Laboratorio No 1. Ecualizador de audio

Juan Sebastián Carrillo Quiñones
Universidad Sergio Arboleda
Bogotá, Colombia
juan.carrillo2@correo.usa.edu.co

Nicolás Cifuentes Barriga
Universidad Sergio Arboleda
Bogotá, Colombia
nicolas.cifuentes01@correo.usa.edu.co

Juan Camilo Peña Espinosa
Universidad Sergio Arboleda
Bogotá, Colombia
juan.pena2@correo.usa.edu.co

En el presente laboratorio se creará un ecualizador de audio el cual podrá leer archivos de extensión .WAV, por medio de un menú se adelantara y retrasara la grabación en un segundo, se integra la transformada rápida de Fourier para realizar un análisis en frecuencia y se podrá visualizar la misma en la consola.

I. Introducción

Ante todo, Un archivo de audio .WAV Es un formato de audio utilizado en computadoras que permite guardar señales de audio sin pérdida de calidad, los archivos .WAV se utilizan a menudo en la producción de audio profesional debido a su alta calidad de sonido y capacidad para almacenar audio en una variedad de tasas de muestreo y profundidades de bits.Sin embargo, los archivos .WAV también son más grandes que otros formatos de audio, como MP3 o AAC, por lo que no son tan convenientes para el almacenamiento y la transmisión de audio en línea. [1]. La Transformada Rápida de Fourier (FFT) es un algoritmo numérico que permite calcular la Transformada de Fourier Discreta (DFT) de una señal discreta en tiempo. La DFT es una representación matemática de una señal en el dominio de la frecuencia, lo que permite analizar la distribución de frecuencias en una señal de tiempo. [2] La FFT es un método más eficiente para calcular la DFT en comparación con los algoritmos clásicos, ya que utiliza una estrategia de dividir y conquistar para reducir el número de operaciones necesarias. Esto la hace adecuada para el procesamiento en tiempo real de señales y para la implementación en hardware y software. Además de la aplicación en el análisis de señales, la FFT también tiene aplicaciones en áreas como procesamiento de imágenes, la bioinformática, la economía, la física, la ingeniería y la criptografía [3].

II. Resultados

Partiendo desde la compresión del archivo junto con su distribución de bytes, se realiza la lectura de un archivo de audio con extensión .WAV, para la cual fue necesaria tomar en cuenta los ”Headers”.

Se elaboró una función “menu” para facilitar el uso de las opciones que se integró en el ecualizador, para manejar el tiempo de reproducción del audio y una opción para salir del programa.

Teniendo en cuenta esto, se procede a crear la función “size” para ver el tamaño del archivo en bytes, este valor se utilizará para poder crear el arreglo que va a contener los datos de sonido del archivo. Con el fin de poder verificar el valor que contiene cada uno de los bytes que se están utilizando, se elaboró la función ”printbytes”, con ayuda de una estructura la cual cuenta con diferentes figuras/size.pngime en consola cada

WAV Header Synopsis:

Description	Size (Bytes)	Usual Contents
RIFF file description header	4 bytes	The ASCII text string "RIFF"
size of file	4 bytes	The file size LESS the size of the "RIFF" description (4 bytes) and the size of file description (4 bytes).
The WAV description header	4 bytes	The ascii text string "WAVE".
fmt description header	4 bytes	The ascii text string "fmt " (note the trailing space).
size of WAV section chunk	4 bytes	The size of the WAV type format (2 bytes) + mono/stereo flag (2 bytes) + sample rate (4 bytes) + bytes/sec (4 bytes) + block alignment (2 bytes) + bits/sample (2 bytes). This is usually 16.
WAV type format	2 bytes	Type of WAV format. This is a PCM header, or a value of 0x01.
mono/stereo flag	2 bytes	mono (0x01) or stereo (0x02)
sample frequency	4 bytes	The sample frequency.
bytes/sec	4 bytes	The audio data rate in bytes/sec.
block alignment	2 bytes	The block alignment.
bits per sample	2 bytes	The number of bits per sample.
data description header	4 bytes	The ascii text string "data".
size of the data chunk	4 bytes	Number of bytes of data is included in the data section.
data	variable length	The audio data.

Figure 1. Distribución de bytes extensión .WAV..

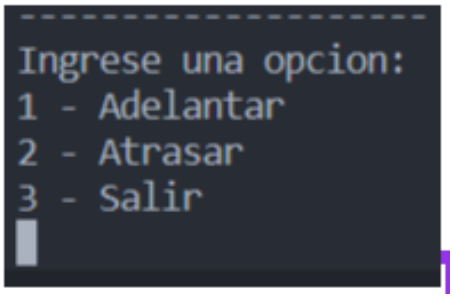

Figure 2. Menú principal del programa

uno de los valores correspondientes a su respectivo ”Headers”.

```
Nombre del archivo: 2.wav
Bytes: 88108
```

Figure 3. Resultado de la función size

```
Riff: RIFF
Size of file: 88100
Wave: WAVE
Fmt: fmt
Chunk: 16
Format type: 1
Channel: 1
Sample frequency: 44100
Bytes per second: 88200
Block alingment: 2
Bits per sample: 16
Data: data
Data chunk: 88064
```

Figure 4. Resultado de la función print bytes

Una vez validada la correcta lectura de valores, con ayuda de la aplicación “HxD”, se desarrolló la función “loaddata” la cual obtiene la información de sonido y los almacena en el arreglo previamente creado. En la imagen 4 se puede apreciar tanto los datos de sonido que se encuentran dentro del archivo como el array en el cual se están guardando cada uno de ellos.

```
Data of byte: -76
Data of byte: -63
Data of byte: -58
Data of byte: -66
Data of byte: -53
Data of byte: -48
Array in 44: -76
Array in 45: -63
Array in 46: -58
Array in 47: -66
Array in 48: -53
Array in 49: -48
```

Figure 5. Resultado de la función load data

Seguido a esto se creó una función la cual se llama ”fft”, como su nombre lo indica, esta función calculara mediante un algoritmo optimizado la transformada rápida de Fourier. El vector de valores que devuelve tiene tanto parte real como parte imaginaria, teniendo en cuenta esto, se realizó una función la cual almacena los valores reales que nos devuelve la transformada y la parte imaginario la igualamos a 0 ya que para el propósito que nosotros tenemos, esta información no nos es relevante.

Considerando que en los requerimientos se nos pide poder adelantar y atrasar el audio que fue cargado en una porción de tiempo de a un segundo, se creó una funcion la cual realiza una copia de la fft que se obtuvo un segundo antes y vuelve a calcularla.

III. Conclusiones

- A la hora de realizar la lectura de los encabezados es necesario tener en cuenta que hacerlo byte a byte altera el valor real a apreciar, razón por la cual en caso de hacerlo de este modo es necesario concatenar de derecha a izquierda el valor binario almacenado en cada uno de estos bytes.
- Teniendo en cuenta la data que se encuentra dentro de cada uno de los archivos, podemos darnos cuenta de que para utilizarlos tanto en la transformada rápida de Fourier como en cualquier otra aplicación debemos tener cuidado ya que estos están distribuidos cada 2 bytes, es decir, para poder extraer la información necesaria en un instante determinado de tiempo tenemos que leer la información de los 2 bytes siguientes al largo de todo el contenido de la información.
- El desplazamiento dentro de un fichero por medio de apuntadores haciendo uso de funciones de C implica tomar en consideración la forma en la que estos se desplazan sobre los bytes a razón de los parámetros de cada función, puesto que errar en la dirección de memoria de interes altera el contenido de todo lo procesado.
- La elaboración de la transformada rápida haciendo uso de niveles hace referencia a segmentar los datos de audio en muestras de tiempo de un segundo y aplicar dicha operación con las frecuencias deseadas y obtener las potencias en cada ancho de banda.
- A lo largo de todo el desarrollo del laboratorio se pudo evidenciar que el tamaño que presentan los datos del audio y el tipo de variable en el cual estan guardados afecta de forma negativa el funcionamiento del programa a la hora de poder graficar
- Observamos que al tener diferentes algoritmos para poder ejecutar la transfomada rapida de Fourier, muchos de ellos no toman en cuenta las frecuencias de muestreo como tambien el tamaño de los datos que se van a procesar y el tipo de variables en el cual se van a guardar.

References

[1] S. Whibley, M. Day, P. May, and M. Pennock, “Wav format preservation assessment,” *British Library: London, UK*, 2016.

[2] B. Champagne and F. Labeau, “Discrete time signal processing,” *Class Notes for the Course ECSE-412, Department of Electrical & Computer Engineering, McGill University*, pp. 190–194, 2004.

[3] J. Markel, “Fft pruning,” *IEEE transactions on Audio and Electroacoustics*, vol. 19, no. 4, pp. 305–311, 1971.