SirenAlert - Ensuring Pedestrian Safety: Report
CS7470: Mobile and Ubiquitous Computing

Allend Osman Botani
abotani3@gatech.edu

Bryan Edward Johnson
bjohnson423@gatech.edu

Karan Bhasin
karanbhasin@gatech.edu

Rayvan Watson Jr
rwatson73@gatech.edu

Tim Hung
timhung@gatech.edu

ABSTRACT

Ubiquitous computing has advanced the lives of individuals in recent years. Technology has been integrated into the lifestyles of people to a point where devices integrate seamlessly into activities. Thus, the growing demand for new features in smart city development has flourished. This project utilized the principles of ubicomp to develop a mobile application that detects sirens nearby and provides notifications to users. The motivation behind the development consists of integrating smart city features to provide contextual awareness and safety to locals or help those with auditory impairment. This idea emerges from the significant number of accidents that have historically happened due to users being unaware of their surroundings. The application utilizes advanced mathematical concepts such as the Fast Fourier Transform (FFT) to filter out frequencies from the microphone on the user's smartphone. Development via Android Studio allows for live updates to users establishing situational awareness via notifications consisting of audio, visual, and haptic feedback. The integration of mathematical concepts, ubicomp principles, and software development led to an application with a high efficiency and accuracy rate in detecting sirens, displaying advancements in today's technology.

INTRODUCTION

Due to a busy world, individuals must be able to react fast in situations. We determine that distractions and physical disabilities can cause problems regarding situational awareness which ultimately puts people's lives in harm's way as they are unable to hear/detect local sirens that can consist of emergency vehicles or even public service announcements. Critical thought led us to the conclusion that the best way to interact with users and their environment is to use something that is always nearby, a smartphone. The development of a mobile application using Android Studio was the approach for this idea as it allows for interoperability and ease of use. The application utilized mathematical concepts such as the Fast Fourier Transform to filter out frequencies and then use the peak frequencies as detection thresholds. Due to the advancement in the application detection methods, the microphone can pick up local noises and filter them for siren detection. A significant emphasis on detection methods existed due to the need to distinguish background noises from actual sirens.

The origination of the mobile application idea stemmed from the idea of a user being preoccupied, listening to music, and not aware of their surroundings. Unaware users can lead to catastrophic events given certain circumstances. Due to this mixing of urbanization and technological advancements, our team developed a mobile application to detect sirens nearby to make users aware of their surroundings and utilize the principles of ubiquitous computing and fundamentals of developing smart city infrastructure to support vulnerable populations. The mobile application is relevant to today's fast-paced society by contributing to situational awareness and safety by tackling the challenges faced with densely populated environments all while utilizing ubiquitous computing fundamentals.

BACKGROUND

The application is developed in Android Studio, an integrated development environment for Android applications that have supporting features for smartphones. Android Studio allowed for debugging, code implementation, development of the user interface, and the establishment of haptic and audio feedback.

The powerhouse component of development revolves around the Fast Fourier Transform, filtering input from the user's microphone. FFT methods are dominant in the signal-processing community [1]. They utilize time domain inputs and turn them into a frequency domain output to determine patterns and signals.

$$S(f) = \int_{-\infty}^{\infty} s(t)\, e^{-j2\pi f t}\, dt \quad [1]$$

Equation 1 displays the fast Fourier transform formula where S(f) is the FFT output and s(t) is the decomposing wave. In the modern day, FFT can be seen in audio signal processing, sonar and vibration analysis, and even biomedical signal analysis. Our mobile application revolves around the audio signal processing realm of FFT to filter input generated from the user's microphone.

Prior literature and experiments have utilized mathematical concepts for auditory filtering similar to what we have done. Historical research related to our design consisted of the utilization of convolutional; neural networks (CNNs) for signal detection [2]. This method utilizes feature extraction

to analyze distinctive patterns in the wavelengths of audio, machine learning algorithms and classification methods for filtering [2]. A study by Cantarini et. Al utilizes CNNs, activation functions, and Doppler effect simulations to determine sirens [2]. Different implementations could have been used since accuracy rates were skewed due to overfitting and the system relied heavily on the training data used, indicating a more realistic dataset being needed [2]. The implementation also showed low signal-to-noise rates, indicating a decrease in the quality of filtering [2]. Another study by Fatimah, et. Al utilized the Fourier decomposition method (FDM) to turn input audio into sub-bands [3]. From these sub-bands, energy, and variance are extracted for feature selection to determine if sirens were detected [3]. Although this study showed significant accuracy rates (98%) [3], the amount of computational power and resources needed proved it to be inefficient for our situation. FDM and FFT are similar but FDM is a more general approach to wave decomposition compared to FFT which has been optimized for performance.

These prior experimentations show the impact and advancement audio signal processing has recently had on society. For our implementation, FFT was the best due to its speed, efficiency, and accuracy. Current research methods have shown significant advancements in FFT which is why we decided to use it as the baseline for our backend computations [4]. FFT is beneficial for noise reduction which would address the errors in the first case study. Its relatively straightforward implementation (thanks to existing Java libraries) reduces the computational power needed for the analysis of audio inputs, developing a speedy solution to our issue. Another crucial aspect compared to the literature above is our implementation's decrease in time complexity. Compared with the literature, FFT would result in the time complexity going from $O(N^2)$ to $O(N \log N)$, decreasing the filtering rate by 100.

APP DEVELOPMENT

Siren Detection

FFT and General

Our initial proposal was to build a mobile app which detected sirens and alerted the user in order to increase their situational awareness. That goal was achieved, albeit in a modified form due to time constraints. Our initial plan assumed sirens would be fairly uniform and variations would be within a narrow range, but our research and testing revealed a wide variety of sirens in use. Our Android app used the built-in microphone to record an audio stream via `android.media.AudioRecord`, which was then processed to extract the required data for siren detection. The audio buffer was received and a Fast Fourier Transform (FFT) was applied using the `DoubleFFT_1D` function from the JTransforms library. This allowed us to extract the frequency and amplitude data needed [5]. The JTransforms library is a FFT library for Java, which includes functions for various transforms. `DoubleFFT_1D` was used to compute the 1D Discrete Fourier Transform, allowing us to convert the time domain audio data into the frequency domain data we needed for the next step [6].

Detected Frequencies

One of the main considerations of the detection algorithm is the specific high and low frequencies that we aimed to detect. The target high frequency was 1400 Hz and the low frequency was 600 Hz, which approximately matches the frequency used by a popular siren manufacturer [7]. A 10% buffer range was added to both frequencies to account for any imperfect data readings. This resulted in an error buffer for the lower frequency of 60 Hz, making the actual low frequencies to be detected in the range of 540 Hz to 660 Hz. For the high frequency, the error buffer was 140 Hz, making the actual high ranges detected to be from 1260 Hz to 1540 Hz.

Algorithm

Our detection algorithm focused on reading the alternating high and low frequency values. When a high frequency target was read, a `True` value was added to a buffer stack. If a low frequency value was subsequently encountered, a `False` value was stored. Only target frequencies opposite of the frequency represented at the top of the buffer stack are added. When the buffer size reaches 4, it has encountered a full cycle and a half of an oscillating siren, and the system recognizes this as a successful detection.

UI and Alert System

As initially planned, a simple and intuitive UI with an effective alert system was developed. The **homepage**, seen in Figure 1A, includes a green status box that informs the user there are no sirens nearby. A message below the box warns the user that the app must remain open for siren detection. A "I hear a siren!" button was added for users to log false negatives as a part of an unimplemented logging system. The warning text below the permissions button appears if the necessary permissions for the app to detect sirens and alert the user have not been granted. The 'Permissions' button opens the permissions page.

The **permissions page,** seen in Figure 1B, consists of radio buttons which afford that clicking on them will grant the permission. The button's status as being checked or unchecked informs the user if permissions have been granted or not. The text below each permission explains why it is needed. The text at the bottom addresses any privacy concerns. The 'Back Home' button takes the user back to the homepage. The warning text on the homepage disappears once all the permissions are granted.

When a siren is detected the **alert page**, seen in Figure 1C, is displayed in a red box to warn the user about a siren nearby. Audio and haptic feedback are activated when the alert page is open. The 'Thanks for the alert!' button confirms the user has seen the message and takes the user to

the homepage. The 'I didn't hear anything' button was added to log false positives for the intended logging system.

The buttons in the UI are a simple implementation that opens the appropriate page on being clicked. The warning in the homepage is displayed dynamically. The app checks if both permissions are granted. If they are, the TextView element is set to be invisible, if any are not granted, then it is set to be visible [8]. When the permissions page is opened, the app checks both permissions and checks/unchecks the respective radio buttons accordingly. When the user clicks the radio button in the permissions page, the app checks if the permission is granted or not [9]. A message informs the user if the permission is already granted [9]. If it is not, then the permission is requested and the user is shown the dialog to grant or deny permission [9]. Based on the user's choice a message is displayed whether the permission was granted or denied [9]. The audio and haptic alerts are activated on opening the alert page. The app accesses the device's MediaPlayer and Vibrator to play a set alert tone and vibration pattern [10, 11].

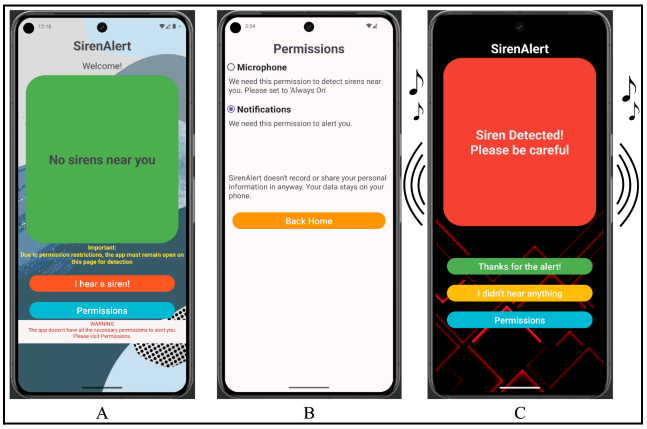

Figure 1. App's homepage, permissions page, and alert page

Compared to the original idea, the final UI implementation was concise by centering around HCI principles.

RESULTS

Siren Detection

Overall, the project achieved its primary goal of developing a mobile app which could detect emergency sirens and generate an alert to the user. The detection module was built using a specific siren sound sample found online and is able to detect it without issue, but further research and development would be needed to support detecting other sirens. Due to difficulties in testing a mobile app on real sirens, we opted to use recorded siren sounds to conduct an experiment on the app's ability to detect sirens, 27 different siren and non-siren, but "siren-like", sounds were played to test the app. Each sound was played 5 times to note consistency. The result was marked: successful - correctly detected all 5 times, partial - correctly detected at least 2 times, false positive - detected non-siren as a siren, and false negative - did not detect siren. Figure 2 displays the results of the test.

Our analysis determined the reason for the false positives was the method of siren detection, where a pattern of frequencies was used. Natural non-siren sounds may occur at the targeted frequencies, resulting in a false positive detection. Increasing the required cycles before an alert is triggered could be a solution, but a careful analysis of the trade off would need to be conducted. While requiring additional cycles would lower the false positive rate, it would also result in delayed alerting when there is a siren.

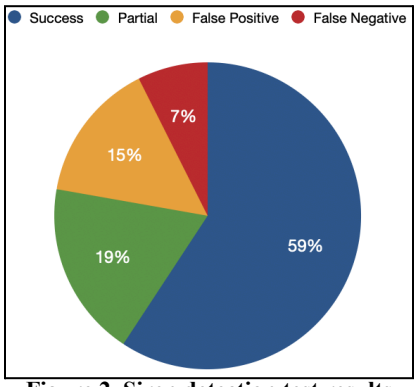

Figure 2. Siren detection test results

UI and Alert System

The UI and alert system were evaluated using subjective quantitative and qualitative methods. The tests were conducted remotely with 5 users of varying ages using the Android Studio emulator. The users were introduced to the app's intention and functionality. They were then asked to navigate through each of the three pages shown in Figure 1 and describe their understanding of the elements on the page. After this run-through, they were asked quantitative and qualitative questions. The app could not be tested outdoors or with a physical phone due to a lack of access to Android smartphones. Likert scale questions were used for subjective quantitative evaluation. The ratings ranged from 1 to 5 where 1-strongly disagree, 2-disagree, 3-neither, 4-agree and 5-strongly agree. The questions and average ratings are shown in Table 1.

Table 1: Likert scale survey results

Statement	Avg. Rating
It was easy to navigate the app	4.6
Clear and easy instructions	4.1
The alert system was effective in alerting me	4.8
Successfully addressed privacy concerns	4.0

Open ended survey questions were used for qualitative evaluation. The questions were: "What did you like most about the UI and alert system?", "Did you feel stuck, confused or frustrated while navigating the app? Where?", "Is there a feature that you would have liked to see in the app?", "Do you have any additional feedback?".

Overall, the testers found the app useful and practical, and the UI and alert system simple and intuitive. The function of the buttons intended for logging false negatives/positives

was not immediately clear to all users. Useful suggestions for features and improvements include: ability to dismiss the alert using side button; changing volume of alert and intensity of vibration based on proximity of siren; replacing green and red status boxes to look more like the pedestrian signals at intersections; using flashing lights and text when alerting user; and using GPS to track when user is nearing an intersection and provide them with an approval/alert if it is safe/unsafe to cross.

CHALLENGES AND ALIGNMENT TO CLASS

Siren Detection

Hardware differences resulted in variations in the data read in via the microphone, despite the identical sound file being used. This posed a challenge as we had planned for the detection to be based on the frequencies present. To address this challenge, we added a buffer to the range of frequencies, allowing for a wider frequency range to trigger the alert. Noise in the input data on the same hardware was also problematic, as the microphone's audio data was inconsistent despite no environmental changes. One possible source for the variation is the Android audio framework itself, which has been shown to have a large variation in latency depending on system load [12]. The buffer approach discussed previously also addressed this issue and we were able to overcome the challenge.

Options for detection algorithms

One of the methods that we considered was focusing purely on differences in frequencies being read to determine if a siren is going off nearby. Alternative methods using more data points such as the wavelength, frequency shift, period, total duration, and other signals may produce more accurate results at the cost of additional complexity and processing time. A machine learning based approach was also briefly considered, but the power and processing limitations on mobile devices precluded their use.

UI and Alert System

The main objectives of creating a simple, intuitive app that effectively alerts the user were accomplished. Developing the features of the app took longer as all members of the team were developing such an app on Android Studio for the first time. The testing and debugging of the app also took longer as only one member had an Android smartphone. A logging system to record instances of false positives/negatives could not be implemented due to time and expertise constraints. Due to Android restrictions, the app can only access the microphone and be effective when it is open and not detect sirens in the background.

The UI and alert system use a lot of principles learned in the course. A number of improvements were made due to feedback received from another team. The app was later tested using evaluation methods introduced in the course. A wireframe prototype was created and refined to form the basis of the UI. The UI was designed with usability principles in mind. Each page has clear visibility of options available to the user, maintains consistency in the elements used, and provides affordances like buttons and radio buttons that hint at what clicking them will do. Every user action maps to a change in the app, and effective feedback is provided when a task is executed. Permission warnings and notifications constrain the user effectively. However, the app can be further refined and improved with more extensive testing and further usability considerations.

CONCLUSION AND REFLECTION

The development of our mobile application ingrained the fundamentals of ubiquitous computing throughout the processes. We successfully developed a mobile application using advanced mathematical concepts and everyday devices utilizing appropriate sensors. Human-computer interaction and design fundamentals along with ubicomp have sparked our mobile app to be a revolutionary advancement for individuals by making society safer. The application has successfully integrated with the environment providing a high efficiency and success rate of siren detection after numerous trials. This development has taught us the importance of sensor data acquisition, connectivity challenges, interoperability, and efficiency. The importance of data storage and filtering mechanisms allowed us to see the concepts and concerns associated with technological advancements today. Connectivity challenges regarding edge case scenarios in real life such as having the sensor covered in a pocket allowed us to think from a designer perspective on the successes and failures, allowing us to adjust. The development of the app shed light on interoperability and future cases for integrating the mobile app into more wearable devices such as smartwatches via Android Studio's smartwatches features. Overall, we learned the importance of developing an efficient mobile application connecting users to the app. We learned that numerous iterations are needed to develop a sophisticated app consisting of low and high-fidelity prototypes. Due to our ability to adapt to these changes, we succeeded.

Figure 2 shows the detection is effective 78% of the time which is due to careful analysis of a siren sample and focusing on the critical properties. This provides a strong foundation to develop a more effective algorithm. The current UI and alert system meet the initial objectives and are simple, intuitive and effective due to the prototyping and usability principles used during development and effective feedback. The logging system development, study of different sirens and trial of alternate detection strategies was limited due to time, expertise and technology constraints. Further steps to improve the app, which were restricted due to the remote nature of the class, as well as limited time and hardware availability, could include: analyzing different siren samples, trying alternate detection methods, conducting real-world testing, evaluating button size, position and color using Fitts' Law, and performing objective quantitative testing of different alert sounds and vibration patterns. The app could also be tested on wearable devices for effectiveness.

REFERENCES

[1] E. Oran Brigham, The Fast Fourier Transform and Its Applications. Pearson, 1988.

[2] Michela Cantarini, L. Serafini, L. Gabrielli, E. Principi, and Stefano Squartini, “Emergency Siren Recognition in Urban Scenarios: Synthetic Dataset and Deep Learning Models,” Lecture notes in computer science, pp. 207–220, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-60799-9_18.

[3] B. Fatimah, A. Preethi, V. Hrushikesh, A. Singh B., and H. R. Kotion, “An automatic siren detection algorithm using Fourier Decomposition Method and MFCC,” IEEE Xplore, Jul. 01, 2020. https://ieeexplore.ieee.org/document/9225414

[4] R. G. Lyons, Understanding Digital Signal Processing. Pearson Education, 2010.

[5] S. Rapuano and F. Harris, “An introduction to FFT and time domain windows,” IEEE Instrum. Meas. Mag., vol. 10, no. 6, pp. 32–44, Dec. 2007, doi: 10.1109/MIM.2007.4428580.

[6] “DoubleFFT_1D (JTransforms 3.1 API).” Accessed: Jul. 20, 2024. [Online]. Available: https://wendykierp.github.io/JTransforms/apidocs/org/jtransforms/fft/DoubleFFT_1D.html

[7] Federal Signal: Safety and Security Systems, "Electronic Siren System 8616033-01: installation, maintenance, and service manual," Federal Signal Corporation, 2012.

[8] S. S. Kshatriya, “What is Android Visibility?,” Educative. https://www.educative.io/answers/what-is-android-visibility (accessed Jul. 21, 2024).

[9] A. Neekhara, “Android | How to Request permissions in Android Application?,” GeeksforGeeks, Jun. 27, 2019. https://www.geeksforgeeks.org/android-how-to-request-permissions-in-android-application/ (accessed: Jul. 21, 2024).

[10] “MediaPlayer | Android Developers,” *Android Developers*. https://developer.android.com/reference/android/media/MediaPlayer (accessed: Jul. 21, 2024).

[11] “Create custom haptic effects | Views,” *Android Developers*. https://developer.android.com/develop/ui/views/haptics/custom-haptic-effects (accessed Jul. 21, 2024).

[12] G. Gokul, Y. Yan, K. Dantu, S. Y. Ko, and L. Ziarek, “Real Time Sound Processing on Android,” in Proceedings of the 14th International Workshop on Java Technologies for Real-Time and Embedded Systems, Lugano Switzerland: ACM, Aug. 2016, pp. 1–10. doi: 10.1145/2990509.2990512.