Cairo University, Faculty of Engineering
Electronics and Electrical Communications
Department (EECE)

SIGNALS DENOISING

Final report

Name	ID
Sarah Abdelatty Ibrahem	9220349
Khaled Ahmed Hamed Abd El-Aziz	9220276
Sohaila Ahmed Mohamed	9220377
Sama Saeed Mohamed	9220371
Abdelrahman Hatem Nasr Youssef	9220461
Yousef Khaled Omar Mahmoud	9220984
Abdelrahman Ahmed Mohamed Abdellatif	9220457
Eslam Fathy Mohamed Soliman	9220145
Ali Mohamed Mohamed Said	9210687

Abstract:

Audio denoising is a fundamental art making life easier. As we perform our jobs. we often make online meetings usually in low quality. In such cases, audio denoising plays an important role in eliminating background noise.

In our project, we implemented a radio simulator device with Arduino which represents the receiver to receive the audio and filter it, software application that process the sound files for the device and chose to focus on this valuable task by attempting to implement it using three techniques: Wavelet, FIR, and IRR. we tested with our own dataset, with various types of frequencies obtained at different stages.

We concluded that the wavelet method outperforms other methods, For instance, when we tested the parameter PSNR (peak signal-to-noise ratio), the wavelet method yielded 115.30 dB in high frequencies, while FIR only gave 13.77 db.

Table of content:

Introduction

In the realm of audio processing, denoising plays a pivotal role in enhancing the quality of sound recordings by mitigating unwanted background noise. Noise, characterized by random and undesirable audio signals, can be introduced during the recording process or emerge from various environmental sources. These unwanted artifacts can compromise the clarity and intelligibility of audio content, impacting its overall quality.

The denoising process involves the identification and isolation of the noise component within an audio signal, followed by its subsequent removal or reduction. Advanced algorithms and machine learning techniques have become instrumental in achieving effective denoising, allowing for the preservation of the desired signal without compromising its fidelity.

As technology continues to advance, audio denoising methods have evolved from traditional filters to sophisticated adaptive algorithms that can intelligently distinguish between signal and noise components. Real-time denoising applications have also become more prevalent, enabling immediate noise reduction during live performances, audio streaming, and communication.

In this introduction to audio denoising, we will explore the underlying principles, methodologies, and emerging technologies that contribute to the refinement of audio quality. Whether applied in professional audio production or everyday communication, the pursuit of pristine sound remains a constant goal, and audio denoising stands as a crucial tool in achieving this objective.

Types of audio denoising:

1. Statistical Denoising
2. Frequency Domain Denoising
3. Time Domain Denoising
4. Adaptive Filtering
5. Machine Learning-Based Denoising
6. Waveform Modeling
7. Time-Frequency Domain Denoising

Advantages of audio denoising:

1. Enhanced Clarity
2. Improved Quality
3. Better Communication
4. Increased Signal-to-Noise Ratio (SNR)
5. Reduced Listener Fatigue
6. Preservation of Original Content
7. Improved Speech Recognition
8. Adaptability to Varied Environments
9. Better User Experience

Problem Definition

With the development of technology over the centuries, the transfer of information, photos, and videos between people has become easier. However, the process of sending and receiving is not that easy, as the signals are exposed to a lot of noise that affects their quality and may lead to some details disappearing or appearing in a faded manner.

We will now review what this noise that affects signals is to realize the importance of working to get rid of it as much as possible. In signal processing, noise is a general term for unwanted modifications that a signal may experience during capture, storage, transmission, processing, or conversion. Sometimes the noise is random and unpredictable signals that do not carry any useful information; Even if it is not interfering with other signals or may have been introduced intentionally, as in the case of comfort noise.

Noise types are different such as Gaussian noise (random variations in the signal amplitude), Poisson noise (error in the statistical nature of electromagnetic waves), Salt and Pepper noise (errors in data transfer) ……, Noise reduction techniques exist for audio, images, and videos to enhance the image's quality by reducing the noise while preserving important details and structures. Denoising is a common goal in designing signal processing systems (especially filters), and its algorithms may distort the signal to some degree. This process is commonly used in various applications such as document processing, medical imaging, and digital. [1]

History of signal denoising:

The problem of audio noise has been a challenge throughout the history of audio recording and communication. Here's a brief overview of this issue:

Early Audio Recording (Late 19th Century):

Back in the late 1800s, people started recording sound using machines like the phonograph. But these early recordings had problems. The machines made their own noises, and that got mixed with the recorded sound.

Wireless Communication (Early 20th Century):

In the early 1900s, when radio was becoming popular, there were new issues. Radio broadcasts had trouble with interference – that means other signals or natural things like weather could mess up the sound, causing static or unwanted noise.

Analog Magnetic Tape Era (Mid-20th Century):

Around the mid-1900s, we got better with analog tapes for recording. However, tapes still had their own issues, like hissing sounds and fluttering. These were problems that came from the way analog tapes worked.

Digital Audio (Late 20th Century Onward):

In the late 1900s, we started using digital technology for recording sound. Digital was a big improvement because it could make cleaner and more accurate sound reproductions. But even digital recordings aren't perfect – they can have issues like weird noises and distortions.

Internet and Streaming (21st Century):

In the 2000s, we started streaming audio over the internet. This brought new challenges. Sometimes, the sound could get messed up because of internet issues like not enough data getting through or compressing the sound too much. Also, using phones and wireless gadgets created worries about the sound quality.

Noise Reduction Technologies:

Over time, we developed ways to fix these sound problems. In the past, we used methods like adjusting the sound or using special systems to reduce noise. Nowadays, with digital technology, we have smart computer programs that can make sound quality better.

Environmental and Ambient Noise:

Apart from technical problems, sounds from the environment have always been an issue. Background noise from where you record, like busy streets or crowded places, can make it hard to get clear and good-quality sound.

Audio Denoising:

Applications of audio signal processing all struggle with a major problem, noise. So, removing this noise has become essential.

1. Types of noises:

noise sources come from the used hardware in the processing, the noise which is produced from such sources is called “Internal noise”. other noise sources come from the nature in the case of transferring the audio signal over long distances using communication systems, the noise which is produced from such sources is called “External noise”.

Examples of the sources of “Internal noise” are amplifiers, transmitters, receivers, or wires. Examples of the sources of “External noise” are lightning, solar flares, cosmic rays, or atmospheric turbulence.

Another noise source is the noise which is added by the transmitter's side under the complete will to encrypt the audio before sending it to the receiver.

From the statements above we should realize the importance of the presence of noise removal techniques to get rid of the effect of this noise and restore the original audio without any change. There are two types of techniques for noise cancellation which are:

- **Noise reduction techniques:**
  Prevents the noise process from happening originally.
- **Audio filtering techniques:**
  Filters the audio from noise after it's made noise.

If the audio signal is a "Real-time signal", then the used noise cancellation technique is the "Noise reduction technique". But if the audio signal is a "non-real-time signal", then the used noise cancellation technique is the "Audio filtering technique", and this is some techniques to do audio denoising.

2. Solutions:

2.1. LMS filter method:

Audio noise reduction employs a least mean square (LMS) filter to minimize the difference between the desired clean signal and the noisy input by adaptively adjusting the filter coefficients. The LMS algorithm repeatedly updates the weights based on the error signal and gradually reduces the noise. In audio processing, this real-time adjustment improves the filter's ability to suppress unwanted noise, making it an effective tool for improving audio quality in a variety of applications such as speech recognition and music processing.

2.2. FIR filter method:

A Finite Impulse Response (**FIR**) filter is a digital filter commonly used in audio processing to remove noise from audio signals. It operates by convolving the input audio signal with a sequence of filter coefficients, which are typically designed to attenuate specific frequency components associated with noise. The **FIR** filter can reduce unwanted noise while preserving the desired audio content.

2.3. IIR filter method:

An Infinite Impulse Response (**IIR**) filter is a digital filter used for denoising purposes. Unlike FIR filters, IIR filters use feedback in their design, which allows for more efficient filtering with fewer coefficients. **IIR** filters can remove noise from audio signals by attenuating specific frequency components associated with noise. They are commonly used in applications where real-time processing or low computational complexity is desired. [2]

2.4. WAVELET TRANSFORM method:

This algorithm is different from the **FIR** and IIR where it doesn't care for the design of the used filters, but it divides the Fourier coefficients of the audio signal into "Approximation coefficients" using a low pass filter and "Detailed coefficients" using a high pass filter then using the threshold(soft/hard) it removes the unwanted part, then concatenate the output to produce the output audio.

2.5. Deep learning method:

This work presents an end-to-end approach to audio noise reduction using fully convolutional networks with deep feature loss. This method outperforms traditional methods and pre-trained speech classification networks and is ideal for difficult and noisy situations. This architecture focuses on convolutional context aggregation networks and feature loss to address SNR challenges and demonstrate improved performance without increasing complexity.

Literature review:

• FIR & IIR filters

In order to implement any low pass, high pass and band stop or pass filter, we can create an electric circuit for them, but if we want to implement them **digitally**, then weneed to put them in mathematical equations by relating their output signal to input signal using **convolution theorem** without the presence of complex numbers. While doing the previous we get Finite Impulse Response (FIR) filter or Infinite Impulse Response (IIR) filter. which each of them can be a low pass, high pass and band stop or pass.

FIR filters have a finite impulse response and linear phase characteristics, allowing them to remove specific frequency components associated with noise in audio denoising. They are typically implemented using a tapped delay line structure.

IIR filters can have a compact filter structure because of their infinite impulse response and feedback. Phase distortion may be introduced, but by highlighting specific frequency ranges in audio denoising, they can be used to reduce noise.

Implementation:

In order to implement any low pass, high pass and band stop or pass filter, we can create an electric circuit for them, but if we want to implement them digitally, then we need to put them in mathematical equations by relating their output signal to input signal using **convolution theorem** without the presence of complex numbers. While doing the previous we get Finite Impulse Response (**FIR**) filter or Infinite Impulse Response (**IIR**) filter. which each of them can be a low pass, high pass and band stop or pass. [3]

As an example, we will apply on the low pass filter:

By applying the Inverse discrete Fourier or wavelet transform (**IDCT** or **IDWT**) we get the filter's response in time domain. By sampling (using impulse function) and windowing method using rectangular window. we get the truncated impulse response of the filter because we have a limited resource digital system.

In order to put equations, we will return to frequency domain through **DFT** or **DWT** which is considered an approximation. By convolving of the input signal and impulse response for discrete time signal we get the output signal which will be in frequency domain.

Summary:

FIR and IIR filters are commonly used in audio denoising,

- FIR filters having a finite impulse response and linear phase characteristics,
- IIR filters may introduce phase distortion.

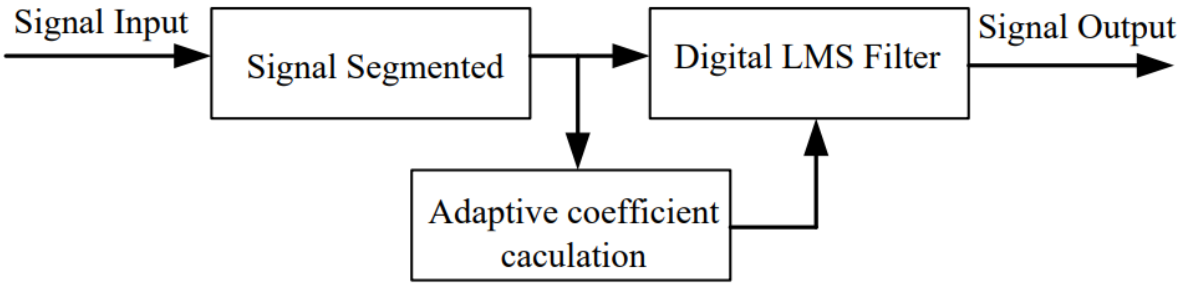
• LMS filter:

Noise reduction in audio signals is a significant challenge in speech enhancement, recognition, and communication applications. The optimal linear filtering method, including Wiener, Kalman, spectral restoration, and subspace methods, has been widely used to reduce noise and improve signal-to-noise ratio (**SNR**). However, the complexity of these algorithms makes implementation difficult. This paper proposes an LMS adaptive filtering algorithm, filtering the signal in the time domain and calculating filter coefficients adaptively.

LMS is an adaptive filtering algorithm used in audio denoising to estimate and remove noise by minimizing the mean squared error between the filtered and desired signal.

Noise Reduction Algorithm Based on LMS Filter:

The proposed noise cancelling scheme uses the **LMS** filtering algorithm for optimal performance. The block diagram illustrates the process. To achieve effective noise reduction, the input signal must be segmented every 40ms. The unprocessed noisy signal is segmented every 40ms, with a noisy test signal representing the frame at time t. This method is suitable for time-varying audio signals.



Flow chart 1 flow of operation

Summary:

The LMS technique is an adaptive filtering algorithm used for audio denoising, iteratively adjusting a filter to minimize the difference between the filtered signal and the desired noise-free signal, but it has limitations in practical applications.

- **deep learning approach:**

We present an end-to-end deep learning approach to audio denoising. Our approach trains a fully convolutional de-noising network using a deep feature loss.

The study compares a pre-trained audio classification network to denoising techniques, comparing internal activation patterns induced by two signals. Experiments show the approach's advantages are most pronounced for the hardest, noisiest inputs.

Denoising systems used spectrogram-domain statistical signal processing methods before deep networks, but most still operate in the spectrogram domain. Loss function inspired by computer vision research. [4]

Method:

The study aims to find a denoising operator g that minimizes background signal noise in a noised audio signal x = s + n. Using a fully-convolutional network architecture, the output signal is synthesized sample by sample. The architecture uses 16 convolutional layers, dilated convolution, adaptive normalization, and a nonlinear activation function without bias.

Experiments showed lower quality output at lower SNRs due to improper processing of low-energy speech information. Deep feature loss was employed to compare features at different audio scales, creating a custom audio classification network inspired by VGG architecture. [3]

Conclusion:

The study presents an **end-to-end** speech denoising pipeline using a fully-convolutional network and deep feature loss network, achieving better performance without added complexity or expert knowledge. Experiments show the approach outperforms state-of-the-art baselines, especially in noisy conditions, and validates the combined use of convolutional context aggregation networks and feature losses.

- **Wavelet transform:**

Wavelet transform is a mathematical tool used for analyzing and representing signals, when a signal is broken down into constituents with distinct frequency bands, it becomes possible to analyze the high- and low-frequency components of the signal in greater depth. [6]

Types It is divided into two parts as follows:

1- Continuous wavelet transforms: it is a collection of fundamental functions that allows for simultaneous data analysis in the frequency and time domains.
This analysis is completed by the time-frequency range covered by a scalable fixed-length window and moving around the signal from start to end.

2- Discrete wavelet transform: the key difference between continuous wavelet transform (CWT) and DWT is that DWT involves the discrete sampling of the signal at different scales and positions. [7]

There are a lot of wavelets that can be used according to the requirements, as shown:

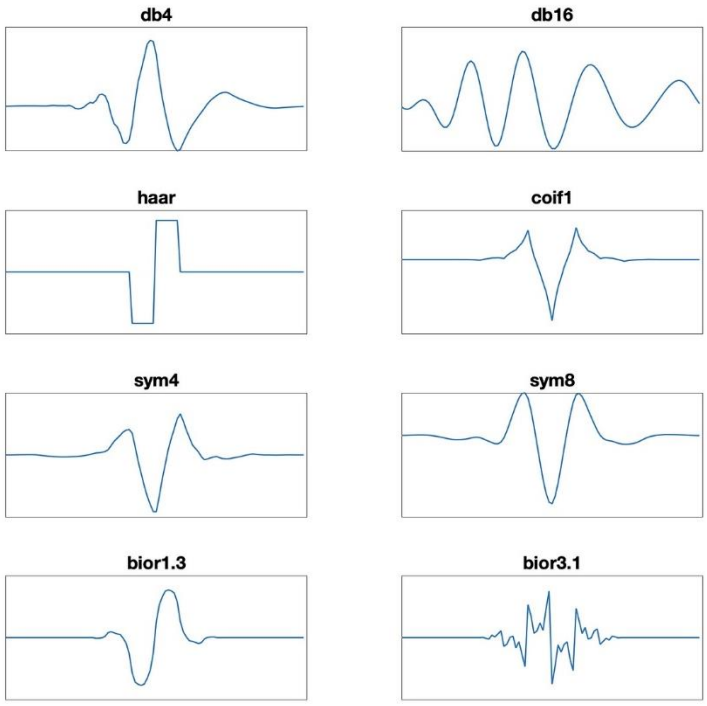


Figure 1 different shapes of wavelet transform

Summary:

The wavelet transform is an effective mathematical technique for signal analysis that offers a flexible and effective means of representing and analyzing signals that are localized in both time and frequency which helps us to do audio denoising more efficiently and accurately.

Methodology

Previously in literature review, we have seen what others have done to denoise the audio, and we choose the best three methods for audio denoising, and they are (FIR, IIR Filters and Wavelet transform)

1. Wavelet transform:

It is one of the most efficient methods for audio denoising, especially the Threshold algorithm (the potent technique to compress noise in digital signal).

It is divided into two parts as follows :

1.1. Continuous Wavelet Transform:

The continuous wavelet transform is a collection of fundamental functions that allows for simultaneous analysis of data in the frequency and time domains.

This analysis is completed by the time-frequency range covered by a scalable window of fixed length and moving around the signal from start to end by changing s (will be explained).

All wavelets are various of the mother wavelet as follows: eq [1]

$$\boldsymbol{\Psi(t,s) = \frac{1}{\sqrt{s}}\,\psi\left(\frac{t-\tau}{s}\right)} \qquad \textit{Equation 1}$$

Where $\boldsymbol{\psi\left(\frac{t-\tau}{s}\right)}$ is the mother wavelet,

And s reflects the scale or width of a basis function, τ is the translation that specifies its translated position on the time axis, and $(s)^{-\frac{1}{2}}$ is the normalized factor used to ensure energy across different scales remains the same. [8]

The continuous wavelet transform equation is: eq [2]

$$\boldsymbol{f(t,s) = \frac{1}{\sqrt{|s|}}\int_{-\infty}^{\infty} f(t)\,\psi^{*}\left(\frac{t-\tau}{s}\right)} \qquad \textit{Equation 2}$$

Wavelet coefficients are values that come from signal or data wavelet modifications. They depict the properties of the signal at various scales and positions.

1.2. Audio denoising with combining PDEs and CWT thresholding:

Combining PDEs equations with wavelet thresholding is one widespread approach. The basic approach is determined using the heat equation eq [3] and smooth thresholding in the wavelet domain.

$$\boldsymbol{\frac{\partial u}{\partial t} = \alpha \nabla^2 u} \qquad \textit{Equation 3} \qquad \textit{where } \alpha \textit{ is a constant and } \nabla^2 \textit{ is laplacian operator}$$

Decompose the audio signal $u(t)$ into different frequency components using a wavelet transform. Let $W(u)$ represent the wavelet coefficients, apply wavelet thresholding to the wavelet coefficient $W(u)$ within the PDE framework, soft thresholding is used (will be explained). eq [4]

$$\boldsymbol{S\lambda(x) = sign(x).max(|x| - \lambda, 0)} \quad \textit{Equation 4} \qquad \textit{where } \boldsymbol{\lambda} \textit{ is threshold value}$$

where $sign(x) = \begin{cases} -1 & if\ x < 0 \\ 0 & if\ x = 0 \\ 1 & if\ x > 0 \end{cases}$ sign(x) returns the sign of the input

and $\max(|x| - \lambda, 0)$ represents a mathematical operation that calculates the maximum value between |x|, λ and 0. [9]

We can use numerical methods like finite difference method to solve the **PDEs** with the modified wavelet coefficient, then reconstruct the denoised audio signal by applying **IWT** to the modified wavelet coefficient.

The combined equation for denoising process can be expressed as eq [5]

$$\boldsymbol{\frac{\partial u}{\partial t} = \alpha \nabla^2 u + S\lambda(W(u))} \quad \textit{Equation 5}$$

1.3. Thresholding:

Thresholding is used in audio denoising to separate desired signals from undesired noise. The fundamental principle is to establish a threshold level, below which any audio elements (amplitudes or frequencies) are regarded as noise and are either deleted or decreased.

In audio denoising, a variety of thresholding techniques are considered:

- Setting a threshold amplitude level and treating any signal components below it as noise is known as amplitude thresholding.
- Applying thresholds to certain frequency bands in order to minimize noise inside them is known as frequency thresholding.
- Time-Frequency Thresholding: Applying adaptive thresholds to various time-frequency components by using time-frequency analysis techniques like wavelet transform.

There is to methods used in wavelet denoising, **soft thresholding** decreases coefficients, it tends to diminish them more, frequently or amplitude to zero, whereas **hard thresholding** sets coefficients below a given threshold to zero. In contrast to hard thresholding, soft thresholding preserves some small magnitude features. [10]

1.4. Discrete Wavelet Transform:

If s and τ are based on being discrete and based on the power of 2, the analysis will be more accurate.

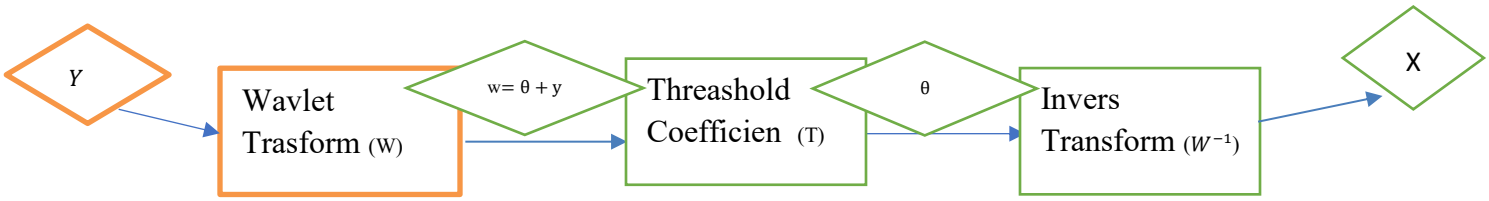
The discrete wavelet transform equation is: eq [6]

$$D[a,b] = \frac{1}{\sqrt{b}} \sum_{m=0}^{p-1} f[tm]\, \psi\, [\frac{tm-a}{b}] \quad \textit{Equation 6}$$ **where** a represent τ and **b** represent **s**

It is one of the most efficient methods for audio denoising, especially the Threshold algorithm (the potent technique to compress noise in digital signal).

The technique comprises applying the **DWT** to the first information, thresholding the definite wavelet coefficients, and converse transforming using the inverse wavelet transforms- **IDWT**.

The intensity of the noise (σ) implemented in the signal is a key factor in determining compress levels and the threshold value. [11]



Flow chart 2 flow of operation for wavelet trans

The steps to remove noise from audio signals are as follows:

1- Read the original signal s(i).
2- Add the Gaussian noise to the original signal to form a noised signal, we do that to test the efficiency of noise on the original signal. eq [7]

$$r(i) = s(i) + \sigma\varepsilon(i), \quad i = 0, 1, 2 \ldots n. \quad \textit{Equation 7}$$

where n is the length of the signal, r(i) is the noised signal, and σ is the noise intensity.

3- We calculate the ratio of pure signal to noise eq [8] and the root mean square of the original signal. eq [9]

$$SNR(dB) = 10\, log10 \left[\frac{\sum_{i=1}^{n} Xi^2}{\sum_{i=1}^{n} (ri-si)^2} \right] \quad \textit{Equation 8}$$

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} (ri-si)^2}{n}} \quad \textit{Equation 9}$$

Where s is the original signal, r is the noisy signal and n is the magnitude of the signal.

4. For the observed **SNR/RMSE** of the received signal, compute the threshold value for wavelet thresholding algorithm (will be explained). The level-dependent threshold value is computed using the modified form of the universal threshold. eq [10]

$$\boldsymbol{Tnew} = \overset{\sigma}{\sqrt{\boldsymbol{2log}\frac{\boldsymbol{N}}{\boldsymbol{2^I}}}} \quad \textit{Equation 10}$$

Where σ is the noise intensity, N represents the number of samples, and I is the decomposition stage. [12]

5- Decompose the received signal into wavelet coefficients using the corresponding analysis filter banks.

6- Selecting the threshold value is more important in wavelet threshold denoising method, if its value is too small or too large, the signal cannot be accurately measured.

To apply **threshold value** to the decomposed wavelet coefficients there is two types:

$$Soft\ threshold: \{\boldsymbol{r} = \boldsymbol{sign(s)}\ (|\boldsymbol{s}| - \boldsymbol{T}) \quad \textit{Equation 11}$$

$$Hard\ threshold: \{\boldsymbol{r} = \boldsymbol{s}, \boldsymbol{if}\ |\boldsymbol{s}| > \boldsymbol{T}, \boldsymbol{r} = \boldsymbol{0}, \boldsymbol{if}\ |\boldsymbol{s}| < \boldsymbol{T}\} \quad \textit{Equation 12}$$

Where s will be the input signal, r will be the signal after applying threshold signal and T is the threshold point.

7- Reconstruct the signal and calculate SNR and RMSE of the denoised signal using equations (3) and (4) and compare the values with the values of previous step 2 for grading the performance.
If the SNR is high or the RMSE is low, they will decide the realization of the denoising method. [13]

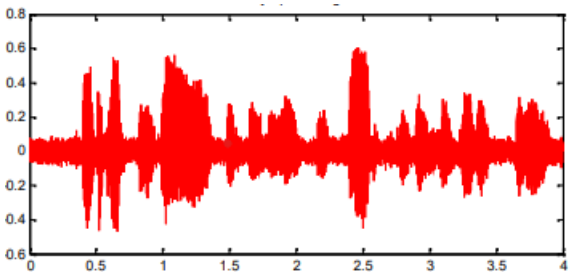


Figure 2: noisy signal

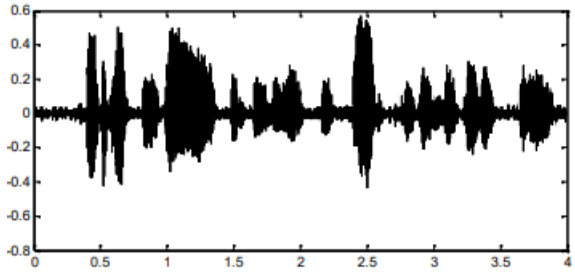


Figure 3: denoised signal

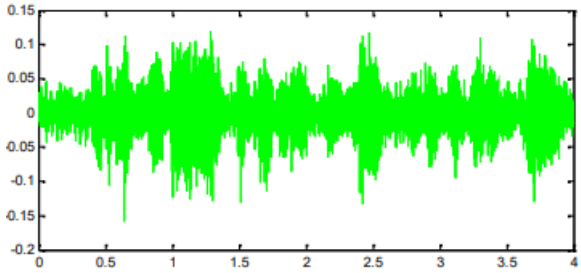
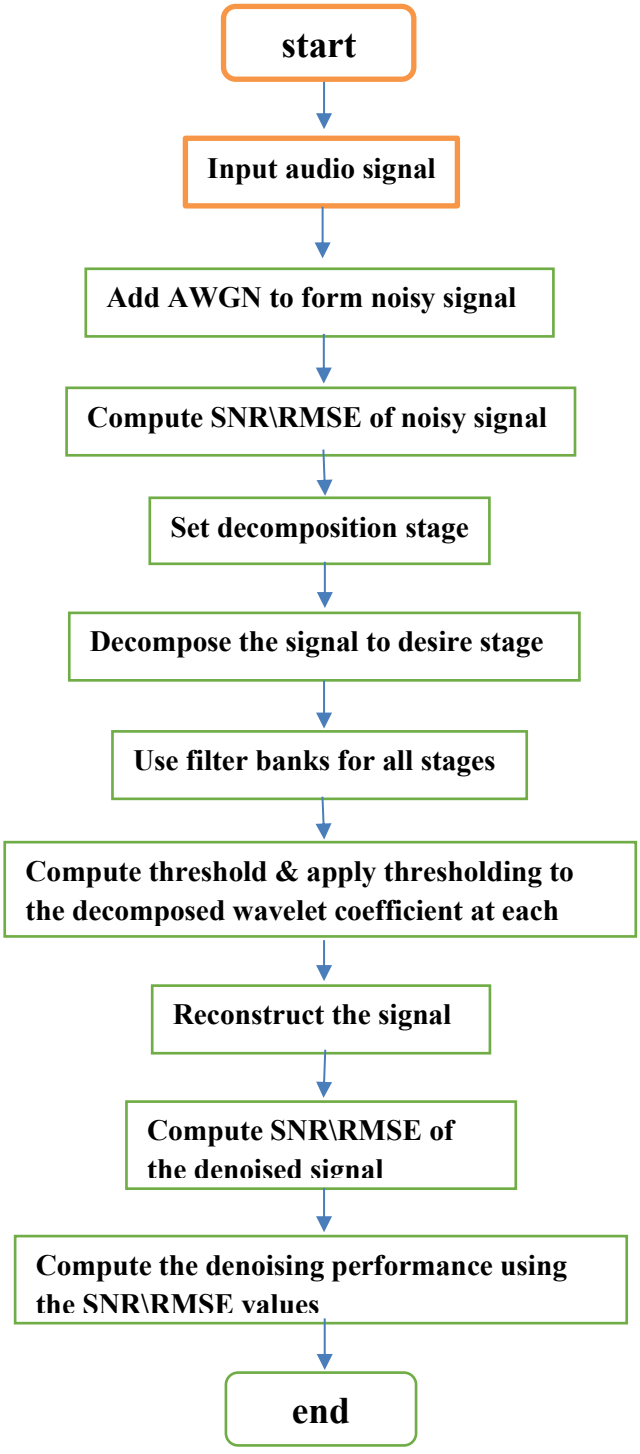


Figure 4: residual signal

Flow chart of the steps to remove noise from audio signals



Flow chart 3 steps to remove noise

2. FIR and IIR Filters:

In order to implement any low pass, high pass and band stop or pass filter, we can create an electric circuit for them, but if we want to implement them DIGITALLY, then we need to put them in mathematical equations by relating their output signal to input signal using convolution theorem without the presence of complex numbers. While doing the previous we get **Finite Impulse Response (FIR)** filter or **Infinite Impulse Response (IIR)** filter which each of them can be a low pass, high pass and band stop or pass.

As an **example,** we will apply on the LOW PASS filter:

- The graphical graph of **Ideal** LOW PASS FIR filter will be as in Fig. (1):

By applying the Inverse discrete Fourier or wavelet transform (IDCT or IDWT) we get the filter's response in time domain as in Fig (5). By sampling (using impulse function) and windowing method using rectangular window (we can also use Hamming, Hann…etc. windows instead Fig (6) because we have a limited resource digital system, we get the truncated impulse response of the filter as in Fig (7).

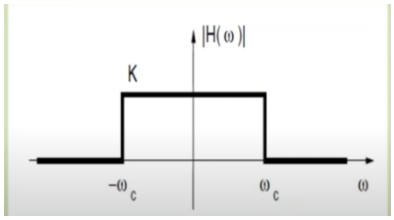


Figure 5 ideal LPF F domain

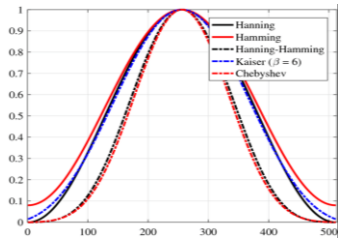


Figure 6 Ideal LPF (t domain)

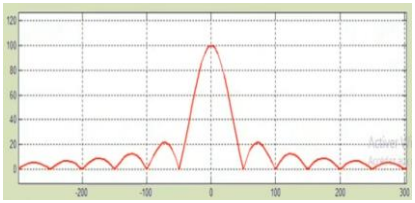


Figure 7 windowing functions

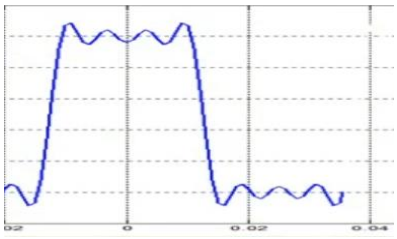


Figure 8 Truncated impulse response (t domain)

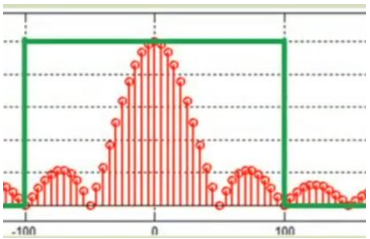


Figure 9 Truncated impulse response (f domain*)*

- In order to put equations, we will return to frequency domain through DFT or DWT which will give us Fig (8) which is considered an approximation for Fig (5).

By convolving of the input signal and impulse response for discrete time signal we get the output signal which will be in frequency domain **as following:**

$$\boldsymbol{y(n) = \sum_{m=0}^{N-1} h(m).x(n-m)}$$ Equation 13 **where** N is the samples number

Let h(m) be the coefficient a_m, therefore the eq. (14) **will be:**

$$y(n) = \sum_{m=0}^{N-1} a_m . x(n-m) \quad \textit{Equation 14}$$

By applying the Z transform (ZT):

$$y(z) = \sum_{i=0}^{N-1} a_i . x(z) . z^{-i} \quad \textit{Equation 15}$$

Therefore, the Transfer function in Z domain will be:

$$H(z) = \sum_{i=0}^{N-1} a_i . z^{-i} \quad \textit{Equation 16}$$

We derived the equations above generally, so that they are valid for any FIR filter whether it's low or high or band pass or stop filters.

- Another idea is to put the transfer function in the form:

$H(z) = \frac{N(z)}{D(z)}$, because finding the $\mathbf{a_i}$ **coefficients** in this form is easier. This will lead to the following equation:

$$H(z) = \frac{y(z)}{x(z)} = \frac{\sum_{n=0}^{N-1} a_n . z^{-n}}{\sum_{m=0}^{N-1} b_m . z^{-m}} \quad \textit{Equation 17}$$

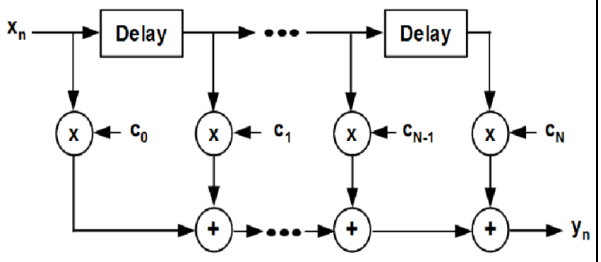


Figure 10

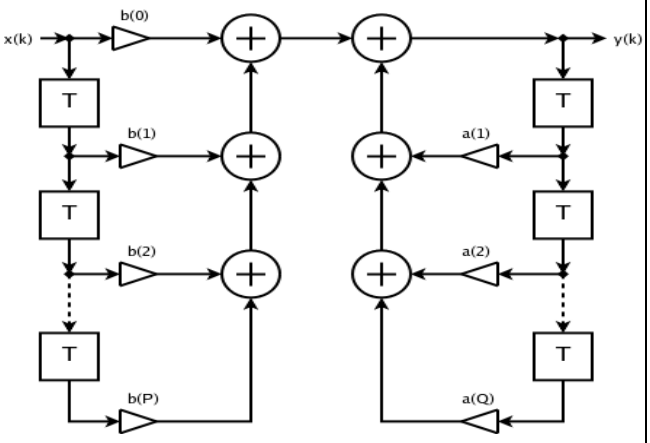


Figure 11 the operation

By performing inverse Z transform:

$$y(n) = \sum_{k=0}^{M} a_k . x(n-k) - \sum_{k=0}^{N} b_k . y(n-k) \quad \textit{Equation 18}$$

2.1. Finite & Infinite Impulse Response (FIR) filter:

FIR & IIR filters are digital filters, FIR whose impulse response is of finite duration, because it settles to zero in finite time. Alike FIR , IIR Has impulse response of infinite duration

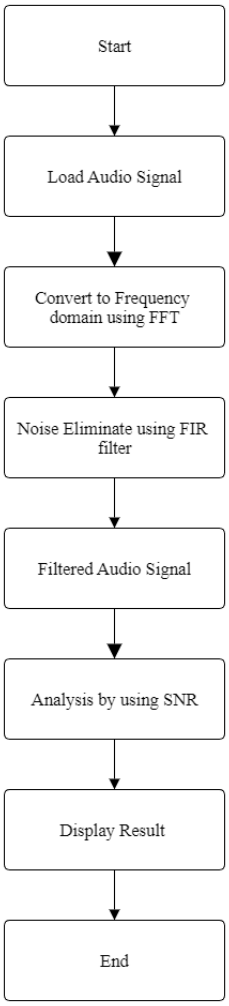
There are some advantages of using FIR filter such as:

- It requires no feedback. This means that any rounding errors are not compounded by summed iterations.
- It is inherently stable

Advantages of using IIR filter such as:

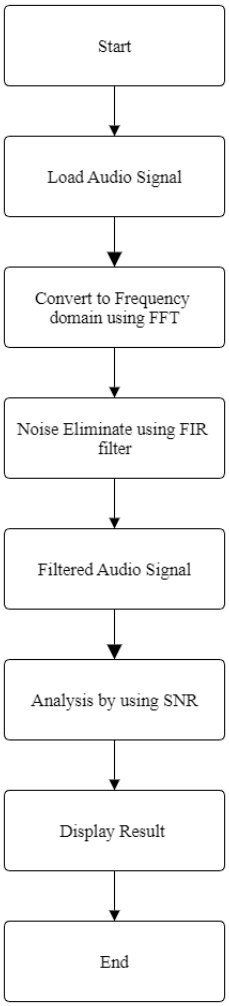
- IIR filters usually require fewer coefficients to execute similar filtering operations.
- work faster, and require less memory space

Audio denoising using FIR filter:



Flow chart 4 FIR filter

Audio denoising using IIR filter:



Flow chart 5 IIR filter

2.2. Fast Fourier transform (FFT)

Fast Fourier Transform (FFT) is widely used for many applications. A Fourier transform converts time to frequency and vice versa. The FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum. Let x0, ..., xN-1 be complex numbers. The DFT is defined by the formula as follows:

$$\boldsymbol{X(k) = \sum_{n=0}^{N-1} x(n) W_N{}^{nk}} \qquad \textit{Equation 19}$$

Where X(k) is the sequence of N complex-valued numbers, k = 0, 1..., N-1 and $\boldsymbol{W_N = e^{-J2\pi/N}}$

Direct computation of the **DFT** is basically inefficient primarily because it does not exploit the symmetry and periodicity properties of the phase factor $\mathbf{W_N}$

Actually, direct computation of DFT requires the order of N2 operations where N is the transform size. The **FFT** opens a new area in digital signal processing by reducing the order of complexity of **DFT** from **N2** to $\boldsymbol{Nlog2N}$. The advantages of this technique are its speed because **FFT** spectrum analyses all frequency components at the same time.

2.3. Denoising using FIR & IIR Filters:

The basic idea of FIR filter is to convolve the input signal with the impulse response of the filter. The impulse response of the filter is the output of the filter when the input is an impulse function. FIR Filters have finite Impulse response meaning that they have a finite duration because it settles to zero in finite time. The impulse response of an Nth-order discrete-time **FIR** filter lasts for **N + 1** samples, and then settles to zero. The phase of a FIR filter is linear.

$$\boldsymbol{h(k) = \sum_{k=0}^{N}(b_k \delta(n-k)) = b_k} \qquad \textit{Equation 20}$$

Where:

h(k) is the impulse response can be calculated if the input $\boldsymbol{signal\ x(n) = \delta(n)}$ in the above relation, where **δ(n)** is the impulse. The impulse response for an FIR filter then becomes the set of coefficients $\boldsymbol{b_k}$.

For FIR:

$$\boldsymbol{y(n) = b_0 x(n) + b_1 x(n-1) + .. + b_N x(n) = \sum_{k=0}^{N} b_k x(n-k)} \qquad \textit{Equation 21}$$

For IIR:

$$\boldsymbol{y(n) = \sum_{k=0}^{M} a_k . x(n-k) - \sum_{k=0}^{N} b_k . y(n-k)} \qquad \textit{Equation 22}$$

Where:

x(n) is the input signal, **y(n)** is the output signal, $\mathbf{b_k}$,**ak** are the filter coefficients, also known as tap weights, that make up the impulse response, and N is the filter order; an Nth order filter has **(N+1)** terms on the right-hand side.

The **x(n-i)** in these terms is commonly referred to as taps, based on the structure of a tapped delay line that in many implementations provides the delayed inputs to the multiplication operations.

2.4. Signal-To-Noise Ratio (SNR):

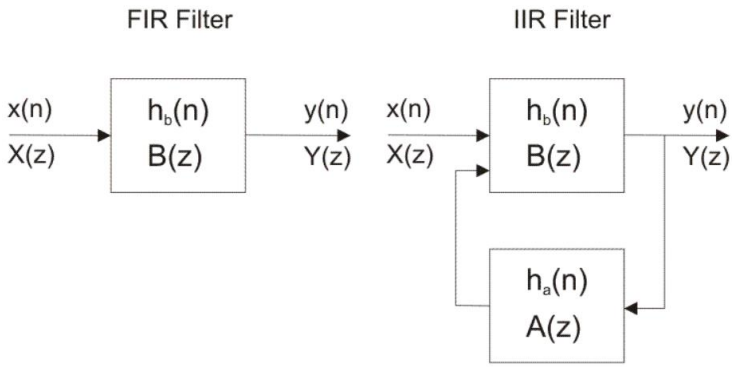
SNR is a measure used to quantify how much a signal has been corrupted by noise. The higher the ratio, the less obtrusive the background noise is. A higher SNR usually correlates with an observable improvement in quality and data accuracy. SNR is defined as the power ratio between a signal (meaningful information) and the background noise (unwanted signal).

$$\boldsymbol{SNR = 20log10\ \frac{A(signal)}{A(noise)}} \qquad \textit{Equation 23}$$

Where:

A is root mean square (**RMS**) amplitude. The signal and noise powers must be measured at the same or equivalent points in a system, and within the same system bandwidth.

Finally, we can summarize difference between FIR & IIR as follows:



Flow chart 4

Hardware Implementation:

In our project, we are going to implement our system with Arduino-UNO-R3. The system will have two outputs: The noised sound file and the denoised sound file.

To understand how it works we are going to illustrate each component, as shown in Fig [12]

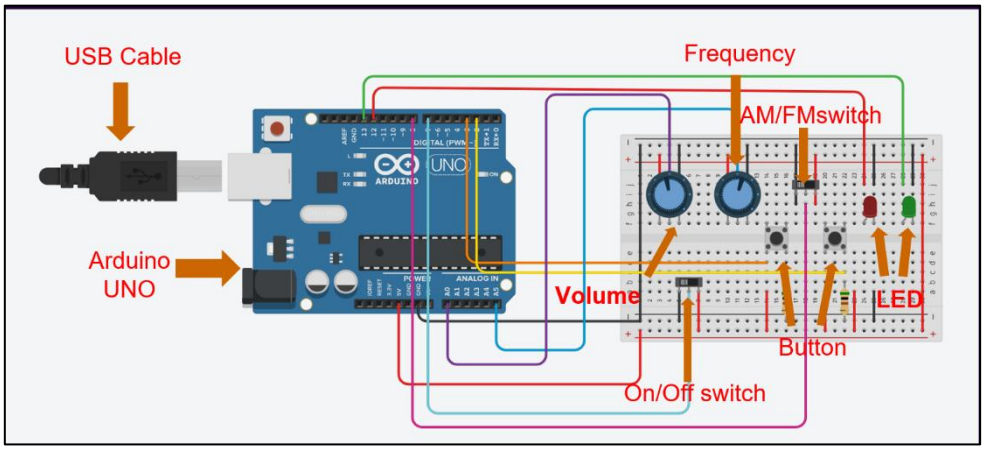


Figure 12 Arduino circuit design

1. Components:

1.1. USB Cable:

The USB cable is USB Cable Standard Type A - Type B Mini for Arduino Nano, It is used as a power source and communication channel between the Arduino and the CPU.

1.2. Arduino UNO:

The Arduino UNO R3 represents the system's Control Unit as it contains a microcontroller. so the user can easily program it using the C++ language, The Arduino Contains pins that are programmable either as an input or output.

1.3. LED:

LEDs (Light-Emitting-Diodes) are used to indicate the status of a circuit or to visualize data. We will use them as flags to show which sound file is playing.

1.4. Push Button:

The push button is used as a switch; we will use them as selectors to select which sound file will be played.

1.5. Potentiometer:

It is a variable resistor. So, we use it for analog input such as volume and frequency.

1.6. Slide Switch:

It is a circuit key. So, we use it for switching between modes: FM-AM and On-Off.

1. System Description

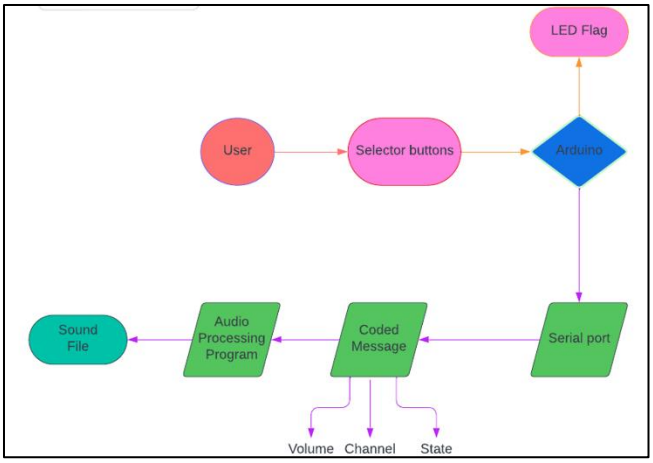


Figure 13 System Description Diagram

After Understanding the system components, we will discuss how the system works, as shown in Fig [13]

1.1. User Selection:

First of all, the user chooses what audio file to play whether it is the original file or the denoised one. The choice is made by pushing the button a gentle push. And can modify the frequency and volume.

1.2. Arduino:

The Arduino receives the signal from the buttons. Then it makes sure that it is a gentle push not a long one in order to send to the serial port the coded message that contains:

- Volume: from 0-200%
- Chanel: which sound file it will process
- State: steady state (0), receive (1) and filter (2)

1.3. Audio Processing program:

We are using Matlab. Its purpose is to receive from the serial port the coded message and decode it in order to process what the user demanded. [14]

2. Software Implementation:

In our project, we made a software application using MATLAB GUI as shown in Fig [14].

Figure 14 Software Application

In our App you can encrypt, modify play back speed, denoise and save audio files.

We made those features:

- Selecting the audio file.
- Display the file information.
- Play, Pause, Resume and Stop the file.
- Encrypt the file by adding noise that you control.
- Play back speed control.
- Volume control.
- Denoising by choosing which filter to use.
- Save your audio file masterpiece [15]

This application processes the audio files to provide it to the hardware device.

Testing and comparison:

Introduction:

At the beginning, we will present the experimental work, results, and analysis for our selected three techniques. First, we will introduce some definitions of parameters that we will check for each method.

We have three main parameters to measure: MSE, PSNR and SNR.

MSE (Mean square error):

MSE provides the distance between a regression line and a set of points. It accomplishes this by squaring the distances referred to as the "errors" between the points and the regression line. Squaring is required to eliminate any unfavorable indicators. Additionally, it accords greater weight to bigger variations. Since we're calculating the average of a group of errors, it's also known as the mean squared error. The lower the MSE, the better the results.

Here's the equation for calculating MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \qquad \textit{Equation 22}$$

- n: is the number of data points.
- $Y_{i:\ is}$ the actual observed value at data point i.
- $Y^e_{i:\ is}$ the predicted value at data point i.

PSNR (peak signal-to-noise ratio):

PSNR is the ratio between the greatest signal power and the corrupting noise power that affects the fidelity of its representation.

Here's the equation for calculating PSNR:

$$PSNR = \mathbf{10}.\boldsymbol{log}_{\mathbf{10}}\left(\frac{\boldsymbol{MAX}^2}{\boldsymbol{MSE}}\right) \qquad \textit{Equation 23}$$

- MAX: represents the maximum possible amplitude of the audio signal.
- MSE: mean square error illustrated before.

SNR (signal-to-noise ratio):

SNR is a measure used to compare the level of a desired signal to the level of background noise.

Here's the equation for calculating SNR:

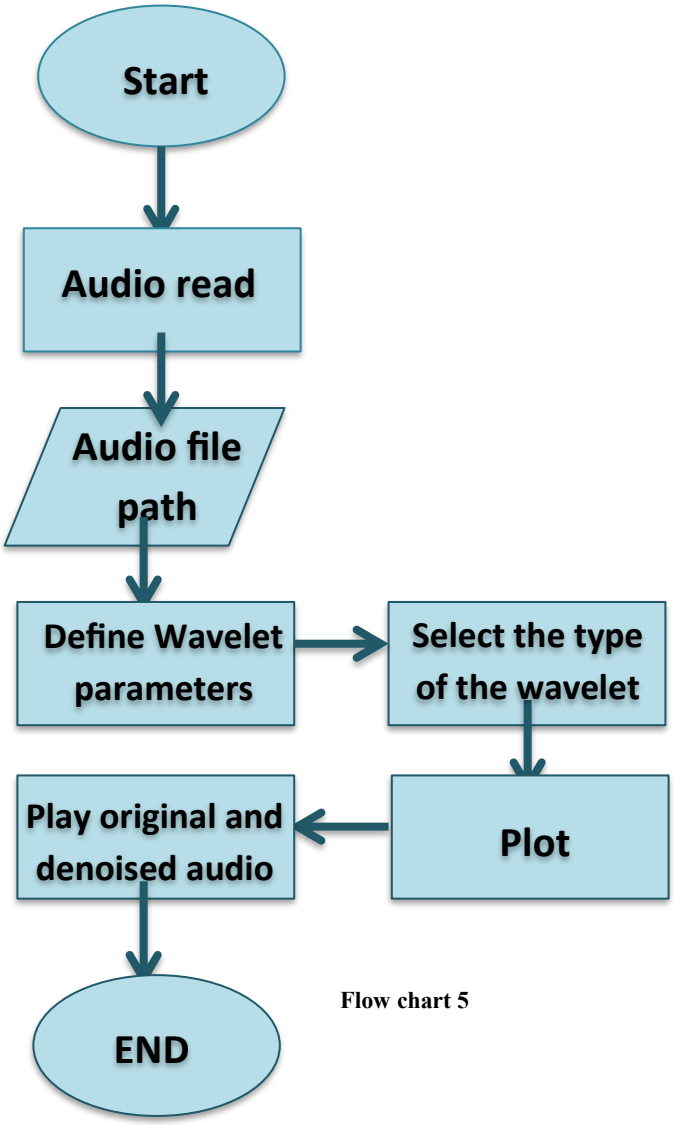
$$SNR = \mathbf{10}.\boldsymbol{log}_{\mathbf{10}}\left(\frac{\boldsymbol{SIGNAL\ power}}{\boldsymbol{NOISE\ power}}\right) \qquad \textit{Equation 24}$$

Signal power: is the power of the desired audio signal.

Noise power: is the power of the unwanted noise.

Results and Analysis of Wavelet:

1.1. Flow chart:



Flow chart 5

Explain the code:

- **Load audio file:** Audio read is used to read the path of the audio file and stored in the variable
- **Check if the Audio is Stereo:** If the audio is stereo (two channels), it takes the average of the channels to convert it to a mono signal.
- **Wavelet Denoising Parameters:** Wavelet denoising parameters are defined. The chosen soft thresholding method is 0.1, the wavelet utilized is 'db4' (a Daubechies wavelet with 4 coefficients), and the decomposition level is 5.
- **Thresholding:** Depending on the type of threshold selected, either soft or hard thresholding is applied to the wavelet coefficients.

- **Reconstruction of Denoised Signal:** The thresholded wavelet coefficients are used to recreate the denoised signal.
- **Plotting:** Two subplots are created to display the original noisy signal and the denoised signal.
- **Play Audio:** For the original and denoised signals, two audioplayer objects (K and M) are made, respectively. There are intervals to allow for listening in between each time the signals are played.

1.2. Results:

First of all, we set our parameters to be:

- Wavelet type = 'db4'
- Decomposition level = 5
- Threshold value = 0.1

And we are going to compare between 3 signals with 3 different frequencies (Low,Normal,High) by the most important parameter : Threshold type either soft or hard

1.2.1. EXPERIMENT 1 (Normal Frequency):

Comparing Parameter	Threshold Type: SOFT	Threshold Type: HARD
MSE	**0.003182 dB**	**0.001675 dB**
PSNR	**115.28 dB**	**118.07 dB**

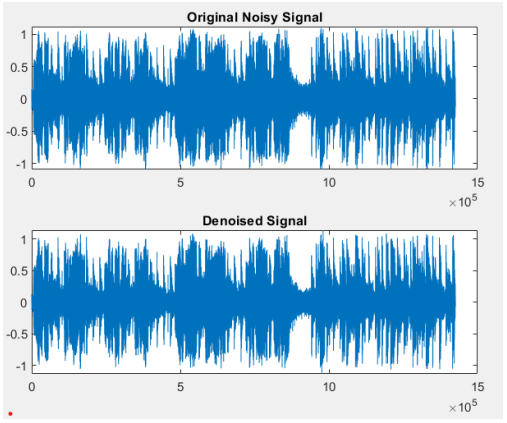


***Figure15**: Normal frequency SOFT*

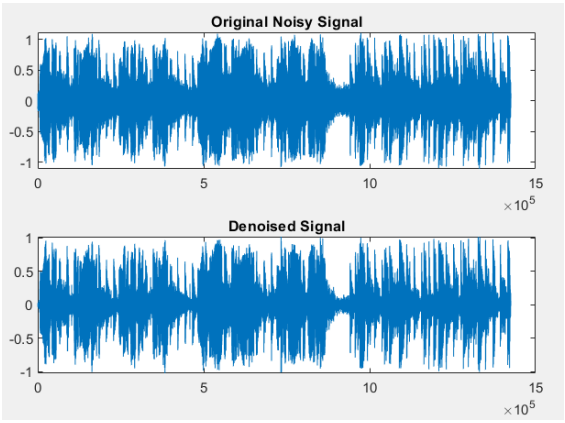


***Figure16**: Normal frequency HARD*

1.2.2. EXPERIMENT 2 (High Frequency):

Comparing Parameter	Threshold Type: SOFT	Threshold Type: HARD
MSE	0.003171 dB	0.001678 dB
PSNR	115.30 dB	118.06 dB

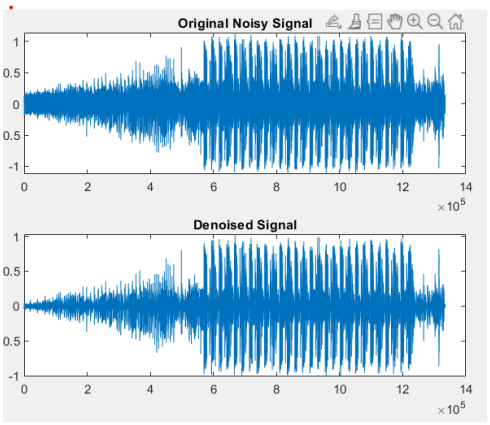


Figure17: High frequency **SOFT**

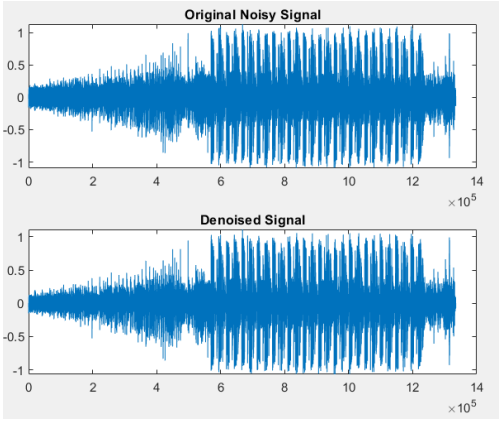


Figure18: *High frequency* ***HARD***

1.2.3. EXPERIMENT 3 (Low Frequency):

Comparing Parameter	Threshold Type : SOFT	Threshold Type : HARD
MSE	0.000340 dB	0.000181 dB
PSNR	125.00 dB	127.73 dB

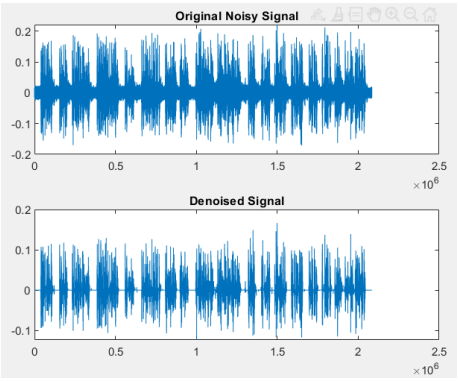


Figure19*: Low frequency* ***SOFT***

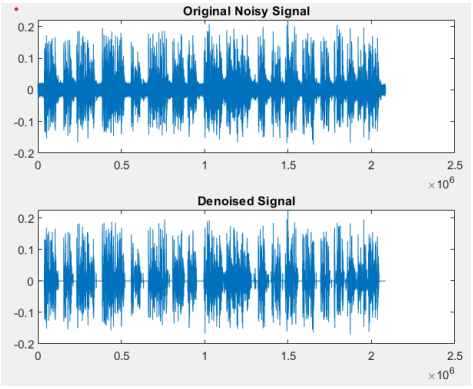
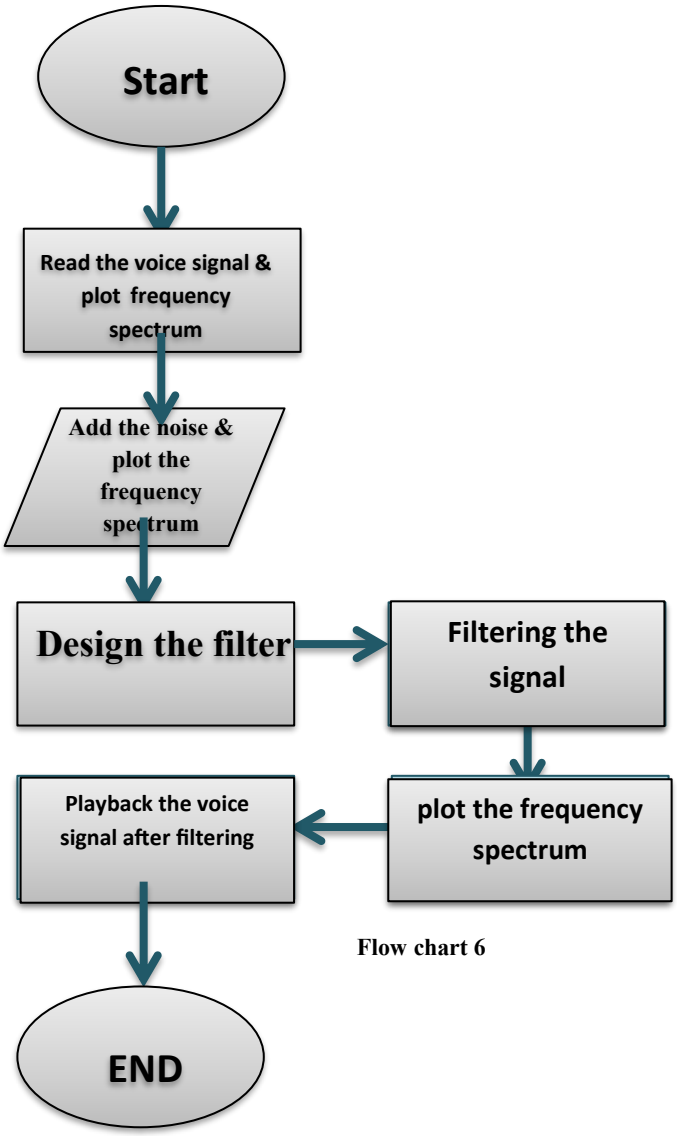


Figure20: *Low frequency* ***HARD***

2. IIR FILTER:

We will do an experiment by using an IIR filter to denoise the audio signal and test the result to know its accuracy and efficiency.

2.1. Flow chart:



Flow chart 6

2.1 Explain the code

- We add the original noise to the program to read and calculate its **FFT**.
- Plot the original signal in the frequency spectrum as**single-sided** and play the audio.
- Generate the required noise by choosing the lower and upper-frequency bounds and the noise magnitude, then add it to the original audio to produce the "**Noisy audio**".
- Calculate the **FFT** for the "**Noisy audio**" and use it to plot the **single sided** spectrum of it.
- Design the filter which is suitable to remove the added noise signal: We used the MATLAB "filter Designer" tool to generate our required filters. To design a typical IIR using it, we follow the following steps:
- Determine the filter type: [Lowpass, High pass, …. etc.].
- Determine the filter designing method: [IIR or FIR].

- Determine the filter order or choose the “Minimum order” option.
- Determine the filters pass frequency and stop frequency.
- Click” Design filter”, then export it.

We were able to use a MATLAB code also for designing low pass filters only without needing the “filter Designer” tool’ (This code is present in the appendix).

- Play the filtered signal to show the difference between itand the original
- Calculate the SNR, MSE, PSNR.

Results:

2.1.1 Experiment_1: Low_pass_butterworth_filter

We choose to pass the frequency up to 10000 HZ and stop the frequency at 11000 HZ and choose the minimum order of the filter.
The noise amplitude is .005 and lower frequency is 11000 and the upper is 12000 HZ.

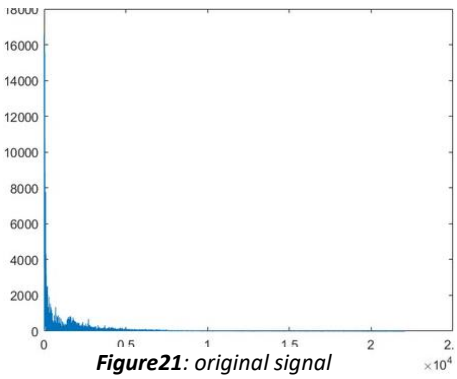


***Figure21**: original signal*

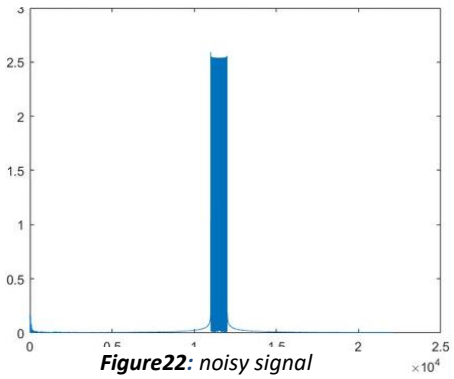


***Figure22**: noisy signal*

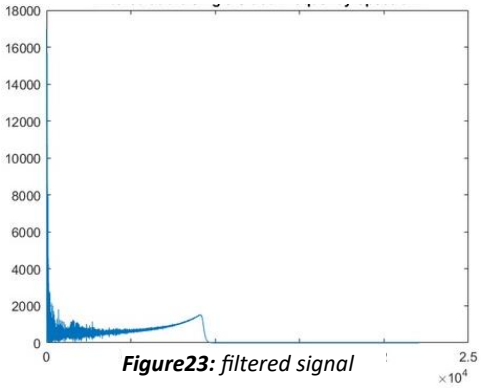


***Figure23:** filtered signal*

2.1.2 Experiment_2: Low_Pass_Cheybshev_Type_II

We choose to pass the frequency up to 9000 HZ and stop the frequency at 10000 HZ and choose the order of the filter is 20.

The noise amplitude is .005 and lower frequency is 12000 and the upper is 12500 HZ.

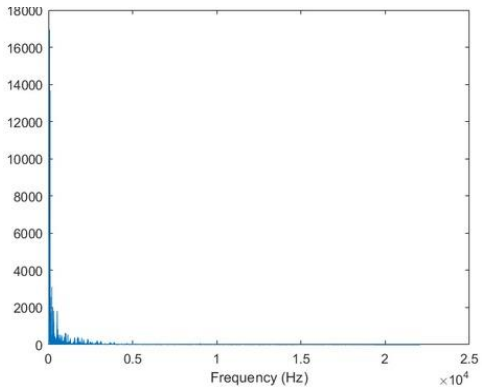


***Figure24:** original signal*

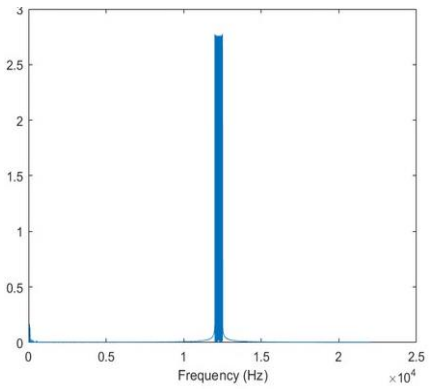


Figure25: noisy signal

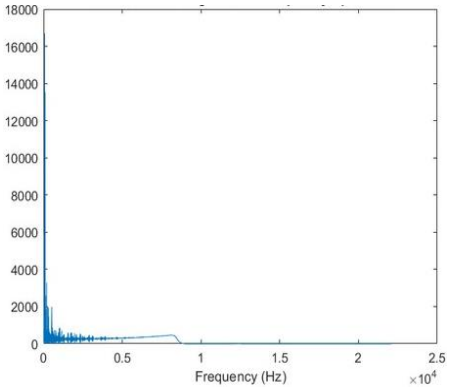


***Figure26**: filtered signal*

2.1.3 **Experiment_3:** Low_Pass_Elliptic

We choose to pass the frequency up to 2400 HZ and stop the frequency at 2500 HZ and choose the order of the filter is 13.
The noise amplitude is .005 and lower frequency is 2500 and the upper is 3000 HZ.

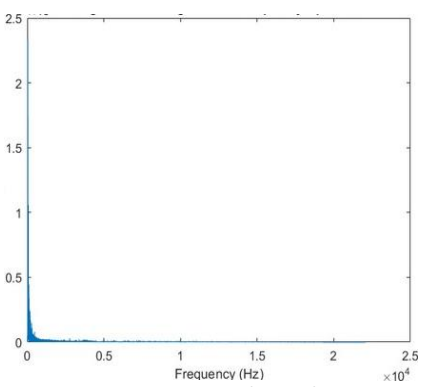


Figure 27: *original signal*

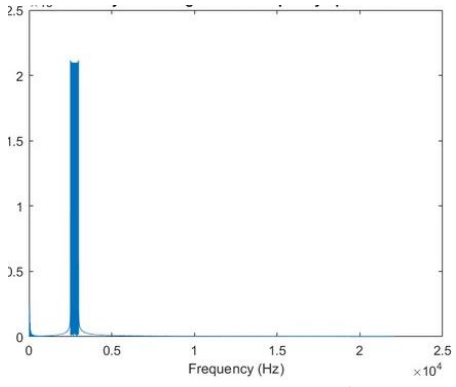


Figure28*: noisy signal*

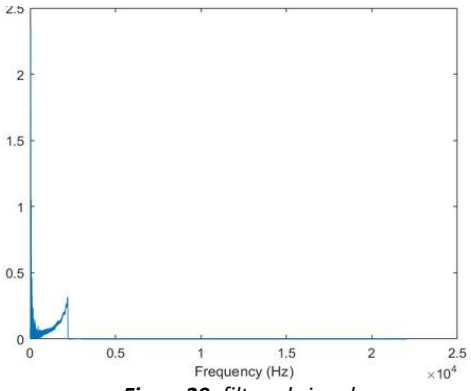


Figure29: *filtered signal*

Exp 2: Low Pass Cheybshev Type II

gives better audio denoising results as the value of **PSNR** is high and the value of **MSE** is low.

2.2 Results

We calculate the values of **SNR**, **MSE**, and **PSNR** to know the results of the experiments and the better one of them as shown in the following table:

EXP	SNR	MSE1	MSE2	PSNR
1	-5.36 dB	0.526733	0.488028	2.940 dB
2	-3.82 dB	0.073803	0.057031	9.761 dB
3	-6.71 dB	0.299896	0.754180	1.935 dB

3. FIR Filter

3.1. Flow Chart:

1. Select an audio signal.

2. Read the selected audio.

3. Plot the waveform of the original audio.

4. Design FIR filter:

Design (FIR) filters with Kaiser and Hamming windows , respectively. The filter order and cutoff frequency can be adjusted as needed.

5. Apply Kaiser and Hamming filters to the original audio:

Apply Kaiser and Hamming filters to the input-audio signal and return the filtered audio signals.

6. Plot Kaiser and Hamming filtered audio waveforms.

7. Calculate the parameters :

Calculate parameters SNR, MSE, and PSNR for :

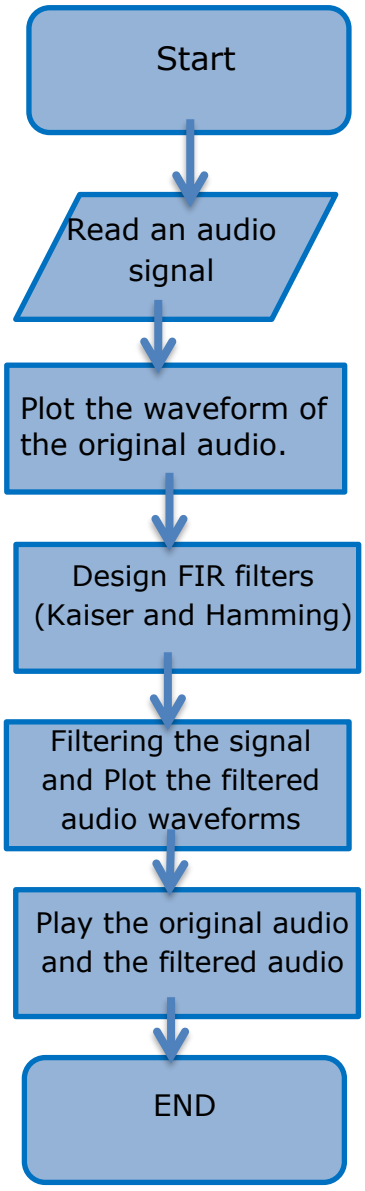
- Kaiser filtered audio.
- Hamming filtered audio.

We calculate these parameters to determine the quality of the filtered signal and accuracy of filer.

8. Plot the results.

9. Play Audio:

Play original audio, Kaiser filtered audio and hamming filtered audio.



Flow chart 7

3.2. Explain the code:

we choose to test the filter using two types of windowing: **Kaiser and Hamming.** The FIR filter order is set to 100. This means that the FIR filter will have 101 taps or coefficients. The FIR filter's cutoff frequency is set to 2000 Hz normalized to the sampling frequency. Normalized frequency is a value between 0 and 1, where 1 corresponds to the Nyquist frequency (half of the sampling frequency). In this case, 2000/Fs places the cutoff at 2000 Hz. Then we will calculate

MSE and PSNR values for each method to be able to observe which is better. In this experiment our audio signal is stereo (it has two channels, typically left and right) so the MSE is calculated independently for each channel, providing a measure of the denoising performance for each channel separately.

3.3. Results:

3.3.1. Low Frequency Signal

Window	SNR	MSE1	MSE2	PSNR
Kaiser	**-3.48 dB**	**0.146296**	**0.196021**	**7.575331 dB**
Hamming	**-3.35 dB**	**0.141983**	**0.190274**	**7.704783 dB**

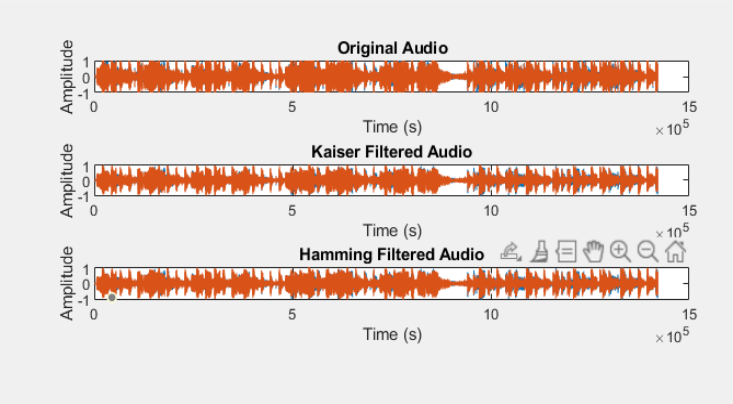


Figure 30 *LOW Freq.*

3.1.1. High Frequency Signal

Window	SNR	MSE1	MSE2	PSNR
Kaiser	**2.07 dB**	**0.042481**	**0.042482**	**13.718042 dB**
Hamming	**2.13 dB**	**0.041907**	**0.041924**	**13.776244 dB**

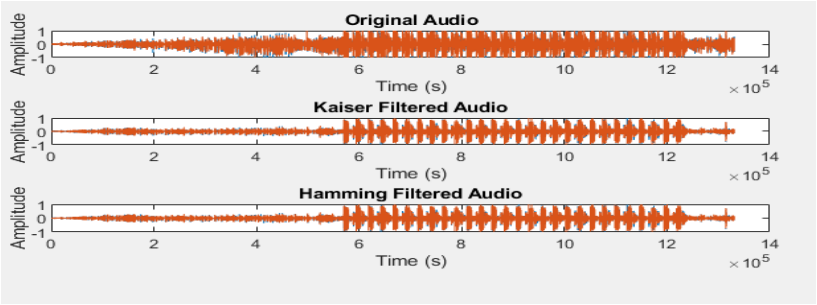


Figure 31 *HIGH Freq.*

3.3.2. Normal Frequency Signal

Window	SNR	MSE1	MSE2	PSNR
Kaiser	**-3.08 dB**	**0.001620**	**0.001620**	**14.762365 dB**
Hamming	**-2.96 dB**	**0.001575**	**0.001575**	**14.884338 dB**

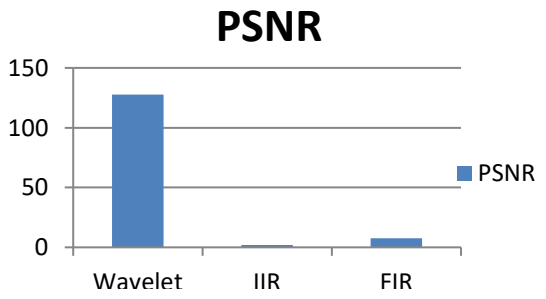
Figure 32 *NORMAL Freq.*

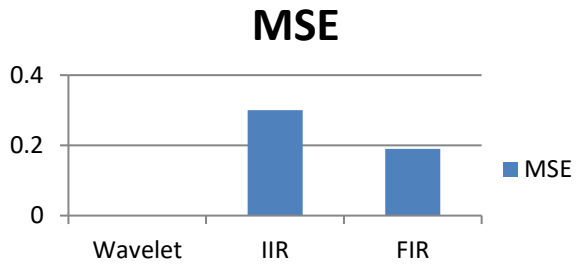
3.4. Conclusion:

By observing the table previous we find that using the filter with **the Hamming window** shows less error and shows a better denoising performance so it is recommended to use the filter with the hamming window.

4. Summary

	Wavelet		IIR		FIR	
Frequency	**MSE**	**PSNR**	**MSE**	**PSNR**	**MSE**	**PSNR**
Low	**0.000181 dB**	**127.73 dB**	**0.299896 dB**	**1.935 dB**	**0.190274 dB**	**7.704783 dB**
Normal	**0.001675 dB**	**118.07 dB**	**0.057031 dB**	**9.761 dB**	**0.001575 dB**	**14.884338 dB**
High	**0.001678 dB**	**118.06 dB**	**0.488028 dB**	**2.940 dB**	**0.041907 dB**	**13.776244 dB**





As we see **MSE** for **Wavelet** is the lowest and **PSNR** is the biggest so wavelet technique is the best way for audio denoising.

Conclusion:

After doing the experiment with the data set and analyzing the results in *table 1*.
We concluded that:

	Wavelet		IIR		FIR	
Frequency (k Hz)	MSE	PSNR	MSE	PSNR	MSE	PSNR
Low (1-30)	0.000181 dB	127.73 dB	0.299896 dB	1.935 dB	0.190274 dB	7.704783 dB
Normal (40-60)	0.001675 dB	118.07 dB	0.057031 dB	9.761 dB	0.001575 dB	14.884338 dB
High (60-100)	0.001678 dB	118.06 dB	0.488028 dB	2.940 dB	0.041907 dB	13.776244 dB

Table 1 Experiment Results

- The Wavelet method is obviously the best method.
- The difference between FIR and IIR is not as much, but the FIR is better.
- Highest performance is achieved at normal frequency that ranges from (40-60) kHz.

So, we recommend to use the **Wavelet** frequency with normal-frequency audio files to get the best performance.

Future Work:

Hardware:

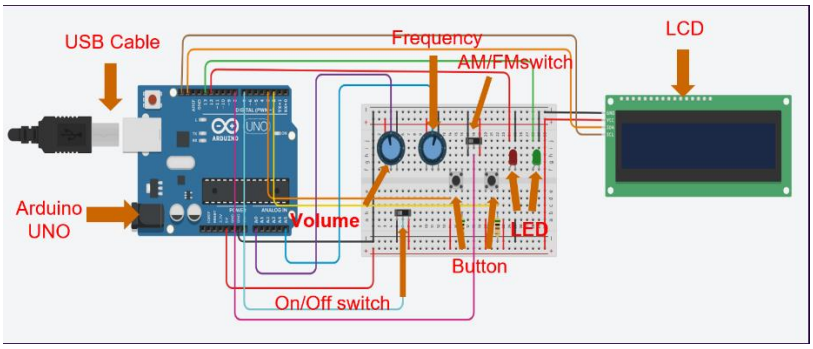


Figure 32 Future Hardware Device

As for hardware, we can improve our device by adding LCD display to show which mode we are , the frequency and the volume[16]

References:

[1] Signal-noise identification and targeted denoising for the measured mt ... (n.d.). https://www.researchgate.net/figure/Signal-noise-identification-and-targeted-denoising-for-the-measured-MT-data-a -Square_fig3_350337642

[2] **Aiyetigbo, M. D., Ravichandran, D., Chalhoub, R., Kalivas, P., & Li, N**. (2023, July 1). *Unsupervised coordinate-based video denoising*. arXiv.org. https://arxiv.org/abs/2307.00179

[3] **Kumar, Nishant.** (2013). Optimal Design of FIR and IIR Filters using some Evolutionary Algorithms.

[4] **C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi,** "Investigating RNN-based speech enhancement methods for noise-robust textto-speech," in ISCA Speech Synthesis Workshop, 2016.

[5] **R. S. Kumar** (2005), "Performance improvement in the bivariate models by using modified marginal variance of noisy observation for image-denoising applications", World Academy of Science, Engineering & Technology, Vol.5. pp.38-45.

[6] **S. H. Moon** (2010), "Importance of phase information in speech enhancement", International conference on complex, intelligent and software intensive systems, pp.770-773.

[7] **J Murphy, S. God sill** (2011), "Joint Bayesian removal of impulse and background noise", IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 261–264.

[8] **J. Jebastine, B. S. Rani** (2012), "Design and implementation of noise free Audio speech signal using fast block least Mean square algorithm", Signal & Image Processing : An International Journal, Vol.3, No.3, pp.39-53.

[9] **R. S. Kumar** (2005), "Performance improvement in the bivariate models by using modified marginal variance of noisy observation for image-denoising applications", World Academy of Science, Engineering & Technology, Vol.5.pp.38-45.

[10] **S. H. Moon** (2010), "Importance of phase information in speech enhancement", International conference on complex, intelligent and software intensive systems, pp.770-773.

[11] **J Murphy, S. God sill** (2011), "Joint Bayesian removal of impulse and background noise", IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 261–264.

[12] **M. Niedzwiecki, M. Ciolek** (2013), "Elimination of impulsive disturbances from archive audio signals using bidirectional processing,", IEEE Transactions on Audio, Speech and Language Processing.

[13] Powerdesign.com/article_monteith_112509.html> [6th Dec 2011].Mohammad Motiur R ahman Mithun Kumar P K and Mohammad Shorif Uddin, Optimum Threshold Parameter Estimation of Wavelet Coefficients Using Fisher Discriminant Analysis for Speckle Noise Reduction', The International Arab Journal of Information Technology, vol.11, No_6, Nov2014.

[14] https://github.com/youefkh05/The_9_Sines

[15] https://drive.google.com/drive/folders/1Ibf_xtesOiC7FMdtRZ8CuD5CH0KI0yy0

[16] https://www.tinkercad.com/things/iuqdhKYt6eQ-the-9-sines-future/editel?returnTo=%2Fdashboard

APPENDIX:

Our codes are here:

https://drive.google.com/drive/u/0/folders/1Y5geGvLuV8uRkiI-eebuQKIvkGOPMNaM

Or scan QR Code: