Data Transmission Using Sound

Prashanth Soundarapandian
111498721
psoundarapan@cs.stonybrook.edu

Arjun Mathew Dan
111492985
amathewdan@cs.stonybrook.edu

Mohan Gandhi Alapati
111493744
malapati@cs.stonybrook.edu

***Abstract*—Our work encodes data into a series of audible near-ultrasonic pitches to form a "sonic codeword".The data is encoded on the sending device before being transmitted.The data is then transmitted over the air, to a receiving device, or group of devices where it is decoded. Any device with a speaker can emit a chirp and most devices with a microphone and a small amount of processing power can receive and decode it. We explored the feasibility of using sound waves as a medium of data transmission. We describe the implementation of an encoder and decoder, as well as two prototype models based on the this concept - a chat client and a fire alarm detector to demonstrate this functionality. We were able to achieve transmission rate of upto 1 byte per second for arbitrary binary data, with an accuracy of 78.5% when broadcasting across 0 feet, and up to 65.6% accuracy when broadcasting across 10 feet.The data rate is bit low but it suffices requirement of the IOT applications prototyped.**

***Keywords*—ConditionalFreqDist, Markov-API, SMTP, Android, NLTK**

I. Introduction

In this project, we explored the possibility of using sound waves for data transmission. Using sound for data transfer comes with many advantages. The main one being, you can do data transfer without the reliance on network connectivity. We started off with an Amplitude modulation based approach. i.e to use different amplitude levels for the different characters in the text message that is to be transmitted. Owing to performance reasons, we switched to a Frequency based approach. i.e every character in the English alphabet is mapped to a particular frequency. Ignoring the background sound or reducing its impact on the data transfer is key to the performance of this data transfer technique. The throughput is still quite low; However, there are numerous applications or use-cases wherein this is not a major bottleneck. We had also considered a few such use-cases as part of this project. We also demonstrated that for small messages communication is quite feasible using standard microphones and speakers shipped in consumer electronics.

II. Motivation

WiFi is one of the most dominant radio's used in wireless embedded systems today. But the main drawback of using such a radio is that, it consumes a lot of power, especially in devices that are small. We wanted to explore the possiblity of using alternative communication channel in such devices. Given the fact that the data exchange rate between such devices is small, gave us some flexiblity in coming up with a prototype. Therefore, we were interested in exploring the feasibility of ultrasonic communication as a covert channel in standard consumer hardware.

These are some of the usecases that we think are some interesting applications of data transmission using sound.

- Can be a cost-efficient solution to communicate with limited or no network coverage like in the case of travelling by an aircraft.
- Can be used as a covert channel for wireless communication which is a means of security by obscurity.
- Devices like fire alarm detector and coffee maker have their own signature sounds which can be detected using this mechanism to take alternative actions.
- Can be run on a low-cost device like a raspberry-pi device or an Arduino board which is cost effective compared to Bluetooth or WiFi, especially in conferences where there is too much stress on the WiFi network, ultrasonic encoding can be used to send messages.

III. Signature detection

signature detection works on this model : it has an encoder and decoder which can exchange a signature over sound corresponding to a particular byte of data and decode the signature and pass it to a model to predict the data being transmitted. Two applications have been explored using this idea.

IV. Implementation

We designed and developed multiple applications that rely on small scale data transmission using sound. The list is as follows :

A. Fire Alarm Detection

The first application we have prototyped is a basic fire alarm detector. The fire alarm emits a continuous monotonous frequency when activated .This is a unique signature compared to other background noises. A small device like a raspberry-pi can decode this sound signature and trigger a push notification or an e-mail accordingly. For our prototype, we have used a MAC book pro laptop for detection of this signature and an e-mail is used to notify about the event.

B. Sound based messaging

The second application is a sound based communicator which can be used for low distance communication. Basically the encoder uses 26 distinct frequencies in the range of 200-1200Hz range to transmit the distinct characters and the decoder performs FFT and prediction logic to decipher the symbols received and maps them accordingly.

C. SoundBot

We wanted to explore on the possiblity of using the data decoded at the receiver's end to develop additional usescases. So we came up with a self learning bot that could take intelligent decisions given a set of states to respond to a given situation. This application can be easily extended to several other complex IOT devices like connected cars etc.Here the states are defined by the data that is decoded at the receiver's end. Ex. What is the right set of actions in the event of a fire alarm.

V. Encoding

For a given signal frequency, a .wav file of 1 second duration is scaled up to an amplitude which almost maxes out the volume of a generic audio signal and then transmitted.

A. Modified Frequency Shift Keying

Our encoding scheme use a modified version of FSK. We add an addtional frequency to mark the start of the new character. This provides error correction: if we end up missing a bit or adding a bit, the next presence of the byte start signal indicates that we should start the next byte at that point.

Additionaly we have used a range of frequencies to decrease the error rate. Presently, we are using the frequency range of 200-1200Hz for prototyping our application. This range of spectrum is divided in 26 distinct band of frequencies each frequency being an increment of 40Hz from the other one as shown below.

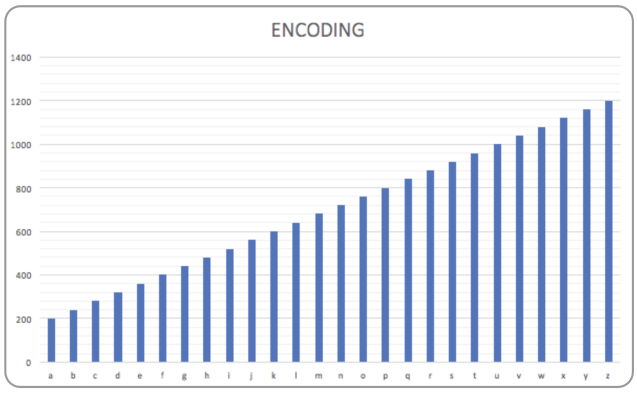

Fig. 1. This range of spectrum is divided in 26 distinct band of frequencies each frequency being an increment of 40Hz from the other one as shown.

VI. Decoding

$PyAudio$[3] was used to establish an input audio stream, using a buffer size of 2048 samples. frequency space transforms. The receiving hardware in our experimental setup is a MAC book pro laptop device. A python based script is configured to listen on the microphone. The received signals are then fed into an FFT algorithm to generate the pitch of the sound. This pitch is then scaled by a factor of 8 and a perceived frequency value is obtained. The perceived frequency is then passed into a pre-constructed lookup table to decipher the character being received. To improve upon the error rate and false positive detection we have considered several samples and the decision algorithm matches it to the nearest matching symbol.

We determined thresholds for which signal was present during each transmission window. As the power of each signal rose dramatically when that signal was on, the threshold was generally highly effective; however, noisy environments would sometimes cause the recorded signal at high frequencies to partially drop out.

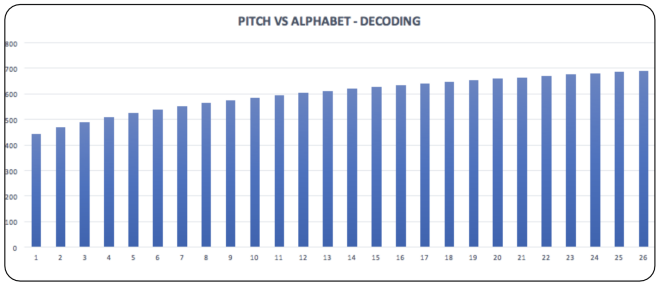

Fig. 2. This figure illustrates the range of perceived frequencies

A. Auto Correction

Since we have used the audible range for the prototype, there is still a possibility of symbol corruption. The fact that US English words are used for the communication can be utilized to have a model to predict the exact dictionary word it matches to. Here, each character is stored in a temporary buffer until a space is encountered. When a new word is generated, a new thread is launched to process the word generated. The generated word is passed into a python based Markov API which takes in the word and predicts a set of nearly matching word from the dictionary if the given word is not exactly matched. Currently we are using the word with the highest match length as a parameter to decide on which word to fetch.

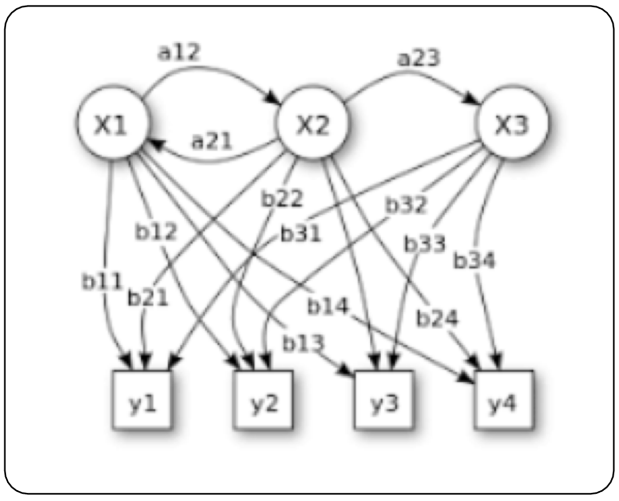

Fig. 3. Markov model based auto correction

VII. SoundBot

Once the receiver was able to detect the message generated by the source with some decent accurancy, we attempted to explore the possiblities of using the post processed text to

design additional usecases. We were intrigued by the possiblity of using the data received to design a bot that could take intelligent decisions depending on the type of data that it detected. Ex. During the event of a fire alarm, what is the series of actions that it must take, where each could depend on the circumstance (time of day) of the event and so on. Using statistical data analysis methods, the bot would determine the best possible course of action and implement it and attempts to verify if the solution succeeded or to retry again with a different possiblity.

A. Case Based Reasoning

The inspiration for our Soundbot is $ELIZA$[5] natural language conversion program described by *Joseph Weizenbaum* in January 1966 which features the dialog between a human user and a computer program representing a mock Rogerian psychotherapist. The aim of a bot is to take real life decisions that is almost as indistinguishable from humans. Most programs tackle this problem with a form of case based reasoning. CBR is the process of solving new problems based on solutions of similar past problems. There are many varieties to CBR as to determine which cases are similar and how it is stored etc.

B. Training the bot

The aim of the training phase is to build a context table which is used as an input to a scoring function that uses pattern matching to attempt to arrive at an ideal solution to the situation. The training phase starts by initiating a conversation between a consumer of the application and the bot. This conversation is used to build a conditional frequency distribution table. This enables the bot to know about the user and take better descisions.

Some of the questions are listed below.

1) Where do you work?
2) Where is your office located?
3) How many rooms are there in the house?
4) What is the phone number of the nearest fire dept?
5) What is the phone number of your neighbour?
6) Describe the sounds of the microwave?

C. Building the CFD Table

The SoundBot is configured with pre- defined workflows to repond to a particular situation based on the noise that was detected. For example, a very simple scenario would be to send an e-mail, when a fire alarm is detected. But there is a possiblity that the user does not respond. Our workflow uses decision trees to identify the right strategy to respond to the situation.

This workflow is augmented with the data from the conditional frequency distribution table to construct an optimal response to the situation.

We used an instance of ConditionalFreqDist from NLTK-Lite to store the training data in a useful format.

The following describes the occurrence of a particular event, and how our model generates a series of possible responses.

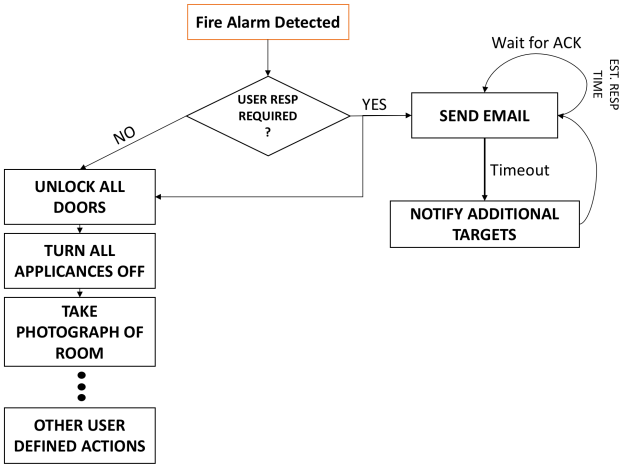

Fig. 4. Fire alarm detection algorithm and automatic decision making.

Tokenize(The fire alarm was triggered before 0 minutes) = ['fire','alarm','triggered','before','0','minutes']

The following table describes a scoring function containing the set of all possible responses.

CFD[fire] = [(boolean : notify user,$\frac{2}{6}$),('boolean : Take pictures',$\frac{1}{6}$), ('Blob : detect approx location',$\frac{1}{6}$), ('boolean : notify fire station',$\frac{2}{6}$),('boolean : unlock all doors',$\frac{2}{6}$)]

CFD[triggered] = [(int : what type of alarm,$\frac{1}{2}$),('boolean: notify User',$\frac{1}{2}$)]

CFD['0 minutes'] = [(boolean : is the event happening now,$\frac{1}{2}$), (timestamp : what is the time of the day,$\frac{1}{2}$), (timestamp : Where is the user,$\frac{1}{2}$),(boolean : did the user respond,$\frac{1}{2}$)]

One possible ordering of events based on the scoring table described above is to continously check for the occurance of the event (fire alarm) and notify the user of the occurrence and validity of the event(fire alarm). In absense of acknowledgement from the user, notify additional targets. Take pictures and notify user.

VIII. Email Server

We built an e-mail client application using python to send and receive e-mail notifications. This piece of code uses Google's SMTP Server to perform this task. The main purpose of this client was to notify the user of the occurrence of a new event and wait for acknowledgements. At the current state of this project, e-mail is the only form of remote communication method that we have employed. We further explored on the possiblity of using Programmable SMS API to communicate.

IX. Evaluation

We evaluated our endoder and decoder using a command-line application which takes input and prints the transmitted data. One assumption that we made while we evaluated this

model is that the transmitted data is nothing but only proper English words. This assumption enabled us to easily apply error-counting algorithm, such as Levenstein's distance, which we have employed here.

We were able to transmit data with predictable accuracy in a noiseless environment and low bit rate conditions. Some of the interesting observations was that the continous ambient noise, like people talking did not cause a significant reliablity issue. This is because most of these sounds occur in a frequency lower than our frequency range. However sharp noises like knock of the door caused far greater disruption.

The following breifly summarizes our experimental setup used for evaluating our model.

1) At $0ft$ in a noise less surrounding.
2) At $10ft$ in a noise less surrounding.
3) At $0ft$ in a surrounding with people talking.
4) At $10ft$ in a surrounding with people talking.

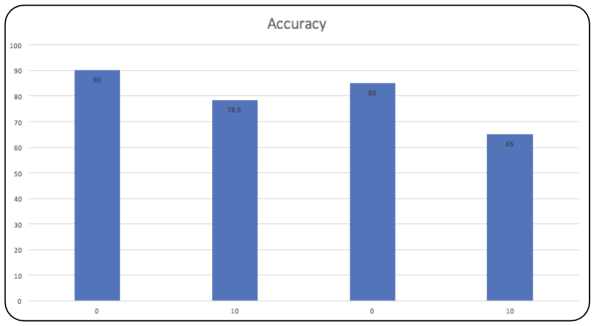

Fig. 5. Accuracy of the Chatbot in noisy and quite environment

X. Constraints

Humans can typically hear sounds roughly in the range of 20Hz to 20kHz and frequencies over about 19.5kHz is generally inaudible thereby leaving very little bandwidth for data transmission using ultrasonic sound. Also, the sampling rate of the microphone provides a hard upper bound to the frequencies we can accurately detect. The hardware used in our Macbook Pro will support sampling rates by up to 48kHz, by the Nyquist-Shannon Sampling Theorem this means that the highest frequency signal that can be perfectly reconstructed without aliasing is a little over 22kHz. As it turns out, this hard upper bound is one of the limiting factors. So, we have set the present sampling rate as 44100Hz.

We were not able to detect any definitive measurements of the frequency response of either the MacPro's speaker or its microphone, therefore, we experimented with the ability to send and receive signals at high frequencies and met with success until just over 20kHz, where either the speaker or microphone sharply falls off in response.

XI. Future Work

Some of the improvements to our existing implementaion are listed below.

- Presently we have a data throughput of 1 bit/sec which can be improved by using more sophisticated coding schemes like CDMA and by adapting multiple frequency transmission protocols like Dual-Tone Multi-Frequency signalling.
- Move the transmission band to Ultrasonic range to improve upon the error rate which would require testing of our code on alternative hardware than from what we used in our evaluation.
- Using a better prediction model to improve the decoding mechanism.
- Use a more secure protocol for communication.
- Expand the decoding hardware from its current form of MacOS/Windows to mobile phones to understand how well they can detect Ultrasonic signals.

XII. Android Applications

Two Android applications were developed as part of this project.

A. Acoustic Data Sender

This application accepts the text data from the user. It then converts each of the English characters to the pre-fixed frequencies assigned to them. Ex: Character 'a' is assigned a frequency of 200Hz, 'b' is assigned a frequency of 240Hz. It then transmits the text data as sound.

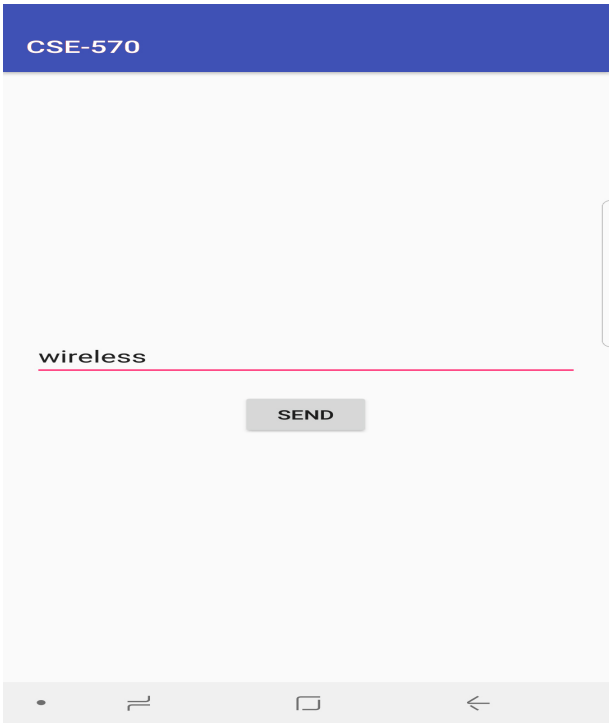

Fig. 6. Screenshot of the Android Data Sender application.

B. Alarm Generator

This application generates an Alarm sound upon pressing the Alarm Button. This application was used to aid the testing of our Fire Alarm Detector.

XIII. Desktop Transmitter-Receiver

The transmitter receives the input text message from the user via the command line interface. It then converts each character in the text to a sound wave of a particular frequency. The receiver receives the acoustic signal and based on the frequencies of the samples it collected, decodes it to reconstruct the original text message.

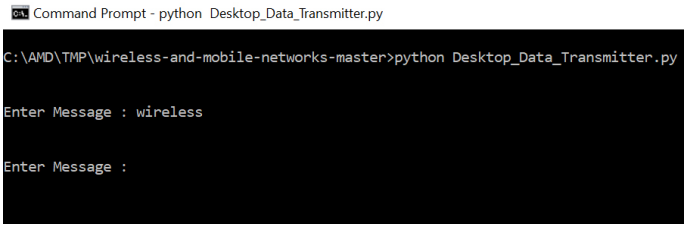

Fig. 7. Screenshot of the Desktop Data Sender.

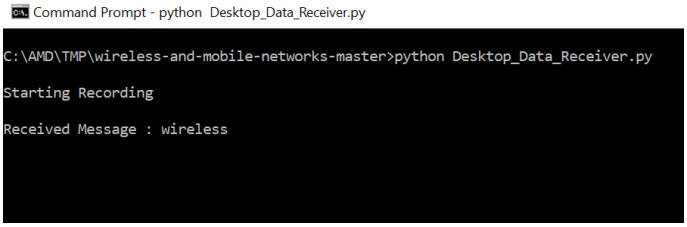

Fig. 8. Screenshot of the Desktop Data Receiver.

A. Email Client

The following is an outcome of the fire-alarm use case implementation that sends out e-mail in the event of a fire alarm.

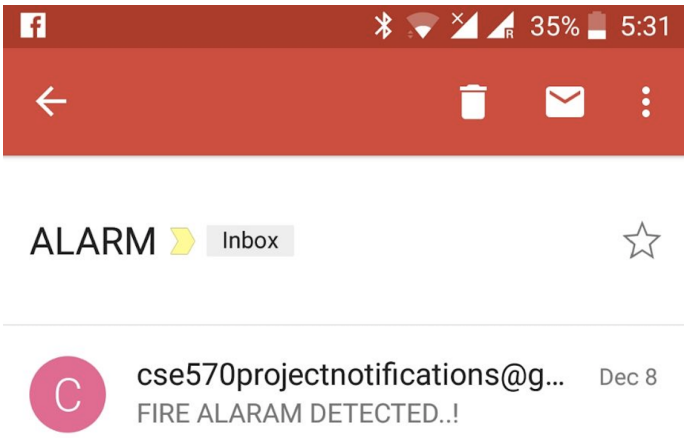

Fig. 9. Screenshot of email notification

XIV. Individual Contributions

Our work predominantly involved lot of discussions amongst ourselves. We made sure each had almost equal workload towards completing this project.

The following table describes a rough estimate of the contributions made by each individual.

Arjun Mathew Dan	33%	a) Created the initial Text transmitter-receiver b) Audio signal detection handling c) Created the Android Applications
Mohan Gandhi Alapati	33%	a) Fire Alarm use-case implementation b) Auto-correction of received text c)Markov model based classification
Prashanth Soundarapandian	33%	a) SoundBot and related works b) Mail server handling

TABLE I. Individual Contributions Chart

Acknowledgment

We sincerely thank Prof. Jie Gao for her valuble feedback in shaping this project.

References

[1] Harvest. Zhang and Bonnie. Eisenman, *SqueakyChat: Ultrasonic communication using commercial notebook computers*

[2] E. W. Weisstein. *Hamming Function.* From MathWorld–A Wolfram Web Resource. http: //mathworld.wolfram.com/HammingFunction.html.

[3] *PyAudio* http://people.csail.mit.edu/hubert/pyaudio/.

[4] G. Goertzel. *An algorithm for the evaluation of finite trigonometric series*. The American Mathematical Monthly, 65(1):pp. 3435, 1958.

[5] http://www.manifestation.com/neurotoys/eliza.php3

[6] Chantarotwong. Bonnie *The Learning Chatbot*, IMS-256 Final Project, Fall 2006

[7] *GitHub: quietnet*. https://github.com/Katee/quietnet.