Audio Data Transfer Over Commodity Embedded Devices

Cole Feely (cfeely@umass.edu), Aidan Murray (atmurray@umass.edu),
Owen Lheron (olheron@umass.edu), Sashank Rao (sashankrao@umass.edu)

Electrical and Computer Engineering, University of Massachusetts Amherst

***Project Repository**: https://github.com/ColeFeely6/Project-Chime*

Abstract

In this paper, we present a system for transferring data over audio sound using off-the-shelf hardware such as the Arduino Uno and an electret microphone breakout. Our system allows for the secure and reliable transmission of data over short distances, making it suitable for use in scenarios where internet connectivity is limited or unreliable.

To demonstrate the capabilities of our system, we conducted a series of experiments to transmit data using different frequencies and modulation schemes. Our results show that our system is capable of accurately transmitting data over a range of frequencies and with a low data rate of 0.5 bits per second.

Overall, our system represents a low-cost and reliable alternative for transmitting data in a variety of settings where traditional communication methods may be unavailable or insecure.

1 Introduction

Traditionally, Commercial Off-The-Shelf (COTS) sensors (microphones and speakers) have lower sampling rates in contrast to expensive alternatives with custom front-end interfaces and high sampling rates. Google had previously created Google Tone that shares data between personal computers through sound, but this again requires devices with high-end hardware. Our goal was to embed information over sound like google chime but by using COTS hardware and to optimize signal demodulation and decoding at the receiver.

2 Motivation

The motivation for this project stems from the need for secure and reliable communication in various settings. In today's digital age, nearly all transmission of data is performed over the internet. However, internet connectivity is not always available or secure, particularly in settings such as military operations, disaster response, or rural areas. Additionally, the use of radio frequencies for communication is often restricted or monitored, making it difficult to transmit sensitive information. In these scenarios, alternative methods of communication are necessary.

One such alternative is the use of sound waves to transmit data. Sound waves can be used to transmit information in a secure and reliable manner, as they are difficult to intercept and can be transmitted over short distances without the need for specialized equipment. In this project, we have developed a system for transferring data over audio sound using off-the-shelf hardware. Our system allows for the transmission of data in a secure and reliable manner, making it suitable for use in a variety of settings where internet connectivity is limited or unreliable.

3 Problem Statement

The transmission of data over the internet has become the norm in today's digital age. However, there are scenarios where internet connectivity is not available or is unreliable, such as in military operations, disaster response, or rural areas. Additionally, the use of radio frequencies for communication is often restricted or monitored, making it difficult to transmit sensitive information.

In these scenarios, alternative methods of communication are necessary. One such alternative is the use of sound waves to transmit data. Sound waves can be used to transmit information in a secure and reliable manner, as they are difficult to intercept and can be transmitted over short distances without the need for specialized equipment.

4 System Specifications

1. System shall use off-the-shelf hardware supplied from either class or the Makerspace
2. System shall transfer data with background noise
3. System shall be able to transfer data over varying distances
4. System shall be able to transfer data over different angles between the receiver and speaker
5. System shall be able to transfer data with an obstacle in between the microphone and the speaker

5 Design Alternatives

Many alternative designs exist for improving the usability of audio data transfer. Below we evaluate two alternatives to our use of audio data transfer.

5.1 Google Tone

Google Tone is a Chrome extension developed by Google that allows users to share web pages with others by transmitting a URL over audio using a microphone and speakers. When the extension is enabled, users can click the Google Tone button in their browser to transmit a URL to nearby computers that also have the extension installed. The URL is transmitted as a series of tones, which are then received by the other computers and translated back into the original URL.

Google Tone is designed to make it easy to share web pages with others in close proximity, without the need for a shared network or other types of connectivity. It is particularly useful for sharing web pages in a group setting, such as during a meeting or presentation. To use Google Tone, both the transmitting and receiving computers must have the extension installed on their Chrome browser and have their microphones and speakers enabled. When a user clicks the Google Tone button, the extension generates a series of tones based on the URL of the current web page. These tones are then transmitted via the microphone and can be received by other computers in the same location that have the extension installed and their microphones and speakers enabled. The receiving computers decode the tones and open the corresponding web page in their browser. Overall, Google Tone provides a convenient way to share web pages with others in a group setting, and has the potential to be used in a variety of other applications where the quick and easy sharing of information is desired.

5.2 Universal Timestamping with Ambient Sensing

This paper presents a new design for coordinating functionality among Internet of Things (IoT) devices that leverages Electric Network Frequency (ENF) fluctuations as a global time reference for sensing devices. The design allows IoT devices to harvest timing signals using off-the-shelf sensing capabilities, resulting in a universal sense of time without the need for custom hardware or network dependability. The design is resource efficient and requires no modifications to existing devices, and is evaluated for its extensibility in remote setups and robustness in real-world settings.

5.3 Ultrasound

Sonarax uses ultrasonic data transfer technology to transmit data wirelessly over short distances using high-frequency sound waves. This technology relies on the use of transducers, one to transmit the data and one to receive it, to send and receive data at frequencies beyond the range of human hearing. The transducers work by converting electrical signals into ultrasonic waves, which can then be transmitted through the air or other media. Upon receiving the ultrasonic waves, the receiving transducer converts them back into electrical signals, allowing the data to be received and processed by the device.

Sonarax's ultrasonic data transfer technology is designed to be low power and low cost, and is capable of transmitting data at speeds of up to 10 Mbps. The company has developed a range of products based on this technology, including a small, low-cost module that can be integrated into a variety of devices and systems. Sonarax has also demonstrated the use of its ultrasonic data transfer technology for applications such as wireless charging, wireless sensor networks, and smart home automation.

5.3 Waver App

The Waver application was developed by Georgi Gerganov, TJ Porter and their team to transfer data and messages between two phones using sound with the intent that the project could be applied in severless one-to-many file transfer, Internet-of-Things applications, Audio QR-codes, device pairing or authentication. The application, built off of the Node.js framework, implements a simple Frequency Shift Keying (FSK) based transmission protocol and a bandwidth rate between 8-16 bytes/sec. The app additionally uses Error Correction Codes (ECC) to improve demodulation robustness. The Waver app is an independently developed application available on the Apple App Store, Google Play Store and

Snap Store. At certain points in the documentation, the developers demonstrate data transfer over commodity devices when they plugged their phone running the application into an Arduino interface so that the data transmitted by the other Arduino with a speaker could be picked up by an Arduino microphone.

6 Solution

We propose a much simpler method for transferring data over sound than those methods discussed in the previous section. For our project, we will refrain from using much simpler software than such things like internet protocols to help data transfer like that demonstrated in Google Chime and significantly simpler hardware than Raspberry Pi 4's, Photon MCU, ENF sampler and more used in Universal Timestamping with Ambient Sensing. By doing so, we aim to demonstrate that audio data transfer can be implemented using commodity devices, simple software libraries and filters built with basic materials.

We will use a basic electret microphone breakout board connected to an Arduino Uno board so that we can use basic Fast Fourier transform Arduino Libraries to determine the frequency of sound being transmitted from the speaker. In this code, we will implement filters to eliminate background noise picked up by the microphone. Additionally, we will present the hardware filters we also created for this project, although we found that the software filters were more effective.

7 Software Setup

7.1 Fast Fourier Transform

In order to separate different frequencies from each other, we used a Fast Fourier Transform Arduino library. This mathematical model allows us to represent functions as a sum of sinusoidal functions. With this information we were able to see which frequency had the highest amplitude, which is the frequency emitted by the sender. Understanding how Fast Fourier Transforms work is much easier if you begin at a Discrete Fourier Transform, which computes the same result. Although Fast Fourier Transforms use a much more efficient algorithm. The Discrete Fourier Transform works by periodically recording the magnitude of a signal, then transforms those values into an array of complex numbers. This array represents the frequencies within the signal and their magnitudes. Complex numbers are used because Euler's numbers connect sinusoids to imaginary numbers using the following formula.

$$e^{j\omega_m t_n} = \cos(\omega_m t_n) + j\sin(\omega_m t_n).$$

This means that the frequencies and magnitudes can be represented as an array of complex numbers

$$X[m] = \sum_{n=0}^{N-1} x[n]s_m[n]$$

X[m] is the magnitude, and x[n] is the frequency
From this array we can find which frequencies have the highest amplitude. However we still need to keep two things in mind to avoid getting false results from the Fourier Transform equations. First, we need to be aware of aliasing, which is when our sampling rate is not high enough that it appears signals are present which are not truly there. This can be avoided by making sure we are sampling at more than twice the highest frequency we are trying to record and disregarding anything past this, which is known as Nyquist Theorem. In this project, the analog to digital converter within the Arduino Uno we are using has a refresh rate of 2048, so we can only measure sounds up to 1000 Hz. Another problem that arises when implementing a Fourier Transform is that the formula only works when the finite time period sampled exactly matches a multiple of the periods of each frequency, which is extremely unlikely. To combat this we use a technique called windowing. Windowing reduces the amplitude of the signal being measured towards the ends, start and beginning, of the signal. This results in the middle of the period being measured weighing more, reducing the effect of frequencies being cut off at each end.

7.2 Decoding Frequency

The software that runs our Arduino Uno revolves around using Fast Fourier Transforms to find which frequency is the loudest. Specifically, it records 128 samples over about one twentieth of a second. Then, these samples are used to compute the Fast Fourier Transform of the data set, while using a preset hamming window. This results in an array of frequencies and their respective magnitudes. Next, the code will count how many times the same frequency is read as the strongest signal in a row, with a margin of error of three percent. If the same frequency is read many times consecutively that frequency is decoded to its corresponding number. This is because even if we weren't outputting a number from our python script, there will still be some predominant frequency just from background noise, speaking, ect.

7.3 Audio Emitter

We emit our code through different frequencies between 550 Hz and 1000 Hz which correspond to different digits. This allows us to have each digit 50 Hz away from each other, which makes it very easy for the hardware to accurately output and detect even though they are relatively simple and cheap. Although in general higher frequencies tend to function better, we could not choose

any frequencies higher than 1000 Hz because the analog to digital converter only operates at 2048 Hz as previously mentioned in 7.1. The python program itself is extremely simple consisting of two libraries, time and winsound, which can output any chosen frequency. We request an input code from a user, then we iterate through that string and output the corresponding frequency to each digit.

8 Hardware Setup

Sticking to our off-the-shelf design, our hardware set-up consisted of an Sparkfun Electret microphone breakout which sent a signal through the low-pass filter and into an input port on the Arduino Uno as seen in the following diagram:

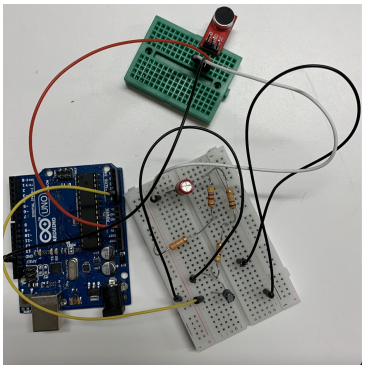

Diagram 1

8.1 Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328 microcontroller. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It is designed to be used as a simple computer for controlling electronics projects. The board can be programmed using the Arduino Integrated Development Environment (IDE) and the C programming language. It is widely used in DIY electronics projects, educational settings, and in industry. The Arduino Uno is an open source hardware platform, meaning that the hardware design and schematics are freely available for anyone to use and modify. We took advantage of this open source aspect and used Fast Fourier Transform libraries made for the board. The board took input from the microphone connected to one of its input ports and manipulated that data using our algorithm.

8.2 Microphone

The SparkFun Electret Microphone Breakout is a small, easy-to-use microphone designed for use with a wide range of electronics projects. It consists of an electret microphone element, a small amplifier circuit, and a 3.5mm audio jack. The microphone element is a type of microphone that uses a static charge to produce an electrical current, which is then amplified by the amplifier circuit and output through the audio jack. The microphone has a wide frequency response, making it suitable for a variety of applications including voice recognition, sound sensing, and audio recording. We used this powerful off the shelf microphone to pick-up sound from our laptop speaker and send that information either through the low-pass filter and then to the input port on the Arduino, or to the Arduino board directly.

8.2 Band-Pass Filter

While working on interpreting the signal outputted by our microphone, we constructed a bandpass filter to reduce noise. Originally we planned on using Piano Keys as our input frequencies to represent each digit, so I used 20 Hz and 4200 Hz as the low and high cutoffs. This resulted in us using 100nF and 1nF capacitors, and 80kOhm and 38kOhm resistors to reach those specifications. However, eventually we phased out the bandpass filter because upon testing it wasn't needed. This was likely due to us only using frequencies between 550 and 1k Hz. This means that we already filtered out frequencies above 1k Hz due to the Nyquist Theorem (our board's analog to digital converter only sampled at 2048 Hz). Also, frequencies below 20 Hz are likely low strength and are unlikely to interfere with the higher frequency signals we are looking for.

9 Evaluation

To evaluate our system our methodology was to output a series of ten different audio signals from our speaker and to determine the accuracy in which the receiver properly identifies the transmitted signal. Using our Python script we were able to output ten different audio signals within the range of [550, 1000] with intervals varying by 50hz. Each signal was transmitted for two seconds and the next signal was sent immediately after the other. On the receiver side, we would print the demodulated signal to the serial monitor if the frequency was determined. To determine the accuracy of the system, we would look at the serial monitor after transmitting the full audio signal and determine if a) there were ten digits printed on the serial monitor and b) if the frequency matched that of the transmitted signal.

We then determined a comfortable output volume level from our computer that would allow for data to be properly received by the microcontroller. Keeping the receiver directly in front of the computer speakers, we varied our computer volume level and tracked the accuracy in identifying the output frequency.

Using a Dell XPS 13 9305 computer, we determined that operating at over 90% volume would be ideal for our implementation while within 6 inches of the computer

speakers. However, at further distances the ability for the receiver to accurately detect which frequencies were being transmitted drastically decreased as highlighted by table 1. Therefore, in order to further test our system, we compared the accuracy of the receiver using an off the shelf speaker, the JBL Flip 4.

9.1 Data Transfer over Distance

To test the resiliency of distance based variations in our signal demodulation techniques we tested the accuracy of our receiver at various distances from the speaker. For distance based variations, we observed zero incorrectly identified frequencies (e.g. detecting 400 Hz rather than 550 Hz). Rather we only noticed that at a certain distance away from the speaker the ability for the receiver to pick up on the signal would diminish. Therefore, it will not meet our system minimum number of frequency observations in order to confidently make a determination of the signal. So for these signals, the serial would print nothing to the screen indicating that it did not detect anything.

We first used our computer speakers and tested the receiver at five different locations from the speaker. We tested this multiple times and took the mean value of the results. It is evident that with the computer speakers we need to be operating within 6 inches of the speakers if we want to have accurate results.

Distance (in)	Accuracy (%)
0	100
3	100
6	85
9	43
12	20
15	0

Table 1

Using the JBL Flip 4, we get much more range for our data transfer. We were able to get a 90% accuracy rating over 3 feet away from the speaker.

Distance (ft)	Accuracy (%)
0	100
1	100
2	100
3	90
4	85
5	42

Table 2

9.2 Data Transfer with Angles

Testing our system at various different angular positions was our next step in validating our system. Rather than use our computer speakers, we decided to solely evaluate angular changes in our set up using the JBL Flip 4. We tested the receiver at different angles in front of the JBL Flip 4 to determine the influence that the position had on accuracy of the receiver. The effect of the angular position of the receiver in comparison to the speaker was negligible within a two foot radius, proving to cause no change in the accuracy ratings of the receiver. This could mainly be due to the fact that the JBL Flip 4 does not project sound directly out from the speaker but instead it emits noise in a 180 degrees from the frontwards facing speakers.

We then tested placing the receiver behind the speaker and we noticed a substantial difference in the accuracy levels of the receiver. In contrast to the results in Table 2 at 1 foot directly behind the speaker the receiver only has a 50% accuracy rating, with a 0% accuracy rating at 2 feet away from the speaker.

9.3 Data Transfer with Noise and Obstacles

Lastly we tested our system using obstacles placed in front of the speaker as well as external noise from human speech and other audio sources. In our experimentation, we placed the receiver one foot away as this distance granted us 100% accuracy ratings in our previous trials. We observed that human speech had negligible effect on the accuracy of the system, as human speech is typically below the minimum frequency that we output (550 Hz). At normal conversational volume, we observed zero inaccuracies caused by the external noise.

To test for the influence other audio signals would have on the system, we played music over the signal and observed the effect. Compared to the distance based variations, this is the first test that we tried that actually caused inaccurate frequencies to be identified by the system. We saw at least 10% of digits be incorrectly identified while music was playing over the audio signal from our computer speakers from 6 inches away.

Obstacles placed in front and around the receiver had varying effects on the accuracy. We used acoustic sound foam and placed it in various locations in front of the receiver to observe the effects. Distances greater than within 1 inch of the system had negligible effect of the accuracy, however at very close distances we observed the foam tile could influence accuracy by up to 50%.

Lastly, we determined if the system would identify any frequencies when there were no transmitted signals. We left the system searching for signals for a 2 minute period and observed that there were 5 occurrences of the system detecting a 600 Hz signal when nothing was transmitted. With a transmission rate of 1 bit per 2 seconds, this accounted for an 8% error. We believe this had to do with the harmonics of the room that we were testing in, due to the fact that there was external noise being emitted from nearby computers and lights.

10 Discussion

As the results show, the system's accuracy decreases exponentially as the distance from the mic increases from the Dell laptop. However, a 100% accuracy can still be achieved from these close distances. This proves to be a possible avenue for an audio data transfer between two devices at up to 6 inches. Our analysis of the system is also shown through our evaluation as we changed everything that we could in the environment to test the accuracy of the system.

Although our system was able to achieve a perfect accuracy, there were certain limitations we encountered with the mic that couldn't allow us to read more frequencies and allow us to conduct a real time detection of audio data transfer. As shown in the results, the distance in achieving a perfect accuracy for this mic is nowhere past 6 inches, further explaining why a better off-the-shelf mic could be used to complete this system. Not only would a better mic help the system but it could also contribute to better the denoising of the system.

One limitation was the fact that the analog to digital converter could only operate at a 2048 Hz frequency. This means that we could only audibly detect frequencies of up to 1000 Hz. If an ultrasonic mic could be implemented, the system could process higher frequencies. Higher frequencies process much less noise compared to lower frequencies. This could also be the case for when we process noise well above what human ears can process (around 20 KHz).

These limitations also bring about some trade offs which show in our denoising techniques for obtaining perfect accuracy. One denoising technique used was checking for a frequency between 550 - 1000Hz. This allowed us to remove other frequency noise from interfering with the reading. This works the same with the interval based audio emission as it removes chances for the mic to receive noise.

11 Future Steps

Because we proved that data transfer can be performed using very low cost devices, it would be interesting to see our device built with just an Atmega328P chip, external clock, low pass filter, external battery, voltage regulator and speaker on a PCB with two copies so that the two devices can send data back and forth, communicating with one another.

Another thing we would like to see is if we had a one-to-many broadcast, where files could be shared across many devices at once.

Our data transmission did not have any mechanisms to detect if there is a transmission error from noise. In the future, it would be interesting to add error detection into the transmission.

On that note, we did not incorporate any security into our set-up, in the future it would be another aspect that would be interesting and critical to explore.

12 Conclusion

In conclusion, the results of our study demonstrate the feasibility of using commodity embedded devices for audio data transfer over various distances. We were able to successfully transmit data at distances of up to 46 centimeters using inexpensive, off-the-shelf hardware.

Our results suggest that this technique has potential applications in a variety of settings, including data transfer in low-infrastructure environments and situations where traditional wireless communication is not possible or practical. Future work could explore the use of more advanced audio encoding techniques, as well as the potential for adapting this method for use in other types of embedded devices.

Overall, our study highlights the potential of using commodity embedded devices for audio data transfer and underscores the importance of exploring alternative communication methods in the face of evolving technology and changing communication needs.

References

[1] Kaufmann, Alex and Smus, Boris. “Tone: An experimental Chrome extension for instant sharing over audio” Google Research. May 19, 2015. https://ai.googleblog.com/2015/05/tone-experimental-chrome-extension-
for.html Accessed October 30, 2022.

[2] Anwar, Fatima Muhammad, et al. “Universal Timestamping with Ambient Sensing.” IEEE Xplore, IEEE,2017, ieeexplore.ieee.org/document/9918555/authors#authors.

[3]“Technology”Sonarax.https://www.sonarax.com/technology Accessed December 16, 2022.

[4] Gerganov, Georgi, Porter, TJ. “ggwave” GitHub. Accessed December 16, 2022. https://github.com/ggerganov/ggwave/