INFORME 1: REPRODUCTOR .WAV

Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Sede Bogotá
Docente: Camilo Camacho
Autores: William Quitian (william.quitian01@usa.edu.co),
Brayan Herrera (brayan.herrera02@usa.edu.co)
Neyder Vargas (neyder.vargas01@correo.usa.edu.co)

***Resumen*—En este documento encontrará el desarrollo de la práctica de laboratorio, la cual tiene por objetivo principal, desarrollar un reproductor de archivos WAV de tal forma que sea posible identificar su frecuencia de muestreo, número de bits, sus canales, entre otros, siendo posible identificar las potencias existentes en el archivo por medio de la transformada rápida de Fourier y representando de forma gráfica los datos obtenidos, obteniendo varios de los ítems mencionados anteriormente pero teniendo ciertas dificultades a la hora de ejecutar el programa siendo necesario cambiar en varias ocasiones de compilador.**

I. INTRODUCCIÓN

La historia del archivo de audio en formato WAVE se remonta a 1991 donde Microsoft e IBM lo desarrollaron, es el formato mas usado en windows para trabajar con audios sin editar y sin comprimir por lo que este formato es considerado como un formato de audio sin perdida que conserva un alta calidad de sonido[1].

Para el análisis de los datos entregados por el archivo WAV es de suma importancia hacer uso de la transformada rápida de Fourier debido a que esta tiene la capacidad de descomponer la señal en sus componentes espectrales individuales y de este modo entregar información propicia sobre la composición de los datos que nos entrega el archivo WAV, este método fue desarrollado por Joseph Fourier el cual publico sus resultados sobre este método en 1807 en su investigación sobre el análisis armónico[2]. En el presente informe visualizará el desarrollo de un programa el cual permita cargar un archivo en formato WAV en la RAM descomponiendo sus canales, frecuencias y bits de muestreo, entre otros datos.

II. MONTAJE - PROCEDIMIENTO

Para la práctica de laboratorio es necesario cargar en memoria RAM un archivo en formato WAV, el cual será el de una onda sinusoidal, el cual debe cumplir con las siguientes condiciones:

- Debe mostrar su frecuencia de muestreo, número de bits, canales, entre otros.
- Es necesario diseñar un menú que permita cargar porciones de bytes de 1 segundo de audio.
- Debe Calcular la transformada rápida de Fourier de 10 niveles de la porción de audio seleccionada y extraer la potencia de sus frecuencias existentes.
- Hacer el diseño para representar en nivel de potencia de forma gráfica con caracteres de tipo discreto.

A continuación se presenta un diagrama de flujo con la estructura a seguir para la ejecución de la práctica de laboratorio:

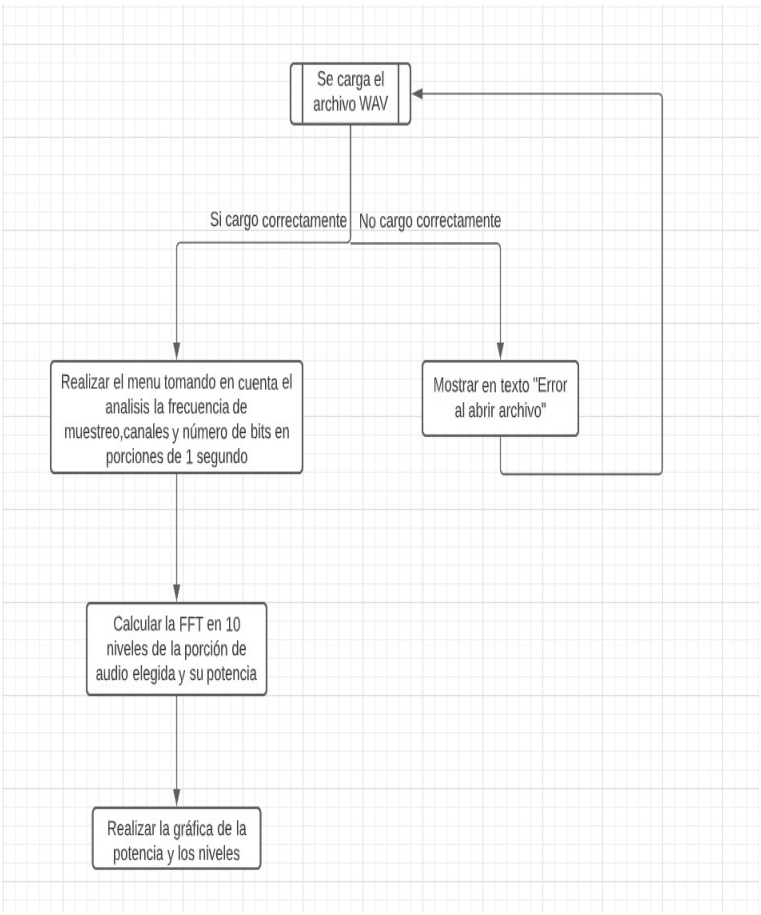

Figura 1: *Diagrama de flujo del programa*

Para la realización de la práctica de laboratorio fue necesario hacer uso de un compilador en lenguaje C para el cual se escogió el compilador online Replit, a continuación se presentan fragmentos del código y la explicación de dichos fragmentos:

III. RESULTADOS

En primer lugar, se procedió a cargar las librerías stdio.h y stdlib.h para acceder a las funciones de entrada y salida del lenguaje C y para manejar la memoria dinámica, respectivamente.

Luego, se definieron tres estructuras para almacenar los chunks del archivo WAV: RIFF_Header, FMT_Chunk y DATA_Chunk. Cada una de estas estructuras representa un chunk específico en el archivo WAV y tiene una serie de campos que almacenan información relevante para ese chunk.

En la función main(), se abre el archivo WAV en modo binario usando la función fopen(). En caso de que el archivo no se pueda abrir, se imprime un mensaje de error y se devuelve un código de error para un mejor seguimiento.

A continuación, se lee y deserializa el chunk RIFF usando la función fread(). La función fread() lee sizeof(riffChunk) bytes desde el archivo y los guarda en la variable riffChunk. Si todos los chunks son válidos, se pueden utilizar los datos de la estructura para hacer cualquier cosa que se desee con el archivo WAV.

Por último, se imprimen los valores como lo son la frecuencia de muestreo, canales, tasa de bytes, entre otros, con base al formato canónico del archivo WAVE.

Una vez expuesto el funcionamiento del código se procede a adjuntar imágenes del mismo para su visualización:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define FILE_NAME "seno.wav"

typedef struct {
    char chunkID[4];
    int chunkSize;
    char format[4];
} RIFFHeader;

typedef struct {
    char subchunk1ID[4];
    int subchunk1Size;
    short audioFormat;
    short numChannels;
    int sampleRate;
    int byteRate;
    short blockAlign;
    short bitsPerSample;
} FMTChunk;

```

Figura 2: *Librerías, Estructura para el chunk RIFF y FMT*

```c
typedef struct {
    char subchunk2ID[4];
    int subchunk2Size;
} DataChunk;
```

Figura 3: *Estructura para el chunk de datos*

```c
int main() {
    int saltosegundo;
    char o,n,opcion,p=0;
    // Abrir el archivo WAV en modo binario
    FILE *fp;
    fp = fopen(FILE_NAME, "rb");
    if (fp == NULL) {
        printf("Error al abrir el archivo\n");
        return 1;
    }

    // Deserializar la cabecera RIFF
    RIFFHeader riffHeader;
    fread(&riffHeader, sizeof(RIFFHeader), 1, fp);

    // Verificar que el archivo es un archivo WAV válido
    if (strncmp(riffHeader.chunkID, "RIFF", 4) != 0 ||
        strncmp(riffHeader.format, "WAVE", 4) != 0) {
        printf("El archivo no es un archivo WAV válido\n");
        return 1;
    }
```

Figura 4: *Cuerpo del codigo*

```c
    // Deserializar el chunk FMT
    FMTChunk fmtChunk;
    fread(&fmtChunk, sizeof(FMTChunk), 1, fp);

    // Verificar que el chunk es un chunk FMT válido
    if (strncmp(fmtChunk.subchunk1ID, "fmt ", 4) != 0) {
        printf("El archivo no tiene un chunk FMT válido\n");
        return 1;
    }

    // Deserializar el chunk de datos
    DataChunk dataChunk;
    fread(&dataChunk, sizeof(DataChunk), 1, fp);

    // Verificar que el chunk es un chunk de datos válido
    if (strncmp(dataChunk.subchunk2ID, "data", 4) != 0) {
        printf("El archivo no tiene un chunk de datos válido\n");
        return 1;
    }
```

Figura 5: *Lectura, deserializaciòn y verificación de los chunks*

```c
    // Imprimir información sobre el archivo WAV
    printf("Tamaño del archivo: %d bytes\n", riffHeader.chunkSize);
    printf("Formato de audio: %d\n", fmtChunk.audioFormat);
    printf("Número de canales: %d\n", fmtChunk.numChannels);
    printf("Tasa de muestreo: %d Hz\n", fmtChunk.sampleRate);
    printf("Tasa de bytes: %d\n", fmtChunk.byteRate);
    printf("Alineación de bloques: %d\n", fmtChunk.blockAlign);
    printf("Bits por muestra: %d\n", fmtChunk.bitsPerSample);
    printf("Tamaño de los datos: %d bytes\n", dataChunk.subchunk2Size);

    printf("Elija una opción del menú:\n");
    printf("n. saltar 1 segundo\n");
    printf("o. retroceder 1 segundo\n");
```

Figura 6: *Impresión de los datos*

A partir de la compilación de este código es posible obtener los siguientes valores:

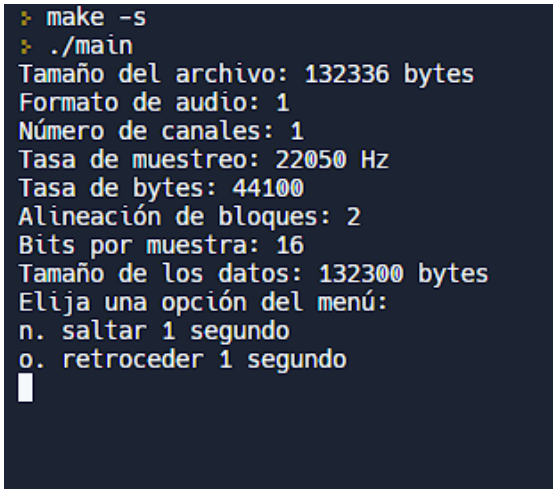

Figura 7: *Datos de la compilación del archivo*

Cabe resaltar que el código tiene la posibilidad de saltar 1 segundo con dificultades al momento de retroceder, además se agregó la ubicación de los datos que usaran para el tratamiento por el medio de la FFT, la cual por falta de conocimiento en la manipulación de señales no pudo ser graficada.

IV. CONCLUSIONES

Es posible concluir a partir de la práctica de laboratorio que se cargó exitosamente el archivo WAV en la memoria RAM, siendo capaces de identificar en el código valores como lo son su frecuencia de muestreo, el número de bits y los canales presentes en el archivo, entre otros. Fue posible identificar el tamaño del archivo y la cantidad de bits que tiene este por segundo, se logró implementar un menú que le permita al usuario identificar la cantidad de bits por segundo teniendo dificultades en hacer que este retroceda, fue posible identificar la ubicación de los datos necesarios para aplicar la FFT presentando dificultades en su implementación y en el análisis de los datos de manera gráfica, por lo anterior es posible concluir que el proyecto requiere de varios conocimientos tanto en lenguaje c como en la manipulación de señales las cuales necesitan ser estudiadas por el grupo de investigadores para completar los objetivos de este laboratorio.

REFERENCIAS

[1] Çonocimiento general del formato WAV". Apowersoft - Aplicaciones y Soluciones en Línea para el Trabajo y la Vida Diaria. Rescatado el 31 de enero del 2023

[2] Colaboradores de los proyectos Wikimedia. "Serie de Fourier - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/Serie_de_Fourier
Rescatado el 31 de enero del 2023