BIRZEIT UNIVERSITY

Faculty of Engineering & Technology Electrical & Computer Engineering Department

Digital Signal Processing– ENCS4310

Maha Mali *1200746*

Hazar Michael *1201838*

Afaf Amwas *1203359*

1. Introduction

This Digital signal processing project focuses on encoding a written English text into its corresponding frequency components by using the low, middle and high frequency components of each lower-case English alphabet and generating an audio signal that the user can hear or save as an audio file. This process allows us to distinguish between different letters as each alphabet has its unique frequency. Moreover, the second phase explores the decoding of frequency signals back into its corresponding written English format. The decoding is done using two methods, which are frequency analysis (Fast Fourier transform) and bandpass Filters. This project provides an interesting educational system to learn about digital signal processing using linguistics by translating English text to audio sound and vice versa. The figure below shows the all system that we will do it.

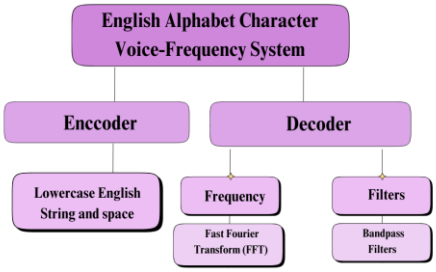

Figure 1: *System Description*

2. Problem Specification

The goal of this project is to create an encoding system that maps each English alphabet into its corresponding frequency and generates its audio signal. Further, two decoding mechanisms are created to decode the audio signal back into its text form.

3. Data

This project was designed to take English text as input from the user in order to encode it to its corresponding audio frequency. The user is only allowed to input lowercase English alphabets; thus, any number or special characters like colons, semicolons, commas, and brackets will not be taken. If the input was rejected, the user will have to re-enter another text.

After the encoding process, an audio signal will be generated. This generated audio signal can be used in the decoding stage to ensure correct encoding of the English text the user has typed in. Moreover, there are many signals that are sent as test cases to run on our program. Figure 2 shows the data in the system for encoding phase and decoding phase.

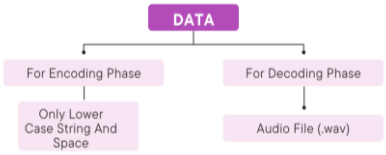

Figure 2: *Data for The System*

4. Evaluation Criteria

The evaluation criteria to assess the performance of the project depends heavily on the output of the decoded signals. After encoding a text message to an audio signal, the decoding of this signal is compared with the original text. The accuracy is calculated by dividing the correctly recognized letters by the length of the string. Therefore, a number of different test cases with different string lengths were tested as shown in the figure below with an accuracy 100%. Moreover, the two methods of decoding are compared with one another to see if both produce the same decoded text for the same input signal. To further increase the validity and assurance of the results, the encoded signal was plotted using the library “matplotlib”. The three indexes of the peaks shown in the plot are compared with the three frequencies of the specified letter given by the table of frequencies.

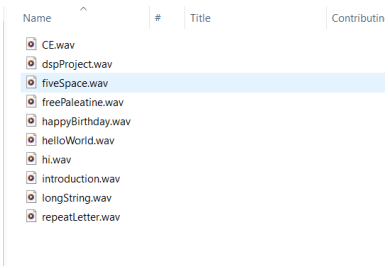

Figure 3:*Test Cases*

5. Approach

The encoding stage of the project was coded to distinctly represent each character as a combination of three frequencies. This is done by summing the three sinusoidal functions with the frequencies of the specified letter given in the table provide from the instructor. The sinusoidal function is in the form (cos(2*pi*freq*n/fs)), where the variable n is the number of samples and fs is the sampling frequency. The number of samples is calculated by multiplying the signal duration with the sampling frequency. To ensure the Nyquist rate, the sampling frequency must be greater than or equal to 2 multiplied with the greatest frequency (3500hz), thus it is set as 8000hz. The user can input a text of any length, thus the concatenation of all the letters of the string is done as seen in the below figure.

```
def phase_one_encode():
    global l, E1, is_encoded
    l = np.array([])
    stri = E1.get()

    # Validate the input: check if there are uppercase letters, numbers, or symbols
    if any(char.isupper() or char.isdigit() or (not char.isalpha() and not char.isspace()) for char in stri):
        messagebox.showerror( title: "Invalid Input",  message: "Please enter only lowercase letters and spaces.")
        return

    for i in stri:
        l = np.concatenate( arrays: (l, [np.cos(character_frequencies[i][0] * 2 * np.pi * n / sampling_freqency) +
                                np.cos(character_frequencies[i][1] * 2 * np.pi * n / sampling_freqency) +
                                np.cos(character_frequencies[i][2] * 2 * np.pi * n / sampling_freqency)
                                for n in range(number_of_samples)]), axis=None)

    is_encoded = True
    # Show an alert message for successful encoding
    messagebox.showinfo( title: "Success",  message: "String encoded successfully!")
```

Figure 4 :*Encode Phase*

The approach taken to solve this project is implementing two different decoding methods, which are frequency analysis and filters to contrast between the two different approaches. The magnitude of the FFT was found for each signal using the library called “scipy”, then the maximum magnitudes of these peaks were calculated. These peaks are then compared with the low, middle and high frequency components given in the predefined table. If there is a match, the letter is added to the output string as seen in the figure below.

```
def phasetwo_decode():
    global l, newstr
    result_str = "FFT Result: "

    for i in range(0, len(l), number_of_samples):
        freqMag = abs(fftpack.fft(l[i:i + number_of_samples], fast_forier_transform_freq))
        maxpoints = [
            int(np.argmax(freqMag[int(100 * (fast_forier_transform_freq / sampling_freqency)):int(
                500 * (fast_forier_transform_freq / sampling_freqency))]) + 100 * (
                    fast_forier_transform_freq / sampling_freqency)) * (
                    sampling_freqency / fast_forier_transform_freq),
            int((sampling_freqency / fast_forier_transform_freq) * (np.argmax(
                freqMag[int(1100 * (fast_forier_transform_freq / sampling_freqency)):int(
                    1500 * (fast_forier_transform_freq / sampling_freqency))]) + 1100 * (
                                                                                    fast_forier_transform_freq / sampling_freqency))),
            int((sampling_freqency / fast_forier_transform_freq) * (np.argmax(
                freqMag[int(2500 * (fast_forier_transform_freq / sampling_freqency)):int(
                    3500 * (fast_forier_transform_freq / sampling_freqency))]) + 2500 * (
                                                                                    fast_forier_transform_freq / sampling_freqency)))
        ]
        for j in range(3):
            maxpoints[j] = int(round(maxpoints[j] / 100) * 100)
        if maxpoints in character_frequencies.values():
            for char, freqs in character_frequencies.items():
                if freqs == maxpoints:
                    result_str += char

    newstr.set(result_str)
# Show an alert message for successful decoding using FFT
    messagebox.showinfo( title: "Decode using FFT",  message: "String decoded successfully using Fast Fourier Transform.")
```

Figure 5:*Decode Using FFT*

Applying bandpass filters is the second approach taken to decode a signal back to its original text. Here, each alphabet has three bandpass filters. These bandpass filters are narrow enough to prevent any intersections of other frequency spectrums. The filtered signal is compared with the frequencies in the predefined table. If a match was found, the corresponding alphabet is appended to the output string as seen in the figure below.

```
def passBass_decode():
    global l, newstr
    result_str = "BPF Result: "

    for i in range(0, len(l), number_of_samples):
        letterfrq = [0, 0, 0]
        for j, (low, high) in enumerate([(100, 500), (1100, 1500), (2500, 3500)]):
            x = apply_bandpass_filter(l[i:i + number_of_samples], low, high, sampling_freqency, filter_order=1)
            letterfrq[j] = np.argmax(abs(fftpack.fft(x, fast_forier_transform_freq))) * (
                    sampling_freqency / fast_forier_transform_freq)
            letterfrq[j] = int(round(letterfrq[j] / 100) * 100)
        if letterfrq in character_frequencies.values():
            for char, freqs in character_frequencies.items():
                if freqs == letterfrq:
                    result_str += char

    newstr.set(result_str)
    messagebox.showinfo( title: "Decode using Band Pass Filter",  message: "String decoded successfully using Band Pass Filter.")
```

Figure 6: *Decode Using BPF*

6. Results and Analysis

After building the code, for the first phase which is encoding, and the second phase which is decoding, we examined the system through several stages.

At the beginning, as shown in below figure, the user interface appears and it’s divided into two parts, the left part is encoding and the right part is decoding.

Figure 7:*User Interface*

In the encoding stage, the user must enter lower case string or string with spaces. If the user enters numbers or symbols, then click on the Encode String button an error message will appear to alert the user as shown in figure 8. If the user enters correct string, then click on the Encode String button a successful message will appear to notify the user that the encoding process was completed successfully as shown in figure 9.

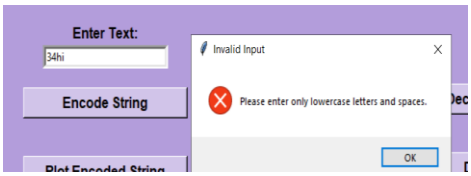

Figure 8: *Invalid Input String*

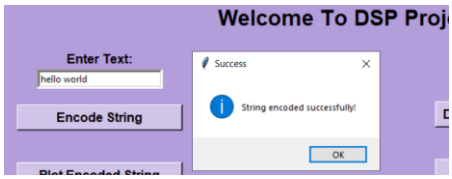

Figure 9:*Valid Input String*

If the user clicks on the plot button, a plot figure will appear for the encoded text (text that user enters it). The plot appears in the Time and Frequency domain, and we notice that each letter in the plot has a distinctive behavior that is different from the rest of the letters. Also, we notice each letter has three frequencies which are the low, middle, and high frequency.

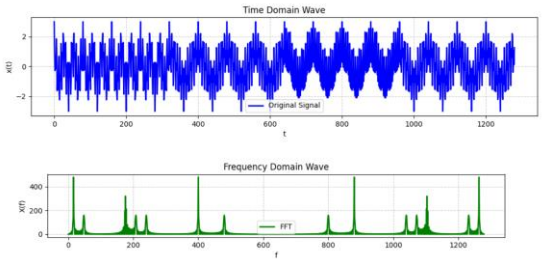

Figure 10:*Plot Encoded Signal*

Moreover, the user can listen to the generated signal by clicking on the Play button, and he can also save the audio file of the generating signal by clicking on the Save button, also the user can save the audio of generating signal in specific directory by clicking on Save As button The Exit button is to exit the program and close the user interface.

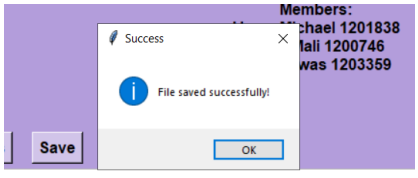

Figure 11:*Save Generated Signal*

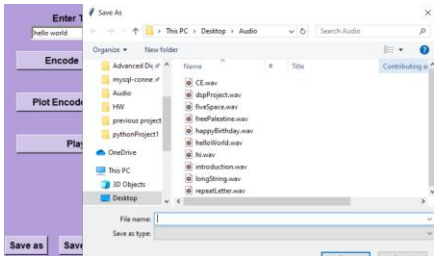

Figure 12:*Save In Specific Directory*

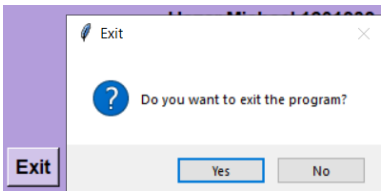

Figure 13:*Exit the Program*

For the decoding stage, the user can upload an audio file stored on his device by clicking on the Decode an audio signal stored in a file button then choose the audio file. After uploading the audio file, Alert Message will appear to the user to alert the user that the file has been uploaded successfully as shown in figure below.

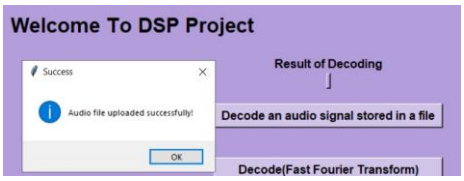

Figure 14:*Upload Audio File*

If the user clicks on Decode (Fast Fourier Transform) button, the result of decoding will appear under the Result label on the right side of the user interface. As shown in figure below.

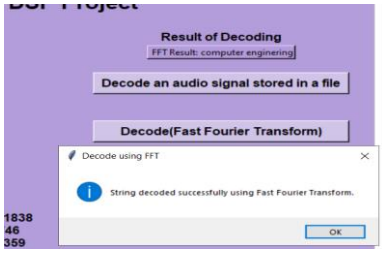

Figure 15:*Decode Using FFT*

The same steps are repeated if the user clicks on Decode (Band Pass Filter). As shown in figure below.

Figure 16:*Decode Using BPF*

As mentioned in the evaluation criteria section, to check that the program was working correctly, we made encoding with 10 strings as shown in figure below, so that some of the strings contained long sentences, short sentences, letters, and repeated characters, and also some of the strings contained only spaces. After that, we decoded these strings using Fast Fourier Transform (FFT) and Band pass filter (BPF), and the result was correct with accuracy 100%, as the original string was restored.

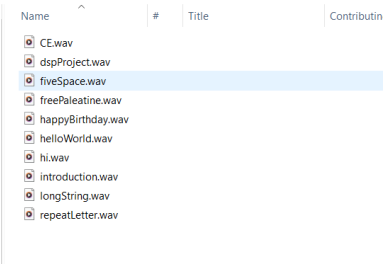

Figure 17:*Test Cases*

7. Development

At the present time, the system only receives strings that contain only lowercase letters. To improve the system and make it comprehensive, we are thinking to develop the system so that it receives uppercases letters, as well as, numbers and other special characters to allow the encoding of meaningful sentences.

For the system to be more effective, instead of writing the string as user input to encode the text, user's voice record can be inputted, and then encoded. We are also thinking about adding a feature so that the user can upload a plot of an encoded signal, so that the program decodes this plot and reveal the original message.

8. Conclusions

This project has provided an educational insight on digital signal processing specifically in the encoding of English text and decoding of an audio signal. We have gained knowledge on representing a signal in the frequency domain using fast Fourier transform. Moreover, by using filters to extract specific frequencies we could compare these peaks with the predefined table and find the letter it corresponds to. By changing the bandwidth of the bandpass filters, we saw that making it too wide would cause intersection of frequencies and inaccurate results. From this project, we were able to implement a real-world scenario and reinforce our understanding of digital signal processing.

9. References

[1] https://realpython.com/python-scipy-fft/.
Accessed on 29-12-2023 at 4:12 PM.

[2] https://www.youtube.com/watch?v=ZqpSb5p1xQo .
Accessed on 29-12-2023 at 6:12 PM.

[3] https://ieeexplore.ieee.org/abstract/document/1163283 .
Accessed on 30-12-2023 at 2:19 PM.

[4] https://www.youtube.com/watch?v=geSl3fmtQFs .
Accessed on 30-12-2023 at 5:19 PM.

[5] https://www.geeksforgeeks.org/digital-band-pass-butterworth-filter-in-python/ .
Accessed on 5-1-2024 at 8:18 PM.

[6] https://www.youtube.com/watch?v=boHWH_8ODv8 .
Accessed on 7-1-2024 at 9:45 PM.

[7] https://www.tutorialspoint.com/digital-band-pass-butterworth-filter-in-python .
Accessed on 10-1-2024 at 7:34 PM.

[8] Digital Signal Processing Text Book.