Accommodating the Deaf Population in the Home Using Raspberry Pi

David Esquivel

Gainesville High School

AP Research

Richard Howard and Chandra Karnati

October 15, 2020

Accommodating The Deaf Population In The Home

In a study on deaf trauma survivors, 17.6% of the deaf or hard-hearing participants in the study have experienced a serious accident in the home or at work. (Anderson et. al, 2016) Although the study mentions that it is unclear whether these accidents are due to the participants being deaf or not, they can be easily avoided if there are accommodations for deaf or hard hearing people in the home. Furthermore, the study failed to mention whether these deaf or hard of hearing individuals had been born deaf or if they developed hearing loss. Regardless, both of these groups are important to accommodate. In 1990, the Americans with Disabilities Act (ADA) was passed. This piece of legislation was made to protect people with disabilities in three areas: employment, public life, and public accommodation. (Stein & Teplin, 2011) Although this piece of legislation has been helpful for the deaf and hard-hearing community, there are still several accidents happening at work but more importantly in their own home. The ADA, or any other piece of legislation, does not address accomodations for the deaf and hard of hearing in their own home.

The use of technology can be a way to accommodate the deaf or hard of hearing population in their own homes. The most common technology used among deaf or hard of hearing people are hearing aids. However, hearing aids can be quite expensive and it is inconvenient to wear around the house. This is where technologies such as the Raspberry Pi can be introduced. Raspberry Pi is a mini-computer that can be programmed to do whatever the programmer desires. The creators of Raspberry Pi call the Raspberry Pi 4 B, the most recent model, “your tiny, dual-display, desktop computer… and robot brains, smart home hub, media centre, networked AI core, factory controller, and much more”. (*Buy a Raspberry Pi 4 Model B –*

Raspberry Pi, 2020) What makes the Raspberry Pi a considerable option is how cost efficient it is. The Raspberry Pi 4 B model costs a mere $35. (*Buy a Raspberry Pi 4 Model B – Raspberry Pi*, 2020) This is compared to the cost of a hearing aid which can range anywhere from just shy of $1,000 to $4,000. (*Hearing Aids Cost & Pricing Models | Miracle-Ear*, 2017)

Raspberry Pi can be used in the home as a visual alarm system to alert a deaf or hard of hearing person of sounds going off in the home. Examples of this can be smoke alarms, oven timers, microwave ovens, and even washing machines. Raspberry Pi can be used to recognize these sounds and send the deaf or hard of hearing person a notification straight to their mobile device, regardless of brand, as long as they have an active email.

Literature Review

The establishment of the Americans with Disabilities Act (ADA) was a breakthrough in the disabled community as it was the "world's first comprehensive declaration of equality for people with disabilities." (*ADA - Findings, Purpose, and History | ADA Anniversary Tool Kit*, 2010) The congressional intent behind the ADA, according to Hayley M. Koteen, who is from the Benjamin N. Cardozo School of Law, was to "to remove barriers, to bring equality to individuals with disabilities and to give them the opportunity to fully participate in society". As previously stated, the ADA only addressed three areas: employment, public life, and public accommodation. Thus, the ADA does not fulfill Congress's intent as there are still barriers inside the home and disparities between deaf or hard hearing people and regular people. Public buildings are required to have a visual signal and/or a vibratory signal linked with the building alarm system by the ADA. (Glick, 1998) Glick continues by saying that "people who are deaf or hard of hearing require visual or vibrating alerting devices to inform them of visitors, messages,

etc., in their rooms or offices." Although this is required at the workplace, it is not required in homes. Homes are not even required to have visual signals for smoke alarms and if the deaf or hard of hearing person is not in the same room as an oven, microwave or washing machine, they are unable to see the alarm going off. This concept must be adapted inside of the home.

Using technology to help the deaf and hard of hearing population is not groundbreaking nor is it anything new. A prime example of this would be the use of the hearing aid. It is fair to say that the hearing aid is one of the most, if not the most, common technologies used in the deaf community. Although most hearing aids are effective and even helps older, deaf or hard of hearing people prevent or reduce the effects of dementia. (Livingston et al., 2020) However, the use of hearing aids remains low. A study found that the hearing aid use among those who it could potentially benefit ranged from a small percentage in some countries to 20-30% in others. (Chien and Lin, 2012) Another study that analyzed Chien and Lin's study stated that the low use in hearing aids could be caused, in part, by the high cost of hearing technologies. There have also been attempts to make face-to-face assistive technology. A 2007 study tried to make a PC supporting software, which also worked on a mobile device, that could translate sign language into oral french. Even though it made face-to-face communication possible, it only went from the deaf person to the hearing person. The main purpose of the study was to examine how much more comfortably the deaf or hard of hearing person would act in public or social settings. The study concluded that the assistive did not have a significant effect on the deaf or hard of hearing person's confidence or comfort level in a social setting. (Vincent et. al, 2007) Although some may have seen the study as a failure, the intent behind the study and the development of the assistive technology itself was a step forward into accommodating the deaf and hard of hearing

population. Nonetheless, the two previously mentioned technologies were proven too costly or ineffective in order to be used in the household.

There is an alternative, cost-efficient, and effective technology that could be used inside the home to accommodate the deaf or hard of hearing population: Raspberry Pi. Raspberry Pi, as previously mentioned, is a mini-computer that can be programmed from scratch in multiple programming languages such as Python and C++. This mini-computer has great capabilities as it has been used to make a fully functioning home automation system. A 2020 study used the Raspberry Pi Model 2B to make the home automation system. This study thoroughly described the 2B model; it "contains A 900 MHz quad-core ARM Cortex-A7 CPU (Central Processing Unit) processor along with 1 GB of RAM (Random Access Memory). Arm Cortex-A7 is built on the energy-efficient 8-stage pipeline and has an integrated L2 cache designated for low-power and reduced latency. It supports 100 MBPS Ethernet and contains 4 USB ports along with 40 GPIO pins. Full HDMI (High Definition Multimedia Interface) support with camera and card interface is available as well as Micro SD card. It contains 3.5 mm audio jack and composite video support too." (Ashraf et al, 2020) The home automation system created using Raspberry Pi in this study was actually fully functional but was made to a small scale. The home in the study was actually a small, 3-dimensional, home model with three floors that included regular home appliances such as lights, fans, etc. The user could command the home automation system through an application on their mobile phone which connected to the Raspberry Pi through either the Microsoft IP cloud or the local network. From there, the Raspberry Pi could turn any appliance on or off any appliance the user desired. The capabilities of Raspberry Pi grow further. In a 2018 study, Raspberry Pi was used to capture and display noise levels in a library in certain

areas of the library. Once these noise levels were displayed to library-goers, they could decide which areas of the library were the quietest in which they could go study or perform tasks. (Kung, 2018) These studies truly display the capabilities of Raspberry Pi and its ability to capture sound will be important in this study.

Technology that recognizes sound, and even music, exists outside of Raspberry Pi. A 2005 study was able to create an automatic transcriber of music that could adapt to different instruments such as a guitar and piano. The algorithm used in the study analyzed a PCM Wave File and put it through a GT Filter Bank and a Meddis Filter Bank. From then on, the algorithm found which mode to transcribe it in: polyphonic, monophonic, and training. Lastly, the algorithm whether to use partial tracking, neural net bank, pitch detection, or pattern set creations in order to find the pitch and onset file or the pattern file that corresponds to the instrument. (Bruno & Nesi, 2005) A similar, simpler, and much more applicable form of this technology has been made since. Today, there is an application known as SHAZAM or as the SHAZAM Music Recognition Service. This application records a small segment of a song that is currently being played out loud and recognizes it. Once it recognizes the song, it gives you the song title, song artist, and even album information. The algorithm uses tokens, or tones that are specific to a song, and the time between each token to identify a song. (Wang, 2006) This application has immense popularity as it has over a billion downloads (themusicnetwork.com, 2019) and is pre-installed in the popular social media platform, Snapchat.

Technology that is used in these music recognition services can be applied to Raspberry Pi in order recognize alarms in a deaf or hard of hearing person's house such as a smoke alarm,

oven timer, microwave oven, and a washing machine. Once it recognizes one of these tones, it can notify the deaf or hard of hearing person through an email notification.

Problem Statement

There is a problem with disabled people and accessibility within their own home. Despite the need for technological advancement in order to improve accessibility, few advancements have been made to do so. This problem has negatively impacted deaf people specifically, because they are unaware of most noises that go off in their home, such as a microwave, oven, or washing machine. A possible cause of this problem is the failure to take into consideration the deaf population and the home into the issue of accessibility. Perhaps a study that investigates music recognition using machine learning so that a computer can recognize a washing machine tone and notify a deaf person that their clothes are done could remedy the situation.

Research Question and Hypothesis

Is Raspberry Pi a viable solution to create assistive technology for home appliances to accommodate the deaf or hard of hearing population? More specifically, can it be used to notify a deaf or hard of hearing person whenever their clothes are done washing?

Raspberry Pi will be a viable solution to create assistive technology for home appliances to accommodate the hard of hearing population because it has the best combination of cost-efficiency and effectiveness. More specifically, it will be able to be used to notify a deaf or hard of hearing person whenever their clothes are done washing.

Limitations and Delimitations

This study will specifically focus on the recognition of a washing machine tone and notifying a deaf or hard of hearing person through email. This is in order to take into consideration the time the researcher has and an increase in the complexity of the code required to be able to recognize multiple sounds.

The main limitation to this study is that there will be deaf or hard of hearing people who will not be as technologically inclined as the researcher. This is important because the deaf or hard of hearing user must be able to input their email in order to use it. Furthermore, the adopted algorithm is simply a prototype evaluating only its effectiveness and not its market value or convenience.

Methods

The question at hand remains, is Raspberry Pi a viable solution to create assistive technology for home appliances to accommodate the deaf or hard of hearing population and can it be used to notify someone that their clothes are done washing? In order to answer the question, a model of such a system was produced. Therefore, the methodology used was project-oriented experimental design. Experimental design is defined as “a method of research… in which a controlled experimental factor is subjected to special treatment for purposes of comparison with a factor kept constant,” according to Merriam-Webster. The reason the methodology was a project-oriented hybrid of experimental design was because there was a project goal to achieve. The project goal was an audio recognition system that can send a notification via email. The main study that this research was modeled after was a 2020 study that used Raspberry Pi to create a home automation system that could be controlled by a smartphone application. The

study, named “Home automation using general purpose household electric appliances with Raspberry Pi and commercial smartphone,” was a project-oriented experimental design which had a home automation system controlled by a smartphone application as the project goal and the system was then experimented with. This study was posted in a journal by the Public Library of Science, holding great credibility. The methodology of this study will be fairly modeled after the mentioned 2020 study. Meta-analysis was inappropriate because there were no array of studies that cover the research topic or project goal specifically. With similar studies having similar methodology and the lack of any other methodology being able to best answer the question at hand, project-oriented experimental design was the most appropriate methodology to use. The functionality of the audio recognition system was not compared to other systems or assistive technologies due to the lack of access to other assistive technologies to the researcher. In order to create the most optimal model of this system, a microphone must be accompanied with the Raspberry Pi 4 Model B. Furthermore, open coding forums were used to assist the researcher and were properly cited.

Raspberry Pi 4 Model B was used to create an audio recognition system that sent a notification to the deaf or hard of hearing user once the audio was recognized. The programming language used was Python. Python was used because it is the most typical language used to code audio recognition systems and it is the language that needed to be used in the Ubuntu Desktop environment. A condenser microphone had to be plugged into Raspberry Pi through an audio interface. Third-party programming libraries were imported for the completion of the code such as Git, which is where the researcher found the open source project audio_recognition_system. The open source project contains an array of codes that must be used in order to have an

effective algorithm. Furthermore, several Python packages were installed such as matplotlib, termcolor, scipy, pydub, wave, pyaudio, and several others. The main basis behind the code stemmed from the Ubuntu Desktop environment and its ability to create a similar program that the researcher plans to create. This similar code, which was originally meant to be used specifically on the Ubuntu Desktop environment, was then changed and manipulated in order to function in Raspberry Pi. Using the Ubuntu Desktop environment, specific dependencies were needed in order for the program to run such as the installation of python-tk and ffmpeg. This was manipulated by the researcher in order for the code to properly run without the use of the Ubuntu Desktop environment and instead runs using Raspberry Pi. From there, the audio_recognition_system was imported. Then, an audio recording of a washing machine tone in a wav format was recorded. This wav file was imported to the researcher's computer and then into the Raspberry Pi as reference for audio recording. From there, the wav audio file's fingerprints were generated and stored using pyaudio functions as data known as hashes. Lastly, the code was asked to recognize the audio using the microphone with a pyaudio function and return the name of the wav file. Five seconds of the washing machine tone were recorded in three different circumstances: silence, conversation, and supply noise in the washing room.

Continuing, an email must have been sent to the user. A testing Gmail account was set up for the researcher in order to have a clean inbox specifically used for test emails. First, a local Simple Mail Transfer Protocol (SMTP) server will be created for test emails. Then, a secure SMTP server was created, connecting to the researcher's Wi-Fi and using their WiFi as a constant. Since only a plain text email was required, the Yagmail package was used. The code consisted of an if statement that checks whether the audio file recognized is the same as the

audio file imported to Raspberry Pi. If the statement is true, the email code will run. The section of the code regarding the email simply consisted of setting up a secure SMTP, entering the sending email address, the receiving email address, and the reception of the message. From there, 15 trials of each condition were tested and the amount of successes in each condition, with a success being a sent and received email when the washing machine goes off, were gathered. The target goal was a 75% success rate for each condition. The success rate was then put through a z-test for proportion to determine whether or not the success rate was significant even if it was above 75%. The time it took to receive the email was also recorded using a stopwatch.

This specific instrument, which was a computer algorithm using Raspberry Pi, was the most appropriate in order to answer the question at hand. The development of the algorithm was gathered from several different sources along with development from the researcher, making it the most optimal algorithm development possible. The instrument best gathered the specific data needed data to illustrate effectiveness and efficiency. No other instrument or set of instruments could better gather data. Analyzing several studies could not have worked because there are no arrays of studies that cover the research topic or the research goal of this study. Surveying or interviewing individuals would not be appropriate because this study focuses on the effectiveness of a system that can only be developed and not individuals.

Data/Findings

Table 1. Raspberry Pi Successes, Time Response, Confidence and Hash Matches for Condition 1, Silence

Trial (#)	Condition 1 Responses			
	Successes	Time (seconds)	Confidence	Hash Matches

1	1	4.37	63	67
2	1	5.57	49	49
3	1	5.24	10	13
4	1	6.22	54	55
5	1	6.0	12	13
6	1	6.29	47	48
7	1	5.94	72	72
8	1	5.67	80	80
9	1	5.42	5	8
10	1	6.12	61	61
11	1	6.14	48	48
12	1	6.27	25	26
13	1	5.74	4	7
14	1	5.84	60	60
15	1	5.77	60	60

Table 2. Raspberry Pi Successes, Time Response, Confidence and Hash Matches for Condition 2, Background Conversation

Trial (#)				
	Condition 2 Responses			
	Successes	Time (seconds)	Confidence	Hash Matches
1	1	6.22	59	59
2	1	5.69	3	4
3	1	5.99	20	20

4	1	5.82	56	56
5	1	5.69	3	4
6	1	4.8	7	7
7	1	5.74	22	22
8	1	5.02	17	17
9	1	4.94	2	2
10	1	5.69	22	22
11	1	5.59	45	45
12	1	6.14	8	9
13	1	6.37	9	9
14	1	6.02	12	12
15	1	10.44	31	31

Table 3. Raspberry Pi Successes, Time Response, Confidence and Hash Matches for Condition 3, Sudden Background Noise

Trial (#)				
	Condition 3 Responses			
	Successes	Time (seconds)	Confidence	Hash Matches
1	1	5.05	26	26
2	1	7.41	8	8
3	1	5.9	10	10
4	1	6.04	6	6
5	1	5.77	45	46
6	1	5.30	26	26
7	1	4.89	38	38

8	1	5.5	6	7
9	1	5.59	19	19
10	1	5.35	56	56
11	1	5.46	58	56
12	1	5.67	62	62
13	1	6.25	4	5
14	1	6.04	9	12
15	1	6.62	14	14

Analysis

In order to determine the effectiveness of the audio recognition system, the success rate of the system for each condition and its significance must be evaluated. As seen above, Condition 1 was silence, Condition 2 was background conversation, and Condition 3 was sudden background noise that you would typically hear in a laundry room. Successes, as shown in the tables, were marked as a 1 and failures were marked as a 0. Response times were evaluated in order to illustrate effectiveness of the system in emergency use. Lastly, the hash matches, which are the matches of audio data from the reference wav file and the file being recorded, were recorded along with the confidence, which is the amount of hashes out of the hash matches that the code is entirely sure are a match, in order to analyze the quality of recognition.

As shown in Tables 1-3, there was an average success rate of 1 per trial, meaning there was a 100% success rate for all three conditions. A one-sample z-test for proportion for each condition was going to be used to test the significance of each success rate. However, with an

average of 1 success per trial and a standard deviation of 0 for each condition, a z-test for proportion is redundant. The reasoning behind this is because the p-value would be 0, meaning that the probability of the success rate being less than 0.75, the intended success rate, in a large number of trials is 0, according to the data the researcher collected. The average response time for the algorithm under Condition 1 was 5.77 seconds, under Condition 2 was 6.01 seconds, and under Condition 3 was 5.79 seconds, meaning that the average response time for each condition was able to stay within a range of 0.34 seconds. There were an average of 44.5 hash matches with an average confidence of 43.33, which is a 97.4% confidence rate, under Condition 1. Under Condition 2, there were an average of 21.27 hash matches with an average confidence of 21.07, which is a 99% confidence rate. Lastly, under Condition 3, there were an average of 26.07 hash matches with an average confidence of 25.80, which is a 99% confidence rate.

Since, there was a 100% success rate in the trials and there is a p-value of 0, the next best measure of effectiveness is hash matches and confidence. Under Condition 1, as shown in Table 1, identifying hash matches was not very difficult as it had the highest average hash matches out of the three conditions. This could likely be due to Condition 1 since the room was silent and there were no other sounds being recorded. We can also say this because under Condition 2, the average number of hash matches dropped significantly. With the average number dropping by 23.23 hashes, it is appropriate to say that the difference in numbers could be due to the condition, especially because in Condition 2, there is constant background noise (conversation) that the microphone is picking up in comparison to silence. Furthermore, with the microphone being a condenser microphone, meaning it is made to catch background noise, it is likely that the background conversation had an impact on the number of hash matches. Lastly in Condition 3,

the number of hash matches began to rise but still remained fairly low. Once again this could be due to Condition 3 only being a sudden, instantaneous noise, rather than a constant noise like Condition 2. Therefore, the algorithm was still recording a noise it did not recognize but, since it was a noise that only lasted a few moments rather than seconds, it could continue to record and pick up other hash matches that it could not under Condition 2. Nonetheless, it is important to consider extreme low hash match cases such as Trial 9 under Condition 2. As seen in Table 2, there were only 2 hash matches recorded but since the confidence was also 2, the algorithm allowed it to run as a match, however, it should be noted that the trial was two less hash matches away from being a failure. This means that even though there was a 100% success rate in the trials, it could have easily not been and it does not mean that with a large amount of trials there will be a 100% success rate. Each p-value of each z-test for proportion only showed that there was 0 probability that with a large amount of trials the success rate could not be lower than 75%.

Since confidence was high and did not vary much across the three conditions, the only conclusion that can be made is that whenever the algorithm identifies a hash match, it is confident that the hash match is a correct one, at least, on average 97% of the time.

Conclusion

This study has contributed to filling the gap previous research has left regarding assistive technology for the deaf or hard-of-hearing population at home. With the results shown above, it can be concluded that Raspberry Pi is a viable solution to create assistive technology for home appliances to accommodate the deaf or hard of hearing population by notifying them when these home appliances make specific sounds. The study has, therefore, found not only a viable solution but also introduced a cheaper solution to other technologies such as hearing aids. After complex

and rigorous manipulation of the program, the researcher's hypothesis was proven. With the implementation of this system, the number of accidents in the home involving the deaf and/or hard-of-hearing population can be reduced. Furthermore, the implementation of this system could create a market for revolutionary assistive technology for not only the deaf but also other disabilities in order to accommodate them in their own home.

Future Directions

Even though the data presented above concludes that the assistive technology is effective, more research must be done on the convenience of the device. Since the Raspberry Pi setup required a monitor, mouse, keyboard, audio interface, and a condenser microphone, research can be conducted in order to reduce the space taken up by the setup in order to make it more convenient to the consumer. This research could turn the viable prototype presented in this study and turn it into a product that is ready for the market.

Finally, in order for Raspberry Pi to compete with other forms of assistive technology, such as hearing aids, a comparative study between Raspberry Pi and other assistive technology must be set up. This study has simply introduced a viable prototype but its effectiveness and viability compared to other assistive technologies has not been studied, therefore, it cannot yet be considered a better alternative. For now, due to the price, the viable prototype presented in this study can only be considered a cost-efficient alternative.

References

ADA - Findings, Purpose, and History | ADA Anniversary Tool Kit. (2010). Adaanniversary.Org. https://www.adaanniversary.org/findings_purpose#:~:text=At%20the%20signing%20of%20the,26%2C%201990%2C%20President%20George%20H.W.&text=The%20ADA%20was%20the%20world's,equality%20for%20people%20with%20disabilities.

Anderson, M., Wolf Craig, K., Hall, W., & Ziedonis, D. (2016). A Pilot Study of Deaf Trauma Survivors' Experiences: Early Traumas Unique to Being Deaf in a Hearing World. *Journal of Child & Adolescent Trauma*, *9*(4), 353–358. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.1007/s40653-016-0111-2

Ashraf, I., Umer, M., Majeed, R., Mehmood, A., Aslam, W., Yasir, M. N., & Choi, G. S. (2020). Home automation using general purpose household electric appliances with Raspberry Pi and commercial smartphone. *PLoS ONE*, *15*(9), 1–20. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.1371/journal.pone.0238480

Bruno, I., & Nesi, P. (2005). Automatic Music Transcription Supporting Different Instruments. *Journal of New Music Research*, *34*(2), 139. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.1080/09298210500148652

Buy a Raspberry Pi 4 Model B – Raspberry Pi. (2020). Raspberrypi.Org. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

Definition of EXPERIMENTAL DESIGN. (2020). Merriam-Webster.com. https://www.merriam-webster.com/dictionary/experimental%20design

Glick, P. B. (1998). The ADA and Technological Solutions for Achieving Effective Communication with Hard of Hearing and Deaf People. *Journal of Urban Technology*, *5*(1), 45–63. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.1080/10630739883985

Hearing Aids Cost & Pricing Models | Miracle-Ear. (2017). Miracle Ear. https://www.miracle-ear.com/hearing-aids/cost

Koteen, H. M. (2011). Ending the Disconnect for the Deaf Community: How Amendments to the Federal Regulations Can Realign the Ada with Its Purpose. *Cardozo Arts & Entertainment Law Journal*, *29*(2), 425–457.

Kung, J. Y. C. (2018). Chapter 3: Raspberry Pi and Arduino Prototype: Measuring and Displaying Noise Levels to Enhance User Experience in an Academic Library. *Library Technology Reports*, *54*(1), 18–22.

Livingston, G., Huntley, J., Sommerlad, A., Ames, D., Ballard, C., Banerjee, S., Brayne, C., Burns, A., Cohen-Mansfield, J., Cooper, C., Costafreda, S. G., Dias, A., Fox, N., Gitlin, L. N., Howard, R., Kales, H. C., Kivimäki, M., Larson, E. B., Ogunniyi, A., … Mukadam, N. (2020). Dementia prevention, intervention, and care: 2020 report of the Lancet Commission. *The Lancet*, *396*(10248), 413–446. https://doi.org/10.1016/s0140-6736(20)30367-6

Reid, B. E. (2020). Internet Architecture and Disability. *Indiana Law Journal*, *95*(2), 592–647.

Stein, M. S., & Teplin, E. (2011). Rational discrimination and shared compliance: lessons from Title IV of the Americans with Disabilities Act. *Valparaiso University Law Review*, *45*(3), 1095–1141.

themusicnetwork.com (2019, October 13). *Shazam has 478m users worldwide, profits up after*

Apple takeover. The Music Network. https://themusicnetwork.com/shazam-active-users/#:~:text=Shazam%20counts%20over%2020%20million,the%20app%20across%20the%20world.

Vincent, C., Deaudelin, I., & Hotton, M. (2007). Pilot on evaluating social participation following the use of an assistive technology designed to facilitate face-to-face communication between deaf and hearing persons. *Technology & Disability*, *19*(4), 153–167.

Wang, A. (2006). The Shazam Music Recognition Service. *Communications of the ACM*, *49*(8), 44–48. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.1145/1145287.1145312

Yong, M., Willink, A., McMahon, C., McPherson, B., Nieman, C. L., Reed, N. S., & Lin, F. R. (2019). Access to adults' hearing aids: policies and technologies used in eight countries. *Bulletin of the World Health Organization*, *97*(10), 699–710. https://doi-org.proxygsu-sgai.galileo.usg.edu/10.2471/BLT.18.228676

Appendix A

Sample Code of audioRecorder.py

```
import pyaudio
import wave

#record and save audio
form_1=pyaudio.paInt16
chnnls=1
samp_rate=44100
samp_buff=4096
rec_secs=5
dev_index=2
wav_filename='washingMachineTone6.wav'

audio = pyaudio.PyAudio()

stream=audio.open(format = form_1,rate = samp_rate,channels = chnnls,\
                input_device_index = dev_index,input = True,\
                frames_per_buffer = samp_buff)

print("recording audio")
frames=[]

for ii in range(0,int((samp_rate/samp_buff)*rec_secs)):
    data=stream.read(samp_buff)
    frames.append(data)

print("audio recorded")

stream.stop_stream()
stream.close()
audio.terminate()

wavefile = wave.open(wav_filename,'wb')
wavefile.setnchannels(chnnls)
wavefile.setsampwidth(audio.get_sample_size(form_1))
wavefile.setframerate(samp_rate)
wavefile.writeframes(b''.join(frames))
wavefile.close()
```

Appendix B

Sample Code of collect-fingerprints-of-songs.py

```
for filename in os.listdir(path):
    if filename.endswith(".wav"):
      reader = FileReader(path + filename)
      audio = reader.parse_audio()

      song = db.get_song_by_filehash(audio['file_hash'])
      song_id = db.add_song(filename, audio['file_hash'])

      msg = ' * %s %s: %s' % (('id=%s'),       # id
        ('channels=%d'), # channels
        ('%s')           # filename
      )
      print (msg % (song_id, len(audio['channels']), filename))

      if song:
        hash_count = db.get_song_hashes_count(song_id)

        if hash_count > 0:
          msg = '   already exists (%d hashes), skip' % hash_count
          print (msg)

          continue

      print ('   new song, going to analyze..')

      hashes = set()
      channel_amount = len(audio['channels'])

      for channeln, channel in enumerate(audio['channels']):
        msg = '   fingerprinting channel %d/%d'
        print ((msg) % (channeln+1, channel_amount))

        channel_hashes = fingerprint.fingerprint(channel, Fs=audio['Fs'],
plots=config['fingerprint.show_plots'])
        channel_hashes = set(channel_hashes)

        msg = '   finished channel %d/%d, got %d hashes'
        print ((msg) % (
```

```
        channeln+1, channel_amount, len(channel_hashes)
      ))

      hashes |= channel_hashes

    msg = '   finished fingerprinting, got %d unique hashes'

    values = []
    for hash, offset in hashes:
      values.append((song_id, hash, offset))

    msg = '   storing %d hashes in db' % len(values)
    print (msg)

    db.store_fingerprints(values)

print('end')
```

Appendix C

Sample Code of washingMachineAlerter.py

```
def align_matches(matches):
  diff_counter = {}
  largest = 0
  largest_count = 0
  song_id = -1

  for tup in matches:
    sid, diff = tup

    if diff not in diff_counter:
      diff_counter[diff] = {}

    if sid not in diff_counter[diff]:
      diff_counter[diff][sid] = 0

    diff_counter[diff][sid] += 1

    if diff_counter[diff][sid] > largest_count:
      largest = diff
      largest_count = diff_counter[diff][sid]
```

```
    song_id = sid

  songM = db.get_song_by_id(song_id)

  nseconds = round(float(largest) / fingerprint.DEFAULT_FS *
             fingerprint.DEFAULT_WINDOW_SIZE *
             fingerprint.DEFAULT_OVERLAP_RATIO, 5)

  return {
     "SONG_ID" : song_id,
     "SONG_NAME" : songM[1],
     "CONFIDENCE" : largest_count,
     "OFFSET" : int(largest),
     "OFFSET_SECS" : nseconds
  }

total_matches_found = len(matches)

print ('')

if total_matches_found > 0:
  msg = ' ** totally found %d hash matches'
  print (colored(msg, 'green') % total_matches_found)

  song = align_matches(matches)

  msg = ' => song: %s (id=%d)\n'
  msg += '    offset: %d (%d secs)\n'
  msg += '    confidence: %d'

  print (colored(msg, 'green') % (
    song['SONG_NAME'], song['SONG_ID'],
    song['OFFSET'], song['OFFSET_SECS'],
    song['CONFIDENCE']))
  #sending emmail
  yagmail.register('davesqtest@gmail.com','Imdavid03')
  reciever="davesqtest@gmail.com"
  body="your clothes are done washing"
  #filename="wav/testaudio8.wav"

  yag=yagmail.SMTP("davesqtest@gmail.com")
  yag.send(
     to=reciever,
     subject="laundry update",
```

```
      contents=body,
      #attachments=filename
      )
  print("email sent")

 else:
  msg = ' ** not matches found at all'
  print (msg)
```

Appendix D

Sample Code of db.py, a helper algorithm

```
class Database(object):
  TABLE_SONGS = None
  TABLE_FINGERPRINTS = None

  def __init__(self, a):
    self.a = a

  def connect(self): pass
  def insert(self, table, params): pass

  def get_song_by_filehash(self, filehash):
    return self.findOne(self.TABLE_SONGS, {
      "filehash": filehash
    })

  def get_song_by_id(self, id):
    return self.findOne(self.TABLE_SONGS, {
      "id": id
    })

  def add_song(self, filename, filehash):
    song = self.get_song_by_filehash(filehash)

    if not song:
      song_id = self.insert(self.TABLE_SONGS, {
        "name": filename,
        "filehash": filehash
      })
    else:
```

```
    song_id = song[0]

  return song_id

 def get_song_hashes_count(self, song_id):
  pass

 def store_fingerprints(self, values):
  self.insertMany(self.TABLE_FINGERPRINTS,
   ['song_fk', 'hash', 'offset'], values
  )
```

Appendix E

Sample Code of fingerprint.py, a helper algorithm

```
def generate_hashes(peaks, fan_value=DEFAULT_FAN_VALUE):
   if PEAK_SORT:
    sorted(peaks,key=itemgetter(1))

   # bruteforce all peaks
   peaks=list(peaks)
   len_peaks=len(peaks)
   for i in range(len_peaks):
    for j in range(1, fan_value):
     if (i + j) < len(peaks):

      # take current & next peak frequency value
      freq1 = peaks[i][IDX_FREQ_I]
      freq2 = peaks[i + j][IDX_FREQ_I]

      # take current & next -peak time offset
      t1 = peaks[i][IDX_TIME_J]
      t2 = peaks[i + j][IDX_TIME_J]

      # get diff of time offsets
      t_delta = t2 - t1

      # check if delta is between min & max
      if t_delta >= MIN_HASH_TIME_DELTA and t_delta <= MAX_HASH_TIME_DELTA:
       freq1=str(freq1)
```

```
        freq2=str(freq2)
        t_delta=str(t_delta)
        h = hashlib.sha1(("%s|%s|%s" % (str(freq1), str(freq2), str(t_delta))).encode('utf-8'))
        yield (h.hexdigest()[0:FINGERPRINT_REDUCTION], t1)
```

Appendix F

Sample Code of db_sqlite.py, a helper algorithm

```
class SqliteDatabase(Database):
  TABLE_SONGS = 'songs'
  TABLE_FINGERPRINTS = 'fingerprints'

  def __init__(self):
    self.connect()

  def connect(self):
    config = get_config()

    self.conn = sqlite3.connect(config['db.file'])
    self.conn.text_factory = str

    self.cur = self.conn.cursor()

    print(colored('sqlite - connection opened','white',attrs=['dark']))

  def __del__(self):
    self.conn.commit()
    self.conn.close()
    print(colored('sqlite - connection has been closed','white',attrs=['dark']))

  def query(self, query, values = []):
    self.cur.execute(query, values)

  def executeOne(self, query, values = []):
    self.cur.execute(query, values)
    return self.cur.fetchone()

  def executeAll(self, query, values = []):
    self.cur.execute(query, values)
    return self.cur.fetchall()
```