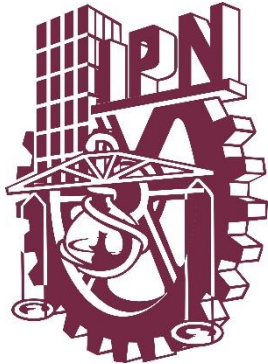


PRÁCTICA 1

REDUCCIÓN DE VOLUMEN A UN ARCHIVO WAV



Teoría de comunicaciones y señales

Profesor: Gutiérrez Aldana Eduardo

Alumno:

Amador Nava Miguel Ángel

Grupo: 3CM8

Introducción

Esta práctica servirá para comenzarnos a familiarizar con las señales digitales y entender el tratamiento de las mismas, necesitaremos el uso de software como es GoldWave para poder generar las señales que después analizaremos y le haremos la modificación correspondiente.

Esta práctica se realizará con archivos WAV los cuales son ideales dado que no se encuentran comprimidos y muestran una alta fidelidad de las señales.

Objetivos

Generales: Poder empaparnos de la teoría de señales digitales para poder manipular dichas señales, así como aprender el formato WAV.

Específicos: Reducir el volumen de una señal de un archivo WAV al 50%.

Software y equipo utilizado

Ubuntu 17.10

Wine 1.7

GoldWave 4.26

Frhead 1.3.10

Sublime Text 3

Marco teórico

Muestreo

Como primera instancia necesitamos tomar muestras de una señal analógica, con un convertidor analógico-digital. Al número de muestras por segundo se le conoce como frecuencia de muestreo que representaremos como F_s , es importante mencionar que la mayoría de los archivos de audio tienen una frecuencia de 4.1 kHz (ósea 4100 muestras por segundo) y los archivos de video son de 4.8kHz.

El periodo de muestreo T es el tiempo en segundos que existe entre muestra y muestra y a su vez es el inverso de la frecuencia de muestreo $T = \frac{1}{F_s}$.

A la señal que se genera de tomar muestras de una señal continua se le conoce como señal discreta. Generalmente las muestras son de 16 bits u 8 bits.

Canales

Mono: Significa que la misma señal la tendremos para todos los canales, por ejemplo en unos audífonos en ambas bocinas se escucharía lo mismo.

Estéreo: Significa que tendremos dos señales independientes, una muestra tendrá la parte de un canal y del otro. Si es el caso de que escucháramos un audio estéreo con unos audífonos, en un audífono saldrá una señal y por el otro la otra señal.

Teorema de muestreo de Nyquist

Este teorema nos dice que la frecuencia de muestreo debe ser por lo menos dos veces mayor a la frecuencia contenida en la señal analógica.

$$F_s = 2F_{max}$$

$F_s = \text{Frecuencia de muestreo.}$

$F_{max} = \text{Frecuencia maxima en la señal analogica.}$

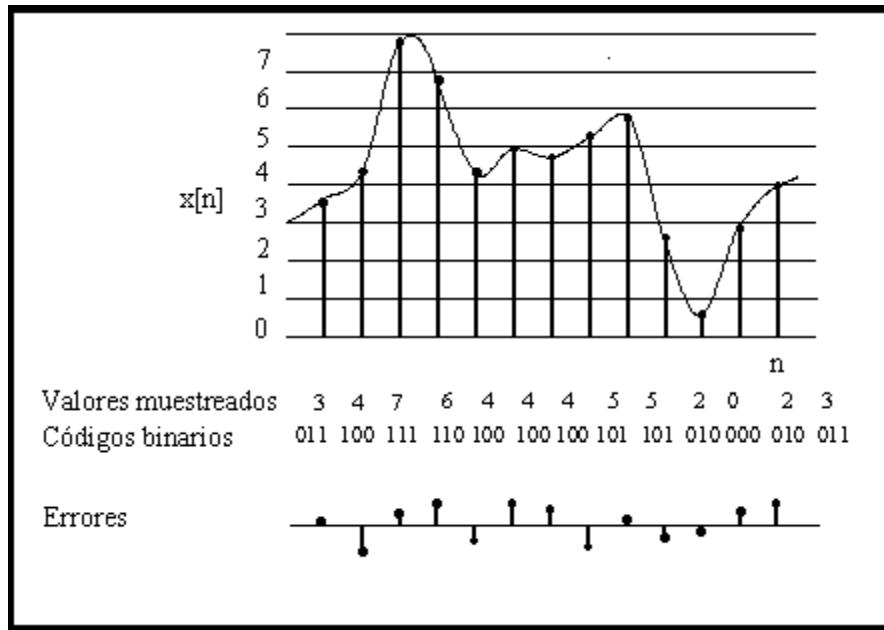


Ilustración 1 Ejemplo de discretización de una señal analógica

Teoría del sonido

Aunque la teoría del sonido es mucho más amplia de lo que mencionaré aquí, lo que se mencionará servirá para entender mejor y lograr los objetivos de la práctica.

El sonido es un fenómeno físico que consiste en la alteración de las partículas de un medio elástico, producida por un elemento en vibración, que provoca una sensación de audición, pues estas vibraciones generan ondas que se introducen en el pabellón del oído haciendo vibrar la membrana del tímpano de ahí pasa al oído medio e interno y excita las terminales del medio acústico que transporta al cerebro los impulsos y estas generan la sensación sonora.

Principio del funcionamiento de las bocinas

Las bocinas están conformadas por una bobina (Voice Coil) la cual al polarizarse genera un campo electromagnético que al estar cerca del mismo polo del imán genera una repulsión y al estar cerca lo polos opuestos se atraen este movimiento hace que el cono (Cone) genere ondas que se convertirán en el sonido.

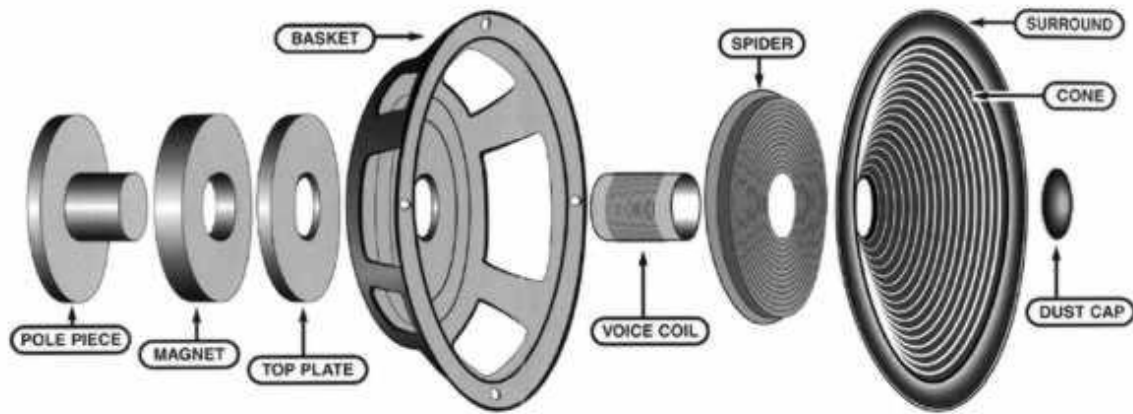


Ilustración 2 Composición de una bocina

La señal que se mandará a la bocina irá alternado lo que generará el cambio de polarización en la bobina y esto generará las ondas sonoras.

Y aquí se encuentra la parte que nos interesa, si nosotros quisiéramos incrementar el volumen de esta señal solamente tendríamos que mandar más voltaje a la bobina y esto amplificaría la señal.

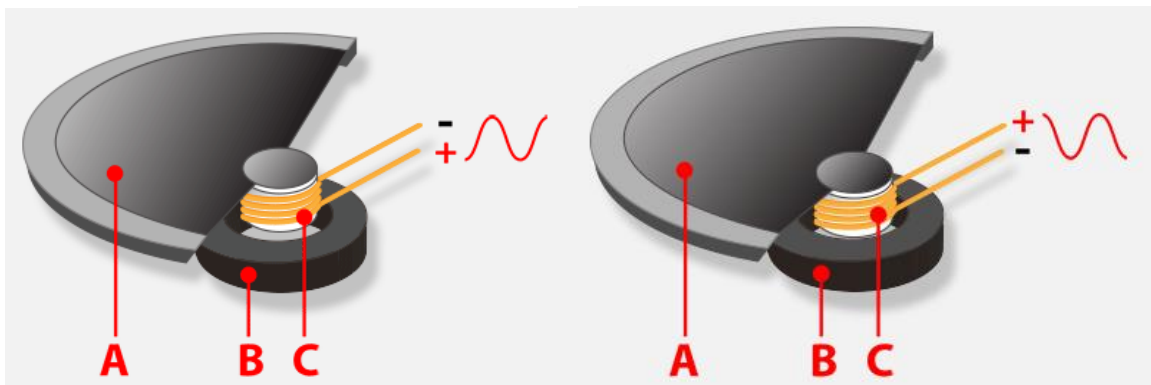


Ilustración 3 Funcionamiento de una bocina

Preparación del entorno para el desarrollo de la práctica

GoldWave 4.26 y Frhead 1.3.10 fueron creados para Windows por lo cual para poder ejecutarlos en Ubuntu 17.10 necesitaremos instalar wine.

Introduciremos los siguientes comandos:

```
sudo add-apt-repository ppa:ubuntu-wine/ppa
```

```
sudo apt-get update
```

```
sudo apt-get install wine1.7 winetricks
```

Una vez instalado wine para ejecutar GoldWave y Frhead los ejecutaremos de la siguiente manera:

```
WINEPREFIX=~/.wine32 wine "nombre del ejecutable"
```

Planteamiento del problema

Deseemos reducir el volumen de un archivo WAV al 50%, generando una señal en GoldWave con las siguientes especificaciones:

Primer archivo:

- Tipo de señal: Monoaural
- Duración: 0.016 segundos.
- Frecuencia de muestreo: 2000 Hz
- Guardar el archivo como wav.
- Con la herramienta f(x) de GoldWave introducir:

$$(n(N - 0.5) * 2)$$

- Guardar el archivo como wav.

Se pide realizar un sistema en lenguaje C que se le pase el nombre del archivo wav de entrada, y el nombre que tendrá de salida. El archivo de salida tendrá la mitad del volumen que el de entrada.

Análisis Teórico

Del archivo generado anteriormente tendremos una salida como se muestra en la ilustración 4

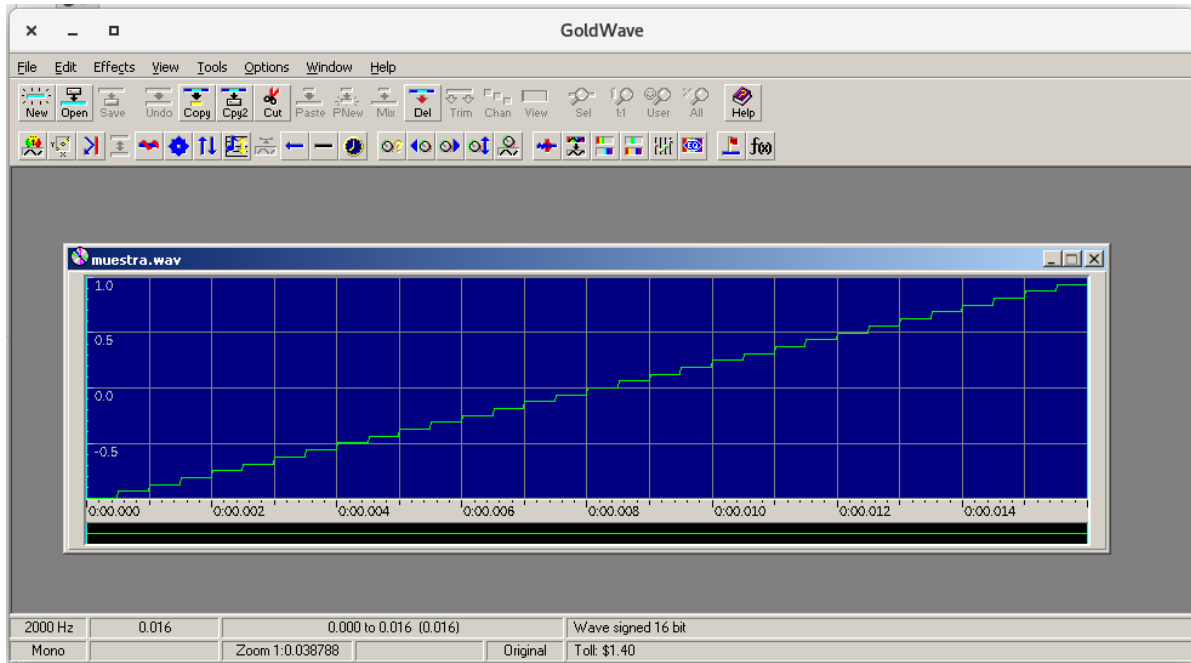


Ilustración 4 Muestra de archivo wav con las especificaciones indicadas

Para reducir el volumen con GolgWave iremos a efectos, despues a volumen y finalmente a cambiar, como se muestra en la ilustración 5.

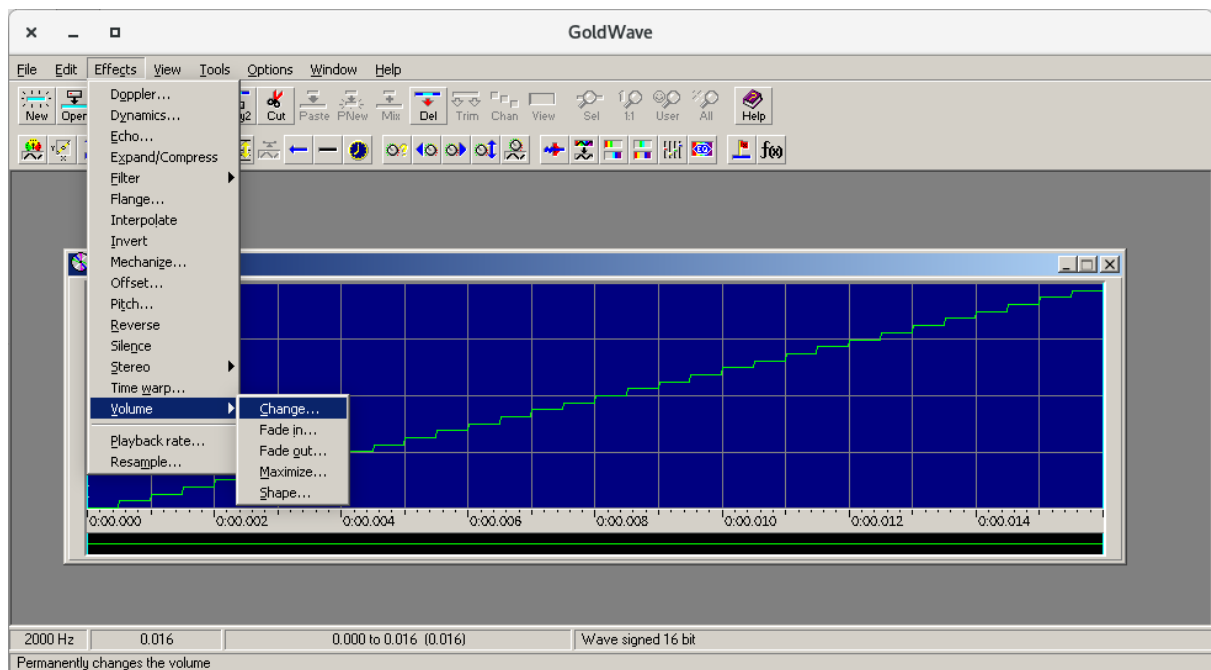


Ilustración 5 Procedimiento para reducir el volumen

Procederemos a indicar el 50% y a guardar el archivo. Para su posterior analisis.

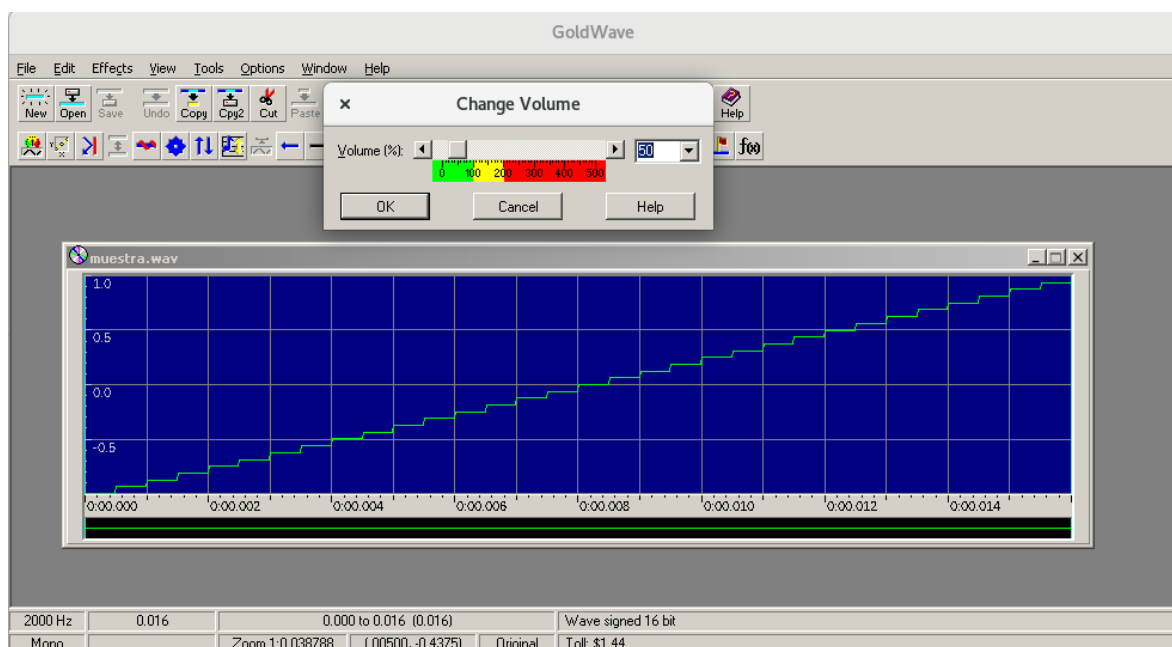


Ilustración 6 Modificación al 50% del volumen

La salida deberá ser como la siguiente:

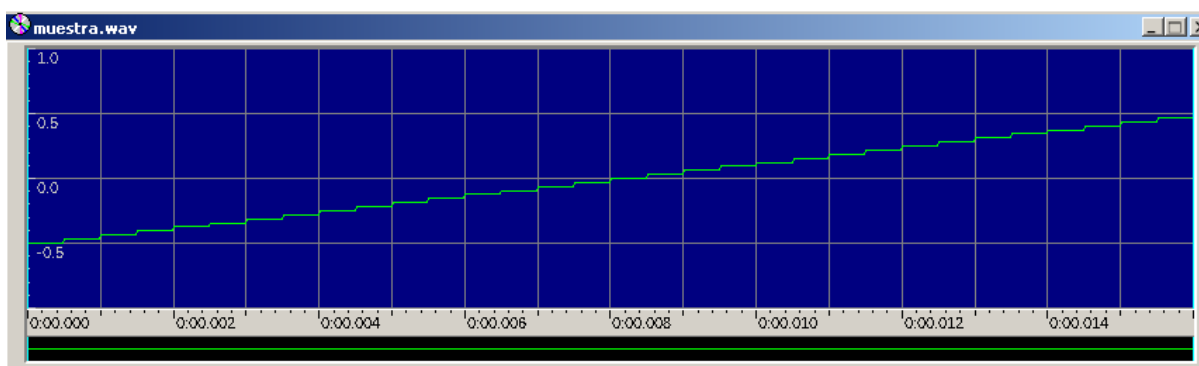


Ilustración 7 Salida de la señal con reducción de volumen al 50%

Cabecera del archivo wav

Bytes	Campo	Descripción
4	ChunkID	Contiene las letras "RIFF" en ASCII
4	ChunkSize	36 bytes más SubChunk2Size. Este es el tamaño entero del archivo en bytes menos 8
4	Format	Contiene las letras "WAVE"
4	Subchunk1ID	Contiene las letras "fmt"
4	Subchunk1Size	Este es el tamaño del resto del Subchunk
2	AudioFormat	Los valores distintos de 1 indican alguna forma de compresión.
2	NumChannels	Numero d canales: 1= mono, 2=Stereo
4	SampleRate	Número de muestras por segundo
4	ByteRate	$\text{SampleRate} * \text{NumChannels} * \text{BitsPerSample}/8$
2	BlockAlign	$\text{NumChannels} * \text{BitsPerSample}/8$. El número de bytes para una muestra incluyen todos los canales
2	BitsPerSample	Bits por muestra. 8,16,etc.
4	Subchunk2ID	Contiene las letras "data"
4	Subchunk2Size	$\text{NumSamples} * \text{NumChannels} * \text{BitsPerSample}/8$. Este es el número de bytes en la data
*	Data	Datos del sonido

Tabla 1 Información de la cabecera de un archivo wav

Veamos un ejemplo de la cabecera:

```
52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d
```

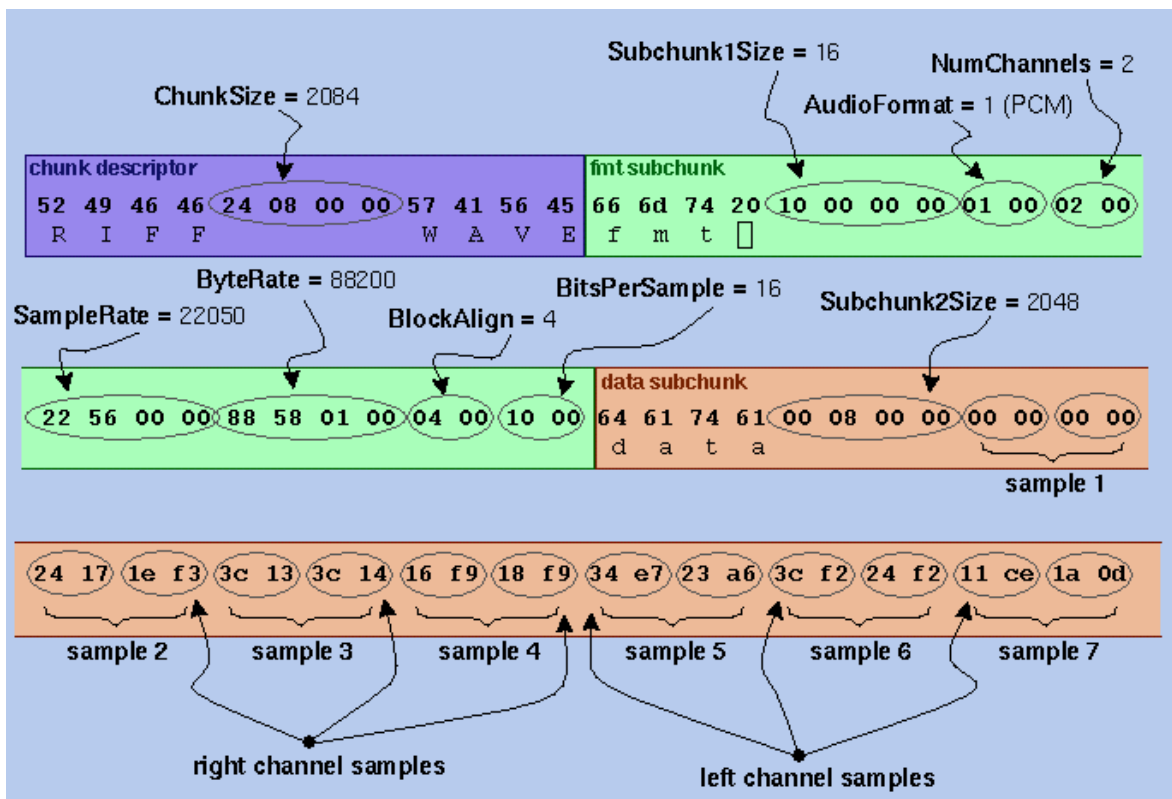


Ilustración 8 Análisis de la cabecera de un archivo wav.

Una vez que tenemos la explicación de la cabecera y los dos archivos wav procederemos a abrirlos con `fread`.

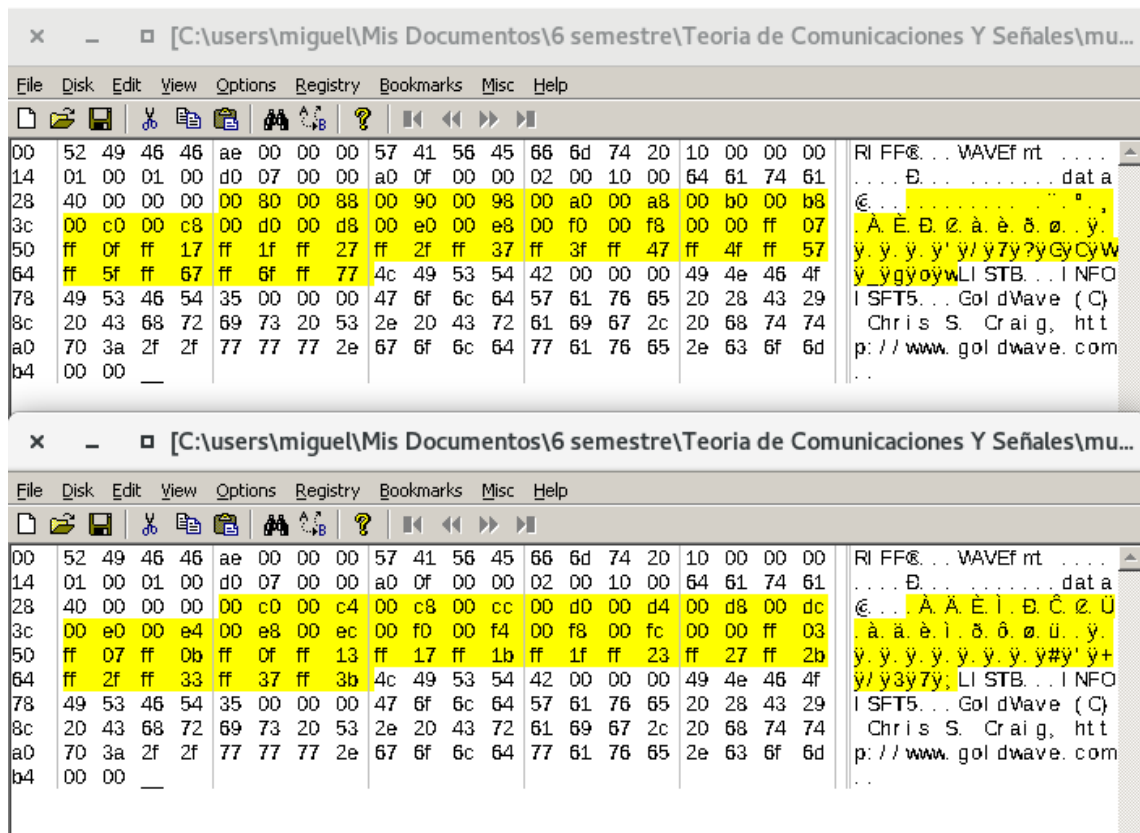


Ilustración 10 Bytes de información de sonido de los archivos wav

Después de las muestras de información podemos observar que hay 74 bytes en ambos archivos que son idénticos por lo cual concluimos que es el pie de los archivos wav.

En esta sección podemos decir como información relevante que:

Tamaño total del archivo = ChunkSize + 8 bytes

Numero de muestras=Subchunk2Size/NunChannels*(BitsPerSample/8)

Bytes de la información del sonido = Subchunk2Size

Bytes de pie del archivo = ChunkSize – 36 bytes – Subchunk2Size

Análisis Práctico

Es importante considerar que al abrir el archivo en binario debemos utilizar el tipo de dato correcto para ir trabajando los bytes de la cabecera y las muestras. Una consideración importante es que si el archivo wav tiene muestras de 8 bits el tipo de dato adecuado es un char y si es de 16 bits el tipo adecuado es short.

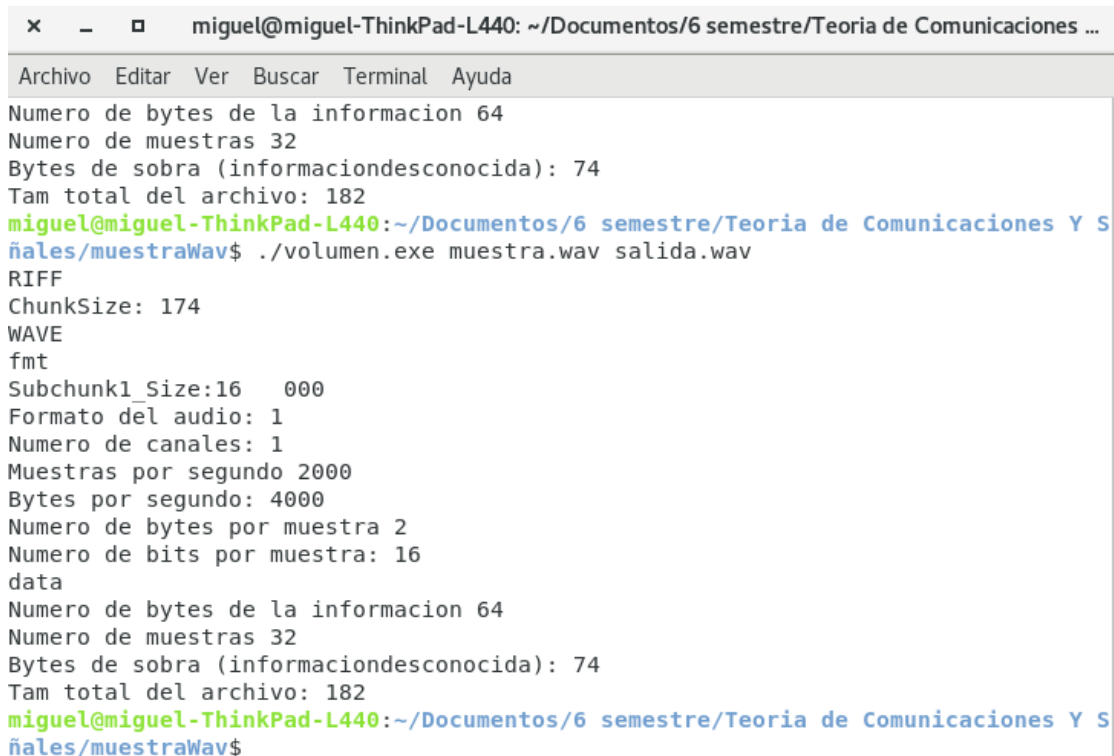
Se debe considerar que las muestras pueden tener valores positivos y negativos y para poder reducir el volumen solo hay que dividir entre dos cada muestra.

Siempre es aconsejable dividir nuestro sistema y no buscar lo óptimo cuando se está programando algo que nos pudiera parecer complejo, una vez que es funcional podemos optimizar.

Para compilar nuestro programa teclearemos en la terminal:

```
gcc "nombre del programa.c" -o "nombre de salida"
```

Una vez compilado ejecutaremos como se muestra en la ilustración 11.



```
x _ □ miguel@miguel-ThinkPad-L440: ~/Documentos/6 semestre/Teoria de Comunicaciones ...
Archivo Editar Ver Buscar Terminal Ayuda
Numero de bytes de la informacion 64
Numero de muestras 32
Bytes de sobra (informaciondesconocida): 74
Tam total del archivo: 182
miguel@miguel-ThinkPad-L440:~/Documentos/6 semestre/Teoria de Comunicaciones Y S
ñales/muestraWav$ ./volumen.exe muestra.wav salida.wav
RIFF
ChunkSize: 174
WAVE
fmt
Subchunk1_Size:16 000
Formato del audio: 1
Numero de canales: 1
Muestras por segundo 2000
Bytes por segundo: 4000
Numero de bytes por muestra 2
Numero de bits por muestra: 16
data
Numero de bytes de la informacion 64
Numero de muestras 32
Bytes de sobra (informaciondesconocida): 74
Tam total del archivo: 182
miguel@miguel-ThinkPad-L440:~/Documentos/6 semestre/Teoria de Comunicaciones Y S
ñales/muestraWav$
```

Ilustración 11 Ejecución del programa

No se mostrará el archivo de salida del programa pues es idéntico al que se analizó en la sección teórica.

Conclusiones

Con esta práctica se cumplieron los objetivos de poder familiarizarse con los archivos wav y poder entender como es el almacenamiento de una señal en forma discreta. Las herramientas sugeridas por el profesor fueron de gran utilidad pues al analizar la teoría de la cabecera todo es muy abstracto pero al visualizar los bits en hexadecimal de la señal de entrada y de salida antes de empezar a implementar el programa podemos entender perfectamente lo que se está pidiendo hacer para bajar el volumen de sonido.

Referencias

<http://soundfile.sapp.org/doc/WaveFormat/>

Cousera Digital Signal Processing École Polytechnique Fédérale de Lausanne

Código

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

/**
 *   La informacion de la cabecera se encuentra:
 *   http://soundfile.sapp.org/doc/WaveFormat/
 */

typedef struct CabeceraWAV{

    unsigned char ChunkID[4];
    unsigned int ChunkSize;
    unsigned char Format[4];
    unsigned char Subchunk1_ID[4];
    unsigned char Subchunk1_Size[4];
    unsigned char AudioFormat[2];
    unsigned char NumChannels[2];
    unsigned int SampleRate;
    unsigned int ByteRate;
    unsigned short BlockAlign;
    unsigned short BitsPerSample;
    unsigned char Subchunk2_ID[4];
    unsigned int Subchunk2_Size;//Catidad de bytes de informacion de la
señal

}Cabecera;

void informacionAudio(Cabecera cabeceraAudio){

    printf("%c%c%c%c\n",cabeceraAudio.ChunkID[0],cabeceraAudio.ChunkID[
1],cabeceraAudio.ChunkID[2],cabeceraAudio.ChunkID[3]);
    printf("ChunkSize: %d\n",cabeceraAudio.ChunkSize);

    printf("%c%c%c%c\n",cabeceraAudio.Format[0],cabeceraAudio.Format[1]
,cabeceraAudio.Format[2],cabeceraAudio.Format[3]);

    printf("%c%c%c%c\n",cabeceraAudio.Subchunk1_ID[0],cabeceraAudio.Sub
chunk1_ID[1],cabeceraAudio.Subchunk1_ID[2],cabeceraAudio.Subchunk1_ID[3])
;
    printf("Subchunk1_Size:%d
%d%d%d\n",cabeceraAudio.Subchunk1_Size[0],cabeceraAudio.Subchunk1_Size[1]
,cabeceraAudio.Subchunk1_Size[2],cabeceraAudio.Subchunk1_Size[3]);
    printf("Formato del audio: %d\n",cabeceraAudio.AudioFormat[0]
);
    printf("Numero de canales: %d\n",cabeceraAudio.NumChannels[0]
);
    printf("Muestras por segundo %d\n",cabeceraAudio.SampleRate);
    printf("Bytes por segundo: %d\n",cabeceraAudio.ByteRate);
    printf("Numero de bytes por muestra
%d\n",cabeceraAudio.BlockAlign );
    printf("Numero de bits por muestra: %d\n",
cabeceraAudio.BitsPerSample);

    printf("%c%c%c%c\n",cabeceraAudio.Subchunk2_ID[0],cabeceraAudio.Sub
```

```

chunk2_ID[1],cabeceraAudio.Subchunk2_ID[2],cabeceraAudio.Subchunk2_ID[3])
;
    printf("Numero de bytes de la informacion
%d\n",cabeceraAudio.Subchunk2_Size);
    printf("Numero de muestras
%d\n",cabeceraAudio.Subchunk2_Size/(cabeceraAudio.NumChannels[0]*(cabecer
aAudio.BitsPerSample/8)));
    printf("Bytes de sobra (informaciondesconocida):
%d\n",cabeceraAudio.ChunkSize-36-cabeceraAudio.Subchunk2_Size);
    printf("Tam total del archivo:
%d\n",cabeceraAudio.ChunkSize+8);

}

void modificacionVolumenAudio8bits(double
factorDeModificacion,unsigned int numeroMuestras,FILE
*archivoWavEntrada,FILE *archivoWavSalida, char tamEnBytesMuestra){
    char muestra;
    for(unsigned int i=0;i<numeroMuestras;i++){
        fread(&muestra,tamEnBytesMuestra,1,archivoWavEntrada);
        muestra=muestra*factorDeModificacion;
        fwrite(&muestra,tamEnBytesMuestra,1,archivoWavSalida);
    }
}

void modificacionVolumenAudio16bits(double
factorDeModificacion,unsigned int numeroMuestras,FILE
*archivoWavEntrada,FILE *archivoWavSalida, char tamEnBytesMuestra){
    short muestra;
    for(unsigned int i=0;i<numeroMuestras;i++){
        fread(&muestra,tamEnBytesMuestra,1,archivoWavEntrada);
        muestra=muestra*factorDeModificacion;
        fwrite(&muestra,tamEnBytesMuestra,1,archivoWavSalida);
    }
}

void main(int argc, char *argv[]){

    Cabecera CW;
    FILE *archivoWavEntrada,*archivoWavSalida;
    archivoWavEntrada=fopen(argv[1],"r");

    if(archivoWavEntrada==NULL){
        printf("Error en la lectura del archivo de audio\n");
    }else{

        archivoWavSalida=fopen(argv[2],"w");
        fread(&CW,44,1,archivoWavEntrada);
        fwrite(&CW,44,1,archivoWavSalida);
        char tamEnBytesMuestra=(CW.BitsPerSample/8);

        unsigned int
numeroMuestras=CW.Subchunk2_Size/(CW.NumChannels[0]*(CW.BitsPerSample/8))
;

        /**
        *    Recordar que si el archivo wab es stereo la muestra
        tendrá el doble de bytes pero se tendran que trabajar

```



```

    *      por separado. Por lo cual no afecta.
    **/
    informacionAudio(CW);

    if(tamEnBytesMuestra==1){

        modificacionVolumenAudio8bits(.5,numeroMuestras,archivoWavEntrada,archivoWavSalida,1);
    }else if(tamEnBytesMuestra==2){

        modificacionVolumenAudio16bits(.5,numeroMuestras,archivoWavEntrada,archivoWavSalida,2);

    }

    /**
    *      Terminamos de copiar la informacion restante no sabemos
    que sea esa informacion pero
    *      si sabemos que debe coincidir con
    chunkSize=subchunk2_Size -36.
    **/
    unsigned int bytesSobrantes=CW.ChunkSize-
    CW.Subchunk2_Size-36;

    char muestra;
    for(unsigned int i=0;i<bytesSobrantes;i++){

        fread(&muestra,sizeof(char),1,archivoWavEntrada);
        fwrite(&muestra,sizeof(char),1,archivoWavSalida);

    }

    fclose(archivoWavEntrada);
    fclose(archivoWavSalida);

}
}

```