Audio - Python - Octave*

Emerson Aldair Pérez Rivera, 201902852[1,**]

[1] *Facultad de Ingeniería, Universidad de San Carlos, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

La grabación, reproducción y análisis de audio son procesos clave en el procesamiento de señales, y tanto Python como Octave ofrecen herramientas para llevar a cabo estas tareas. En Python, bibliotecas como sounddevice permiten capturar y reproducir audio fácilmente, mientras que numpy y matplotlib son ideales para analizar y graficar el espectro de la señal. Octave, por su parte, utiliza funciones como audiorecorder, fft, y plot para lograr resultados similares, permitiendo la visualización y análisis de las componentes frecuenciales del audio. Estas técnicas son esenciales en aplicaciones como el reconocimiento de voz, filtros de audio y sistemas de acústica.

I. OBJETIVOS

- Poder grabar,reproducir y graficar audio en octave
- Realizar el mismo codigo en Python

II. CODIGOS UTILIZADOS

A. Codigo de Octave

```
% Comprobar si estamos ejecutando en MATLAB O GNU Octave
if (exist('OCTAVE_VERSION', 'builtin') ~= 0)
  % Estamos en Octave
  pkg load signal;
end

% Men  Principal
opcion = 0;
while opcion ~= 5
  % Men  de Opciones
  disp('Seleccione una Opci n: ')
  disp('1. Grabar')
  disp('2. Reproducir')
  disp('3. Graficar')
  disp('4. Graficar Densidad')
  disp('5. Salir')
  opcion = input('Ingrese su Elecci n: ');

  switch opcion
    case 1
      % Grabar Audio
      try
        duracion = input('Ingrese la Duraci n de la grabaci n en segundos: ');
        disp('Comenzando la Grabaci n...');
        recObj = audiorecorder;
        recordblocking(recObj, duracion);
        disp('Grabaci n Finalizada');
        data = getaudiodata(recObj);
        audiowrite('audio_octave.wav', data, recObj.SampleRate);
        disp('Archivo de audio grabado correctamente.');
      catch
        disp('Error al grabar el audio');
      end
    case 2
      %Reproducir Audio
      try
        [data,fs] = audioread('audio_octave.wav');
        sound(data,fs);
      catch
        disp('Error al Reproducir el audio.');
      end
    case 3
      % Grafico de Audio
      try
        [data,fs] = audioread('audio_octave.wav');
        tiempo = linspace(0,length(data)/fs,length(data));
        plot(tiempo,data);
        xlabel('Tiempo (s)');
        ylabel('Amplitud');
        title('Grafico de Audio');
      catch
        disp('Error al graficar el audio');
      end
     case 4
      %Graficando espectro de frecuencia
      try
        disp('Graficando espectro de frecuencia .. ');
        [audio,Fs] = audioread('audio_octave.wav');
        N = length(audio);
        f = linspace(0,Fs/2,N/2+1);
        ventana = hann(N);
        Sxx = pwelch(audio,ventana,0,N,Fs);
        plot(f,10*log10(Sxx(1:N/2+1)));
        xlabel('Frecuencia (Hz)');
        ylabel('Densidad Espectral de Potencia (dB/Hz)');
        title('Espectro de Frecuencia de la se al grabada');
      catch
        disp('Error al graficar el audio');
      end
     case 5
      disp('Saliendo del Programa');
    otherwise
      disp('Opcion no Validad');

  endswitch
```

---

* Redes

** g-mail: 3741958620101@ingenieria.usac.edu.gt

```
endwhile
```

B. Codigo Python

```

import pyaudio
import wave
import simpleaudio as sa
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import scipy.io.wavfile as wav

FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
CHUNK = 1024

opcion = 0
audio = pyaudio.PyAudio()
while opcion != 5:
    opcion += 1
    print(" 1. Grabar Audio\n 2. Reproducir Audio\n 3. Graficar\n 4. Graficar Espectro de Audio\n 5. Salir")
    opcion = int(input("\n Seleccione una Opcion : "))

    if opcion == 1:
        timpo_audio = int(input("Ingrese la duracion de Audio en Segundos(s): "))
        name_audio = input("Ingrese el nombre del archivo de Audio: ")
        stream = audio.open(format=FORMAT, channels=CHANNELS,rate = RATE, input=True, frames_per_buffer=CHUNK)
        print("Comenzando la Grabacionde Audio ...")
        frames = []
        for i in range(0, int(RATE / CHUNK * timpo_audio)):
            data = stream.read(CHUNK)
            frames.append(data)
        print("Grabacion de Audio Finalizada..." )

        stream.stop_stream()
        stream.close()
        audio.terminate()

        with wave.open(name_audio, 'wb') as wf:
            wf.setnchannels(CHANNELS)
            wf.setsampwidth(audio.get_sample_size(FORMAT))
            wf.setframerate(RATE)
            wf.writeframes(b''.join(frames))
            print(f"Archivo Guardado como: {name_audio}")

    elif opcion == 2:
        name_audio = input("Ingrese el nombre del archivo de Audio: ")
        wave_obj = sa.WaveObject.from_wave_file('/home/emerson/Documentos/Tareas_Proyectos'+'/'+name_audio)
        play_obj = wave_obj.play()
        play_obj.wait_done()

    elif opcion == 3:
        print("Graficando Audio...")

        name_audio = input("Ingrese el nombre del archivo de Audio: ")
        wf = wave.open(name_audio, 'rb')
        data = wf.readframes(wf.getnframes())
        data = np.frombuffer(data, np.int16)
        plt.plot(data)
        plt.show()
        wf.close()

    elif opcion == 4:
        name_audio = input("Ingrese el nombre del archivo de Audio:")
        Fs,audio = wav.read(name_audio)
        print("Graficando Espectro de Audio...")
        Nw = len(audio)
        f = np.linspace(0,Fs/2,Nw//2+1)
        ventana= np.hanning(Nw)

        f,Sxx = signal.welch(audio,fs=Fs,window=ventana,nperseg=Nw,noverlap=Nw/2,nfft=Nw)

        plt.plot(f,10*np.log10(Sxx))
        plt.grid(True)
        plt.xlabel('Frecuencia (Hz)')
        plt.ylabel('Densidad de Potencia Espectral (dB/Hz)')
        plt.show()
    elif opcion == 5:
        print("Saliendo...")
        break
```

III. DEPENDENCIAS INSTALADAS

```

#Dependencias Necesarias
sudo apt-get install portaudio19-dev
pip install simpleaudio
sudo apt-get install libportaudio2 liboctave-dev
     pulseaudio alsa-utils
```

IV. LIBRERIAS INSTALADAS

```
    pip install matplotlib
    pip install pyaudio
    pip install numpy
    pip install scipy
```

V. LIBRERIS UTILIZADAS

```
import pyaudio
import wave
import simpleaudio as sa
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import scipy.io.wavfile as wav
```

VI. MARCO TEORICO

Espectro de Audio

El espectro de frecuencias muestra todos los elementos sónicos individuales de un sonido. Se muestra de bajo a alto y de izquierda a derecha a lo largo del tiempo. Los respectivos niveles de todos los armónicos se reflejan verticalmente, y los picos más altos indican los niveles más elevados.

VII. RESULTADOS

A. Resultados Octave

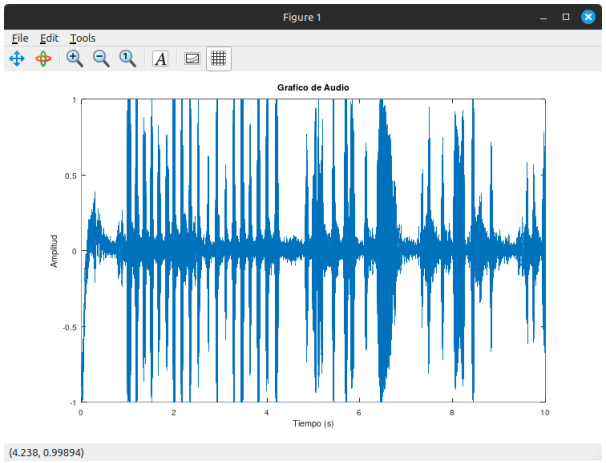


Figura 1: Grafica de Audio
Fuente: Elaboracion Propia, 2024

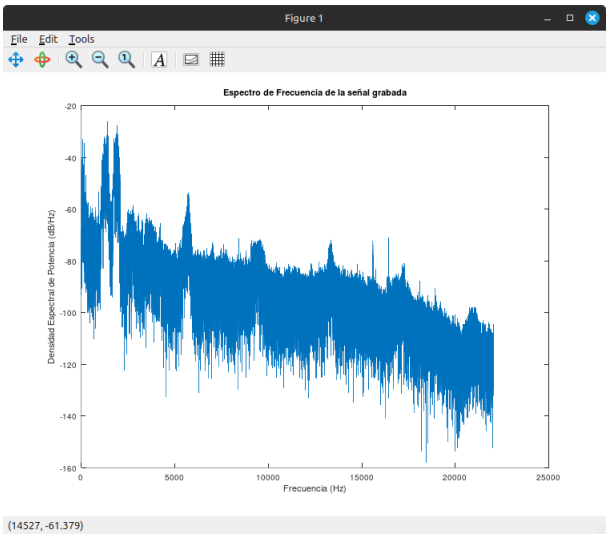


Figura 2: Grafica Espectro de Audio
Fuente: Elaboracion Propia, 2024

B. Resultados Python

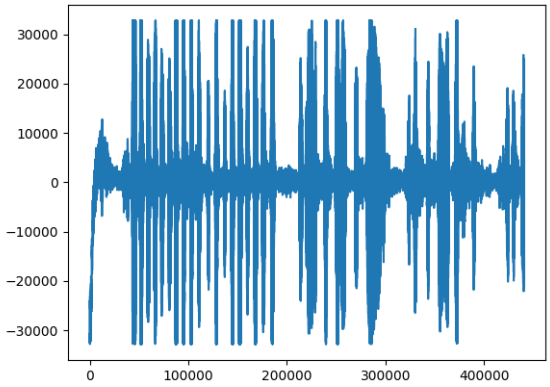


Figura 3: Grafica Audio
Fuente: Elaboracion Propia, 2024

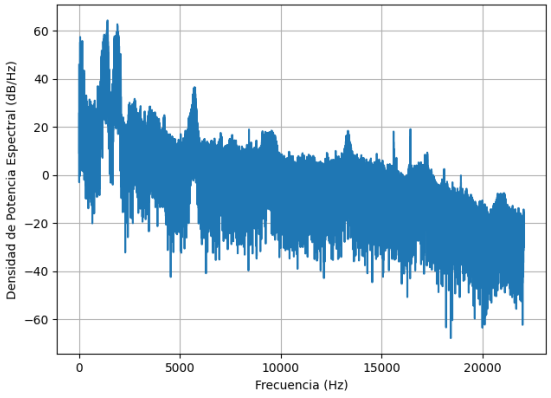


Figura 4: Grafica Espectro de Audio
Fuente: Elaboracion Propia, 2024

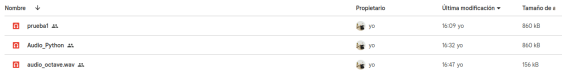
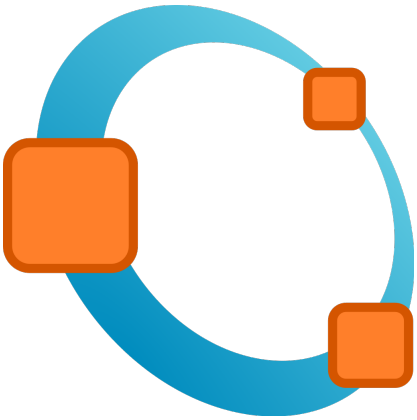


Figura 5: Grafica Espectro de Audio
Fuente: Elaboracion Propia, 2024

VIII. AUDIO

A. Audio Python

B. Audio Octave



IX. CONCLUSIONES

- 1. Python y Octave ofrecieron herramientas muy utilices para la grabación, reproducción y análisis de audio. Python destaca por su facilidad de uso y la integración de librerias para el análisis y visualización, mientras que Octave proporciona un entorno especializado para el procesamiento de señales.

- 2. Sobre reproducir el mismo codigo que estaba escrito en GNU Octave esto resulto ser algo sencillo ya que poseen similares sintaxis y que en python existen librerias que nos ayudaron a replicar el codigo.

X. REPOSITORIO GITHUB