ENCS4310 (DSP) Course Project

Zainab Jaradat[1], Manar Shawahni[2], Ro'a Nafi[3]

Department of Electrical and Computer Engineering, Birzeit University[1]

1201766@student.birzeit.edu [1], 1201068@student.birzeit.edu [2] , 1201959@student.birzeit.edu [3]

Abstract

This project explores digital signal processing (DSP) techniques for encoding and decoding text into audio signals. Utilizing a Python-based graphical interface, the system translates English alphabet characters into unique voice-frequency combinations and employs Fast Fourier Transform (FFT) and bandpass filtering for decoding. The study assesses the accuracy and efficiency of these methods, revealing that while both are effective, bandpass filtering is slower. This finding underscores the importance of algorithm selection in DSP, particularly for real-time applications below.

Index Terms: Signal Encoding and Decoding, FFT, Bandpass Filtering.

1. Introduction

In the realm of digital signal processing (DSP), the conversion of text to audio signals presents intriguing possibilities for secure data transmission. This project develops a system that encodes textual information into audio signals using specific frequency mappings for each character, and decodes these signals back into text. The system, accessible through a Python-based graphical interface, leverages Fast Fourier Transform (FFT) and bandpass filtering for decoding. This study aims to evaluate the efficacy and efficiency of these DSP techniques, contributing to their application in secure communication and data processing [1].

2. Problem Specification

In this project, the primary problem is the effective encoding of textual information into audio signals and its subsequent decoding back to text. This technical challenge can be further broken down into several key aspects:

2.1. Encoding Text to Audio Signals

The first half of the task is to map each character in the English alphabet to a distinct set of voice-frequency components. This necessitates developing a frequency mapping technique in which each character is represented by a unique signal. The problem is to ensure that these mappings are not only unique, but also strong enough to resist signal processing procedures and any noise interference during transmission [2].

2.2. Signal Generation and Transmission

Once the mapping is established, the system must generate audio signals corresponding to the input text. This involves creating a synthesis algorithm that can produce clear, distinct sounds for each character. The integrity of these signals during transmission or storage also poses a significant challenge, as any distortion can lead to inaccuracies in decoding.

2.3. Decoding Audio Signals to Text

The decoding process involves analyzing the received audio signal to identify the frequency components and map them back to the corresponding characters. This task is complicated by the need to accurately distinguish between frequencies, especially when they are close in range or overlapped [3].

2.4. Implementing Fast Fourier Transform (FFT) and Bandpass Filtering

The use of FFT and bandpass filtering for decoding the audio signals is central to this project. FFT is employed to transform the time-domain signal into its frequency-domain representation, making it easier to identify the constituent frequencies. Bandpass filtering, on the other hand, is used to isolate specific frequency bands corresponding to each character. The efficiency and accuracy of these methods in extracting the correct text from the audio signal are crucial metrics for evaluation [4] [5].

By addressing these challenges, the project aims to develop a reliable and efficient method for secure communication through the encoding and decoding of text as audio signals, using advanced DSP techniques.

3. Data

For this project, the type of signals used to develop and evaluate the encoding and decoding system was Synthetic Signals, these are artificially created signals using the Python. The code generates audio signals for each English alphabet character based on predefined frequency mappings. These signals are synthesized using simple **sinusoidal** functions to represent each character's unique frequency combination. The synthetic signals allow for controlled testing conditions, ensuring that the system's functionality can be validated in an ideal environment without external noise or interference.

4. Evaluation Criteria

The performance of the encoding and decoding system will be evaluated using the following criteria:

4.1. Accuracy of Decoding:

The primary metric for evaluation is the accuracy with which the system can decode the encoded signals back into text. This will be quantified by comparing the decoded output with the original input text calculating the percentage of correctly decoded characters.

4.2. Processing Time

The time taken to encode and decode signals is critical, especially for real-time applications. The efficiency of both FFT and bandpass filtering methods will be compared in terms of their processing speed.

4.3. Comparative Analysis of FFT and Bandpass Filtering

The effectiveness of FFT and bandpass filtering in decoding will be compared. This includes analyzing the trade-offs between the speed of decoding and the accuracy for each method.

5. Approach

The problem of encoding and decoding text into audio signals was approached by developing a system that could convert each character in the English alphabet into a unique set of voice-frequency components. This was achieved by creating a frequency mapping technique where each character was represented by a unique signal.

The first part of the solution involved encoding text into audio signals. A Python function was written to take a text string as input and convert each character into a signal with three frequency components. These signals were then concatenated to form the output signal.

The second part of the solution involved decoding the audio signals back into text. Two different methods were used for this: Fast Fourier Transform (FFT) and bandpass filtering. For both methods, the input signal was divided into segments corresponding to each character.

In the FFT method, the discrete Fourier transform of the signal segment was computed, and the magnitude spectrum was plotted to identify the constituent frequencies. A threshold was used to filter out the noise and a tolerance range was used to match the peaks with the closest frequencies in the frequency map.

In the bandpass filtering method, a series of bandpass filters were designed and applied, each covering one of the frequency ranges in the frequency map. The root mean square (RMS) of the filtered signal was computed, and if the RMS was above a threshold, it was assumed that the signal segment contained the frequency of the filter.

Finally, a graphical user interface (GUI) was developed using Tkinter to allow users to encode text into audio signals, save the audio signals, and decode them using either FFT or bandpass filtering, then Compare Decoding Methods. The GUI also provided a text area for displaying the decoded text.

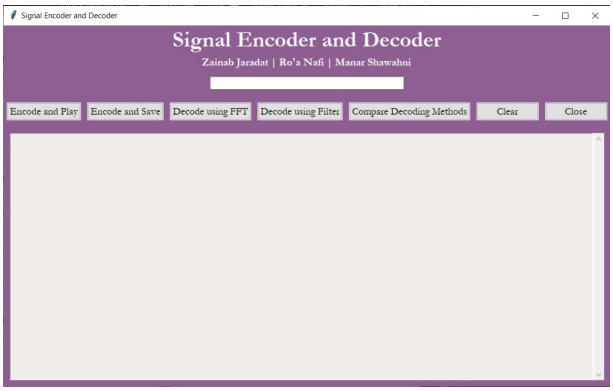

Figure 1:*GUI*

6. Results and Analysis

6.1. Test Cases

Here are Some Test Cases that we have tried on our program :

1. "welcome to dsp course"

When clicking at encode and play button , the text encoded to audio signals. Then, to Decode we can use three buttons as photo shown below.

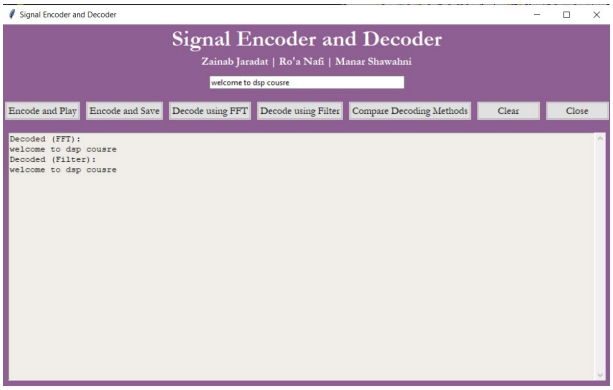

Figure 2 : *Test Case1*

To compare FFT Decoding and Filtering Decoding we used "Compare Decoding Methods " button , and we compare the accuracy and time as figure shown below result:

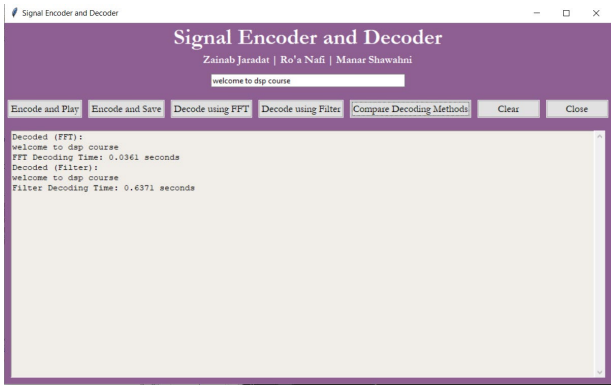

Figure 3 : *Comparing Decoding methods*

```
Decoding accuracy: 100.00%
FFT Decoding Time: 0.0361 seconds
Filter Decoding Time: 0.6371 seconds
```

The accuracy of both decoding methods are 100% . Decoding took 0.0361 seconds using FFT and 0.6371 seconds using bandpass filtering. So, the FFT is faster than BPF Filter in decoding.

2. "zainabZAINAB_12345 67890 !@#$%"

This test case tests the system's response to characters not included in your frequency map.

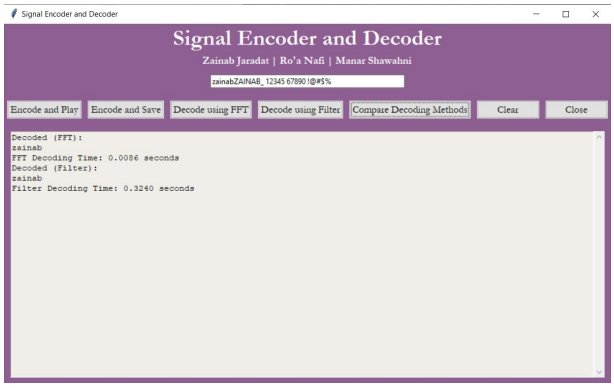

Figure 4 :*Test Case2*

We noticed that our program handle these gracefully by skipping them as shown in Fig 5.

```
Warning: Character 'Z' not in frequency map. Skipping.
Warning: Character 'A' not in frequency map. Skipping.
Warning: Character 'I' not in frequency map. Skipping.
Warning: Character 'N' not in frequency map. Skipping.
Warning: Character 'A' not in frequency map. Skipping.
Warning: Character 'B' not in frequency map. Skipping.
Warning: Character '_' not in frequency map. Skipping.
Warning: Character '1' not in frequency map. Skipping.
Warning: Character '2' not in frequency map. Skipping.
Warning: Character '3' not in frequency map. Skipping.
Warning: Character '4' not in frequency map. Skipping.
Warning: Character '5' not in frequency map. Skipping.
Warning: Character '6' not in frequency map. Skipping.
Warning: Character '7' not in frequency map. Skipping.
Warning: Character '8' not in frequency map. Skipping.
Warning: Character '9' not in frequency map. Skipping.
Warning: Character '0' not in frequency map. Skipping.
Warning: Character '!' not in frequency map. Skipping.
Warning: Character '@' not in frequency map. Skipping.
Warning: Character '#' not in frequency map. Skipping.
Warning: Character '$' not in frequency map. Skipping.
Warning: Character '%' not in frequency map. Skipping.
```

Figure 5 : *Response of non-maping characters*

7. Development

One possible way to improve the system is broadening the Frequency Map since it currently handles a limited set of characters. Expanding the frequency map to include a wider range of characters and symbols would increase the system's utility. Also, we can Optimize the Bandpass Filtering, Since FFT was found to be significantly faster than bandpass filtering, efforts could be made to optimize the bandpass filtering method. This could involve refining the filter design or implementing more efficient filtering algorithms to reduce processing time.

8. Conclusions

From doing this project, we learned how to apply digital signal processing (DSP) techniques to encode and decode text into audio signals. We demonstrated how to use a frequency mapping technique to convert each character in the English alphabet into a unique signal with three frequency components, and how to use Fast Fourier Transform (FFT) and bandpass filtering to decode these signals back into text. We also evaluated the efficacy and efficiency of these decoding methods, and identified their strengths and weaknesses. We found that while both methods were effective, FFT was faster and more accurate than bandpass filtering. We also suggested some possible ways to improve the decoding methods. This project showed us the potential and challenges of using DSP for secure data transmission and processing.

9. References

[1] "sending-data-over-sound-how-and-why," lectronicdesign, [Online]. Available: https://www.electronicdesign.com/markets/automation/article/21808186/sending-data-over-sound-how-and-why.

[2] "know-digital-signal-processing," lisnr, [Online]. Available: https://lisnr.com/resources/blog/know-digital-signal-processing/.

[3] "articles," springeropen, [Online]. Available: https://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-023-00274-x.

[4] "python-scipy-fft," realpython, [Online]. Available: https://realpython.com/python-scipy-fft/.

[5] "applying-digital-filters-to-speech-sounds," ipython, [Online]. Available: https://ipython-books.github.io/116-applying-digital-filters-to-speech-sounds/.