Universidad de Granada

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Grado en Ingeniería Informática

Tecnologías WEB

Practica Final

Autora:
DE LA VIEJA LAFUENTE, CLAUDIA
EZZAMARTY, SALMA

Curso:
2023-2024

Índice

1. Ejercicio 1

Leer dos ficheros de sonido (WAV o MP3) de unos pocos segundos de duración cada uno. En el primero debe escucharse el nombre de la persona que realiza la práctica. En el segundo debe escucharse el apellido.

Para resolver el primer ejercicio hemos usado las siguientes funciones

```
# Ejercicio 1

# Cargar archivos .wav
Fs, sen = wavfile.read('./nombre.wav')  # Ruta del archivo con la señal
sen = sen.astype(float)
t = np.linspace(0, len(sen) / Fs, len(sen), endpoint=False)  # Asegura misma longitud

Fs2, signal2 = wavfile.read('./apellido.wav')  # Ruta del archivo con la señal
signal2 = signal2.astype(float)
t2 = np.linspace(0, len(signal2) / Fs2, len(signal2), endpoint=False)  # Asegura misma longitud

# ------------------------------------------------------------ #
```

Figura 1: Codigo Ejercicio 1

2. Ejercicio 2

Dibujar la forma de onda de ambos sonidos. Para resolver este ejercicio hemos utilizado la biblioteca **matplotlib.pyplot** que es la .^encargada"de generar las gráficas

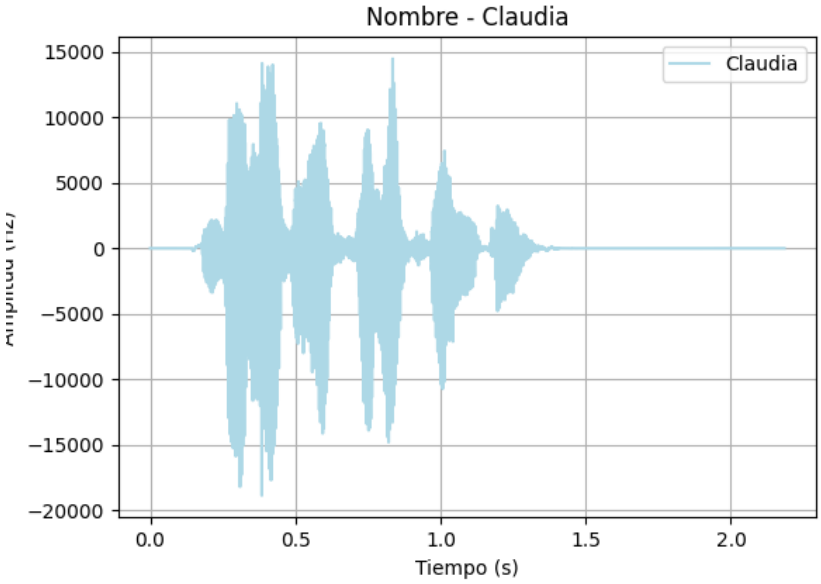

Figura 2: Onda nombre

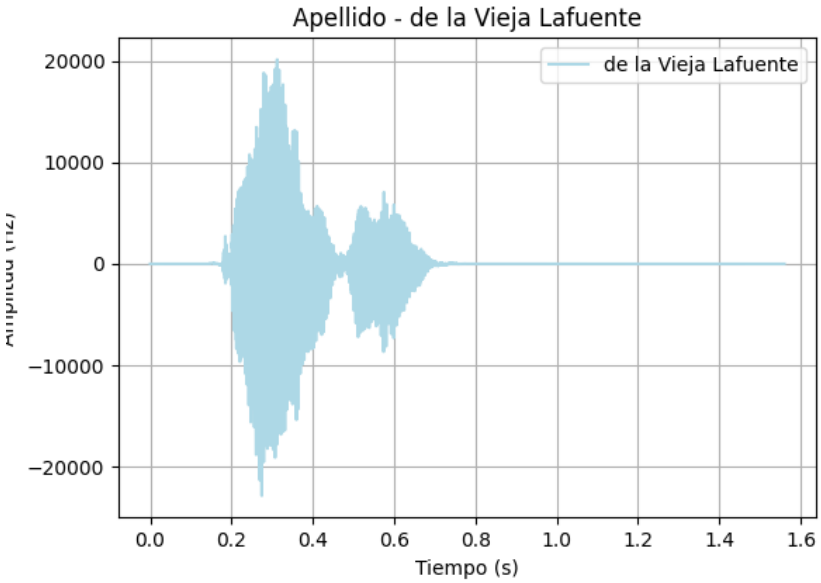

Figura 3: Onda apellidos

3. Ejercicio 3

Obtener la información de las cabeceras de ambos sonidos.

Para resolver este ejercicio se ha usado la biblioteca **wave** para obtener la información sobre los dos archivos de audio en formato WAV

```
→ PS git:(main) X python3 sonido.py
Nombre - Claudia


Número de canales ->  1
Ancho ->  2
Frame rate ->  24000
Número de frames ->  52416
Parámetros ->  _wave_params(nchannels=1, sampwidth=2, framerate=24000, nframes=5
2416, comptype='NONE', compname='not compressed')


------------------------------------------------------


Apellido - de la Vieja Lafuente


Número de canales ->  1
Ancho ->  2
Frame rate ->  24000
Número de frames ->  37440
Parámetros ->  _wave_params(nchannels=1, sampwidth=2, framerate=24000, nframes=3
7440, comptype='NONE', compname='not compressed')
```

Figura 4: Información cabeceras

4. Ejercicio 4

Unir ambos sonidos en uno nuevo.

```
# Ejercicio 4

# Cargar archivos de audio
audio1 = AudioSegment.from_wav("./nombre.wav")
audio2 = AudioSegment.from_wav("./apellido.wav")

# Concatenar los audios
audio_concatenado = audio2 +  audio1

# Guardar el audio concatenado en un nuevo archivo
audio_concatenado.export("./sonido_concatenado.wav", format="wav")
```

Figura 5: Código ejercicio 4

5. Ejercicio 5

Dibujar la forma de onda de la señal resultante.

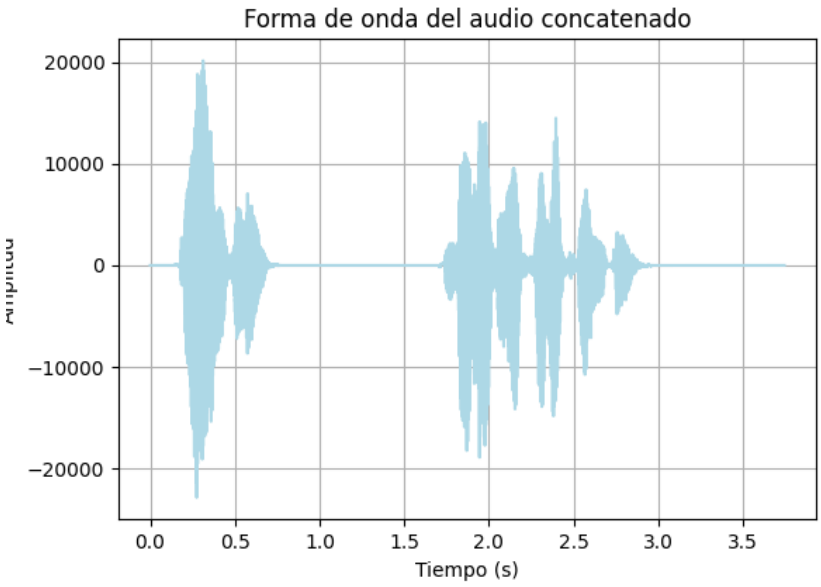

Figura 6: Onda audio concatenado de nombre + apellido

6. Ejercicio 6

Pasarle un filtro de frecuencia para eliminar las frecuencias entre 10000Hz y 20000Hz

Para resolver este ejercicio he tenido un problema, y es que si ponia que la frecuencia de corte fuese 20000Hz me salia un error diciendo que mi frecuencia de muestreo era 24000Hz, por lo que la frecuencia de corte superior debia ser menor que la mitad de esa frecuencia.

```python
# Ejercicio 6

# Cargar el archivo de audio concatenado
Fs_concatenado, sen_concatenada = wavfile.read("./sonido_concatenado.wav")

# Verificar la frecuencia de muestreo
print("Frecuencia de muestreo:", Fs_concatenado)

# Calcular el tiempo correspondiente a cada muestra
t_concatenado = np.arange(len(sen_concatenada)) / Fs_concatenado

# Definir las frecuencias de corte deseadas en Hz
lowcut = 10000  # Frecuencia de corte inferior en Hz
highcut = 10000  # Frecuencia de corte superior en Hz

# Verificar que las frecuencias de corte estén en el rango correcto
assert 0 < lowcut < Fs_concatenado / 2, "Frecuencia de corte inferior fuera de rango"
assert 0 < highcut < Fs_concatenado / 2, "Frecuencia de corte superior fuera de rango"

# Convertir las frecuencias de corte a frecuencias normalizadas
nyquist_freq = 0.5 * Fs_concatenado  # Frecuencia de Nyquist
lowcut_norm = lowcut / nyquist_freq  # Frecuencia de corte inferior normalizada
highcut_norm = highcut / nyquist_freq  # Frecuencia de corte superior normalizada

# Crear filtro pasa bajo
sos_low = butter(10, lowcut_norm, btype='low', output='sos')

# Aplicar filtro pasa bajo
sen_filtrada_low = sosfiltfilt(sos_low, sen_concatenada)

# Crear filtro pasa alto
sos_high = butter(10, highcut_norm, btype='high', output='sos')

# Aplicar filtro pasa alto
sen_filtrada = sosfiltfilt(sos_high, sen_filtrada_low)

# Calcular el tiempo correspondiente a cada muestra
t_filtrado = np.arange(len(sen_filtrada)) / Fs_concatenado

# Mostrar gráfica de la forma de onda del audio filtrado
plt.plot(t_filtrado, sen_filtrada, color='lightblue')
plt.title('Forma de onda del audio filtrado')
plt.xlabel('Tiempo (s)')
plt.ylabel('Amplitud')
plt.grid()
plt.show()
```

Figura 7: Código ejercicio 6

7. Ejercicio 7

Almacenar la señal obtenida como un fichero WAV denominado “mezcla.wav”.

```python
# Ejercicio 7

# Generar archivo de salida

output_file = "mezcla.wav"
wavfile.write(output_file, Fs_concatenado, sen_filtrada)

# ------------------------------------------------------------ #
```

Figura 8: Código ejercicio 7

8. Ejercicio 8

Cargar un nuevo archivo de sonido, aplicarle eco y a continuación darle la vuelta al sonido. Almacenar la señal obtenida como un fichero WAV denominado "alreves.wav".

Este ejercicio no he sido capaz de solucionar el error que dejo a continuación.

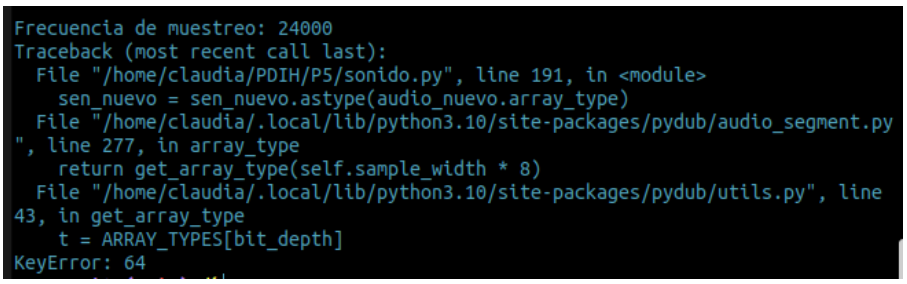

Figura 9: Error ejercicio 6