UNIVERSIDAD DE GRANADA

Periféricos y Dispositivos de Interfaz Humana

Práctica 5 - Sonido

Arturo Alonso Carbonero

ÍNDICE

1. Introducción

Para la realización de la práctica he empleado el lenguaje de programación Python, así como metodología obtenida de forma independiente combinada con la aportada por el fichero de la práctica. Además, aclarar que en el ejercicio 8 hay un comentario explicativo con el procedimiento a seguir en caso de fallo.

2. Leer ficheros

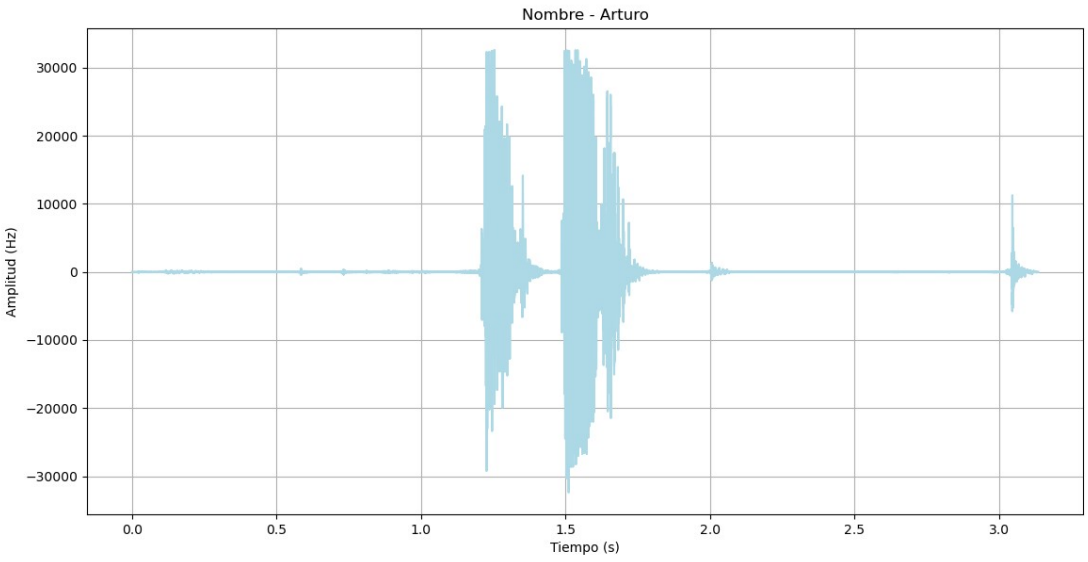
Para leer los ficheros he empleado la librería ‘wavfile’ de ‘scipy.io’.

```
20  # Cargar archivos .wav
21
22  Fs, sen=wavfile.read('./Arturo.wav') # Ruta del archivo con la señal
23  sen=sen.astype(float)
24  t=np.arange(0, len(sen)/Fs, 1/Fs)
```

3. Dibujar forma de onda

Para mostrar las gráficas he utilizado la librería ‘matplotlib.pyplot’. A continuación se muestra un ejemplo de uso.

```
35  # Mostrar gráficas
36
37  plt.plot(t, sen, color='lightblue', label='Arturo')
38  plt.title('Nombre - Arturo')
39  plt.xlabel('Tiempo (s)')
40  plt.ylabel('Amplitud (Hz)')
41  plt.grid()
```



4. Información de las cabeceras

```
# Obtener información de las cabeceras

signalAr=wave.open('./Arturo.wav' ,'r')

print("Arturo")
print("\n")
print("Número de canales -> ", signalAr.getnchannels())
print("Ancho -> ", signalAr.getsampwidth())
print("Frame rate -> ", signalAr.getframerate())
print("Número de frames -> ", signalAr.getnframes())
print("Parámetros -> ", signalAr.getparams())

print("\n")
print("-------------------------------------------------------")
print("\n")
```

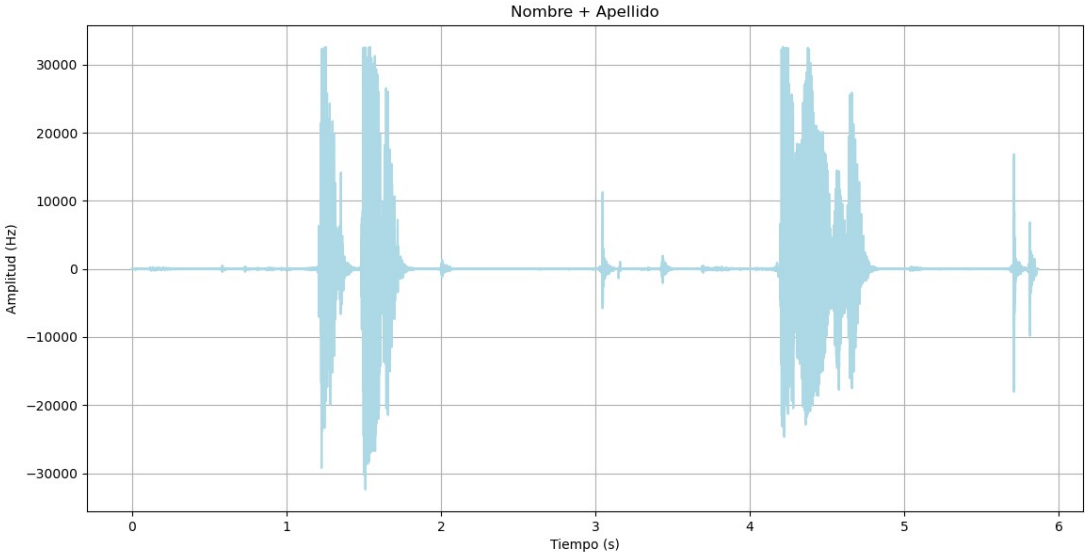
5. Unir sonidos

```
infiles = ["Arturo.wav", "Alonso.wav"]
outfile = "ArturoAlonso.wav"

data=[]
for infile in infiles:
    w = wave.open(infile, 'rb')
    data.append( [w.getparams(), w.readframes(w.getnframes())] )
    w.close()

output = wave.open(outfile, 'wb')
output.setparams(data[0][0])
for i in range(len(data)):
    output.writeframes(data[i][1])
output.close()
```

6. Dibujar forma de onda



7. Filtro

```
umbral1=1
umbral2=10000
umbral3=20000

def butter_bandpass(umbral1, umbral2, fs, order=5):
    nyq=0.5*fs
    low=umbral1/nyq
    high=umbral2/nyq
    b, a=signal.butter(order, [low, high], btype='band')
    return b, a

def butter_bandpass_filter(data, umbral1, umbral2, fs, order=5):
    b, a=butter_bandpass(umbral1, umbral2, fs, order=order)
    y=signal.lfilter(b, a, data)
    return y

def bandpass_filter(buffer):
    return butter_bandpass_filter(buffer, umbral1, umbral2, 10000, order=6)

signal4=butter_bandpass_filter(signal3, umbral1, umbral2, Fs3)
signal5=butter_bandpass_filter(signal3, umbral2, umbral3, Fs3)
signal6=signal4+signal5
```

8. Almacenar resultado e un archivo

```
# Generar archivo de salida

nombreArchivo='mezcla.wav'
wf.write(nombreArchivo, Fs3, signal6.astype('int16'))
```

9. Eco e inversión

```
# Delay e inversión

Fs4, signal7=wavfile.read('./mezcla.wav') # Ruta del archivo con la señal
# NOTA -> Probar con el fichero 'muneca.wav' si el actual no funciona
signal7=signal7.astype(float)
t4=np.arange(0, len(signal7)/Fs4, 1/Fs4)

signalDelay=np.zeros(len(signal7)+1800)

for i in range(0, len(signalDelay)):
    if i<1800:
        signalDelay[i]=signal7[i]
    elif i<len(signal7):
        signalDelay[i]=0.9*(signal7[i]+0.8*signalDelay[i-1800])
    else:
        signalDelay[i]=0.9*0.8*signalDelay[i-1800]

signalDelayCorte=signalDelay[1800:]

signalInvert=np.flip(signalDelayCorte)
```