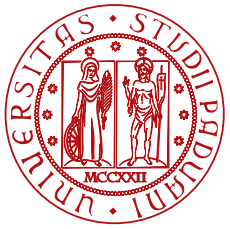

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT for Internet and Multimedia

Performance of encoding/decoding of bit strings using coded sound signals

Master Candidate

Yükselcan Sevil

Student ID 1216127

Supervisor

Leonardo Badia

University of Padova

Academic Year
2021/2022

This paper is dedicated to my supervisor Prof. Leonardo Badia under whose constant guidance I have completed this dissertation. He not only enlightened me with academic knowledge but also gave me valuable advice whenever I needed it the most.

Abstract

The aim of the thesis is to design a communication system based on encrypted sound code using the FEC (Forward Error Correction) method, STFT (Short-time Fourier Transform), and scrambling method to ensure transmission reliability even in the presence of noise interference. Encrypted sound can be transmitted and received over any device which has a speaker and microphone. Transmission over the channel is inevitably degraded by the presence of environmental noises added to the transmitted signal and therefore requires a channel code with high correction capacity. Reed Solomon, one of the forward error correction algorithms, is used in this project to correct channel errors caused by environmental noises. In this project, the effectiveness and performance of the system are evaluated using various Python simulations to evaluate the amount of errors created and corrected by varying the type and power of the noisy signal.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

The project is the result of a collaboration with the technology company "Omniaweb" and provides a feasibility assessment for the implementation of a data transfer method including duplication of a correctly prepared audio file and listening to it via a microphone-equipped device. The technique underlying QR codes [44][9] and the Shazam application [49] inspired the concept. QR codes exploit the image of a matrix to represent a string of bits, which is then replicated when the camera of a device captures the code. Shazam generates an auditory fingerprint from the audio and compares it to a central database to identify a match. Similarly, the purpose is to create a sound signal for transmission through a loudspeaker that, when decoded by a listening device, will lead back to the original message. Frequencies of the audible should be exploited in data transmission. Moreover, it is essential that the sequence of sounds is not overly disturbing to those who listen to them. However, it is challenging to create melodious sounds due to the number of frequencies that were used. Therefore every frequency that sounds musically pleasing to the ear can not be used in order to avoid distortions that may occur due to noise. In conclusion, due to the vast number of frequencies used, it is challenging to generate musical sounds in sufficient combinations. The unavoidable existence of noise that interferes with the sampling of the microphone of the receiving device is a second issue to address. To circumvent this issue as much as possible, a Reed-Solomon code [30][44] is implemented, which, by adding redundancy, permits any mistakes to be rectified during the decoding phase. Additionally, STFT was implemented to extract frequencies using various window types [24]. The primary purpose of

windowing in the spectral analysis is the ability to zoom in on the signal's finer details as opposed to analyzing the entire signal. In speech signal processing, where information such as pitch or formant frequencies are recovered by evaluating signals through a window of a certain duration. Moreover, the scrambler, [12] a frequently used technology in telecommunications, was also implemented to modify a data stream before the transmission. The model's capacity to correct errors has been demonstrated using mentioned methods and various noise environments. In the section on the state of the art, the technologies that this project inspired are described. In the background section, the approaches utilized for this project are described. In the section on structure and analysis, the key parts of this project and the outcomes of the research are discussed.

2

State of the Art

2.1 QR Codes

A QR code is a matrix bar code or two-dimensional code that can hold information and is designed to be interpreted by smartphones. QR stands for "Quick Response," implying that the contents of the code should be decoded at a rapid rate. On a white background, the code consists of black modules arranged in a square pattern. The encoded data may be text, a URL, or other information [44] [42]. The QR code was created to enable rapid decoding of its data. QR codes are becoming increasingly popular all over the world. Today, mobile phones with an integrated camera are commonly used to read QR Codes.

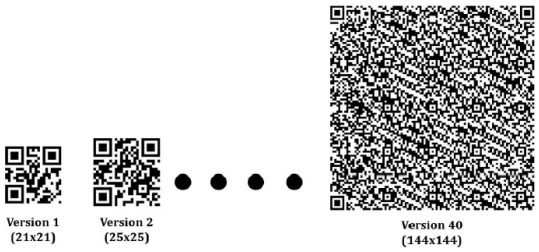

Figure 2.1: Versions of the QR Codes [44]

QR codes were originally created for the purpose of monitoring parts during the construction of vehicles; however, they have since found use in a diverse range of other contexts, such as industrial tracking, entertaining, in-store marketing materials, and smartphone-user applications [44]. After scanning a QR code, visitors will be able to open a URL and retrieve a text. Utilizing QR code-

generating websites or mobile apps, users can generate and print their own QR codes for others to scan and use. These codes are applied in a number of everyday contexts. For instance, they make it easier for visitors to access promotional content, which is particularly beneficial during advertising and marketing operations.

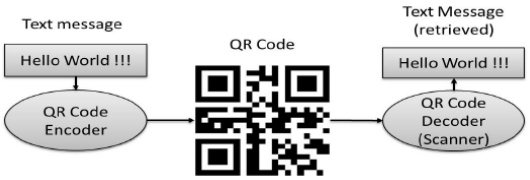

Figure 2.2: Working Principle of the QR Code [44]

Encoders and decoders are the two essential components of the QR code system [9]. While the encoder is in charge of encoding data and producing the QR Code, the decoder is in charge of extracting the data from the QR code. Figure 2.2 presents a high-level summary of how the QR code operates. The plain text, URL, or other data are input into the QR code encoder, and it generates the necessary QR code. When the information contained in the QR code needs to be accessed, the QR code is decoded using a QR Code decoder which retrieves the information contained in the QR code [22].

The concept behind the technology of the QR code was the limited amount of information capacity of barcodes (can only hold 20 alphanumeric characters). QR codes are among the 2D barcodes with the highest capacity and most widespread use. QR codes, because of the way they are constructed, are able to encode the whole 256-byte ASCII character set [43]. The standardized, two-dimensional QR code is capable of storing information and functioning as a framework. QR codes can be created in a total of forty distinct forms, numbered variant 1 through variant 40 as shown figure 2.1 [44] [42].

The structure of a QR code is shown here in figure 2.3. QR code is composed of several segments. These are; version information, format information, data and error correction keys, required patterns, and quiet zone.

A. Version Information

QR code determines the appropriate form number of the QR code by inspecting two blue boxes, as shown in figure 2.3. These boxes are used to determine if the QR code should use version 1 or 2, and so on. After that, it will obtain

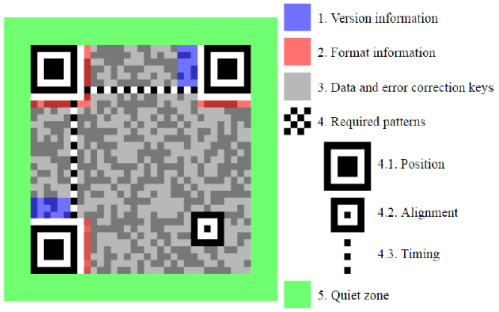

Figure 2.3: Design of the QR Code [9]

information regarding the total number of modules that are included in the QR code. Form 1 has a 21*21 module, and different variants have a different module estimate, thus this information will be used [22].

B. Format Information

The error correction level and mask pattern used in the generation of this QR code can be determined from the format pixels. Only larger QR codes require the version pixels, which encode the size of the QR matrix and are only used there.

C. Data and Error Correction Keys

It includes genuine information that has been stored in a QR code and a blunder adjustment codeword in accordance with the Reed-Solomon computation. During the process of encoding a QR code, initially information is stored in these dark cells, and then a mistake redress codeword is stashed away. In QR codes, we have four different mistake amendment levels (L-Level M-Level Q-Level, and H-Level), so there are a total of 28 possible configuration data strings [42]. Error-correction levels and their approximate ability of error correction are listed below.

- Level L : 7% or less errors can be corrected.
- Level M : 15% or less errors can be corrected.
- Level Q : 25% or less errors can be corrected.
- Level H : 30% or less errors can be corrected.

C. Required (Function) Patterns

The required patterns are able to convey to the QR code scanner the correct position of the Brisk Reaction code as well as the auxiliary state it should be in [9]. Because of the position part, QR codes may quickly respond from any direction. It renders the QR code capable of being read in any direction. The alignment process addresses and corrects any twists that may have occurred in the QR codes. It was introduced at the release of form 2 of the QR codes, and the number of illustrative configurations grows with each succeeding iteration of the technology. The timing component is designed to function as an alternative for high contrast areas in the various discoverer designs. In the event that an image has been corrupted, it is used to locate the code and make necessary adjustments to the focus organization.

C. Quiet Zone

After establishing the symbol version and module size, the size of the QR Code symbol is determined. It is essential to include a margin or "silent zone" around the QR Code symbol in order for it to be correctly identified [22]. The margin is the blank space surrounding a symbol that is not printed on. To be deemed a QR Code, a symbol must have a four-module-wide margin on all four sides.

In conclusion, the idea behind QR codes is similar to that discussed in the thesis: both use multimedia content as a transmission channel and, above all, want to make the transmission as reliable as possible even in the event of noisy events. The method used to correct these errors (Reed-Solomon Codes) is actually the subject discussed in the thesis. QR codes are very efficient technology as they allow to achieve a very high bit rate that depends only on the reading speed of the device framing the code, and this thesis will discuss how bit-rate affects it.

2.2 Shazam

The music identification app known as Shazam has been available for nearly two decades at this point. Audio can be recognized by Shazam by means of an audio fingerprint that is derived from a time-frequency graph that is known as a spectrogram [49]. It does this by collecting a small sample of the audio that is currently being played using the microphone that is already incorporated into a smartphone or computer. Shazam maintains a database that contains a catalogue of audio fingerprints. In short, Shazam's work steps are:

- The entire music database is scanned with Shazam's fingerprinting technology, and the associated fingerprints are saved in a database.
- A user records the music they are listening to, which fingerprints a 10-second audio clip.
- The fingerprint is sent to Shazam's server, where it is compared to their database in an effort to locate a match.
- Music information is only sent back to the user if there is a positive match; otherwise, an error message will be displayed.

A spectrogram is a three-dimensional graph that is utilized to illustrate sound. The spectrogram depicts the fluctuation of frequencies over time, taking into account the amplitude or volume. Through the use of the spectrogram as seen in figure 2.4, a constellation of peaks is produced; each peak denotes a time-frequency point that has a higher energy level in comparison to the points that are surrounding it.

After the constellation has been determined as seen in figure 2.5, each peak is looked at once more and used as a reference to determine the target zone (which is shown in figure 2.7), which is a zone consisting of sequential peaks. A string of information is saved for each pair in the target zone. This information is then fed into a hash algorithm (which is shown in figure 2.6), and the results are stored in a hash table.

When the application samples a fragment of a song, its footprint and relative hashes are immediately generated and transferred to the server; the server then matches the fragment's hash with those of the songs using the hash tables in its database. The piece with the largest number of consecutive matches will be selected.

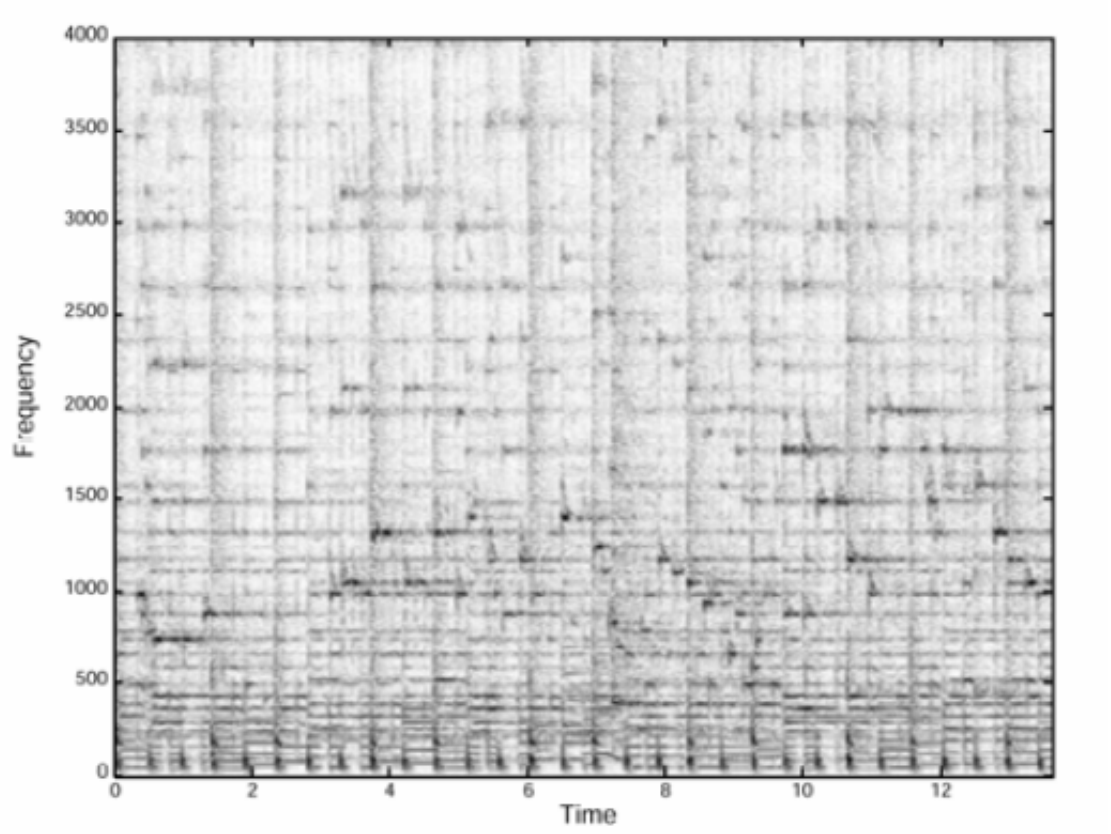

Figure 2.4: Example of Spectrogram [49]

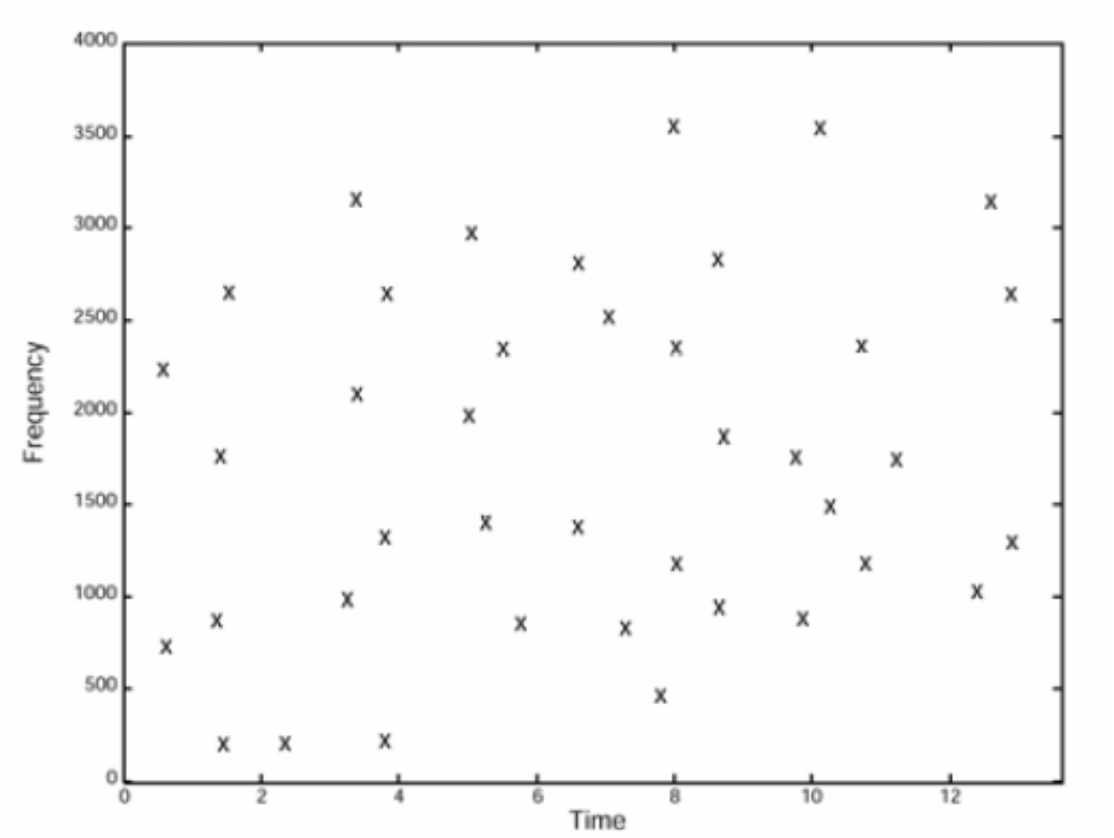

Figure 2.5: Example of Constellation Map [49]

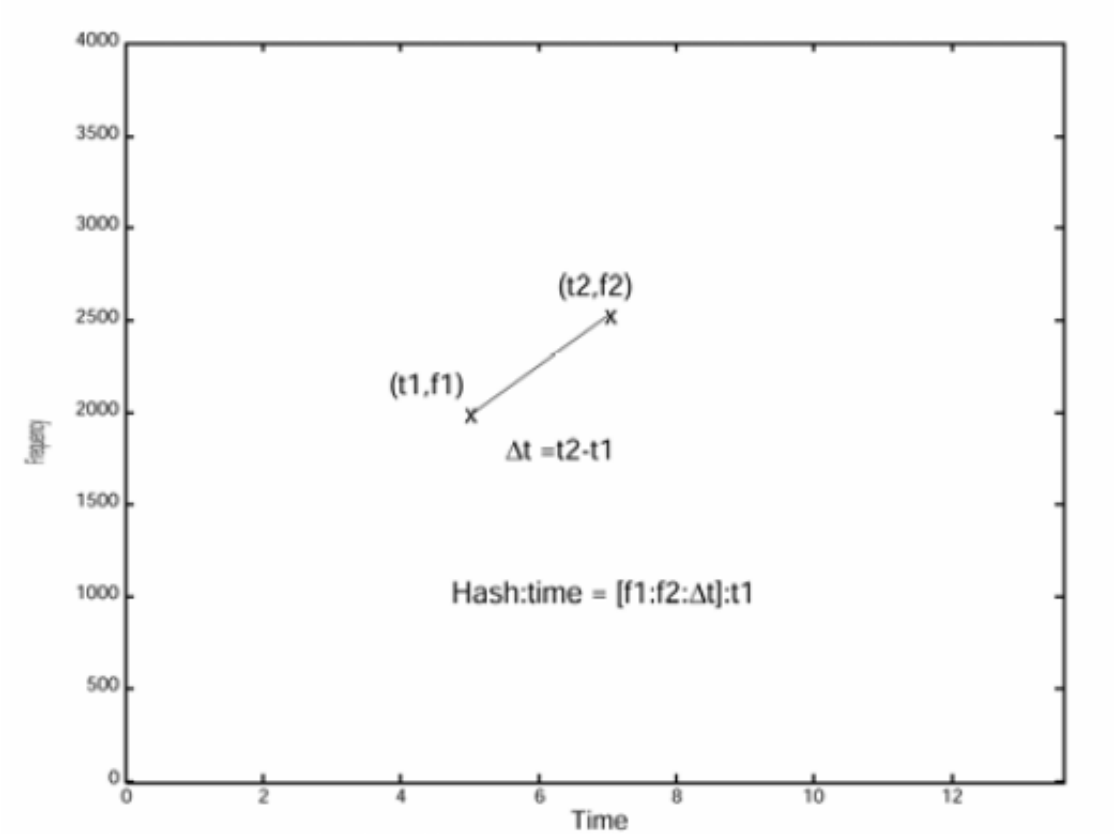

Figure 2.6: Example of Hast Details [49]

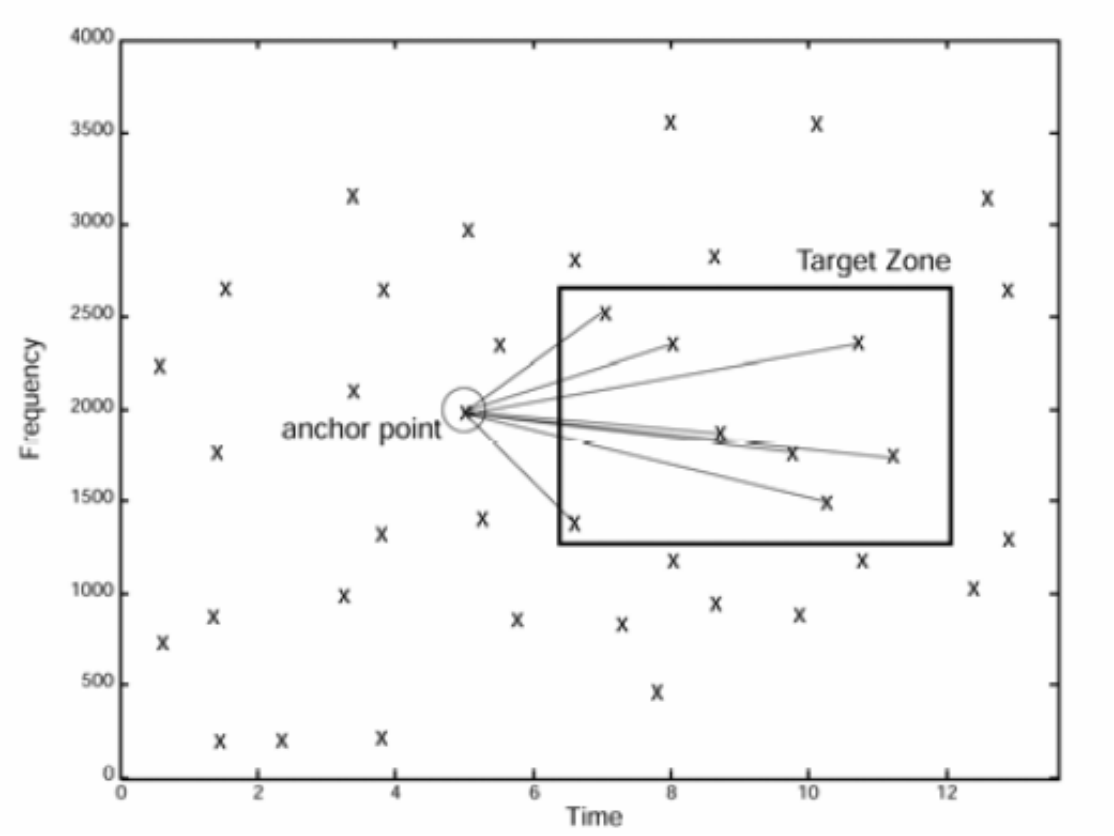

Figure 2.7: Example of Combinatorial Hash Generation [49]

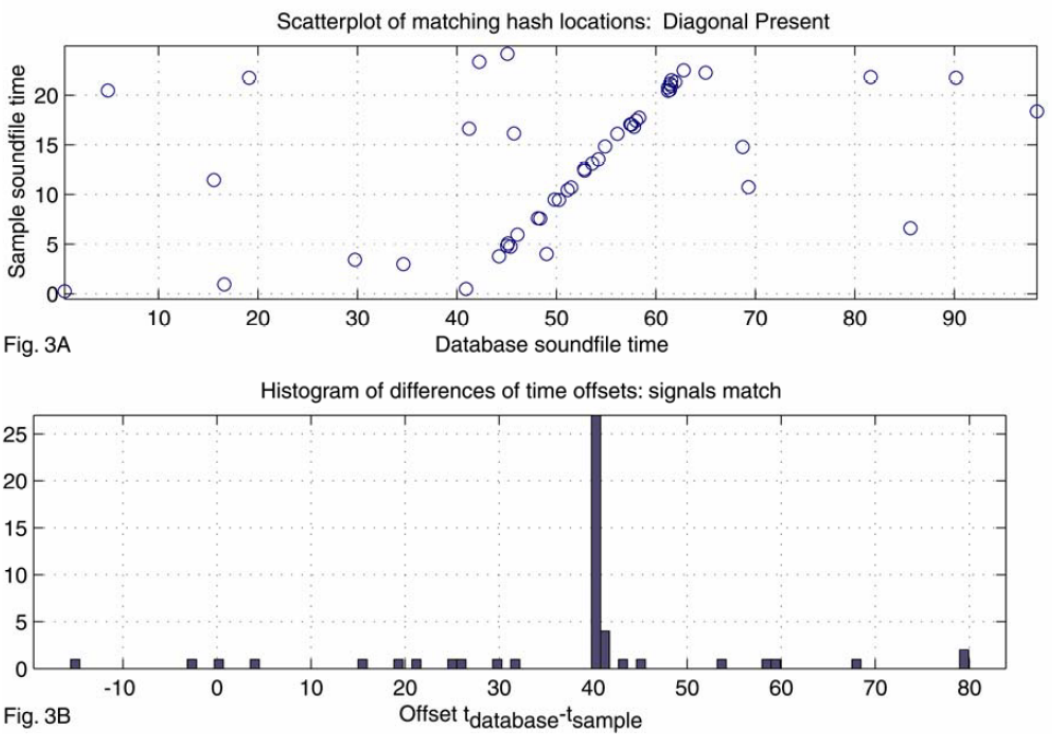

Figure 2.8: Example of Matched Hash Locations [49]

Figures 2.8 and 2.9 illustrate two examples of comparisons between the sampled portion and the entire piece, respectively. The presence of the diagonal, as shown in Figure 2.8, confirms the presence of the fragment within the piece. The presence of the diagonal, as shown in Figure 2.9, does not confirm the presence of the fragment within the piece.

Attempts have been made to include the approach into the thesis, which was inspired by Shazam's speech recognition technology. The thesis involves the creation of a sound alphabet (database) by making use of a specified frequency range, and the spectrogram is applied to find the frequencies that are being used in created sound. The created sound is still melodic in character and discernible by human ear, which is deliberate. This is comparable to QR codes in that they don't seem like anything in particular (such as a Leonardo Da Vinci or Claude Monet painting), yet they are easily recognized by the human eye. This issue is addressed in the analysis section.

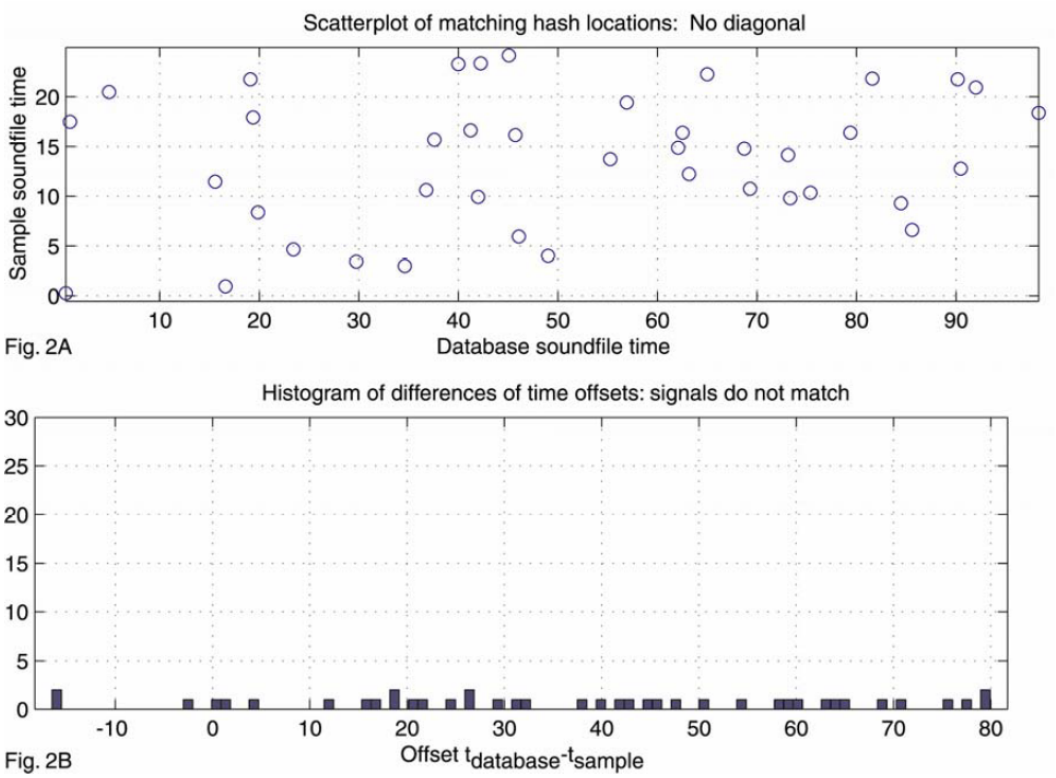

Figure 2.9: Example of Unmatched Hash Locations [49]

3 Background

In this chapter will show the methods and algorithms that were utilized, with reference to the approaches taken by the significant technologies covered in the previous chapter. Additionally, a different approaches are showed in this section of the thesis.

3.1 Reed-Solomon Codes

Reed-Solomon codes, which were given their name after Reed and Solomon following the publication of their work in 1960 [30], have been utilized in a broad variety of contexts in conjunction with hard decision decoding.

Reed-Solomon codes are a type of code that is used for forward-error correction and are implemented in data transmissions that are susceptible to channel noise. Reed-Solomon codes are a type of block code that are capable of detecting and correcting faults inside a block of data. This is accomplished by inserting redundant data before transmission of the code. The demonstration of standard Reed-Solomon codes can be seen on Figure 3.1.

Reed-Solomon codes have been implemented in a wide variety of burst error correction applications, digital storage and communication systems, hybrid ARQ systems, and various applications [10] [11] [46] [45] [19].

The research indicates that using forward error correction coding, which is both the most effective and economical solution, is preferable when there is noise and interference present. This allows for more efficient communication, as well

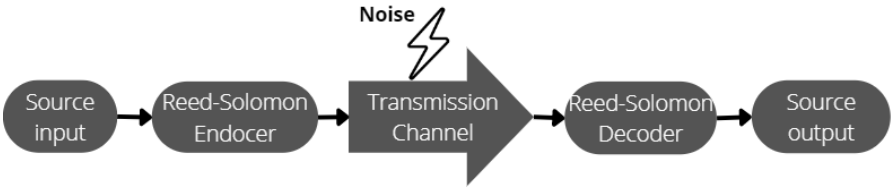

Figure 3.1: Transmission chart of Reed-Solomon Codes

as a reduction in the requirement for retransmissions and the consumption of energy. The convolutional encoder with the Viterbi decoder can find compound errors with an Es/No ratio that is equivalent to 20 dB, whereas the RS coding [28] [36] offers a superior BER for a gain that is greater than 7 dB.

3.1.1 Characteristics of Reed-Solomon Codes

Reed-Solomon codes are a type of error-correcting code that uses blocks. The encoder will append redundant data, known as parity data, to each block of data that is sent in as input. Data that has been corrupted as a result of noise that occurred during the communication transfer can be recovered with the use of redundant data. The number of errors and the kinds of defects that could be rectified dependent on the properties of the Reed-Solomon code. The Bose-Chaudhuri-Hocquenghem (BCH) codes codes contain a non-binary sub-type that is known as the Reed-Solomon codes [18].

They are represented by the expression $Rs(n, k)$ using m-bit symbols, where k represents the number of information symbols that have a length of m bits. Encoding a word into n symbols requires the encoder to start with k data symbols, each of which has s bits, and then add parity symbols. There are $(n - k)$ parity symbols, and each one consists of s bits. A Reed-Solomon decoder has the capability of correcting up to t symbols in a codeword that include errors, where $2t = n - k$. The amount of parity symbols is related to the code's ability to correct and the number of symbols. A typical Reed-Solomon codeword is illustrated in Figure 3.2.

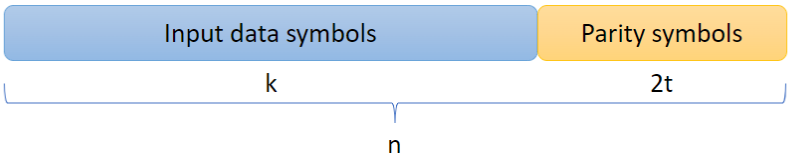

Figure 3.2: Reed-Solomon Codeword

Example:

A popular Reed-Solomon code can be created using 8-bit symbols and written it as $RS(255, 223)$. Each codeword is made up of 255 bytes, with one symbol equaling one byte. Out of these 255 bytes, 223 bytes provide information, while the remaining 32 bytes contain parity. The parameters are:

$$n = 255, \quad k = 223, \quad m = 8, \quad 2t = 32$$

The decoder has the capability to correct up to 16 faulty symbols where they appear in the codeword. Within the codeword, the code can fix up to 16 bytes if required.

Encoding and decoding Reed-Solomon codes requires a level of computational difficulty [32] that is dependent on the number of parity symbols contained within the codeword. The greater the number of parity symbols that are connected to the information symbols, the greater the number of mistakes that the code is able to rectify; nevertheless, this needs a greater amount of processing power.

The maximum length of a codeword using the Reed-Solomon algorithm is $n = 2^m - 1$. This is defined by size of m. A codeword comprised of 8-bit symbols, for instance, has an absolute maximum length of 255 bytes.

It is feasible to shorten Reed-Solomon codes by setting a particular number of data symbols to zero at the encoder, canceling the transmission of those data, and then re-entering those symbols at the decoder [31].

3.1.2 Encoding Reed-Solomon Codes

To do Reed-Solomon encoding, it is necessary to first compute the generating polynomial $GF(2^m)$. The amount of parity check symbols in the code is

exactly directly proportionate to this polynomial's degree. In another formulation, both input and output data consist of symbols, which are elements of $GF(2^m) = \{0, 1, \alpha, \alpha^2, ...\}$. These variables are structured as polynomial coefficients, and the power of the polynomial variable x determines the order in which the encoder and decoder traverse the respective symbol. For instance, the information polynomial $g(x) = \alpha x^2 + \alpha^3 x + \alpha^4$ indicates that the encoder will process the symbol α first. Finally, symbol α^4 follows symbol α^3. In Equation 3.2, the formula for the generator polynomial can be found.

Information symbols are moved on higher power coefficients x^{n-k} and parity symbols are inserted. Parity symbols are determined by dividing the relocated information polynomial by the generator polynomial and taking the residual. This generates the codeword $c(x)$.

The equation is defined as [37]:

$$c(x) = i(x)x^{n-k} + [i(x)x^{n-k}]modg(x) \tag{3.1}$$

Description:

$c(x)$: primitive element of the field

$i(x)$: the generator polynomial

$g(x)$: the generator polynomial

$[i(x)x^{n-k}]modg(x)$: the parity polynomial of degree

And the generator polynmial is expressed as,

$$g(x) = \prod_{p=l}^{l+2t-1} (x + a^p) = (x - a^l)(x - a^{l+1})...(x - a^{l+2t-1}) \tag{3.2}$$

The generated codeword would be acceptable since it is evenly divisible by generator polynomial.

3.1.3 Decoding Reed-Solomon Codes

Reed-Solomon decoders make an effort to determine the location and size of up to t errors or $2t$ erasures, and then rectify them if they are able to. Calculating the syndrome of the receiver $c(x)$ is a necessary step in the decoding of a Reed-Solomon code. This reveals whether or not $c(x)$ is a member of the codeword set that we are looking for [18]. The presence of a zero syndrome will serve as evidence that it is a component of the collection. On the other hand, the presence of mistakes in the received vector will be indicated by the return of a value that is not zero. Therefore, $c(x)$ will be the sum of the vector that was sent, which is denoted by $r(x)$, and the vector that represents the error, which is denoted by $e(x)$.
Decoding steps of Reed-Solomon codes are [18]:

- Computation of syndrome.
- Computation of the error-locator polynomial $\sigma(x)$, whose multiplicative inverse determines the error's location. Various techniques exist for locating the roots of an error-locator polynomial. These are:
  1. Berlekamp-Massey algorithm[18]
  2. Euclidean algorithm [33]
- The Chien search technique is used to discover the roots of the error-locator polynomial step [18].
- After the calculation of the error-locator polynomial, error values (magnitudes) are calculated by using Forney's algorithm [17].

The decoding process diagram can be seen in Figure 3.3:

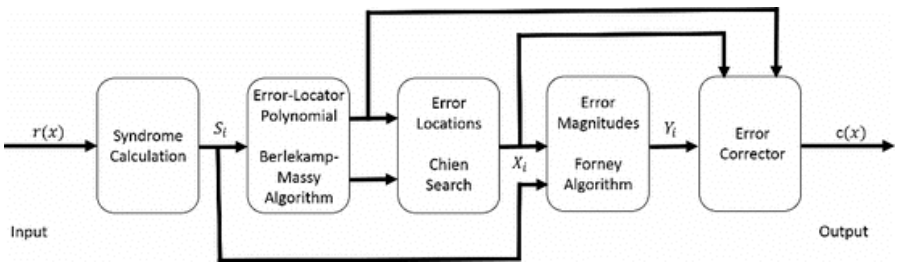

Figure 3.3: Decoding Process of Reed-Solomon Code [27]

The codeword polynomial is expressed as in Equation (3.3) and received polynomial is expressed as in Equation (3.4):

$$c(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1} \tag{3.3}$$

$$r(x) = r_0 + r_1 x + ... + r_{n-1} x^{n-1}, \tag{3.4}$$

If the polynomial that was received includes values of τ that are greater than zero, this indicates that τ errors occurred during the transmitting and are located at the location $(j_1, j_2, ..., j_\tau)$, where j is an integer between 0 and n-1. So error polynomial is expressed in Figure 3.5

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + ... + \tau x^{j_\tau}, \tag{3.5}$$

and the relation between them is expressed in Equation 3.6

$$c(x) = r(x) + e(x). \tag{3.6}$$

Following the calculation of the first step, which is the determination of the error polynomial (shown in Equation (3.5)), the next step is the rectification of errors in the polynomial that was received using Equation (3.6). The output of the encoder is the real codeword polynomial, which can be found by adding the received polynomial to the error polynomial. If the Reed-Solomon code has sufficient error correction capability, which is defined by its characteristic (n, k), then this sum will give the correct codeword polynomial [37].

Syndrome Computation

The syndromes for Reed-Solomon codes can be calculated using

$$S_k = r(a^i) = e_{j_1}(a^i)^{j_1} + e_{j_2}(a^i)^{j_2} + \cdots + e_{j_\tau}(a^i)^{j_\tau}, \tag{3.7}$$

where b is an integer and $b \leq i \leq b + 2t - 1, \qquad k = 0, 1, ..., 2t$

and the error locator polynomial is expressed as

$$\sigma(x) = \prod_{\ell=1}^{\tau}(1 + \alpha^{j_\ell}x) = 1 + \sigma_1 x + \sigma_2 x^2 + \ldots + \sigma_2 x^\tau \tag{3.8}$$

The inverses of the error locations are equivalent to the roots of the error-locator polynomial, which are used to find errors. If this is the case, there is a connection between the coefficients of $\sigma(x)$ and the syndromes can be seen in Equation (3.9) [20]:

$$\begin{bmatrix} S_{\tau+1} \\ S_{\tau+1} \\ \vdots \\ S_{2\tau} \end{bmatrix} = \begin{bmatrix} S_1 & S_2 & \ldots & S_\tau \\ S_2 & S_3 & \ldots & S_{\tau+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\tau & S_{\tau+1} & \ldots & S_{2\tau-1} \end{bmatrix} \begin{bmatrix} \sigma_\tau \\ \sigma_{\tau-1} \\ \vdots \\ \sigma_1 \end{bmatrix} \tag{3.9}$$

Error locator polynomial can be found by result of Equation (3.9). If each syndrome has a value of zero, therefore the codeword is transmitted correctly, and the decoding process for the given chunk of data can be considered to have reached its conclusion.

Determination of the Error-locator Polynomial

Using the Berlekamp technique, the coefficients of the error location polynomial $\sigma(x)$ are computed iteratively. When a set of coefficients is calculated using the Berlekamp algorithm, the coefficients are verified by utilizing the algorithm to estimate a syndrome utilizing the calculated coefficients. In the case that the check fails, a correction factor will be used to compute the coefficients of the error location polynomial [18] during the future iteration.

We know that from Equation (3.9), $\hat{S}_{\tau+1}$ is the estimated syndrome which is gathered using the estimated coefficients. Accordingly, discrepancy factor is expressed as

$$d_i = \hat{S}_{i+1} + S_{i+1} \tag{3.10}$$

In the equation that the estimated coefficients are accurate, then the estimated

syndrome $\hat{S}_{\tau+1}$ will become equivalent to the syndrome that was determined to be $S_{\tau+1}$. Then equation is equivalent to

$$d_i = \hat{S}_{i+1} + S_{i+1} \rightarrow d_i = 0 \tag{3.11}$$

Otherwise, the equation translates into,

$$d_i = \hat{S}_{i+1} + S_{i+1} \rightarrow d_i \neq 0 \tag{3.12}$$

Now, suppose that we have a parameter n_k as

$$n_k = k - r_k \tag{3.13}$$

where r_k is the degree of $\sigma_k(x)$ and k is the iteration index.

Accordingly, Berlekamp algorithm can be expressed in a few steps. These steps are [18]:

1. The iteration index i starts from -1. That means

$$\sigma_{-1}(x) = 1 \qquad r_{-1} = 0 \qquad d_{-1} = 0 \qquad n_{-1} = -1 \tag{3.14}$$

and if i = 0, we have

$$\sigma_0(x) = 1 \qquad r_0 = 0 \qquad d_0 = S_1 \qquad n_0 = 0 \tag{3.15}$$

2. The iteration index i is between $i = 1, \ldots, 2t$
3. Estimated syndrome is determined by using

$$\hat{S}_{i+1} = \sigma_1^{'} S_i + \sigma_2^{'} S_{i-1} + \ldots + \sigma_{r_i}^{'} S_{i+1-r_i} \tag{3.16}$$

Calculating the discrepancy factor requires using both the estimated syndrome

and the computed syndrome. So we have

$$d_i = \hat{S}_{i+1} + S_{i+1} \tag{3.17}$$

4. If $d_i = 0$, then relation between iterations of error location polynomial are

$$\sigma_{i+1}(x) = \sigma_i(x) \qquad r_{i+1} = r_i \tag{3.18}$$

where r_{i+1} represents the degree of $\sigma_{i+1}(x)$ and r_i denotes the degree of $\sigma_i(x)$
If $d_i \neq 0$, then relation between iterations of error location polynomial are

$$\sigma_{i+1}(x) = \sigma_i(x) + e_i(x) \tag{3.19}$$

where $e_i(x)$ is expressed as

$$e_i(x) = \frac{x^i d_i}{x^k d_k}\sigma_k(x) \qquad k < i \tag{3.20}$$

in which $\sigma_k(x)$ indicates which of the previously generated polynomials such that

$$n_k = k - r_k \qquad k < i \tag{3.21}$$

has the highest value when all of the previously generated $\sigma_k(x)$ values are taken into consideration. Therefore, the degree of $e(x)$ can be calculated as

$$p_k = i - k + r_k \rightarrow p_k = i - n_k \tag{3.22}$$

where r_k is the degree of $\sigma_k(x)$. Therefore, calculation of $\sigma_{i+1}(x)$ is expressed as

$$r_{i+1} = max(r_k, p_k). \tag{3.23}$$

5. If $i > 2t$, finish the process.

Finding roots of error-locator

To locate the roots of a polynomial $g(x)$ [18], we test each field element in $g(x) = 0$ to see if the results are zero, i.e., we test and determine those satisfying $g(\alpha^i) = 0$.

$$g(\alpha^i) = 0 \qquad i = 1, ..., n \qquad \text{where: } n = 2^m - 1 \tag{3.24}$$

Chien search is the name given to the recommended method for performing a search in a systematic manner, which is done in order to facilitate the execution of the roots search [18]. The Chien search for error location polynomial is represented as,

$$\sigma(x) = \sigma_\tau x^\tau + \sigma_{\tau-1} x^{\tau-1} + \cdots + \sigma_2 x^2 + \sigma_1 x + \sigma_0 \tag{3.25}$$

When the Equation (3.25) is implemented for α^i, the equation translates into,

$$\sigma(\alpha^i) = \sigma_\tau \alpha^{i\tau} + \sigma_{\tau-1} \alpha^{i(\tau-1)} + \cdots + \sigma_2 \alpha^{2i} + \sigma_1 \alpha^i + \sigma_0 \tag{3.26}$$

So the states can be defined as,

$$q_\tau = \sigma_\tau \alpha^{i\tau} \qquad q_{\tau-1} = \sigma_{\tau-1} \alpha^{i(\tau-1)} \qquad q_2 = \sigma_2 \alpha^{2i} \qquad q_1 = \sigma_1 \alpha^i \qquad q_0 = q_0 \tag{3.27}$$

If $x = \alpha^{i+1}$, we get

$$q_\tau^{'} = \sigma_\tau \alpha^{i\tau} \qquad q_{\tau-1}^{'} = \sigma_{\tau-1} \alpha^{i(\tau-1)} \qquad q_2^{'} = \sigma_2 \alpha^{2(i+1)} \qquad q_1^{'} = \sigma_1 \alpha^{(i+1)} \qquad q_0^{'} = q_0 \tag{3.28}$$

For q_l and $q_l^{'}$, where $l > 0$, we have

$$q_\tau^{'} = \sigma_\tau \alpha^{\tau} \qquad q_{\tau-1}^{'} = \sigma_{\tau-1} \alpha^{\tau-1} \qquad q_2^{'} = \sigma_2 \alpha^{2} \qquad q_1^{'} = \sigma_1 \alpha^{1} \qquad q_0^{'} = q_0 \tag{3.29}$$

First, in the Chien search algorithm, we calculate Equation (3.27) with $i = 1$, and then we check to see whether we have or not.

$$\sum_{j=0}^{\tau} q_j = 0 \tag{3.30}$$

If the conditions of Equation (3.30) are met, then the symbol is a root. First, when I equals 2, we use Equation (3.29) to update the states, and then we check to see if we have or not.

$$\sum_{j=0}^{\tau} q_j^{'} = 0 \tag{3.31}$$

If the condition in Equation (3.31) is satisfied, then α^2 is a root.

Calculation of Error Values

The Forney algorithm [18] is responsible for the evaluation of error values in positions j_l. It is calculated like Equation (3.32).

$$e_{j_l} = \frac{(\alpha^{j_l})^{2-b}\Lambda(\alpha^{-j_l})}{\sigma'(\alpha^{-j_l})} \tag{3.32}$$

$\Lambda(x)$: is the error evaluator polynomial
$\sigma^{'}$: is the formal derivative of error-locator polynomial with regard to error-locator polynomial x

3.2 Digital Signal Processing

The Fourier transform is a mathematical method used to transform data into a spectrum of sinusoidal components in order to simplify signal representation and system performance analysis [25]. In some applications, the Fourier transform can be used for spectral analysis, and in others, it is applied for spectrum forming, which modifies the relative contributions of various frequency components to the filtered output. In certain applications, the Fourier transform is utilized for its ability to decompose the input signal into uncorrelated components, allowing for more efficient signal processing on the individual spectral components.

In different situations, the CT Fourier transform, the discrete-time Fourier Fourier transform (DTFT), the discrete Fourier Fourier transform (DFT), the fast Fourier transform (FFT), and the short-time Fourier transform (STFT) are utilized [14] [23] [40] [50] [35].

In this section, the signal processing algorithms (such as STFT and FFT) will be discussed.

3.2.1 Fast Fourier Transform

FFT is a faster variant of the DFT. Since Cooley and Tuckey introduced the fast Fourier transform in 1965 [15], it has evolved into a significant digital signal processing (DSP) method. The FFT takes use of the fact that the straightforward method for computing the Fourier transform repeats numerous multiplications. Utilizing the algebraic features of the Fourier matrix, the FFT algorithm integrates these redundant calculations in an incredibly efficient manner. In particular, the FFT utilizes patterns in the sines multiplied to complete the calculation [25]. Essentially, the FFT factorizes the Fourier matrix into many sparse matrices. Numerous entries in these sparse matrices are equal to zero. Using sparse matrices minimizes the total number of needed calculations. The FFT eliminates nearly all of these unnecessary calculations, which saves a substantial amount of calculation time and makes the Fourier transform significantly more applicable in numerous applications today [14] [38] [26].

As a starting point, the DTFT mathematical model is used to produce a

computer tool for Fourier transformations and it is expressed as

$$X_m[\omega] = \sum_{n=0}^{N-1} x[n]e^{-j\omega n} \tag{3.33}$$

where $x[n]$ is the input signal and N is the number of points indicating the signal's duration. The signal in the frequency domain is sampled equally with N points per period, $0 < \omega < 2\pi$, i.e.,

$$\omega_k = \frac{2\pi}{N}k, \qquad k = 0, 1, \ldots, N-1 \tag{3.34}$$

The FFT can be used to effectively estimate the DFT [41]. DFT algorithm computational complexity is on the order of N^2 operations. In fact, the number of computations performed by the FFT algorithm is about comparable to $Nlog_2N$. As a result of this significant reduction in computing complexity, FFTs are almost always favored over DFTs.

3.2.2 Short-Time Fourier Transform

The DTFT is a sort of Fourier Transform that identifies the sinusoidal phase and frequency characteristics of local or divided regions of a signal as it varies over time. STFT is one sub-type of DTFT [41]. As a result, STFT is able to maintain some information regarding the passage of time. Given a time signal x the STFT matrix X_m can be constructed as follows:

$$X_m[\omega] = \sum_{n=-\infty}^{\infty} x[n]\omega[n - mR]e^{-j\omega n} \tag{3.35}$$

where,

$X[n]$ = input signal at time n

$\omega[n]$ = sliding analysis window

$X_m[\omega]$ = DTFT of window data centered about time mR

R = hop size in samples. The hop size is the difference between the window length M and the overlap length L.

Notice that if $n = m$ and $\omega[0] = 1$, then $x(n)$ is calculated using Equation (3.36).

$$\omega[n - mR]x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X_m[\omega]e^{j\omega n}d\omega. \tag{3.36}$$

STFT algorithm leading to a twodimensional graphical representation of the squared magnitude of the STFT called the spectrogram. The graphical representation of STFT can be seen in Figure 3.4.

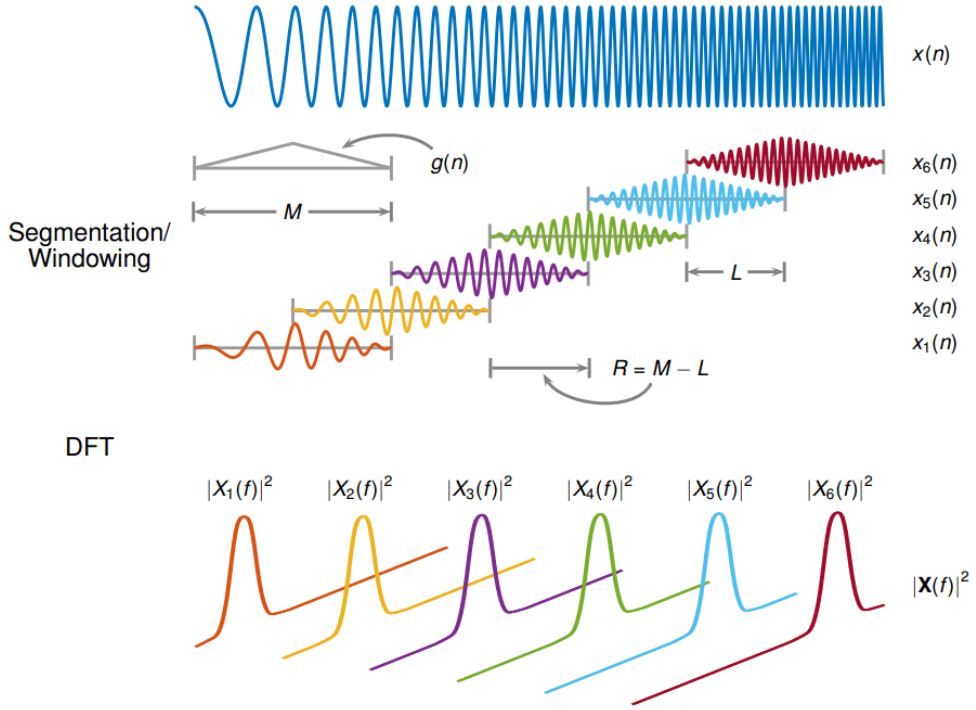

Figure 3.4: Graphical representation of STFT [39]

Windowing

As described in chapter 3.2.2, the window length is one of the most important parameters influencing the STFT value [51]. The window length also influences the STFT's temporal resolution and frequency resolution. The narrower windows have a small time duration but a broad bandwidth, they result in a fine

time resolution but a coarse frequency resolution [47]. Because wide windows have a long duration but a restricted frequency bandwidth, they result in a fine frequency resolution but a coarse temporal resolution. The term for this phenomena is the window effect [24]. The STFT cannot simultaneously provide a fine time resolution and a fine frequency resolution. The STFT has the same time resolution and frequency resolution over the whole time-frequency plane when using a time-invariant window. There are several windowing function to be utilized. These are:

Blackman Window:

The Blackman window is a taper formed from the first three factors of a cosine sum [21] It was created to have the least amount of leakage possible. The Blackman window is characterized by:

$$\omega[n] = 0.42 - 0.5\cos\left(\frac{2\pi n}{M}\right) + 0.08\cos\left(\frac{4\pi n}{M}\right) \tag{3.37}$$

The Blackman window was designed to eliminate the 3rd and 4th lower order harmonics, however its boundaries are discontinuous, leading to a 6 dB/oct falloff. This window is an estimation of the "precise" window, which nulls the sidelobes less effectively but has smooth edges, leading in a rise in the fall-off rate to 18 dB/oct [21].

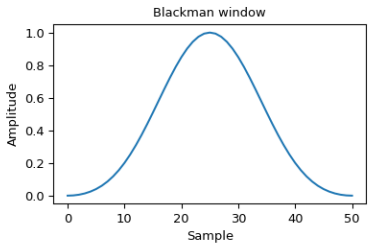

Figure 3.5: Blackman window [34]

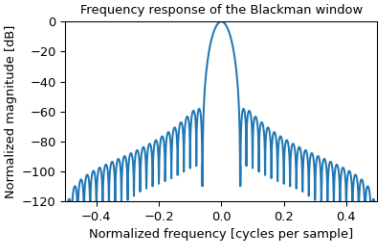

Figure 3.6: Frequency response of the Blackman window [34]

Hamming Window:

The Hamming window is a taper created by utilizing a rising cosine with

non-zero endpoints, with the goal of minimizing the closest side lobe. The Hamming window is defined as

$$\omega[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{M-1}\right), \qquad 0 \leq n \leq M-1 \tag{3.38}$$

The Hamming was called after R. W. Hamming, a J. W. Tukey associate, and is detailed in Blackman and Tukey [13]. It was suggested for smoothing the shortened time-domain autocovariance function. The majority of mentions to the Hamming window are found in the signal processing literature, where it is one of several windowing functions used to smooth values.

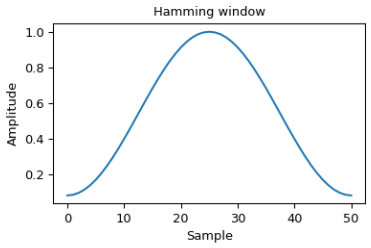

Figure 3.7: Hamming window [34]

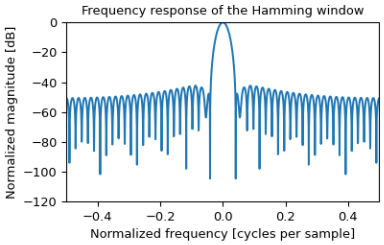

Figure 3.8: Frequency response of the Hamming window [34]

Hann Window:

The Hann window is a taper created using an increased cosine or sine-squared with zero-touching ends. The Hann window is defined as

$$\omega[n] = 0.5 - 0.5 \cos\left(\frac{2\pi n}{M-1}\right), \qquad 0 \leq n \leq M-1 \tag{3.39}$$

A significant number of references to the Hann window are found in the signal processing literature, where it is utilized as one of numerous smoothing windowing procedures.

Blackman-Harris Window:

Blackman-Harris windows are a straightforward generalization of Hamming

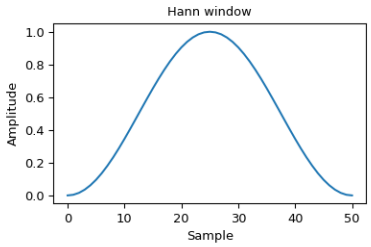

Figure 3.9: Hann window [34]

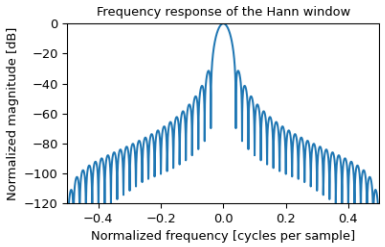

Figure 3.10: Frequency response of the Hann window [34]

windows. The Blackman-Harris window is defined as

$$\omega[n] = 0.35875 - 0.48829\cos\left(\frac{2\pi n}{M-1}\right) + 0.14128\cos\left(\frac{4\pi n}{M-1}\right) - 0.01168\cos\left(\frac{6\pi n}{M-1}\right), \qquad 0 \leq n \leq M-1 \tag{3.40}$$

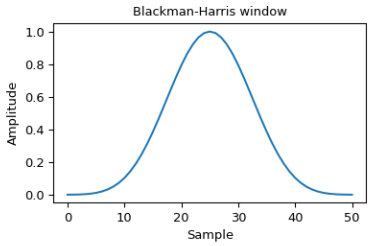

Figure 3.11: Blackman-Harris window [34]

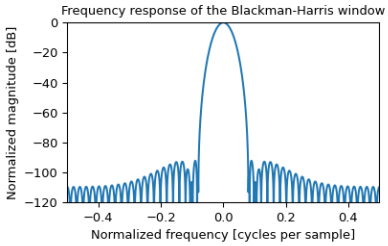

Figure 3.12: Frequency response of the Blackman-Harris window [34]

Nuttall Window:

The Nuttall window is a type of 4-term Blackman-Harris window with different coefficients. The Nuttall window is defined as

$$\omega[n] = 0.3635819 - 0.4891775\cos\left(\frac{2\pi n}{M-1}\right) + 0.1365995\cos\left(\frac{4\pi n}{M-1}\right) - 0.0106411\cos\left(\frac{6\pi n}{M-1}\right), \qquad 0 \leq n \leq M-1 \tag{3.41}$$

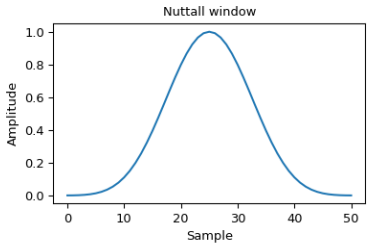

Figure 3.13: Nuttall window [34]

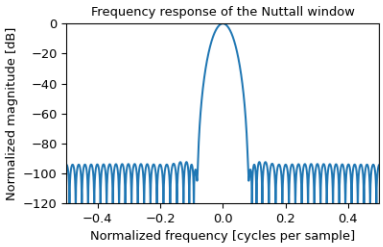

Figure 3.14: Frequency response of the Nuttall window [34]

Window Length

Instead of looking at the signal as a whole, the primary objective of windowing in the spectral analysis is to enable the user to zoom into the more minute aspects of the signal. When it comes to audio signal processing, where information such as pitch and formant frequencies are recovered by evaluating the signals via a window of a certain duration, STFT is of the utmost importance. The width of the windowing function has an effect on the way in which the signal is represented; specifically, it impacts whether there is good frequency resolution (that is, frequency components that are close together may be separated) or good temporal resolution (the time at which frequencies change) [47]. When the window is wide, the frequency resolution is improved, but the time resolution is decreased. When the window is made narrower, the temporal resolution is improved while the frequency resolution is decreased. These two types of transformations are referred to as narrowband and wideband, respectively. This is the exact reason why a wavelet transform was developed, where a wavelet transform is capable of giving good time resolution for high-frequency events and good frequency resolution for low-frequency events. In other words, a wavelet transform can give good resolution for both time and frequency. The application of this method of analysis works very well with real signals. The function of the window can be understood by examining examples of its use.

As can be seen from Figure 3.15 and Figure 3.16, if window is not preferred to use, rectangular window can be used. It is mentioned for completeness and because the rectangular window is one of the window options no tapering. When the window types are checked, for window length is equal to 1024, hamming window has side order of -43 dB and dynamic range is approximately 40dB for the first peak. When examining the rectangular window, rectangular window has side order of -21 dB. It shows that, we have better dynamic range with

hamming window.

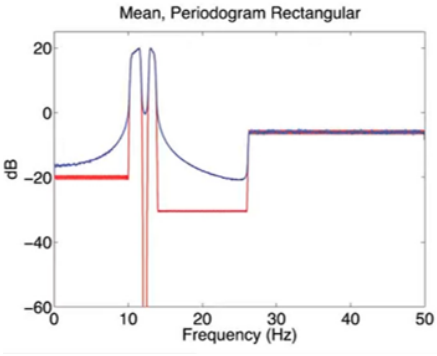

Figure 3.15: Example of Periodogram (Rectangular) [48]

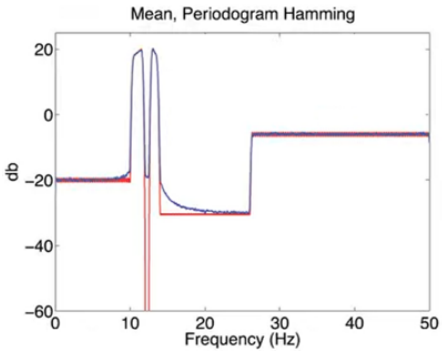

Figure 3.16: Example of Periodogram (Hamming) [48]

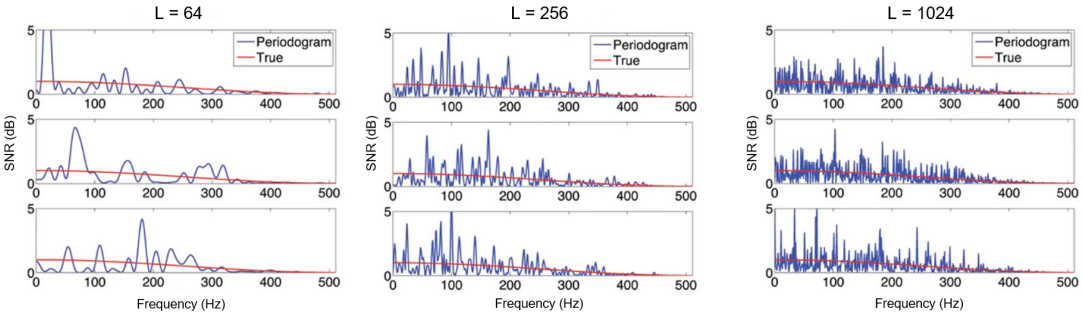

Figure 3.17: Example of Windowing (with different length) [48]

As can be seen in Figure 3.17, If window length is equal to 1024, it means that the interval is divided into one segment to estimate power spectrum density. If the segment length is equal to number of FFT, we can observe that the spectrum is quite noisy If window length is equal to 256, significant reduction on variance can be seen. If window length is down to 64, variance decreases vertically. However, the ability to distinguish the peaks has been lost and we can not longer to see the details associated to peaks.

Overlap Length

Another critical parameter of the Short-time Fourier transform is the length of overlap of individual STFT windows. Overlapping, as shown in Figure 3.4, indicates that consecutive windows overlap when obtaining data corresponding to a specific window size.

The optimal window length is determined by the parameters of the signal being analyzed. The window length influences the STFT's temporal resolution and frequency resolution. Because narrow windows have a small time

period but a broad bandwidth, they result in a fine time resolution but a poor frequency resolution. Because broad windows have a long duration but a restricted frequency bandwidth, they result in a fine frequency resolution but a coarse temporal resolution.

3.2.3 Power Spectral Density

The power spectral density (PSD) presents a graphical representation of the frequency distribution of signals that is simpler to comprehend than the DFT [29]. As the acronym indicates, it represents the fraction of the overall signal power that each frequency component of a voltage signal contributes.

Consider a random process $X(t)$ that is wide-sense stationary (WSS) and has an autocorrelation function $R_X(\tau)$. PSD of $X(t)$ is defined as the Fourier transform of $R_X(\tau)$. PSD of $X(t)$ is demonstrated by $S_X(f)$. In particular, we can write

$$S_X(f) = F\{R_X(\tau)\} = \int_{-\infty}^{\infty} R_X(\tau) e^{-2j\pi f\tau} d\tau. \tag{3.42}$$

The PSD can be visualized using a variety of approaches, such as periodogram and Welch's method, among others.

Periodogram Method

The periodogram [25] represents a nonparametric estimation of the PSD of a wss random process. The periodogram is the Fourier transform of the autocorrelation sequence estimate with a bias. Periodogram definition for a signal x_n sampled at fs samples per unit time:

$$\hat{P}(f) = \frac{\tau}{N} \left| \sum_{n=0}^{N-1} x_n e^{-j2\pi f\tau n} \right|^2, \qquad -1/2\tau < f \leq 1/2\tau \tag{3.43}$$

3.2.4 Feature Extraction

Finding the peak frequencies of forwarded signal is one of the most essential components of a decoder. The open-source software Scipy is used to locate these

frequencies. The "find peaks" function is one of Scipy's essential tools that may be used to locate signal peaks [2].

This approach compares neighboring values to identify all local maxima in a one-dimensional array. By creating conditions for the characteristics of a peak, it is feasible to select a subset of these peaks. For a reliable peak extraction, it is necessary to know its properties width, threshold, distance, and prominence. The parameters of the function are [2]:

Height: The first parameter (height) sets a threshold for the height of the signal. If a certain part of the signal is below the threshold, the function does not determine the peak there.

Threshold: The second parameter (threshold) sets the vertical distance to its neighboring samples.

Distance: The third parameter (distance) sets horizontal distance (>= 1) in samples between neighboring peaks. Initially, the smaller peaks are eliminated until the criterion is met for the remaining peaks.

Prominence: The fourth parameter (prominence) calculates the peak by taking the difference between the height of the peak and the higher minimum of the two intervals.

Width: The fifth parameter (width) estimates the peak by taking the difference between the height of the peak and the higher minimum of the two intervals.

Window length: The fifth parameter (wlen) is a sample window length that potentially limits the assessed region for each peak to a subset of the signal.

We can see in Figure 3.18 that the width parameter is not very effective in this case. This is due to the fact that if you choose a minimum width that is wide, it would be unable to follow extremely near peaks in the high frequency component of the signal. If you configure the width to be too small, you will get multiple peaks in the left half of the signal that you do not want. The same issue pertains to the distance. threshold is only able to compare itself to its immediate neighbors, which is not helpful in this situation. The answer that comes from

prominence is the most effective one.

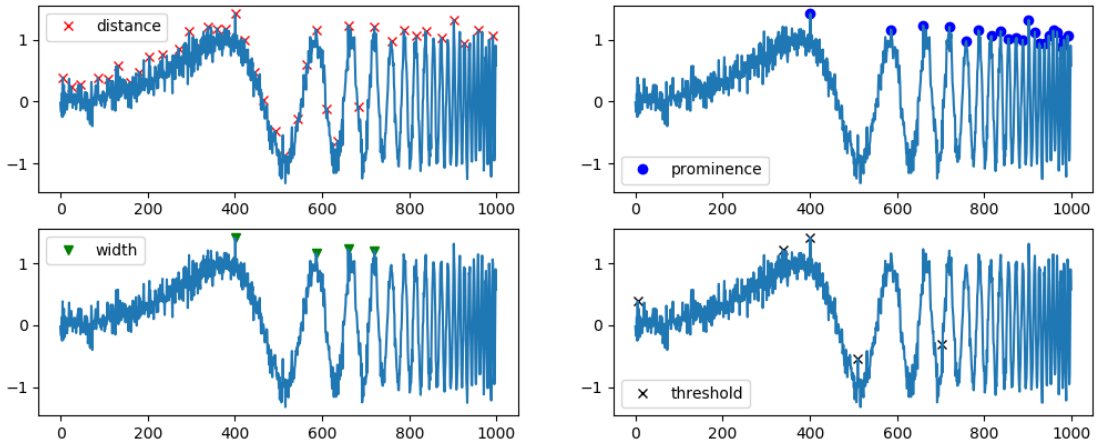

Figure 3.18: Test of find peak parameters

3.3 Scrambling

Scramblers are a type of substitution ciphers [12] that have been deemed acceptable for a variety of security requirements, including those used by cable and satellite television companies and providers of mobile phone services. Scrambler is a coding process that randomizes data streams. In addition to its usage as a cryptographic algorithm, a scrambler is frequently employed to prevent DC wander and synchronization issues in communication circuits caused by lengthy sequences of 0s and 1s. Scramblers are widely used to encrypt video and audio data for broadcasting and numerous other purposes. Scramblers are characterized by their minimal complexity and cost, rapid operation speed, and easy to implement.

Scrambling consists of a shuffling of the bits following the channel coding but prior to digital modulation, with the aim of distributing the content of a codeword within several sound fragments. Once the demodulation has been carried out, the reverse shuffling of the previous one takes place to get the code words back. In this way, in the event that a noisy intervention causes an error in demodulation, the wrong bits are distributed among several codewords, increasing the probability that these are correctable.

3.4 Speech Recognition

It was indicated in the introduction that the application would identify audio signals and then send information over the phone in a transmission. In order to recognize and transmit sound signals via mobile phone, TensorFlow Lite Maker has been implemented. The TensorFlow Lite Model Maker module makes the process of training a TensorFlow Lite model with the desired dataset more straightforward and easy to understand [3]. Transfer learning is utilized so that the required amount of training data can be reduced, hence cutting the overall training time in half. Additionally, it is easy to integrate into Android.

For a better recording of the audio signal, it is planned to use the opening marker before the transmitted audio signal. The opening marker is utilized to activate the application before the encrypted sound signal is recorded. Thus, it is ensured that the entire audio signal is recorded. In order to do that, the proper opening marker should be chosen because transmitted sound and opening sound should not interfere with each other. Therefore, it is important that the frequencies used in the opening sound are as different as possible from the frequencies in the transmitted audio signal.

4

Structure and Analysis

4.1 Overall Structure

The block diagram relating to the thesis project is shown in Figure 4.1 and Figure 4.2.

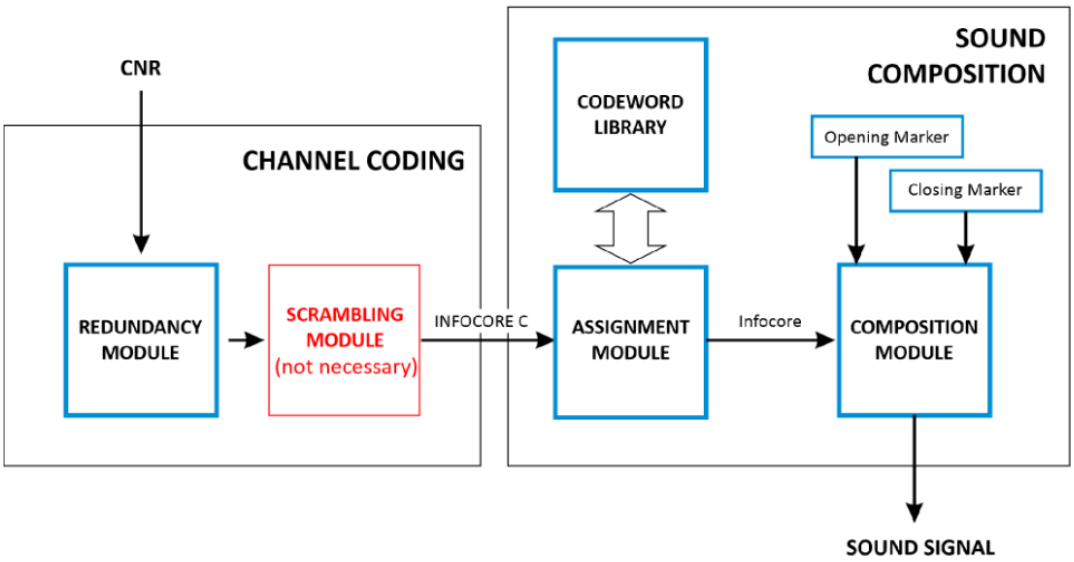

Figure 4.1: Encoder block diagram

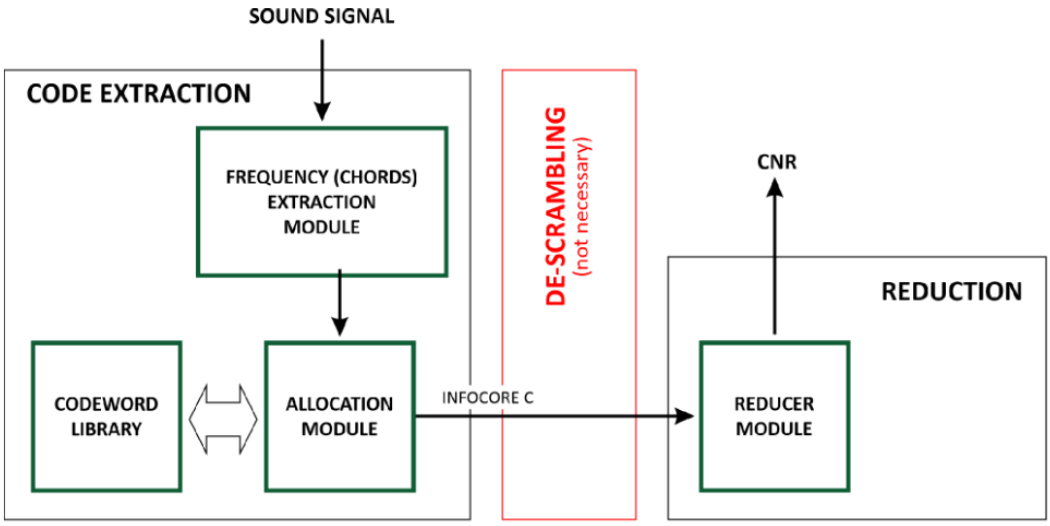

Figure 4.2: Decoder block diagram

4.1.1 Preparation of Chord Dictionary

The first step is to design a chord dictionary to encrypt the data stream bits. In order to create a chord dictionary, it is necessary to evaluate the sum of all the combinations from $1 \rightarrow k$ sound frequencies chosen from a preselection of n predetermined frequencies. The k value for this project was determined to be 4 and the n is determined to 87. In practice, the dictionary of association between the numerical values of the binary strings and the unions of sound frequencies of the chosen chords contains a number of terms (fixed n) equal to the largest multiple of 2 lower than the sum of the simple combinations.

$$C(n,k) = \frac{n!}{k!(n-k)!} \tag{4.1}$$

When the combination is calculated which is equal to $C\binom{87}{4} = 2225895$. The choice of the n sound frequencies (which make up the chords) can be made by taking into account the need for pleasant (or at least not annoying) listening. In this case, you will have to choose musical chords governed by precise harmonic laws. However, the composition of up to a maximum of 4 sound chords leads to an estimate of a maximum of 781 codewords [1], which will have the advantage of being harmonically acceptable to the ear, but the limited number of combinations would force to lengthen (and greatly) the duration of the sound signal. However, some of the sounds in this project are not designed to be musical because of

my lack of music theory knowledge. At the same time, the frequency values should not be too close to each other. As the interval between frequency values decreases, it can lead to incorrect estimations when estimating frequency. For this purpose, the frequencies were selected from a pool of 87 distinct musical frequencies, each spaced by a specific amount. In order to distribute each of 87 frequencies, k values are selected as

k_1 contains first 64 (2^6) frequencies

k_2 contains 8 (2^3) frequencies which is between $k_1 + 2 \leq k_2 \leq k_1 + 9$

k_3 contains 8 (2^3) frequencies which is between $k_2 + 2 \leq k_3 \leq k_2 + 9$

k_4 contains 4 (2^2) frequencies which is between $k_3 + 2 \leq k_4 \leq k_3 + 5$

According to this, total number of combination is

$$k_1 \times k_2 \times k_3 \times k_4 = 2^6 \times 2^3 \times 2^3 \times 2^2 = 16384. \tag{4.2}$$

The range for 87 frequencies was determined to be between 415.30 Hz and 4978.03 Hz using mentioned method. In the analysis part, it is clarified why low frequencies are not chosen.

4.1.2 Encoder

First of all, the source code is a series of randomly generated 14 bits representing the message to be sent. The source code is generated by Omniaweb and it is acquired from the server by using python. Input data is indicated as CNR in Figure 4.1.

In the coding and decoding phase (Redundancy Module and Reducer Module), the Reed-Solomon coding was used which, through the use of redundancy, allows the correction of any errors caused by noisy disturbances. First of all, the input bit is set to 14 bits, and Omniaweb decides to transmit 42 (14×3) bits. The Reed-Solomon code is designated as $(12, 6)$ where $n = 12$, $k = 6$, and $c_{exp} = 2^7$. The input bits are divided into 6 decimal parts, each less than or equal to 2^7). To correspond to 6 decimal parts, 6 more unnecessary parts were added with

the Reed-Solomon code. After that, decimal integers are converted into binary digits and back to binary digits. That way, 84 bits are achieved.

After passing through the scrambler, the input bits are transformed to decimal digits. Using the codeword library described in section 4.1.1, each of the produced components is encrypted. The sound signal is then generated utilizing frequencies related with the newly produced components. Thus, a 2-second audio signal is generated, with each component equal to 0.33 seconds.

In general, it is best to choose the sampling rate to fit the device, typically 44.1 kHz or 48 kHz [4]. However, a 16 kHz sampling rate was used in the project because of the sampling rate which is used in the sound recognition model by the TISCODE application (see section 3.4). Tensorflow-lite bases 16 kHz sampling rate for its training and if the sound is created by using a 44.1 kHz sampling rate, down-sampling or up-sampling sometimes causes a lack of information.

Algorithm 1 Encoder

$dec \leftarrow reshape(input)$ {The input bits are converted into decimal values}
$encoded \leftarrow RSEncode(dec)$ {The decimal bits pass through Reed-Solomon codes and redundant component are created}
$encoded2 \leftarrow reshape(encoded)$ {Encoded symbols are converted into binary values}
$scr \leftarrow scrambler(encoded2)$ {The reshaped bits pass through the scrambler}
$x \leftarrow reshape(scr)$ {Scrambled bits are converted into decimal integers}
for $i \leftarrow (0 \rightarrow length(x))$ {Integers pass through the alphabet, and for each integer, four frequencies are obtained} **do**
  $freq[i] = alphabet[i, x]$
end for
$y \leftarrow createSound(freq)$

4.1.3 Decoder

The decoder consists of two subsections. The first subsection is the recognition of the sound heard from a device with a microphone, the second part is the decoding of this recognized sound.

Encoded Signal Localization

A microphone-equipped gadget listens the sound produced by the encoder. The audio signal is recognized by the AppyTech-developed TISCODE application. AppyTech utilizes the tenforflow-lite module for sound recognition, as

stated in section 3.4. This application recognises the combination of the opening marker and the created sound then begins to record the audio signal. The application begins recording three seconds after detecting the opening marker. This 4-second audio stream is then sent to the decoder.

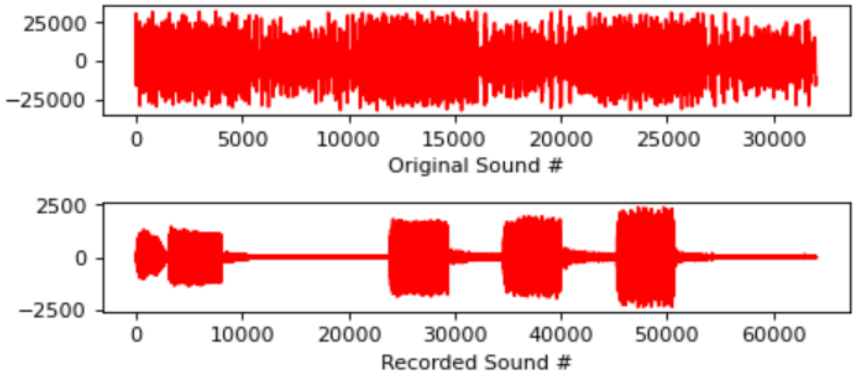

Figure 4.3: Example of Recorded Sound

The audio signal is firstly converted from pcm to wav format. The converted sound can be seen in Figure 4.3. The data between the 24k and 56k is the encoded sound signal that needs to be decoded. The data of the openning marker can be seen at the beginning of the recorded sound. As can be seen in Figure 4.3, there is a 10 times difference between the amplitude of produced sound and the amplitude of the recorded sound. Due to the Sox sound processing function utilized by the sound recognition method [5], the sound signal appears to have 10 times less amplitude. Since sound recognition is not the focus of this thesis, it has not been highlighted.

The converted audio signal is then trimmed to capture all of the information contained in the four seconds of audio. As stated in the encoder section, the actual duration of the information sound is two seconds, and periodogram method has been used to eliminate the redundant two second from the data. Received sound has 64k sample which equals to 4 seconds. The information part which is called "rinfo" part has exact 31998 sample which is inside the received sound. However location is not certain. To estimate the location of the rinfo piece, the Scipy periodogram method is utilized [6]. Periodogram estimates the power spectral density by segmenting the data into overlapping segments, calculating an adjusted periodogram for each segment, and averaging the modified periodograms. Periodogram is utilized to determine the PSD of each note frequency (which is used in alphabet) for each 32k sample of the 64k

	Number of peak frequency					
	4	5	6	7	8	9
Mean decoding time (sec)	0.72	0.86	1.02	1.37	3.81	9.67
Decoding rates (for 50 attempts)	52%	64%	78%	84%	86%	88%

Table 4.1: Number of peak frequency comparison

sample set. Following the calculation, the section with the highest power for the note frequencies is selected. After the estimation of the highest power part in the four-second sound, the trimmed part is found as in Figure 4.4.

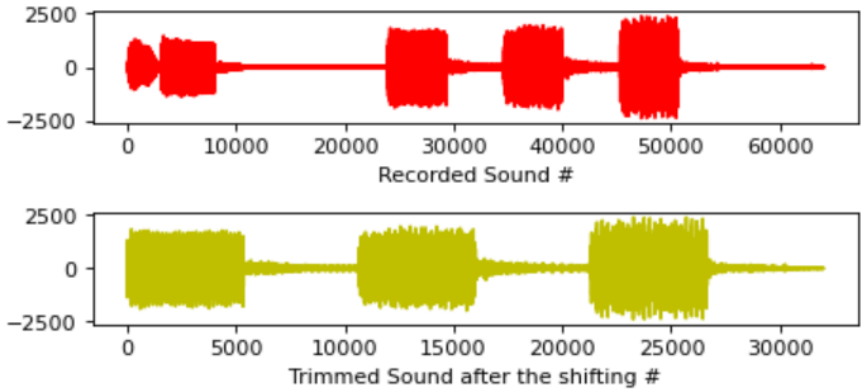

Figure 4.4: Example of Trimmed Sound

As shown in Figure 4.4, the power appears to be relatively low in locations where low frequency is utilized. A higher-frequency wave with the same amplitude has greater power than a lower-frequency wave with the same amplitude, while a higher-amplitude wave with the same frequency has greater energy.

Feature Extraction

After the estimation of the sound signal location, the decoder starts to decode the trimmed sound signal by using the spectrogram [7]. The 32k sample data is initially divided into six parts. This is due to the fact that every 0.33 seconds of data corresponds to 5333 data samples. Using the spectrogram, the first seven peak frequencies of each 0.33 second of data are determined. The reason behind the selection of seven frequencies is presented in Table 4.1. Increasing the number of selected peak frequencies improves the decoder's performance. However, decoding processing time increases as well.

Utilizing a variety of window functions, the spectrogram is used to locate the sound signal's frequencies as well as the magnitudes of those frequencies.

The Scipy spectrogram will return the following values: PSD, complex, magnitude, angle, and phase are the acronyms for these [7]. The output of STFT without any padding or boundary extension is comparable to the value referred to as "complex." The value that is returned by the'magnitude' command is the absolute magnitude of the STFT. Both 'angle' and 'phase' will give you back the complicated angle that the STFT has, with or without unwrapping, depending on which one you use. The 'magnitude' is preferred as a parameter in this thesis and it is utilized.

The window length is set as three parameters in a decoder. These are 5333, 4000, and 2667. Firstly, the decoder starts to decode the sound signal by setting window length 5333, if it can not find any decodable bit string then it tries to decode the sound signal again by using 4000 window length and then 2667 window length. After that, the "find peaks" method in scipy is used to locate the peaks in the spectrogram [7]. For the figures in below (between Figure 4.5 and Figure 4.10), the used bit string and the frequencies in the created sound signal are:

Rinfo: 00000000101010001000100000010010101001000 1

Peak Frequencies : [164.81 185 293.66 329.63]
Peak Frequencies : [659.25 739.99 830.61 1046.5]
Peak Frequencies : [174.61 196 246.94 277.18]
Peak Frequencies : [220 246.94 311.13 392]
Peak Frequencies : [392 587.33 698.46 830.61]
Peak Frequencies : [349.23 415.3 587.33 639.8]

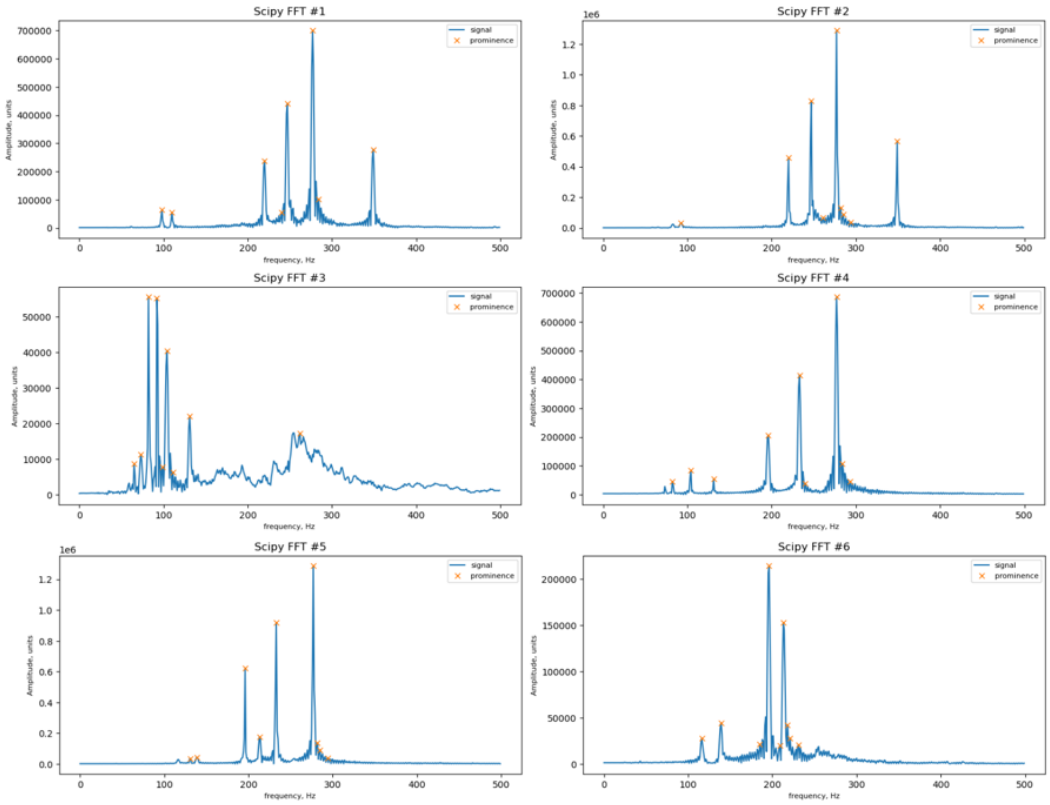

Figure 4.5: Graphical Representation of Scipy FFT

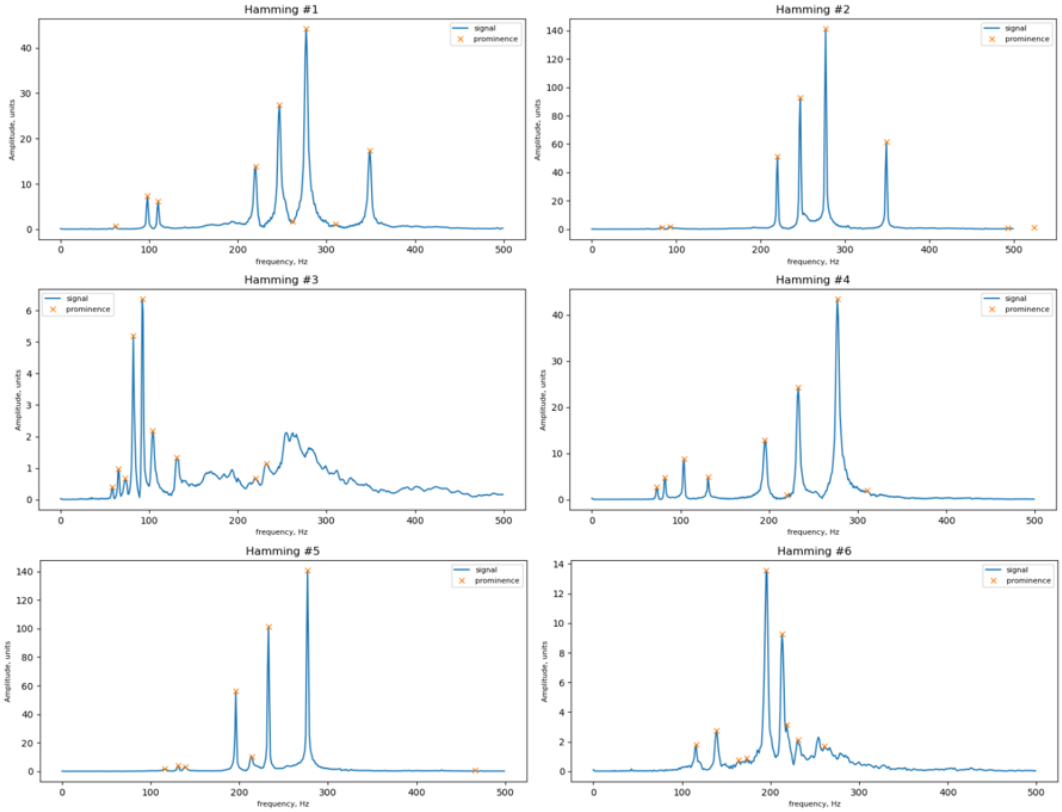

Figure 4.6: Graphical Representation of Scipy STFT (Hamming)

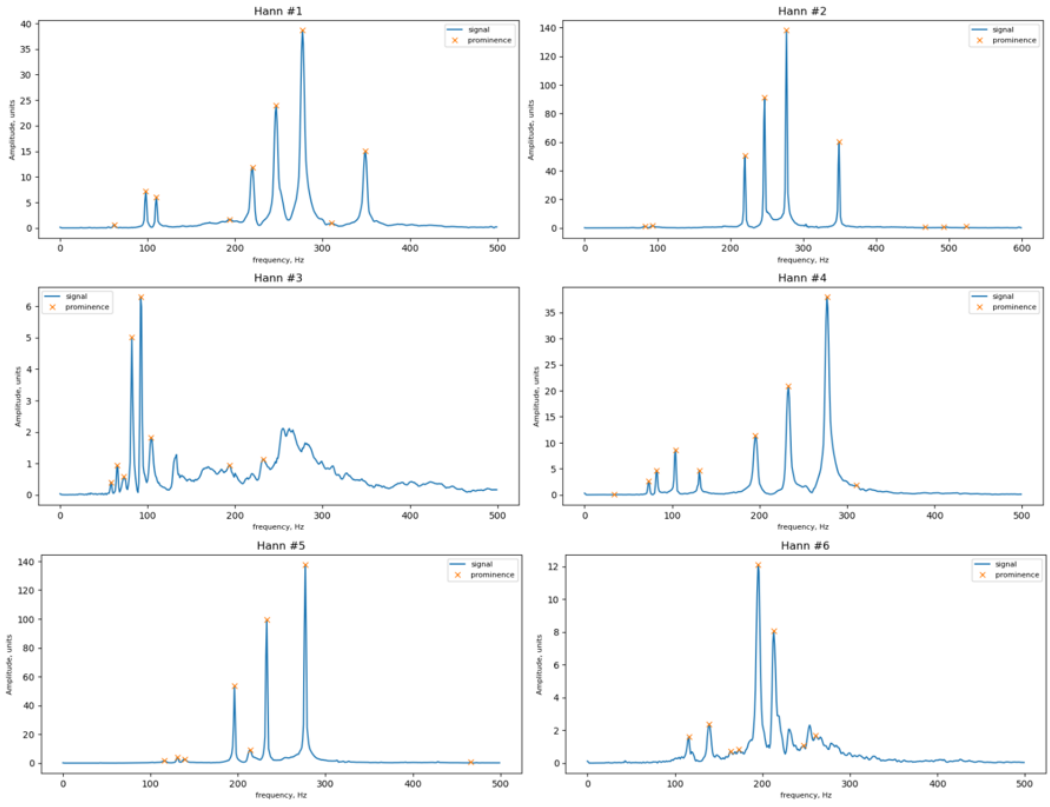

Figure 4.7: Graphical Representation of Scipy STFT (Hann)

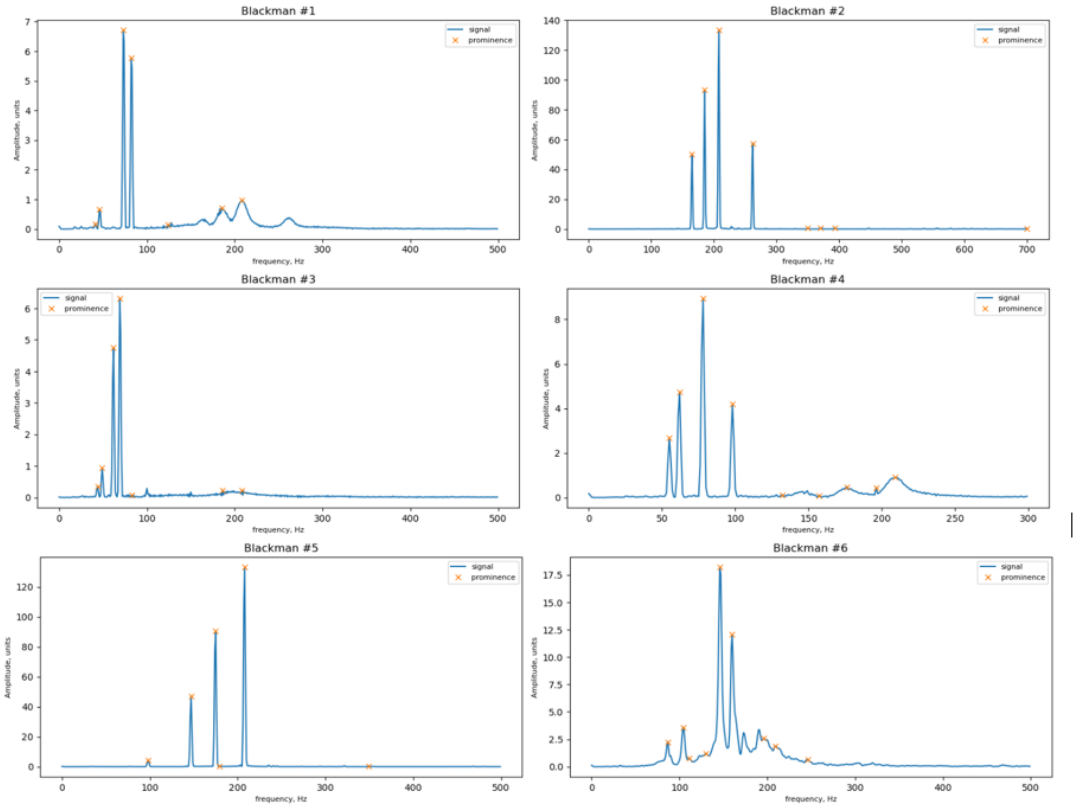

Figure 4.8: Graphical Representation of Scipy STFT (Blackman)

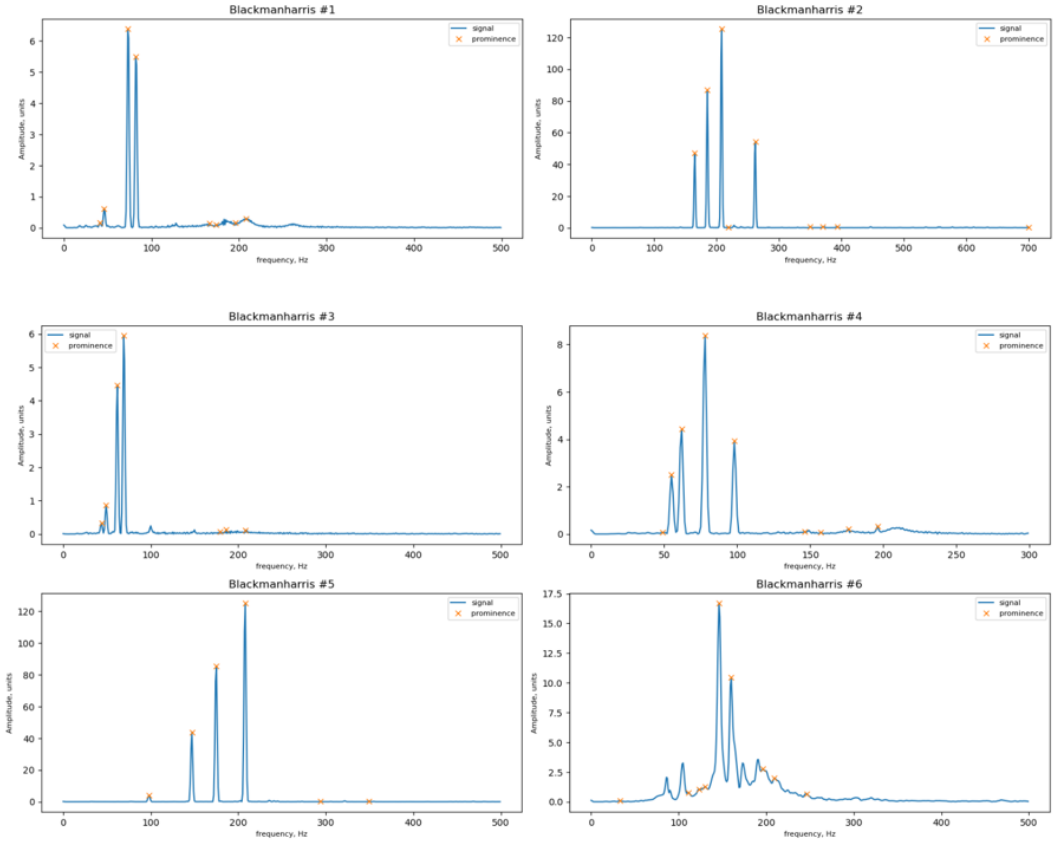

Figure 4.9: Graphical Representation of Scipy STFT (Blackman-Harris)

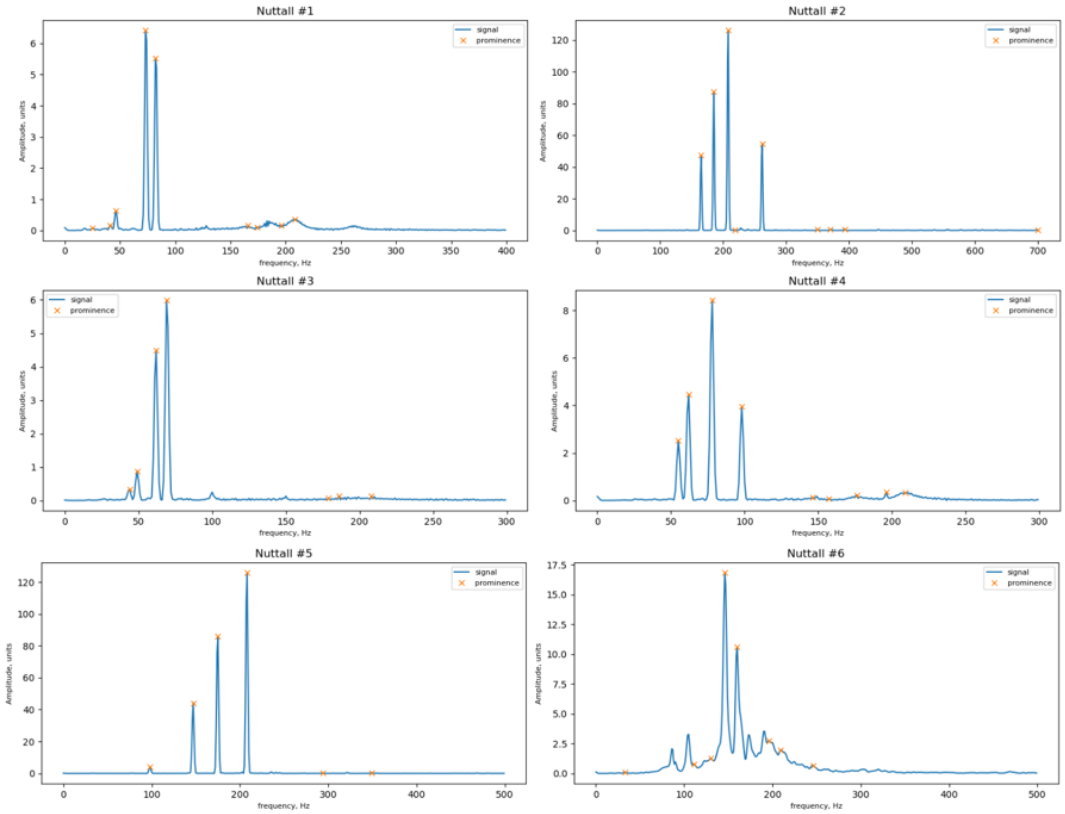

Figure 4.10: Graphical Representation of Scipy STFT (Nuttall)

	Founded Frequeny (Hz)												
Peaks 1	160.7	**162.81**	**188.21**	213.9	254.54	**291.3**	**327.87**	359.1	381.23	**439.61**	599.89	**659.25**	**830.6**
Peaks 2	**174.1**	238.7	**276.31**	**346.66**	599.89	650.54	**659.25**	732.21	**739.99**	806.2	**830.6**	1020.3	**1042.8**
Peaks 3	169.6	**174.1**	190.7	**196**	238.7	**246.9**	254.4	**276.31**	303.6	**310.8**	339.9	**351.1**	**369.99**
Peaks 4	190.7	**220**	**249.94**	254.4	303.6	**310.8**	339.9	**351.1**	**369.99**	383.65	**392**	505.8	**586.6**
Peaks 5	**309.6**	339.8	**349.23**	360.36	**369.99**	505.7	536.67	**586.6**	606.9	**639.80**	672.52	**698.46**	**830.61**
Peaks 6	227.46	**309.6**	339.8	**349.23**	**369.99**	401.32	**415.3**	**439.61**	505.7	531.61	**587.33**	599.89	**639.8**

Table 4.2: Example of Extracted Frequencies

Afterwards, intervals are defined so that note frequencies can be determined. For instance, if the extracted frequency is quite close to the note frequency, then the extracted frequency will be used as the note frequency. If there is a gap of more than 4 Hz between the founded frequency and the note frequency, the frequency in consideration is not chosen to be a note frequency. Therefore, the frequencies are removed one by one until a final set of seven peak frequencies are chosen. For instance, we suppose that the alphabet contains frequencies between 138.59 Hz and 7092 Hz and the function finds thirteen peak frequencies by setting prominence parameter then we find the frequencies as in Table 4.2.

For example, the frequencies that should be found among 13 peak frequencies in "Peaks 1" (164.81 Hz, 185 Hz, 293.66 Hz, 329.63 Hz) were found. In addition, for example, 213.9 Hz on "Peaks 1" is a frequency not found in the alphabet. The alphabet frequency closest to 213.9 Hz is 207.65 Hz and 220 Hz. If we observe the difference between them, $213.9 - 207.65 = 6.25$ Hz and $220 - 213.9 = 6.1$ Hz it is not among the alphabet frequencies of 213.9 Hz, since the difference between them is more than 4 Hz. Beginning with this principle, all detected peak frequencies are eliminated one by one until only seven peak frequencies remain. The seven frequencies found for each of the 5333 sampled data are highlighted in Table 4.1.

The selection of seven frequencies is necessitated by the fact that the decoding time increases excessively with more frequency combinations. Since it takes less than one second to decode the combination of seven frequencies, seven peak frequencies were chosen during decoding. After that comes the combining of these different frequencies. The Reed-Solomon decoder is utilized on the data. Reed-Solomon is going to begin decoding the specified string if the Reed Solomon syndrome discovers only three or fewer values that are greater than 0. Because the The utilized Reed-Solomon check system can only find decodable string if the data are 100% accurate. The use of Reed-Solomon syndrome allows that the decoder does not have to verify every possible combination in order to find a string that is 100% correct. The decoder is able to locate a string that can

be decoded by checking a reduced number of possible possibilities.

4.2 Experimental Results

In this section, the previously mentioned procedures are evaluated under various scenarios. The first subsection discusses noise environments and their characteristics. In the second subsection, the periodogram's capacity to detect sound from a microphone-equipped device is evaluated. In the third subsection, the ability of the window functions is tested. The test was carried out according to the window types and the performance of the window type at different noise types.

Noise

The decoder was tested using different types of sounds. Sounds are collected from [8]. This subsection shows the frequency range of the noises and will explain why low frequencies are not preferred when choosing a frequency range for the alphabet. Five different noise environments were selected to test the decoder. These are cafe environment, bank ambiance, airport terminal, office environment and car horn noise. The noise signals used can be seen more clearly in the graphs below.

The primary purpose behind the selection of various types of noises is to explore how the extraction of features is affected by the noises that are typically encountered in daily life. For instance, cafe background noise has significant amplitude between 190 Hz and 390 Hz. The office background noise shows its effect between 60 Hz and 240 Hz. The car horn has a variety on frequencies. The highest amplitudes are seen between 340 Hz - 390 Hz and 3000 Hz - 3500 Hz. The bank ambience has a huge impact between 110 Hz and 700 Hz. The airport terminal noise has an impact around 300 Hz. For all the noises, the highest amplitudes are observed around 300 Hz. In conclusion, we can observe that, the frequencies below the 350 Hz should be used in an alphabet.

```
1st Quartile : -35.34523815907035
2nd Quartile : -28.610327200460034
3rd Quartile : -23.571802280858172
Mean : -30.580906522344403
Median : -28.610327200460034
Standard Deviation : 10.225064806735613
Variance : 104.5519503019432
```

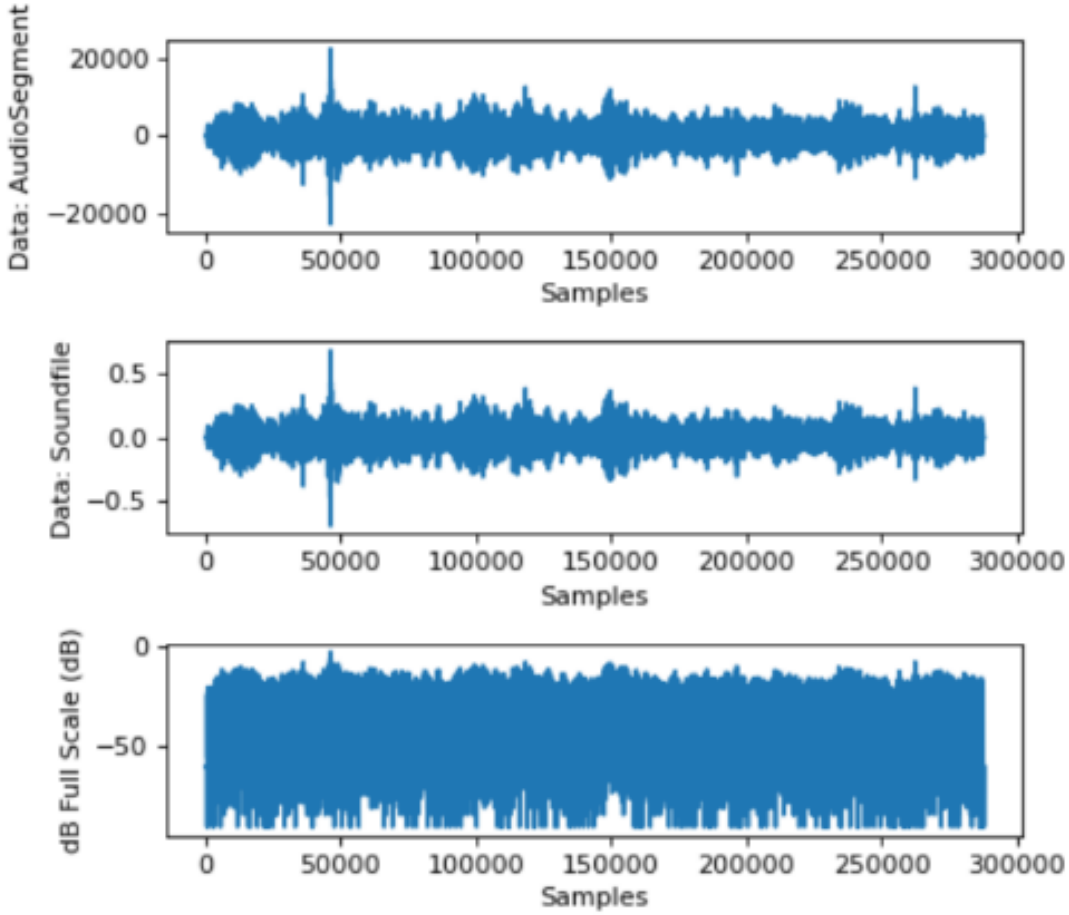

Figure 4.11: Graphical Representation of Cafe Background Noise

```
1st Quartile : -40.09809849506212
2nd Quartile : -33.138254747801575
3rd Quartile : -28.164799306236993
Mean : -35.41606435242237
Median : -33.138254747801575
Standard Deviation : 10.549649252915856
Variance : 111.29509935954808
```

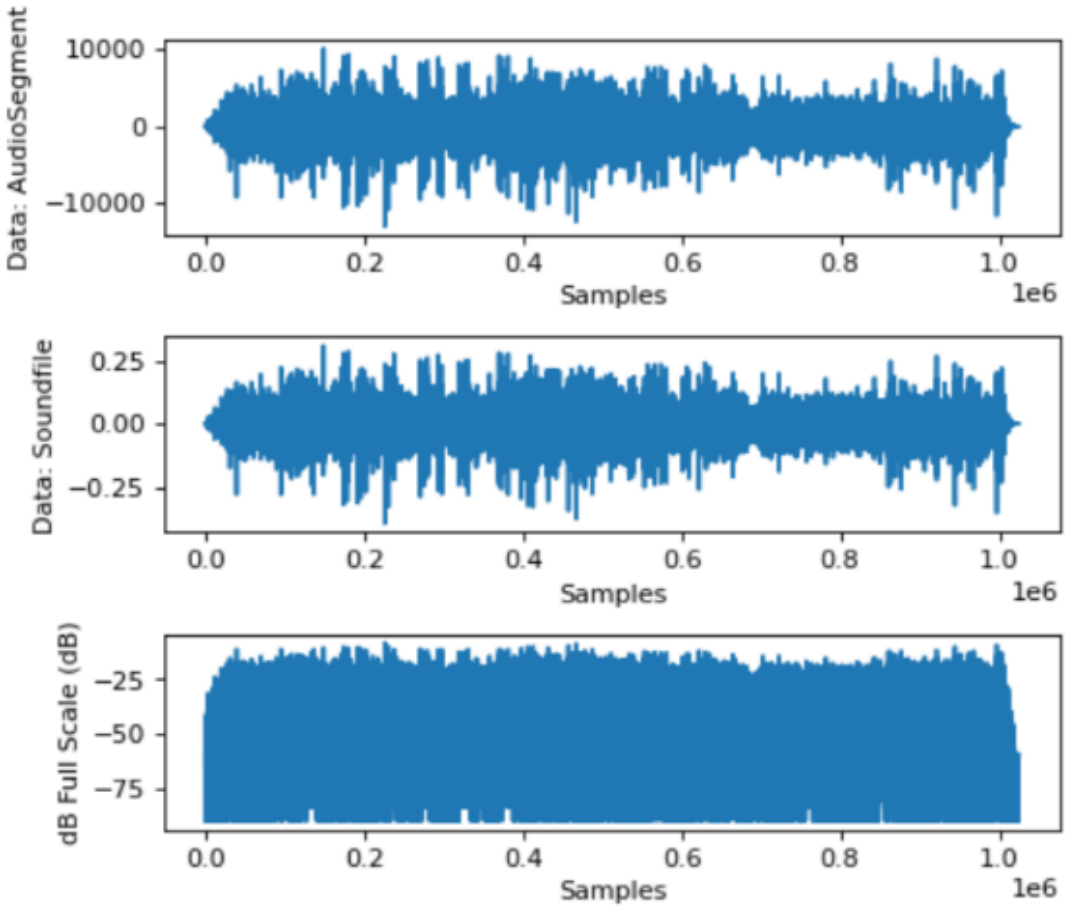

Figure 4.12: Graphical Representation of Office Background Noise

```
1st Quartile : -38.3113372577206
2nd Quartile : -31.156852957992452
3rd Quartile : -25.694912426942977
Mean : -33.15356520679605
Median : -31.156852957992452
Standard Deviation : 10.75955652721527
Variance : 115.76805666234074
```

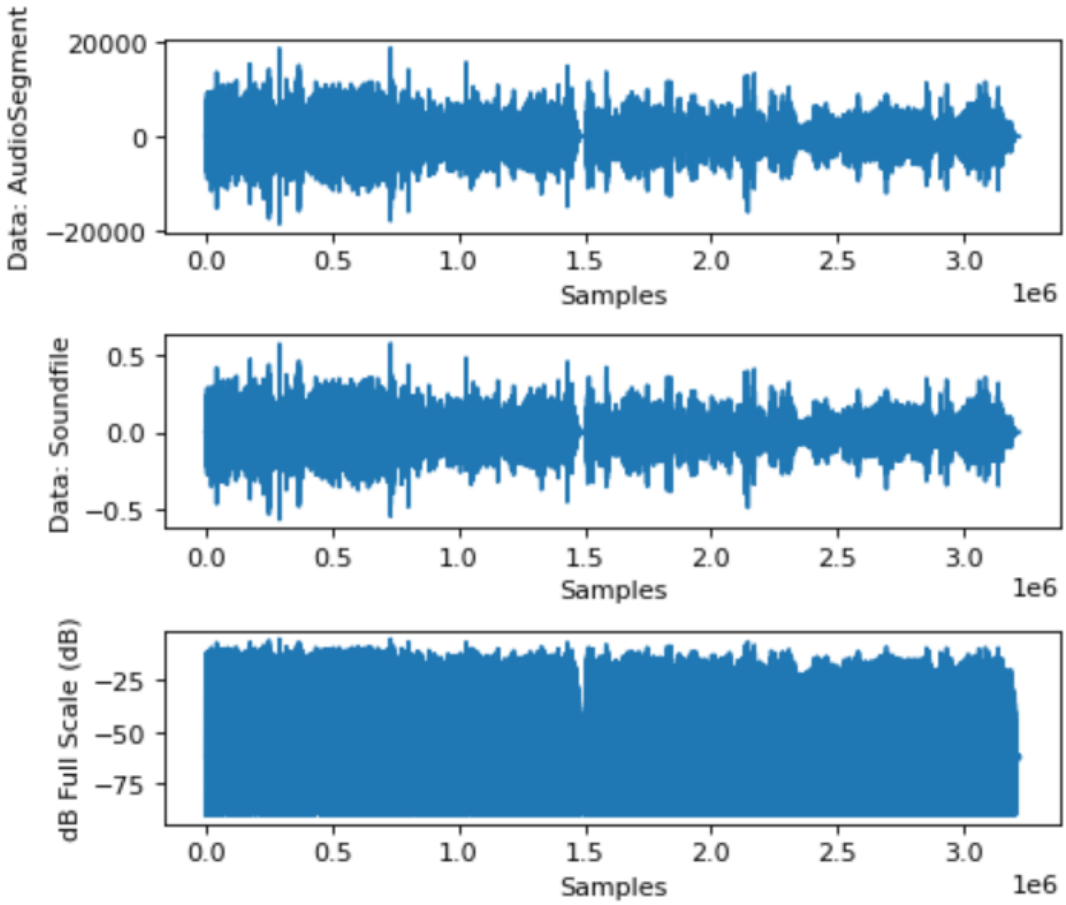

Figure 4.13: Graphical Representation of Bank Ambience Noise

```
1st Quartile : -36.793431865712655
2nd Quartile : -30.231175374456146
3rd Quartile : -25.458170133226673
Mean : -32.21031053111722
Median : -30.231175374456146
Standard Deviation : 9.775447991365011
Variance : 95.55938343188222
```

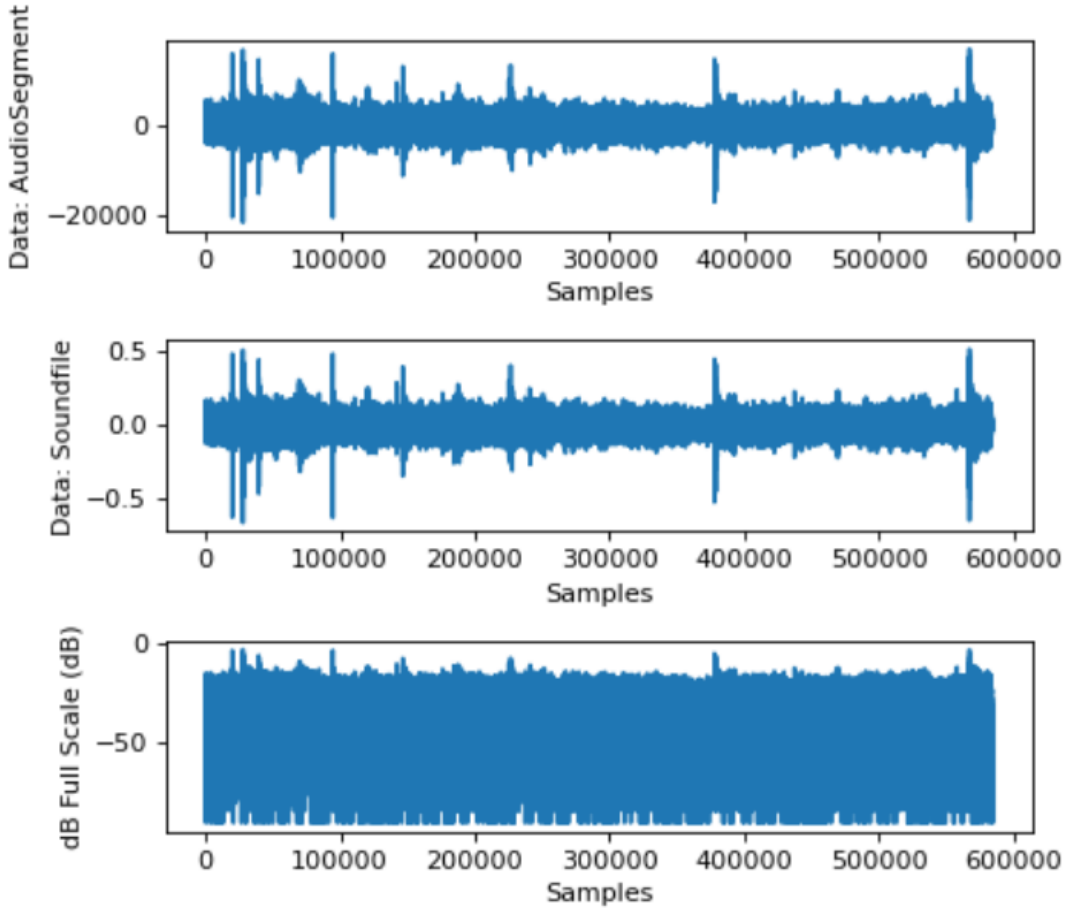

Figure 4.14: Graphical Representation of Airport Terminal Noise

```
1st Quartile : -44.116395350676385
2nd Quartile : -29.35571480716316
3rd Quartile : -20.71749249569459
Mean : -33.48367770744433
Median : -29.35571480716316
Standard Deviation : 15.974877936260649
Variance : 255.1967250784273
```

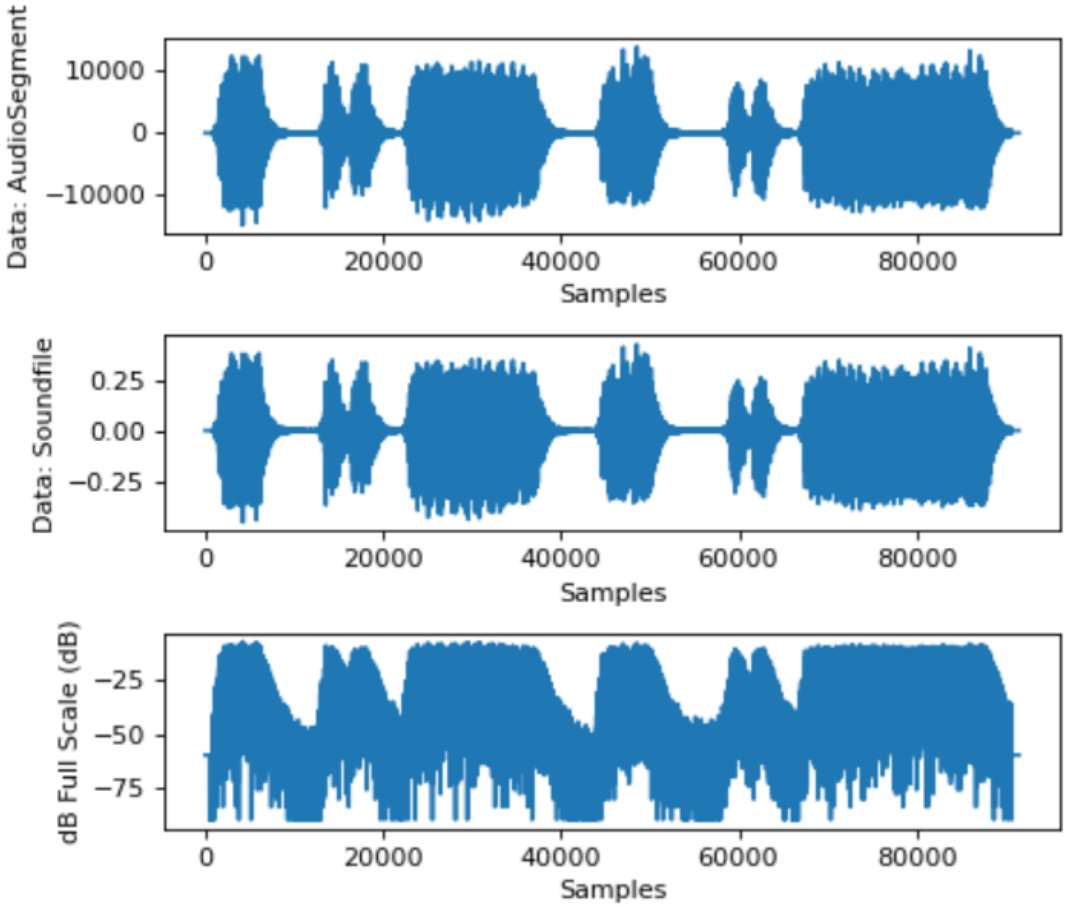

Figure 4.15: Graphical Representation of Car Horn Noise

Analysis of Encoded Sound Detection

As explained in encoded audio localization, the recorded audio signal is trimmed to capture audio information. According to the encoder part, the actual duration of the information sound is two seconds, and the periodogram method was utilized to eliminate the extra two seconds of data. Figure 4.16 shows how accurate the sound clipping was after 50 trials using different window functions.

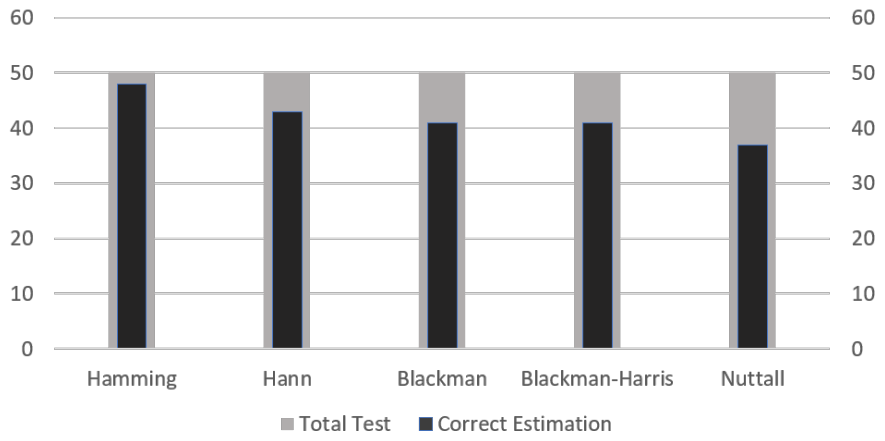

Figure 4.16: Test results of encoded sound detection

There are two key reasons why sound clipping is not always appropriate. First, sound recognition cannot capture the encrypted sound in its entirety. Second, if the surrounding environment is excessively noisy, the audio cannot be clipped correctly. Figure 4.17 demonstrates that the complete encrypted audio stream, which typically consists of 32k sample data, is not recorded correctly. Only 24k sample data was collected for a portion of the sound signal that should have had 32k sample data. An example of the correct clipping operation can be observed in Figure 4.4.

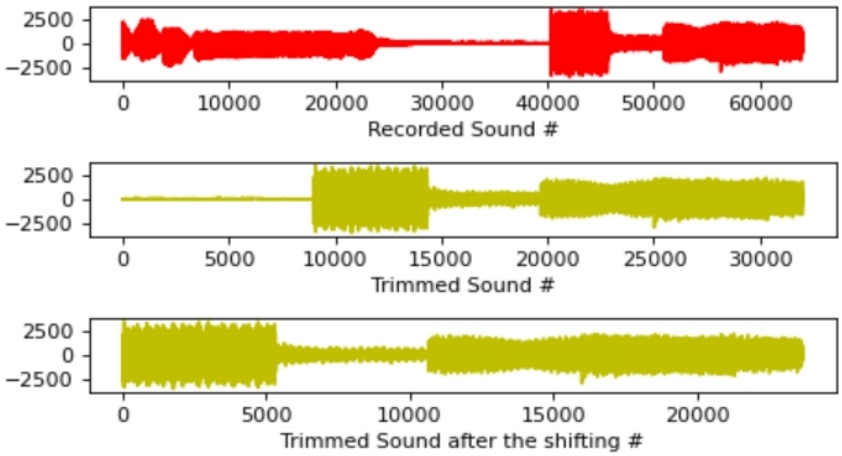

Figure 4.17: Example of missing recorded sound

Analysis of the Decoder

The decoder was analyzed using the mentioned sounds and several windowing techniques for two distinct frequency ranges. Using low frequencies in a variety of noisy environments has been analyzed in terms of the outcomes observed. In the initial investigation, the audio signal was generated using frequencies ranging from 164.81 Hz to 1046.5 Hz. In the second examination, frequencies between 415.3 Hz and 3228 Hz were selected.

Test 1 (164.81 Hz - 1046.5 Hz)

The data string used is the same as the data string mentioned in feature extraction part of decoder section. The data string is:

Rinfo: 00000000101010001000100000010010101001000 1

Peak Frequencies : [164.81 185 293.66 329.63]
Peak Frequencies : [659.25 739.99 830.61 1046.5]
Peak Frequencies : [174.61 196 246.94 277.18]
Peak Frequencies : [220 246.94 311.13 392]
Peak Frequencies : [392 587.33 698.46 830.61]
Peak Frequencies : [349.23 415.3 587.33 639.8]

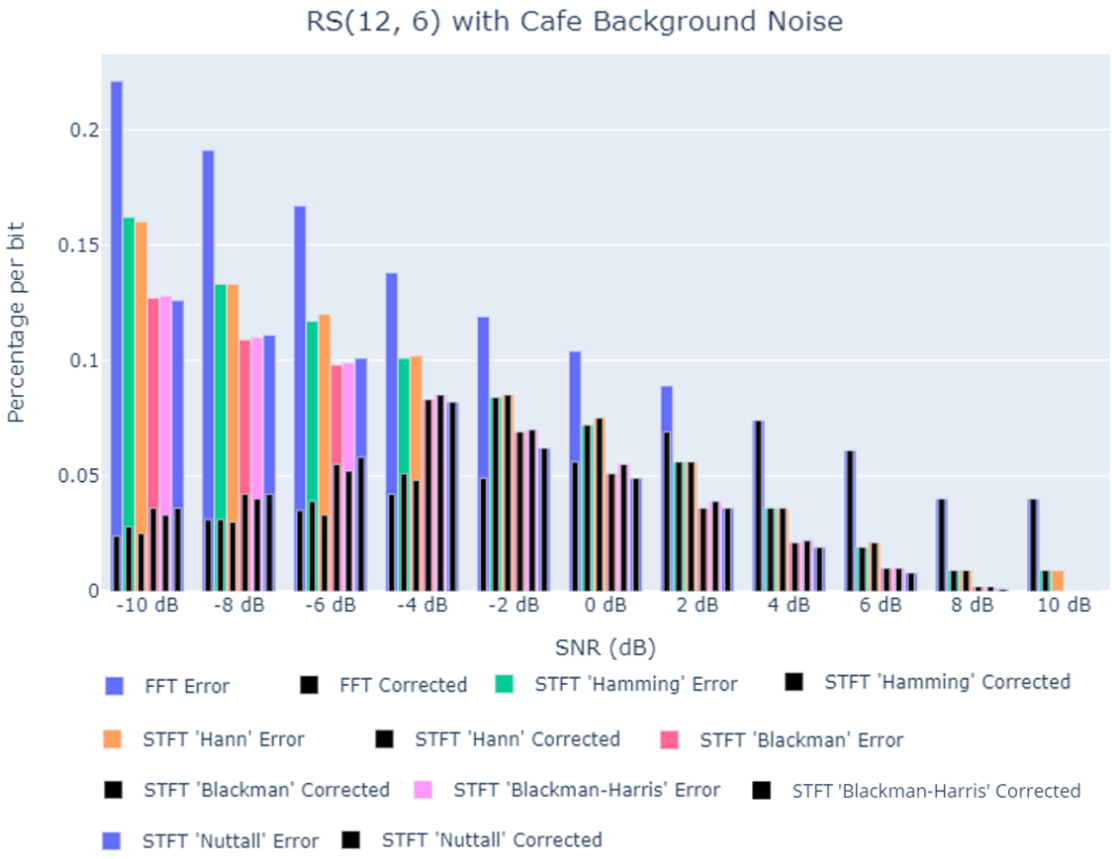

Figure 4.18: Cafe Background Noise Performance on Low Frequency

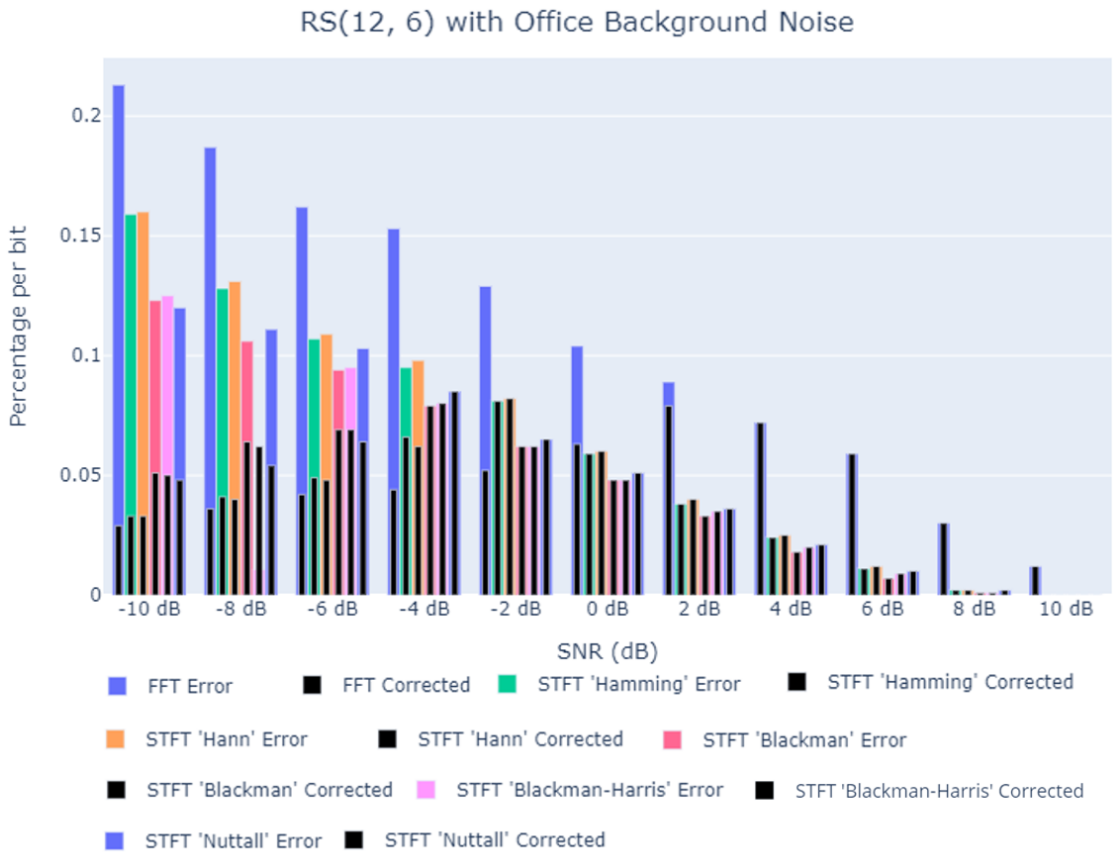

Figure 4.19: Office Background Noise Performance on Low Frequency

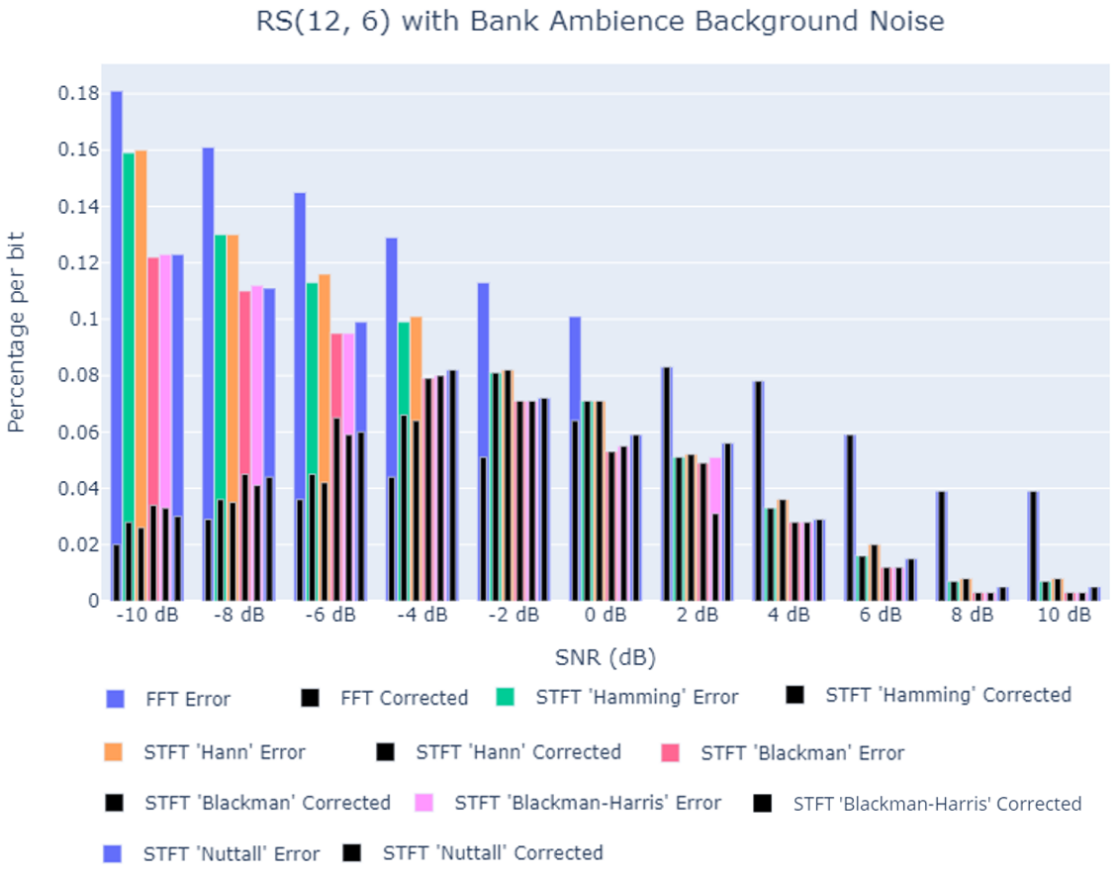

Figure 4.20: Bank Ambience Background Noise Performance on Low Frequency

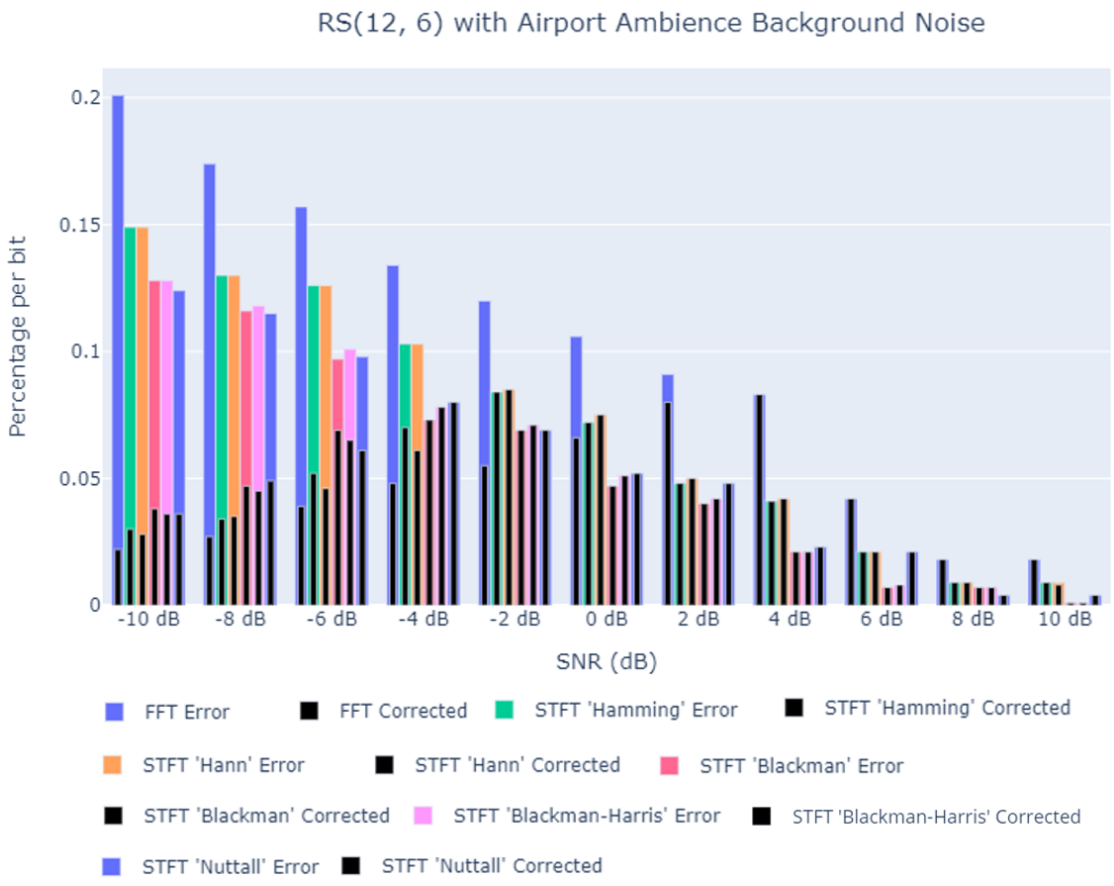

Figure 4.21: Airport Terminal Background Noise Performance on Low Frequency

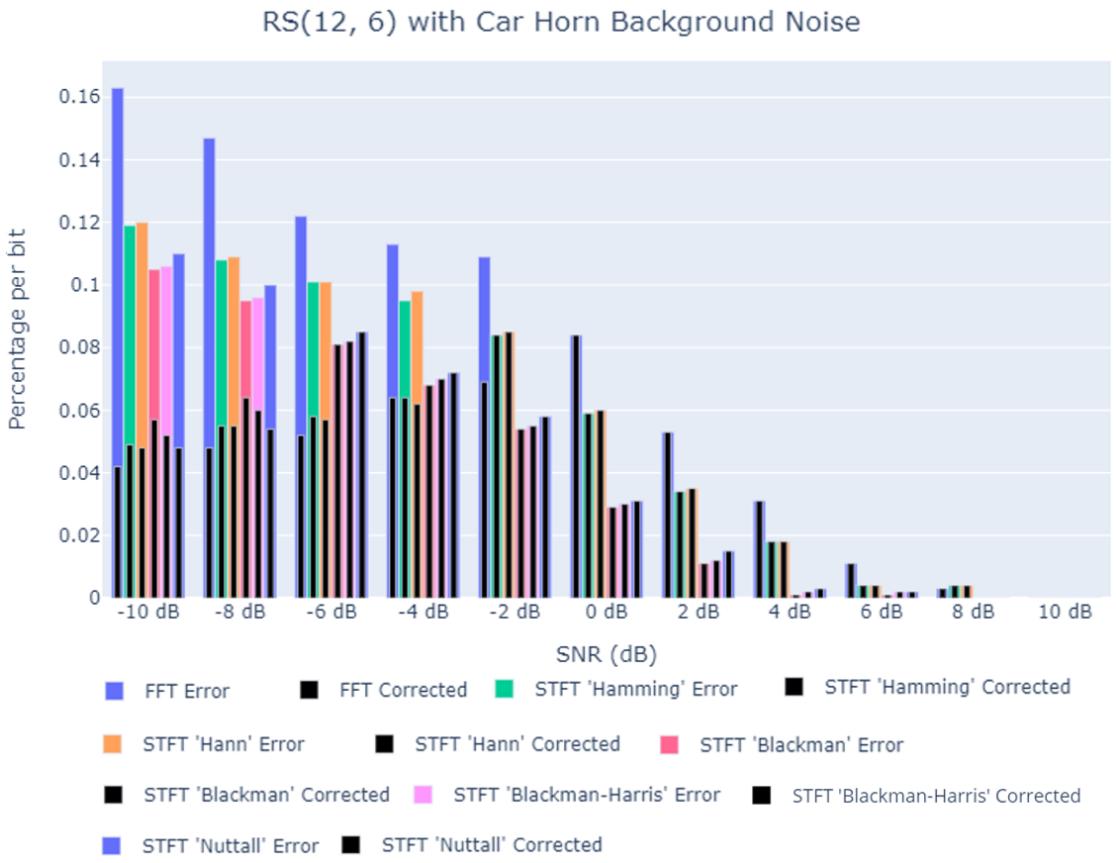

Figure 4.22: Car Horn Background Noise Performance on Low Frequency

As can be seen in the tables that are located above, FFT provides the right bits when the SNR is 4 dB in environments such as cafes, offices, and airports because it does not use any kind of noise reduction method. It produced better results depending on the ambient noise present in the environment of the bank and the car horn.

When we observe the STFT 'Hamming' and the STFT 'Hann' in the figures, we can see that these two windowing method give the bits appropriately at the -2 dB level. It is possible for us to draw the conclusion that these two approaches produce outcomes that are 6 dB more favorable than FFT.

If we look closely at the STFT 'Blackman', 'STFT 'Blackman-Harris,' and 'STFT 'Nuttall' window methods, it is noticeable that all three of these window approaches give accurate bits at the -4 dB level.

STFT "Blackman," STFT "Blackman-Harris," and STFT "Nuttall" are able to rectify faults even at -4 dB, as was mentioned in the section in overall structure. This is possible due to the fact that frequency ranges that are not in the alphabet are excluded while detecting frequencies. As can be seen in the table that have been presented above, when the error rate is approximately 8.5%, it is clear that the error has been totally repaired.

As was discussed in the noise section, each noise environment has a frequency range that predominately dominates. Therefore, frequencies below 400 Hz in the bit strings are unable to deliver accurate results in surroundings with a significant level of noise.

Test 2 (415.3 Hz - 3228 Hz)

For the second test, the following data is the bit string and the frequencies used to store this string are given. These are:

Rinfo: 011000101011100011001001111100101010110001

Peak: [415.3, 466.16, 698.46, 739.99]
Peak: [493.88, 554.37, 783.99, 987.77]
Peak: [1432.3, 1760.0, 1918.6, 2217.46]
Peak: [415.3, 639.8, 739.99, 880]
Peak: [415.3, 622.25, 720.4, 850.0]
Peak: [2217.46, 2489.02, 2876, 3228]

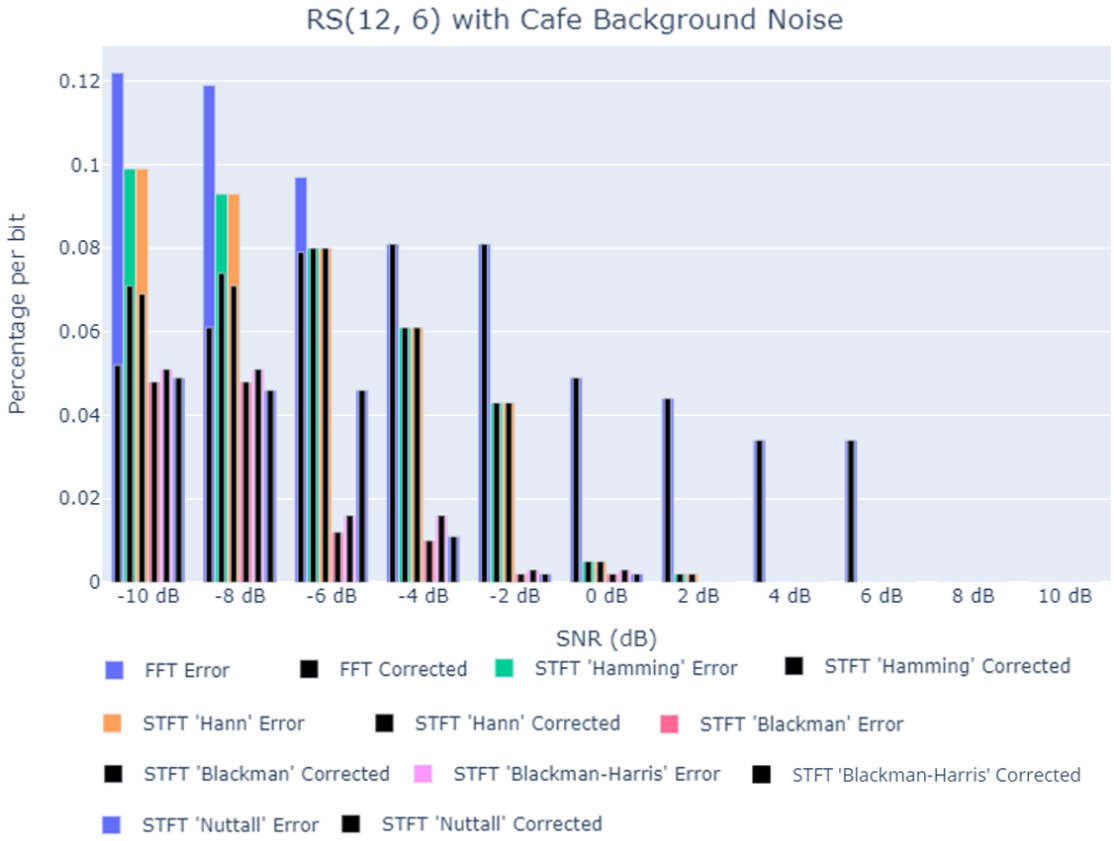

Figure 4.23: Cafe Background Noise Performance on Medium Frequency

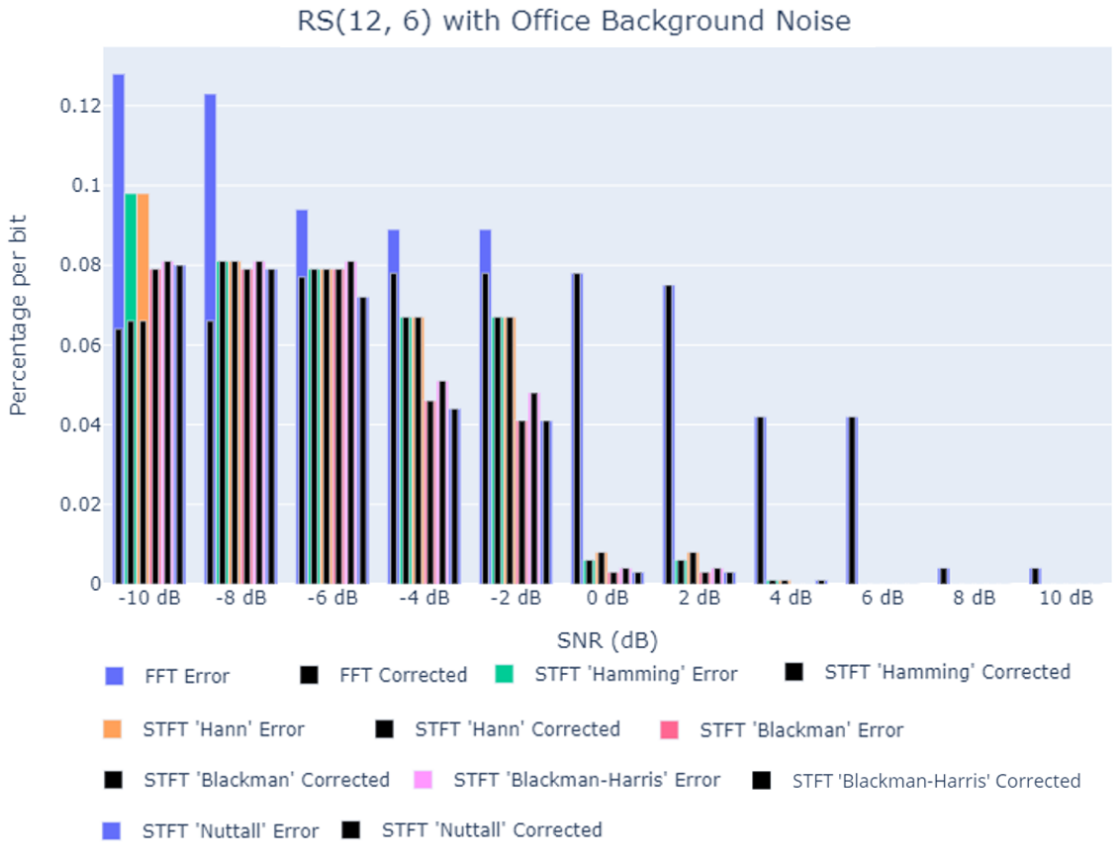

Figure 4.24: Office Background Noise Performance on Medium Frequency

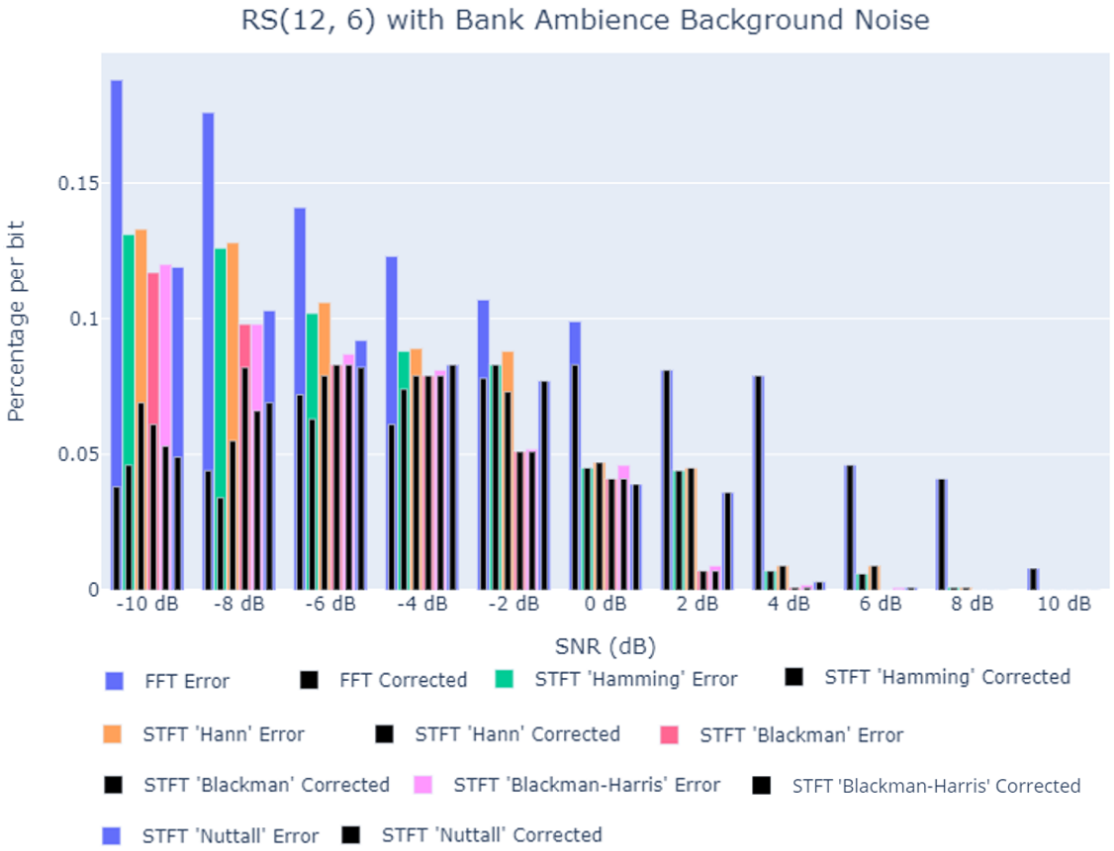

Figure 4.25: Bank Ambience Background Noise Performance on Medium Frequency

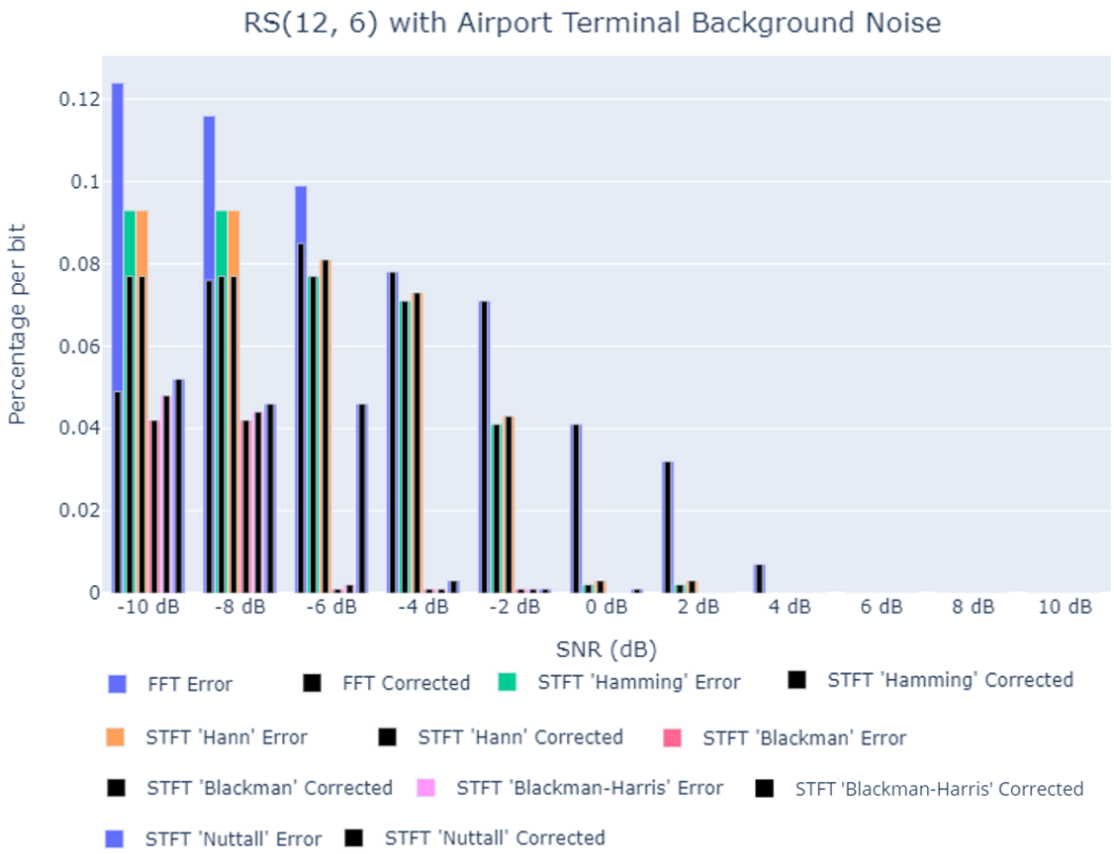

Figure 4.26: Airport Terminal Background Noise Performance on Medium Frequency

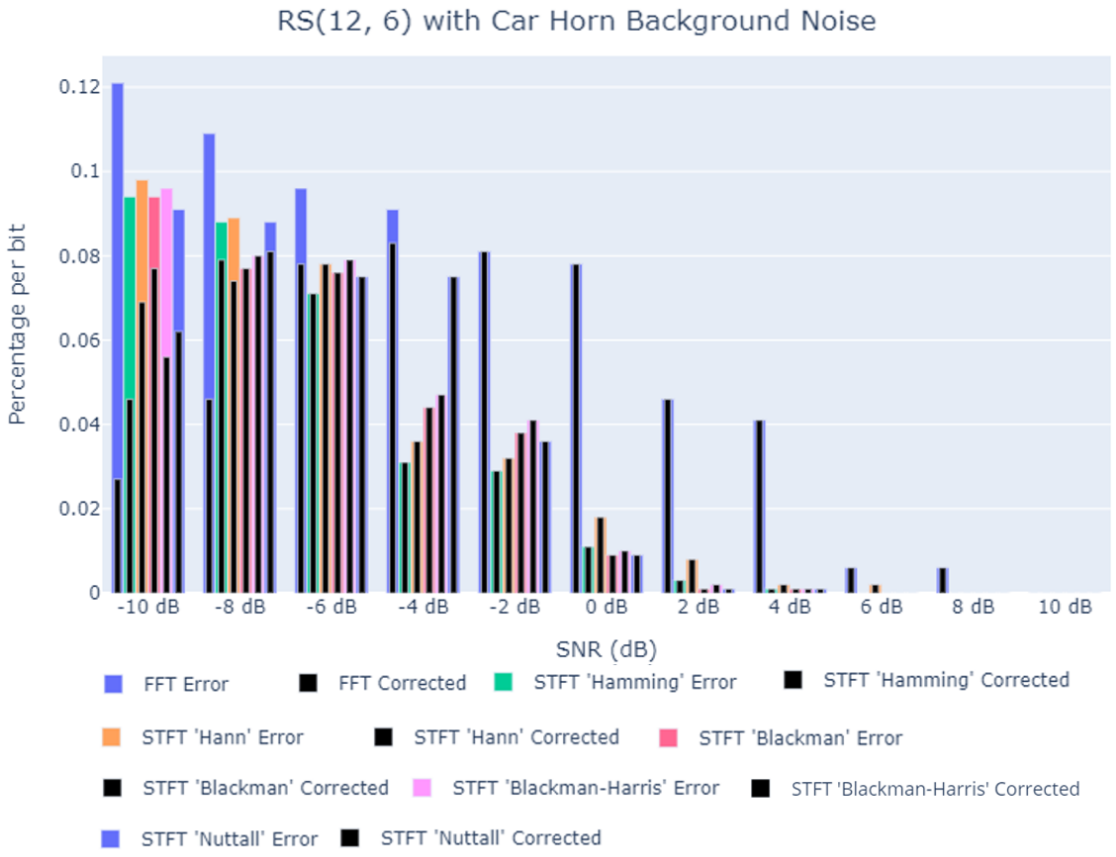

Figure 4.27: Car Horn Background Noise Performance on Medium Frequency

The FFT, as can be seen in the chart that is located above (Figure 4.21 and Figure 4.24), correctly identified the bit sequence this time even at a level of -2 dB or even -4 dB. This can be explained by the fact that the frequencies utilized in the bit string are those with a frequency higher than 400 Hz. For instance, when determining the frequencies, the sound range in the cafe noise environment, which ranges from 190 Hz to 390 Hz, did not have a significant impact on the process.

When the graphs are compared for STFT "Hamming" and STFT "Hann," it is discovered that these two windowing methods produce the right bits for the office background noise environment at the level of -8 dB. It is possible to notice that it corrects the bit sequence precisely at the level of -6 dB because the frequency range in the cafe sound environment is slightly wider than in the office sound environment.

Even at a -10 dB level, the STFT "Blackman," "Blackman-Harris," and "Nuttall" window functions are able to produce reliable frequency determinations. This is demonstrated by the figures. When the figures are observed (Figure 4.21, Figure 4.22, Figure 4.23, Figre 4.24, and Figure 4.25), it is discovered that when frequencies above 400 Hz are used in the alphabet, it is feasible to detect the frequencies that are used in the alphabet even in noisy sound environments.

5

Conclusions and Future Works

5.1 Conclusions

The aim of the thesis was to accomplish encryption of any data (text, sound, image) using the approaches used in QR codes (Reed-Solomon) and (Spectrogram) used in Shazam, and transmitting it from one device to another device via an audio signal. The research can be summed up as follows:

What role does frequency range have in sound transmission?

As described in the section on analysis, it has been shown that low frequencies are the most affected frequency range in noise environments when examining a variety of noise environments. Examining the noises reveals that the frequency range most affected is 60 Hz to 390 Hz.

How might Reed-Solomon and scrambling methods be applied for error correction?

Reed-Solomon could have used it as $RS(12, 6)$ or $RS(6, 3)$ since it was decided to use 14 bits in audio signal transmission as mentioned in the project. While $RS(6, 3)$ can correct only 1 information symbol out of 6 information symbols (this ratio is about 16%), $RS(12, 6)$ is used in this project because $RS(12, 6)$ can correct 3 information symbols out of 12 information symbols (this ratio is about 25%).

How effective are STFT and FFT in feature extraction and denoising?

As noted in the analysis section, the FFT method is ineffective in noisy environments since it does not utilize any filtering technique. It has been discovered that STFT's blackman window function accurately transmits audio signals in high-noise settings.

5.2 Future Works

By reviewing the information collected for this thesis and the methodologies employed, this project can be enhanced by implementing following suggestions:

- First, the number of possible combinations while creating the note alphabet can be expanded. By adding to an additional note, or as an alternative to using four frequencies for the symbol, two or three frequencies and a quiet part can be utilized. Double, triple, or quadruple chords can be combined in a single information symbol to create pleasing sounding chord combinations. Since the STFT indicates which frequencies are utilized during which time interval, this technique is realizable.
- Second, noise cancellation can be improved by trying additional STFT window functions. In addition, each produced chord can be isolated from noisy environments by training utilizing noise-prevention techniques based on deep learning. Facebook's denoiser tool can be used to isolate an audio stream from its noisy environment [16]. In order not to complicate this project even more, these methods are not included in the thesis.

In the end, the proposed technology can be considered a very promising solution that can be expected to be developed in practical contexts, even without strong connection capabilities, to simplify ubiquitous communications and multimedia content delivery. For instance, we are aware that Internet of Things (IoT) is a crucial technology for the present and the future. Due to the rapid expansion of IoT, the number of connected devices has increased dramatically. Nonetheless, this massive data collecting and processing presents its own unique challenges. In these circumstances, error-free data transmission and data acquisition becomes considerably more challenging. Numerous strategies have been implemented to address these issues [52]. The method mentioned in this thesis can also play a significant role in the transfer of data in situations such as reducing data size and ensuring data security.

References

[1] 2010. URL: https://it.paperblog.com/ars-magna-magna-295175/.

[2] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html?highlight=find_peak.

[3] URL: https://www.tensorflow.org/lite/api_docs/python/tflite_model_maker.

[4] URL: https://developer.android.com/ndk/guides/audio/sampling-audio#:~:text=In%5C%20general%5C%2C%5C%20it%5C%20is%5C%20best.

[5] URL: http://sox.sourceforge.net/.

[6] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.periodogram.html.

[7] URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html.

[8] URL: https://pixabay.com/tr/music/.

[9] Mukesh Arora, Chetan kumar, and Atul Kumar Verma. "Increase Capacity of QR Code Using Compression Technique". In: *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*. 2018, pp. 1–5. DOI: 10.1109/ICRAIE.2018.8710429.

[10] L. Badia, M. Levorato, and M. Zorzi. "Analysis of Selective Retransmission Techniques for Differentially Encoded Data". In: *2009 IEEE International Conference on Communications*. 2009, pp. 1–6. DOI: 10.1109/ICC.2009.5198746.

[11] Leonardo Badia, Marco Levorato, and Michele Zorzi. "Markov analysis of selective repeat type II hybrid ARQ using block codes". In: *IEEE Transactions on Communications* 56.9 (2008), pp. 1434–1441. DOI: 10.1109/TCOMM.2008.060374.

[12] Gm Bhat et al. "VHDL modeling and simulation of data scrambler and descrambler for secure data communication". In: 2 (Jan. 2009), pp. 41–42. DOI: 10.17485/ijst/2009/v2i10/30718.

[13] R. B. Blackman and J. W. Tukey. "The measurement of power spectra from the point of view of communications engineering Part I". In: *The Bell System Technical Journal* 37.1 (1958), pp. 198–228. DOI: 10.1002/j.1538-7305.1958.tb03874.x.

[14] Harish Chuppala et al. "Modified Cooley-Tukey algorithm for implementation of integrated serial FFT/IFFT processor in half-duplex OFDM systems". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. 2017, pp. 2328–2331. DOI: 10.1109/ICPCSI.2017.8392133.

[15] James W. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19.90 (1965), pp. 297–301. DOI: 10.1090/s0025-5718-1965-0178586-1.

[16] Alexandre Defossez, Gabriel Synnaeve, and Yossi Adi. *Real Time Speech Enhancement in the Waveform Domain*. 2020. DOI: 10.48550/ARXIV.2006.12847. URL: https://arxiv.org/abs/2006.12847.

[17] G. Forney. "Generalized minimum distance decoding". In: *IEEE Transactions on Information Theory* 12.2 (1966), pp. 125–131. DOI: 10.1109/TIT.1966.1053873.

[18] Orhaz Gazi. *Forward Error Correction via Channel Coding*. Jan. 2020, pp. 219–257. ISBN: 978-3-030-33379-9. DOI: 10.1007/978-3-030-33380-5.

[19] André Goalic et al. "Underwater acoustic communication using Reed Solomon Block Turbo Codes channel coding to transmit images and speech". In: *OCEANS 2010 MTS/IEEE SEATTLE*. 2010, pp. 1–6. DOI: 10.1109/OCEANS.2010.5664507.

[20] Robert H. and Morelos Zaragoza. *The art of error correcting coding*. 2nd ed. Oct. 2002, pp. 61–72. ISBN: 0-471-49581-6. DOI: 10.1002/0470847824.ch4.

[21] F.J. Harris. "On the use of windows for harmonic analysis with the discrete Fourier transform". In: *Proceedings of the IEEE* 66.1 (1978), pp. 58–64. DOI: 10.1109/PROC.1978.10837.

[22] *information capacity and versions of qr code*. Last accessed 29 July 2022. URL: https://www.qrcode.com/en/about/version.html.

[23] Aimé Lay-Ekuakille et al. "Leak Detection in Waterworks: Comparison Between STFT and FFT with an Overcoming of Limitations". In: *Metrology and Measurement Systems* Vol. 24 (Dec. 2017), Pages: 631 –644. DOI: 10.1515/mms-2017-0049.

[24] Rica Mangueira Lima et al. "Analysis of the influence of the window used in the Short-Time Fourier Transform for High Impedance Fault detection". In: *2016 17th International Conference on Harmonics and Quality of Power (ICHQP)*. 2016, pp. 350–355. DOI: 10.1109/ICHQP.2016.7783465.

[25] Vijay K Madisetti. *The digital signal processing handbook*. 2nd ed. CRC Press, 2010, pp. 1.1–1.7, 14.1–14.11. ISBN: 978-1420046045.

[26] Badri Narayan Mohapatra and Rashmita Kumari Mohapatra. "FFT and sparse FFT techniques and applications". In: *2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN)*. 2017, pp. 1–5. DOI: 10.1109/WOCN.2017.8065859.

[27] Hamidreza Mohebbi. "Parallel SIMD CPU and GPU Implementations of BerlekampMassey Algorithm and Its Error Correction Application". In: *International Journal of Parallel Programming* 47 (Feb. 2019), pp. 6–7. DOI: 10.1007/s10766-018-0574-x.

[28] Nejah Nasri et al. "Efficient encoding and decoding schemes for wireless underwater communication systems". In: *2010 7th International Multi-Conference on Systems, Signals and Devices*. 2010, pp. 1–6. DOI: 10.1109/SSD.2010.5585540.

[29] *Power Spectral Density*. URL: https://www.probabilitycourse.com/chapter10/10_2_1_power_spectral_density.php.

[30] Irving S. Reed and Gustave Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of The Society for Industrial and Applied Mathematics* 8 (1960), pp. 300–304.

[31] Martyn Riley and Iain Richardson. *reed-solomon codes*. URL: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html.

[32] F.E. Sandnes. "Efficient large-scale multichannel audio coding". In: *Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey*. 2001, pp. 392–397. DOI: 10.1109/EURMIC.2001.952480.

[33] Dilip V. Sarwate and Zhiyuan Yan. "Modified Euclidean algorithms for decoding Reed-Solomon codes". In: *2009 IEEE International Symposium on Information Theory*. 2009, pp. 1398–1402. DOI: 10.1109/ISIT.2009.5205901.

[34] *Scipy Get Window*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.get_window.html.

[35] Ahmet Serbes. "Fast and Efficient Sinusoidal Frequency Estimation by Using the DFT Coefficients". In: *IEEE Transactions on Communications* 67.3 (2019), pp. 2333–2342. DOI: 10.1109/TCOMM.2018.2886355.

[36] SS Shah, S Yaqub, and Faiza Suleman. "Self-correcting codes conquer noise - Part one: Viterbi codecs". In: *Edn -Boston then Denver then Highlands Ranch Co-* 46 (Feb. 2001), pp. 131–+.

[37] SYED SHAHZAD SHAH et al. *Self-correcting codes conquer noise, part 2: Reed-Solomon codecs*. Last accessed 29 July 2022. 1969. URL: https://www.edn.com/self-correcting-codes-conquer-noise-part-2-reed-solomon-codecs/.

[38] Hundo Shin and Ramesh Harjani. "Low-Power Wideband Analog Channelization Filter Bank Using Passive Polyphase-FFT Techniques". In: *IEEE Journal of Solid-State Circuits* 52.7 (2017), pp. 1753–1767. DOI: 10.1109/JSSC.2017.2700792.

[39] *short-time fourier transform - matlab stft*. URL: https://www.mathworks.com/help/signal/ref/stft.html.

[40] Hye-young Son et al. "Prediction of Flooded Compartment Damage Locations in Ships by Using Spectrum Analysis of Ship Motions in Waves". In: *Journal of Marine Science and Engineering* 10 (Dec. 2021), p. 17. DOI: 10.3390/jmse10010017.

[41] Andreas Spanias, Ted Painter, and Venkatraman Atti. *Audio signal processing and coding*. 1st ed. John Wiley and Sons, 2007, pp. 13–25. ISBN: 978-0471791478.

[42] Phaisarn Sutheebanjard and Wichian Premchaiswadi. "QR-code generator". In: *2010 Eighth International Conference on ICT and Knowledge Engineering*. 2010, pp. 89–92. DOI: 10.1109/ICTKE.2010.5692920.

[43] Burcu Tepekule, Utku Yavuz, and Ali Emre Pusane. "On the use of modern coding techniques in QR applications". In: *2013 21st Signal Processing and Communications Applications Conference (SIU)*. 2013, pp. 1–4. DOI: 10.1109/SIU.2013.6531318.

[44] Sumit Tiwari. "An Introduction to QR Code Technology". In: *2016 International Conference on Information Technology (ICIT)*. 2016, pp. 39–44. DOI: 10.1109/ICIT.2016.021.

[45] Beatrice Tomasi et al. "A Study of Incremental Redundancy Hybrid ARQ over Markov Channel Models Derived from Experimental Data". In: *Proceedings of the Fifth ACM International Workshop on UnderWater Networks*. WUWNet '10. Woods Hole, Massachusetts: Association for Computing Machinery, 2010. ISBN: 9781450304023. DOI: 10.1145/1868812.1868816. URL: https://doi.org/10.1145/1868812.1868816.

[46] M. Tomlinson et al. *Error-Correction Coding and Decoding*. Jan. 2017, p. 167. ISBN: 978-3-319-51102-3. DOI: 10.1007/978-3-319-51103-0.

[47] Thanh Tran et al. "Separate Sound into STFT Frames to Eliminate Sound Noise Frames in Sound Classification". In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2021, pp. 1–4. DOI: 10.1109/SSCI50451.2021.9660125.

[48] Barry Van Veen. *The Periodogram for Power Spectrum Estimation*. 2022. URL: https://www.youtube.com/watch?v=Qs-Zai0F2Pw&t=272s.

[49] Avery Wang. "An Industrial Strength Audio Search Algorithm." In: Jan. 2003, pp. 2–7.

[50] Tarun Kumar Yadav et al. "Real Time Audio Synchronization Using Audio Fingerprinting Techniques". In: *2022 1st International Conference on the Paradigm Shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS)*. 2022, pp. 16–20. DOI: 10.1109/PCEMS55161.2022.9808050.

[51] Wang Yuegang, Ji Shao, and Xu Hongtao. "Non-stationary Signals Processing Based on STFT". In: *2007 8th International Conference on Electronic Measurement and Instruments*. 2007, pp. 3.301–3.304. DOI: 10.1109/ICEMI.2007.4350914.

[52] Alberto Zancanaro, Giulia Cisotto, and Leonardo Badia. "Challenges of the Age of Information Paradigm for Metrology in Cyberphysical Ecosystems". In: *2022 IEEE International Workshop on Metrology for Living Environment (MetroLivEn)*. 2022, pp. 127–131. DOI: 10.1109/MetroLivEnv54405.2022.9826967.

Acknowledgments

I cannot adequately express my appreciation to my professor Leonardo Badia for his invaluable patience and feedback. In addition, this undertaking would not have been possible without the generous support of Omniaweb, which supported my research.

I would be remiss if I did not mention my family and friends, especially my mother. Their confidence in me has kept my spirits and motivation high throughout this endeavor.