

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 practica de laboratorio la cual tiene por objetivo principal, desarrollar un reproductor de archivos .WAP 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 rapida de Fourier y representando de forma gráfica los datos obtenidos, Obteniendo varios de los items mencionados anteriormente pero teniendo ciertas dificultades a la hora de ejecutar el programa viendonos en la necesidad de cambiar en varias ocasiones de compilador.

- Es necesario diseñar un menú que permita cargar porciones de bytes de 1 segundo de audio.
- 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 en linea uno encima de otro.

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 transformación 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 entregas el 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.

II. MONTAJE - PROCEDIMIENTO

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

- Debe mostrar su frecuencia de muestreo, número de bits, canales, entre otros.

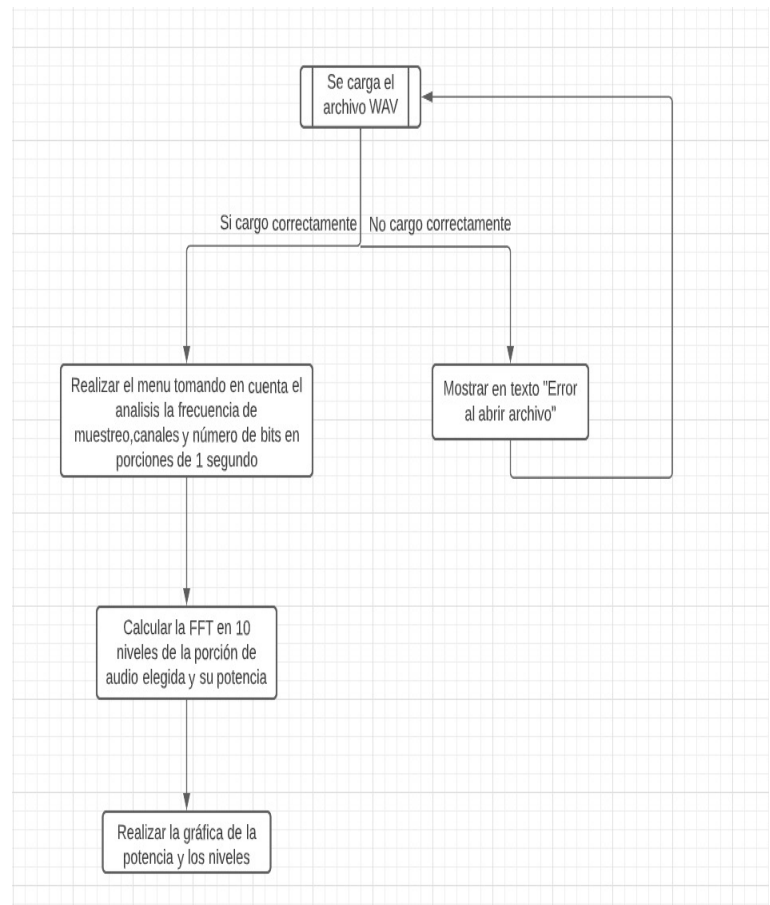


Figura 1: Diagrama de flujo del programa

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 cargan las librerías `stdio.h` y `stdlib.h` para acceder a las funciones de entrada y salida de C y para manejar la memoria dinámica, respectivamente.

Luego, se definen 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()`. Si el archivo no se puede abrir, se imprime un mensaje de error y se devuelve un código de error.

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.

A partir de lo anterior se procede a mostrar los :

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define FILE_NAME "seno.wav"
6
7  typedef struct {
8      char chunkID[4];
9      int chunkSize;
10     char format[4];
11 } RIFFHeader;
12
13 typedef struct {
14     char subchunk1ID[4];
15     int subchunk1Size;
16     short audioFormat;
17     short numChannels;
18     int sampleRate;
19     int byteRate;
20     short blockAlign;
21     short bitsPerSample;
22 } FMTChunk;
23

```

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

```

24  typedef struct {
25      char subchunk2ID[4];
26      int subchunk2Size;
27  } DataChunk;
28

```

Figura 3: Estructura para el chunk de datos

```

29  int main() {
30      int saltosegundo;
31      char o,n,opcion,p=0;
32      // Abrir el archivo WAV en modo binario
33      FILE *fp;
34      fp = fopen(FILE_NAME, "rb");
35      if (fp == NULL) {
36          printf("Error al abrir el archivo\n");
37          return 1;
38      }
39
40      // Deserializar la cabecera RIFF
41      RIFFHeader riffHeader;
42      fread(&riffHeader, sizeof(RIFFHeader), 1, fp);
43
44      // Verificar que el archivo es un archivo WAV válido
45      if (strcmp(riffHeader.chunkID, "RIFF") != 0 ||
46          strcmp(riffHeader.format, "WAVE") != 0) {
47          printf("El archivo no es un archivo WAV válido\n");
48          return 1;
49      }
50

```

Figura 4: Cuerpo del código

```

51      // Deserializar el chunk FMT
52      FMTChunk fmtChunk;
53      fread(&fmtChunk, sizeof(FMTChunk), 1, fp);
54
55      // Verificar que el chunk es un chunk FMT válido
56      if (strcmp(fmtChunk.subchunk1ID, "fmt ") != 0) {
57          printf("El archivo no tiene un chunk FMT válido\n");
58          return 1;
59      }
60
61      // Deserializar el chunk de datos
62      DataChunk dataChunk;
63      fread(&dataChunk, sizeof(DataChunk), 1, fp);
64
65      // Verificar que el chunk es un chunk de datos válido
66      if (strcmp(dataChunk.subchunk2ID, "data") != 0) {
67          printf("El archivo no tiene un chunk de datos válido\n");
68          return 1;
69      }
70

```

Figura 5: Lectura, deserialización y verificación de los chunks

```

71      // Imprimir información sobre el archivo WAV
72      printf("Tamaño del archivo: %d bytes\n", riffHeader.chunkSize);
73      printf("Formato de audio: %d\n", fmtChunk.audioFormat);
74      printf("Número de canales: %d\n", fmtChunk.numChannels);
75      printf("Tasa de muestreo: %d Hz\n", fmtChunk.sampleRate);
76      printf("Tasa de bytes: %d\n", fmtChunk.byteRate);
77      printf("Alineación de bloques: %d\n", fmtChunk.blockAlign);
78      printf("Bits por muestra: %d\n", fmtChunk.bitsPerSample);
79      printf("Tamaño de los datos: %d bytes\n", dataChunk.subchunk2Size);
80
81      printf("Elija una opción del menú:\n");
82      printf("n. saltar 1 segundo\n");
83      printf("o. retroceder 1 segundo\n");
84

```

Figura 6: Impresión de los datos

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

```

❖ make -s
❖ ./main
Tamaño del archivo: 132336 bytes
Formato de audio: 1
Número de canales: 1
Tasa de muestreo: 22050 Hz
Tasa de bytes: 44100
Alineación de bloques: 2
Bits por muestra: 16
Tamaño de los datos: 132300 bytes
Elija una opción del menú:
n. saltar 1 segundo
o. retroceder 1 segundo

```

Figura 7: Datos de la compilación del archivo

Cabe resaltar que el código tiene la posibilidad de saltar 1 segundo y retroceder a gusto del usuario.

IV. CONCLUSIONES

Es posible concluir a partir de la práctica de laboratorio que fue posible cargar 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, presentando dificultades en la implementación de la transformada rápida de Fourier para analizar los datos de manera gráfica.

REFERENCIAS

- [1] Conocimiento 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