SIGNALS AND SYSTEMS PROJECT

audio processing using python

Prepared for: Dr. Islam Shalaan
Prepared by: Nour Sherif Abuelenin & Hla Essam Dawoud

table of contents

introduction

SIGNALS AND SYSTEMS PROJECT

what are audio signals?

Audio signals are signals that vibrate in the audible frequency range. When someone talks, it generates air pressure signals; the ear takes in these air pressure differences and communicates with the brain. That's how the brain helps a person recognize that the signal is speech and understand what someone is saying.

what is audio processing?

Audio signal processing is a subfield of signal processing that is concerned with the electronic manipulation of audio signals. Audio signals are electronic representations of sound waves—longitudinal waves which travel through air, consisting of compressions and rarefactions.

The sound is typically represented as a waveform: a float or integer (quantized) array representing sound signal A(t) over the discrete time variable t. It can have multiple channels for stereo, 5.1, etc.

audio processing with python

In Python, the waveform can be numpy.ndarray or a similar format.
The waveform has sampling rate fs, a number of samples per second, e.g. 8k, 16k, 22k, 44k, 48k etc.

Sound-processing algorithms often require a fixed fs, thus if you have an input waveform of different fs, you must resample it first, i.e. interpolate the signal A(t) to a different sample rate.

anaconda and jupyter notebook

SIGNALS AND SYSTEMS PROJECT

ANACONDA

Anaconda is an open-source distribution for python and R. It is used for data science, machine learning, deep learning, etc.

With the availability of more than 300 libraries for data science, Anaconda offers the easiest way to perform Python and R data science and machine learning on a single machine.

jupyter notebook

The Jupyter Notebook is a web-based interactive computing platform that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

jupyter

libraries and packages

SIGNALS AND SYSTEMS PROJECT

libraries and packages used

- Playsound

The playsound module contains only a single function named playsound(). It requires one argument: the path to the file with the sound we have to play. It can be a local file, or a URL.

- Soundfile

Soundfile is a minimal library for reading and writing uncompressed WAV files as NumPy.ndarray plus fs waveforms.

- Librosa

Librosa is a python package with lots of sound processing, spectrograms, and such, developed for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It is the easiest and the most used in this project.

libraries and packages used

- Sounddevice

But how can we play the sound? The simplest option is SoundDevice, based on PortAudio.

- Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects

- Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It was used to plot the audio signals.

libraries and packages used

- Tkinter

Tkinter is a standard library for GUI creation. The Tkinter library is the most popular and very easy to use and it comes with many widgets (these widgets help in the creation of nice-looking GUI Applications).

- Pygame

Pygame is a Python module that works with computer graphics and sound libraries and is designed with the power of playing with different multimedia formats like audio, video, etc. While creating our Music Player application, we will be using Pygame's mixer.music module for providing different functionality to our music player application that is usually related to the manipulation of the song tracks.

python code process

SIGNALS AND SYSTEMS PROJECT

recorded audio functions

- Record Audio Function

```
def recordaudio():
  import sounddevice
  from scipy.io.wavfile import write
  from playsound import playsound
  #44100 or 48000 is used frequently in CDs and computer audio , there are other more commom frequency samples
  fs = 44100
  #duration of record
  second = 15
  print("recording....")
  #sounddevice.rec records an audio and save it in a form of numpy array
  record_voice = sounddevice.rec(int(second*fs),samplerate=fs,channels=2)
  sounddevice.wait()
  #write(filnename,rate,data) which converts a numpy array into a wav file
  write('record.wav',fs,record_voice)
  print("playing record....")
  #Play record saved as wav file
  playsound('directory')
```

recorded audio functions

- Noise Reduction Function

```
def noisereduction():
   #loading the pre-recorded WAV file
   filename = ('record.wav')
   y0, sr0 = librosa.load(filename)
   #play pre-recorded audio as an array
   display(IPython.display.Audio(data=y0, rate=sr0))
   #reduce noise by median
   def reduce_noise_median(y, sr):
      y = sp.signal.medfilt(y,3)
      return (y)
   wavfile.write("mywav_reduced_noise1.wav", sr0,
reduce_noise_median(y0, sr0))
   #loading the 1st noise reduced WAV file
   filename = ('mywav_reduced_noise1.wav')
   y1, sr1 = librosa.load(filename)
```

```
   #perform 2nd noise reduction by noisereduce
   reduced_noise = nr.reduce_noise(  y=y1, sr=sr1,
thresh_n_mult_nonstationary=2, stationary=False, n_jobs=2, )
   wavfile.write("mywav_reduced_noise2.wav", sr1,
reduced_noise)

   #loading the final noise reduced WAV file
   filename = ('mywav_reduced_noise2.wav')
   y2, sr2 = librosa.load(filename)

   #play pre-recorded audio after noise reduction as an array
   display(IPython.display.Audio(data=y2, rate=sr2))
```

recorded audio functions

- Trim Silence Function

```
def trimsilence():

   #loading the final noise reduced WAV file
   filename = ('mywav_reduced_noise.wav')
   y2, sr2 = librosa.load(filename)
   #trim the beginning and ending silence
   yt, index = librosa.effects.trim(y2)
   #print the durations
   print(librosa.get_duration(y2), librosa.get_duration(yt))
   from IPython.display import Audio
   wave_audio_trim = np.sin(yt)
   display(Audio(wave_audio_trim, rate=20000))
```

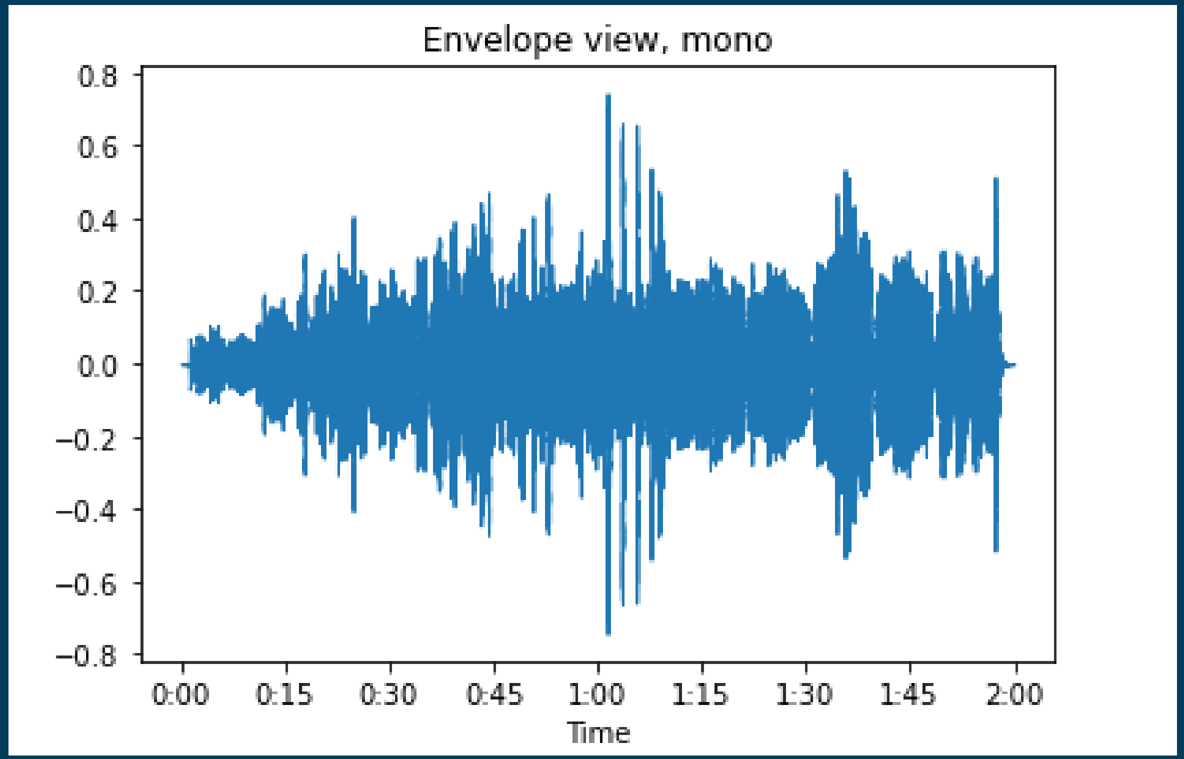
audio functions

- Pitch Shift (Up and Down) Function

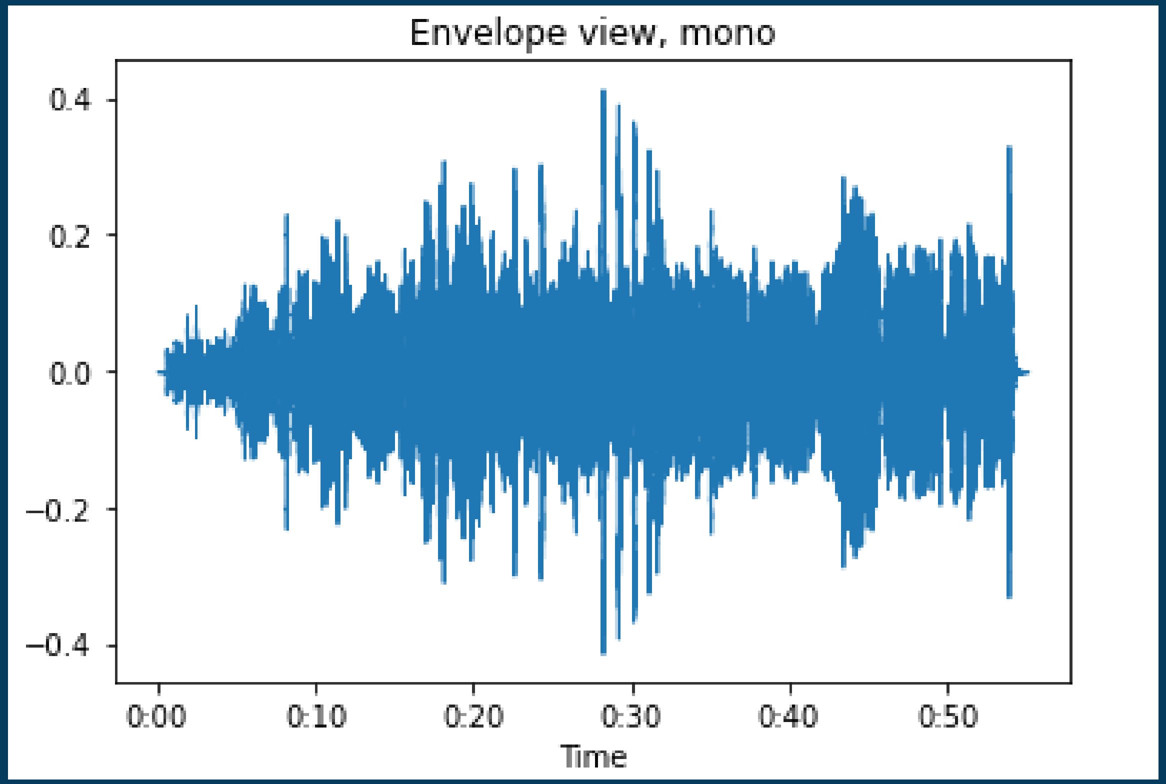
```
def pitchshiftdown():
    #loading the WAV file
    filename = librosa.example('nutcracker')
    y, sr = librosa.load(filename)

    #pitch shift by -5, 5 octaves down (frequency and pitch decrease)
    y_tritone = librosa.effects.pitch_shift(y, sr=sr, n_steps=-5)

    #play the shifted audio
    from IPython.display import Audio
    wave_audio1 = np.sin(y_tritone)
    display(Audio(wave_audio1, rate=20000))

    #save 1st shifted audio as a WAV file
    import soundfile as sf
    sf.write('wave_audio1.wav',wave_audio1,48000)
```

audio functions

- Pitch Shift (Up and Down) Function - Ploting



Before



After

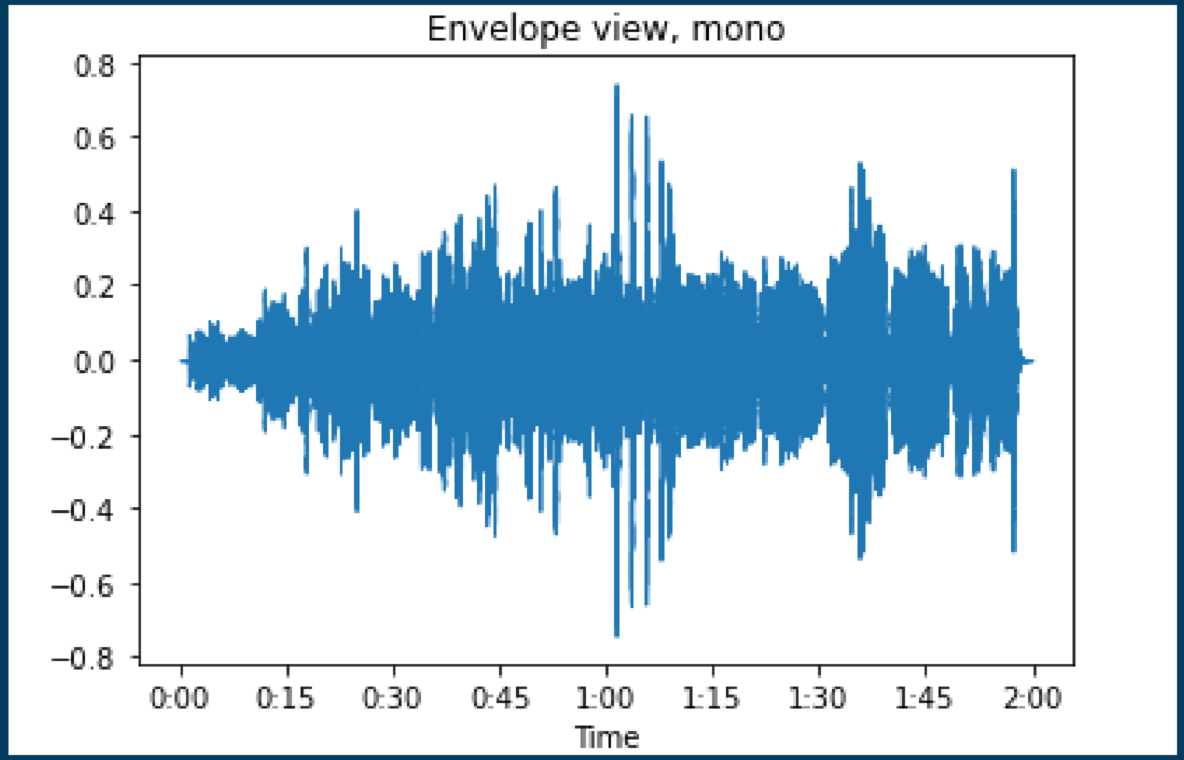
audio functions

- Pitch Shift (Up and Down) Function

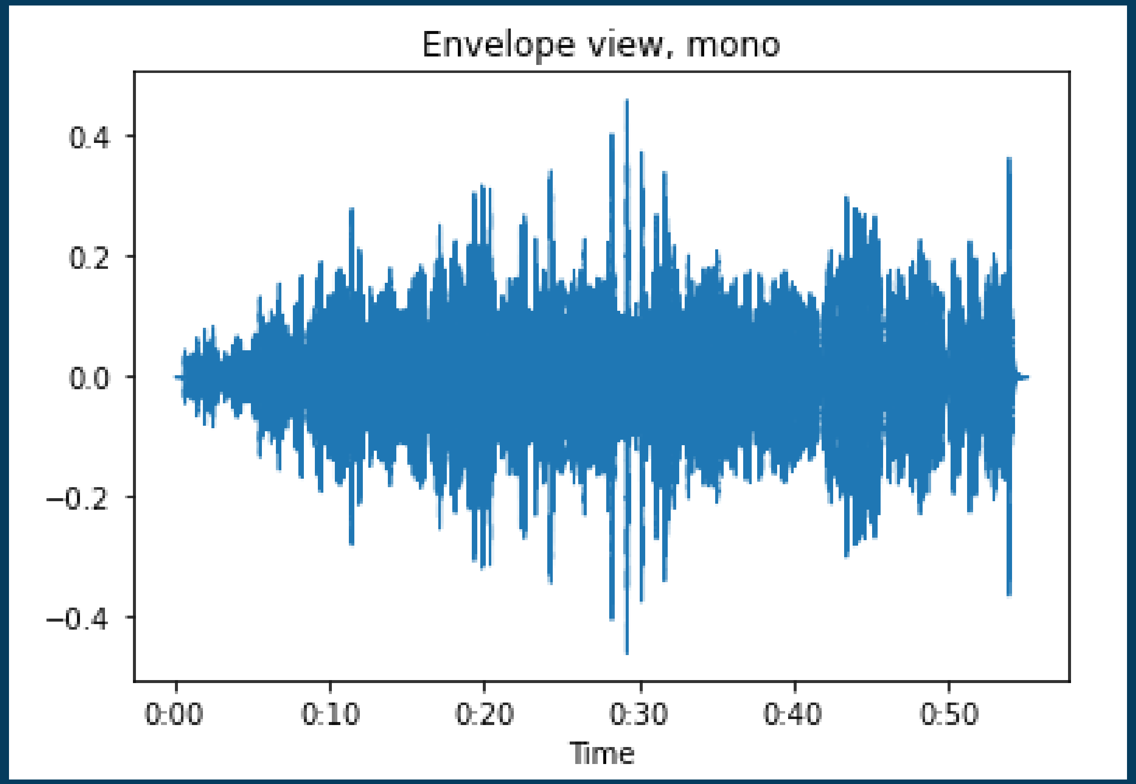
```
def pitchshiftup():
    #loading the WAV file
    filename = librosa.example('nutcracker')
    y, sr = librosa.load(filename)

    #pitch shift by 5, 5 octaves up (frequency and pitch increase)
    y_third = librosa.effects.pitch_shift(y, sr=sr, n_steps=5)

    #play the 2nd shifted audio
    from IPython.display import Audio
    wave_audio2 = np.sin(y_third)
    display(Audio(wave_audio2, rate=20000))

    #save 2nd shifted audio as a WAV file
    import soundfile as sf
    sf.write('wave_audio2.wav',wave_audio2,48000)
```

audio functions

- Pitch Shift (Up and Down) Function - Ploting



BEFORE



AFTER

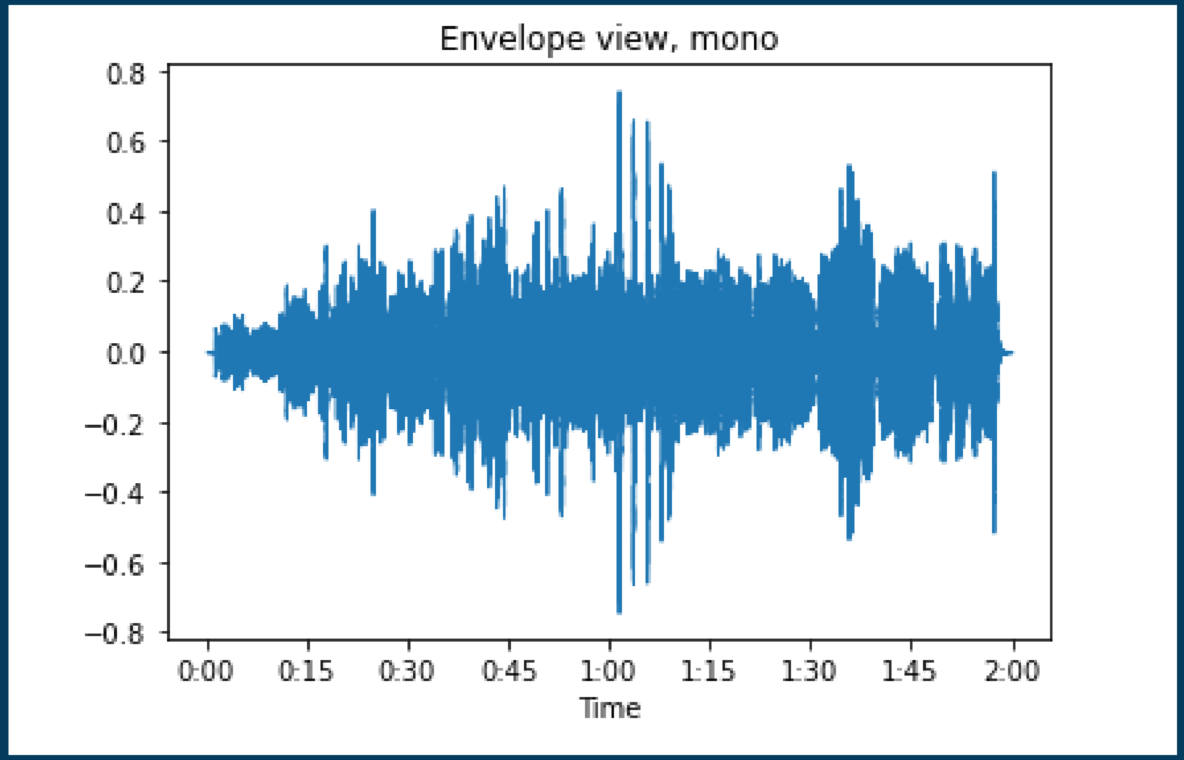
audio functions

- Speed-Up Audio Function

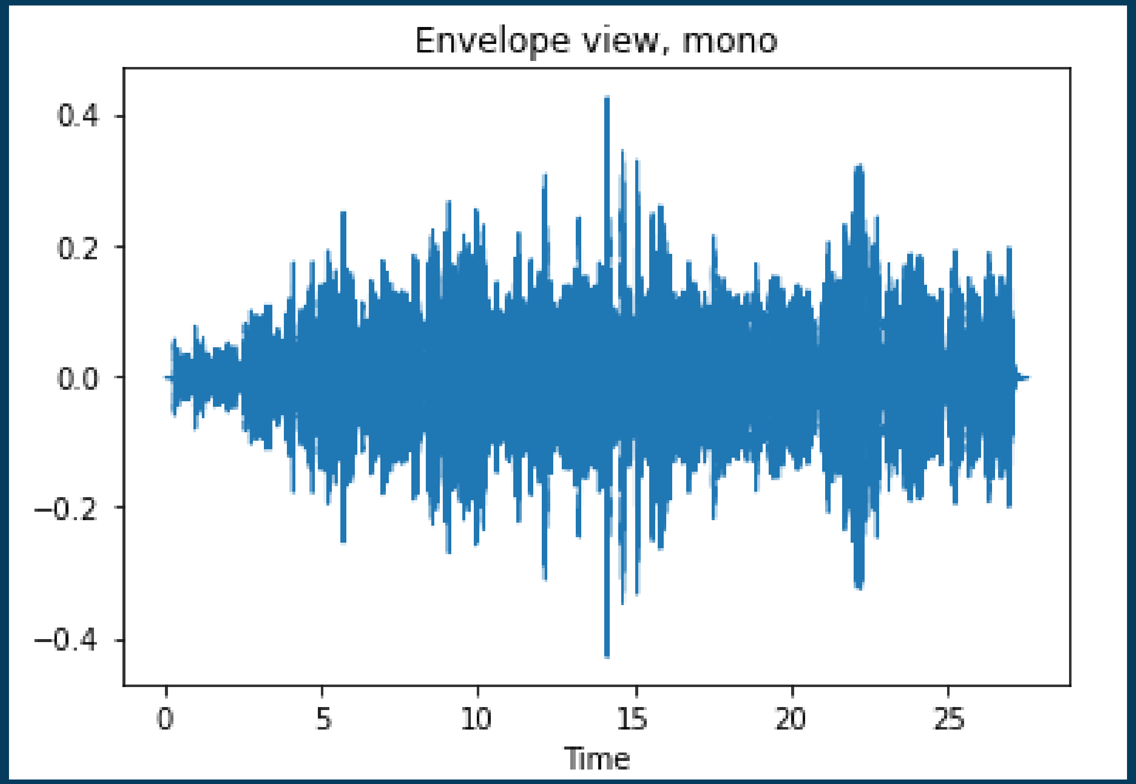
```
def fast():
   #loading the WAV file
   filename = librosa.example('nutcracker')
   y, sr = librosa.load(filename)

   #speed-up audio (compresses the audio to be twice as fast)
   y_fast = librosa.effects.time_stretch(y, rate=2.0)

    #play the 2nd shifted audio
    from IPython.display import Audio
    wave_audio3 = np.sin(y_fast)
    display(Audio(wave_audio3, rate=20000))

   #save the sped-up audio as a WAV file
   import soundfile as sf
   sf.write('wave_audio3.wav',wave_audio3,48000)
```

audio functions

- Speed-Up Audio Function - Ploting



Before



After

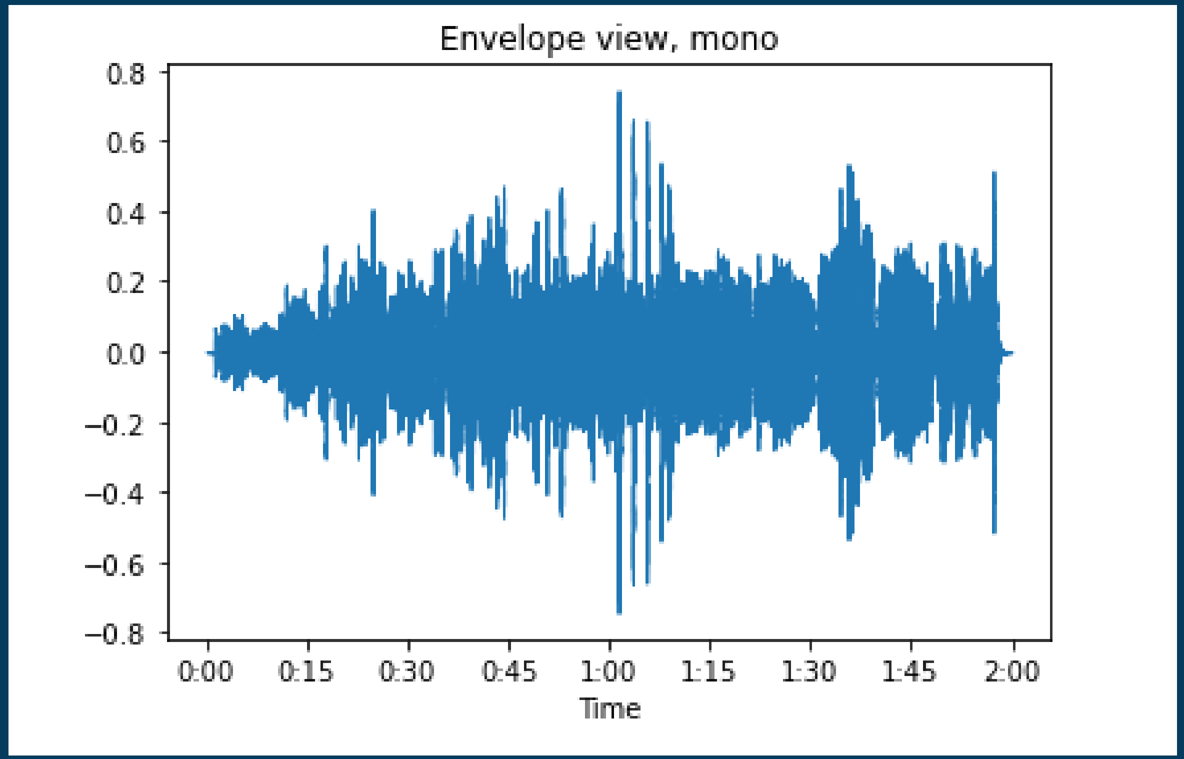
audio functions

- Slowed-Down Audio Function

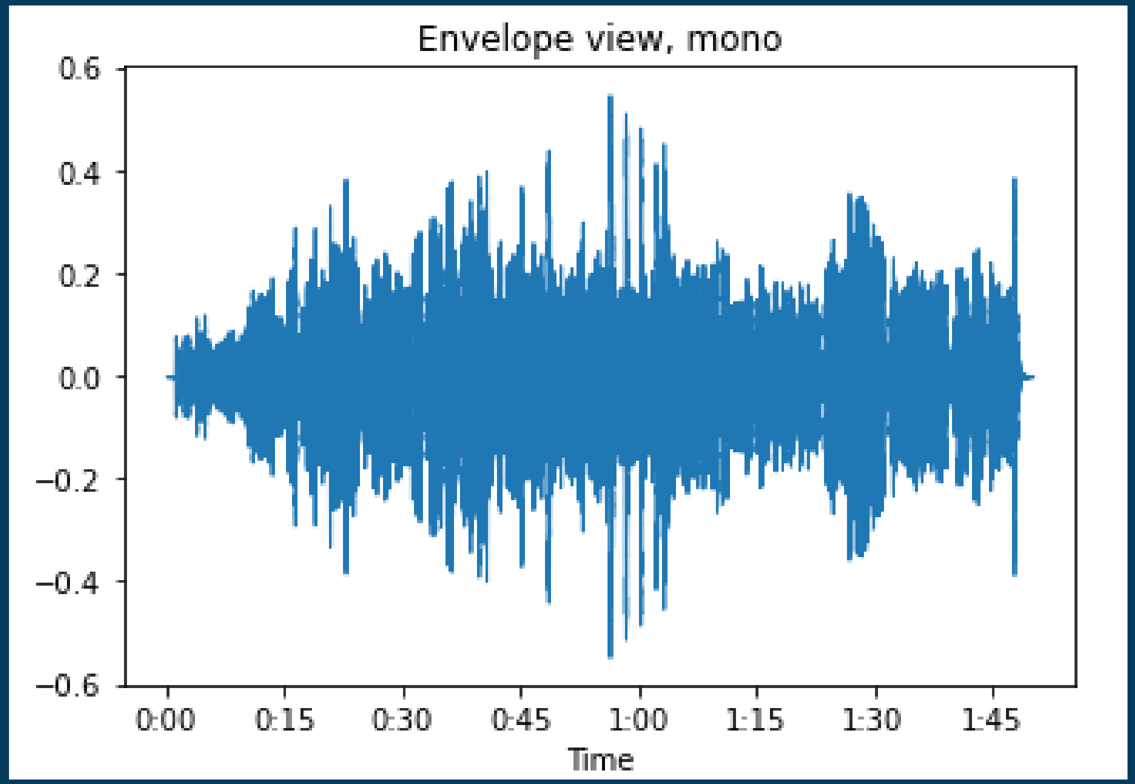
```
def slow():
    #loading the WAV file
    filename = librosa.example('nutcracker')
    y, sr = librosa.load(filename)

     #slow-down audio (compresses the audio to half the original speed)
     y_slow = librosa.effects.time_stretch(y, rate=0.5)

    #play the 2nd shifted audio
    from IPython.display import Audio
    wave_audio4 = np.sin(y_slow)
    display(Audio(wave_audio4, rate=20000))

    #save the slowed-down audio as a WAV file
    import soundfile as sf
    sf.write('wave_audio4.wav',wave_audio4,48000)
```

audio functions

- Slowed-Down Audio Function - Ploting



BEFORE



AFTER

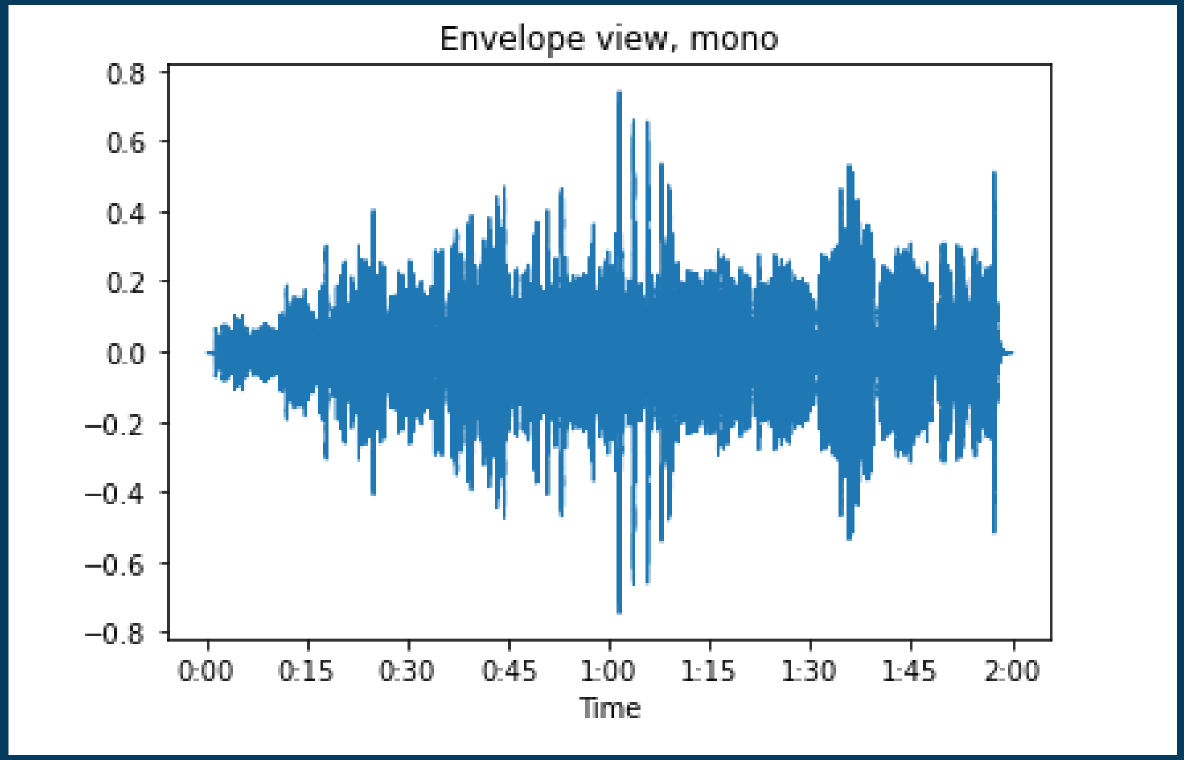
audio functions

- Remix Audio Function

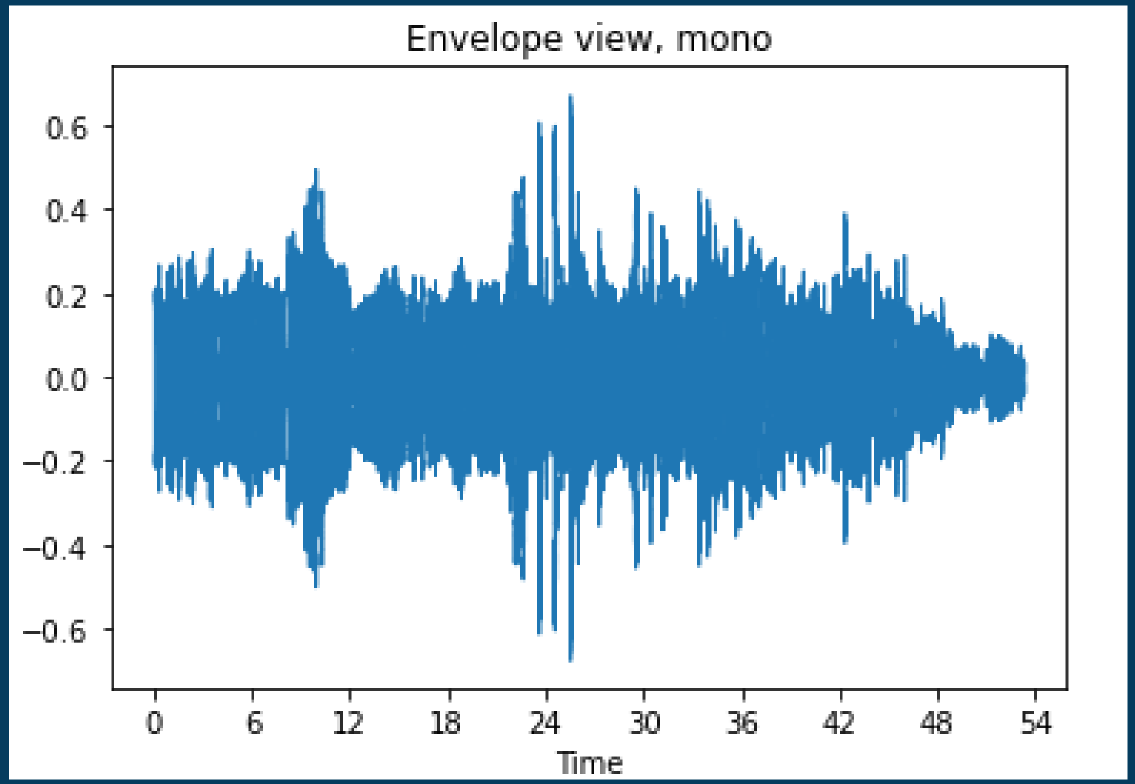
```
def remix():
    #loading the WAV file
    filename = librosa.example('nutcracker')
    y, sr = librosa.load(filename)
    #compute beats
    _, beat_frames = librosa.beat.beat_track(y=y, sr=sr, hop_length=512)
    #convert from frames to sample indices
    beat_samples = librosa.frames_to_samples(beat_frames)
    #generate intervals from consecutive events
    intervals = librosa.util.frame(beat_samples, frame_length=2,hop_length=1).T
    #reverse the beat intervals
    y_out = librosa.effects.remix(y, intervals[::-1])
    #play the remix audio
    from IPython.display import Audio
    wave_audio_remix = np.sin(y_out)
    display(Audio(wave_audio_remix, rate=20000))
```

audio functions

- Remix Audio Function - Ploting



Before



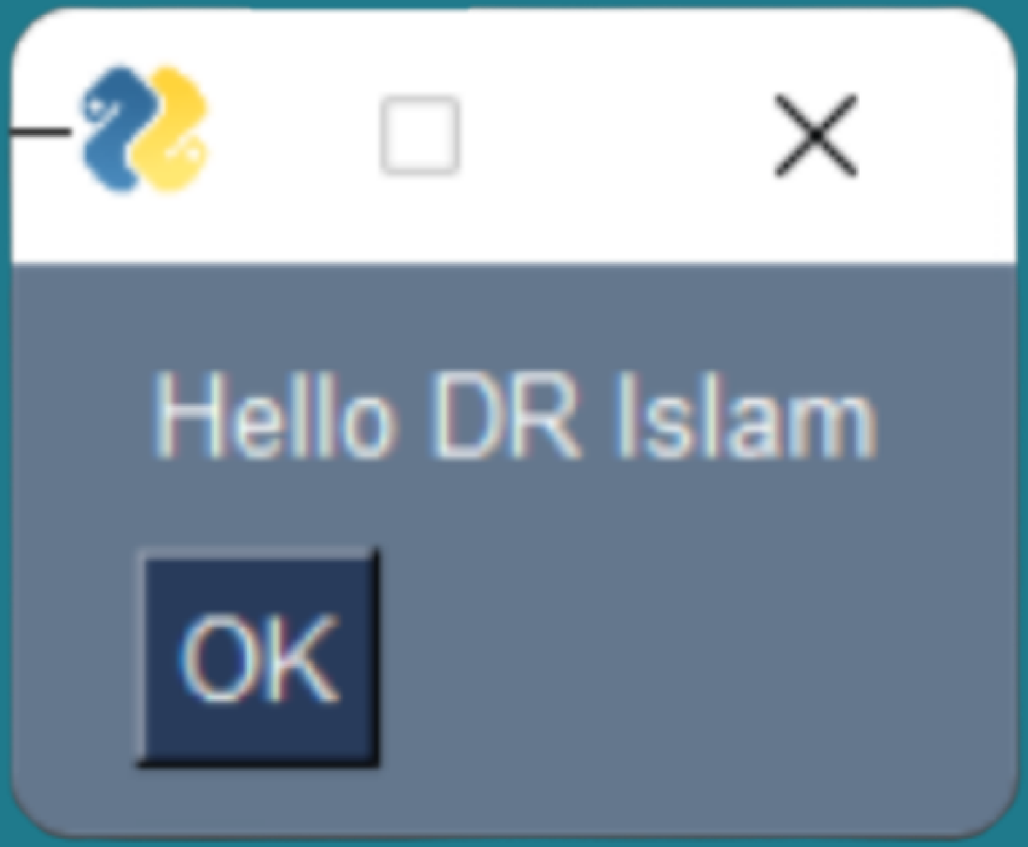
After

graphical user interface

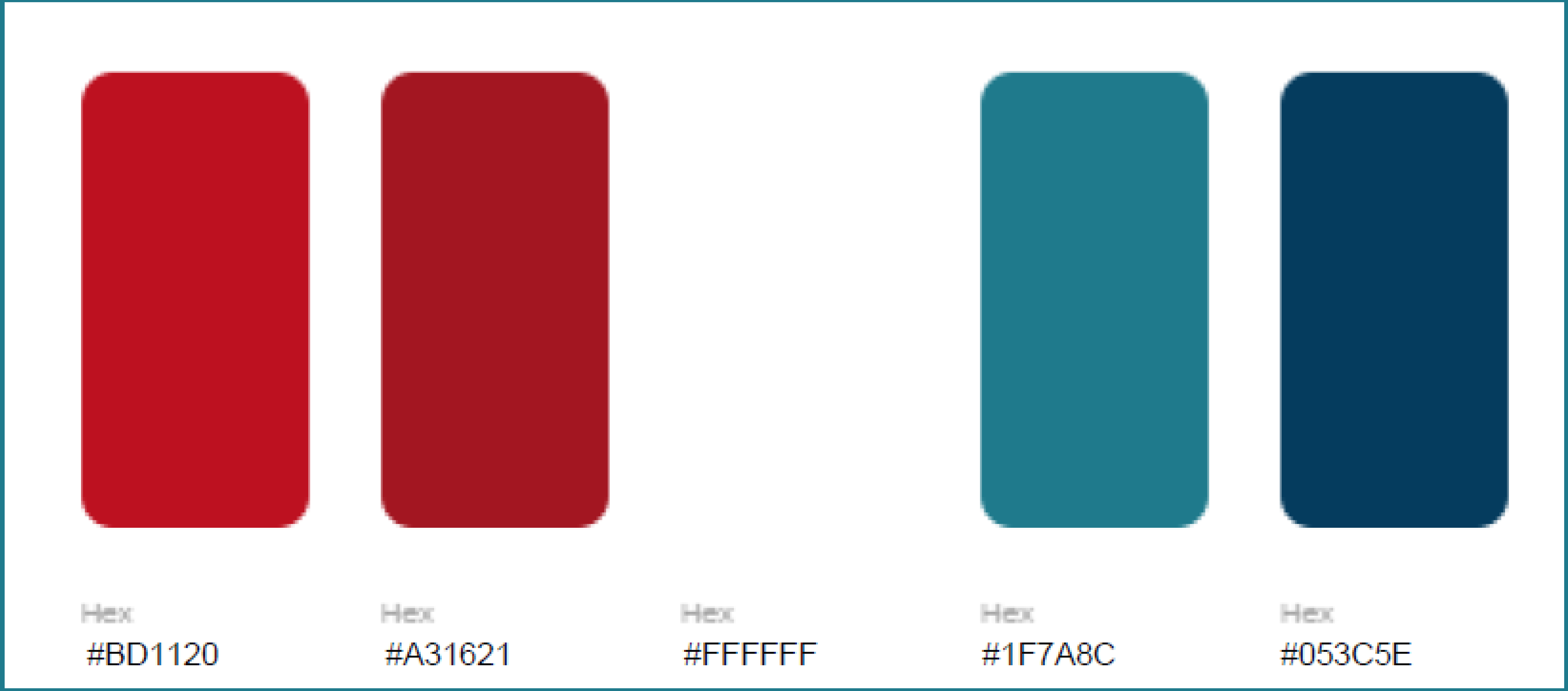
SIGNALS AND SYSTEMS PROJECT

what is graphical user interface?

The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators such as primary notation, instead of text-based UIs, typed command labels, or text navigation.



gui color palette



nour & hla's music player

thank you!