ENHANCING MUSICAL ACCESSIBILITY: A NOVEL DEVICE FOR INDIVIDUALS WITH HEARING IMPAIRMENTS USING VIBRATIONS AND LED LIGHTS SYNCHRONIZED WITH MUSIC TEMPO

Carie Chen[1], Carol Chen[2], Justin Lou[3]

[1]Arcadia high school, 180 Campus Drive, Arcadia, 91007
[2]Arcadia high school, 180 Campus Drive, Arcadia, 91007
[3]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Our project is centered on the concept of making music accessible to all individuals, including those with hearing impairments. Music can often play a large role in many lives, able to serve as a source of emotional comfort for many. As the numbers of hearing-impaired individuals rise significantly each year, it is important to enhance accessibility within their lives, including ways to experience the joy of music. By creating a device with vibrations and LED lights that sync with the beat/tempo of each song, this new way of music is easily accessible to all, especially to those with hearing impairments [7]. Leveraging our coding skills, we designed a program that could analyze a specific audio file. The analyzed information is then sent to our raspberry pi, connected to a haptic controller and LED lights through GPIO pins, this allows an easily accessible and functioning process for all users [8].

KEYWORDS

Musical Accessibility, Hearing Impairments, Haptic Feedback, Synchronized Music Device

1. INTRODUCTION

As an auditory form of media, music inadvertently marginalizes individuals with auditory issues and leaves them with limited avenues of engagement. The reliance on auditory senses establishes formidable barriers that prevents those with hearing impairments from fully enjoying the intricate nuances and elements of music. Statistical data highlights the prevalence of hearing loss, revealing that one out of every eight individuals aged 12 and above in the United States experiences some degree of hearing impairment [1]. This figure emphasizes the widespread nature of hearing issues within the population, its impact further underscored by a recent study indicating a 63% increase in the number of people transitioning from mild to profound hearing loss since 2021 [2].

The implications of this issue extend beyond the personal realm into broader social dimensions. An article from Pfizer explains how a study found an intricate connection existing between music and the limbic system, a part of the brain that controls memories and emotional responses [3].

David C. Wyld et al. (Eds): SE, SAIM, SIPM, CoNeCo, ICITE, ACSIT, CMIT, FCST, SNLP – 2024
pp. 93-102, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.140809

Such responses when hearing music through the ear have been found to trigger feelings of pleasure and well-being, highlighting the profound influence that music can have on mood and emotional states. However, for hearing-impaired individuals, this exclusion from musical engagement results in a missed opportunity to positively influence their emotional well-being. The deprivation of musical experiences not only denies individuals with hearing impairments the inherent advantages associated with such experiences, but also infringes upon their right to access a fundamental aspect of human expression. Through recognizing the integral role that music can play in many aspects, it becomes imperative to address this disparity and provide access to those without conventional access.

We propose the development of MeloSign, an application that enables users to experience music by engaging their senses beyond only auditory perception. Through the use of AI learning and pattern recognition, users can select a song or sound from a database, which is then analyzed and translated into the physical world through movement and changing lights [9]. This translation introduces a tactile and visual dimension that represents various elements of the song.

The process begins with the upload of an audio file to be interpreted by code. The code identifies a constant component of the song and organizes it into lists. These lists are then analyzed with respect to time and are sent to the client, which processes and initiates a response through a physical medium.

The app is designed to be comprehensible and user-friendly, allowing access through various devices and screen sizes. Users can benefit from an organized platform that facilitates navigation through our simple selection layout, enabling easy access to their desired choices. In addition to the previously mentioned methods of engagement, the app includes features that enhance accessibility to music for those who are hearing-impaired, providing a dynamic responsiveness and more inclusive method to enjoy music.

METHOD A: Method A is about a project conducted using beat synchronization to create vibrotactile experiences with music for both hearing and hearing-impaired individuals. The main shortcomings are its applicability to social events where vibrations can be generated. Our project offers a solution that may be used by hearing and hearing-impaired people alone as well as at social gatherings, which helps address the limitations of the earlier studies.

METHOD B: In method B, a group of researchers and scientists created a study using vibrotactile stimulation to transmit emotional reactions and brain activity in an audio-visual soundtrack. The researchers tried to experiment and understand how individuals with hearing impairment react and respond to these vibrotactile stimuli [15]. The shortcomings include certain song choices as only orchestra music fragments were played. When experimenting they also only experimented this on females, which could have a different reaction with men.

METHOD C: Method C goes over the reaction of hearing-impaired people to visual stimuli in the form of the lights, examines how they are more receptive than those with their hearing, and also briefly talks about the effectiveness of other devices. This method lacks information on responses to other stimuli where our project tackles the physical aspect.

In experiment 1, we tested the accuracy of MeloSign's translation of audio files. The project design entailed coordinating the haptic controller, LED lights, and the song's beats for synchronization. The data showed 100% accuracy for the tested songs: Retro, Waves, and Guitar. The analysis indicates that there was perfect synchronization but in more complex audio files, results may differ.

In our second experiment, we tested different audio files' compatible with our program. We set this up by finding different audio file formats such as MP3, MP4, and Wav Files, then plugging it into our code. The MP3 and MP4 audio files were not compatible, as it didn't sync, and the LEDlights also didn't connect [10]. However, the Wav audio files were compatible working every time with an accuracy rate of 100%. This is probably because of how these audio files are formatted making Wav files more accessible for coding.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Navigation of Coding Obstacles due to Inexperience

The success and completion of our project depend greatly on our programming knowledge. To ensure the project fulfills our envisioned outcome, we plan to invest time and resources to learn relevant coding skills so we can effectively express our ideas in a programming language. For example, we would be unsure how to implement third-party programs into our code that we deem necessary to the functionality of our project, such as a database or a physical component. Our projected plan is to take a coding class and perform research in our own time to gain a better grasp on the tools we need to tackle issues we expect to encounter in our design.

2.2. Ensuring Project Compatibility with Various Hardware

A concern that is held is the ability for physical devices to function with our program. Our original plan was a bracelet that could stimulate physical motion to reflect elements of the song being played, but we are concerned with the choice of device being too limiting on how we might be able to implement our design. Our new vision is to establish a system where a variety of devices can be compatible with the program by testing with a simple haptic motor setup. By initially working with a simple method, we can find ways to make our code work with a variety of different electronics.

2.3. Creation of Simple Software to Allow Ease of Access and Universal Device Compatibility

The program should be capable of access for users on a variety of devices. Therefore, the underlying technology and program used must be flexible enough to work across various platforms without any issues. The application should be simple to use and intuitive so that all users may easily understand and make use of its features. For users to easily navigate the software and access its functions, it is important to have a simple and clear user interface. Also, we plan to test our application with various devices and present our application at various stages to other people to observe human responses.

3. SOLUTION

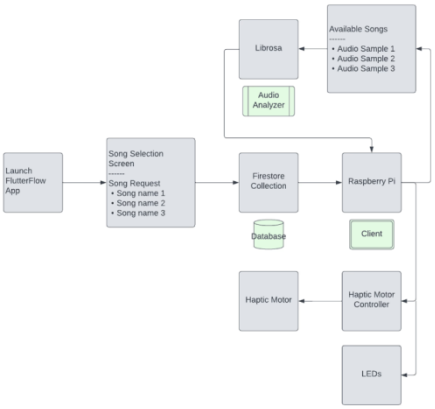

Figure 1. Flowchart diagram of program

The 3 major components of the program are the interface, the hardware, and the backend. In the interface, we used Flutterflow, a simple app development software, to create a straightforward and user-friendly interface. The Futterflow app allows users to select their song of choice as displayed on the screen.

This marks the start of the program, and the selection information is transmitted over to the hardware components which consists of all the physical devices: the Raspberry Pi, a haptic controller, a haptic motor, and LED lights. This section generates responses depending on the received signals from both the code and the song selection. The Raspberry Pi is a small rectangular single-board computer that uses the code written to relay instructions to the haptic controller on how to vibrate the motor. The Raspberry Pi stores all .wav files of the project and has many other features that allow it to work with other components and serve as the transmitter for all necessary information in our project.

The backend of the program contains instructions on the initialization and connecting of all the pieces of the program together. The backend includes tools such as the Firestore database, where all audio files are stored, and Librosa, a Python package that can perform music and audio analysis. Librosa can perform many tasks such as audio extraction and time series analysis. When selecting an audio file, Librosa can efficiently detect all the required data and pull a timeframe of all percussive beats in a song. This flow is important as it ensures a simple and engaging user experience as well as a synchronized response.

The backend of the program contains instructions on the initialization and connecting of all the pieces of the program together. The backend includes tools such as the Firestore, a document database that receives song selection information from Flutterflow and sends it to the hardware, and Librosa, a Python package that can perform music and audio analysis. Librosa can perform many tasks such as audio extraction and time series analysis. When selecting an audio file, Librosa can efficiently detect all the required data and pull a timeframe of all percussive beats in a song. This flow is important as it ensures a simple and engaging user experience as well as a synchronized response.

One of the components of the application is the interface. This intractable part of the program is where users can select a song of their choice from an available list connected to a database. This component is constructed using Flutteflow, an application development software. The song

selections on the app interact directly with a database that serves as a pathway to the rest of the components.

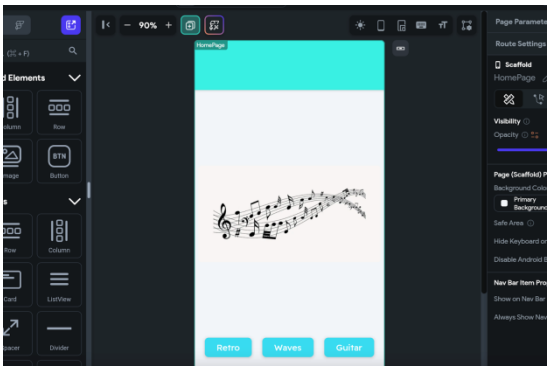

Figure 2. FlutterFlow App Editor

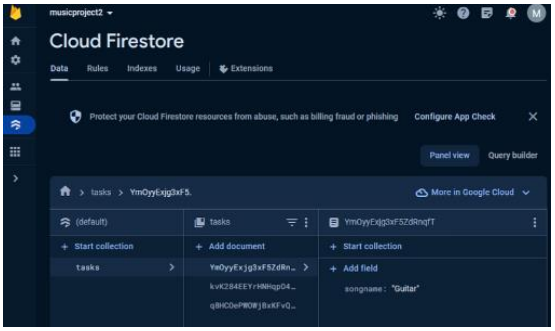

Figure 3. Firestore Database

```
while True:
    # get information from the server firestore
    songcollection = db.collection("tasks").stream()
    # parse the data in the collection
    songrequests = []
    for doc in songcollection:
        print(f"{doc.id} => {doc.to_dict()}")
        addtoqueue = doc.to_dict()
        songrequests.extend(addtoqueue.values())

    # Select only one song the list by priority if there are multiple requests
    if "Guitar" in songrequests:
        songname = "Guitar"
    elif "Waves" in songrequests:
        songname = "Waves"
    elif "Retro" in songrequests:
        songname = "Retro"
    else:
        songname = ""
```

Figure 4. Initialization Code Sample

When starting, the connection first is established with the database where all song requests are sent and stored. After the initial setup, the code runs when a song selection is made. The song name is sent from Flutterflow to Firestore as a string and saved as a document in a collection named “tasks” [13]. The code then runs indefinitely in a loop, continually scanning the Firestore’s “tasks” collection for any new song requests. During each loop, the code actively retrieves and processes newly added documents containing song requests or data. The script analyzes these documents during this loop, adding song requests to a list along with identifying and prioritizing certain songs, like “Guitar,” “Waves,” or “Retro.” The code can effectively recognize and handle incoming song requests that are stored in the Firestore database without unnecessary processes.

Another component of the project includes the physical devices used. The hardware initiates a response as a result of the code. Librosa will find any percussive beats of a chosen song, and the

Raspberry Pi will send signals to the LED lights, haptic controller, and haptic motor [14]. This allows the LED lights to start turning on and off and for the haptic motor to start vibrating.

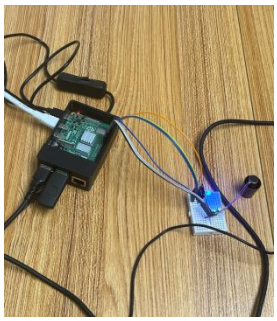

Figure 5. Raspberry Pi, haptic motor, and LED lights

```
# set up gpio for led lights
leds = LEDBoard (17, 27, 22, pwm=False)
leds.off()
leds.on()

# set up the motor controller
i2c = busio.I2C(board.SCL, board.SDA)
drv = adafruit_drv2605.DRV2605(i2c)
# tell the contoller what effect I want to use
drv.sequence[0] = adafruit_drv2605.Effect(1)
time.sleep(1)
```

Figure 6. Code for Hardware Setup

```
# function to control the motor and the leds
def vibrate():
    drv.play()
    leds.on()
    time.sleep(0.2)
    leds.off()
    drv.stop()
```

Figure 7. Function for Hardware Activity

This screenshot contains our code that allows the LED lights and haptic motor to function and run during the initialization process. This code is sent to the Raspberry Pi, a small board computer that is our main controller and is responsible for both executing the code and relaying information on how to move to the other parts. The LED lights are connected to the Raspberry Pi through GPIO pins 17, 27, and 22. These pins serve as a communication channel that allows the LED lights to be turned on and off. The haptic motor and controller are also connected to the Raspberry Pi using the I2C pins. The code initiates a signal directed to the haptic controller, that triggers specific reactions and feedback. The haptic motor then responds generating vibrations from the programmed code.

A significant part of the functionality of the application is contributed through our code which analyzes the audio files and songs. Librosa converts the wav files into simpler audio files which are inputted into the code. The audio wav files/ the songs are stored in the pi, allowing the code to process the file. The database allows for the storage of multiple audio files that can be accessed quickly and managed efficiently.

```
#the audiofile is imported to librosa
#sr is how many data points per second (hz)
y,sr=librosa.load(album[songname], sr=44100)
#bpm the steady beat of a song
#the group of frames a beat happens
tempo, beat_frames= librosa.beat.beat_track(y=y , sr=sr)
#the list of times a beat happens
beat_times = librosa.frames_to_time(beat_frames, sr=sr)
print(beat_times)
print(len(beat_times))

# Start counting, and calculate time passed
start_time = time.time() #1000000
latest_time = time.time() #1000000.081 right, almost no difference
time_elapsed = latest_time - start_time #0.001

# start at the first beat of the list
i = 0
# keep going until the end of the song (the last beat)
while i < len(beat_times):
    if time_elapsed >= beat_times[i]:
        #make the pi do stuff
        vibrate()

        i=i+1 #move onto the next beat
        if i == len(beat_times):
            break

    latest_time = time.time() #1000000.001 right, almost no difference
    time_elapsed = latest_time - start_time #0.001
```

Figure 8. Code Sample of Librosa and Functional Loop

After receiving data from the application, the database stores and organizes the data. After the song is chosen by the Pi, it is loaded onto Librosa and converts the .wav file into a simpler array of audio amplitudes [12]. The sample rate is then fixed to 44100 Hz, which is usually the standard sample rate for audio files, and the frames where beats occur in a song are found and stored into the variable called beat_frames. With that sample rate, every second contains 44100 samples, if a beat happens at the two second mark, that beat frame is 88200. Those frames are then converted to seconds for simplicity, and the script uses another while loop to vibrate the motors according to those timestamps, now a variable called beat_times. The loop repeats , activating the LEDs and motor controller until a placeholder variable equals the number of beats listed in beat_times. Finally, the Firebase collection resets and deletes all documents and the Pi waits for another song request from Firestore.

4. EXPERIMENT

4.1. Experiment 1

Another blind spot we wish to examine is the overall accuracy of the program's translation of audio files. We are concerned that the output of the program might misrepresent the choice of audio file.

Our objective is to evaluate the coordination between the haptic controller, LED lights, and the actual song by comparing their beats. The experiment entails initiating the haptic controller and LED lights simultaneously while playing the audio file audibly. We will carefully observe and listen to ascertain if the beats consistently synchronize throughout the entire duration of the song.

Song	Accuracy
Retro	100%
Waves	100%
Guitar	100%

Figure 9. Figure of experiment 1

The analysis of the data reveals that all three songs achieved a synchronization rate of 100%. This implies that the beats and lights were perfectly aligned for each song. While this outcome is a goal of this project, results may vary in different scenarios with more complicated beats and

rhythms. The limited number of songs included in our test could also be a contributing factor to the consistent results.

4.2. Experiment 2

A possible blind spot in our program that we want to test out is compatibility with different types of audio files. This is important that this works well because it would ensure that our code can cater to wide ranges of user preferences.

The purpose of this experiment is to determine how well different audio file formats such as MP4 and MP3 are able to identify tempo, BPM, and spectrograms when compared to WAV files which is our control variable. We will go over several audio file formats, beginning with MP4s. The process is audibly playing the audio and determining whether the LED lights and haptic controller sync with the song. The purpose of this is to see the accuracy of the system in identifying variables that in file types are not WAV.

Audio File	Accuracy
MP3	0%
MP4	0%
WAV File	100%

Figure 10. Figure of experiment 2

After analyzing the experiment data, it is evident that the program's accuracy varies greatly depending on the audio file format. Specifically, both MP3 and MP4 files show an accuracy rate of 0%, while WAV files, which serve as the control variable, show an accuracy rate of 100%. The lack of accuracy in MP3 and MP4 files suggests a weakness in the program's compatibility with these specific audio formats. This is particularly significant as it poses a challenge in accurately identifying tempo, BPM, and spectrograms for these formats. The findings show the need to address compatibility issues with different audio file types in order to ensure the program's effectiveness for a wide range of user preferences.

5. RELATED WORK

In 2017, Frontiers published research using beat synchronization to vibrotactile music in hearing and early deaf people [4]. This solution works as it relies on the sound waves of music to generate vibrations through the sensory receptors in our bodies. Using a vibrating speaker that produced a quiet noise, participants wore eyeglasses, ear muffs, and a mask, and stood barefoot for this experiment while music at 110,115, and 120 MPH was playing. Studying this group of 14 hearing and 7 deaf people, both groups were able to synchronize a bouncing motion in the music to a vibrotactile beat. However, there are a few limitations in this research. This can only work in some sort of social event setting or a party where vibrations can be generated. Our project improves on this as it is more portable and easily accessible.

In 2010, computer science researchers and scientists came together to create a study and project similar to ours by using a vibrotactile stimulation to transmit emotional reactions in an audio-visual soundtrack for 2 minutes [5]. This experiment was conducted among a group of female participants from the University of Madrid, and out of all the volunteers, 9 were normal-hearing, 7 had hearing loss, and 4 were deaf. Two orchestral music fragments were played at a fast, then

slow tempo, two vibrotactile stimuli were programmed on a glove to exert a soft tactile vibration. From the data collected, all the participants were able to feel the vibrations and generate brain activity from it.

Engineering and computer science researchers from different universities across the US came together to discuss and analyze past studies of deaf people feeling music rhythms and beats using devices [6]. While the most typical devices involved an audio sensor, 2 microcontrollers, lights, and vibrotactile outputs, these turned out to be the most effective and convenient. The researchers have proved that deaf people have a better reaction to visual flashes than individuals with normal hearing, so lights on the devices are an extra benefit. Most process the music through part of the brain and activate activity.

6. CONCLUSIONS

Our application has several limitations that should be considered. One drawback is the reliance on WAV files [11]. This dependency can create difficulties when incorporating other audio formats, such as M4A, which may result in a loss of accuracy. The restricted compatibility with various audio file types may pose challenges for users with different preferences. Additionally, users' choices are limited because they cannot add new music or audio files unless they are creators or individuals with access to the source code.

Addressing these compatibility and variety issues would significantly enhance the efficiency and accessibility of the platform. The limitations with user input and support only for WAV files demand a more in-depth version that can be easily used. Since only people who have access to the coding platform can add new songs, the application's capabilities for user input and audio file management are restricted for average users. Furthermore, using WAV files inhibits compatibility with many specific audio formats, which might impact user preferences. To offer a more thorough, user-friendly, and inclusive platform, it is crucial to address these issues.

REFERENCES

[1] Kuenburg, Alexa, Paul Fellinger, and Johannes Fellinger. "Health care access among deaf people." Journal of deaf studies and deaf education 21.1 (2016): 1-10.

[2] McIlroy, Guy, and Claudine Storbeck. "Development of deaf identity: An ethnographic study." Journal of deaf studies and deaf education 16.4 (2011): 494-511.

[3] Hodges, Donald A., and Robin W. Wilkins. "How and why does music move us? Answers from psychology and neuroscience." Music Educators Journal 101.4 (2015): 41-47.

[4] Tranchant, Pauline, et al. "Feeling the beat: Bouncing synchronization to vibrotactile music in hearing and early deaf people." Frontiers in neuroscience 11 (2017): 281459.

[5] Lucía, María J., et al. "Vibrotactile captioning of musical effects in audio-visual media as an alternative for deaf and hard of hearing People: an EEG study." IEEE Access 8 (2020): 190873-190881.

[6] Florian, Horațiu, et al. "Deaf people feeling music rhythm by using a sensing and actuating device." Sensors and Actuators A: Physical 267 (2017): 431-442.

[7] Zhuang, Yuan, et al. "A survey of positioning systems using visible LED lights." IEEE Communications Surveys & Tutorials 20.3 (2018): 1963-1988.

[8] Clark, Liz, and Liz Clark. "Programming with the GPIO Pins." Practical Tinker Board: Getting Started and Building Projects with the ASUS Single-Board Computer (2019): 81-129.

[9] Haussler, David. "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework." Artificial intelligence 36.2 (1988): 177-221.

[10] Timothy, Adeboje Olawale, Adetunmbi Adebayo, and Gabriel Arome Junior. "Embedding text in audio steganography system using advanced encryption standard, text compression and spread spectrum techniques in Mp3 and Mp4 file formats." International Journal of Computer Applications 177.41 (2020): 46-51.

[11] Koenig, Bruce E., and Douglas S. Lacey. "Forensic authenticity analyses of the metadata in re-encoded WAV files." Audio Engineering Society Conference: 54th International Conference: Audio Forensics. Audio Engineering Society, 2014.
[12] Suman, Shubham, et al. "Visualization of audio files using librosa." Proceedings of 2nd International Conference on Mathematical Modeling and Computational Science: ICMMCS 2021. Singapore: Springer Nature Singapore, 2022.
[13] Kesavan, Ram, et al. "Firestore: The nosql serverless database for the application developer." 2023 IEEE 39th International Conference on Data Engineering (ICDE). IEEE, 2023.
[14] Zhao, Cheah Wai, Jayanand Jegatheesan, and Son Chee Loon. "Exploring iot application using raspberry pi." International Journal of Computer Networks and Applications 2.1 (2015): 27-34.
[15] Verrillo, Ronald T. "Psychophysics of vibrotactile stimulation." The Journal of the Acoustical Society of America 77.1 (1985): 225-232.