human can hear is from 20 Hz to 20 kHz (Barber et al., 2020). According to the Nyquist theorem, a signal can be reproduced if it is sampled by a frequency that is at least twice of its highest frequency (Wright, 2022). Therefore, to display the full frequency band of a music signal, the sampling frequency should be set around 40 kHz. In this way, the frequency that can be displayed on LCD ranges from around 0 Hz to 20 kHz. There are 32 dots horizontally, so each dot represents a frequency band of 625 Hz.

```
ADCSRA &= ~(bit (ADPS0) | bit (ADPS1) | bit (ADPS2)); // clear prescaler bits
ADCSRA |= bit (ADPS0) | bit (ADPS2); // set prescaler to be 32, gives a 38kHz sampling frequency
```

Figure 8. Code to set Arduino ADC prescaler to 32

Averaging

The sampled signal was then averaged every 32 to decrease the ambient noise that is not the music.

```
for(int i=0; i<SAMPLES; i++){
long sum = 0;
 for(int j=0; j<32; j++)
 {
    sum += analogRead(pinAdc);
 }

 sum >>= 5;//averaging
```

Figure 9. Average the sampled data

Fast Fourier Transform

The library ‘arduinoFFT.h’ was installed in Arduino to perform the FFT calculation. This step converts the signal from time domain to frequency domain. After the signal was sampled and averaged, it was further compressed by dividing it by 6. This was to adjust the magnitude of frequency displayed on the LCD or it will be too big that all

lights will be turned on. Then the data was assigned to an array called ‘vReal[ ]’ to carry out FFT with a sample number specified. The number of samples in each cycle should be twice the number of frequency bins, so it was set to be the number of horizontal lights on LCD, 32. After conversion, the real and imagine values returned by function represent the frequency and magnitude of signal. The amplitude of each frequency was remapped between 0 to 8, which was the number of LEDs in a column.

Display on LCD

‘SPI.h’ and ‘MD_MAX72xx.h’ library was installed for this step. The remapped amplitude was sent to LCD through 4-pin SPI interface. 6 display modes to display the frequency spectrum were designed, as shown in Figure 10. ‘MD_MAX72xx.h’ provides function to turn on or off any LED in a column through a binary number. For example, ‘int MY_ARRAY[]={0, 128, 192, 224, 240, 248, 252, 254, 255};’ this array can be translated into ‘int MY_ARRAY[]={0b00000000, 0b10000000, 0b1100000, 0b11100000, 0b11110000, etc...};’. The ‘1’ represents turning on LED and ‘0’ represents turning off LED.

```
int MODE_1[]={0, 128, 192, 224, 240, 248, 252, 254, 255}; // standard pattern
int MODE_2[]={0, 1, 3, 7, 15, 31, 63, 127, 255}; // upside down
int MODE_3[]={0, 128, 192, 224, 208, 200, 196, 194, 193}; // peak and bottom 2 points
int MODE_4[]={0, 1, 192, 7, 240, 31, 252, 127, 255};//every other column is upside down
int MODE_5[]={0, 128, 64, 32, 16, 8, 4, 2, 1}; // only peak pattern
int MODE_6[]={0, 128, 128, 160, 160, 168, 168, 170, 170};// every other light is on
```

Figure 10. Code for 6 display modes

RESULTS AND DISCUSSIONS

The results of display in 6 different modes switched by buttons are shown in Figure 11 to 16. The switch condition for the display modes considered the debouncing of button, so the button was effective. These display modes are flexible to customize according to personal preference, which gives a variety of ways to visualize the music. It was observed that the display changes with music rhythm nearly synchronously, so the delay caused by processing data is acceptable.

Figure 11. Mode 1 - Standard mode

Figure 12. Mode 2 - Upside down

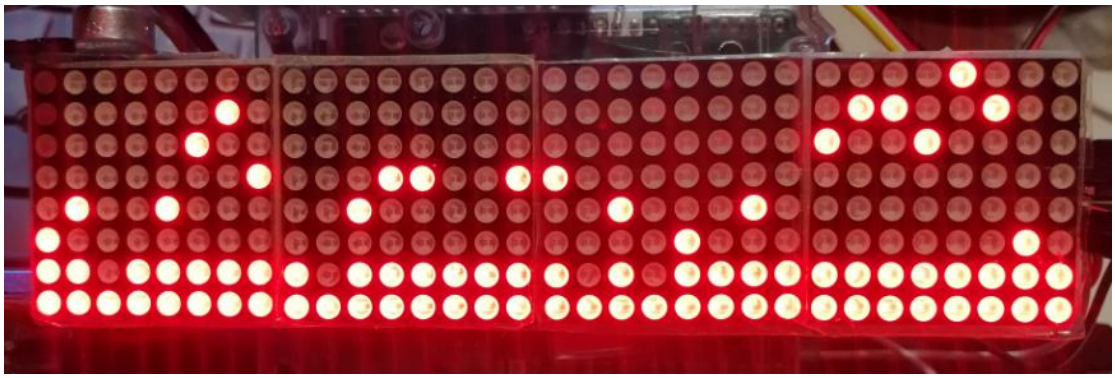

Figure 13. Mode 3 - Peak and bottom two points

Figure 14. Mode 4 - Every other column is upside down

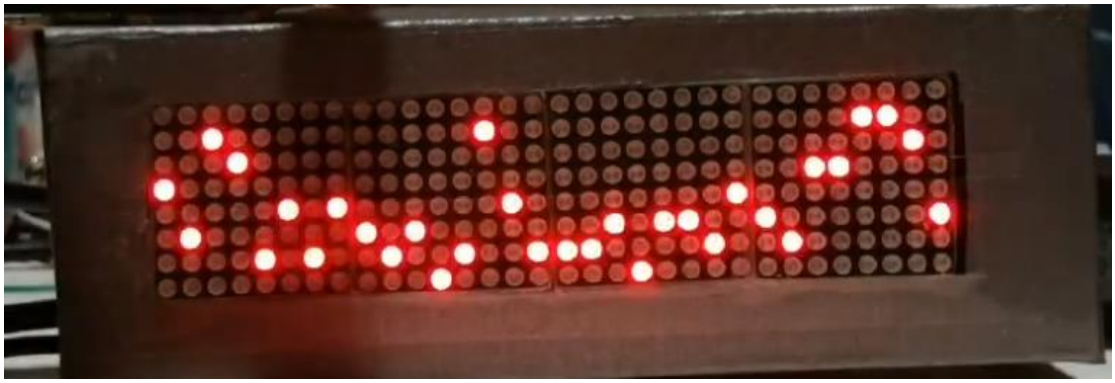

Figure 15. Mode 5 - Only peak

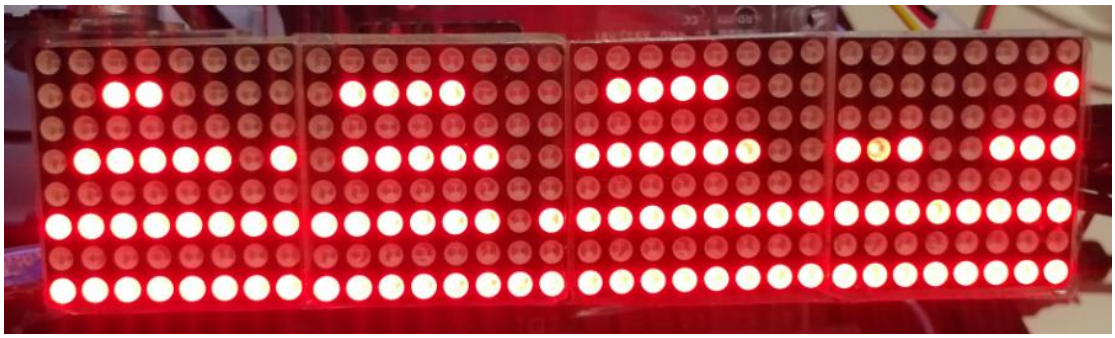

Figure 16. Mode 6 - Every other light is on

There were some difficulties encountered. When no music is displayed, LCD displays the spectrum of ambient noise. As shown in Figure 17, this case has already been improved by averaging the data because without averaging higher frequency band would also light up. To improve the user experience, it is better to remove the low frequency ambient noise. In the development process a high pass filter consisted of resistor and capacitor was added physically in the circuit to experiment. However, it caused an open circuit and blocked the signal because a physical filter should be used for an AC signal. The way to solve this could possibly be building a digital high-pass filter in the data processing part.

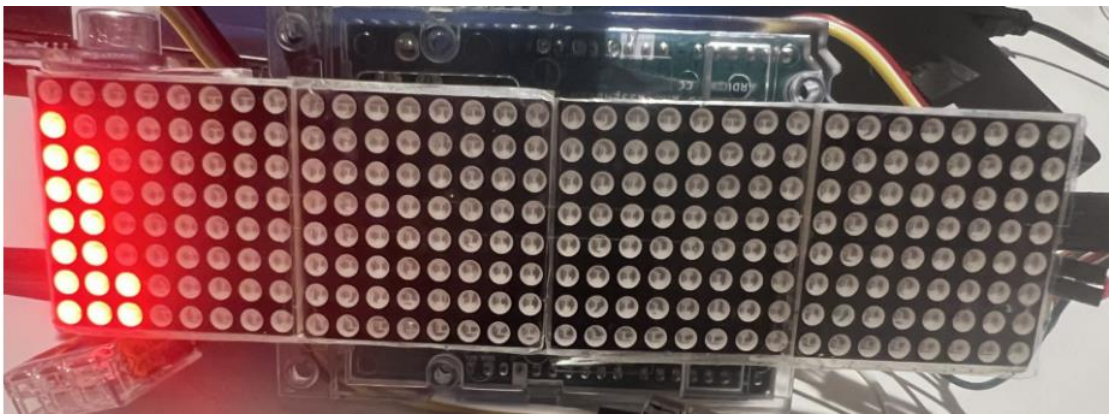

Figure 17. Spectrum of ambient noise

Another major problem encountered was the internal interference between components in the circuit. When the LCD module and microphone module were both connected to the Arduino power line, an unneglectable interference was observed in the output signal of microphone, which was much stronger and drowned the music signal. An oscilloscope was used to test the location of noise and it was found possibly the noise was transmitted through shared power line. In the instructions to use the LCD module, it is suggested to use an external power supply instead of the Arduino's 5V supply, because the display draws much current (Last Minute Engineers, 2022). Therefore, this interference is possibly caused by too much current was drawn by LCD and the microphone couldn't work normally. The easiest way to solve this is to use another USB cable to separately power LCD, which successfully eliminated the interference and was adapted in this project, although it is not an optimal design that two USB cables are required to power. The next step for this project could be finding another way to eliminate the interference.

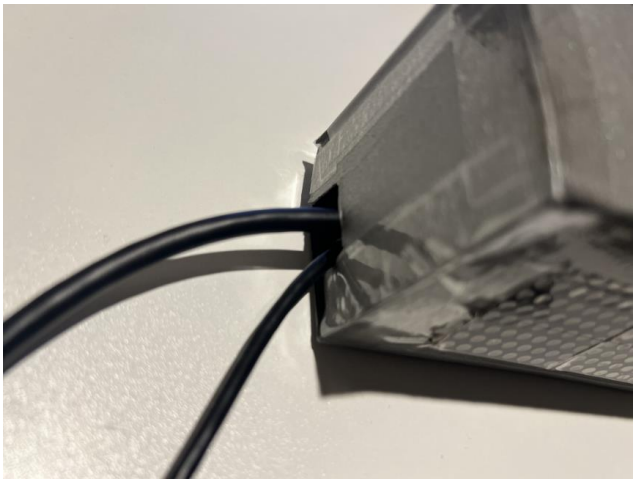

Figure 18. Two USB power cables

REFERENCES

Analogread() (2019) *analogRead() - Arduino Reference*. Available at: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (Accessed: January 6, 2023).

Barber, A.L. *et al.* (2020) "A comparison of hearing and auditory functioning between dogs and humans," *Comparative Cognition & Behavior Reviews*, 15, pp. 45–94. Available at: https://doi.org/10.3819/ccbr.2020.150007.

Gammon, N. (2015) *Gammon forum : Electronics : Microprocessors : ADC conversion on the Arduino (analogread), Written by Nick Gammon - 5K*. Available at: http://gammon.com.au/adc (Accessed: January 11, 2023).

John3904 and Instructables (2020) *Band Audio Spectrum Visualizer Analyzer, Instructables*. Instructables. Available at: https://www.instructables.com/Band-Audio-Spectrum-Visualizer-Analyzer/ (Accessed: January 6, 2023).

Last Minute Engineers (2022) *In-depth: Interfacing MAX7219 led dot matrix display with Arduino, Last Minute Engineers*. Last Minute Engineers. Available at: https://lastminuteengineers.com/max7219-dot-matrix-arduino-tutorial/ (Accessed: January 11, 2023).

MAX7219 (2021) *Serially Interfaced, 8-Digit, LED Display Drivers | Analog Devices*. Available at: https://www.analog.com/en/products/max7219.html#product-reference (Accessed: January 6, 2023).

Wright, G. (2022) *What is the Nyquist theorem?, WhatIs.com*. TechTarget. Available at: https://www.techtarget.com/whatis/definition/Nyquist-Theorem (Accessed: January 6, 2023).

Zuo, B. (no date) *Grove - Sound Sensor, seeedstudio*. Available at: https://wiki.seeedstudio.com/Grove-Sound_Sensor/ (Accessed: January 6, 2023).

Appendix

Video demonstration: https://github.com/ucfnnbx/music/blob/main/video.mp4

Arduino code: https://github.com/ucfnnbx/music/blob/main/Arduino%20code