

# Estudio de señales de audio con la FFT

DANIEL ROBERTO GARCIA MIRANDA

Carrera de Física, Universidad Mayor de San Andrés

Abril del 2025

## Resumen

Para iniciarse con el procesamiento de señales y el uso de la Fast Fourier Transform (FFT), se analizaron señales variadas de audio utilizando el software MATLAB. A estas señales se les aplicó la FFT y, en base a sus coeficientes, se construyeron sus espectrogramas y sonogramas 2D y 3D. Aparte, como aplicaciones de lo aprendido, se programaron tres aplicaciones: un constructor de sonograma 2D en tiempo real, un detector de nota musical y un detector de hablante. Para hacer el análisis más dinámico, todos los códigos utilizados, para cada etapa, fueron integrados en uno solo, donde el usuario escoge la función que desea utilizar.

## Abstract

To get started with signal processing and the use of the Fast Fourier Transform (FFT), various audio signals were analyzed using the MATLAB software. The FFT was applied to these signals, and based on their coefficients, their 2D and 3D spectrograms and sonograms were constructed. Additionally, as applications of what was learned, three programs were developed: a real-time 2D sonogram builder, a musical note detector, and a speaker detector. To make the analysis more dynamic, all the codes used for each stage were integrated into a single program, where the user selects the function they wish to use.

## I. Introducción

Una señal es el resultado de la observación o medición de una cantidad física que varía con el tiempo, espacio o cualquier otra variable o variables independientes. En el contexto de la ingeniería, una señal se define como una "magnitud eléctrica cuya variación con el tiempo representa una información". Ejemplos de señal son el cambio de un semáforo, la tensión en bornes de un micrófono o una sirena de alarma. Las señales pueden servir para comunicaciones entre personas, personas y máquinas o solamente entre máquinas [8].

Las señales son representadas por funciones matemáticas, la señal de voz, por ejemplo, se representa como una función de variable temporal,  $f(t)$ ; en cambio, las imágenes se pueden considerar como funciones de dos variables espaciales  $f(x,y)$ . Además, las señales pueden ser escalares

o vectoriales, una señal de audio capturada con un micrófono monofónico tendrá un solo valor de tensión eléctrica en cada instante de tiempo. Por otro lado, un electroencefalograma provee un conjunto o vector de señales eléctricas provenientes de los diferentes electrodos para cada instante  $t$ :

$$\mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ \dots \\ f_3(t) \end{bmatrix}$$

Cada componente de las señales vectoriales se les llama canal, por eso, a este tipo de señales se las llama multicanal.

Las variables de las que dependen las señales pueden ser discretas o continuas, la distancia entre cada punto de la variable independiente no necesariamente es el mismo, pero por conveniencia matemática y computacional, se las toma

como equidistantes, figura 1.

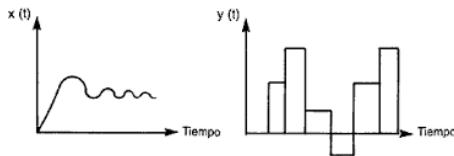


Figura 1: Ejemplo de señales discretas y continuas.

De igual forma, los valores de las señales pueden ser discretos o continuos, respecto a las discretas, estas pueden ser equidistantes o seguir patrones complejos. El término digital, se utiliza para señales de variables independientes discretas y de valores discretos, mientras que analógica es una señal con variables independientes continuas y valores continuos. Suele ser más conveniente analizar señales digitales, ya que se vuelve más simple representarla matemáticamente, ya que son señales en tiempo discreto, donde su variable independiente se representa en instantes de tiempo definidos. [3].

## I. Señales de Audio

El sonido es cualquier fenómeno que involucre la propagación de ondas mecánicas con una rapidez que depende de las propiedades del medio. Su naturaleza es ondulatoria, es decir, se propaga en forma de ondas analógicas desde el objeto que lo produce [9].

- **Intensidad.** La fuerza con la que se produce el sonido, esta se mide en decibeles (dB).
- **Altura.** Es una propiedad la cual clasifica al sonido como agudo, medio y grave; este es el tono del sonido y se mide en Hertz.
- **Timbre.** Se le considera como el sonido característico de una voz o instrumento. De acuerdo a las vibraciones se produce el timbre.
- **Duración.** Comprende el tiempo que se escucha un sonido.

Una señal de audio, es una señal electrónica que es una representación eléctrica exacta de una señal sonora, acotada en el rango de frecuencias audibles para los seres humanos, entre los 20 y 20000 Hz.

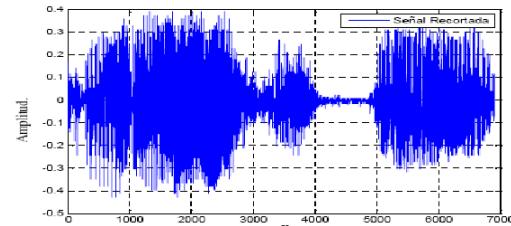


Figura 2

Como el sonido es una onda de presión se requiere un transductor de presión (micrófono) que convierte las ondas sonoras en señales eléctricas. Una señal de audio se puede caracterizar, por su dinámica (valor de pico, rango dinámico, potencia, relación señal-ruido) o por su espectro de potencia (ancho de banda, frecuencia fundamental, armónicos, distorsión armónica, etc.). Estas señales son analógicas, para procesar estas señales por medios digitales es necesario convertirlas a formato digital, este proceso de llama conversión analógico digital (ADC) y los dispositivos conversores (ADCs). Esta conversión ADC consta de 3 pasos:

- **Muestreo.** Se convierte la señal en tiempo discreto con valores continuos a una señal en tiempo discreto con valores discretos (señal digital). Si la entrada es  $x(t)$  la salida será  $x(nT)=x(n)$ , donde  $T$  es el intervalo de muestreo, y su inversa la frecuencia de muestreo.
- **Cuantificación.** El valor de cada muestra de la señal se representa mediante un valor seleccionado de un conjunto finito de valores posibles.
- **Codificación.** Cada valor discreto  $x(n)$  se representa mediante secuencia binaria de  $b$  bits.

La conversión ADC se efectúa en un mismo dispositivo, por ejemplo en los micrófonos de las

computadoras. [6] En lo que se refiere a la representación visual de las magnitudes de medida del sonido, son el oscilograma, espectrograma y el sonograma.

El oscilograma analiza las dimensiones de amplitud y tiempo de un sonido, muestra gráficamente las diferencias de voltaje producidas por una onda sonora durante un cierto tiempo y las representa en un eje, amplitud vs tiempo. Nos permite estudiar detalladamente la dimensión temporal del sonido.

El espectrograma nos ayuda a estudiar el espectro de frecuencias de un objeto sonoro a partir de un procedimiento llamado Transformada Rápida de Fourier (FFT), la FFT es un procedimiento matemático que descompone un sonido complejo en todas las ondas sinusoides que lo forman, el mismo se utiliza para construir el sonograma. Este nos muestra en el eje horizontal todas las frecuencias existentes de menor a mayor y en el eje vertical magnitudes relacionadas con la amplitud, toda esta información en un instante determinado de tiempo.

El sonograma combina la información del oscilograma y el espectrograma, representa la frecuencias presentes en un sonido, la duración temporal del sonido y su intensidad, estas magnitudes pueden ser presentadas en un gráfico 2D, en el eje horizontal se muestra la duración temporal del sonido, la frecuencia en el eje vertical y la intensidad de las frecuencias en escala cromática (densidad de potencia espectral DPS). A un gráfico 3D se le puede incluir la amplitud del sonido en el eje z, el tiempo en el eje y, la frecuencia en el eje x, y la intensidad en escala cromática. Dependiendo que tanta información sea requerida visualizar o qué tipo de variaciones queramos identificar, podemos utilizar cualquiera de estas 3 representaciones gráficas [5].

## II. Transformada Rápida de Fourier (Fast Fourier Transform FFT)

La transformada de Fourier es una herramienta matemática que nos permite pasar de una representación temporal a una representación en el dominio frecuencial. De manera general, la transformada de Fourier mapea cualquier

señal analógica estacionaria a una serie infinita de sinusoides, cada una de ellas con determinada amplitud y fase. La transformada discreta de Fourier (DFT) es una transformada que trabaja con señales discretas, es decir muestradas, ecuación 1.

$$X_{\text{DFT}}(k\omega_0) = \sum_{n=0}^{N-1} x(nT)e^{-j\frac{2\pi kn}{N}} \quad (1)$$

La transformada rápida de Fourier es un algoritmo de cálculo de la DFT, ecuación 1. que requiere mucho menor esfuerzo computacional que el cálculo directo de la DFT. Se basa en el cálculo iterativo de los coeficientes de la DFT de manera que se optimiza el número de operaciones a realizar, optimizando, no solo el tiempo de cálculo sino que también reduce los errores de redondeo asociado [4].

## III. MATLAB y la FFT

MATLAB es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. En MATLAB la función fft se basa en la biblioteca llamada FFTW.

Como se mencionó en la anterior sección, la FFT se aplica sobre un vector de señales grabadas a partir de un micrófono, figura 3.

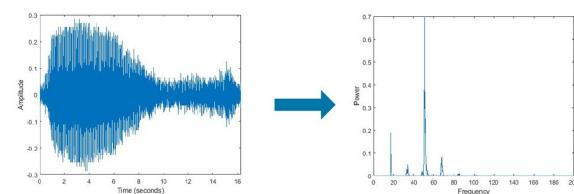


Figura 3: Uso de la FFT.

Para la toma de datos de audio, se utilizan una serie de comandos mostrados en la figura 4.

```

recObj = audiorecorder(Fs, NBits, ch);
recordblocking(recObj, t);
YW = getaudiodata(recObj);
filexxx = 'Audio.wav';
audiowrite(filexxx, YW, Fs);
sound(YW, Fs);

```

Figura 4: Código en MATLAB para grabar una señal de audio.

El código presentado en la Figura 4 graba una señal de audio y la almacena en un archivo ‘.wav’. A continuación, se describe la función de cada comando utilizado:

- **audiorecorder(Fs, NBits, ch):** Crea un objeto de grabación con la frecuencia de muestreo **Fs**, la resolución **NBits** y el número de canales **ch**.
- **recordblocking(recObj, t):** Inicia la grabación y bloquea la ejecución durante **t** segundos.
- **getaudiodata(recObj):** Extrae la señal de audio grabada y la almacena en una variable.
- **audiowrite(filexxx, YW, Fs):** Guarda la señal de audio en un archivo ‘.wav’.
- **sound(YW, Fs):** Reproduce el audio grabado con la frecuencia de muestreo especificada.

Este procedimiento permite registrar y reproducir señales de audio en MATLAB. El Vector **YW** contiene toda la información de la señal de audio, es sobre este vector al que se le aplica la FFT, un código que calcula toda la información requerida es el de la figura 5

```

[YW, Fs] = audioread(filexxx);
FYW = fft(YW);
FYW(1) = [];%Remueve la componente DC
nyquist = Fs/2;
n = length(FYW); %Tamaño de la muestra
freq = (1:n/2) / (n/2) * nyquist;
Amp = 2 * abs(FYW(1:floor(n/2))/N);
DPS = N*0.5 * Amp.^2;
DPSDb = 10 log10(DPS*10.^12);

```

Figura 5: Código en MATLAB que calcula la FFT de la señal y de la misma calcula su rango de frecuencias, amplitud y la densidad de potencia espectral (DPS).

El código de la figura 5, realiza la FFT de la señal de audio grabada con el código de la figura 4 y su frecuencia de muestreo; sobre esta frecuencia es la que se grafica el espectrograma y sonograma, siendo el eje x el de las frecuencias (vector **frec** en el código) y en el eje y la amplitud o la potencia. En los ejes de las frecuencias solamente se grafica hasta cierta frecuencia, la frecuencia de Nyquist, la cual corresponde hasta la mitad de la frecuencia muestral, esto debido a que la FFT calcula la DFT simétricamente respecto a esta frecuencia, por ello, se debe utilizar una frecuencia muestral el doble del máximo del rango de frecuencias a analizar.

Para graficar el sonograma, en un gráfico 2D el eje x corresponderá al tiempo y el eje y a las frecuencias, en escala cromática se grafica la densidad de potencia espectral en dB/Hz, en la figura 5, muestra la fórmula (DPS) mediante la cual la función periodogram de MATLAB la grafica. Si se busca realizar un gráfico 3D, en el eje z se encontraría la amplitud, en el eje x el tiempo, en el eje y la frecuencia y en escala cromática la DPS. [7]. Utilizando todas estas herramientas se realizará el análisis de diferentes señales de audio.

## II. Metodología

Para realizar la práctica, se utilizó un micrófono externo conectado mediante USB a la computadora y el programa MATLAB con las

librerías necesarias. El programa en MATLAB es uno solo, anexo 1, en el cual escoges qué herramienta del programa utilizar, figura 6, cada herramienta corresponde a una etapa del análisis de señales de audio, siendo la primera opción el grabado y visualización de la señal, anexo 2; luego está la herramienta de construcción del espectrograma de audio, anexo 3; el tercero la construcción de los sonogramas 2D y 3D de la señal, anexo 4; la cuarta corresponde a la herramienta de construcción del sonograma en tiempo real, anexo 5; la quinta es la herramienta de reconocimiento de nota, anexo 6; y la última la herramienta de reconocimiento de hablante, anexo 7.

```
Hola, este es el programa de practicas de audio fundamentales para empezar a entender las señales.
Por favor selecciona un una opcion
Grabar audio == 1
Realizar el espectrograma de un audio == 2
Construccion del sonograma == 3
Iniciar sonogram en tiempo real ==4
Iniciar detector de nota musical ==5
Iniciar detector de hablante ==6
Salir == 7
Ingrese un numero (1~7):
```

Figura 6: Interfaz del programa al iniciarse, éste funciona en la consola

### I. Primera etapa: Uso del FFT.

Como primera etapa del análisis de audio, se iniciará el programa y utilizaremos la opción 1 (anexo 2); esta opción activará la grabadora.

Se grabarán señales de 5 segundos y con una frecuencia de muestreo de 8000 Hz, estos datos se ajustan en el programa y éste los guarda con un nombre que colocaremos nosotros.

Se grabará una señal de voz propia, de 4 instrumentos y de 2 diapasones. Para cada señal se graficarán sus espectrogramas de frecuencias (anexo 3), y se observarán sus frecuencias características o timbres.

Tanto las señales de los diapasones como de los instrumentos fueron generadas por aplicaciones de móvil, entre estas: Trompeta Simulator, Clarinete Slim, Flauta Melodiosa, Perfect Piano y Tuner - Pitched.

### II. Segunda etapa: Construcción del sonograma.

Como ya se tienen todas las señales grabadas, utilizaremos la 3ra opción de nuestro pro-

grama(anexo 4). Escogeremos un archivo de cada categoría que grabamos y el programa creará su sonograma 2D y 3D; éstos se pueden guardar con el mismo nombre del archivo .wav.

Para una visualización en tiempo real, utilizaremos la opción 4 de nuestro programa (anexo 5). Al seleccionar esta opción, se iniciará el sonograma 2D en tiempo real. Una vez que se haya definido la frecuencia de muestreo y el tiempo de grabación, el sonograma se actualizará cada medio segundo, mostrando los cambios en la señal en tiempo real.

## III. Tercera etapa: Aplicaciones.

### I. Detector de nota musical

Una de las aplicaciones de todo este análisis será el identificar qué nota musical se está grabando.

Una de las aplicaciones de este análisis es la identificación de la nota musical que se está grabando.

El programa corresponde a la opción 5 (anexo 6), y requiere que se especifique una frecuencia de muestreo. Al iniciar, generará un mensaje en la consola cada segundo, indicando la frecuencia detectada y la nota musical correspondiente. El rango de identificación va desde C4 (DO - 261.63 Hz) hasta B (493.88 Hz). Si la frecuencia detectada supera los 493.88 Hz, el programa simplemente indicará que la nota es B, y si detecta frecuencias menores a 261.63, el programa indicará que la nota es C.

Para probar el funcionamiento del programa, se utilizará la aplicación de diapasón Tuner - Pitched.

### II. Detector de hablante

Este es el último programa, correspondiente a la opción 6 (anexo 7). Para su funcionamiento, es necesario especificar una frecuencia de muestreo.

Al iniciar, el programa solicita a tres personas que hablen durante **tres segundos** cada una. A partir de estas grabaciones, se generan y guardan las imágenes de los sonogramas co-

rrespondientes. Posteriormente, el programa de reconocimiento se activa.

Durante la ejecución, se muestran dos gráficos:

1. **El sonograma en tiempo real**, que representa las características espectrales de la voz.
2. **Un gráfico de identificación**, que indica qué hablante está hablando en cada momento.

Cuando se presiona Enter, la grabación finaliza y el programa pregunta si se desea guardar el gráfico de la pantalla de reconocimiento.

### III. Análisis de datos

#### I. Primera etapa

Los espectrogramas de las 7 señales analizadas son:

##### I. Espectrogramas de audio

Se grabó a dos personas, un varón y una mujer.

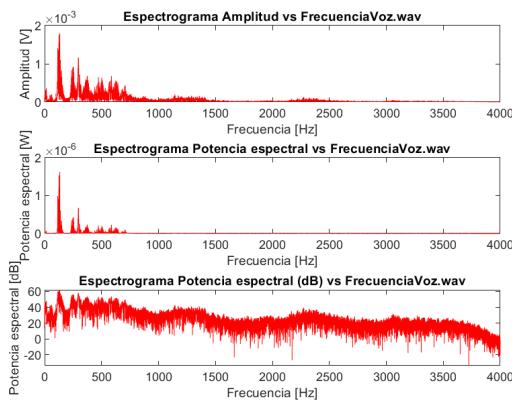


Figura 7: Espectrograma de una señal de voz, se notan ciertos picos por las frecuencias bajas, esto nos indica que la voz es de varón.

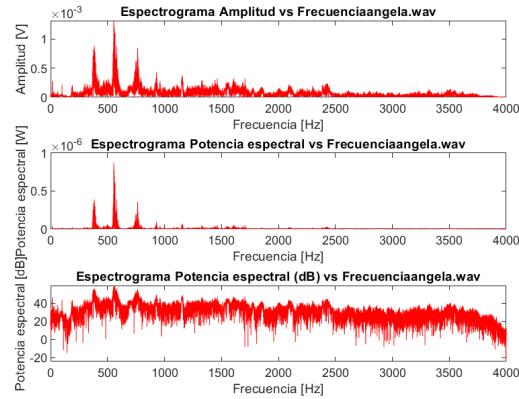


Figura 8: Espectrograma de una señal de voz, ahora los picos se encuentran en frecuencias más elevadas, lo que indica que la señal de audio es de una mujer.

Las figuras 7 y 8 muestran los espectrogramas de señales de voz correspondientes a diferentes hablantes. En ambos casos, se pueden observar los timbres característicos de cada voz, lo que permite identificar si el hablante es un hombre o una mujer al analizar el rango de frecuencias predominante. Generalmente, las voces masculinas presentan frecuencias fundamentales más bajas (alrededor de 85-180 Hz), mientras que las voces femeninas suelen ubicarse en un rango más alto (aproximadamente 165-255 Hz).

Además, al comparar los tres espectrogramas de cada hablante, podemos notar que la magnitud que más contribuye a la identificación del timbre de voz es la potencia espectral. Esto se debe a que la representación en amplitud puede inducir confusión al mezclar el timbre con posibles ruidos presentes en la señal. Por otro lado, cuando se utiliza la escala de decibeles, la información sobre el timbre se vuelve menos evidente, ya que esta escala enfatiza la consistencia en la intensidad de la señal a lo largo del tiempo, en lugar de resaltar diferencias tonales entre hablantes.

#### II. Espectrogramas de instrumentos

Para la grabación de audio, se tocó la escala musical en cada instrumento.

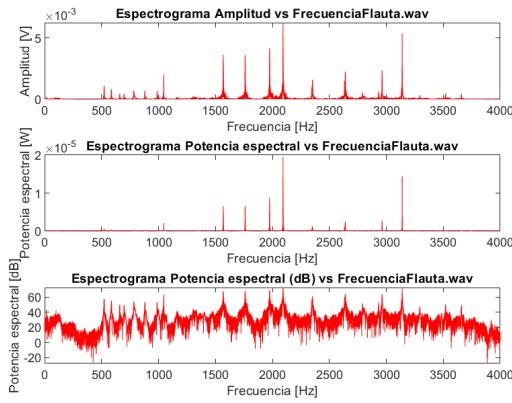


Figura 9: Espectrograma de la escala musical reproducida en una flauta.

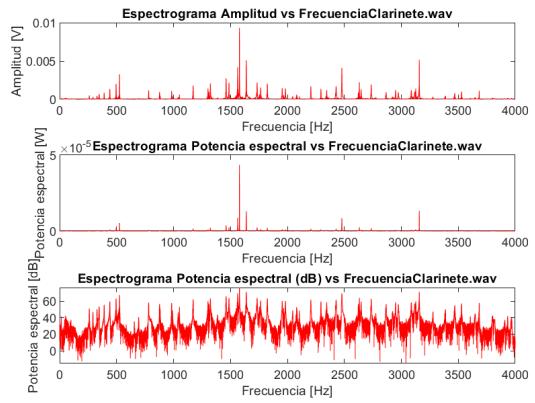


Figura 12: Espectrograma de la escala musical reproducida en un clarinete.

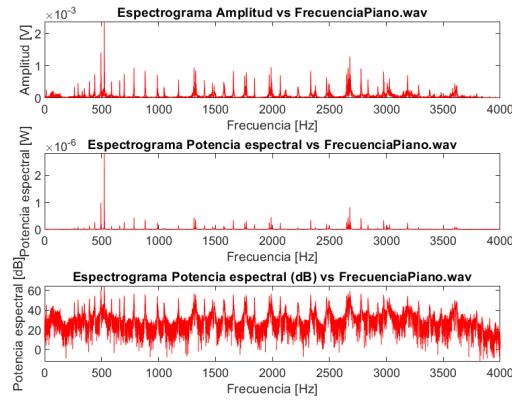


Figura 10: Espectrograma de la escala musical reproducida en un piano.

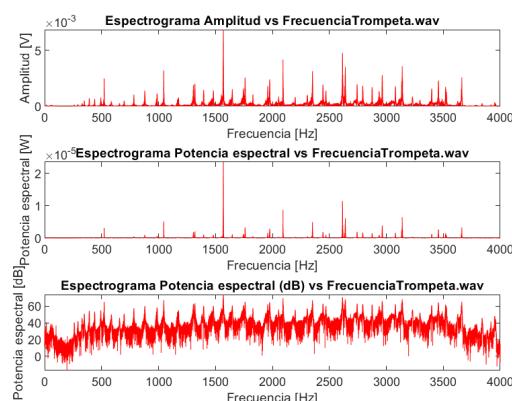


Figura 11: Espectrograma de la escala musical reproducida en una trompeta.

En las figuras 9, 10, 11 y 12, se pueden distinguir los conjuntos de frecuencias correspondientes a las notas de la escala musical en las frecuencias bajas. Sin embargo, su amplitud no es muy pronunciada en la mayoría de los instrumentos, con la excepción del piano, donde se observan con mayor claridad.

A pesar de que en cada caso se ha tocado únicamente el primer armónico de cada nota, los espectrogramas revelan la presencia de todos los armónicos disponibles hasta la frecuencia de Nyquist (en nuestro caso, 4000 Hz). Además, cada instrumento presenta un conjunto único de frecuencias características, lo que define su timbre y permite diferenciarlos auditivamente.

Al igual que en el análisis de la voz humana, la potencia espectral es la magnitud que resalta el timbre de cada instrumento, proporcionando una representación más precisa de sus características tonales.

Por otro lado, en la escala de decibeles, se pueden notar ciertos picos que coinciden en gran medida con los máximos y mínimos observados en el espectrograma de amplitud. Esto indica que, si bien la escala de decibeles ayuda a visualizar la consistencia de la señal en términos de intensidad, la potencia espectral sigue siendo la mejor herramienta para identificar y analizar las diferencias tonales entre los instrumentos.

### III. Espectrogramas de diapasones

Solamente se grabó a 2 diapasones, tocando su nota con cierta frecuencia.

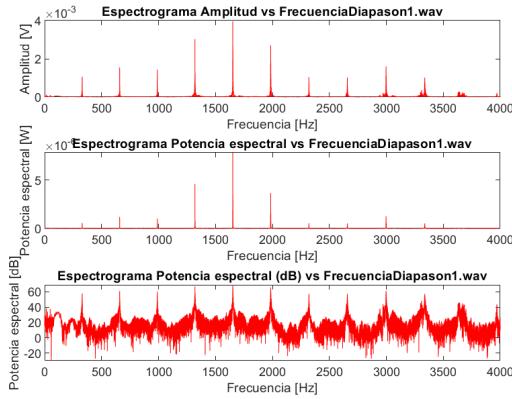


Figura 13: Espectrograma del primer diapasón, de 330 Hz, en el gráfico generado se muestra 329.4 Hz

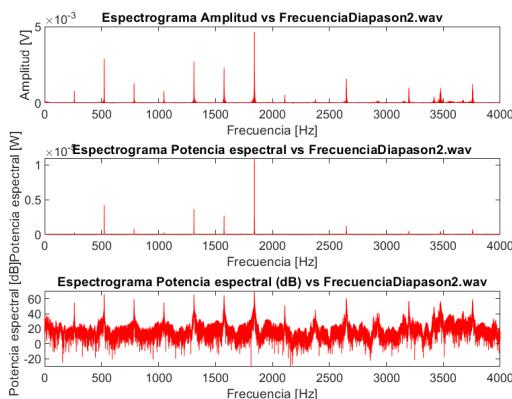


Figura 14: Espectrograma del primer diapasón, de 260 Hz, en el gráfico generado se muestra 261.4 Hz

En las figuras 13 y 14, se pueden observar nuevamente todos los armónicos disponibles, junto con las frecuencias características, que en ambos casos están entre 1500 y 2000, sugiriéndonos características del propio diapasón en la generación de señales.

Como era de esperarse, los principales picos en el espectrograma no son completamente precisos en su mayoría. Esto se debe a que, durante

el análisis, aunque la grabación se realizó exclusivamente con el sonido del diapasón, minimizando al máximo la presencia de ruido externo, la Transformada Rápida de Fourier (FFT) no logró identificar con total exactitud la frecuencia fundamental del diapasón.

Sin embargo, el error en la estimación de la frecuencia es muy pequeño, inferior al 1% en ambos casos, lo que indica que la desviación es mínima y no afecta significativamente la interpretación de los resultados. Este pequeño margen de error puede atribuirse a factores como la resoluciónpectral de la FFT, el muestreo de la señal y posibles fluctuaciones en la estabilidad de la vibración del diapasón.

## II. Segunda etapa

Los sonogramas 2D y 3D de algunas señales analizadas en la primera etapa son:

### I. Señal de voz

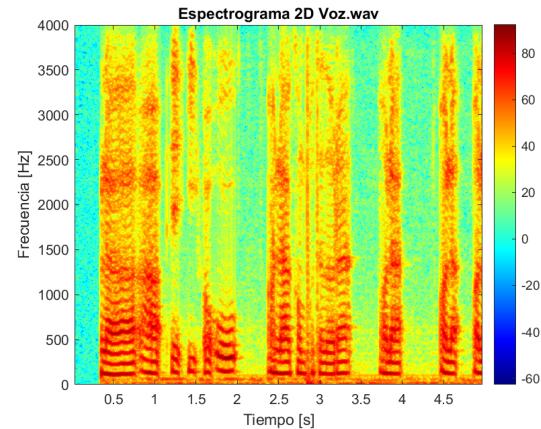


Figura 15: Sonograma 2D de la señal de voz de la figura 7.

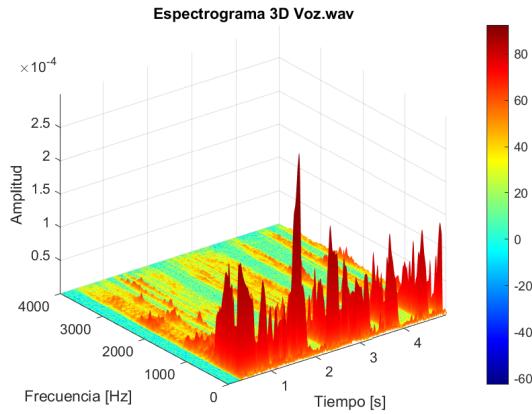


Figura 16: Sonograma 2D de la señal de voz de la figura 7.

Las figuras 15 y 16, son otra representación gráfica de la señal de audio. En éstas podemos notar características que no eran muy claras en el espectrograma. Una característica, es la variedad de armónicos que contiene la voz humana, figura 15. En la figura 16, también vemos que la mayor parte de la energía de la voz, se encuentran en las frecuencias bajas, comportamiento usual en las voces humanas.

En figura 16, también se pueden observar picos que no se observaban en los espectrogramas de la figura 7. Estos picos pueden corresponder a resonancias específicas de la voz del hablante, lo que aporta información sobre las características de la voz del hablante.

## II. Señal de Piano

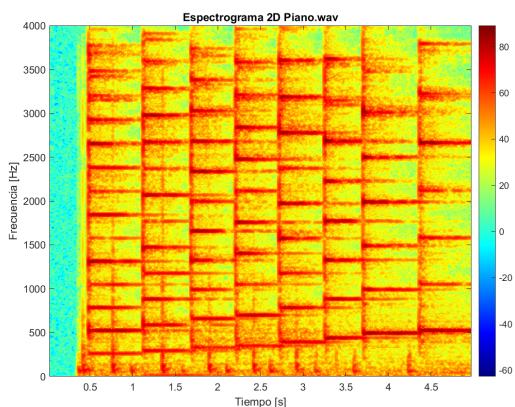


Figura 17: Sonograma 2D de la señal del piano.

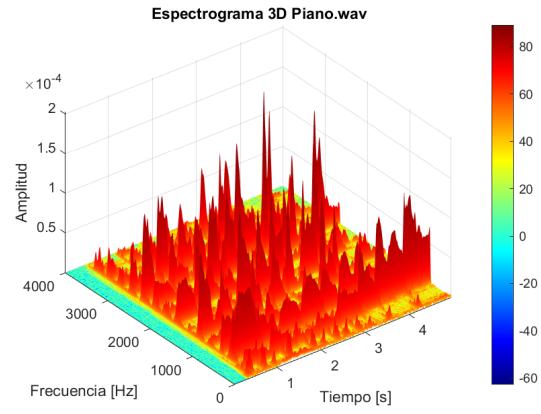


Figura 18: Sonograma 3D de la señal del piano

Para la señal del piano, ya en el espectrograma observábamos una cantidad considerable de armónicos que eran visibles, en la figura 17, se observan nuevamente todos los armónicos disponibles para el piano, y esta vez se ve más claramente la escala musical en cada armónico. En la figura 18 observamos lo mismo, una variedad de picos que representan todos los armónicos disponibles para el piano. Esta visualización resalta también como algunos armónicos se mantienen a lo largo de la señal (ejemplo, la frecuencia 2750 aprox.), mientras que la mayoría aparece cuando se toca algún sonido específico.

## III. Sonograma en tiempo real

se grabó y visualizó en tiempo real una serie de obras famosas en la música, para observar su sonograma.

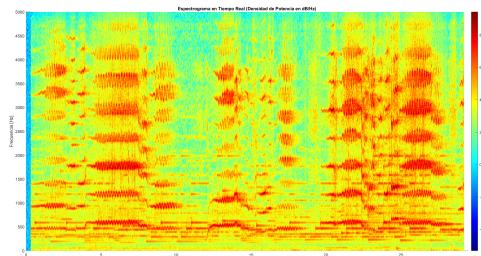


Figura 19: Sonograma de un extracto de la canción Ave María.

La canción Ave María es una expresión de fe y amor hacia Dios y la madre de Cristo que simboliza el comienzo de la vida matrimonial como

familia cristiana [1]. Se trata de una interpretación vocal con estructura armónica bien definida, lo cual se puede apreciar en la figura 19. Además, destacan bastante las transiciones de notas, se muestra una dinámica suave y fluida a lo largo de toda la obra. En el sonograma se observa también los patrones característicos de la obra, por ejemplo, en la primera parte la canción dice ".Ave María", en el sonograma se ven cierto bloque de frecuencias que corresponden a este verso; luego de diferentes versos en latín, vuelve a decirlo, y en la parte final del sonograma, se vuelve a ver un patrón similar.

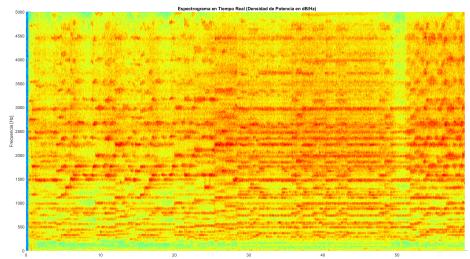


Figura 20: Sonograma de un extracto de la obra  
El lago de los Cisnes.

Como segunda obra analizada, está la de El lago de los Cisnes, es un cuento de hadas-ballet estructurado en cuatro actos, la música fue compuesta por Piotr Ilich, [2], siendo ésta un Opus. Al tratarse de una obra instrumental, en la figura 20, se observan una gran capa de frecuencias a lo largo de toda la obra. Se observa también, ciertas frecuencias características, presentes a lo largo del fragmento, debido a los oboes y violines que representan en su mayoría la obra.

Cuando existen dispersiones en las frecuencias, indican un cambio en la orquestación, esto se muestra por el segundo 20, que es cuando empiezan a incorporarse otros instrumentos a la obra.

### III. Tercera etapa

## I. Detector de nota

Para evaluar el rendimiento del detector de notas, se analizó su respuesta grabando señales

de un diapasón a diferentes frecuencias, para observar que tan preciso es. Además, se examinaron sus alcances y limitaciones, identificando los casos en los que el detector funciona con mayor exactitud y aquellos en los que puede presentar imprecisiones.

Command Window

Figura 21: Primer grabado de señales, se reprodujo algunas notas de la escala musical.

En la figura 21, se observa el funcionamiento

del programa, indica la frecuencia que detecta e identifica la nota que representa. Se observa que el programa es capaz de identificar con precisión la nota con respecto a su frecuencia, incluyendo los naturales y los sostenidos. Que el programa pueda detectar inclusive los sostenidos, muestra una considerable sensibilidad a las frecuencias, dándole la capacidad de ayudar en la afinación de instrumentos.

```
Frecuencia: 68.36 Hz - Nota: C
Frecuencia: 68.36 Hz - Nota: C
Frecuencia: 50.78 Hz - Nota: C
Frecuencia: 3992.19 Hz - Nota: B
Frecuencia: 3992.19 Hz - Nota: B
Frecuencia: 3990.23 Hz - Nota: B
Frecuencia: 3990.23 Hz - Nota: B
Frecuencia: 3990.23 Hz - Nota: B
Frecuencia: 3992.19 Hz - Nota: B
Frecuencia: 3990.23 Hz - Nota: B
Frecuencia: 3988.28 Hz - Nota: B
Frecuencia: 3986.33 Hz - Nota: B
Frecuencia: 398.44 Hz - Nota: G
Frecuencia: 97.66 Hz - Nota: C
Frecuencia: 50.78 Hz - Nota: C
Frecuencia: 50.78 Hz - Nota: C
Frecuencia: 125.00 Hz - Nota: C
Frecuencia: 125.00 Hz - Nota: C
Frecuencia: 125.00 Hz - Nota: C
Frecuencia: 62.50 Hz - Nota: C
Frecuencia: 126.95 Hz - Nota: C
Frecuencia: 126.95 Hz - Nota: C
Frecuencia: 126.95 Hz - Nota: C
Frecuencia: 123.05 Hz - Nota: C
Frecuencia: 121.09 Hz - Nota: C
Frecuencia: 119.14 Hz - Nota: C
Frecuencia: 13.67 Hz - Nota: C
Frecuencia: 115.23 Hz - Nota: C
Frecuencia: 115.23 Hz - Nota: C
Frecuencia: 115.23 Hz - Nota: C
Frecuencia: 50.78 Hz - Nota: C
```

Figura 22: Tercer grabado de señales, se busca identificar el mínimo y máximo detectado por el programa.

Por último, se identificó la frecuencia máxima y mínima que puede detectar el programa, diremos que llegó al máximo o mínimo cuando el programa muestre de manera consistente la frecuencia límite, y no presente muchas variacio-

nes. El diapasón generó frecuencias que se iba aumentando de a poco, para ver desde qué frecuencia empezaba a dejar de identificar la señal. De la figura 22 se observan como frecuencia máxima la de 3990 Hz y como mínima la de 126 Hz. Como se había mencionado en la teoría, cuando se detecten frecuencias muy bajas o muy altas, fuera del rango de la escala natural, el programa las identificará como C o B.

## II. Detector de hablante

Como última aplicación, se tiene la identificación de hablante/instrumento, se grabaron 3 señales iniciales y en tiempo real, el programa identifica quién está hablando o qué instrumento se está tocando.

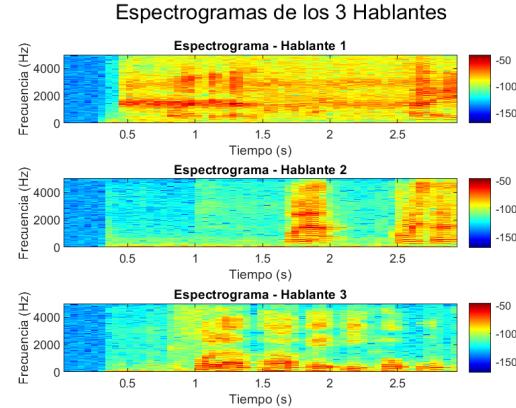


Figura 23: Sonogramas de los hablantes, en este caso, personas.

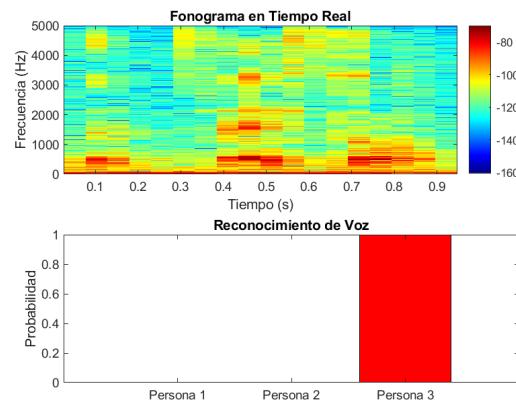


Figura 24: Demostración gráfica de cómo el programa indica quien está hablando.

Una vez iniciado, el programa de reconocimiento de voz compara en tiempo real las frecuencias observadas en el sonograma de la figura 24 con las frecuencias previamente registradas al inicio, como se muestra en la figura 23. A partir de esta comparación, el programa identifica qué persona está hablando en cada momento. En este caso, se observa en la figura 24 que las frecuencias que marca son bajas, al contrastarlas con las frecuencias características en la figura 23, el programa marca que el hablante 3 está hablando. El programa calcula la distancia entre la frecuencia grabada con los centroides espectrales de cada hablante y cuando esta distancia es mínima, nos indica el hablante.

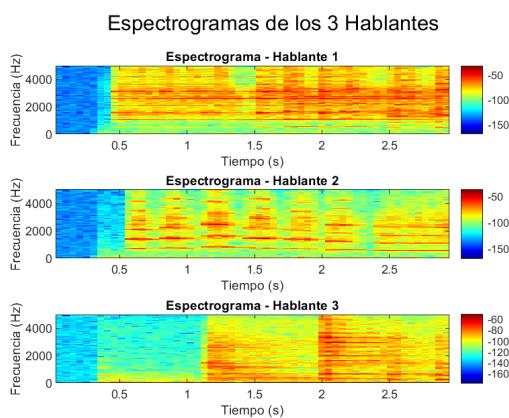


Figura 25: Sonogramas de los hablantes, en este caso, 3 instrumentos, trompeta, flauta y guitarra.

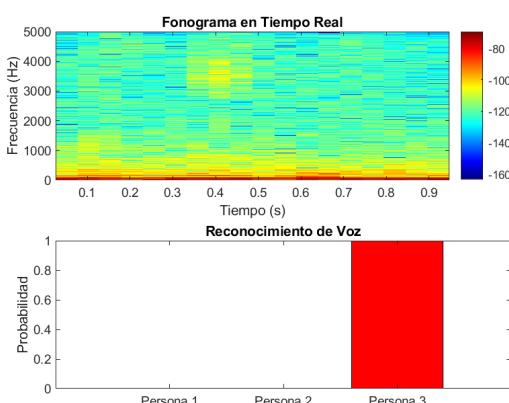


Figura 26: Demostración gráfica de cómo el programa indica quién está hablando.

En la figura 25, se muestran los patrones característicos de los armónicos de cada instrumento. En la segunda figura, 26, se puede observar el espectrograma donde se destaca una frecuencia baja. El instrumento que presenta armónicos predominantemente bajos es el tercero, la guitarra. El programa identifica correctamente que el sonido corresponde a la guitarra.

Nuevamente el programa calcula la distancia entre los centroides espectrales de cada instrumento y la frecuencia detectada e identifica muy bien qué instrumento se está tocando.

#### IV. Conclusiones

Este estudio básico de señales utilizando la FFT nos muestra la gran utilidad de esta herramienta. Al momento del análisis, fue muy útil el uso del software MATLAB, ya que este contaba ya con la FFT en una librería, como se mencionó en el resumen teórico.

Todos los análisis realizados mostraron resultados satisfactorios. El grabado y la construcción de los espectrogramas y sonogramas 2D y 3D nos proporcionan mucha información de la señal grabada, además de la visualización de los armónicos, tanto en la voz como en los instrumentos. Los tres programas, anexos 2, 3 y 4, ya se encuentran bastante optimizados. En los graficadores no se requiere introducir una frecuencia de muestreo, pues el mismo programa toma el dato del archivo .wav, por lo que no se sugieren mejoras relevantes.

El programa que genera el sonograma en tiempo real, descrito en el anexo 5, tiene una frecuencia de muestreo de 10000 Hz. Aunque es posible aumentarla, esto reduciría la resolución, ya que, aunque la FFT pueda graficar hasta la frecuencia de Nyquist (10000 Hz), la resolución en el eje 'y' disminuiría considerablemente. Por lo tanto, se considera que una frecuencia de muestreo de 10000 Hz es adecuada para el análisis de voces y canciones, ya que permite una visualización clara de los armónicos y su densidad de potencia espectral.

El constructor de sonograma 2D en tiempo real solamente pide el tiempo de grabación. Por

las mismas razones de resolución de frecuencias, se tiene la frecuencia de muestreo predeterminada de 10000 Hz. El programa, a pesar de que funciona bastante bien, se puede optimizar y mejorar en varios aspectos, como, por ejemplo, la interactividad del gráfico, el cálculo de los tiempos sin usar "tic", o "toc", la incorporación de buffers de audio, el escalado del gráfico para trabajar con señales más largas (ya que, al momento de observar la señal, los gráficos se van comprimiendo y se reduce la resolución), y el guardado de la señal completa, no solo del sonograma.

El detector de notas termina siendo funcional, ya que recibe e identifica muy bien las frecuencias y la nota correspondiente, aunque, como ya se mencionó, solamente identifica las notas de la escala natural, mas no sus armónicos. Las mejoras posibles en el código pueden ser una mejor optimización y un gráfico visual para observar las frecuencias representativas.

El detector de hablante, a pesar de que funciona, su técnica de detección no es de las mejores. El código calcula el centroide espectral y compara la frecuencia detectada en el momento con cada centroide, y nos indica con qué centroide existe un mínimo. Una mejora considerable al código sería la implementación de los coeficientes MFCC (Mel Frequency Cepstral Coefficients), coeficientes utilizados en los detectores de hablantes más sofisticados. Se decidió no utilizarlos debido a la insuficiente teoría desarrollada.

El análisis de señales es un tópico muy amplio y especializado que requiere un estudio muy detallado. En la práctica de laboratorio se realizó un estudio muy básico e introductorio a este tema. Cada etapa de la práctica siguió un procedimiento típico en el análisis de señales: primero se tomaron y visualizaron las señales, luego se

realizó un tratamiento a los datos, en este caso, con la FFT; posteriormente se analizaron los resultados y, finalmente, se exploraron posibles aplicaciones. Este trabajo sentó las bases para una comprensión más profunda del análisis de señales. Se pretende aplicar el conocimiento adquirido posteriormente en los sistemas de control.

## Referencias

- [1] Ave maría (wikipedia). Accedido: 2021-04-07.
- [2] El lago de los cisnes (wikipedia). Accedido: 2021-04-07.
- [3] Eduard Bertran Alberí. *Procesado digital de señales. Fundamentos para comunicaciones y control - 1*. Edicions UPC, 2006.
- [4] Camilo C. *Fundamentos del análisis de Fourier*. Universidad de Vigo, 2003.
- [5] Francisco C. and Juan José D. *Teoría y técnica del sonido*. Editorial Sintesis, no disponible.
- [6] Edisan G. Procesamiento de audio en tiempo real con el dsp tms320c6711. *Instituto de Ciencias Básicas e Ingeniería*, 2007.
- [7] MathWorks. *MATLAB Documentation*, 2021. Accedido: 2021-04-07.
- [8] Pablo Alvarado Moya. *Señales y sistemas. Fundamentos matemáticos*. CDMB, 2008.
- [9] Harris Benson Sears and Mark W. Zemansky. *Física universitaria*. McGraw-Hill, México, 13 edition, 2011.

## V. Anexos

### A. Código en MATLAB Principal

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3 %%Práctica 1. Análisis de Audio con FFT

```

```

5 %%Código principal
6 clear all;
7 close all;
8 clc;
9 %Este es el programa que combinara todos los codigos, grabar voz, hacer
10 %sonograma o espectrograma de cualquier archivo e iniciar el sonograma en
11 %tiempo real
12 disp("Hola, este es el programa de practicas de audio fundamentales para
    empezar a entender las señales.");
13 disp("Por favor selecciona un una opcion");
14 disp("Grabar audio == 1");
15 disp("Realizar el espectrograma de un audio == 2");
16 disp("Construccion del sonograma == 3");
17 disp("Iniciar sonograma en tiempo real ==4");
18 disp("Iniciar detector de nota musical ==5");
19 disp("Iniciar detector de hablante ==6");
20 disp("Salir == 7");
21
22
23 while true
24     % Solicitar la opción al usuario
25     opcion = input('Ingrese un número (1-7):\n');
26
27     % Validar que la opción es un número entre 1 y 7
28     if isnumeric(opcion) && isscalar(opcion) && opcion >= 1 && opcion <= 7
29         switch opcion
30             case 1
31                 disp("Escogió la grabadora de audio\n");
32
33                 % Validar el tiempo de grabación
34                 tiempo = input("Ingrese el tiempo de grabación (segundos): \n");
35                 ;
36                 if ~isnumeric(tiempo) || tiempo <= 0
37                     fprintf('Error: Debes ingresar un número válido para el
                        tiempo.\n');
38                     continue;
39             end
40
41             % Validar la frecuencia de muestreo
42             FrecS = input("Ingrese la frecuencia de muestreo (Hz): \n");
43             if ~isnumeric(FrecS) || FrecS <= 0
44                 fprintf('Error: Debes ingresar un número válido para la
                    frecuencia de muestreo.\n');
45                 continue;
46             end
47
48             % Ejecutar la función
49             Anexo1Grabadora(tiempo, FrecS);
50
51             case 2
52                 disp("Escogió realizar el espectrograma.\n");
53                 file = input("Ingrese el nombre del audio que desea analizar:
                    ", 's');
54                 Espectrograma(file);
55
56             case 3
57                 disp("Escogió realizar el sonograma.");

```

```

57         files = input("Ingrese el nombre del audio que desea analizar
58                         :", 's');
59         Sonograma(files);
60
61     case 4
62         disp("Activó el sonograma en tiempo real.\n");
63
64         % Validar el tiempo
65         tiempo = input("Ingrese el tiempo que quiere que se realice la
66                         grabación (segundos): ");
67         if ~isnumeric(tiempo) || tiempo <= 0
68             fprintf('Error: Debes ingresar un número válido para el
69                         tiempo.');
70             continue;
71         end
72
73         SonogramaTiempoReal(tiempo);
74     case 5
75         disp("Activó el reconocimiento de nota musical");
76         freq = input("Ingrese la frecuencia de muestreo que quiere
77                         utilizar: ");
78         if ~isnumeric(freq) || freq <= 0
79             fprintf('Error: Debes ingresar un número válido para el
80                         tiempo.');
81             continue;
82         end
83         DetectordeNota(freq);
84     case 6
85         disp("Activó el reconocimiento de hablante");
86         freq = input("Ingrese la frecuencia de muestreo que quiere
87                         utilizar: ");
88         if ~isnumeric(freq) || freq <= 0
89             fprintf('Error: Debes ingresar un número válido para el
90                         tiempo.');
91             continue;
92         end
93         Conversacion(freq);
94     case 7
95         fprintf('Saliendo del programa...\n');
96         break; % Sale del bucle while
97     end
98 else
99     fprintf('Error: Debes ingresar un número válido entre 1 y 5.\n');
100 end

```

Listing 1: Código principal, desde éste se llaman a los demás códigos adjuntos.

## B. Código en MATLAB para grabar y guardar audio

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3 %%Práctica 1. Análisis de Audio con FFT
4
5 %Grabadora de Audio

```

```

7 function Anexo1Grabadora(t,Fs)
8 %Grabadora de audio
9
10 ch = 1; % Canal mono
11 NBits = 16; % Bits por muestra
12
13 % Crear el objeto de grabación
14 recObj = audiorecorder(Fs, NBits, ch);
15
16 % Iniciar grabación
17 disp('Grabando...¡hable¡ahora.');
18 recordblocking(recObj, t);
19 disp('Grabación¡finalizada.');
20
21 % Obtener los datos de audio
22 YW = getaudiodata(recObj);
23
24 filexxx = input("Ingrese el nombre del archivo con extension .wav: ", 's');
25 audiowrite(filexxx, YW, Fs);
26 disp(["Se creo el archivo,", filexxx]);
27
28 % Reproducir el audio grabado
29 sound(YW, Fs);
30
31 % Graficar los datos de la señal grabada
32 plot(YW);
33 title(["Señal de audio ",filexxx]);
34 drawnow;
35 Nombredx = strcat('Sonograma_', erase(filexxx, '.wav'), '.png');
36 %% Guardar Graficos
37 respuesta = input(' Desea ¡guardar¡el¡gráfico?¡(S/N):¡', 's');
38 if upper(respuesta) == 'S'
39     saveas(gcf, Nombredx);
40     disp(['Gráfico¡guardado¡como:¡', Nombredx]);
41 else
42     disp('Gráfico¡no¡guardado.');
43 end
44 close(gcf);
45 end

```

Listing 2: Código que graba y grafica la señal de audio, se escoge la frecuencia de muestreo y el tiempo de grabado.

### C. Código en MATLAB para la construcción del espectrograma

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3 %%Práctica 1. Análisis de Audio con FFT
4
5 %Constructora del espectrograma
6
7
8
9 function Espectrograma(filexxx)
10    [YW, Fs] = audioread(filexxx); % Leer datos de audio y frecuencia de
11                                % muestreo

```

```

11 % Aplicar FFT
12 Y = fft(YW);
13 Y(1) = [];
14
15 %% Calcular la amplitud
16 N = length(Y);
17 Amp = 2*abs(Y(1:floor(N/2)))/N;
18 Poten = 0.5*Amp.^2;
19 Power = 10*log10(Poten*10.^-(12));
20 nyquist = Fs / 2; % Frecuencia de Nyquist
21 freq = (1:N/2) / (N/2) * nyquist;
22
23 %% Graficar el espectrograma con diferentes magnitudes
24
25 figure;
26 subplot(3,1,1);
27 plot(freq, Amp, 'r');
28 title(['Espectrograma de Amplitud vs Frecuencia', filexxx]);
29 xlabel('Frecuencia [Hz]');
30 ylabel('Amplitud [V]');
31
32
33 subplot(3,1,2);
34 plot(freq, Poten, 'r');
35 title(['Espectrograma de Potencia espectral vs Frecuencia', filexxx]);
36 xlabel('Frecuencia [Hz]');
37 ylabel('Potencia espectral [W]');
38
39 subplot(3,1,3);
40 plot(freq, Power, 'r');
41 title(['Espectrograma de Potencia espectral (dB) vs Frecuencia', filexxx]);
42 xlabel('Frecuencia [Hz]');
43 ylabel('Potencia espectral [dB]');
44 Nombredx = strcat('Sonograma_', erase(filexxx, '.wav'), '.png');
45 drawnow;
46
47 %% Guardar Graficos
48 respuesta = input(' Desea guardar el grafico? (S/N): ', 's');
49 if upper(respuesta) == 'S'
50     saveas(gcf, Nombredx); % gcf obtiene la figura actual
51     disp(['Gráfico guardado como:', Nombredx]);
52 else
53     disp('Gráfico no guardado.');
54 end
55 close(gcf);
56
57 end

```

Listing 3: Código que calcula la FFT y grafica los diferentes tipos de espectrogramas, solo se debe escoger el archivo .wav a graficar.

#### D. Código en MATLAB para la construcción de los sonogramas 2D y 3D

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3

```

```

4 %%Práctica 1. Análisis de Audio con FFT
5
6 %Constructora de Sonograma 2D y 3D
7
8 function Sonograma(filexxx)
9
10 [YW, Fs] = audioread(filexxx);
11 sound(YW,Fs);
12 % Parámetros para el espectrograma
13
14 N = length(YW);
15 windowSize = 512; % Tamaño de la ventana
16 overlap = round(0.9*windowSize);
17 nfft = 4096; % Tamaño de la FFT
18
19 % Calcular espectrograma
20 [S, F, T, P] = spectrogram(YW, windowSize, overlap, nfft, Fs);
21
22 % Calcular amplitud y densidad de potencia espectral
23 Amp = (1/N) * 2 * abs(S);
24 Aplog = log10(Amp);
25 Densidad_Potencia = 0.5 * Amp.^2 * N; % En W/Hz
26 Densidad_Potencia_dB = 10 * log10(Densidad_Potencia*10.^12); % En dB/Hz
27
28 %% --- Gráfico 2D ---
29 figure;
30 imagesc(T, F, Densidad_Potencia_dB);
31 axis xy;
32 colorbar;
33 colormap jet;
34 xlabel('Tiempo [s]');
35 ylabel('Frecuencia [Hz]');
36 title(['Espectrograma 2D', filexxx]);
37 nombre_2D = strcat('Espectrograma2D_', erase(filexxx, '.wav'), '.png');
38 drawnow;
39 %% --- Gráfico 3D ---
40 figure;
41 surf(T, F, Aplog, Densidad_Potencia_dB, 'EdgeColor', 'none');
42 view(3); axis tight;
43 shading interp;
44 colorbar;
45 colormap jet;
46 xlabel('Tiempo [s]');
47 ylabel('Frecuencia [Hz]');
48 zlabel('Amplitud');
49 title(['Espectrograma 3D', filexxx]);
50 nombre_3D = strcat('Espectrograma3D_', erase(filexxx, '.wav'), '.png');
51 drawnow;
52 %% Guardar Graficos
53 respuesta = input(' Desea guardar los gráficos? (S/N): ', 's');
54 if upper(respuesta) == 'S'
55     saveas(figure(1), nombre_2D);
56     saveas(figure(2), nombre_3D);
57     disp(['Gráficos guardados como:', nombre_2D, ' y ', nombre_3D]);
58 else
59     disp('Gráficos no guardados.');
60 end

```

```

61     close(gcf);
62 end

```

Listing 4: Código que grafica el sonograma 2D y 3D de una señal grabada, solamente se debe escoger el archivo .wav a graficar.

### E. Código en MATLAB para la construcción de sonograma 2D en tiempo real

```

1
2 %Sistemas de Control
3 %%Daniel Roberto Garcia Miranda
4 %%Práctica 1. Análisis de Audio con FFT
5
6 %Sonograma en tiempo real
7 function SonogramaTiempoReal(duration)
8
9
10    %% Parámetros de audio
11    Fs = 10000;    % Frecuencia de muestreo
12
13    % Tamaño de ventana para tramas de 64 ms y 128 ms
14    window_ms = 128;
15    windowSize = 512;
16    overlap = round(windowSize * 0.7);
17    nfft = 4096;  % Número de puntos en la FFT
18
19    %% Configurar grabación en tiempo real
20    recObj = audiorecorder(Fs, 16, 1);
21    disp('Iniciando grabación en tiempo real... ');
22    record(recObj); % Graba continuamente
23
24    %% Configurar figura
25    figure;
26    colormap jet;
27    colorbar;
28    xlabel('Tiempo [s]');
29    ylabel('Frecuencia [Hz]');
30    title('Espectrograma en Tiempo Real (Densidad de Potencia en dB/Hz)');
31    hold on;
32
33    %% Procesar y graficar en tiempo real
34    tic;
35    while toc < duration
36        YW = getaudiodata(recObj); % Obtener datos de audio en vivo
37
38        if length(YW) < windowSize
39            continue; % Esperar a que haya suficientes datos
40        end
41
42        % Aplicar ventana de Hamming para suavizar la señal
43        ventana = hamming(windowSize);
44
45        % Calcular espectrograma
46        [S, F, T, ~] = spectrogram(YW, ventana, overlap, nfft, Fs);
47
48        % Calcular amplitud y densidad de potencia espectral

```

```

49     Amp = (1/length(YW)) * 2 * abs(S);
50     Densidad_Potencia = 0.5 * Amp.^2 * length(YW);
51     Densidad_Potencia_dB = 10 * log10(Densidad_Potencia * 10^12);
52
53     % Actualizar gráfico}
54     T = T - min(T);
55     imagesc(T, F, Densidad_Potencia_dB);
56     axis xy;
57     xlim([max(0, toc - duration) toc]);
58     ylim([0 Fs/2]);
59     drawnow;
60 end
61
62 % Detener grabación
63 stop(recObj);
64 disp('Grabación finalizada.');
65 %% Guardar imagen
66 respuesta = input('Desea guardar la imagen del sonograma? (S/N): ', 's');
67 if upper(respuesta) == 'S'
68     nombreArchivo = input('Ingrese el nombre para guardar la imagen (sin extensión): ', 's');
69     saveas(gcf, [nombreArchivo, '.png']);
70     disp(['Imagen guardada como: ', nombreArchivo, '.png']);
71 else
72     disp('Imagen no guardada.');
73 end
74 close(gcf);
75 end

```

Listing 5: Código que grafica el sonograma 2D en tiempo real, solamente se le debe indicar cuanto durará la grabación.

## F. Código en MATLAB para el detector de nota musical

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3 %%Práctica 1. Análisis de Audio con FFT
4
5 %%Detector de notas
6 function DetectordNota(fs)
7     % Parámetros de grabación en tiempo real
8     bufferSize = 4096; % Tamaño del buffer de audio
9     duracionSegmento = bufferSize / fs; % Tiempo de cada segmento de audio
10
11    % Configurar grabador de audio en tiempo real
12    recObj = audiorecorder(fs, 16, 1);
13
14    %% Notas musicales y sus frecuencias estándar
15    notas = {'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'};
16    frecuencias = [261.63, 277.18, 293.66, 311.13, 329.63, 349.23, 369.99,
17                  392.00, 415.30, 440.00, 466.16, 493.88];
18
19    disp('Iniciando detección en tiempo real... Presiona ENTER para detener.');
20
21    %% Iniciar grabación en segundo plano
22    record(recObj);

```

```

22 pause(0.5);

23

24 % Variable de control para salir del bucle
25 stop_flag = false;

26

27 %% Hilo paralelo para detectar ENTER
28 h = figure('Name', 'Detector_de_Notas', 'NumberTitle', 'off', ...
29             'KeyPressFcn', @(src, event) setappdata(src, 'stop', true));

30

31 setappdata(h, 'stop', false);

32

33 %% Bucle de detección en tiempo real
34 while ~stop_flag
35     % Obtener el audio grabado hasta el momento
36     audioData = getaudiodata(recObj);

37

38     % Verificar si hay suficiente audio
39     if length(audioData) < bufferSize
40         pause(0.1);
41         continue;
42     end

43

44     % Tomar solo los últimos 'bufferSize' datos
45     audioData = audioData(end-bufferSize+1:end);

46

47     % Calcular FFT para encontrar la frecuencia dominante
48     fftAudio = abs(fft(audioData));
49     fftAudio = fftAudio(1:bufferSize/2); % Tomar solo la primera mitad
50     frequencies = (0:bufferSize/2-1) * (fs / bufferSize);

51

52     % Encontrar el pico de frecuencia
53     [~, index] = max(fftAudio);
54     frecuenciaDominante = frequencies(index);

55

56     % Encontrar la nota más cercana
57     [~, idxNota] = min(abs(frecuencias - frecuenciaDominante));
58     notaDetectada = notas{idxNota};

59

60     % Mostrar resultado en tiempo real
61     printf('Frecuencia: %.2f Hz - Nota: %s\n', frecuenciaDominante,
62           notaDetectada);

63

64     % Revisar si el usuario presionó ENTER
65     if getappdata(h, 'stop')
66         stop_flag = true;
67     end

68

69     % Pequeña pausa para evitar sobrecargar la CPU
70     pause(duracionSegmento);
71 end
72 disp('Detección detenida.');
73 close(h); % Cerrar ventana del detector
74 stop(recObj); % Detener grabación

```

Listing 6: Código que activa un detector de nota musical en tiempo real, solamente se le debe indicar la frecuencia de muestreo a la que trabajará.

## G. Código en MATLAB para la identificación de hablante

```

1 %Sistemas de Control
2 %%Daniel Roberto Garcia Miranda
3 %%Práctica 1. Análisis de Audio con FFT
4
5 %Detector de hablante
6 function Conversacion(Fs)
7
8     %% Configuración
9     T_duracion = 3;    % Duración de cada grabación
10    N_personas = 3;   % Número de personas a grabar
11
12    disp('---\u2014Fase\u2014de\u2014Grabación\u2014---');
13    figure;
14    for i = 1:N_personas
15        fprintf('Hablante %d, \u2014por\u2014favor\u2014hable\u2014durante\u2014%d\u2014segundos...\n', i,
16                T_duracion);
17        recObj = audiorecorder(Fs, 16, 1);
18        recordblocking(recObj, T_duracion);
19        audio_data{i} = getaudiodata(recObj);
20
21        % Extraer centroides espectrales como huella de cada hablante
22        [S, F, T, P] = spectrogram(audio_data{i}, 1024, 512, 1024, Fs, 'yaxis')
23        ;
24        centroides{i} = sum(F .* mean(abs(S), 2)) / sum(mean(abs(S), 2));
25
26        % Graficar espectrograma de cada hablante en una misma figura
27        subplot(3,1,i);
28        imagesc(T, F, 10*log10(P));
29        axis xy;
30        xlabel('Tiempo\u2014(s)');
31        ylabel('Frecuencia\u2014(Hz)');
32        title(['Espectrograma\u2014\u2014Hablante\u2014', num2str(i)]);
33        colormap jet;
34        colorbar;
35    end
36    sgttitle('Espectrogramas\u2014de\u2014los\u20143\u2014Hablantes');
37
38    % Guardar la imagen de los 3 espectrogramas
39    saveas(gcf, 'espectrogramas_hablantes.png');
40    disp('Imagen\u2014de\u2014los\u20143\u2014espectrogramas\u2014guardada\u2014como\u2014espectrogramas_hablantes
41        .png');
42    close(gcf);
43
44    disp('Grabación\u2014finalizada.');
45
46    %% Fase de reconocimiento en tiempo real
47    disp('---\u2014Iniciando\u2014reconocimiento\u2014en\u2014tiempo\u2014real\u2014---');
48    figure;
49    recObj = audiorecorder(Fs, 16, 1);
50    record(recObj);
51
52    while true
53        pause(1);  % Analizar cada 1 segundo
54        audio_actual = getaudiodata(recObj);

```

```

53
54     if length(audio_actual) < Fs
55         continue;
56    end
57
58    % Tomar el último segundo de audio
59    audio_actual = audio_actual(end-Fs+1:end);
60
61    % Calcular espectrograma en tiempo real
62    [S, F, T, P] = spectrogram(audio_actual, 1024, 512, 1024, Fs, 'yaxis');
63    centroid_actual = sum(F .* mean(abs(S), 2)) / sum(mean(abs(S), 2));
64
65    % Comparar con los centroides grabados
66    distancias = cellfun(@(x) abs(x - centroid_actual), centroides);
67    [~, hablante_detectado] = min(distancias);
68
69    % Mostrar espectrograma en tiempo real
70    subplot(2,1,1);
71    imagesc(T, F, 10*log10(P));
72    axis xy;
73    xlabel('Tiempo (s)');
74    ylabel('Frecuencia (Hz)');
75    title('Fonograma en Tiempo Real');
76    colormap jet;
77    colorbar;
78
79    % Mostrar quién está hablando
80    subplot(2,1,2);
81    bar(hablante_detectado, 1, 'r');
82    xlim([0 N_personas+1]);
83    xticks(1:N_personas);
84    xticklabels({'Persona 1', 'Persona 2', 'Persona 3'});
85    ylabel('Probabilidad');
86    title('Reconocimiento de Voz');
87    drawnow;
88
89    % Comprobar si se presiona Enter para salir
90    if ~isempty(get(gcf, 'CurrentCharacter')) && get(gcf, 'CurrentCharacter')
91        ' == 13
92        disp('Finalizando programa... ');
93        break;
94    end
95    end
96    %% Preguntar si se desea guardar la imagen del espectrograma en tiempo real
97    while true
98        guardar = input(' Desea guardar el espectrograma en tiempo real? (S/N)
99        : ', 's');
100       if lower(guardar) == 's'
101           nombreImagen = input('Ingrese el nombre para guardar el
102           espectrograma en tiempo real (sin extensión): ', 's');
103           if isempty(nombreImagen)
104               nombreImagen = 'espectrograma_tiempo_real'; % Nombre por
105               defecto
106           end
107           saveas(gcf, [nombreImagen, '.png']);
108           disp(['Espectrograma guardado como ', nombreImagen, '.png']);
109           break;
110       end
111   end

```

```
106     elseif lower(guardar) == 'n'  
107         disp('No se guardó el espectrograma en tiempo real.');//  
108         break;  
109     else  
110         disp('Respuesta no válida. Ingrese S o N.');//  
111     end  
112 end  
113 close(gcf);  
114
```

Listing 7: Código que activa la detección de hablante, se le debe indicar la frecuencia de muestreo para la FFT.