República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación Superior

Universidad Nacional Experimental de Guayana

Vicerrectorado Académico

Asignatura: Telecomunicaciones

PROYECTO DE TELECOMUNICACIONES I

Profesor:

Manuel Paniccia

Estudiante:

Ramírez Omar

Ciudad Guayana 25 de agosto del 2020

I. INTRODUCCION

El análisis de dominio de frecuencia es una herramienta de suma importancia en las aplicaciones de procesamiento de señales. El análisis de dominio de frecuencia se utiliza ampliamente en áreas tales como comunicaciones, geología, teledetección y procesamiento de imágenes. Mientras que el análisis de dominio de tiempo muestra cómo cambia una señal con el tiempo, el análisis de dominio de frecuencia muestra cómo se distribuye la energía de la señal en un rango de frecuencias. Una representación de dominio de frecuencia también incluye información sobre el desplazamiento de fase que se debe aplicar a cada componente de frecuencia para recuperar la señal de tiempo original con una combinación de todos los componentes de frecuencia individuales

Una señal se puede convertir entre los dominios de tiempo y frecuencia con un par de operadores matemáticos llamados transformación. Un ejemplo es la transformación de Fourier, que descompone una función en la suma de un número (potencialmente infinito) de componentes de frecuencia de onda sinusoidal. El "espectro" de componentes de frecuencia es la representación del dominio de frecuencia de la señal. La transformación inversa de Fourier convierte la función de dominio de frecuencia en una función de tiempo. Las funciones de Python permiten calcular la transformación Discrete Fourier (DFT) de una señal y la inversa de esta transformación respectivamente con la librería scipy usando la función fft y ifft

II. DESARROLLO

¿Qué es el análisis de frecuencia?

Para procesos cíclicos, como rotación, oscilaciones u ondas, la frecuencia se define como un número de ciclos por unidad de tiempo. Para conteos por unidad de tiempo, la unidad SI para frecuencia es hercios (Hz); 1 Hz significa que un evento se repite una vez por segundo.

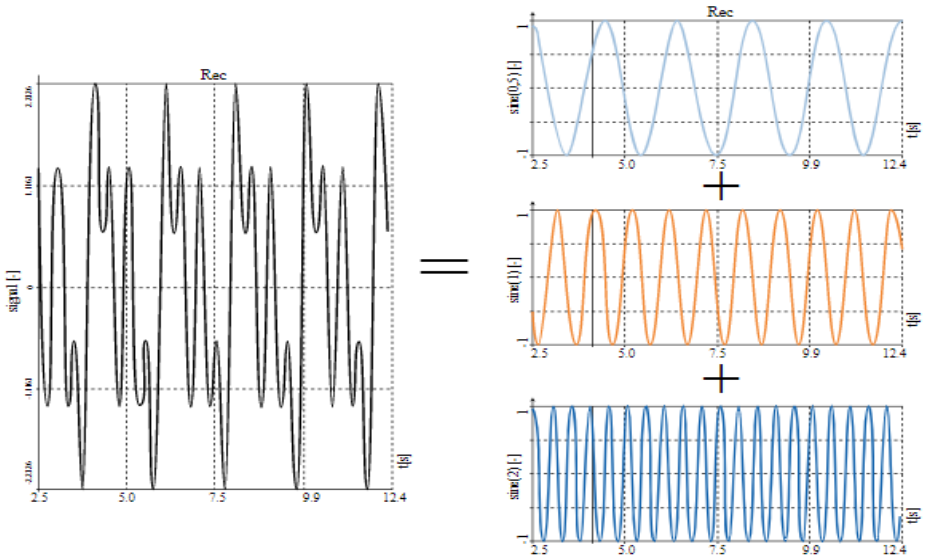
El período de tiempo (T) es la duración de un ciclo y es el recíproco de la frecuencia (f):

$$T = \frac{1}{f}$$

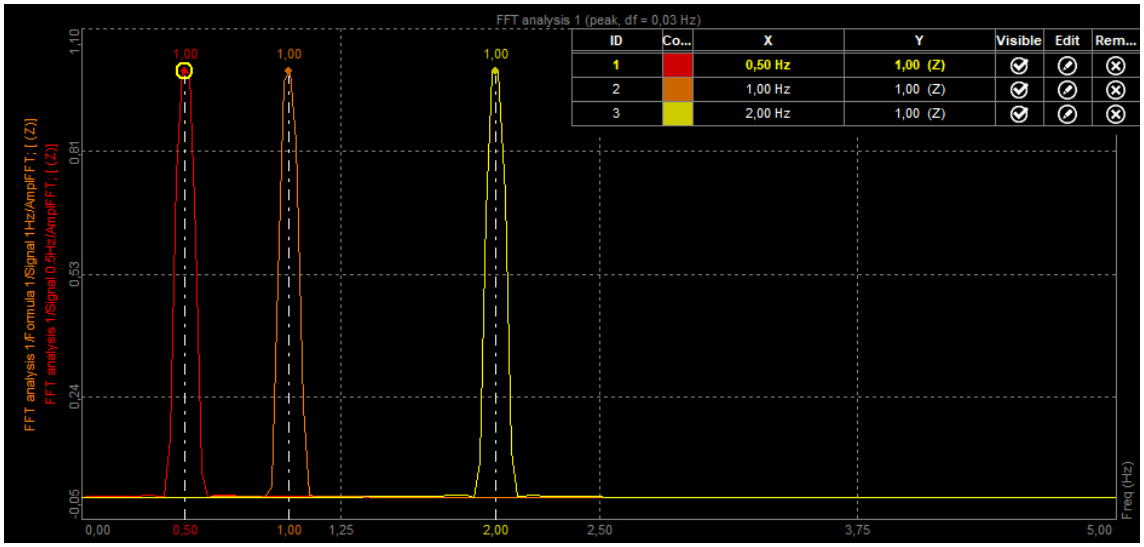
El análisis de frecuencia es solo otra forma de ver los mismos datos. En lugar de observar los datos en el dominio del tiempo, con algunos análisis de frecuencia matemáticos no muy difíciles pero ingeniosos, se descomponen los datos de tiempo en la serie de ondas sinusales.

También podemos decir que el análisis de frecuencia comprueba la presencia de determinadas frecuencias fijas.

La siguiente imagen muestra la señal, que consta de tres ondas sinusoidales con las frecuencias de 0,5 Hz, 1 Hz y 2 Hz, y luego en el lado derecho la señal descompuesta.



Solo para hacer que esas ondas sinusoidales sean más visibles, mostrémoslas de una manera más agradable. En el eje x, hay frecuencias y en el eje y, hay amplitudes de las ondas sinusoidales.



Y de esto se trata realmente el análisis de frecuencia: mostrar la señal como la suma de las señales sinusales. Y la comprensión, cómo funciona, nos ayuda a superar los problemas que trae consigo.

Transformada de Fourier

El matemático Fourier demostró que cualquier función continua podía producirse como una suma infinita de ondas seno y coseno. Su resultado tiene implicaciones de gran alcance para la reproducción y síntesis del sonido. Una onda sinusoidal pura se puede convertir en sonido mediante un altavoz y se percibirá como un tono puro y constante de un solo tono. Los sonidos de los instrumentos orquestales generalmente consisten en un fundamental y un complemento de armónicos, lo que puede considerarse una superposición de ondas sinusoidales de una frecuencia fundamental f y múltiplos enteros de esa frecuencia.

El análisis de Fourier de una función periódica se refiere a la extracción de la serie de senos y cosenos que cuando se superponen reproducirán la función. Este análisis se puede expresar como una serie de Fourier.

Series de Fourier

Cualquier forma de onda periódica se puede descomponer en una serie de ondas seno y coseno:

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} \left[A_n \cos\left(\frac{n\pi t}{p}\right) + B_n \operatorname{sen}\left(\frac{n\pi t}{p}\right) \right]$$

Donde a0, an y bn son coeficientes de Fourier:

$$A_0 = \frac{1}{p} \int_{-p}^{p} f(t)dt$$

$$A_n = \frac{1}{p} \int_{-p}^{p} f(t) \cos\left(\frac{n\pi t}{p}\right) dt$$

$$B_n = \frac{1}{p} \int_{-p}^{p} f(t) \operatorname{sen}\left(\frac{n\pi t}{p}\right) dt$$

Si la función es par:

$$A_0 = \frac{2}{p} \int_{0}^{p} f(t)dt$$

$$A_n = \frac{2}{p} \int_{0}^{p} f(t) \cos\left(\frac{n\pi t}{p}\right) dt$$

$$B_n = 0$$

Si la función es impar:

$$A_0 = 0$$

$$A_n = 0$$

$$B_n = \frac{2}{p} \int_{0}^{p} f(t) \operatorname{sen}\left(\frac{n\pi t}{p}\right) dt$$

Transformada discreta de Fourier

Para datos discretos, la base computacional del análisis espectral es la transformada discreta de Fourier (DFT). La DFT transforma datos basados en el tiempo o en el espacio en datos basados en frecuencia.

La DFT de un vector x de longitud n es otro vector y de longitud n:

$$y_{p+1} = \sum_{j=0}^{n-1} \omega^{jp} x_{j+1}$$

donde ω es una raíz n-ésima compleja de la unidad:

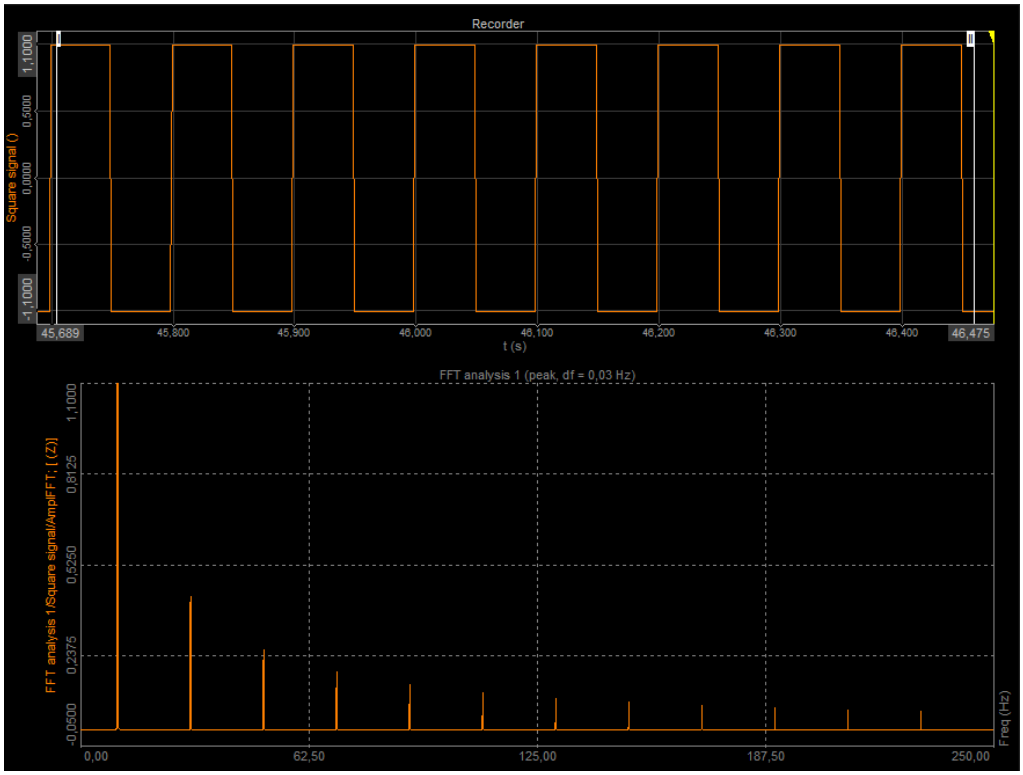
$$\omega = e^{-2\pi i/n}$$

Usamos i para la unidad imaginaria y p y j para índices que van de 0 a n – 1. Los índices p + 1 y j + 1 van de 1 a n.

Se supone que los datos del vector x están separados por un intervalo constante en el tiempo o el espacio, dt = 1 / fs o ds = 1 / fs, donde fs es la frecuencia de muestreo. La DFT y tiene un valor complejo. El valor absoluto de y en el índice p + 1 mide la cantidad de frecuencia (f = p (fs / n)) presente en los datos.

El primer elemento de y, correspondiente a la frecuencia cero, es la suma de los datos en x. Este componente de CC a menudo se elimina de y para que no oscurezca el contenido de frecuencia positiva de los datos.

Un ejemplo de esto es la onda cuadrada de la siguiente imagen. Una onda cuadrada se compone de una suma infinita de ondas sinusoidales.



Pensemos en cómo funciona la ecuación para la transformada discreta de Fourier:

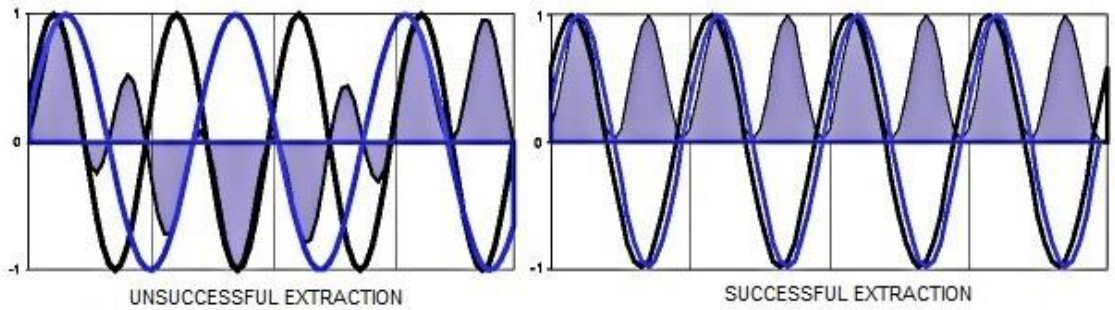
$$X(k\omega_0) = \sum_{n}^{N-1} x[n] \cdot (cos(2\pi kn/N) + i \cdot sin(2\pi kn/N))$$

Para verificar la presencia de cierta onda sinusoidal en una muestra de datos, la ecuación hace lo siguiente:

Multiplica la señal por una onda sinusoidal de esa frecuencia que queremos extraer. La siguiente imagen muestra la señal (línea negra), que consta solo de una onda sinusoidal con 50 Hz. Intentamos extraer los 36 Hz del lado izquierdo y los 50 Hz del lado derecho (se muestran como líneas azules). La onda llena de color azul claro muestra valores multiplicados

Los valores multiplicados se suman y este es el truco principal. Si hay un componente en una señal como en la imagen de la derecha, la multiplicación de las partes de la señal positiva y las ondas sinusoidales de extracción dan un resultado positivo. Además, la multiplicación de las partes de la señal negativa y las ondas sinusoidales de extracción negativa da resultados positivos (observe la imagen de la derecha). En este caso, la suma de las ondas sinusoidales multiplicadas será distinta de cero y mostrará la amplitud de la parte de 50 Hz de la señal. En el caso de 36 Hz, hay ambos lados positivos y negativos de los valores de multiplicación y la suma será (casi, como veremos más adelante) cero.

Y eso es. Esa suma da la estimación de la presencia de frecuencias en la señal. Comprobamos el seno y el coseno para obtener también el cambio de fase (en el peor de los casos, si el cambio de fase fuera de 90 grados, la suma de las funciones del seno siempre daría cero)



El principio que se muestra arriba puede extraer básicamente cualquier frecuencia de la onda sinusoidal, pero tiene una desventaja: es terriblemente lento. El siguiente paso importante en el uso de DFT fue el algoritmo FFT: este análisis reduce el número de cálculos al reorganizar los datos. La desventaja es solo que las muestras de datos deben tener una longitud, que es la potencia de dos (como 256, 512, 1024 y así sucesivamente). Aparte de eso, el resultado es prácticamente el mismo que para el DFT.

FFT - Transformada rápida de Fourier

La transformada rápida de Fourier es un método matemático para transformar una función de tiempo en una función de frecuencia. Se describe como una transformación del dominio del tiempo al dominio de la frecuencia.

La transformada rápida de Fourier (FFT) es un desarrollo de la transformada discreta de Fourier (DFT) que elimina los términos duplicados en el algoritmo matemático para reducir el número de operaciones matemáticas realizadas. De esta forma, es posible utilizar un gran número de muestras sin comprometer la velocidad de la transformación. La FFT reduce el cálculo en un factor de N / (log2 (N)).

FFT calcula la DFT y produce exactamente el mismo resultado que evaluar la DFT; ¡la diferencia más importante es que una FFT es mucho más rápida!

Sean x0, ..., xN-1 números complejos. Ya hemos visto que DFT se define por la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1.$$

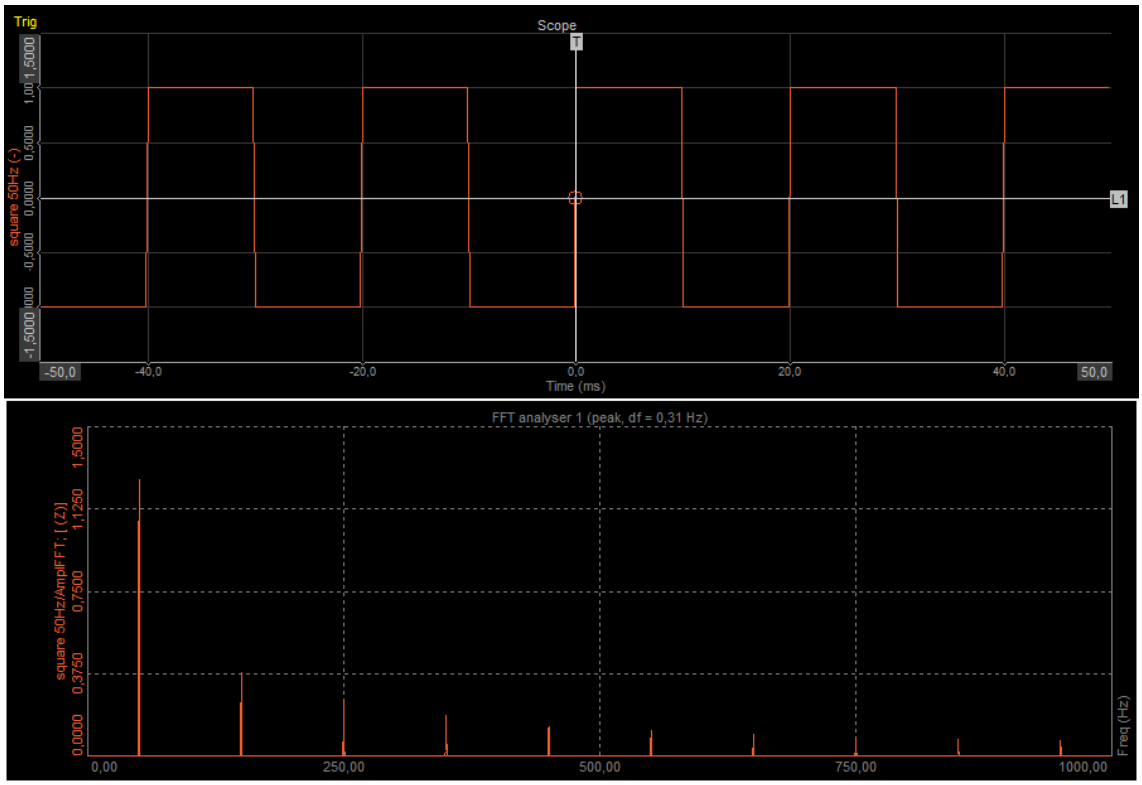
Evaluar esta definición directamente requiere N2 operaciones: hay N salidas de Xk, y cada salida requiere una suma de N términos. Una FFT es cualquier método para calcular los mismos resultados en operaciones N log (N). Todos los algoritmos de FFT conocidos requieren N operaciones log (N).

Para ilustrar los ahorros de una FFT, considere el recuento de sumas y multiplicaciones complejas. La evaluación de las sumas de la DFT implica directamente N2 multiplicaciones complejas y N (N − 1) sumas complejas. El algoritmo FFT puede calcular el mismo resultado con solo (N / 2) multiplicaciones complejas log2 (N) y adiciones complejas Nlog2 (N).

	DFT	FFT
multiplicaciones complejas	N2	(N / 2)log2 (N)
adiciones complejas	N (N-1)	N / log2 (N)

En la práctica, el rendimiento real en las computadoras modernas generalmente está dominado por factores distintos a la velocidad de las operaciones aritméticas y el análisis es un tema complicado, pero la mejora general de N2 a N log2 (N) permanece.

En la imagen a continuación, puede ver los datos originales de una señal en el dominio del tiempo (unidades en segundos [s]) y los datos después de la transformación rápida de Fourier en el dominio de la frecuencia (unidades en hercios [Hz]).



Una vez que conozca el contenido armónico de una señal del análisis de Fourier, tendrá la capacidad de sintetizar esa señal a partir de una serie de generadores de tonos puros ajustando adecuadamente sus amplitudes y fases y sumándolos. Esto se llama síntesis de Fourier.

Modulación (telecomunicación)

Modulación engloba el conjunto de técnicas que se usan para transportar información sobre una onda portadora, típicamente una onda sinusoidal. Estas técnicas permiten un mejor aprovechamiento del canal de comunicación lo que posibilita transmitir más información de forma simultánea además de mejorar la resistencia contra posibles ruidos e interferencias. Según la American National Standard for Telecommunications, la modulación es el proceso, o el resultado del proceso, de variar una característica de una onda portadora de acuerdo con una señal que transporta información. El propósito de la modulación es sobreponer señales en las ondas portadoras.

Básicamente, la modulación consiste en hacer que un parámetro de la onda portadora cambie de valor de acuerdo con las variaciones de la señal moduladora, que es la información que queremos transmitir.

AM y FM, en el mundo de la radiodifusión, son siglas que se refieren dos formas de modular la onda portadora de señales eléctricas. AM corresponde a las siglas de ‘amplitud modulada’, mientras que FM significa ‘frecuencia modulada’.

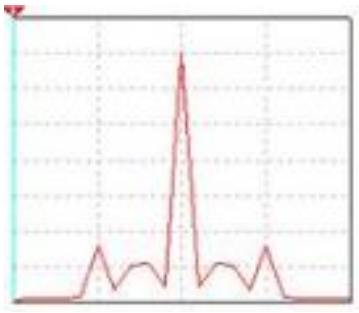
AM o amplitud modulada

AM significa amplitud modulada o modulación de amplitud; es una técnica utilizada en la comunicación electrónica que consiste en hacer variar la amplitud de la onda portadora de la radiofrecuencia. Como tal, fue la primera técnica que se usó para hacer radio.

El canal de la AM tiene un ancho de banda que se encuentra entre 10 KHz y 8 KHz. Debido a que son frecuencias más bajas, cuyas longitudes de onda son mayores, el alcance de su señal es considerablemente más amplio en relación con el de la frecuencia modulada.

En este sentido, las ondas AM pueden medir entre 100 metros (3000 KHz) y 1000 metros (300 KHz). Este es el tipo de onda que llega a la ionosfera y rebota en ella.

No obstante, la calidad de sonido de la amplitud modulada (AM) está muy por debajo de la de la frecuencia modulada (FM). Además, como se trata de ondas de baja frecuencia, son más vulnerables a los ruidos, pues estos se producen en las amplitudes de las ondas. A pesar de ello, es el tipo de onda más aconsejable para zonas montañosas.



Señal AM con dominio en la frecuencia

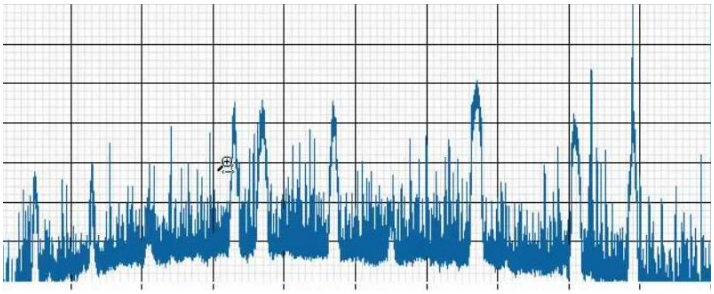
FM o frecuencia modulada

FM significa frecuencia modulada; es una técnica que permite trasmitir información a través de una onda portadora, variando su frecuencia. Como tal, fue patentada en 1933 por el inventor estadounidense Edwin Howard Armstrong.

El canal de frecuencia modulada tiene un ancho de banda de 200 KHz. Semejante ancho permite que los sonidos transmitidos (música y habla) tengan mayor fidelidad y calidad, y que sean más limpios y claros que en la amplitud modulada.

En frecuencia modulada, una emisora transmite en 101.1 MHz (es decir, 101.100 KHz), y la siguiente lo hace en 101.3 MHz (es decir, 101.300KHz). Esto quiere decir que entre un canal y otro quedan libres 200 KHz. Además, permite enviar doble señal, es decir, una señal estéreo.

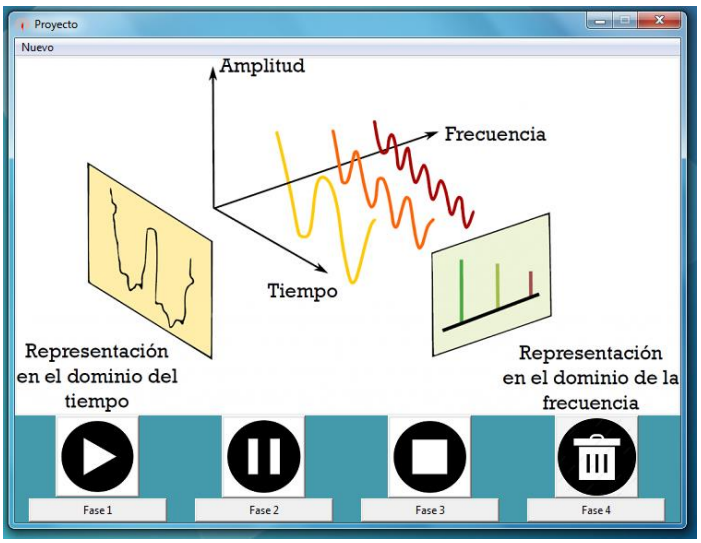
No obstante, el alcance de las señales de frecuencia modulada es inferior al de la amplitud modulada. Esto se debe a que la frecuencia modulada se transmite entre 88 y 108 MHz, es decir, en frecuencias muy altas, cuyas ondas pueden medir entre un metro (300 MHz) y diez metros (30 MHz). Este tipo de ondas, además, tienen longitudes considerablemente pequeñas, de modo que se desplazan en línea recta y se atenúan rápidamente. De allí que sea un tipo de onda idónea para las zonas planas, donde las ondas pueden transmitirse sin obstáculos.



Señal FM con dominio en la frecuencia

III. APLICACION

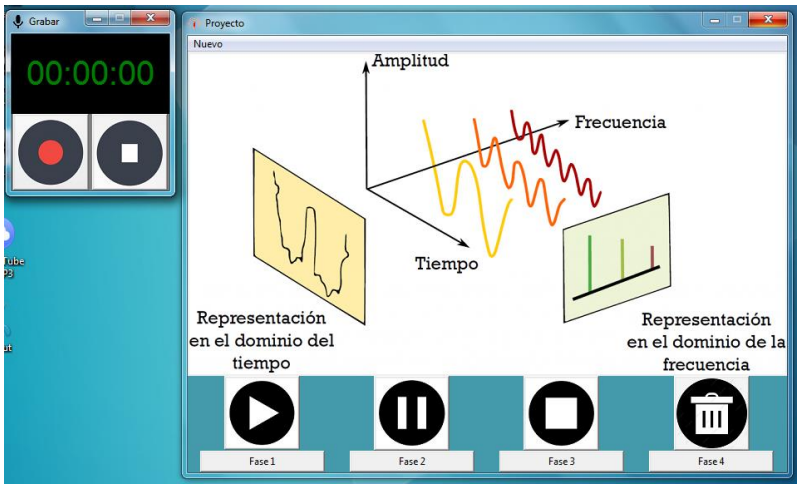
Para el desarrollo del programa se hizo uso de Python en su versión 3.8 y algunas librerías disponibles esenciales, entre ellas: Scipy, Numpy, Matplotlib, Wave y Pyaudio. Scipy y Numpy contienes las funciones y operaciones matemáticas para el desarrollo de todos los cálculos necesarios, Matplotlib se usó para mostrar la gráfica resultante en cada una de las fases y por ultimo Wave y Pyaudio para grabar y leer el audio a utilizar para su análisis en el dominio del tiempo y la frecuencia.



Interfaz principal

El panel inferior permite reproducir, pausar, continuar, parar y eliminar un audio grabado respectivamente como también seleccionar cualquiera de las fases para su respectivo análisis.

En caso de que no haya un audio grabado el programa saltara un aviso de información pidiendo que para continuar se grabe un audio desde la misma aplicación haciendo uso de la barra superior con la opción de “nuevo” desplegando la opción de grabar un audio y salir del programa.



Al seleccionar la opción de grabar hace el llamado a un programa secundario que se encarga de realizar la grabación correspondiente seleccionando la opción de grabar y detener para guardarla en la misma ubicación del programa llamado “Sound.wav”.

Para realizar cada una de las fases pertinentes es necesario convertir el audio estéreo a un audio mono ya que es necesario trabajar desde un solo canal de audio en este caso el izquierdo y asi realizar la transformación del dominio en el tiempo a el dominio en la frecuencia con el siguiente código:

```
f=abs(fft(uncanal))

freq= fftpk.fftfreq(len(f),(1.0/fsonido))
```

A la variable f se le asigna el valor absoluto de la tranformada rápida de Fourier del canal izquierdo del audio porque cuando las series de Fourier son utilizadas de este modo para representar secuencias de duración finita, es llamada la transformada discreta de Fourier y por la información suministrada en el desarrollo del presente informe la transformada rápida de Fourier es solo la transformación mejorada de la transformada discreta de Fourier.

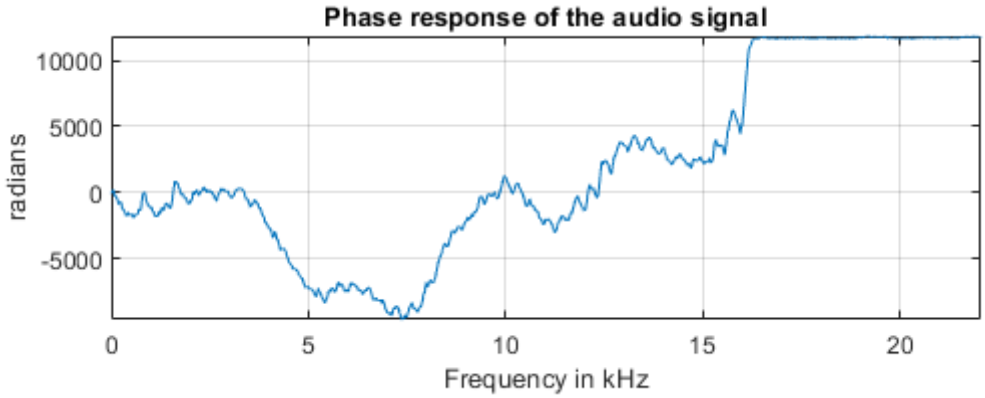
Y por ultimo la variable freq se le asigna la transformada de la frecuencia resultante del largo de f y el periodo 1/fsonido, fsonido es la frecuencia del audio 44100 Hz.

En la fase 1 de muestra el grafico del audio en el dominio del tiempo y de la frecuencia.

La representación de dominio de frecuencia de una señal contiene información sobre la magnitud y la fase de la señal en cada frecuencia. Esta es la razón por la que la salida del cálculo FFT es compleja. Un número complejo, x, tiene una parte real, xr, y una parte imaginaria, xi, de forma que x = xr + xi. La magnitud de x se calcula como $\sqrt{(x_r^2 + x_i^2)}$, y la fase de x se calcula como $\arctan(x_i/x_r)$

La salida del FFT es un vector complejo que contiene información sobre el contenido de frecuencia de la señal. La fase le indica cómo se alinean todos los componentes de frecuencia en el tiempo.

La fase 2 se encargar de trazar los componentes de fase del espectro de frecuencia de la señal. La fase se desenvuelve utilizando la función para que podamos ver una función continua de frecuencia con la función unwrap.



La fase 3 es la modulación en amplitud que a su vez es el proceso de codificación de la información fuente, sonido o moduladora, dentro de una señal pasabanda s(t), resultande o modulada. El algoritmo implementado para esta fase se obtiene del código de la función ammod de matlab:

ammod (x, Fc, Fs) devuelve una señal y modulada en amplitud (AM), dada la señal de mensaje de entrada x, donde la señal portadora tiene la frecuencia Fc. La señal portadora yx tienen una frecuencia de muestreo Fs. La señal modulada tiene fase inicial cero y amplitud de portadora cero, por lo que el resultado es una modulación de portadora suprimida.

Código:

```
t=0:.000001:.005;
Am1=.5;
fm1=400;
Am2=0.25;
fm2=800;
Ac=2.5;
fc=10000;
mt=Am1*cos(2*pi*fm1.*t)+Am2*cos(2*pi*fm2.*t);
```

```
ct=Ac*cos(2*pi*fc.*t);
st=(1+mt).*ct;
figure('Name','Time domain representations of FULL AM signals');
title('AM Modulation of Multitone sinusoidal Signal');
subplot(3,1,1)
plot(t,mt)
xlabel('Time'); ylabel('Message signal');
subplot(3,1,2)
plot(t,ct)
xlabel('Time'); ylabel('Carrier Signal');
subplot(3,1,3)
plot(t,st,t,Ac.*(1+mt),'r')
xlabel('Time'); ylabel('Modulated signal');
sprintf('Carrier frequency: %d Hz',fc)
sprintf('Message frequency: %d Hz and %d Hz',fm1,fm2)
sprintf('USB spectra at: %d Hz and %d Hz',fc+fm1,fc+fm2)
sprintf('LSB spectra at: %d Hz and %d Hz',fc-fm1,fc-fm2)
```

La fase 4 es la modulación de frecuencia que a su vez es una técnica de modulación angular que permite transmitir información a través de una onda portadora variando su frecuencia. En aplicaciones analógicas, la frecuencia instantánea de la señal modulada es proporcional al valor instantáneo de la señal moduladora. El algoritmo implementado para esta fase se obtiene del código de la función fmmod de matlab:

fmmod (x, Fc, Fs, freqdev) devuelve una señal de frecuencia modulada (FM) y, dada la señal de mensaje de entrada x, donde la señal portadora tiene la frecuencia Fc y la frecuencia de muestreo Fs. freqdev es la desviación de frecuencia de la señal modulada.

Código:

```
fs = 1000;
fc = 200;
t = (0:1/fs:0.2)';
x = sin(2*pi*30*t)+2*sin(2*pi*60*t);
fDev = 50;
int_x = cumsum(x)/fs;
xfm = cos(2*pi*fc*t).*cos(2*pi*fDev*int_x)-
sin(2*pi*fc*t).*sin(2*pi*fDev*int_x)   ;
xi=cos(2*pi*fDev*int_x);
xq=sin(2*pi*fDev*int_x) ;
t2 = (0:1/fs:((size(xfm,1)-1)/fs))';
t2 = t2(:,ones(1,size(xfm,2)));
xfmq = hilbert(xfm).*exp(-j*2*pi*fc*t2);
z = (1/(2*pi*fDev))*[zeros(1,size(xfmq,2));
diff(unwrap(angle(xfmq)))*fs];
figure(2);plot(t,x,'c',t2,z,'b--');xlabel('time ');ylabel('amplitude');
legend('Original Signal','Demodulated Signal');grid on
```

IV. CODIGO

Proyecto.pyw

```
from tkinter import *
from tkinter import messagebox
from os import remove
from Fase1 import fase1
from Fase2 import fase2
from Fase3 import fase3
from Fase4 import fase4
import os
import pygame

var=False

#----funciones-----------------------------------------------------------

def grabar():
    pygame.mixer.quit()
    os.system('Grabar.pyw')

def salirapp():
    valor=messagebox.askquestion("Salir", "Desea salir de la app?")
    if valor=="yes":
        raiz.destroy()

def Play():
    try:
        pygame.mixer.init()
        pygame.mixer.music.load("Sound.wav")
        pygame.mixer.music.play()
    except:
        messagebox.showwarning("Error", "El audio no existe, por favor gr
abe uno desde la app")

def Pause():
    global var
    if var==False:
        pygame.mixer.music.pause()
        var=True
    elif var==True:
        pygame.mixer.music.unpause()
        var=False

def Stop():
    pygame.mixer.music.stop()
```

```
def Eliminar():
    pygame.mixer.quit()
    remove("Sound.wav")

#----Interfaz-----------------------------------------------------

raiz=Tk()

raiz.title("Proyecto")

raiz.iconbitmap("C:/Users/Luis Ramirez/Downloads/universidad/tele/pro/sen
al.ico")

raiz.config(bg="#49A")

raiz.resizable(width=False, height=False)

barra=Menu(raiz)

raiz.config(menu=barra, width=400, height=400)

#----------Barra-----------------------------------------------------

menu=Menu(barra, tearoff=0)
menu.add_command(label="Grabar", command=grabar)
menu.add_command(label="Salir", command=salirapp)
barra.add_cascade(label="Nuevo", menu=menu)

#---botones-------------------------------------------------------

fase1=Button(raiz, text="Fase 1", width=20, command=fase1, cursor="hand2"
)
fase1.grid(row=3, column=1)

fase2=Button(raiz, text="Fase 2", width=20, command=fase2, cursor="hand2"
)
fase2.grid(row=3, column=2)

fase3=Button(raiz, text="Fase 3", width=20, command=fase3, cursor="hand2"
)
fase3.grid(row=3, column=3)

fase4=Button(raiz, text="Fase 4", width=20, command=fase4, cursor="hand2"
)
fase4.grid(row=3, column=4)

imagen=PhotoImage(file="signal.png")
im=Label(raiz,image=imagen).grid(row=1, column=1, columnspan=4)
```

```
img1=PhotoImage(file="play.png")
play=Button(raiz, image=img1, command=Play, cursor="hand2")
play.grid(row=2, column=1)

img2=PhotoImage(file="pause.png")
play=Button(raiz, image=img2, command=Pause, cursor="hand2")
play.grid(row=2, column=2)

img3=PhotoImage(file="stop.png")
play=Button(raiz, image=img3, command=Stop, cursor="hand2")
play.grid(row=2, column=3)

img4=PhotoImage(file="trash.png")
play=Button(raiz, image=img4, command=Eliminar, cursor="hand2")
play.grid(row=2, column=4)

raiz.mainloop()
```

Grabar.pyw

```
import pyaudio
import wave
from tkinter import *
import threading
import sys

CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 10
WAVE_OUTPUT_FILENAME = "Sound.wav"

def clear_contador():
    global contador,contador1,contador2
    contador=0
    contador1=0
    contador2=0

def dire():
    directorio_actual.set(os.getcwd())

def iniciar():
    global grabando
    clear_contador()
    bloqueo('disabled')
    grabando=True
```

```
    t=threading.Thread(target=Grabar)
    t.start()
    t1=threading.Thread(target=cuenta)
    t1.start()

def formato(c):
    if c<10:
        c="0"+str(c)
    return c

def cuenta():
    global proceso
    global contador,contador1,contador2
    time['text'] = str(formato(contador1))+":"+str(formato(contador2))+":
"+str(formato(contador))
    contador+=1
    if contador==60:
        contador=0
        contador2+=1
    if contador2==60:
        contador2=0
        contador1+=1
    proceso=time.after(1000, cuenta)

def bloqueo(s):
    btnIniciar.config(state=s)

def parar():
    global grabando
    if grabando==True:
        grabando=False
    bloqueo('normal')
    raiz.destroy()

def Grabar():
    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    #print("* recording")

    frames = []

    while grabando==True:
        data = stream.read(CHUNK)
        frames.append(data)
```

```
    #print("* done recording")

    stream.stop_stream()
    stream.close()
    p.terminate()

    wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))
    wf.close()

raiz=Tk()
raiz.title("Grabar")
raiz.iconbitmap("C:/Users/Luis Ramirez/Downloads/universidad/tele/pro/rec
ord.ico")
raiz.config(bg="grey")
raiz.resizable(width=False, height=False)

grabando=False
p = pyaudio.PyAudio()

time = Label(raiz, fg='green', width=8, height=2, text="00:00:00", bg="bl
ack", font=("","30"))
time.grid(row=1, column=1, columnspan=2)

img1=PhotoImage(file="grabar.png")
btnIniciar=Button(raiz, image=img1, command=iniciar, cursor="hand2")
btnIniciar.grid(row=2, column=1)

img2=PhotoImage(file="detener.png")
btnParar=Button(raiz, image=img2, command=parar, cursor="hand2")
btnParar.grid(row=2, column=2)

raiz.mainloop()
```

Fase1.pyw

```
import scipy.io.wavfile as waves
from scipy.fft import fft
import scipy.fftpack as fftpk
import matplotlib.pyplot as plt
import numpy as np
from tkinter import messagebox

def fase1():
    try:
```

```
        audio='Sound.wav'
        fsonido, sonido = waves.read(audio)

        # Extrae un canal en caso de estéreo
        canales=sonido.shape
        cuantos=len(canales)
        canal = 0
        if (cuantos==1): # Monofónico
            uncanal=sonido
        if (cuantos>=2): # Estéreo
            uncanal=sonido[:,canal]

        # tranformar en la frecuencia
        f=abs(fft(uncanal))
        freq= fftpk.fftfreq(len(f),(1.0/fsonido))

        N = uncanal.shape[0]

        # Salida Grafica
        fig=plt.figure("Filtro")
        fig.subplots_adjust(hspace=0.5, wspace=0.5)

        p1=fig.add_subplot(2,1,1)
        p1.plot(np.arange(N) / fsonido, uncanal)
        p1.set_title('Audio Original')
        p1.set_xlabel("Time [s]")
        p1.set_ylabel("Amplitud")

        p2=fig.add_subplot(2,1,2)
        p2.plot(freq[range(len(f)//2)], f[range(len(f)//2)])
        p2.set_title('Fase 1 con Dominio en la Frecuencia')
        p2.set_xlabel("Frecuencia (Hz)")
        p2.set_ylabel("Amplitud")

        plt.show()
    except:
        messagebox.showwarning("Error", "El audio no existe, por favor gr
abe uno desde la app")
```

Fase2.pyw

```
import scipy.io.wavfile as waves
from scipy.fft import fft
import scipy.fftpack as fftpk
import matplotlib.pyplot as plt
import numpy as np
from tkinter import messagebox
```

```
def fase2():
    try:
        audio='Sound.wav'
        fsonido, sonido = waves.read(audio)

        # Extrae un canal en caso de estéreo
        canales=sonido.shape
        cuantos=len(canales)
        canal = 0
        if (cuantos==1): # Monofónico
            uncanal=sonido
        if (cuantos>=2): # Estéreo
            uncanal=sonido[:,canal]

        # Phase que responde al audio original en la frecuencia
        N=len(uncanal)
        f=fft(uncanal)
        Y=np.unwrap(np.angle(f))
        X=(np.arange(0,1-1/N,1/N))*fsonido

        # Salida Grafica
        plt.plot(X[range(N//2)], Y[range(N//2)])
        plt.title('Fase 2 con Dominio en la Frecuencia')
        plt.xlabel("Frecuencia (Hz)")
        plt.ylabel("Amplitud")
        plt.show()
    except:
        messagebox.showwarning("Error", "El audio no existe, por favor gr
abe uno desde la app")
```

Fase3.pyw

```
import scipy.io.wavfile as waves
from scipy.fft import fft
import scipy.fftpack as fftpk
import matplotlib.pyplot as plt
import numpy as np
from tkinter import messagebox

def fase3():
    try:
        audio='Sound.wav'
        fsonido, sonido = waves.read(audio)

        # Extrae un canal en caso de estéreo
        canales=sonido.shape
        cuantos=len(canales)
        canal = 0
```

```
        if (cuantos==1): # Monofónico
            uncanal=sonido
        if (cuantos>=2): # Estéreo
            uncanal=sonido[:,canal]

        k=500   # k son las muestras que utilizaremos de la senal origina
l. al usar el largo del array como len(uncanal)
                #la amplitud de la señal portadora es mucho mayor
        moduladora=uncanal[0:k].astype(float)
        dt=1/fsonido
        t=np.arange(0,k*dt,dt)

        # Portadora:
        fc = 5500
        portadora = np.cos(2*np.pi*fc*t)

        # normalizar y subir a positiva
        moduladoranorm = moduladora/np.max(moduladora)
        moduladora = (1+ moduladoranorm)

        # Modular portadora
        Ac=1
        modulada = Ac*moduladora*portadora
        modulada=abs(fft(modulada))
        freq= fftpk.fftfreq(len(modulada),(1.0/fsonido))

        # SALIDA GRAFICA
        plt.plot(freq[range(len(modulada)//2)], modulada[range(len(modula
da)//2)],label='modulada')
        plt.title(' Señal modulada en AM S(t)')
        plt.xlabel('Frecuencia (Hz)')
        plt.ylabel('Amplitud')
        plt.legend()
        plt.show()
    except:
        messagebox.showwarning("Error", "El audio no existe, por favor gr
abe uno desde la app")
```

Fase4.pyw

```
import scipy.io.wavfile as waves
from scipy.fft import fft
import scipy.fftpack as fftpk
import matplotlib.pyplot as plt
import numpy as np
from tkinter import messagebox

def fase4():
```

```
    try:
        audio='Sound.wav'
        fsonido, sonido = waves.read(audio)

        # Extrae un canal en caso de estéreo
        canales=sonido.shape
        cuantos=len(canales)
        canal = 0
        if (cuantos==1): # Monofónico
            uncanal=sonido
        if (cuantos>=2): # Estéreo
            uncanal=sonido[:,canal]

        k=500   # k son las muestras que utilizaremos de la senal origina
l. al usar el largo del array como len(uncanal)
            #la cantidad de la señal es mucho mayor
        moduladora=uncanal[0:k].astype(float)
        dt=1/fsonido
        t=np.arange(0,k*dt,dt)

        # Portadora:
        Fc=fsonido
        FS=2.2*Fc
        freqdev=75e3

        # Modular portadora
        int_x = np.cumsum(moduladora)/FS

        xi=np.cos(2*np.pi*freqdev*int_x)
        xq=np.sin(2*np.pi*freqdev*int_x)

        FM = np.cos(2*np.pi*Fc*t)*xi-np.sin(2*np.pi*Fc*t)*xq

        FM=abs(fft(FM))
        freq= fftpk.fftfreq(len(FM),(1.0/fsonido))

        # SALIDA GRAFICA
        plt.plot(freq[range(len(FM)//2)], FM[range(len(FM)//2)],label='mo
dulada')
        plt.title(' Señal modulada en FM S(t)')
        plt.xlabel('Frecuencia (Hz)')
        plt.ylabel('Amplitud')
        plt.legend()
        plt.show()
    except:
        messagebox.showwarning("Error", "El audio no existe, por favor gr
abe uno desde la app")
```

V. CONCLUSION

En este trabajo se ha realizado un estudio acerca de la implementación o aplicación de técnicas para desarrollar un algoritmo que permita la captura de una señal de audio, provenientes del habla humana, su procesado y visualización de los resultados mostrando así cada uno de sus espectros solicitados con dominio en la frecuencia; como resultado de la investigación se halló el uso de un algoritmo denominado la Transformada Rápida de Fourier (FFT), utilizado en entornos computacionales con el propósito de estudiar los patrones de las ondas y sus elementos.

Considerando falta de experiencia previa en las librerías del lenguaje de programación Python y conocimiento en técnicas o modelos matemáticos para el estudio de señales, se puede concluir en que el resultado obtenido es satisfactorio desde el punto de vista de los resultados.

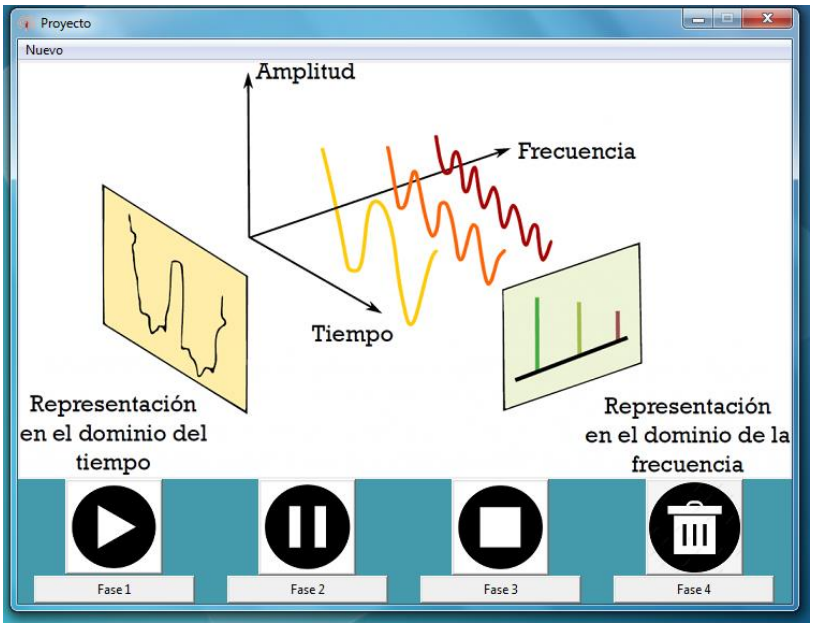
Con respecto a los datos que se obtuvieron a través de los resultados de la simulación y pruebas realizadas con el algoritmo diseñado se concluye que:

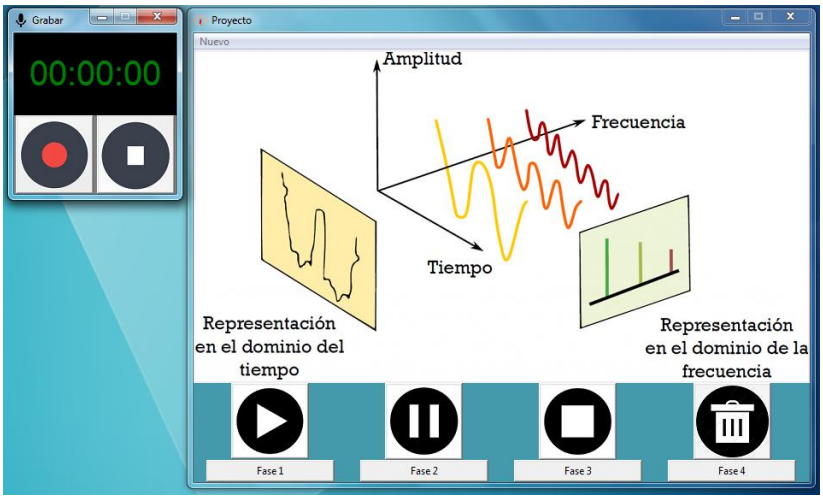
- Los métodos utilizados como la FFT fueron herramientas útiles y fundamentales en el desarrollo del programa.
- Matlab es una gran biblioteca con excelente reportorio de información esencial para el desarrollo del programa.
- La información de fase de una fft indica cómo se alinean todos los componentes de frecuencia en el tiempo visualizando así un gráfico aproximado del audio.
- El desarrollo del análisis de frecuencia de señales AM y FM hace un amplio uso del dominio de la frecuencia.

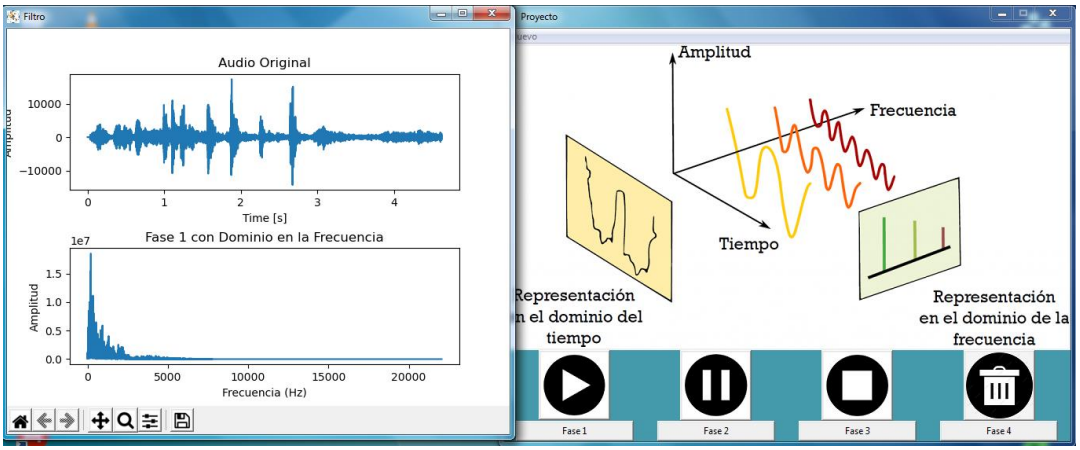
En concusión, se puede afirmar que se han cumplido los objetivos propuestos del proyecto. Se han respetado todas las restricciones iniciales y se ha diseñado y desarrollado un algoritmo de evaluación de sonidos provenientes de la voz y su procesado para su análisis.

Además es de importancia resaltar que consideramos que este es un tema importante y que existen muchos aspectos por investigar. El espectro es una herramienta muy importante para el estudio de la fonética acústica y por lo tanto es un campo de interesante para una mayor investigación.

ANEXOS


Proyecto
Nuevo
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4


Grabar
00:00:00
Proyecto
Nuevo
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4


Filtro
Audio Original
Amplitud
Time [s]
Fase 1 con Dominio en la Frecuencia
1e7
Amplitud
Frecuencia (Hz)
Proyecto
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4

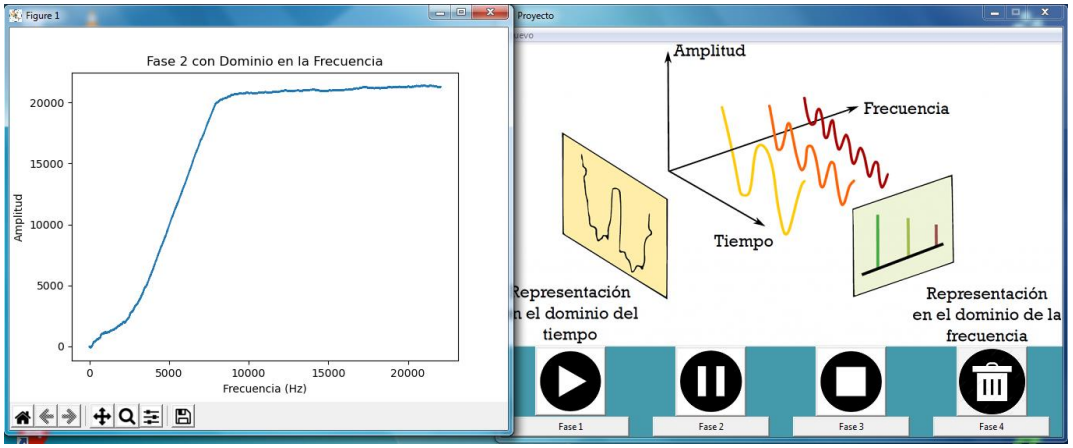


Figure 1
Fase 2 con Dominio en la Frecuencia
Amplitud
Frecuencia (Hz)
Proyecto
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4

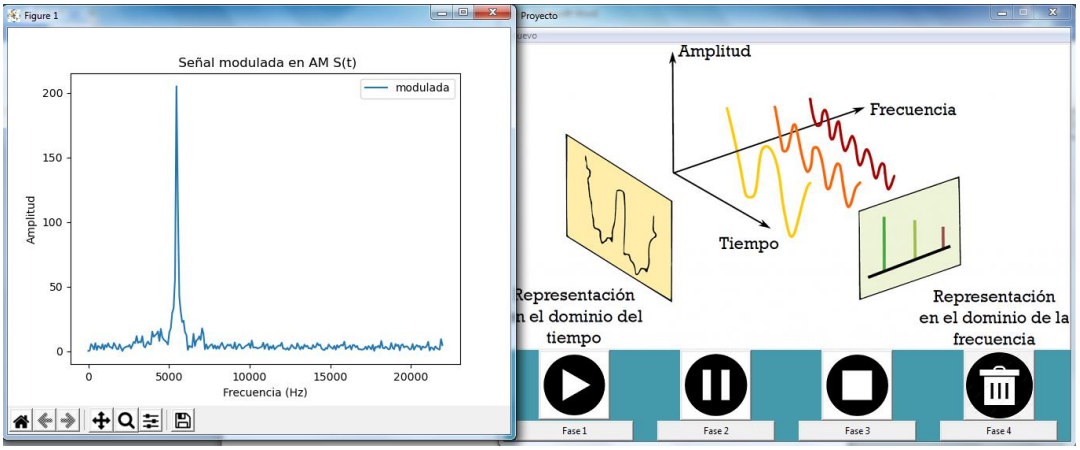


Figure 1
Señal modulada en AM S(t)
modulada
Amplitud
Frecuencia (Hz)
Proyecto
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4

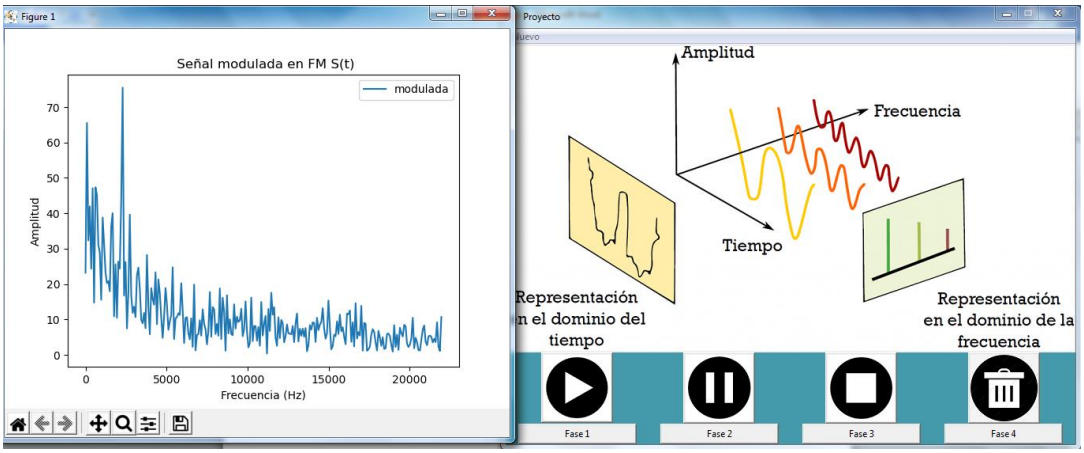


Figure 1
Señal modulada en FM S(t)
modulada
Amplitud
Frecuencia (Hz)
Proyecto
Amplitud
Frecuencia
Tiempo
Representación en el dominio del tiempo
Representación en el dominio de la frecuencia
Fase 1
Fase 2
Fase 3
Fase 4