Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT for Internet and Multimedia

Development of a Robust Signal Coding System for Short Audio Messages

Master Candidate

Arina Kosiakov

Student ID 2092162

Supervisor

Leonardo Badia

University of Padova

Academic Year
2022/2023
12 dicember 2023

Abstract

This thesis explores a collaborative venture with Omniaweb to develop TIScode, an innovative application facilitating information transmission through sound. TIScode is meant to transmit information in form of audio messages,made of different frequencies, through a common speaker, so that the devices with a proper app installed, can automatically capture the message. The project encompasses the creation of both an encoder and decoder, inspired by QR codes, with a focus on overcoming potential risks in data transmission. Urban noise presents a significant challenge, and to counter this, the study employs Reed-Solomon codes, known for their robust error-correction capabilities. These codes introduce redundancy to minimize errors, complemented by a scrambling technique distributing errors among codewords. Implemented in Python through the MATLAB API, the encoder associates each piece of information with a unique identification code represented by a sound sequence. The decoder, in turn, extracts frequencies from received sounds to recover the original message. This research not only presents a technical solution for information dissemination in challenging environments but also contributes valuable insights into error mitigation strategies for audio data transmission.

Contents

List of Figures

List of Acronyms

STFT	Short-Time Fourier Transform
rMQR	Rectangular Micro QR
CSV	Comma Separated Values
BCH	Bose-Chaudhuri-Hocquenghem
DSP	Digital Signal Processing
CFT	Continuous Fourier Transform
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
PSD	Power Spectral Density
AWGN	Additive white Gaussian noise

1 Introduction

The research presented in this thesis represents a collaborative effort with Omniaweb to create TIScode, an application designed to transmit information through sound which will be shared thought speakers and captured by a device provided with a microphone and the appropriate app.
The project draws inspiration from various successful technologies, such as QR codes [7] [44], which are among the most widely used methods for data sharing [33]. QR codes utilize a specific matrix to encode data, accessible as soon as the code is captured by a device's camera. The project's concept revolves around sharing data similarly, but through a different medium. In fact, the data is encoded into sound rather than an image, a microphone is used instead of a camera to retrieve the message.
The study propose to incorporation of Reed-Solomon codes as a solution to limit the number of errors during the decoding process. These codes are employed in QR codes, multimedia contexts, memory system design and in the Underwater Acoustic Communications (UAC) [12] [17] [45]. In particular UAC is a specialized field of study and technology that focuses on establishing reliable communication links underwater and it is an important example of the usage of RS codes for acoustic transmission. Some of the challenges in underwater communication include signal attenuation, dispersion, and the presence of ambient noise [41].

The noise is one of the biggest challenges also when dealing with urban environment, for this reason in this project it was decided to adopt Reed-Solomon codes.

Another source of inspiration for the project was the Shazam application. Similar to the TIScode's objective, Shazam allows users to retrieve information, in this case, details about a song, by using a microphone to record an audio segment. The application extracts "features," trying to limit the impact of noise, which are then used to search for the corresponding song within a database and return its name [38]. This principle aligns with the designed system, where specific frequencies are used to construct the sound. Furthermore, similar to Shazam, a spectrogram is employed to retrieve individual frequencies.

Finally, the efficiency of the system was assessed using Python and Matlab environments. The evaluation involved testing the system's behavior with varying Signal-to-Noise Ratios (SNR) and different types of noises.

The subsequent chapters are organized as follows: Chapter 2 provides an overview of existing technologies that inspired the development of TIScode. Chapter 3 summarizes theoretical concepts crucial to the development of the encoding and decoding mechanisms. Chapter 4 offers an analysis of the development and testing processes, focusing on final results and considerations. Finally, Chapter 5 presents conclusions, along with summaries of future works and potential implementations.

2 State of the Art

2.1 QR Codes

QR codes, short for Quick Response codes, are two-dimensional codes that have become part of everyday life in the modern digital age. They were created in 1994 by a Japanese company called Denso Wave, with the primary aim of efficiently tracking automotive parts during manufacturing [7]. Nowadays QR codes are used in different fields.

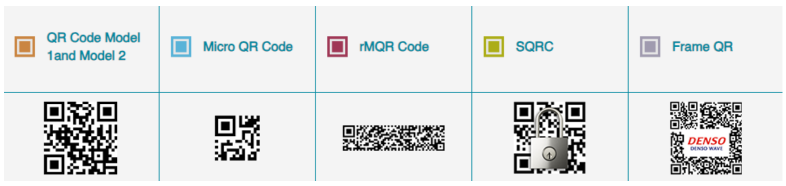


Figure 2.1: Different types of QRcodes [7]

QR codes are usually square in shape and consist of black squares arranged on a white background. These codes are one of the best ways to share information as they can store a significant amount of data, including text, URLs, contact details, and more. The bidimentionality of these codes permits to store more information compared to 1D barcodes, in fact due to their construction. A single QR Code symbol can hold up to 7089 numeric characters, 4296 alphanumeric characters, 2953 bytes (binary data) or 1817 Kanji characters (character set according to JIS X 0208) [31]. QR codes work by encoding data as a pattern of black and white squares that can be quickly scanned in every direction and decoded using a smartphone or dedicated QR code scanner. When scanned, the encoded information is instantly retrieved, making QR codes an invaluable tool for a wide range of applications, such as marketing, ticketing, inventory management, and contactless payments.

This technology is very versatile. There are different types of QR codes that allow to adapt the code to different surfaces or different type of information. For example the Rectangular Micro QR (rMQR) Code efficiently conserves space by minimizing the number of eye-shaped "finder patterns" from three to one and a half. Its space-saving rectangular design allows to use it on narrow surfaces [36].

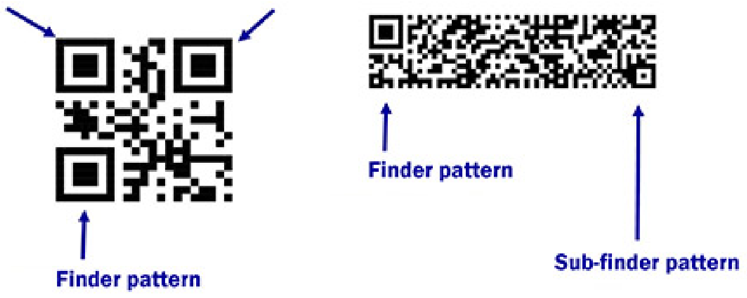


Figure 2.2: Example of rMQR code on the left [7]

Besides the different types of QR codes there are also different versions which span from Version 1 to Version 40, with each version featuring a distinct module configuration, denoting the number of modules within the symbol. Starting at Version 1 with a grid of 21x21 modules and culminating at Version 40 with a grid of 177x177 modules.

Each QR Code symbol version is optimized for maximum data capacity, accommodating various factors such as data volume, character type, and error correction level. In essence, as the data volume increases, QR Code symbols expand in size, necessitating the inclusion of more modules to encode the infor-

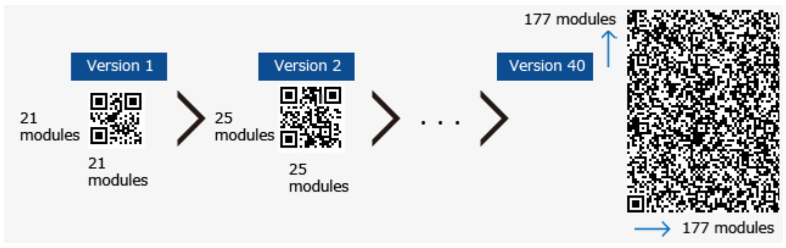


Figure 2.3: Representation of the different versions of QRcodes [7]

mation.
The QR code system consists in an encoder and a decoder. Encoding starts by specifying the desired data, encompassing alphanumeric characters, binary information, or even special characters, and then employing error correction codes for fault tolerance [44]. The error correction is based on Reed-Solomon algorithms and it has four possible levels: level L, level M, level Q, and level H [42]. Depending on the chosen level it is possible to restore between 7% and 30% of unreadable codewords without losing data.
The encoded data is organized into multiple data blocks, and each block undergoes various transformations, including data bit padding and Reed-Solomon error correction encoding. Subsequently, the QR code generator assembles these blocks into a structured grid, incorporating finder patterns, alignment patterns, and timing patterns to facilitate quick and accurate detection. Encoding also encompasses the inclusion of version-specific information and format information modules.
On the decoding front, specialized algorithms and image processing techniques are used to locate the QR code within an image, extract the grid, and discern the module configuration. This decoded matrix is then subjected to reverse transformations, such as Reed-Solomon error correction decoding, data de-padding, and character mapping, culminating in the retrieval of the original data. In essence, the encoding and decoding of QR codes constitute a complex interplay of data transformation, pattern recognition, and error correction.
In figure 2.4 it is possible to observe the different components of the QR code.

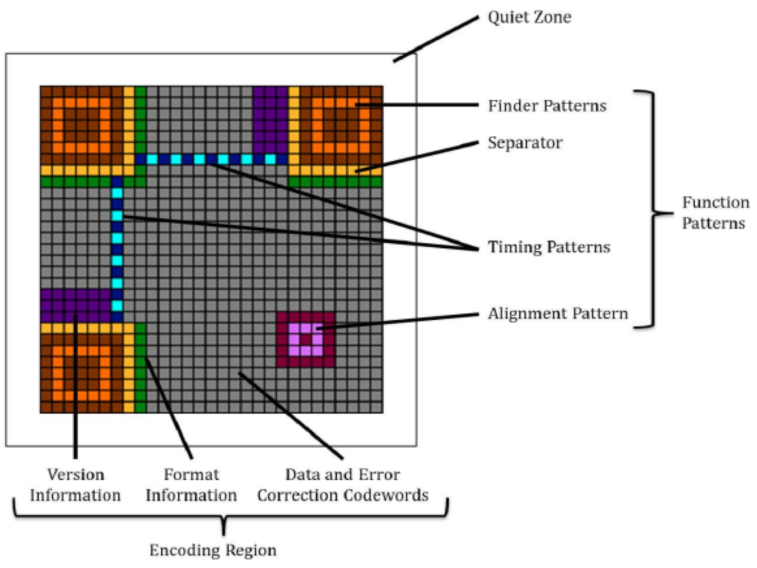


Figure 2.4: Structure of a QR code [44]

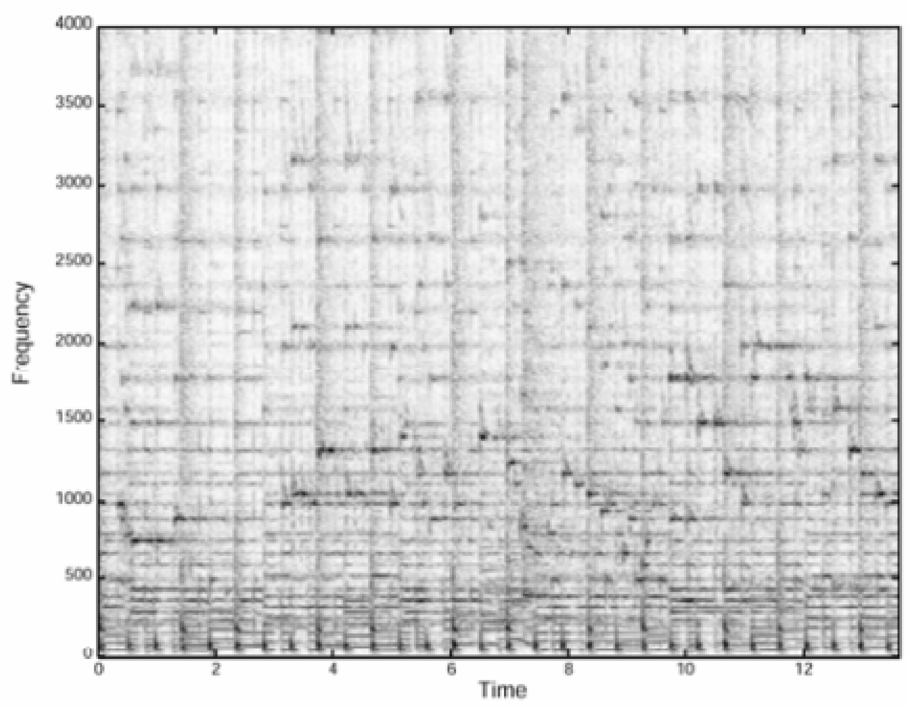
The most interesting ones are discussed below.

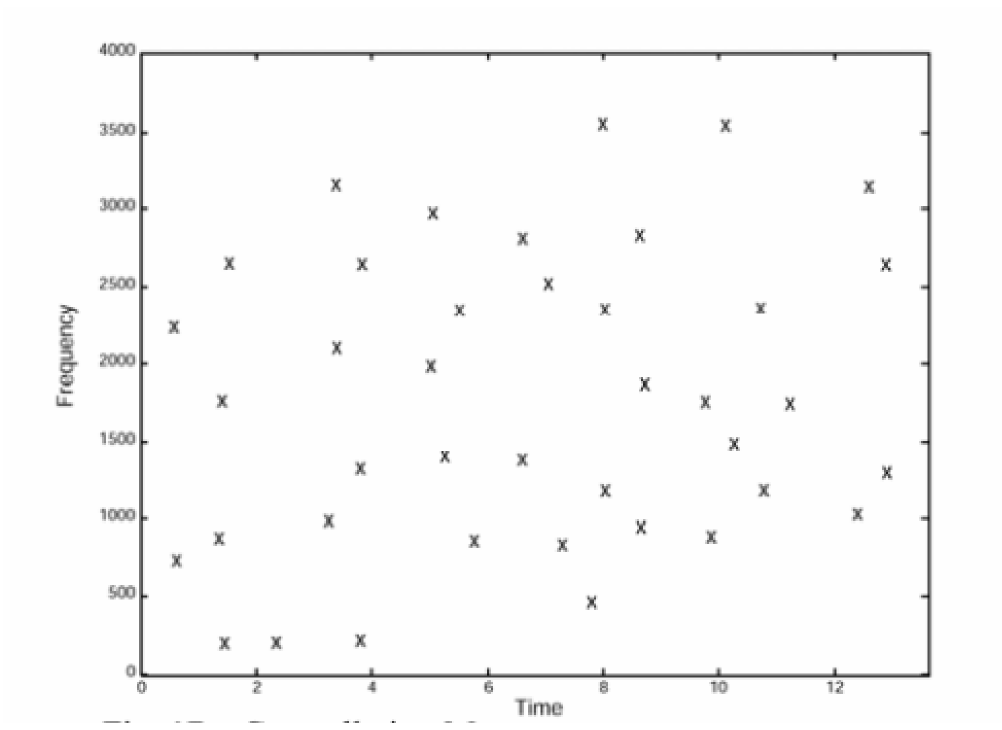
- *Quiet Zone*: It's a 4-module wide blank space with no data, used to make sure that the nearby text or markings don't misguide the QR code information.
- *Finder Pattern*: It is a special pattern which is collocated in precise positions and it is designed to be a pattern that is not likely to be found in the other parts. This module helps to correctly orient the QR code for decoding.
- *Timing Patterns*: There are two timing patterns, one in a horizontal position and the other one in vertical position. These patterns are used to determine the symbol density, module coordinates and version information area.

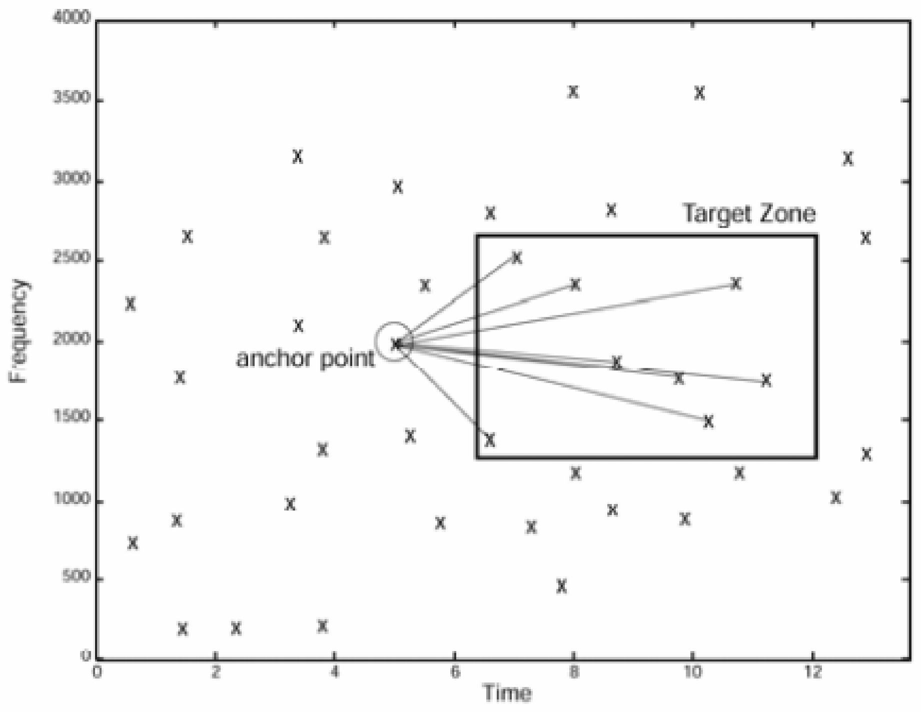
In summary, in the thesis a similar concept is discussed. Both ideas utilize multimedia content as a means of communication with a strong focus on ensuring reliability. One of the focal points of this thesis is the use of the Reed-Solomon algorithm.

2.2 Shazam

Shazam, is an application created in 2002 by Shazam Entertainment that can identify a song by analysing a small fragment of it [38]. The overall functioning is simple, it is enough to start the app and record a fragment of the song with the microphone of the device. Shazam's remarkable functionality hinges on a complex amalgamation of audio fingerprinting, massive song databases, and cloud-based computing power.
At its core, Shazam employs a process similar to musical fingerprinting to discern the unique acoustic signature of a given song. When a user records a snippet of music with the application, the latter converts the captured audio into a digital spectrogram, essentially a visual representation of the audio frequencies and their intensity over time [29], an example of a spectrogram can be seen in figure 2.5. The use of spectogram guarantees robustness when the sound is affected by noise and it is used to find the peaks. A point of spetrogram is defined as a peak if it has a higher energy than all of its neighbors.




Frequency
Time


Target Zone
anchor point
Frequency
Time

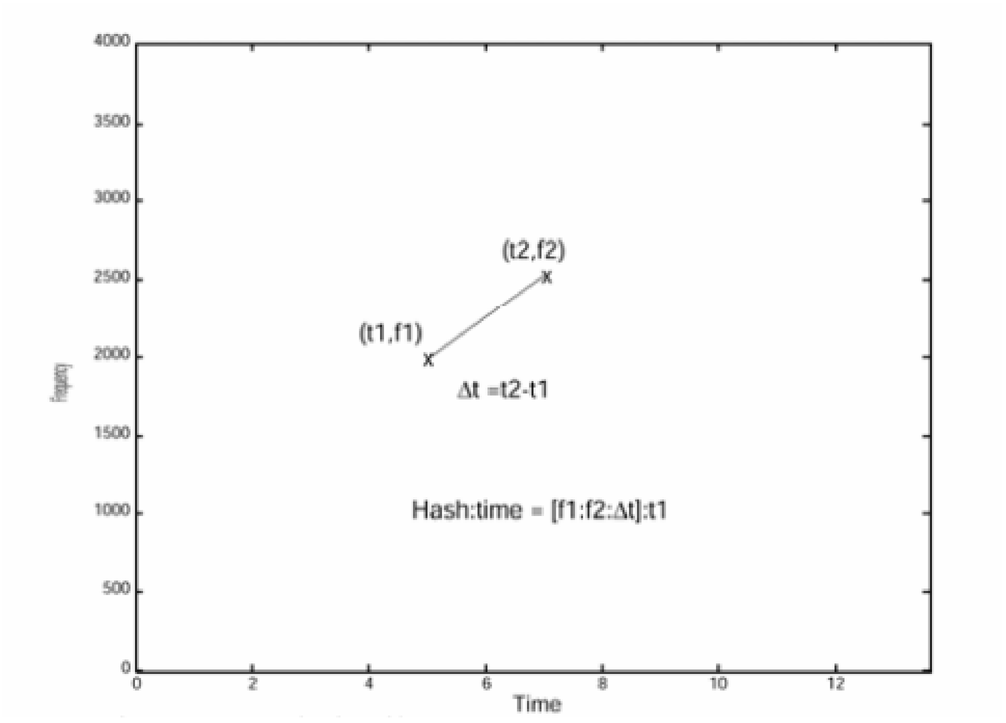
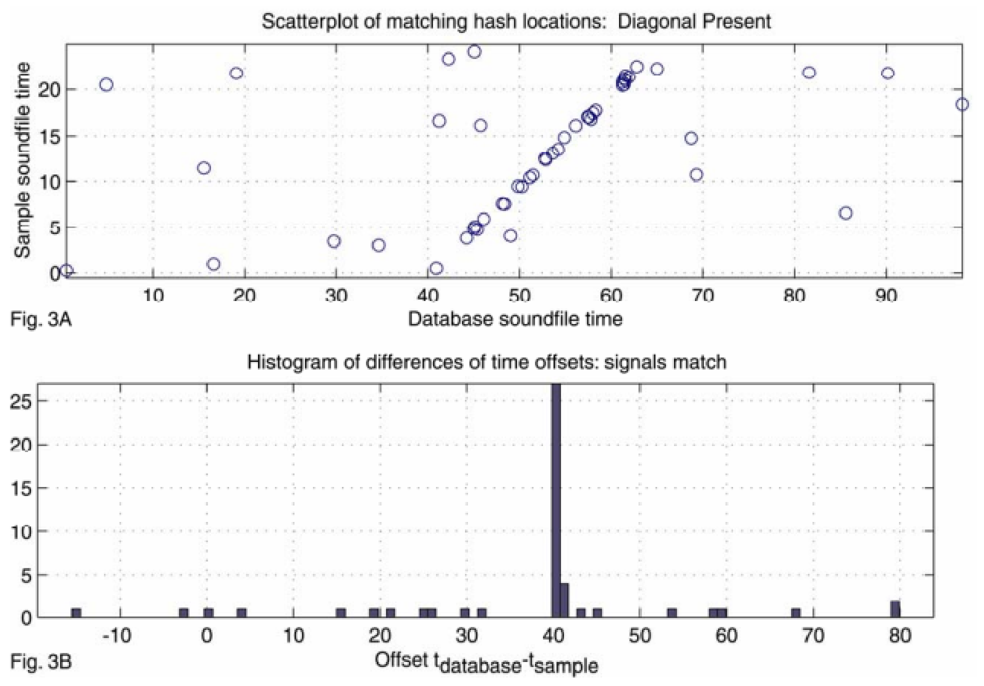
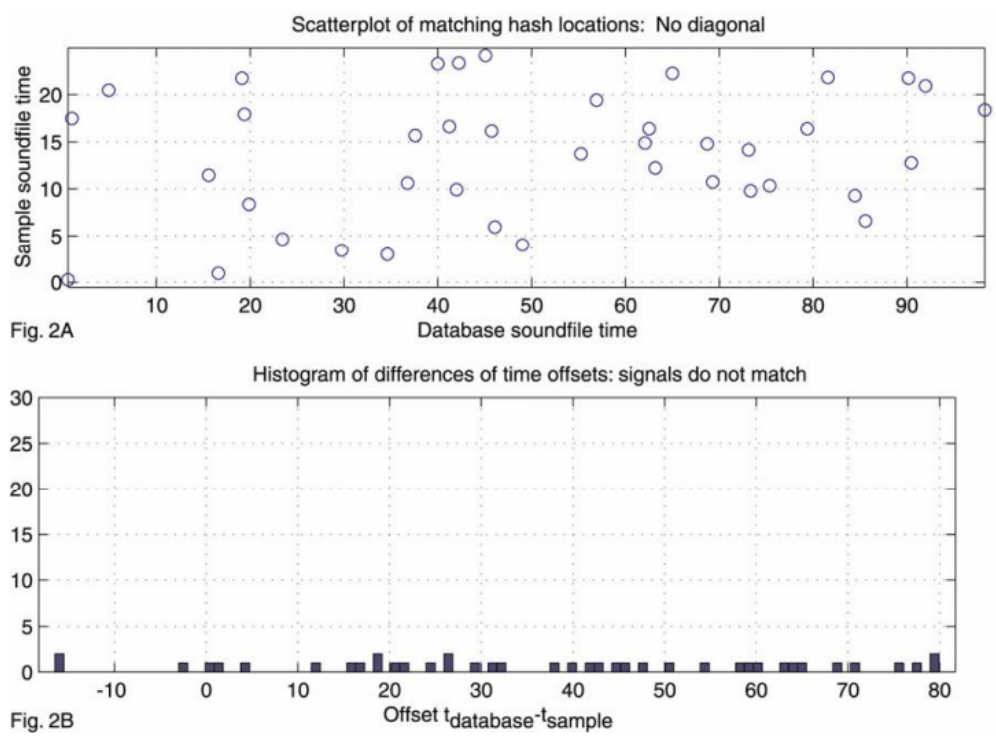


Figure 2.5: Example of spectrogram (first), example of a constellation map (second), combinatorial Hash Generation (third), hash details (fourth) [38]

The complicated spectrogram is then reduced to a "constellation map" of peaks. This procedure is made for both the song and the audio sample. The match between the sample and the entire song is achieved if the number of matching points between the two constellations is significant.
In order to find the correct registration offset the maps are indexed in a specific way by using "anchor" points. These points have a target zone associated with them and they are sequentially paired with points within the zone (figure 2.5). This operation is applied to every track in a database and the outcome is a list of hashes and their associated offset times.
As soon as the application receives the audio sample, it immediately extracts the fingerprint, then a set of hash and time offset records is generated. At this point the hash is compared to the hashes in the database and for each match time pairs, between the corresponding time offsets and database files, are created and distributed in bins.



(a) Representation of a match



(b) Representation of a mismatch

Figure 2.6: Graphical representation of a match and mismatch [38]

The bins are then scanned and in the end the song with a higher matching score, or the number of matching points, is returned as the match. The figure 2.6 shows a scatterplot of matching hash locations and a histogram of differences of time offsets for the match (a) and a mismatch(b).
As it finds a match, Shazam returns information about the song, including the title, artist, album, and sometimes even lyrics, enriching the user's musical experience. Furthermore, it provides links to various streaming platforms, allowing users to listen to the full track, explore related content, and seamlessly integrate their newfound discoveries into their music libraries.
In this thesis the concept behind the functioning of Shazam were considered as an inspiration, especially the functioning of the fingerprints. In particular, the main goal of this research is to find a better way of separating frequency peaks in a given sample. This peaks are easily identifiable by using the spectrogram and constellation maps even in noisy environments.

3 Scientific Background

3.1 Digital Signal Processing

The project of this thesis is based on the principles of Digital Signal Processing (DSP). Digital Signal Processing is a set of different techniques used to manipulate and analyze digital signals, which are numerical representations of real-world signals such as audio, video, or sensor data. Unlike analog signals, which are continuous and vary smoothly, digital signals are discrete and quantized into specific values. DSP algorithms process these digital signals to perform tasks like filtering, noise reduction, compression, and modulation [26] [18].
By applying mathematical techniques and algorithms, DSP allows for the extraction of valuable information from signals, enabling tasks like speech recognition, image processing, and audio enhancement. It plays a crucial role in various applications, including telecommunications, audio and video processing, medical imaging, radar systems, and more, enhancing the efficiency and accuracy of signal analysis and manipulation in the digital domain.

3.1.1 Fourier Transform

The Fourier Transform is a fundamental mathematical tool in the field of signal processing. It provides a way to decompose complex signals into simpler sinusoidal components [16]. Named after the French mathematician and physicist Jean-Baptiste Joseph Fourier, this mathematical technique allows any signal, no matter how complex, to be represented as a combination of sinusoidal waves with different frequencies, amplitudes, and phases. By understanding the frequency components of a signal, the Fourier Transform is widely applied in various fields. In audio processing, it helps analyze and manipulate different frequencies of sound waves, enabling tasks like equalization and noise filtering. There are several types of Fourier Transforms, each designed to handle different types of signals and applications. Here are the main types:

- Continuous Fourier Transform (CFT)
- Discrete Fourier Transform (DFT)
- Short-Time Fourier Transform (STFT)

Continuous Fourier Transform

This type of Fourier Transform is used for continuous, time-domain signals. It decomposes a continuous signal into its constituent frequencies, providing a continuous spectrum. In mathematical terms the CFT of the function $f(t)$ is the function $Ff(t)$, where [21] [39]:

$$Ff(t) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{3.1}$$

where $\omega = 2\pi k$

Discrete Fourier Transform

The DFT is used for discrete, time-domain signals, which are sequences of values sampled at specific intervals. It converts a finite sequence of equally spaced samples of a function into a same-length sequence of complex numbers, which represent the signal in the frequency domain. Given a discrete signal a_n the DFT of the signal is:

$$A_k = \sum_{n=0}^{N-1} e^{-i\omega n} a_n \tag{3.2}$$

where N is the number of points and indicates how the signal in frequency domain should be sampled. In particular taken ω such that the period is $0 < \omega < 2\pi$ and

$$\omega_k = \frac{2\pi}{N} k, k = 0, 1, ..., N-1 \tag{3.3}$$

The computation of the DFT is very slow, in fact it requires $O(N^2)$ operations for N input samples, for this reason a better algorithm should be used. Fast Fourier Transform (FFT) are a family of algorithms for computing the DFT. Unlike the standard DFT, the FFT reduces the computational complexity to $O(N \log N)$ making it significantly faster, especially for large datasets [21]. It is particularly useful for processing digital signals on computers and digital systems. This efficiency is crucial in various applications, such as audio and video processing, telecommunications, and scientific computing, where rapid analysis of signals is required.

Short-Time Fourier Transform

The Short-Time Fourier Transform is a fundamental technique in the field of digital signal processing, widely employed for analyzing and processing time-varying signals. Unlike the standard Fourier Transform, which provides information about the frequency content of an entire signal, the STFT focuses on the variation of frequencies over short, overlapping time intervals. Mathematically the STFT of a signal x(t) is defined as:

$$X(t,f) = \int_{-\infty}^{\infty} x(\tau)w(\tau - t)e^{-i2\pi f(\tau - t)}d\tau \tag{3.4}$$

where X(t, f) represents the STFT of the signal and $w(\tau - t)$ is a window function that tapers the signal to minimize spectral leakage. The window function limits the analysis to a specific time interval allowing to observe the signal's frequency components within that interval.
In practical applications, signals are usually discrete and sampled at certan intervals. In order to apply STFT to discrete signals its equation can be adapted as follows:

$$X[n,k] = \sum_{m=-\infty}^{\infty} x[m]w[m-n]e^{-}-i2\pi k(m-n)/N \tag{3.5}$$

By breaking down a signal into smaller segments, the STFT captures how the frequency components change over time, offering a detailed insight into the signal's behavior in both time and frequency domains. This method is particularly valuable in applications where signals exhibit non-stationary behavior, meaning their frequency content changes over time.
One of the critical points of the STFT is the choice of the window function. Each function affect differently the trade-off between frequency and time resolution. In particular, a narrower window yields better frequency resolution but poorer time resolution, whereas a wider window provides better time resolution at the expenses of frequency resolution [24]. Four principal windowing techniques commonly used are the Rectangular Window, Hamming Window, Hanning Window, Blackman Window and Gaussian Window. Each technique employs a different mathematical function to define the shape of the window, affecting the trade-off between the main lobe width and side lobe levels in the frequency domain.

Rectangular Window

The rectangular window is the most basic and simplest form of windowing technique used in signal processing. Its defining characteristic lies in its abrupt cut-off at the edges, resembling a rectangular shape. The purpose of the rectangular window is to segment a signal into specific intervals for analysis, and its mathematical representation is straightforward. For a signal of length N, the rectangular window function is defined as follows:

$$w(n) = 1, \quad 0 \leq n < N$$

$$w(n) = 0, \quad \text{elsewhere}$$

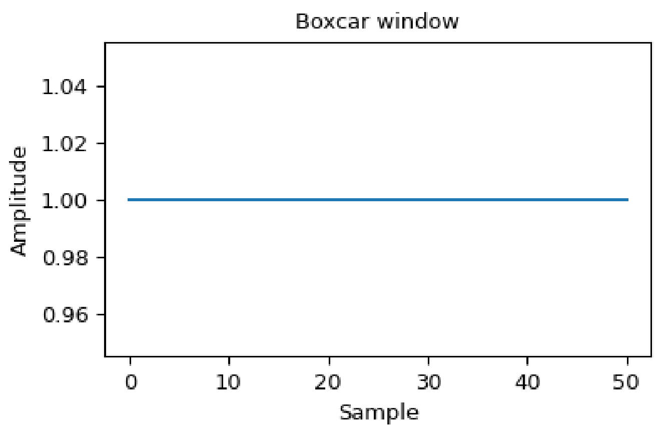
where $w(n)$ represents the window function, and n the discrete time index. The rectangular window assigns a value of 1 to all data points within the window's interval and 0 to points outside this interval. While conceptually simple, the rectangular window has significant implications for the frequency analysis of signals.

The Fourier Transform of a signal windowed by a rectangular window is represented by the sinc function in the frequency domain. The sinc function (sinc(x)) is defined as sinc(x) = $\frac{\sin(\pi x)}{(\pi x)}$. For the rectangular window function $w(n)$, the Fourier Transform $W(f)$ is given by:

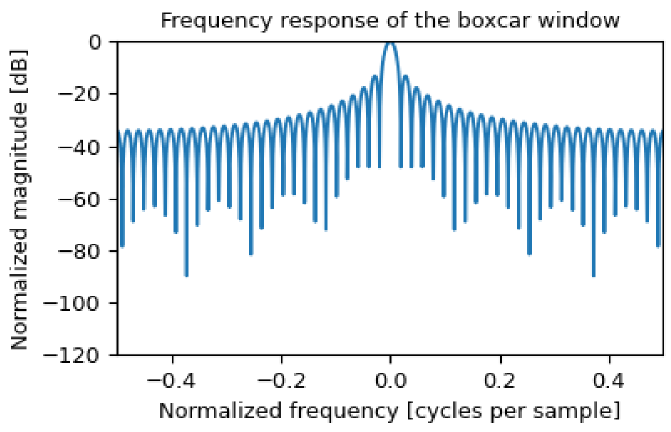
$$W(f) = \text{sinc}\left(\frac{fN}{2}\right)$$

This equation illustrates the frequency response of the rectangular window. The main lobe of the sinc function is centered around the frequency corresponding to the center of the window, and it gradually attenuates as the frequency moves away from the center. However, the abrupt cut-off at the edges of the window results in high side lobes, causing spectral leakage. Spectral leakage occurs when the energy from the main lobe spills into adjacent frequency bins, leading to inaccuracies in frequency analysis. One of the significant drawbacks

of the rectangular window is its poor frequency resolution due to the wide main lobe. While it provides excellent localization in the time domain, it sacrifices precision in the frequency domain. This limitation is a direct consequence of the abrupt transition from 1 to 0 at the edges of the window.
Despite its limitations, the rectangular window finds application in cases where frequency analysis is not the primary concern, and a simple, straightforward segmentation of the signal is sufficient.



(a)



(b)

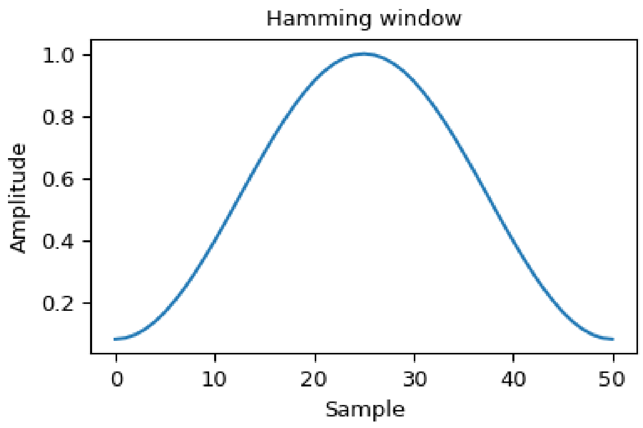
Figure 3.1: Rectangular Window [2]

Hamming Window

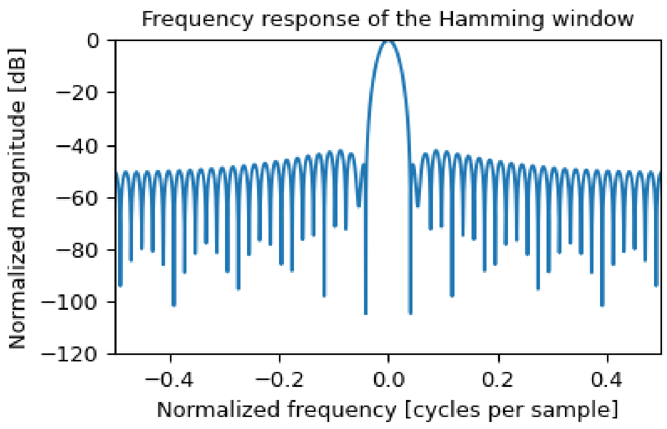
The Hamming window is designed to reduce spectral leakage by tapering the signal smoothly at the edges. Its equation is defined as follows:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

where $w(n)$ represents the amplitude of the window at sample n, and N denotes the total number of samples in the window.



(a)



(b)

Figure 3.2: Hamming Window [2]

The term $0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$ shapes the window function. The coefficient

0.54 at the beginning of the equation ensures that the average power of the window is 1, which is a desirable property for preserving the signal's energy.
One of the distinctive features of the Hamming window is its smooth transition from zero to the maximum value, which reduces abrupt changes in the signal. This smoothness is achieved by the cosine term in the equation. Additionally, the Hamming window effectively suppresses side lobes, the secondary peaks in the frequency spectrum. By minimizing side lobe levels, the Hamming window reduces spectral leakage, a phenomenon where energy from the main lobe spills into adjacent frequency bins, causing inaccuracies in frequency analysis.
The Hamming window strikes a balance between main lobe width and side lobe levels. The main lobe width determines the frequency resolution of the windowed signal. A narrower main lobe provides better frequency resolution but sacrifices the ability to distinguish closely spaced frequencies. On the other hand, side lobes represent the amplitude of unwanted frequency components introduced by windowing. Lower side lobe levels are essential for accurate spectral analysis, especially in applications like radar systems, where precise detection of signals amidst noise is critical. The Hamming window's design ensures that it offers a reasonable compromise between frequency resolution and side lobe suppression, making it suitable for a wide range of applications.

Hanning Window

The Hanning window is used to reduce spectral leakage in the frequency analysis of signals and it effectively mitigates this issue by tapering the signal smoothly at the edges, which reduces the amplitude of the side lobes, resulting in a more accurate representation of the signal's frequency content.

The mathematical equation for the Hanning window is defined as follows:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right)$$

In this equation, cos denotes the cosine function, and the factor 2π ensures that the cosine wave completes one full cycle within the window length. The term $0.5 - 0.5\cos$ smoothly tapers the window from 1 to 0, ensuring continuity at the edges. One of the key characteristics of the Hanning window is its ability

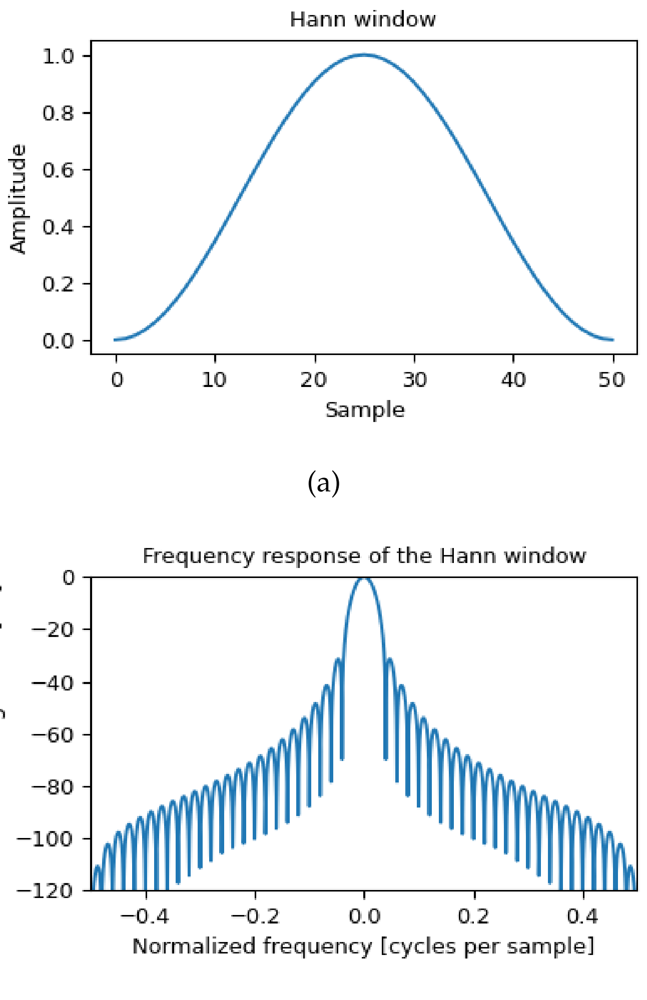


Figure 3.3: Hanning Window [2]

to balance between the main lobe width and the level of the side lobes in the frequency domain. The main lobe refers to the central peak in the frequency spectrum, while the side lobes are the smaller peaks surrounding the main lobe. By tapering the signal with the Hanning window, the main lobe width is effectively reduced, which enhances frequency resolution. Additionally, the side lobes are attenuated, lowering their amplitude. This attenuation of the side lobes is crucial because it minimizes the interference between adjacent frequency components, leading to a more precise representation of the signal's frequency content.

The Hanning window is widely used in various applications where accurate frequency analysis is essential. One such application is in the field of audio signal processing, where the Hanning window is applied to audio signals before performing a Fourier Transform. By using the Hanning window, it is possible to obtain a more detailed and accurate representation of the audio spectrum, enabling tasks such as pitch detection, harmonic analysis, and noise reduction.

Blackman Window

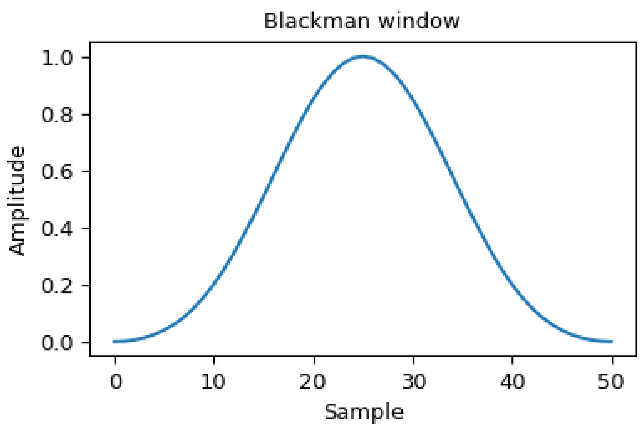
The Blackman window is a function designed to minimize spectral leakage and side lobes in the frequency domain. The Blackman window is a member of the Blackman family of windows, which includes the Blackman-Harris and Nuttall windows. It is named after the American engineer and mathematician Robert J. Blackman. The Blackman window is particularly effective in applications where high suppression of side lobes is essential, such as in spectrum analysis and modulation schemes.

The equation for the Blackman window is defined as follows:

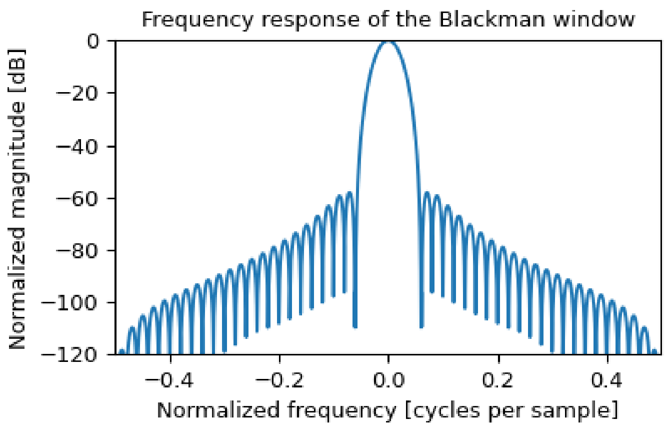
$$w(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$$

The Blackman window's unique formulation includes both cosine and constant terms, which contribute to its superior performance in terms of side lobe suppression and frequency resolution. The first term, 0.42, provides the central lobe of the window, ensuring that the window's values are positive and symmetric around the midpoint. The second term, $-0.5\cos\left(\frac{2\pi n}{N-1}\right)$, represents the first side lobe suppression factor. This term reduces the amplitude of the side lobes, minimizing spectral leakage. The third term, $0.08\cos\left(\frac{4\pi n}{N-1}\right)$, further refines the side lobe suppression by addressing higher-order side lobes.

The Blackman window's effectiveness lies in its balance between the main lobe width and the suppression of side lobes. The central lobe is wider compared to other windows like Hamming and Hanning, resulting in better frequency resolution. Simultaneously, the coefficients of the cosine terms minimize the amplitudes of side lobes, reducing interference and spectral leakage. This unique balance makes the Blackman window suitable for a wide range of applications, including speech and audio processing, radar systems, and sonar applications. In practical terms, the Blackman window is applied to a signal by element-wise multiplication. This process effectively tapers the signal at its edges, reducing the impact of discontinuities during Fourier analysis.



(a)



(b)

Figure 3.4: Blackman Window [2]

Gaussian Window

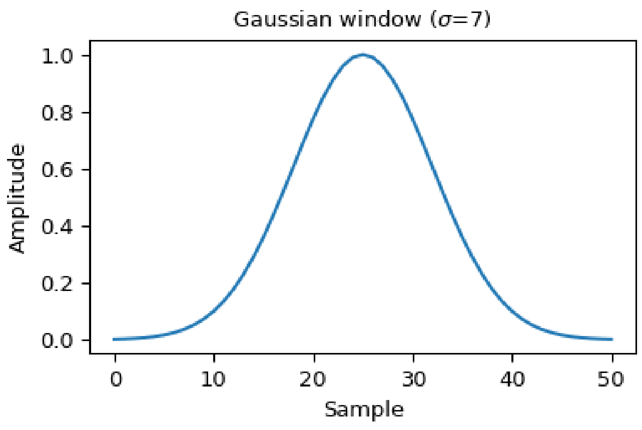
The Gaussian window is a widely used windowing function due to its unique properties, particularly its excellent frequency localization and minimal side lobes. Unlike other window functions, the Gaussian window is defined by a Gaussian distribution, which imparts distinct advantages in applications requiring high-frequency precision.

The Gaussian window is mathematically defined as:

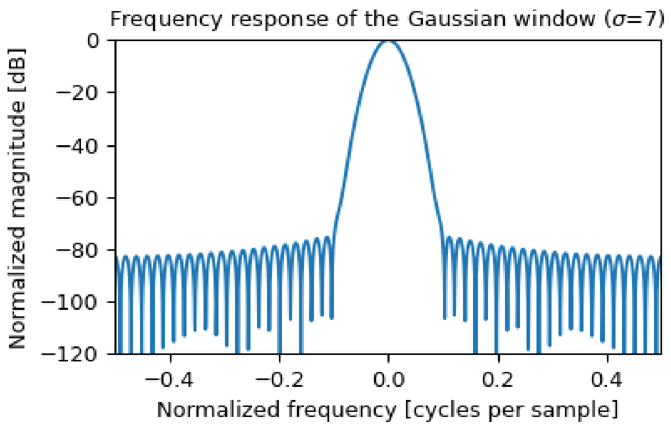
$$w(n) = e^{-\left(\frac{n-(N-1)/2}{\alpha(N-1)/2}\right)^2}$$

where $w(n)$ represents the value of the window at index n, N is the window length, and α is a parameter controlling the width of the Gaussian. The central peak of the Gaussian window is located at $n = (N-1)/2$. The parameter α determines the width of the main lobe; smaller α values result in wider main lobes, providing better frequency localization, while larger α values yield narrower main lobes.

The Gaussian window's defining characteristic is its ability to minimize side lobes, the secondary peaks in the frequency spectrum caused by the windowing process. By employing the Gaussian distribution, this window function reduces the amplitude of side lobes significantly, leading to precise frequency localization. The Gaussian window's primary strength is frequency localization. The narrowness of its main lobe means that energy from the signal is concentrated in a small range of frequencies, allowing for precise frequency measurement. While the Gaussian window excels in frequency localization and side lobe suppression, its main lobe is wider than that of other window functions which can affect frequency resolution. Therefore, the choice of the Gaussian window depends on the specific requirements of the application.



(a)



(b)

Figure 3.5: Gaussian Window [2]

3.1.2 Power Spectral Density

Power Spectral Density (PSD) is a fundamental concept in the field of signal processing and engineering which provides insights into the frequency components of a signal [14]. PSD represents the distribution of a signal's power over its frequency components and offers a view of how the signal's energy is distributed across the frequency spectrum. It plays a central role in various applications, including communications and audio processing. There are two methods for the estimation of the PSD: parametric and non- parametric. Parametric methods assume that the data is generated based on a specific model and do not have limitations on frequency resolution, while non-parametric methods does not make any assumption about the origin of data.

For a continuous-time signal $x(t)$, the PSD, denoted as $S(f)$, represents the power per unit frequency and is defined as the Fourier Transform of the signal's autocorrelation function, $R(\tau)$:

$$S(f) = \lim_{T \to \infty} \frac{1}{T} \left| \int_{-T/2}^{T/2} x(t) e^{-i2\pi f t} dt \right|^2 \tag{3.6}$$

In the case of a discrete-time signal $x[n]$, the PSD is defined as the Discrete Fourier Transform (DFT) of the autocorrelation function:

$$S(f_k) = \left| \sum_{n=0}^{N-1} x[n] e^{-i2\pi f_k n/N} \right|^2$$

Here, f represents frequency, t denotes time for continuous signals, n represents discrete time indices for discrete signals, and f_k represents discrete frequency indices. The PSD indicates the distribution of a signal's power across different frequencies that means that signals with different frequency content will have distinct PSDs.

In communication systems, understanding the signal's power distribution across frequencies is crucial for efficient transmission and reception. By analyzing the PSD, it is possible to design filters and modulation schemes that precisely target specific frequency bands, optimizing signal quality and bandwidth utilization. Additionally, in fields such as audio engineering, PSD analysis helps in noise reduction and equalization problems, ensuring high-quality audio output.

Estimating the PSD from finite-duration signals involves various techniques such as the periodogram and Welch's method [14]. The periodogram is a straight-forward approach where the squared magnitude of the Fourier Transform of

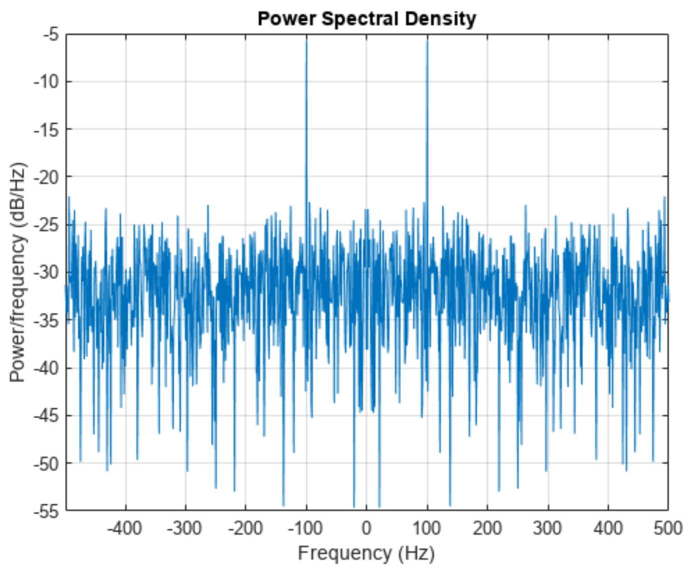


Figure 3.6: Example of a periodogram [4]

the signal is used to estimate the PSD. Welch's method, on the other hand, divides the signal into overlapping segments, computes individual PSD estimates for each segment, and averages them to obtain a smoother and more reliable estimate, especially for non-stationary signals. In the analysis of the real-world challenges, noise and finite observation time can affect the accuracy of PSD estimation. It is possible to counter these issues applying techniques such as windowing.

3.2 Reed-Solomon codes

Reed-Solomon codes (RS codes) are a powerful block-based class of error-correcting codes [35] widely used in digital communication and data storage systems, in particular for satellite communications, storage devices and high-speed modems. Developed by Irving S. Reed and Gustave Solomon in the 1960s [46], these codes offer robust error detection and correction capabilities. A remarkable features of Reed-Solomon codes is their ability to correct a predetermined number of errors while also effectively detecting and flagging others. This makes them particularly well-suited for deployment in situations where

data accuracy is very important, such as optical and magnetic storage devices, QR codes, and noisy channels such as the one taken in analysis in this project.
At their core, Reed-Solomon codes operate by converting data into a sequence of symbols and then appending redundant information to this data. This redundancy is calculated using polynomial mathematics, typically in the Galois field, and is strategically designed to enable the recovery of lost or corrupted symbols, thereby rectifying errors that may occur during data transmission or storage. The versatility of Reed-Solomon codes is also characterized by the possibility to choose the number of redundancy symbols.
A Reed-Solomon code is specified as RS(n,k) with symbols' length equal to **s**. This means that the encoder takes **k** data symbols of **s** bits each and adds parity symbols to make an **n** symbols codeword. There are **n-k** parity symbols of s bits each. A Reed-Solomon decoder can correct up to **t** symbols that contain errors in a codeword, where **2t = n-k**. The values of **n** and **k** have to satisfy the following relation:

$$0 < k < n < 2^m + 2 \tag{3.7}$$

The quantity of parity symbols is tied to both the code's error-correction capacity and the total number of symbols. Reed-Solomon codes can be seen as a subset

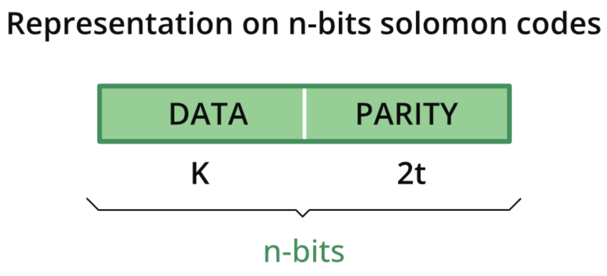


Figure 3.7: Representation of the Reed-Solomon codeword [6]

of Bose-Chaudhuri-Hocquenghem (BCH) codes, but in comparison to binary BCH codes, Reed-Solomon codes offer several advantages that make them a preferred choice in various applications. Firstly, they provide a higher efficiency in utilizing redundancy, ensuring robust error correction. Moreover, Reed-

Solomon codes provide flexibility by allowing for adjustments in block length and symbol size. These codes also offer a wide range of code rates and different efficient decoding techniques.

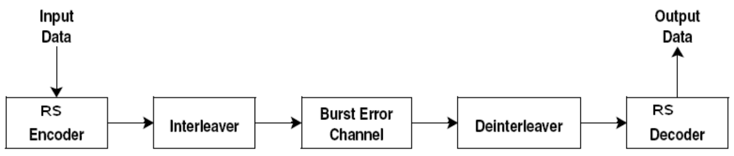


Figure 3.8: Block Diagram of Reed-Solomon code[43]

3.2.1 Encoding in Reed-Solomon code

In the Reed-Solomon encoding process, data is transformed into polynomial form, with each symbol in the data block representing a coefficient in the polynomial. The data polynomial (3.8) is then divided by a generator polynomial (3.9) to create the encoded message.

$$D(x) = x^{n-k}M(x) \tag{3.8}$$

$$G(x) = (x - \alpha^{i})(x - \alpha^{i+1})...(x - \alpha^{i+2t}) \tag{3.9}$$

The generator polynomial is a key element in Reed-Solomon coding, determining the number of redundant symbols added to the original message. These redundant symbols **r(x)** are derived through the previous division as follows:

$$r(x) = mod(D(x)/G(x)) \tag{3.10}$$

Finally the codewords C(x), or code polynomial is obtained by appending r(x) to the data polynomial.

$$C(x) = r(x) + D(x) = r(x) + x^{n-k}M(x) \tag{3.11}$$

This polynomial, which includes both the original data and the redundant symbols, is what is transmitted or stored. During decoding, these redundant symbols enable the receiver to identify and correct errors in the received message by performing polynomial division and evaluating syndromes, thereby ensuring the integrity and accuracy of the transmitted data.

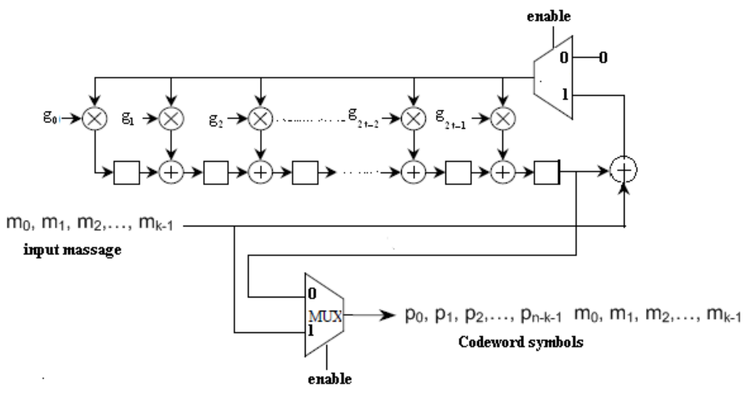

enable
0
1
0
g_0
g_1
g_2
g_{2t-2}
g_{2t-1}
$m_0, m_1, m_2, \ldots, m_{k-1}$
input massage
0
MUX
1
$p_0, p_1, p_2, \ldots, p_{n-k-1}$ $m_0, m_1, m_2, \ldots, m_{k-1}$
Codeword symbols
enable

Figure 3.9: Architecture for Reed-Solomon encoder [17]

3.2.2 Decoding in Reed-Solomon code

In the Reed-Solomon decoding process, the received data, which includes both the original message symbols and additional redundant symbols, undergoes intricate calculations to detect and correct errors.

This process begins with the received symbols being converted into a polynomial. The decoder then works to evaluate this polynomial to locate errors. Reed-Solomon codes are capable of correcting up to **t** errors and **2t** erasures (missing symbols) within the received data. To achieve this, the decoder uses a method called syndrome calculation, which involves performing polynomial

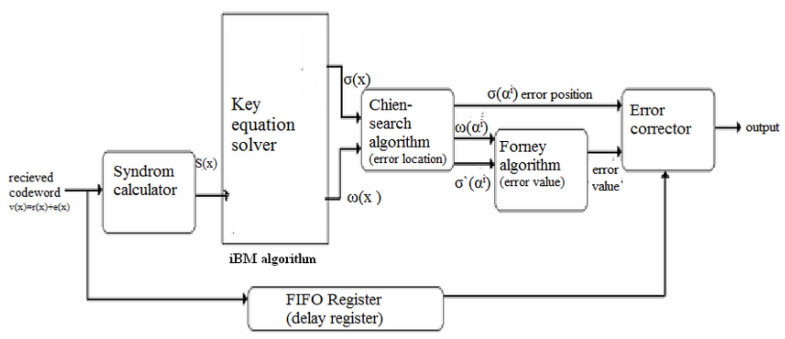


Figure 3.10: Reed-Solomon algorithm decoding process [17]

divisions and evaluating the results to identify errors. The syndromes, representing discrepancies between the received and expected data, are computed and analyzed to determine the error locations. Once the errors are located, the decoder employs mathematical techniques, such as the Berlekamp-Massey algorithm [35], to reconstruct the original message polynomial, effectively correcting the errors.
It is possible to summarize the decoding stages as follows [17][23]:

- Syndrome generator
- Key equation solver (determine error-location polynomial and error evaluator polynomial)
- Solve the error locator polynomial
- Calculate the error magnitude
- Error correction

In the next sections these stages are going to be described in detail.

Syndrome Generator

The first step in the decoding process is to understand if there are any errors in the codeword or not. To do so it is necessary to evaluate the syndromes, where a syndrome polynomial is represented as

$$S(x) = S_0 + S_2x^2 + ... + S_{2t-1}x2t - 1 \tag{3.12}$$

If we assume P(x) is the received codeword polynomial

$$R(x) = R_0 + R_1x^2 + ... + R_{n-1}x^{n-1} \tag{3.13}$$

then n^{th} syndrome can be expressed as such

$$S_i = R_0 + \alpha^i(R_1 + \alpha^i(R_3 + ... + \alpha^i(R_{n-1})...))) \tag{3.14}$$

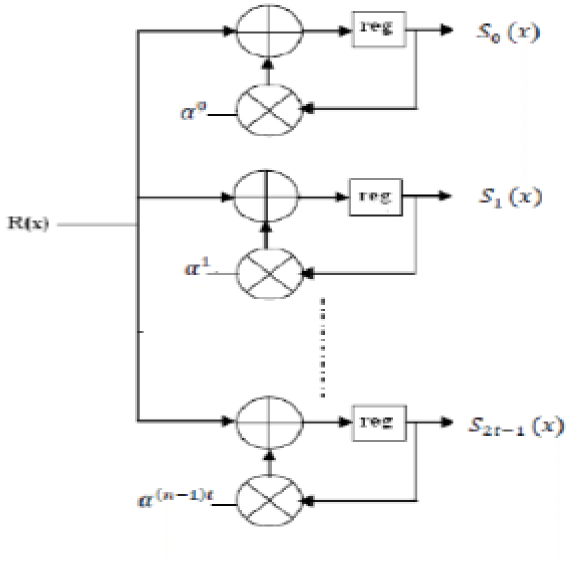


Figure 3.11: Architecture for Syndrome Generator [17]

Key Equation Solver

After computing the syndrome polynomial, the subsequent step is to determine the error values and their corresponding positions. In this stage, the 2t syndrome polynomials, which were generated in the prior step, are solved. These polynomials involve v unknowns, where represents the number of errors in the received codeword.

If the unknown error positions are identified as $i_1, i_2, ..., i$, the error polynomial can be formulated as follows:

$$E(x) = Y_1 x^{i_1} + Y_2 x^{i_2} + ... + Y_v x^{i_v} \tag{3.15}$$

where Y_p represents the magnitude of the error p^{th} at location i_p.
The error location polynomial is given from the solution for the coefficients of the error locator polynomial $\sigma(x)$.

$$\sigma(x) = \sigma_v x^v + \sigma_{v-1} x^{v-1} + ... + \sigma_0 x + \sigma \tag{3.16}$$

The polynomial 3.16 is connected to the error value polynomial through a specific equation referred to as the key equation. When the error value is represented by $\omega(x)$, this key equation is articulated as follows:

$$S(x)\sigma(x) = \omega(x) mod x^{2t} \tag{3.17}$$

These equations are resolved through different methods. In this study, we employ the "inversion-less Berlekamp-Massey algorithm" (iBM algorithm).

Solving of the Error Locator Polynomial

In the next stage he Chien-search algorithm is employed to identify error positions. In this method, the roots of the error locator polynomial represent the inverse error locations in the codeword. This algorithm systematically tests all possible input values, checking if the outputs result in zeros. The computation involves summing the odd values (1, 3, 5, ...) on one side and the even values (0, 2, 4, 6, ...) on the other. The two sums are then added together. If the summation value becomes zero within any clock cycle (<n), the position of that cycle determines the error location.

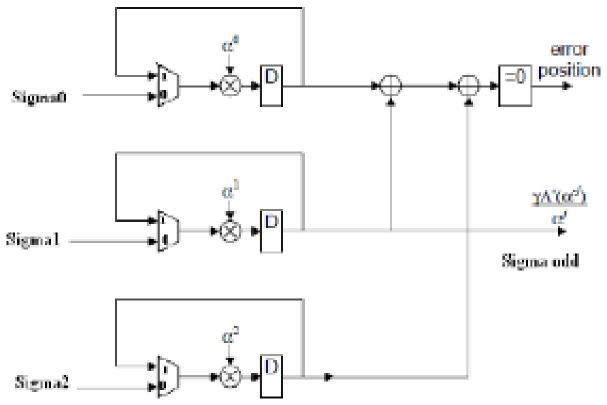


Figure 3.12: Architecture for Chien-Search Algorithm [17]

Calculation of the Error Magnitude

In order to evaluate the error values Forney algorithm is applied, with the input comprising both the error position and the coefficients of $\omega(x)$.Additionally, it utilizes a Galois field multiplier similar to the Chien-search algorithm. The formula for determining the error values is expressed as:

$$Y_i = \frac{\omega(x_1^{-1})}{\sigma'(x_i^{-1})} \tag{3.18}$$

In 3.18 x_i^{-1} the root as computed from the Chien-search and $\sigma'(x)$ denotes the derivative of the error locator polynomial.

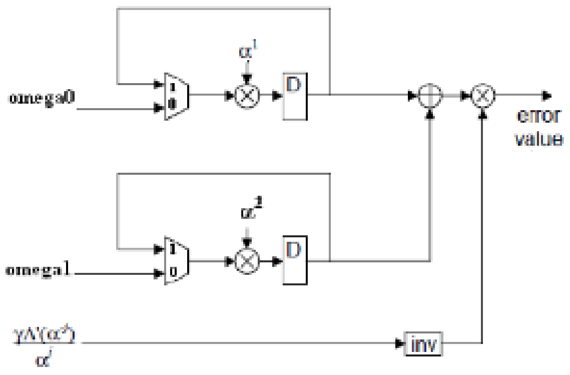


Figure 3.13: Architecture for Forney Algorithm [17]

Error Correction

After determining both the positions and extents of errors, the error correction module processes the received code. It conducts XOR operations between the received code and the computed error magnitudes at their specific positions. This operation results in obtaining the corrected message symbols.

3.3 Scrambling

Scrambling in digital communication refers to the process of systematically altering the arrangement of bits in a data stream before transmission [1]. This technique is used for various purposes, primarily to enhance the quality and security of data transmission.
Scrambling can help in reducing the occurrence of long sequences of identical bits, which might be problematic in certain communication systems. For instance, long sequences of 0s or 1s can lead to synchronization issues or loss of clock signal. Scrambling disrupts these sequences, ensuring a more balanced distribution of bits, which aids in synchronization and reduces the chance of errors during transmission. Another application of scrambling is encryption and security. By rearranging the bits in a specific manner known only to the sender and the receiver, the transmitted data becomes unintelligible to anyone who intercepts it without the correct decryption key. This provides a level of security and privacy for the transmitted information.
Scrambling techniques can be also useful while working with communication channels. In fact, certain frequencies might get attenuated more than others and scrambling can be used to spread the energy across various frequencies, ensuring that no specific frequency range is excessively impacted. This helps in equalizing the channel, improving the overall quality of the transmitted signal.
In essence, scrambling is a technique employed to optimize data transmission, enhance security, and ensure reliable communication in digital systems. Two

common scrambling techniques are Bipolar with 8-zero substitution (B8ZS) and High-density bipolar3-zero (HDB3) and the choice of the best technique is based on the application and the goals of the communication system.

4 Analysis and results

The central focus of this thesis revolves around the development of a system that can provide a reliable data exchange. In the analysis a particular attention is brought to the following aspects:

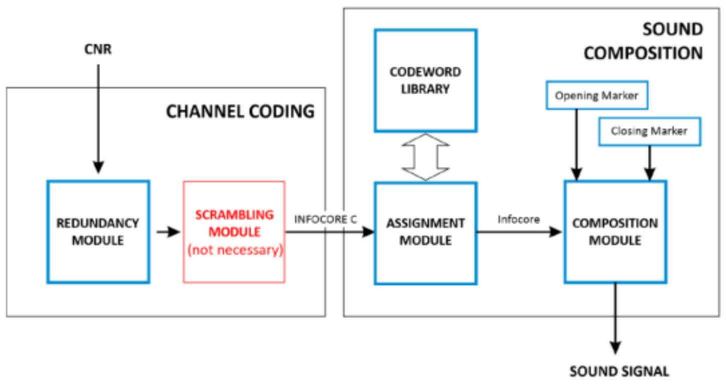
- Assessing the appropriateness of employing Reed-Solomon codes in the context of data exchange through sound.
- Investigating whether variations in amplitudes can have a positive impact on the system.
- Exploring the feasibility of enhancing the system's resilience to selective noises, a significant challenge in urban environments.

The designed system consists in two parts: the encoding module and the decoding module. These two parts are highlighted in the 4.1 and discussed in detail in the following sections.

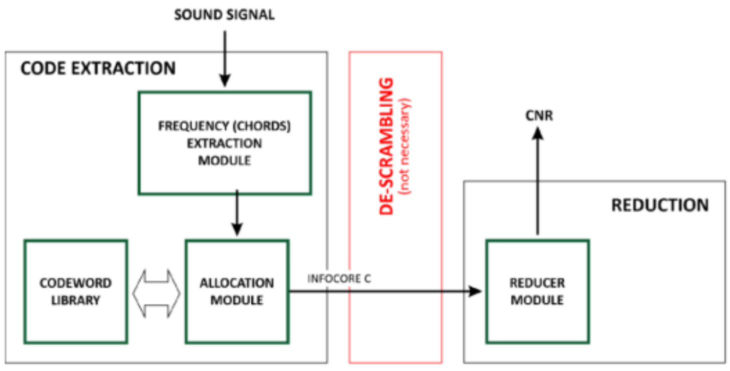
During the encoding process, the data of different type of format are submitted to an external software module that generates an identification code. The identification code is translated into binary format. These binary string undergoes different manipulations and the final result of this process is an audio file of a total length of around 4 seconds, that comprises two distinct components. The first component is the "opening mark", a sound element that remains consistent across all encoded messages. The second component, known as the "infocore," is the sound responsible for actually encoding the information. Between these two components, there is a deliberate insertion of a 2-second interval of silence. This pause is used in order to allow the system to activate. In this thesis the focus was brought on the exchange of the "infocore" as the other component was already designed.

The second part of the application concerns the retrieval of data. This process starts when a device equipped with a microphone and the application, developed by AppyTech, captures the audio signal. This activation occurs automatically as soon as the opening mark within the audio file is detected and the recording begins. The recording lasts about 4 seconds in order to completely capture the "infocore". Later, the captured sound is trimmed to avoid useless information and undergoes a decoding process. The trimming process is performed using the periodogram to estimate the power spectral density of the notes and of the received message. In this way it is possible to select the section where the most of the frequencies are contained.

The trimmed and cleaned sound is passed to the decoder that in response, furnishes the key required to access the data contained within a remote database. This process is invisible to the end user. Following the successful retrieval of the key, the application autonomously retrieves the data and transmits it to the requesting device.
In the upcoming sections the following concepts are going to be discussed: in chapter 4.1 the initial preparations and the choices are presented, in chapter 4.2 the structure of the encoder is discussed, in the chapter 4.3 the decoder is discussed, chapter 4.4 contains the final analysis of the model.



(a) Encoder



(b) Decoder

Figure 4.1: Block diagram

4.1 Initial preparation

For the project discussed in this thesis Python integrated with MATLAB API [5] was used. This integration of the two languages allowed to have the benefits of both. For example it was possible to use functions such as *rsenc()* and *rsdec()* inside the Python program, while the usage of the *numpy* library simplified the collection of the data. The following setup was used for the development of the system.

Figure 4.2: Setup used to develop the project

Before the encoding and decoding processes are possible, some initial preparations are needed. The first step involves creating a suitable frequency alphabet, based on the number of bits designated for the encoding. With each segment requiring 14 bits, the alphabet must consist of

$$2^{14} = 16384$$

components. For this project the frequencies were selected in the range with the minimum frequency of 415.30 Hz and the maximum frequency of 4978.03 Hz. This choice was made after some considerations. The first idea was to use the piano notes. From these frequencies it was necessary to remove the 15 lowest frequencies in order to avoid interference with the alternating current (50Hz-60Hz). For this reason the number of the notes that could be used were insufficient. In the end there were selected 88 notes. These 88 notes were arranged into triplets, resulting in a total of 315,735 arrangements. This number is calculated using the expression

$$C\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

where $n = 87$ and $k = 3$. It is important to note that the project's primary focus is on building a reliable system, for this reason melodic considerations were not taken into account and may be explored in future studies.

The frequencies were selected starting from a middle range, and then the other two frequencies were obtained by summing and subtracting of values. For detailed frequency ranges, refer to Table 4.1.

f_1	f_2	f_3
f_2-2 - f_2-18	18-50	f_2+3 - f_2+35
16	8	16

Table 4.1: Division of the ranges for the mapping

These ranges were carefully chosen to prevent issues related to overlapping frequencies. It is evident that the three frequencies are also in an ascending order, this idea was used in order to simplify the decoding process and also to avoid errors between similar frequencies.
For a better understanding a pseudocode of the function is shown.

Algorithm 1 Alphabet creation

```
$ncodewords \leftarrow 16384$
$alphabet \leftarrow array(3, ncodewords)$
$ranges \leftarrow range(3, 35)$
$ranges \leftarrow range(18, 50)$
$ranges \leftarrow range(2, 18)$
$i \leftarrow 0$
while $i \leq ncodewords - 1$ do
    for m,n,h in ranges do
        $alphabet \leftarrow Notes[m, n, h]$
        $i \leftarrow i + 1$
    end for
end while
return $alphabet$
```

After the construction and analysis of the alphabet it is possible to proceed with the encoding process.

4.2 Encoder

The encoder and the decoder codes are developed in Python using a MATLAB API. This procedure allowed to use some of the MATLAB features for a better analysis and functioning. The encoding mechanism, which is summarized in the following pseudo code, is designed to convert binary strings, into an audible formats. These binary strings of 42 bits are referred to as source code and they are generated by Omniaweb. The 42 bits are an external constraints of the company. During the encoding and decoding reed-solomon code is used. The parameters used during this analysis are n = 12, k=6 and m=7. This means that the total length of the message is 12 and the length of the information part is 6. Throught the reed-solomon six more elements are added. In the end a message of total 84 bits is used to generate the sound.

The process begins by structuring the input bits into an organized 6x7 matrix. This matrix, representing the binary data, undergoes a transformation into decimal values, followed by the encoding using Reed Solomon Coding. This step enhances the encoded data's robustness by introducing error detection and correction capabilities. The data are then transformed back into a binary string of 84 bits. In order to provide even more robustness at this point the data can undergo the scrambling process.

In the end of the process the 84 bits are divided into 14 bits segments that are transformed into six decimal numbers. Each of these numbers represents the index inside the alphabet. At each index corresponds three frequencies.

After selecting the three frequencies, these are used to create the sound. This process is divided in two parts. In the first part three sinusoidal signals are created each of 0.33 seconds. This choice of duration is made in order to obtain a final signal of 2 seconds. The second part involves a frequency mixer function, a component that takes as an input the different frequencies to compose one sound. In total six sounds of three frequencies are merged together to generate the final audio signal that accurately represents the encoded binary information.

Algorithm 2 Encoder

```
$bitsin \leftarrow reshape(bits)$ {Transforms the binary string into a matrix (6,7)}
starts the matlab engine
$bitsdec \leftarrow dec(bitsin)$
$'bitsdec' \leftarrow matlab.double(bitsdec)$
$msg \leftarrow' ReedSoloEnc(bitsdec, n, k)'$
$enc \leftarrow bin(msg)$
$bin14 \leftarrow reshape(enc)$
$index \leftarrow dec(bin14)$
optional $scr \leftarrow scramble(bin14)$
**for** $i \leftarrow range(0, len(bin14))$ **do**
    $bits84 \leftarrow string(scr[i]) or string(bin14[i])$
**end for**
$sound \leftarrow createsound(index)$
**return** $audiofile, bits84$
```

The resulting audio data is saved in a WAV file format, ensuring compatibility and fidelity during the encoding process. Moreover, the code includes functionality to convert the generated WAV file into the MP3 format, optimizing the audio data for practical use and convenient transmission. the final step is adding the 'opening marker' to the generated sound in order to obtain the complete 'tiscode', that can be spread.

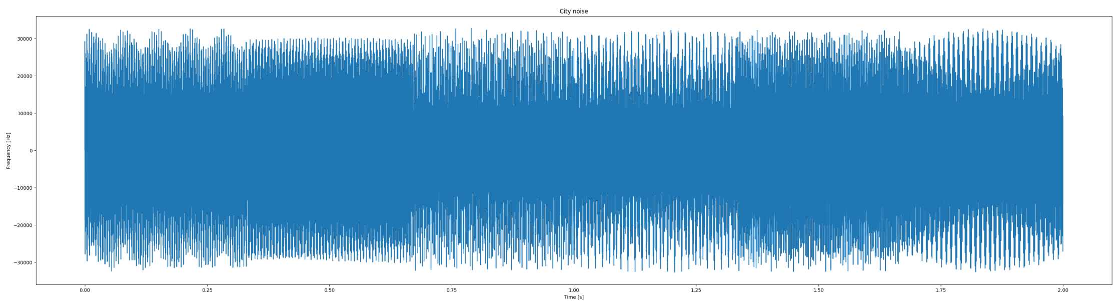


Figure 4.3: Example of an encoded sound

4.3 Decoder

The decoding process integrates signal processing techniques, frequency analysis, and error correction through Reed-Solomon codes [**RSdecoder**]. This process is handled by the decoder function that can be divided into two sections, illustrated in the pseudo-code. In the first section the indexes of the frequencies are obtained and in the second they are used to go back to the binary string that represents the original ID.
The function takes parameters such as the signal, sample frequency, alphabet, and Reed-Solomon decoding parameters (n, k, m) as inputs and returns a binary string that represents the ID.
In the first part the input signal is divided into six segments of 0.33 seconds of 5333 samples, and each segment undergoes a decoding process independently. In particular each segment is given as an input to another function, 'decode', that returns the three frequencies that compound the sound.

The 'decode' function is the focal point of the decoding process. It performs the signal analysis starting by taking as input an audio signal, sample rate, and a set of musical notes. The function employs the 'signal.spectrogram' method from SciPy to calculate the spectrogram of the audio signal, providing information about frequency content over time. The spectrogram uses the 'blackman' window as it was showing a better result during noisy testing. The output of the spectrogram returns the frequencies, the times and the spectral power density.

The top 7 frequencies with the highest power are extracted for each time segment of the spectrogram. The frequencies then undergo a control, the ones falling outside a specified range are filtered out, and the remaining frequencies are matched to the closest musical notes. The resulting frequencies are sorted and returned as an output, this last step is useful for the next processes. Notably, the code is structured to handle scenarios where suitable frequencies might not be found, in which case it returns 'None'.

After the three frequencies are returned their common index is searched in the alphabet using the 'find-idx' function. When all six indexes are found the process continues in the second part.
In the second section the code iterates through combinations of indices, attempting to find the correct combination that matches the received signal. The process is controlled by flags and includes checks for specific conditions, filtering out irrelevant frequencies and ensuring the correctness of the decoded message.

The Reed-Solomon decoding is a critical aspect, introducing redundancy to the data for error correction [34]. The function outputs information such as the reconstructed message ('rinfo-id'), the total number of errors made during decoding ('n-errors'), and the number of errors corrected by the Reed-Solomon code ('n-corrected').

Algorithm 3 Decoder Algorithm

```
1: $var \leftarrow initialvalues$
2: $audio$ ← the 6 segments of the sound {The segments are passed to the 'decode' function that returns three frequencies}
3: $indexes \leftarrow find\_idx(frequenies)$
4: comb ← combination of the indexes
5: idx_binary ← convert_binary14(comb)
6: idx_dec ← convert_decimal(idx_binary) {If the encoded data were scrambled, unscramble}
7: n_errors ← sum(bit1 != bit2 for bit1, bit2 in zip(str1, str2))
8: msg_dec ←$' ReedSoloDec'$
9: $id \leftarrow string(bin(msg\_dec))$
```

4.4 Results

This section summarizes the results obtained during the testing of the previously described system. The first analysis, which is not mentioned in the thesis, was made by simply checking if the encoded sound could be decoded correctly

without any noise addiction. The following step was to add different types of noise and SNR and observe the behaviour of the system. Finally the test with different amplitudes was performed. In this last part of the study the idea was to use different amplitudes calculated from the 14 bits used for each index, that is used to retrieve the three frequencies from the alphabet. The amplitude used to create the components of the final sound can have two values: low and high. The high values is 33% higher than the low.
The noises taken into account are the following:

- AWGN (Additive white Gaussian noise)
- Grocery store noise [11]
- City noise [9]
- Selective noise [10]

The selection of these noises is directly linked to the application of the developed system. In fact, the application will be used in a daily context, and for this reason it is necessary to ascertain that the system performs its task efficiently if used in urban environment. The different noises are shown below.

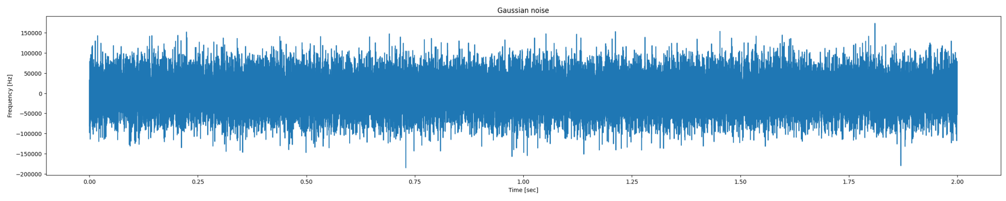


Figure 4.4: Gaussian noise

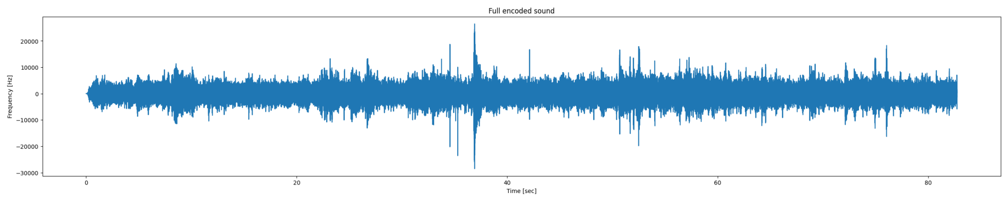


Figure 4.5: Grocery store noise

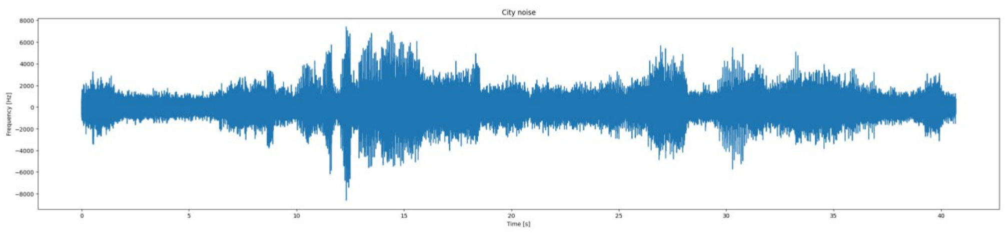


Figure 4.6: City noise

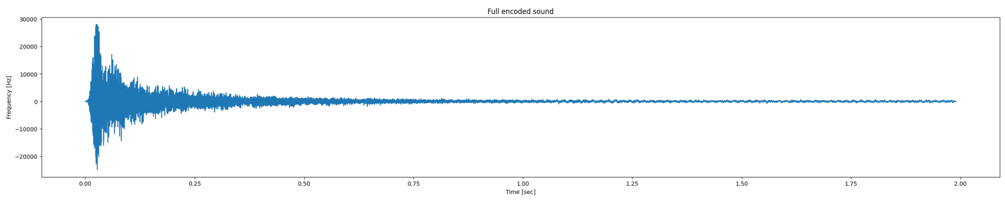


Figure 4.7: Car honk noise

4.4.1 Additive white Gaussian noise

The white noise is not useful for real life applications, but it is interesting to observe how it can affect the performance of the system. The noise used for the analysis was created using the following function.

```
$snr \leftarrow 10 ** (snr\_dB/10)$
$signal\_power$
$noise \leftarrow np.random.normal(0, 0.1, len(signal))$
$noise\_power$
$scalingfactor \leftarrow \sqrt{signal\_power/(snr * noise\_power)}$
$noise \leftarrow scalingfactor * noise$
**return** noise
```

The noise is generated by using 'random.normal' module of the Numpy library. It is possible to see that the mean is set to zero, the standard deviation is set to 0.1 and the size of array is the same of the original sound. The scaling factor is used in order to make the two sounds comparable.

For the white noise two experiments were performed. In the first only one id was used to generate the sound. The only sound was then added to different

noises randomly created at different SNRs. This was repeated several times to obtain an average number of errors and corrected bits.
The second experiment was done by using different ids and in this way creating different sounds. For each SNR were used the same noise with the different sounds and the results averaged.
This procedure was also repeated by using different amplitudes and the results are reported in the following graphs.
For both the experiments without the addiction of the different amplitudes the SNR was chosen in the range between -22dB and -10dB, while in the other case the range was set between -26dB and -8dB. It is possible to observe that for lower SNR the number of errors grows drastically while the number of corrected messages gets smaller. This is caused by the fact that the noise overpowers the signal and a lot of frequencies are lost. Both experiments shows similar results. From the graphical representation is it possible to notice also that when different amplitudes are used, the system reaches its limit for lower SNR and it is able to correct more errors. The reason of this behaviour is that some signals having higher amplitude are still decoded, even when the noise is stronger.
It is interesting to observe how the different values of SNR affect the sound and its spectrogram. The color blue indicates lower values of PSD while the yellow indicates the higher values.
In the first two images the SNR was set to 10 dB, this means that the power of the audio signal is ten times bigger than the power of the noise. In this case we can clearly distinguish the 3 main frequencies used to create the sound, even though in 4.9 seven frequencies are shown some of them are very close to each other. In the last two images it is evident how it is no longer possible clearly distinguish the frequencies of the sound from the noise, this is also confirmed by the clear presence of seven different frequencies in the 4.11 where the frequencies are very distant between them.

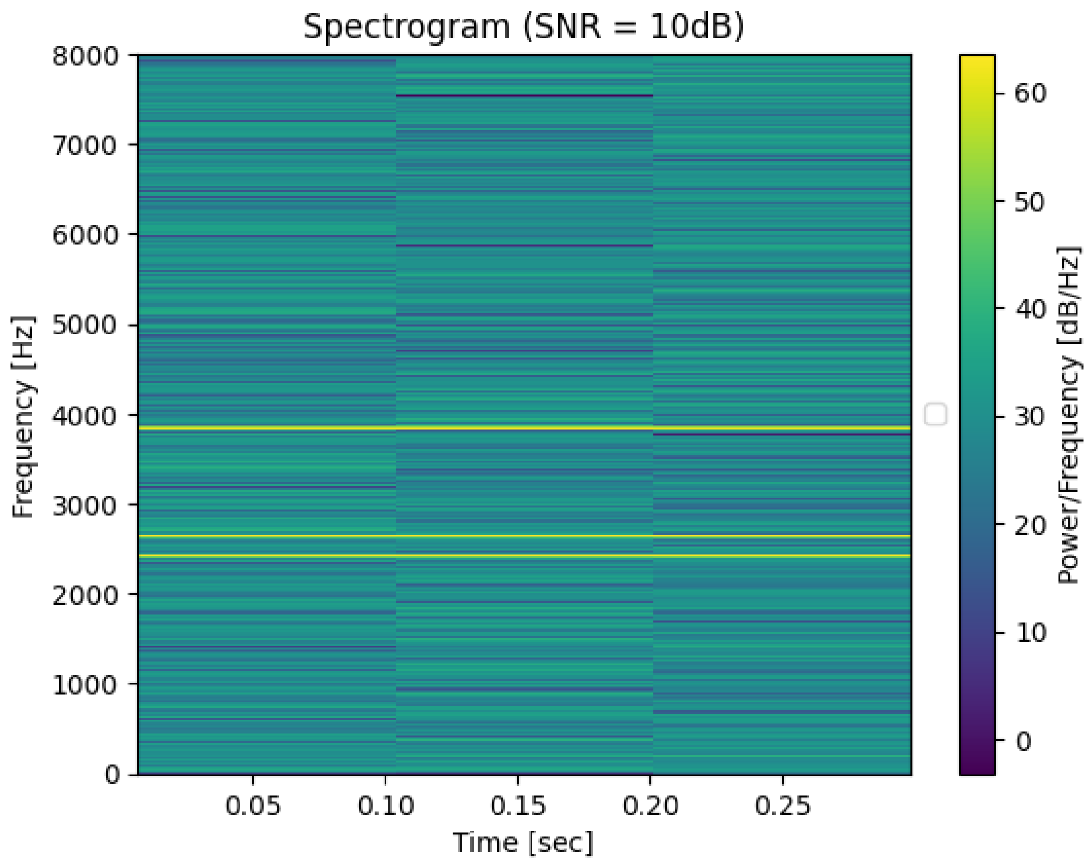


Figure 4.8: Spectrogram of a section of sound at SNR equal to 10dB

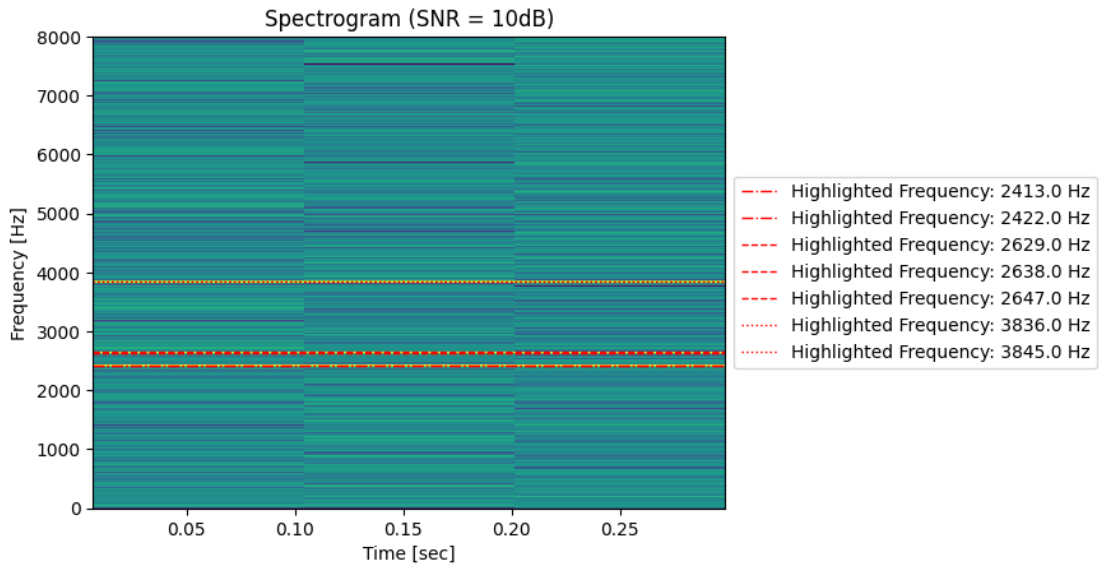


Figure 4.9: Spectrogram of a section of sound at SNR equal to 10dB

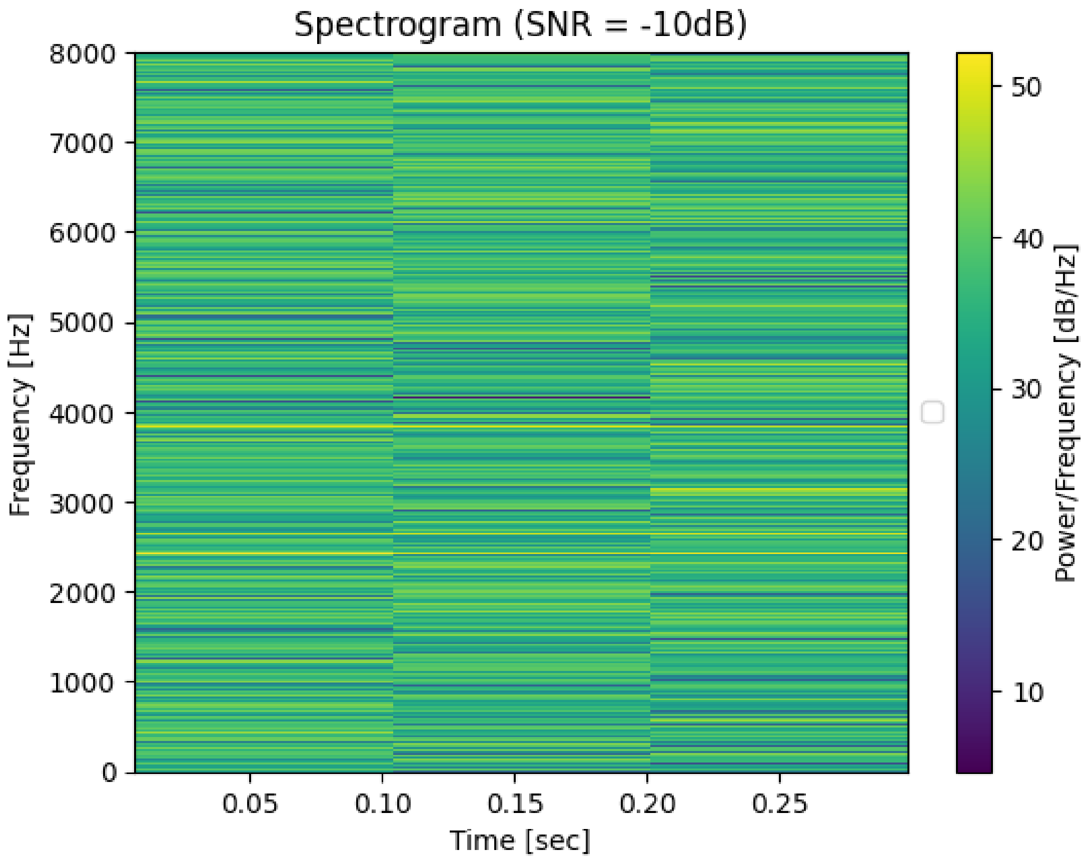


Figure 4.10: Spectrogram of a section of sound at SNR equal to -10dB

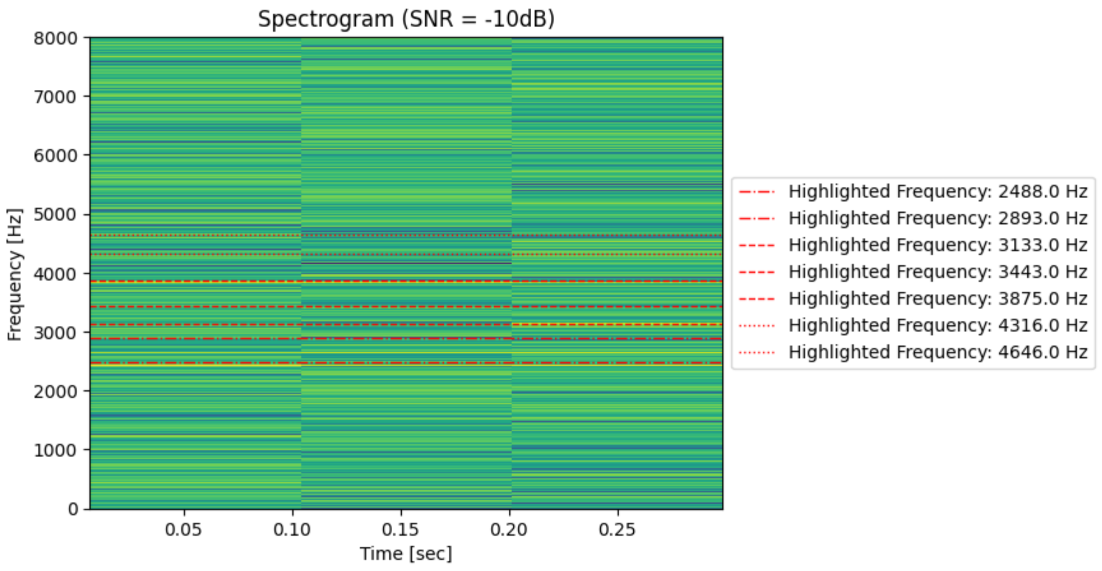
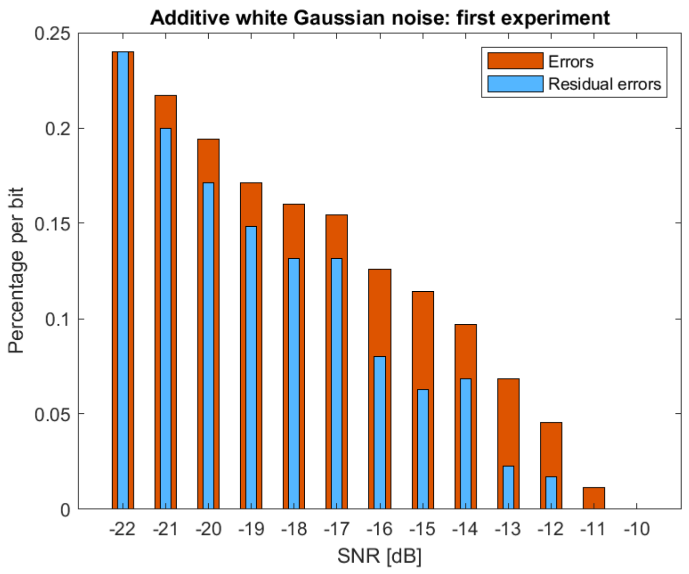


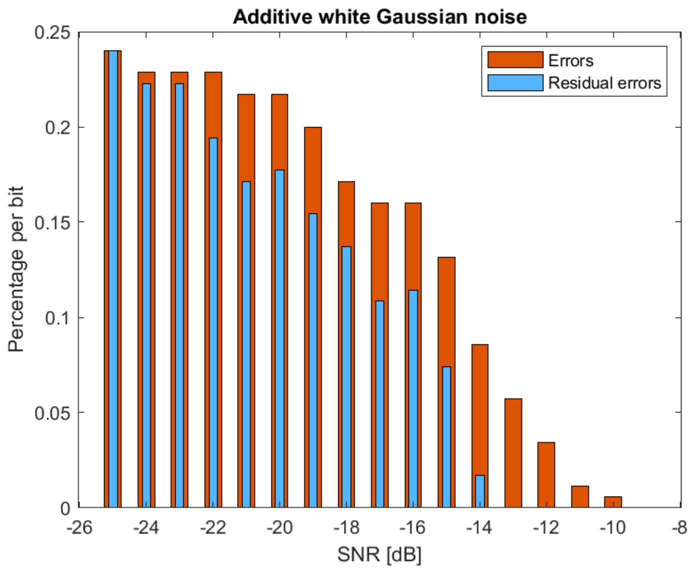
Figure 4.11: Spectrogram of a section of sound at SNR equal to 10dB

First experiment

For the first experiment only one id was used, in order to test the system. **ID:** "111010110110111110010010101100111010001011" In this test one sound was used with different noise and SNR.



(a) Performance of the system with AWGN



(b) Performance of the system with AWGN with different amplitudes

Second experiment

In the second test 15 different ids were used. The binary strings were generated as a random sequence of "0" and "1". For each SNR, only one noise was used to test the response of the system with the 15 different inputs. The results for each SNR were averaged in order to obtain the data represented in the graphs.

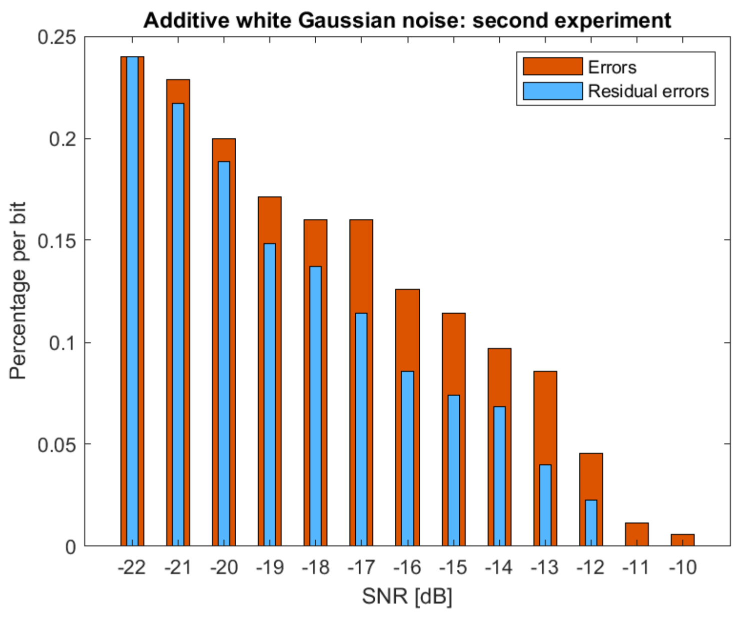


Figure 4.13: Performance of the system with AWGN

The results for both the experiments, with or without considering different amplitudes, are very similar. For this reason in the next graphs only the second type of experiment was considered.

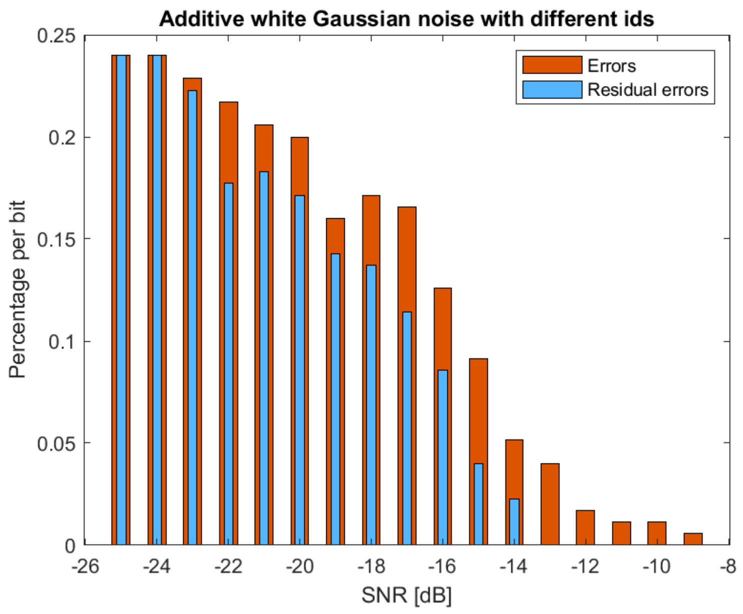


Figure 4.14: Performance of the system with AWGN with different amplitudes

4.4.2 Grocery store noise

The audio signal used in this section is a recording of the environment noise in a german grocery store [11]. It is possible to listen to people talking, sounds from the streets and the typical sound of the scan. In the following graphs the results are reported.
It is possible to observe a significant change in the system's response to Gaussian noise compared to its performance in a more complex environmental noise, like in this case. Specifically, it becomes evident that the system encounters difficulties when the SNR is below -6dB. This challenge arises from the increased diversity of the recorded noise across various frequency ranges. This differences introduces alterations in the spectrum levels, impacting the accurate detection of frequencies and consequently resulting in errors during the decoding process.
Also in this case the graphs show that having different amplitudes makes the system more robust to the noise and less errors are made.

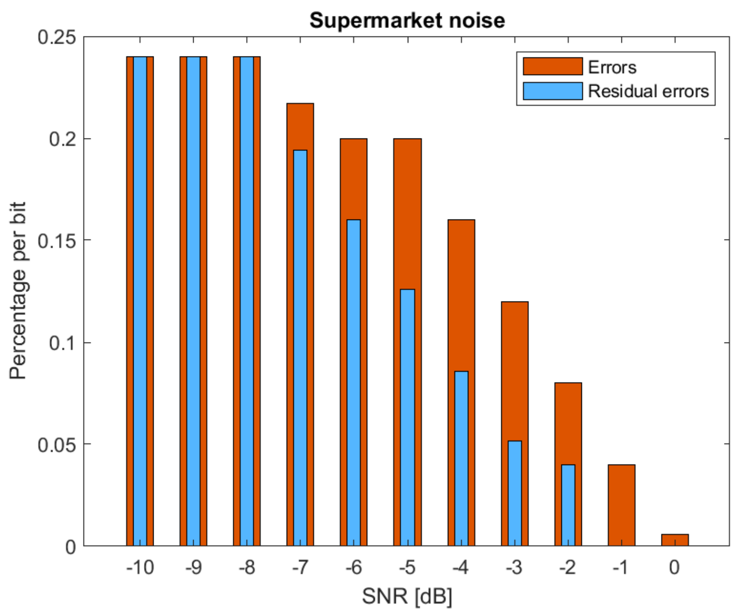


Figure 4.15: Performance of the system with grocery store noise

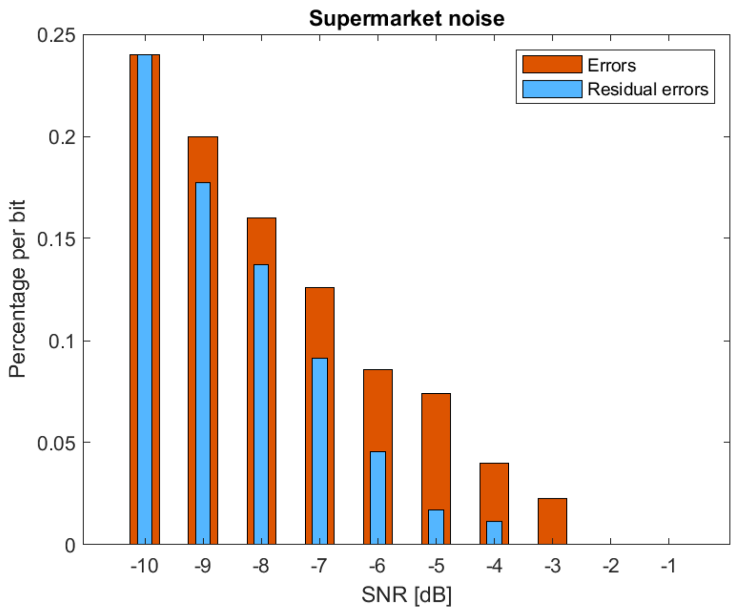


Figure 4.16: Performance of the system with grocery store noise and different amplitudes

4.4.3 City noise

The following analysis were performed on the system using the sound recorded in a crowded street of a city [9]. It is possible to observe the presence of common sounds of the street such the honking of the cars and the motors of the motorbikes. Also this noise shows clear differences if compared to the Gaussian one. In the following graphs it is possible to observe the results.

Also in this case, the system is not able to decode correctly the sound for values of SNR smaller than -7dB in the case when all the signals have the same amplitude.

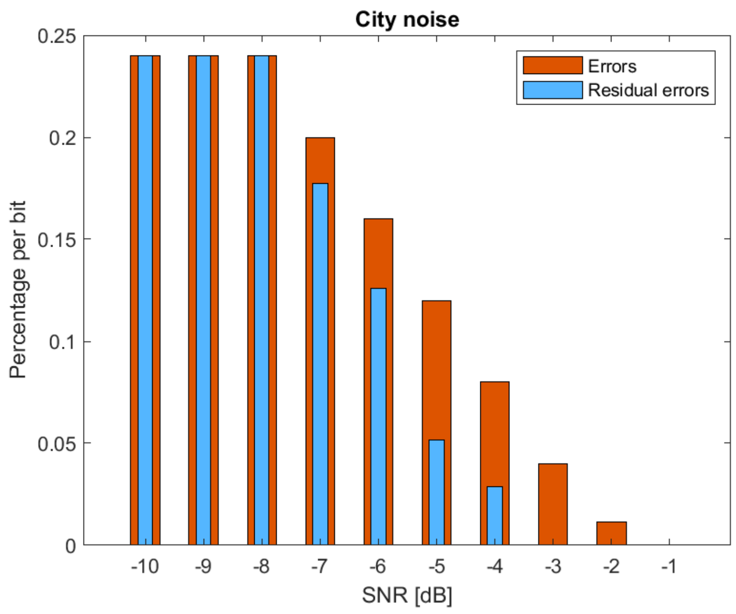


Figure 4.17: Performance of the system with city noise

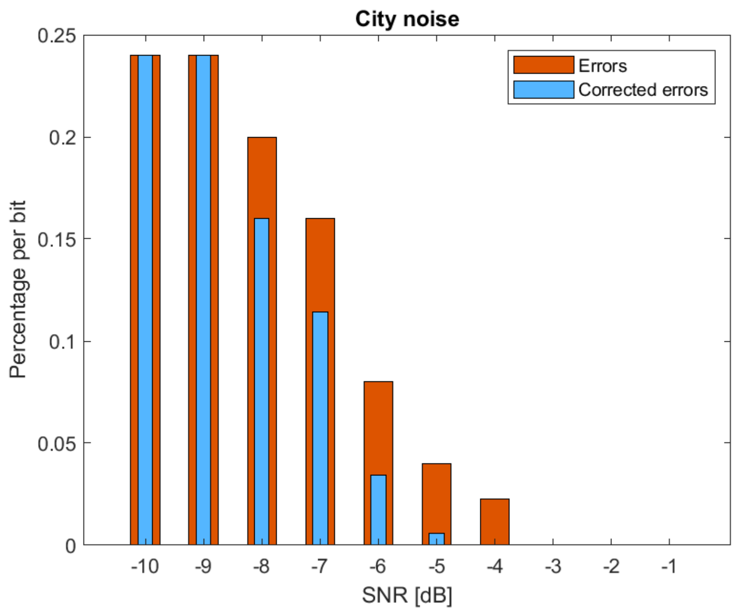
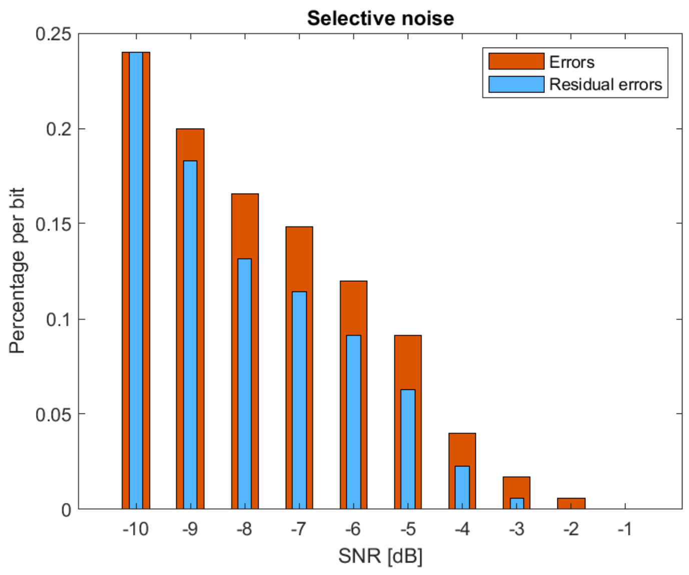


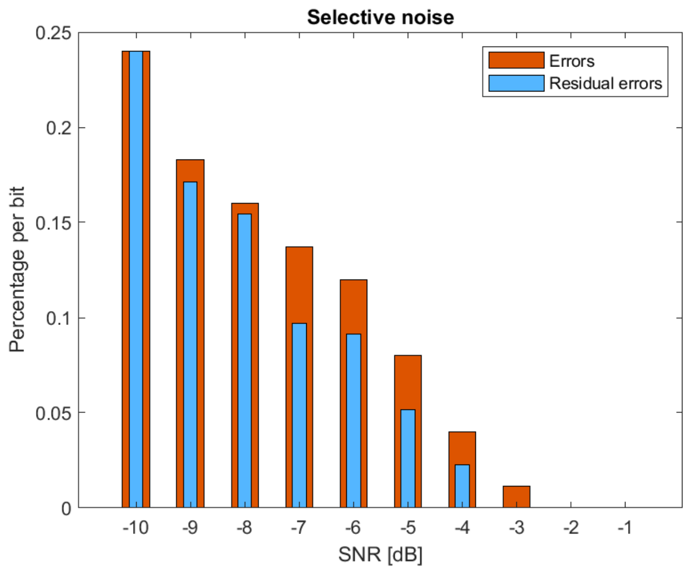
Figure 4.18: Performance of the system with city noise with different amplitudes

4.4.4 Car noise

The audio employed for this part of the testing consists of a recorded car honk. The objective in this phase is to analyse the system's response when subjected to noise that selectively influences only a portion of the original sound. Specifically, the higher frequencies are confined to a narrow segment of the audio, while the remainder is composed of Gaussian noise.
From the graphs it is possible to observe that in this case the number of errors for different SNR is lower, but so it is the number of corrected bits. This behaviour can be explained by the fact that the high frequencies of the used noise sound affect just a part of the original sound and some parts can still be delivered. At the same time the system struggles to reconstruct the message because a high number of bits is lost.
In this last example it is noticeable how the presence of different amplitudes is beneficial to the system, as the number of corrected symbols is larger. An important aspect to consider is the position of the noise peaks. During the simulations the number of errors and corrections was higher whenever the peaks were registered in the middle section, while it was the opposite when the peaks were positioned in the end or the beginning.



(a) Performance of the system with an isolated noise



(b) Performance of the system with car honk noise and different amplitudes

5 Conclusions and Future Implementations

5.1 Conclusions

The objective of this thesis was to develop a system, drawing inspiration from the techniques employed in QR codes [7], Shazam [38] and the error-correcting codes used in the underwater acoustic communications [40] [34], to allow data exchange through audio signals. In particular, the encoding mechanism was designed to generate a sound based on a unique ID linked to a piece of data, while the decoding process, executed by the receiving device, was designed to return the original ID from a recorded sound.

Throughout this project, various techniques were employed, such as scrambling technique that was utilized to enhance the corrective effect of Reed-Solomon algorithms.
The analysis confirms that the RS codes are a good solution to limit the number of errors if used in audio context. Additionally, it was demonstrated that using different amplitudes during encoding improves the performance of the system, if compared to methods that do not incorporate this feature. Lastly it was shown

that it is possible to contrast the selective noises if the single sounds are made of frequencies with different amplitudes.

5.2 Future Implementations

Based on the results obtained during this project's development, several aspects can be explored further.

One potential improvement involves adopting a different Reed-Solomon coding to correct a greater number of errors. A proposed solution is to employ the Reed-Solomon code to obtain 17 bits of code for each sound, allocating 14 bits for encoding the frequency and the remaining 3 for encoding the amplitude. The decoder would then perform amplitude demodulation, enhancing the overall system performance.
Another aspect that can be improved is the musicality of the sounds by following the music theory [22]. While this aspect was overlooked due to a lack of knowledge of music theory in this project, it is crucial to consider, especially since the intended application involves public spaces. The system should be pleasing to the ear or, at the very least, not overly disruptive.
A final potential enhancement involves noise cancellation. Analyzing the first one or two seconds of the recording and its spectrum could provide valuable information for subtracting noise from the recorded sound. This approach, particularly effective for constant noise, would simplify the isolation of individual frequencies during the decoding process.

The proposed system represents a highly promising technology with the potential to yield numerous benefits when implemented in practical scenarios. The core concept of this project is versatile, extending its applicability beyond the specific field described in this thesis. In recent years, data transmission has been a focal concern, emphasizing the crucial need to mitigate errors during this process. Various techniques have already been devised and implemented to tackle such challenges [47] and the proposed project can be a valuable resource to continue this ongoing effort towards achieving more reliable and secure data transmissions.

References

[1] In: URL: https://www.mathworks.com/help/hdlverifier/ug/frame-based-scrambler-using-communications-toolbox.html.

[2] In: URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.get_window.html.

[3] In: URL: https://it.mathworks.com/help/comm/ug/error-detection-and-correction.html#fp12225.

[4] In: URL: https://it.mathworks.com/help/signal/ref/periodogram.html.

[5] In: URL: https://it.mathworks.com/help/matlab/matlab-engine-for-python.html.

[6] “"What is Reed Solomon code?"” In: URL: https://www.geeksforgeeks.org/what-is-reed-solomon-code/.

[7] “A brief history of QR codes.” In: April 28, 2023. URL: https://www.microsoft.com/en-us/microsoft-365-life-hacks/privacy-and-safety/brief-history-qr-codes.

[8] Marco Alecci et al. “Development of an IR System for Argument Search.” In: 2021.

[9] “Audio recording of a city”. In: URL: https://freesound.org/people/embracetheart/sounds/345313/.

[10] “Audio recording of a honk from a car”. In: URL: https://freesound.org/people/ChrisReierson/sounds/433578/.

[11] "Audio recording of the noise inside of a supermarket". In: URL: https://freesound.org/people/haraldgoetz/sounds/615151/.

[12] 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.

[13] Leonardo Badia et al. "Cognition-based networks: Applying cognitive science to multimedia wireless networking". In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. 2014, pp. 1–6. DOI: 10.1109/WoWMoM.2014.6918997.

[14] K. V. Bisina and Maleeha Abdul Azeez. "Optimized estimation of power spectral density". In: *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2017, pp. 871–875. DOI: 10.1109/ICCONS.2017.8250588.

[15] 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. 185–282. DOI: 10.1002/j.1538-7305.1958.tb03874.x.

[16] Ronald N Bracewell. "The fourier transform". In: *Scientific American* 260.6 (1989), pp. 86–95.

[17] Anindya Das, Satyajit Das, and Jaydeb Bhaumik. "Design of RS (255, 251) Encoder and Decoder in FPGA". In: *International Journal of Soft Computing & Engineering* 2 (Jan. 2013), pp. 391–394.

[18] Paulo SR Diniz, Eduardo AB Da Silva, and Sergio L Netto. *Digital signal processing: system analysis and design*. Cambridge University Press, 2010.

[19] 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. 51–83. DOI: 10.1109/PROC.1978.10837.

[20] 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. 51–83. DOI: 10.1109/PROC.1978.10837.

[21] Paul Heckbert. "Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm". In: February 1995. URL: https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf.

[22] Michael Hewitt. *Harmony for computer musicians*. Cengage Learning, 2011.

[23] R.S. Lim. "A Decoding Procedure for the Reed-Solomon Codes". In: *Technical Publication (TP)*. August 1, 1978.

[24] Erica 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] 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.

[26] Bernard Mulgrew, Peter Grant, and John Thompson. "Digital signal processing: concepts and applications". In: (2002).

[27] "Non-Binary BCH Codes: Reed-Solomon Codes". In: *The Art of Error Correcting Coding*. John Wiley & Sons, Ltd, 2002. Chap. 4, pp. 61–72. ISBN: 9780470847824. DOI: https://doi.org/10.1002/0470847824.ch4. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470847824.ch4. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470847824.ch4.

[28] Henri J Nussbaumer and Henri J Nussbaumer. *The fast Fourier transform*. Springer, 1982.

[29] Alan V. Oppenheim. “Speech spectrograms using the fast Fourier transform”. In: *IEEE Spectrum* 7.8 (1970), pp. 57–62. DOI: 10.1109/MSPEC.1970.5213512.

[30] Mark Petersen. “Mathematical Harmonies”. In: URL: https://amath.colorado.edu/pub/matlab/music/MathMusic.pdf.

[31] “QR Code (2D Barcode)”. In: URL: https://www.tec-it.com/en/support/knowbase/symbologies/qrcode/Default.aspx#:~:text=Data%20capacity%3A%20A%20single%20QR,according%20to%20JIS%20X%200208).

[32] Liao Qunying. “On Reed-Solomon codes”. In: *Chinese Annals of Mathematics. Series B* 32 (Jan. 2010), pp. 89–98. DOI: 10.1007/s11401-010-0622-3.

[33] Tom Read. “45+ QR Code Facts, Statistics, Trends 2023”. In: (2023). URL: https://www.websiteplanet.com/blog/qr-code-statistics/#hselect--9.

[34] I. S. Reed and G. Solomon. “Polynomial Codes Over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018. eprint: https://doi.org/10.1137/0108018. URL: https://doi.org/10.1137/0108018.

[35] “Reed Solomon code”. In: URL: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html.

[36] “rMQR”. In: URL: https://www.qrcode.com/en/codes/rmqr.html.

[37] Chulwon Seo et al. “Performance comparison of convolution and Reed–Solomon codes in underwater multipath fading channel”. In: *Japanese Journal of Applied Physics* 53.7S (June 2014), 07KG02. DOI: 10.7567/JJAP.53.07KG02. URL: https://dx.doi.org/10.7567/JJAP.53.07KG02.

[38] Emma. Sheppard. “Shazam co-founder: "We were growing a business in a collapsing market"”. In: December 7, 2016. URL: https://citeseerx.ist.psu.edu/doc/10.1.1.217.8882.

[39] Ian Naismith Sneddon. *Fourier transforms*. Courier Corporation, 1995.

[40] M. Stojanovic. “Underwater acoustic communications”. In: *Proceedings of Electro/International 1995*. 1995, pp. 435–440. DOI: 10.1109/ELECTR.1995.471021.

[41] Milica Stojanovic and James Preisig. “Underwater acoustic communication channels: Propagation models and statistical characterization”. In: *IEEE Communications Magazine* 47.1 (2009), pp. 84–89. DOI: 10.1109/MCOM.2009.4752682.

[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] Kokulathas Thurairajah et al. “Video over wireless, Channel Model and Forward Error Correction”. In: Nov. 2004.

[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: *Proc. ACM WUWNet ’10*.

[46] S.B. Wicker and V.K. Bhargava. *Reed-Solomon Codes and Their Applications*. Wiley, 1999. ISBN: 9780780353916. URL: https://books.google.it/books?id=yws55Rx1orEC.

[47] 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.