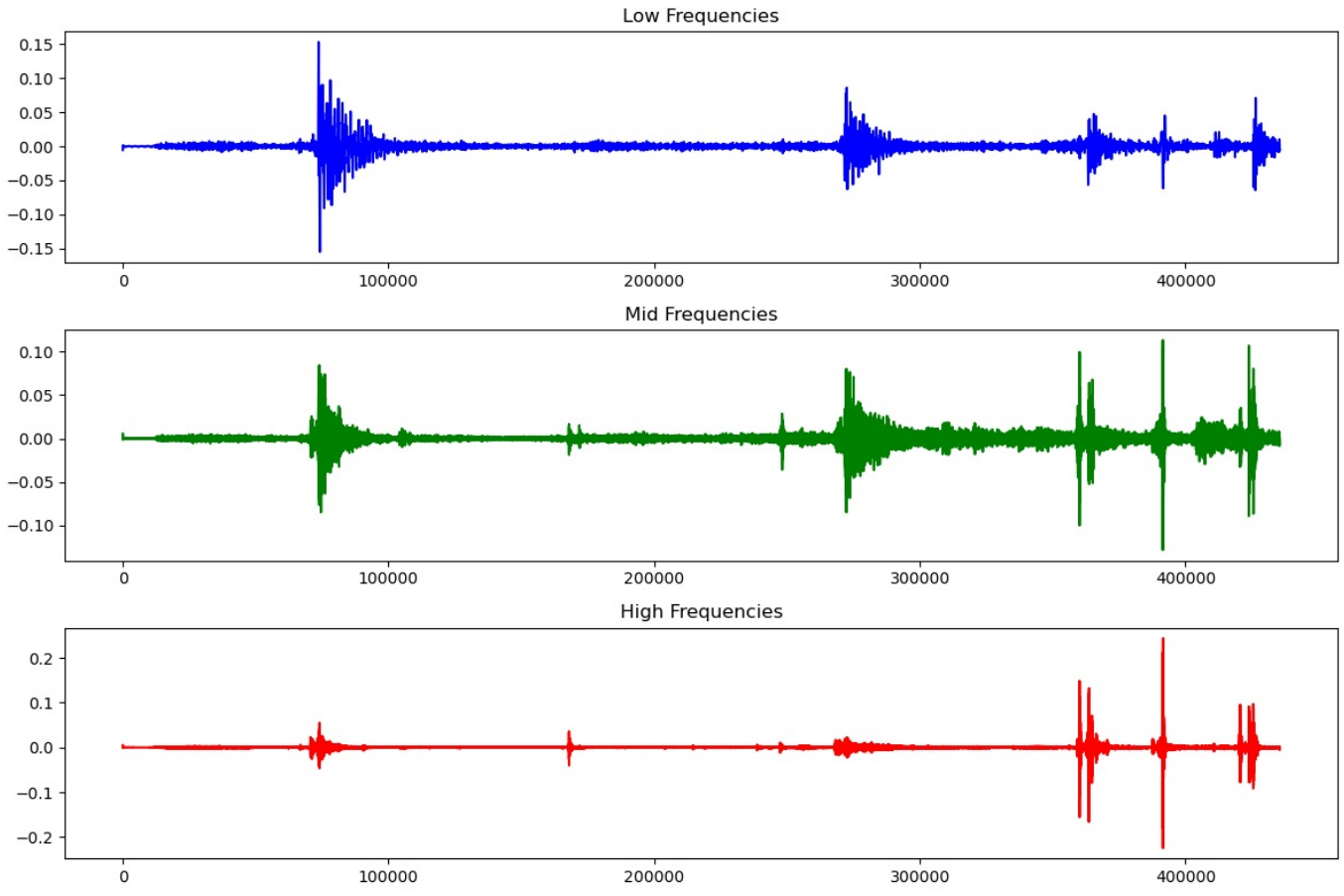
In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
file_path = r'C:\Users\ASUS\doorclosing.wav'
audio, sr = librosa.load(file_path, sr=None)

# Perform FFT on the audio signal
fft_audio = np.fft.fft(audio)
frequencies = np.fft.fftfreq(len(audio), 1/sr)

# Define frequency ranges for low, mid, and high
low_freq = (frequencies > 20) & (frequencies <= 300)
mid_freq = (frequencies > 300) & (frequencies <= 2000)
high_freq = (frequencies > 2000) & (frequencies <= sr / 2)

# Create separate FFT arrays for low, mid, and high frequencies
fft_low = np.zeros_like(fft_audio)
fft_mid = np.zeros_like(fft_audio)
fft_high = np.zeros_like(fft_audio)

# Assign values in the frequency ranges
fft_low[low_freq] = fft_audio[low_freq]
fft_mid[mid_freq] = fft_audio[mid_freq]
fft_high[high_freq] = fft_audio[high_freq]
# Perform inverse FFT to get the time-domain signals
audio_low = np.fft.ifft(fft_low).real
audio_mid = np.fft.ifft(fft_mid).real
audio_high = np.fft.ifft(fft_high).real
```

In [5]:
```
# Plot the signals
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(audio_low, color='blue')
plt.title('Low Frequencies')

plt.subplot(3, 1, 2)
plt.plot(audio_mid, color='green')
plt.title('Mid Frequencies')

plt.subplot(3, 1, 3)
plt.plot(audio_high, color='red')
plt.title('High Frequencies')

plt.tight_layout()
plt.show()
```



In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
file_path = r'C:\Users\ASUS\keyboard.wav'
audio, sr = librosa.load(file_path, sr=None)
```

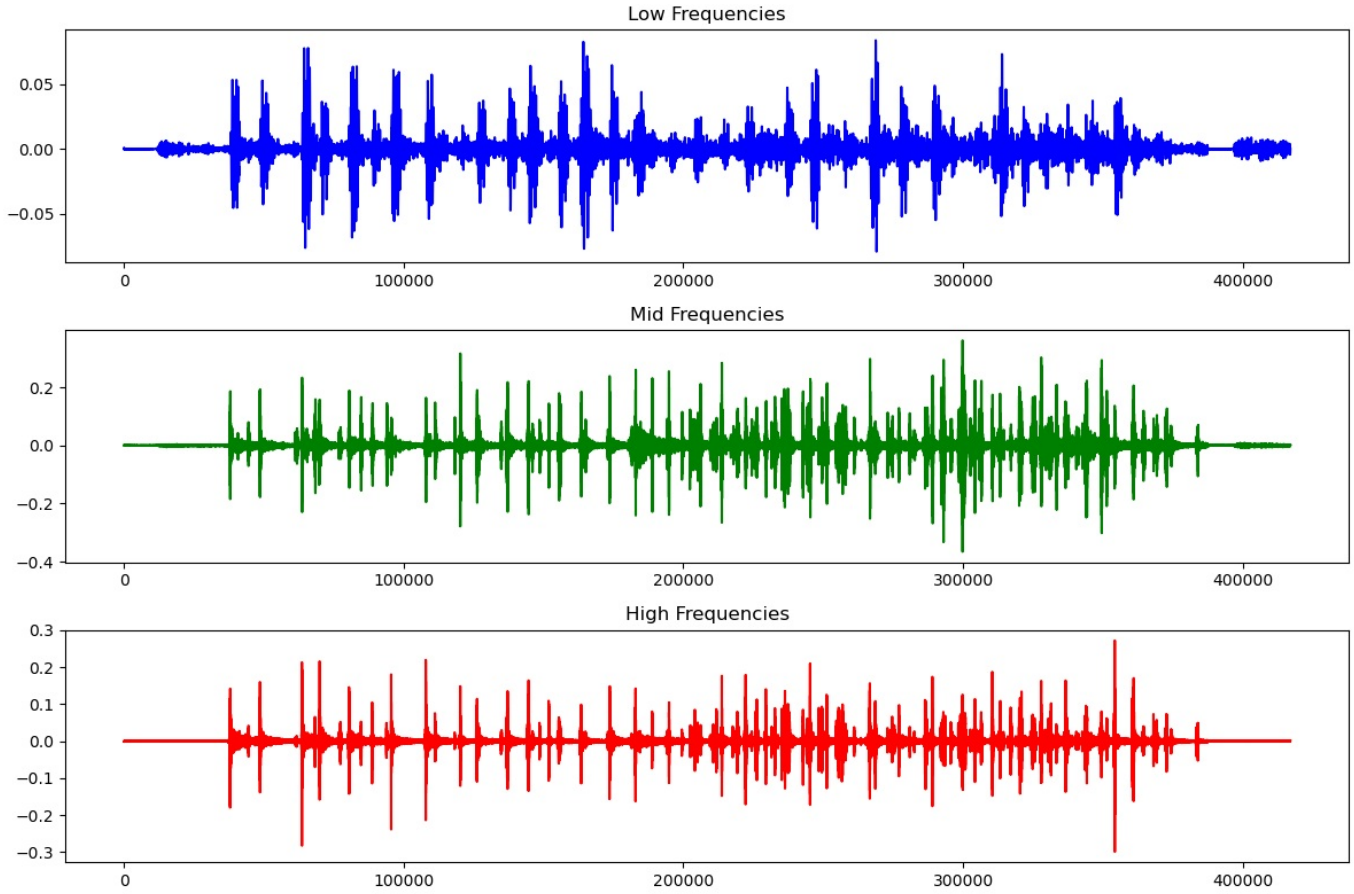
In [5]:
```
# Perform FFT on the audio signal
fft_audio = np.fft.fft(audio)
frequencies = np.fft.fftfreq(len(audio), 1/sr)
```

In [7]:
```
# Define frequency ranges for low, mid, and high
low_freq = (frequencies > 20) & (frequencies <= 300)
mid_freq = (frequencies > 300) & (frequencies <= 2000)
high_freq = (frequencies > 2000) & (frequencies <= sr / 2)
```

In [9]:
```
# Create separate FFT arrays for low, mid, and high frequencies
fft_low = np.zeros_like(fft_audio)
fft_mid = np.zeros_like(fft_audio)
fft_high = np.zeros_like(fft_audio)
```

In [15]:
```
# Assign values in the frequency ranges
fft_low[low_freq] = fft_audio[low_freq]
fft_mid[mid_freq] = fft_audio[mid_freq]
fft_high[high_freq] = fft_audio[high_freq]
# Perform inverse FFT to get the time-domain signals
audio_low = np.fft.ifft(fft_low).real
audio_mid = np.fft.ifft(fft_mid).real
audio_high = np.fft.ifft(fft_high).real
```

In [17]:
```
# Plot the signals
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(audio_low, color='blue')
plt.title('Low Frequencies')

plt.subplot(3, 1, 2)
plt.plot(audio_mid, color='green')
plt.title('Mid Frequencies')

plt.subplot(3, 1, 3)
plt.plot(audio_high, color='red')
plt.title('High Frequencies')

plt.tight_layout()
plt.show()
```

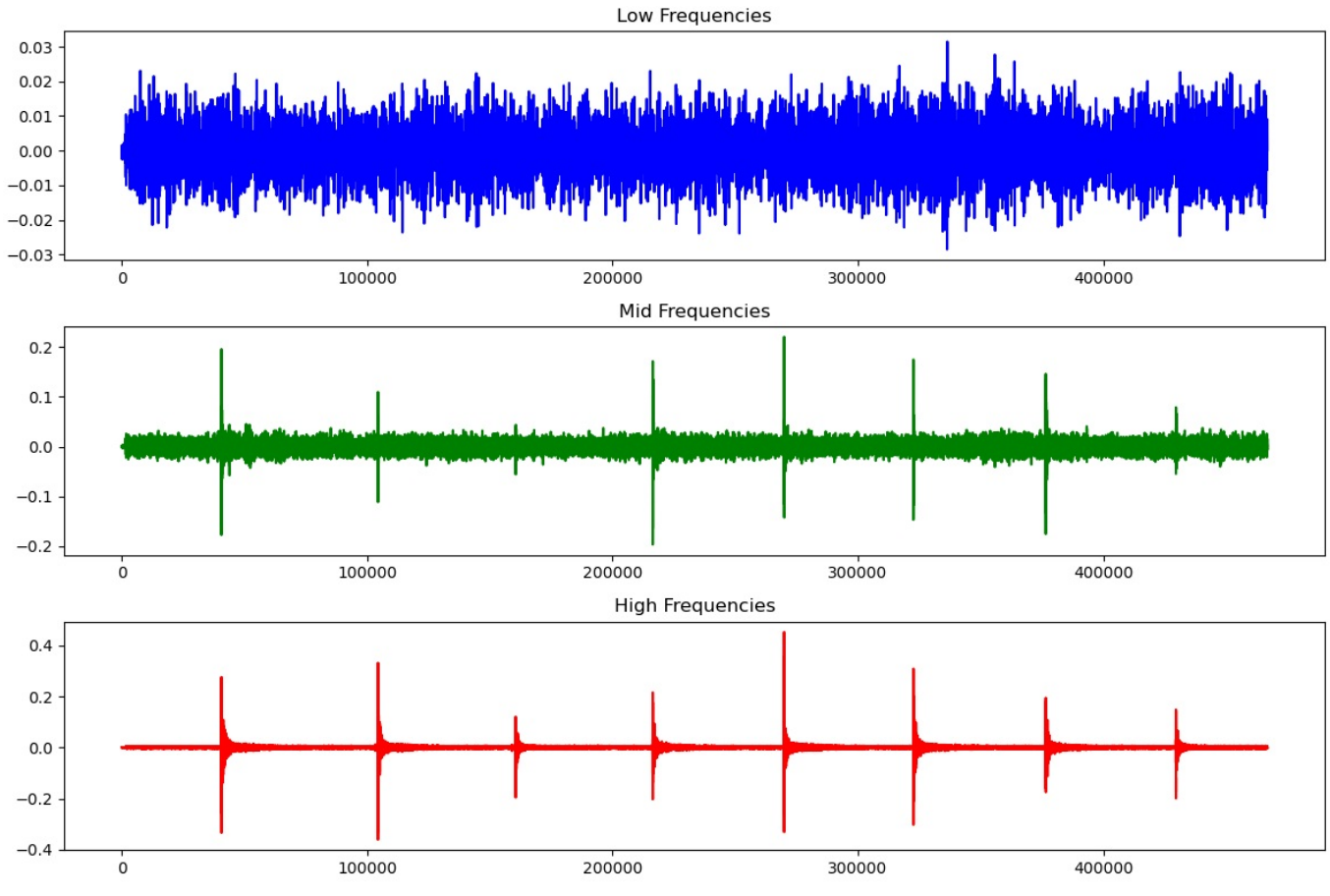


In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
file_path = r'C:\Users\ASUS\switches.wav'
audio, sr = librosa.load(file_path, sr=None)

# Perform FFT on the audio signal
fft_audio = np.fft.fft(audio)
frequencies = np.fft.fftfreq(len(audio), 1/sr)

# Define frequency ranges for low, mid, and high
low_freq = (frequencies > 20) & (frequencies <= 300)
mid_freq = (frequencies > 300) & (frequencies <= 2000)
high_freq = (frequencies > 2000) & (frequencies <= sr / 2)

# Create separate FFT arrays for low, mid, and high frequencies
fft_low = np.zeros_like(fft_audio)
fft_mid = np.zeros_like(fft_audio)
fft_high = np.zeros_like(fft_audio)

# Assign values in the frequency ranges
fft_low[low_freq] = fft_audio[low_freq]
fft_mid[mid_freq] = fft_audio[mid_freq]
fft_high[high_freq] = fft_audio[high_freq]
# Perform inverse FFT to get the time-domain signals
audio_low = np.fft.ifft(fft_low).real
audio_mid = np.fft.ifft(fft_mid).real
audio_high = np.fft.ifft(fft_high).real
```

In [5]:
```
# Plot the signals
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(audio_low, color='blue')
plt.title('Low Frequencies')

plt.subplot(3, 1, 2)
plt.plot(audio_mid, color='green')
plt.title('Mid Frequencies')

plt.subplot(3, 1, 3)
plt.plot(audio_high, color='red')
plt.title('High Frequencies')

plt.tight_layout()
plt.show()
```

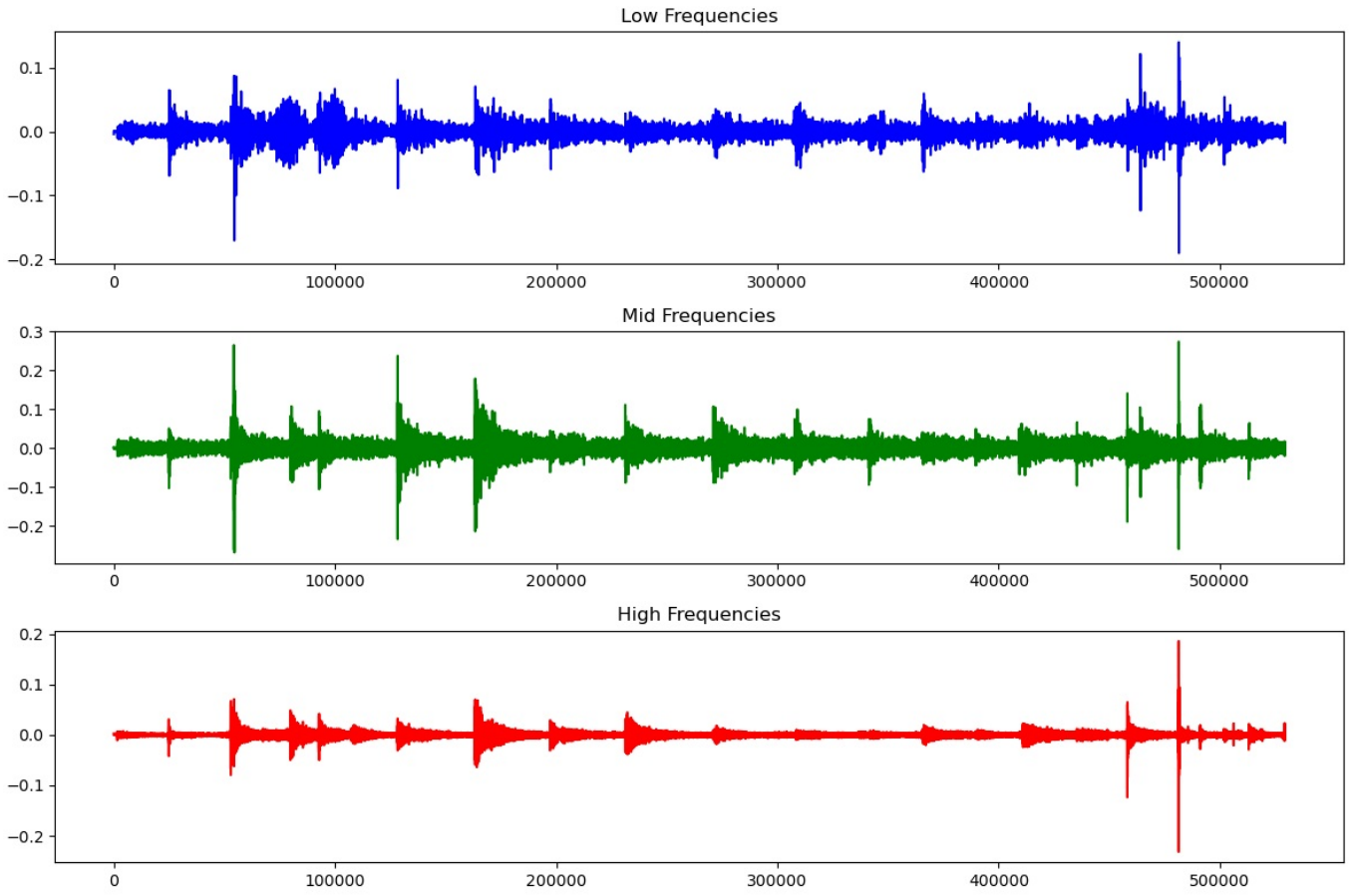


In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
file_path = r'C:\Users\ASUS\walking.wav'
audio, sr = librosa.load(file_path, sr=None)

# Perform FFT on the audio signal
fft_audio = np.fft.fft(audio)
frequencies = np.fft.fftfreq(len(audio), 1/sr)

# Define frequency ranges for low, mid, and high
low_freq = (frequencies > 20) & (frequencies <= 300)
mid_freq = (frequencies > 300) & (frequencies <= 2000)
high_freq = (frequencies > 2000) & (frequencies <= sr / 2)

# Create separate FFT arrays for low, mid, and high frequencies
fft_low = np.zeros_like(fft_audio)
fft_mid = np.zeros_like(fft_audio)
fft_high = np.zeros_like(fft_audio)

# Assign values in the frequency ranges
fft_low[low_freq] = fft_audio[low_freq]
fft_mid[mid_freq] = fft_audio[mid_freq]
fft_high[high_freq] = fft_audio[high_freq]
# Perform inverse FFT to get the time-domain signals
audio_low = np.fft.ifft(fft_low).real
audio_mid = np.fft.ifft(fft_mid).real
audio_high = np.fft.ifft(fft_high).real
```

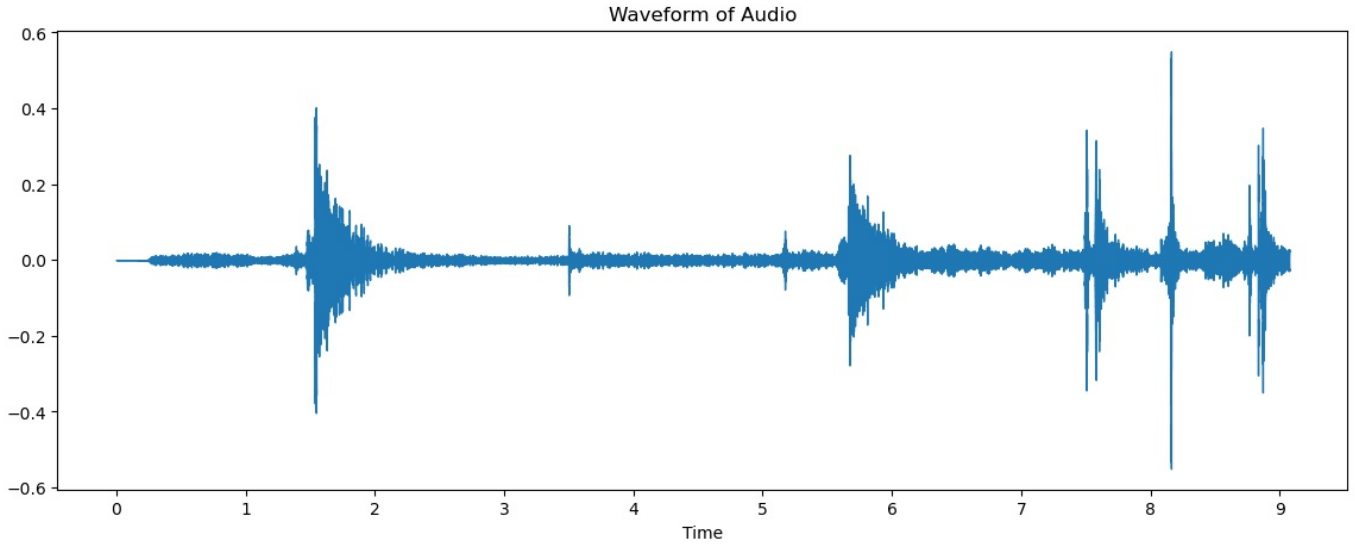
In [4]:
```
# Plot the signals
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.plot(audio_low, color='blue')
plt.title('Low Frequencies')

plt.subplot(3, 1, 2)
plt.plot(audio_mid, color='green')
plt.title('Mid Frequencies')

plt.subplot(3, 1, 3)
plt.plot(audio_high, color='red')
plt.title('High Frequencies')

plt.tight_layout()
plt.show()
```



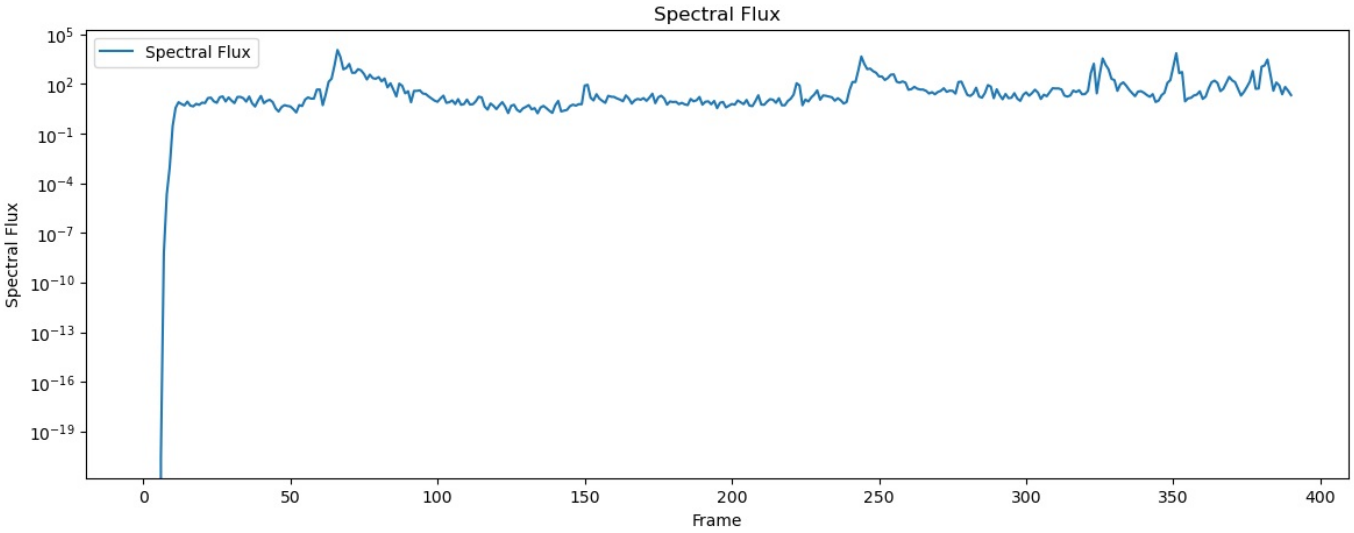
In [ ]:

In [1]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the audio file
audio_path = r'C:/Users/ASUS/doorclosing.wav'
y, sr = librosa.load(audio_path)
# Plot the waveform
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [5]:
```
import numpy as np

# Compute Short-Time Fourier Transform (STFT)
D = librosa.stft(y)

# Compute the magnitude spectrogram
S = np.abs(D)**2

# Compute spectral flux
def spectral_flux(S):
    # Shift S to get previous frame
    S_prev = np.roll(S, shift=1, axis=1)
    # Calculate the flux
    flux = np.sum(np.maximum(S - S_prev, 0), axis=0)
    return flux

# Compute the spectral flux
flux = spectral_flux(S)

# Plot the spectral flux
plt.figure(figsize=(14, 5))
plt.semilogy(flux, label='Spectral Flux')
plt.title('Spectral Flux')
plt.xlabel('Frame')
plt.ylabel('Spectral Flux')
plt.legend()
plt.show()
```

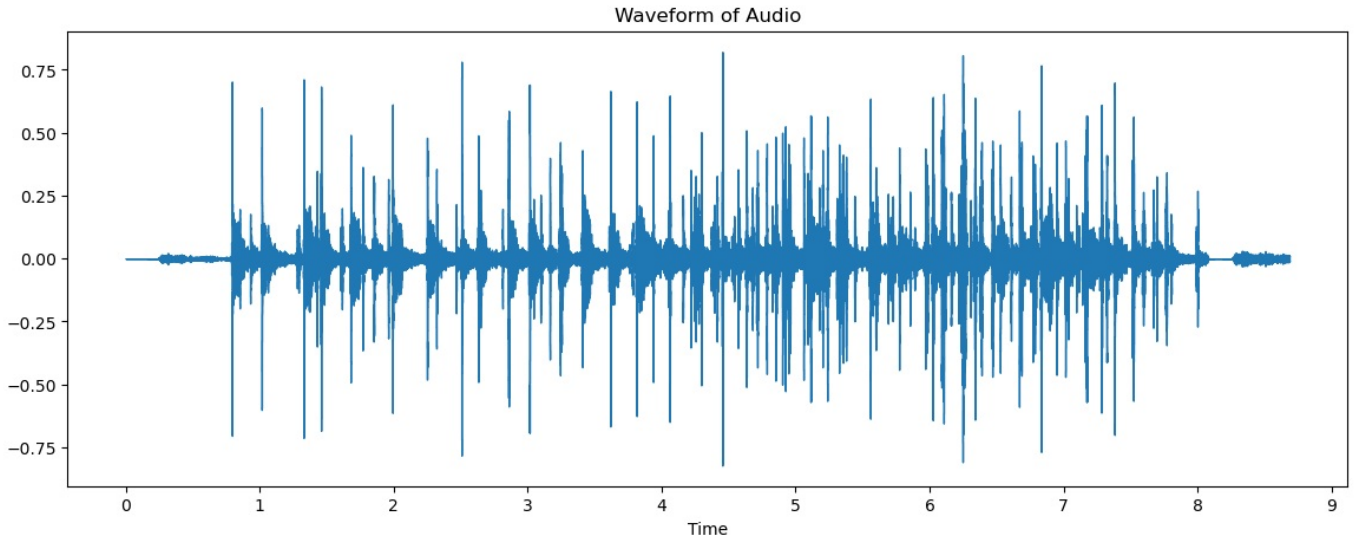


In [ ]:

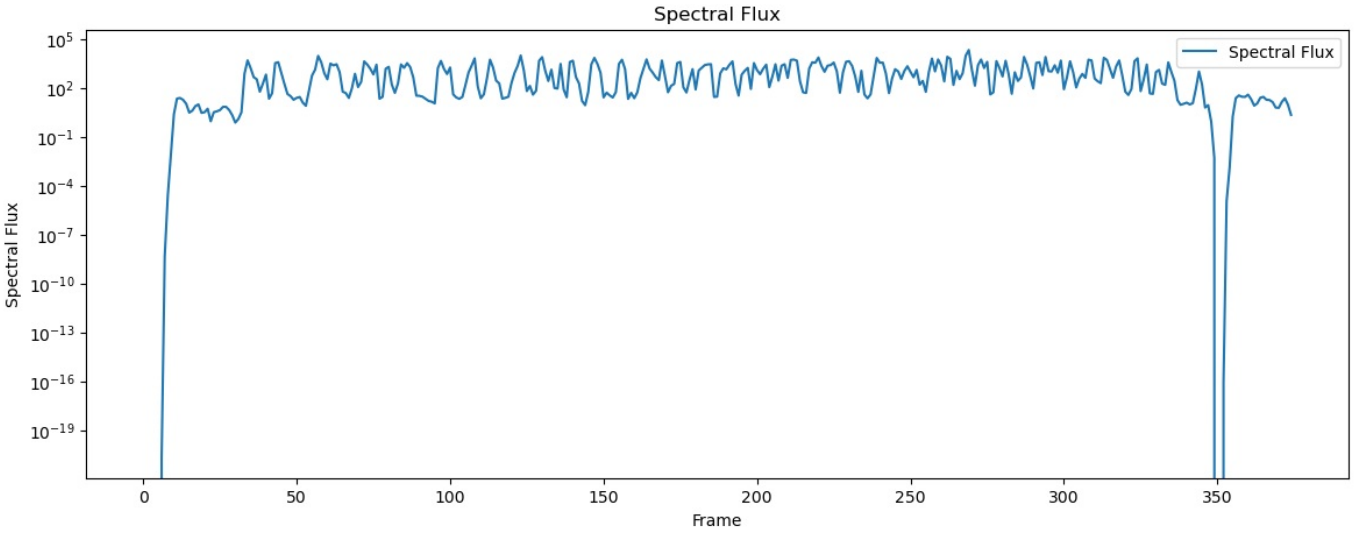
In [1]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
```

In [4]:
```
# Load the audio file
audio_path = r'C:/Users/ASUS/keyboard.wav'  # Use the correct path
y, sr = librosa.load(audio_path)
```

In [6]:
```
# Plot the waveform
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



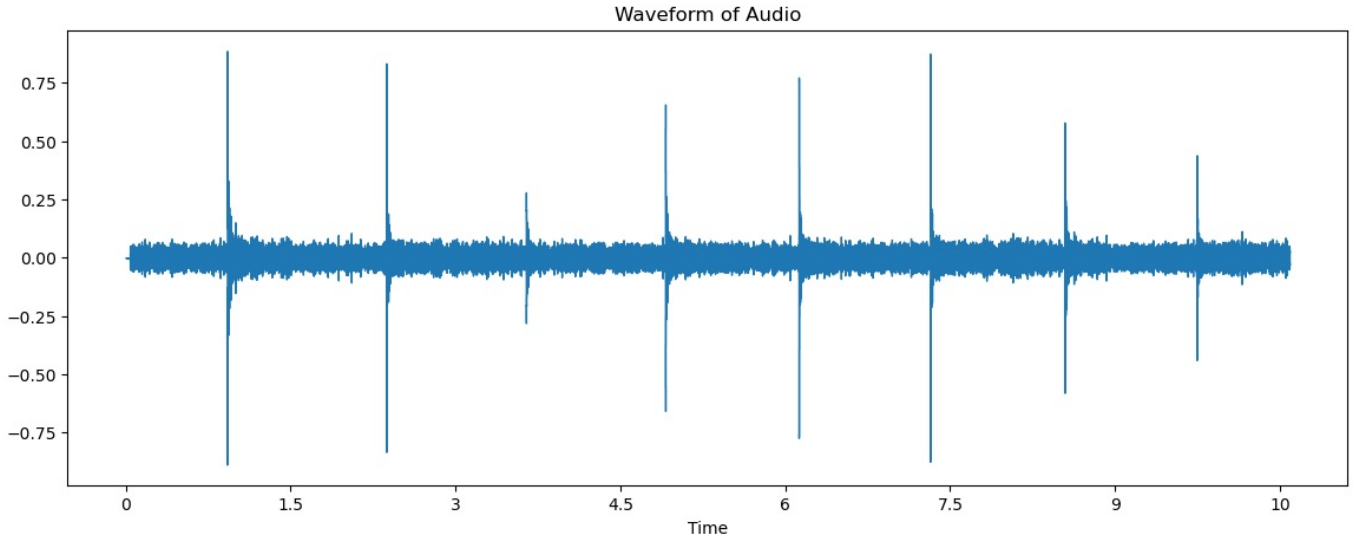
In [8]:
```
import numpy as np

# Compute Short-Time Fourier Transform (STFT)
D = librosa.stft(y)

# Compute the magnitude spectrogram
S = np.abs(D)**2

# Compute spectral flux
def spectral_flux(S):
    # Shift S to get previous frame
    S_prev = np.roll(S, shift=1, axis=1)
    # Calculate the flux
    flux = np.sum(np.maximum(S - S_prev, 0), axis=0)
    return flux

# Compute the spectral flux
flux = spectral_flux(S)

# Plot the spectral flux
plt.figure(figsize=(14, 5))
plt.semilogy(flux, label='Spectral Flux')
plt.title('Spectral Flux')
plt.xlabel('Frame')
plt.ylabel('Spectral Flux')
plt.legend()
plt.show()
```



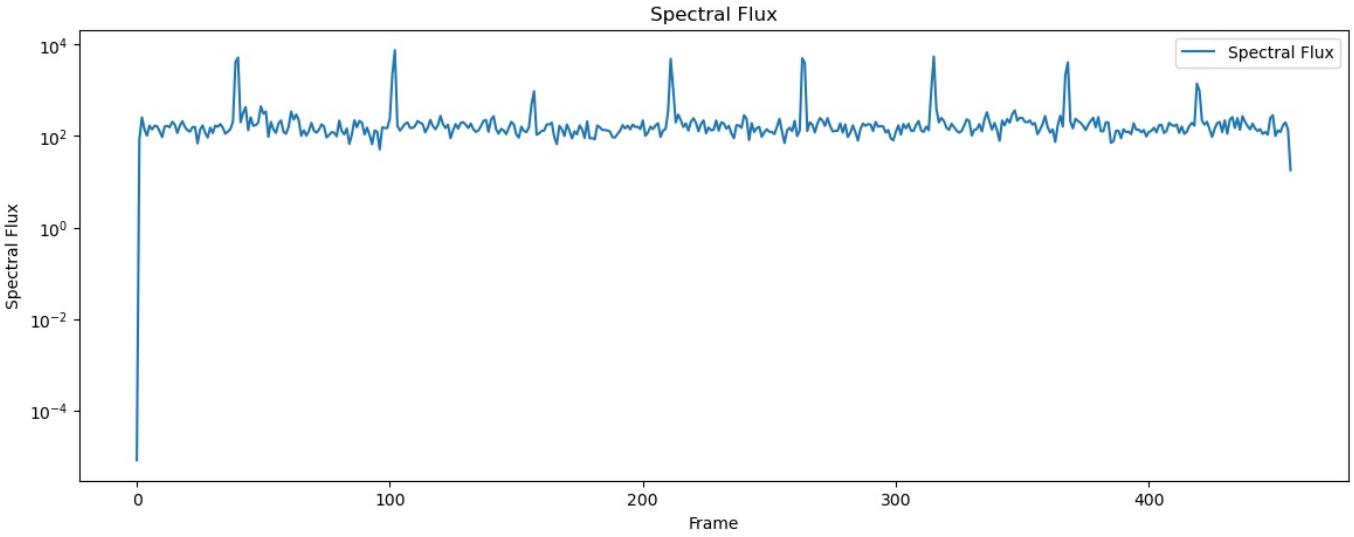
In [ ]:

In [1]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
```

In [5]:
```
# Load the audio file
audio_path = r'C:/Users/ASUS/switches.wav'
y, sr = librosa.load(audio_path)
# Plot the waveform
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [7]:
```
import numpy as np

# Compute Short-Time Fourier Transform (STFT)
D = librosa.stft(y)

# Compute the magnitude spectrogram
S = np.abs(D)**2

# Compute spectral flux
def spectral_flux(S):
    # Shift S to get previous frame
    S_prev = np.roll(S, shift=1, axis=1)
    # Calculate the flux
    flux = np.sum(np.maximum(S - S_prev, 0), axis=0)
    return flux

# Compute the spectral flux
flux = spectral_flux(S)

# Plot the spectral flux
plt.figure(figsize=(14, 5))
plt.semilogy(flux, label='Spectral Flux')
plt.title('Spectral Flux')
plt.xlabel('Frame')
plt.ylabel('Spectral Flux')
plt.legend()
plt.show()
```

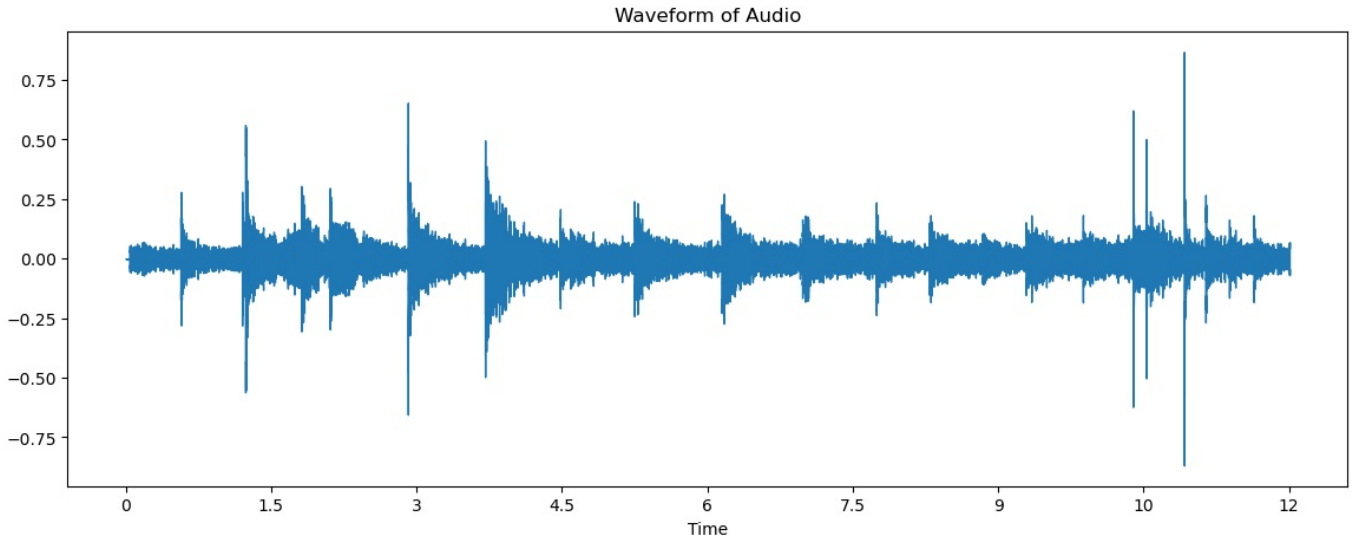


In [ ]:

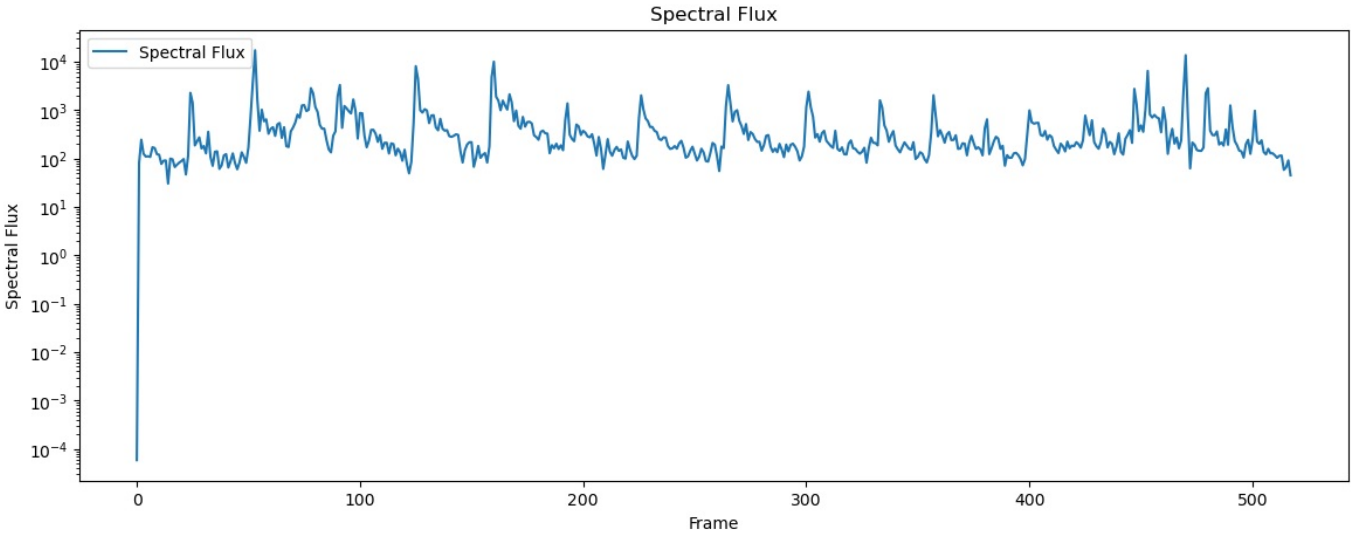
In [1]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the audio file
audio_path = r'C:/Users/ASUS/walking.wav'
y, sr = librosa.load(audio_path)
```

In [5]:
```
# Plot the waveform
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



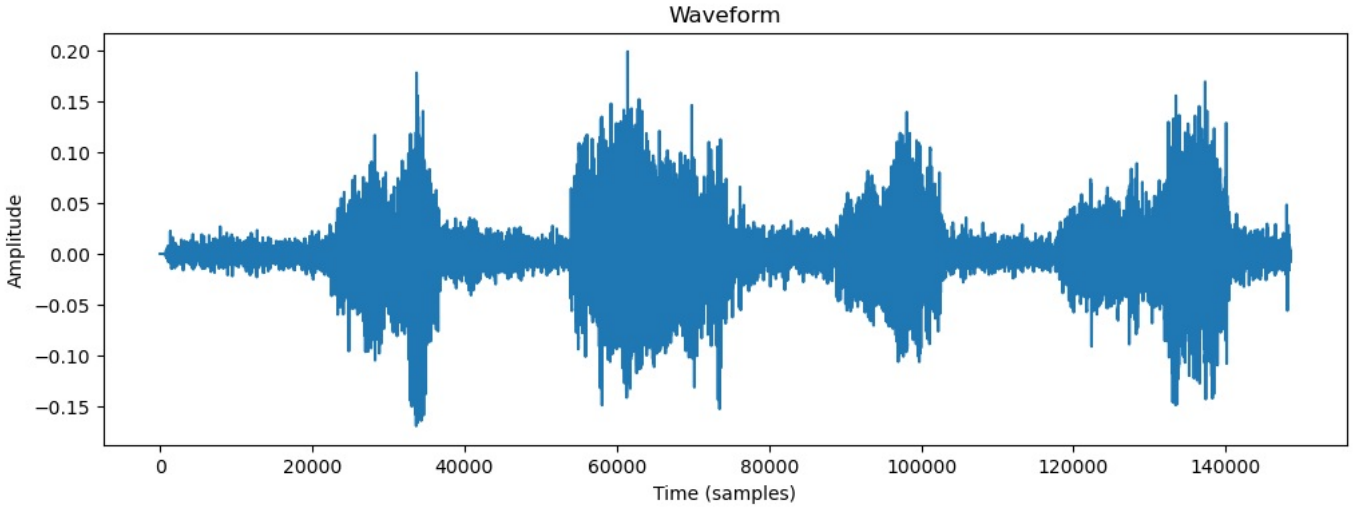
In [7]:
```
import numpy as np

# Compute Short-Time Fourier Transform (STFT)
D = librosa.stft(y)

# Compute the magnitude spectrogram
S = np.abs(D)**2

# Compute spectral flux
def spectral_flux(S):
    # Shift S to get previous frame
    S_prev = np.roll(S, shift=1, axis=1)
    # Calculate the flux
    flux = np.sum(np.maximum(S - S_prev, 0), axis=0)
    return flux

# Compute the spectral flux
flux = spectral_flux(S)

# Plot the spectral flux
plt.figure(figsize=(14, 5))
plt.semilogy(flux, label='Spectral Flux')
plt.title('Spectral Flux')
plt.xlabel('Frame')
plt.ylabel('Spectral Flux')
plt.legend()
plt.show()
```



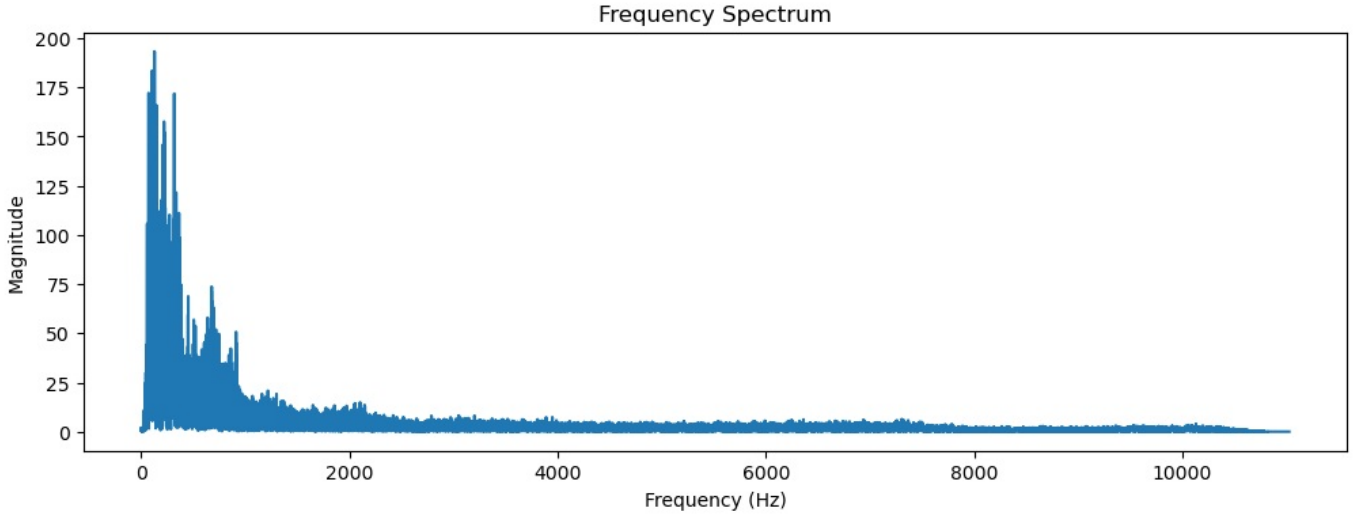
In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the audio file
audio_path = r'C:/Users/ASUS/bagzip.wav'  # Update with your file path
y, sr = librosa.load(audio_path)

# Visualize the waveform
plt.figure(figsize=(12, 4))
plt.plot(y)
plt.title('Waveform')
plt.xlabel('Time (samples)')
plt.ylabel('Amplitude')
plt.show()
```



In [5]:
```
# Apply Fast Fourier Transform (FFT)
Y = np.fft.fft(y)

# Get the frequency values
frequencies = np.fft.fftfreq(len(Y), 1/sr)

# Get the magnitude of the frequencies
magnitude = np.abs(Y)

# Plot the magnitude spectrum
plt.figure(figsize=(12, 4))
plt.plot(frequencies[:len(frequencies)//2], magnitude[:len(magnitude)//2])
plt.title('Frequency Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.show()
```

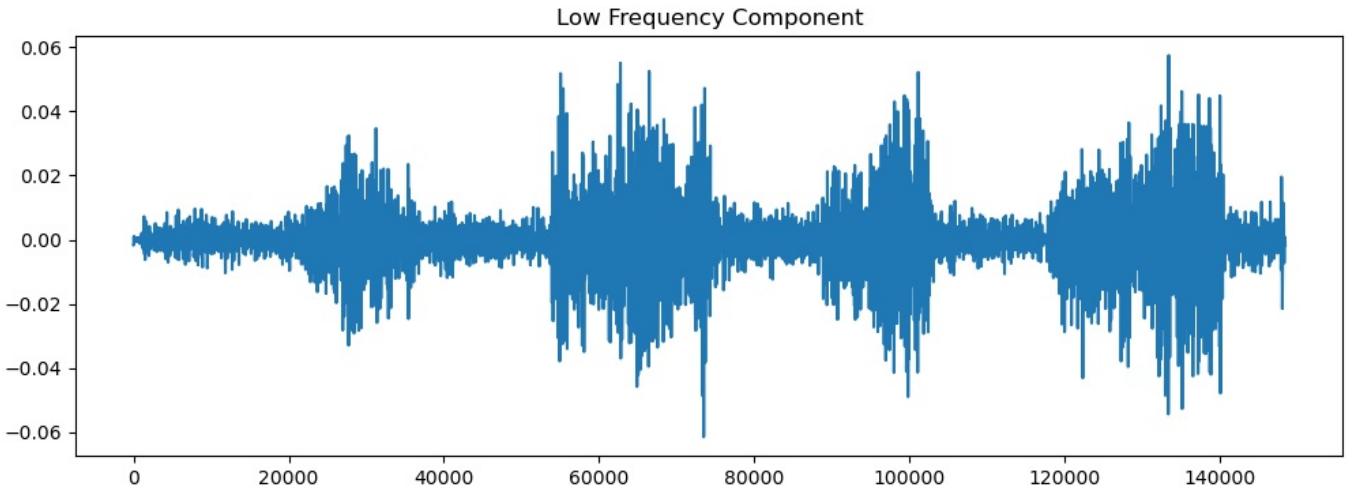


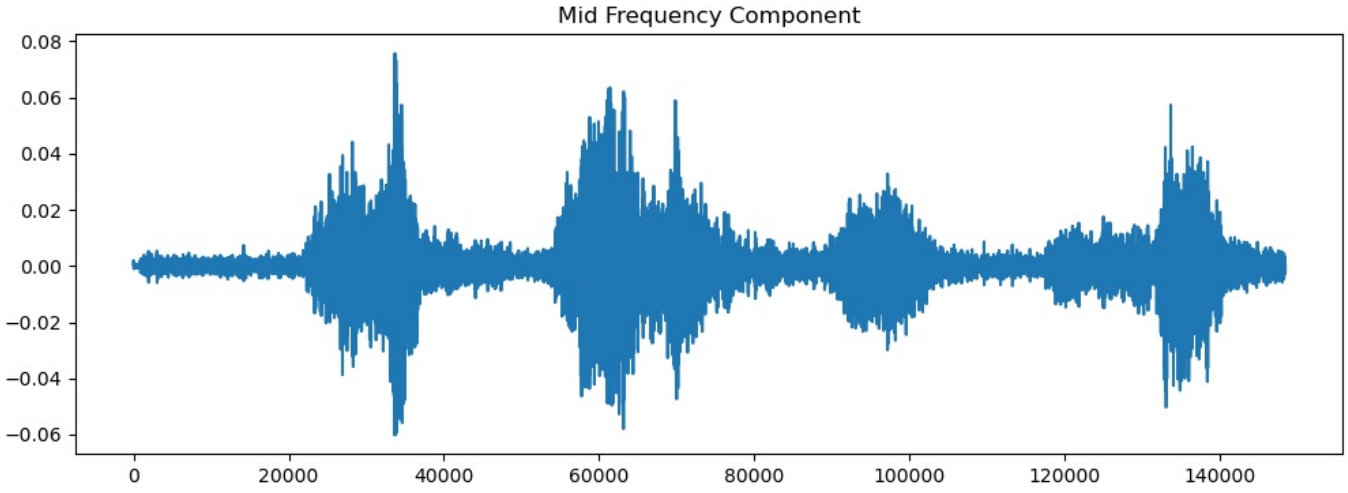
In [7]:
```
# Define frequency ranges (you can adjust these values)
low_freq = (0, 300)
mid_freq = (300, 2500)
high_freq = (2500, 20000)

# Filter out the frequencies in each band
low_filter = (frequencies >= low_freq[0]) & (frequencies <= low_freq[1])
```

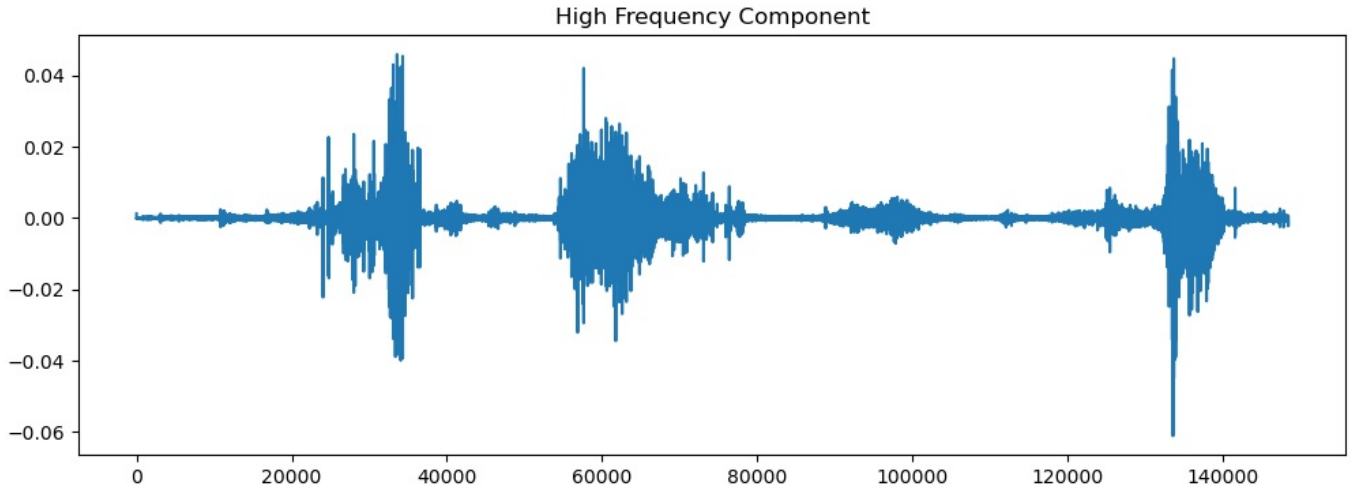
```
mid_filter = (frequencies >= mid_freq[0]) & (frequencies <= mid_freq[1])
high_filter = (frequencies >= high_freq[0]) & (frequencies <= high_freq[1])

# Apply the filters
low_component = Y * low_filter
mid_component = Y * mid_filter
high_component = Y * high_filter

# Convert back to time domain using the Inverse FFT (IFFT)
low_audio = np.fft.ifft(low_component)
mid_audio = np.fft.ifft(mid_component)
high_audio = np.fft.ifft(high_component)
```

In [9]:

```
# Plot the filtered signals
plt.figure(figsize=(12, 4))
plt.plot(low_audio.real)
plt.title('Low Frequency Component')
plt.show()

plt.figure(figsize=(12, 4))
plt.plot(mid_audio.real)
plt.title('Mid Frequency Component')
plt.show()

plt.figure(figsize=(12, 4))
plt.plot(high_audio.real)
plt.title('High Frequency Component')
plt.show()

# Optionally play the audio (requires IPython.display)
from IPython.display import Audio

# Play the low frequency component
Audio(low_audio.real, rate=sr)

# Play the mid frequency component
Audio(mid_audio.real, rate=sr)

# Play the high frequency component
Audio(high_audio.real, rate=sr)
```







Out[9]: Your browser does not support the audio element.

In [13]:
```
import soundfile as sf

# Save the components
sf.write('low_freq_component.wav', low_audio.real, sr)
sf.write('mid_freq_component.wav', mid_audio.real, sr)
sf.write('high_freq_component.wav', high_audio.real, sr)
```

In [21]:
```
from IPython.display import display, Audio

# Display all components in the same cell
display(Audio(low_audio.real, rate=sr))
display(Audio(mid_audio.real, rate=sr))
display(Audio(high_audio.real, rate=sr))
```

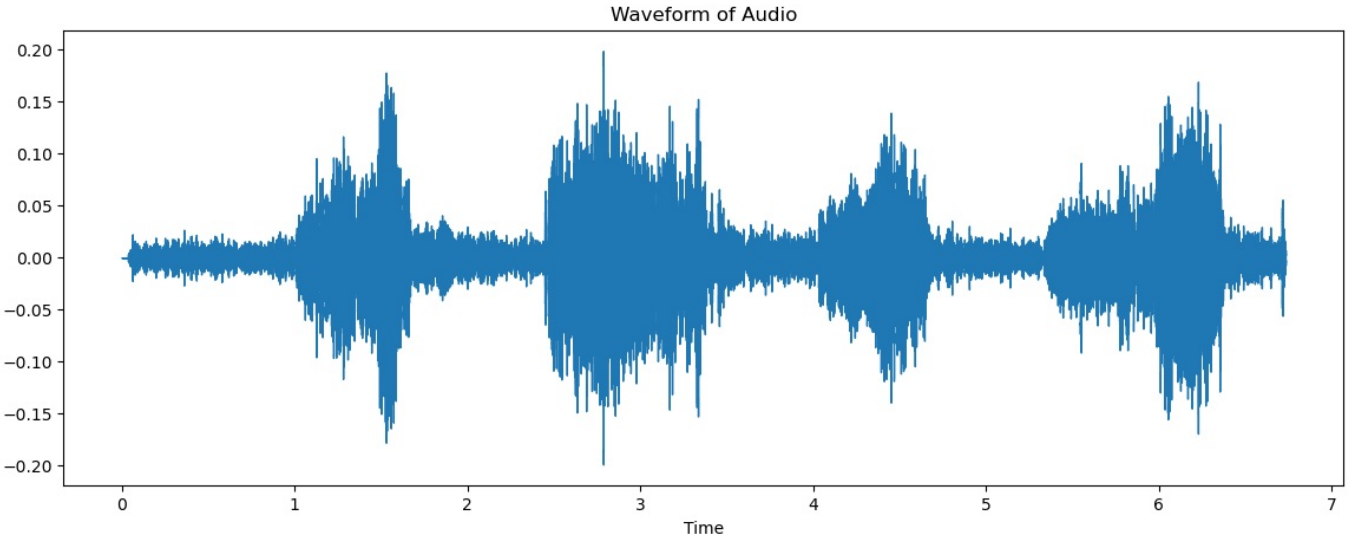
Your browser does not support the audio element.

Your browser does not support the audio element.

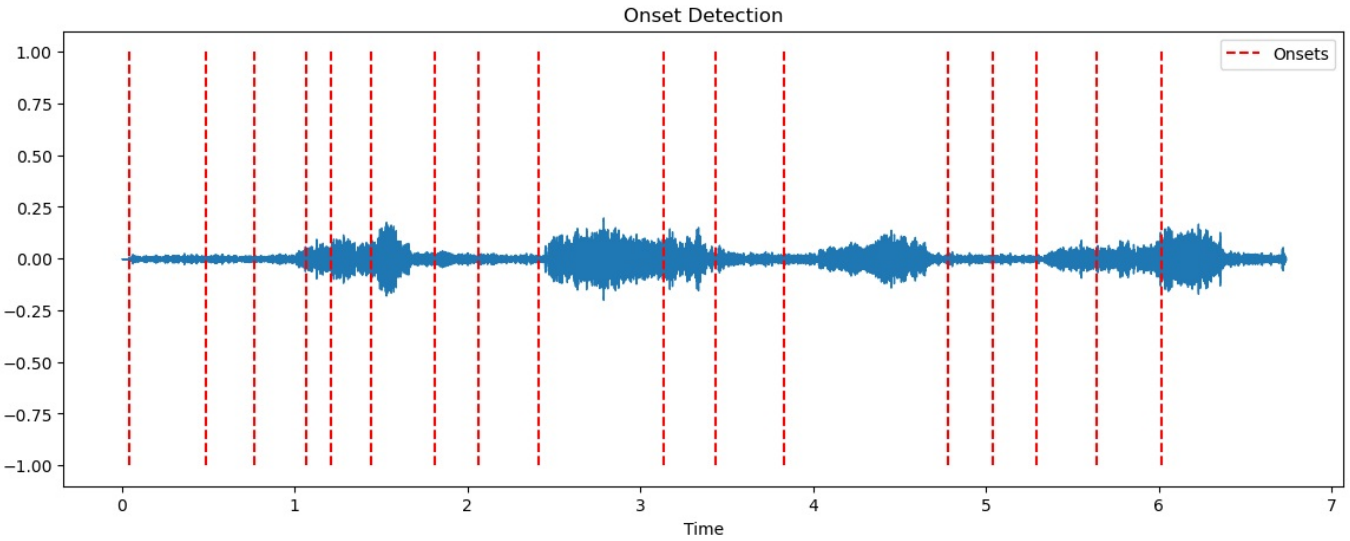
Your browser does not support the audio element.

In [ ]:

In [2]:
```
import librosa
import matplotlib.pyplot as plt
import numpy as np

audio_path = r'C:/Users/ASUS/bagzip.wav'
y, sr = librosa.load(audio_path)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [4]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True)

onset_times = librosa.frames_to_time(onset_frames, sr=sr)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.vlines(onset_times, ymin=-1, ymax=1, color='r', linestyle='--', label='Onsets')
plt.title('Onset Detection')
plt.legend()
plt.show()

print("Detected onset times (in seconds):")
print(onset_times)
```




```
Detected onset times (in seconds):
[0.04643991 0.48761905 0.7662585  1.06811791 1.20743764 1.43963719
 1.81115646 2.06657596 2.41487528 3.13469388 3.43655329 3.83129252
 4.78331066 5.03873016 5.29414966 5.64244898 6.01396825]
```

In [10]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True, delta=0.2, pre_max=20, post_max=20)
# You can adjust the onset detection parameters to capture subtle or stronger changes
```

In [12]:
```
# Extract and save each onset as a separate audio file
for i, onset_time in enumerate(onset_times):
    onset_sample = int(onset_time * sr)

    # Extract 0.5 seconds around the onset
```

```
    start_sample = max(0, onset_sample - int(0.25 * sr))  # 0.25 seconds before the onset
    end_sample = min(len(y), onset_sample + int(0.25 * sr))  # 0.25 seconds after the onset

    onset_segment = y[start_sample:end_sample]

    # Save the segment
    filename = f'onset_{i+1}.wav'
    librosa.output.write_wav(filename, onset_segment, sr)
    print(f"Saved: {filename}")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[12], line 13
     11 # Save the segment
     12 filename = f'onset_{i+1}.wav'
---> 13 librosa.output.write_wav(filename, onset_segment, sr)
     14 print(f"Saved: {filename}")

File ~\anaconda3\Lib\site-packages\lazy_loader\__init__.py:94, in attach.<locals>.__getattr__(name)
     92     return attr
     93 else:
---> 94     raise AttributeError(f"No {package_name} attribute {name}")

AttributeError: No librosa attribute output
```

In [14]:
```
!pip install soundfile
```

```
Requirement already satisfied: soundfile in c:\users\asus\anaconda3\lib\site-packages (0.12.1)
Requirement already satisfied: cffi>=1.0 in c:\users\asus\anaconda3\lib\site-packages (from soundfile) (1.16.0)
Requirement already satisfied: pycparser in c:\users\asus\anaconda3\lib\site-packages (from cffi>=1.0->soundfile
) (2.21)
```

In [16]:
```
import soundfile as sf

# Extract and save each onset as a separate audio file
for i, onset_time in enumerate(onset_times):
    onset_sample = int(onset_time * sr)

    # Extract 0.5 seconds around the onset
    start_sample = max(0, onset_sample - int(0.25 * sr))  # 0.25 seconds before the onset
    end_sample = min(len(y), onset_sample + int(0.25 * sr))  # 0.25 seconds after the onset

    onset_segment = y[start_sample:end_sample]

    # Save the segment using soundfile
    filename = f'onset_{i+1}.wav'
    sf.write(filename, onset_segment, sr)
    print(f"Saved: {filename}")
```

```
Saved: onset_1.wav
Saved: onset_2.wav
Saved: onset_3.wav
Saved: onset_4.wav
Saved: onset_5.wav
Saved: onset_6.wav
Saved: onset_7.wav
Saved: onset_8.wav
Saved: onset_9.wav
Saved: onset_10.wav
Saved: onset_11.wav
Saved: onset_12.wav
Saved: onset_13.wav
Saved: onset_14.wav
Saved: onset_15.wav
Saved: onset_16.wav
Saved: onset_17.wav
```

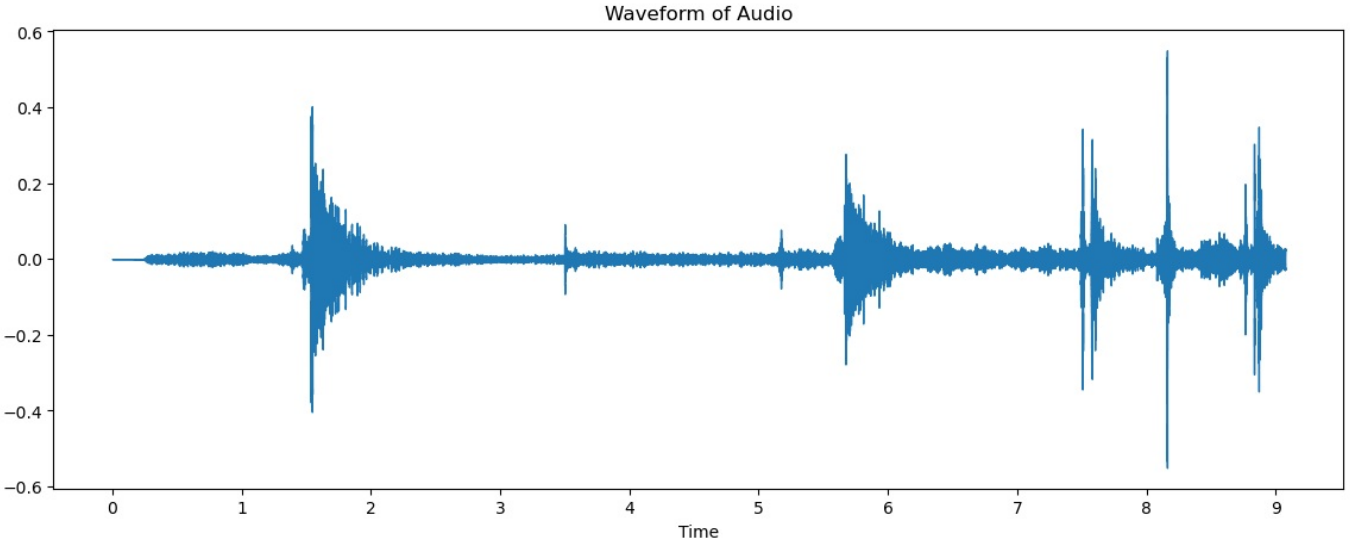
In [18]:
```
from IPython.display import Audio

# Play the entire audio
Audio(y, rate=sr)

# Play the first detected onset segment
Audio(onset_segment, rate=sr)
```

Out[18]: Your browser does not support the audio element.

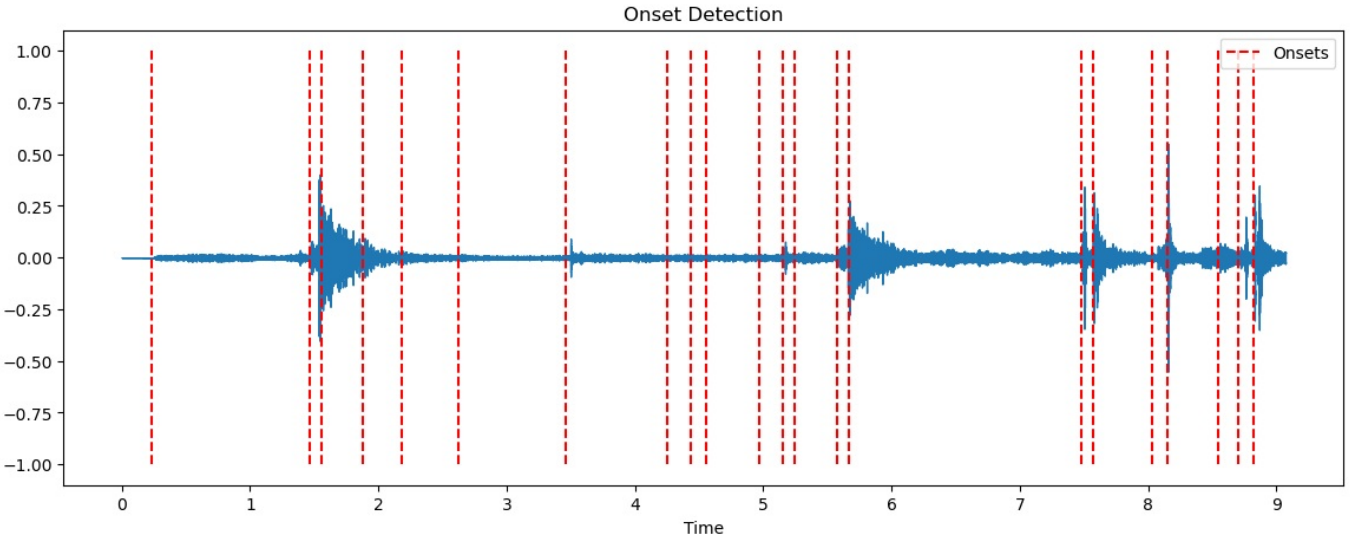
In [ ]:

In [1]:
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
```

In [7]:
```
audio_path = r'C:/Users/ASUS/doorclosing.wav'
y, sr = librosa.load(audio_path)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [9]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True)

onset_times = librosa.frames_to_time(onset_frames, sr=sr)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.vlines(onset_times, ymin=-1, ymax=1, color='r', linestyle='--', label='Onsets')
plt.title('Onset Detection')
plt.legend()
plt.show()

print("Detected onset times (in seconds):")
print(onset_times)
```

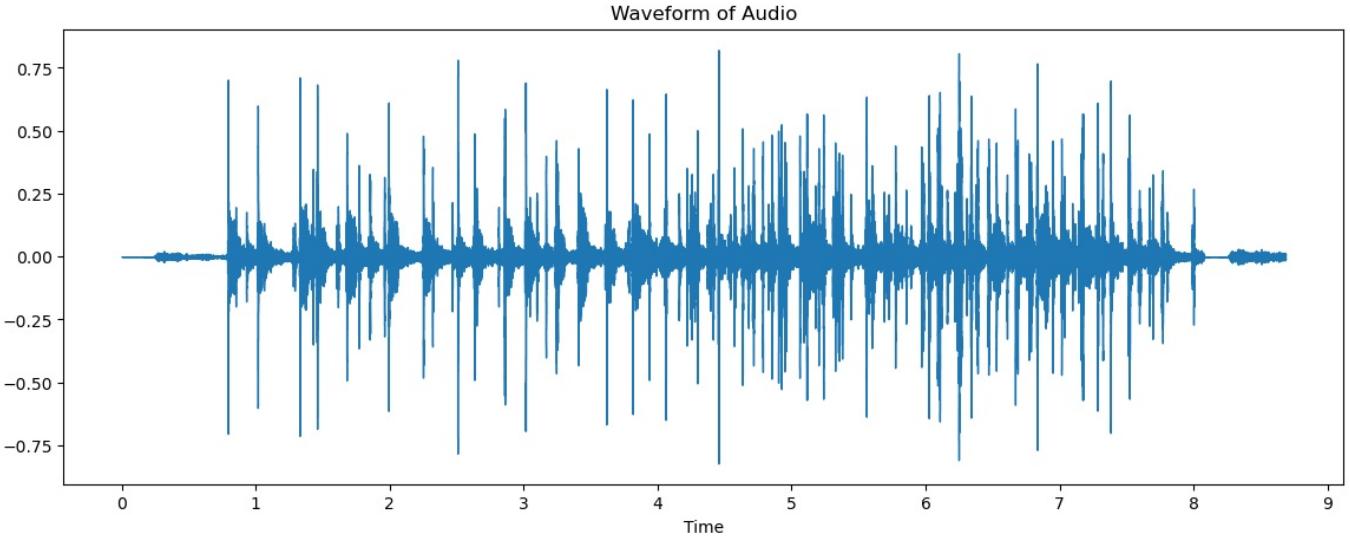



```
Detected onset times (in seconds):
[0.23219955 1.46285714 1.55573696 1.88081633 2.18267574 2.62385488
 3.45977324 4.2492517  4.43501134 4.55111111 4.96907029 5.15482993
 5.24770975 5.57278912 5.66566893 7.4768254  7.56970522 8.03410431
 8.15020408 8.54494331 8.70748299 8.82358277]
```

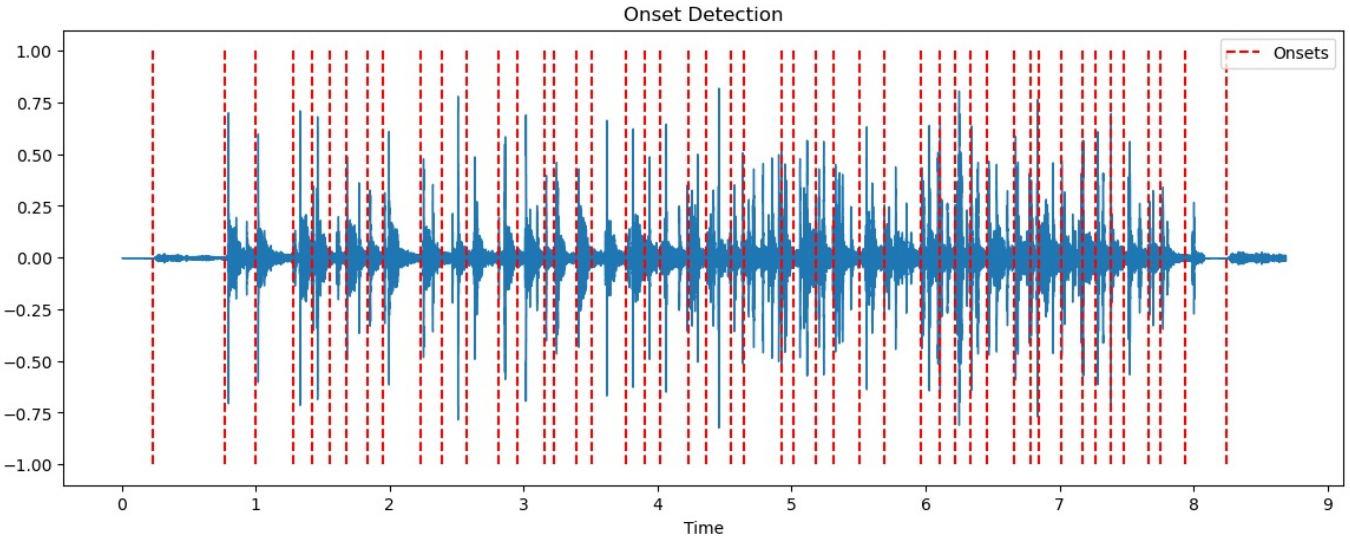
In [ ]:

In [2]:
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
```

In [7]:
```
audio_path = r'C:/Users/ASUS/keyboard.wav'
y, sr = librosa.load(audio_path)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [9]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True)

onset_times = librosa.frames_to_time(onset_frames, sr=sr)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.vlines(onset_times, ymin=-1, ymax=1, color='r', linestyle='--', label='Onsets')
plt.title('Onset Detection')
plt.legend()
plt.show()

print("Detected onset times (in seconds):")
print(onset_times)
```




```
Detected onset times (in seconds):
[0.23219955 0.7662585  0.99845805 1.27709751 1.41641723 1.55573696
 1.67183673 1.83437642 1.95047619 2.22911565 2.39165533 2.57741497
 2.80961451 2.94893424 3.15791383 3.2275737  3.39011338 3.50621315
 3.76163265 3.90095238 4.01705215 4.22603175 4.36535147 4.55111111
 4.64399093 4.92263039 5.0155102  5.17804989 5.31736961 5.50312925
 5.68888889 5.96752834 6.10684807 6.22294785 6.33904762 6.45514739
 6.66412698 6.78022676 6.84988662 7.0124263  7.17496599 7.2678458
 7.38394558 7.4768254  7.66258503 7.75546485 7.94122449 8.2430839 ]
```

In [13]:
```
import soundfile as sf

# Extract and save each onset as a separate audio file
```

```
for i, onset_time in enumerate(onset_times):
    onset_sample = int(onset_time * sr)

    # Extract 0.5 seconds around the onset
    start_sample = max(0, onset_sample - int(0.25 * sr))  # 0.25 seconds before the onset
    end_sample = min(len(y), onset_sample + int(0.25 * sr))  # 0.25 seconds after the onset

    onset_segment = y[start_sample:end_sample]

    # Save the segment using soundfile
    filename = f'onset_{i+1}.wav'
    sf.write(filename, onset_segment, sr)
    print(f"Saved: {filename}")
```

```
Saved: onset_1.wav
Saved: onset_2.wav
Saved: onset_3.wav
Saved: onset_4.wav
Saved: onset_5.wav
Saved: onset_6.wav
Saved: onset_7.wav
Saved: onset_8.wav
Saved: onset_9.wav
Saved: onset_10.wav
Saved: onset_11.wav
Saved: onset_12.wav
Saved: onset_13.wav
Saved: onset_14.wav
Saved: onset_15.wav
Saved: onset_16.wav
Saved: onset_17.wav
Saved: onset_18.wav
Saved: onset_19.wav
Saved: onset_20.wav
Saved: onset_21.wav
Saved: onset_22.wav
Saved: onset_23.wav
Saved: onset_24.wav
Saved: onset_25.wav
Saved: onset_26.wav
Saved: onset_27.wav
Saved: onset_28.wav
Saved: onset_29.wav
Saved: onset_30.wav
Saved: onset_31.wav
Saved: onset_32.wav
Saved: onset_33.wav
Saved: onset_34.wav
Saved: onset_35.wav
Saved: onset_36.wav
Saved: onset_37.wav
Saved: onset_38.wav
Saved: onset_39.wav
Saved: onset_40.wav
Saved: onset_41.wav
Saved: onset_42.wav
Saved: onset_43.wav
Saved: onset_44.wav
Saved: onset_45.wav
Saved: onset_46.wav
Saved: onset_47.wav
Saved: onset_48.wav
```

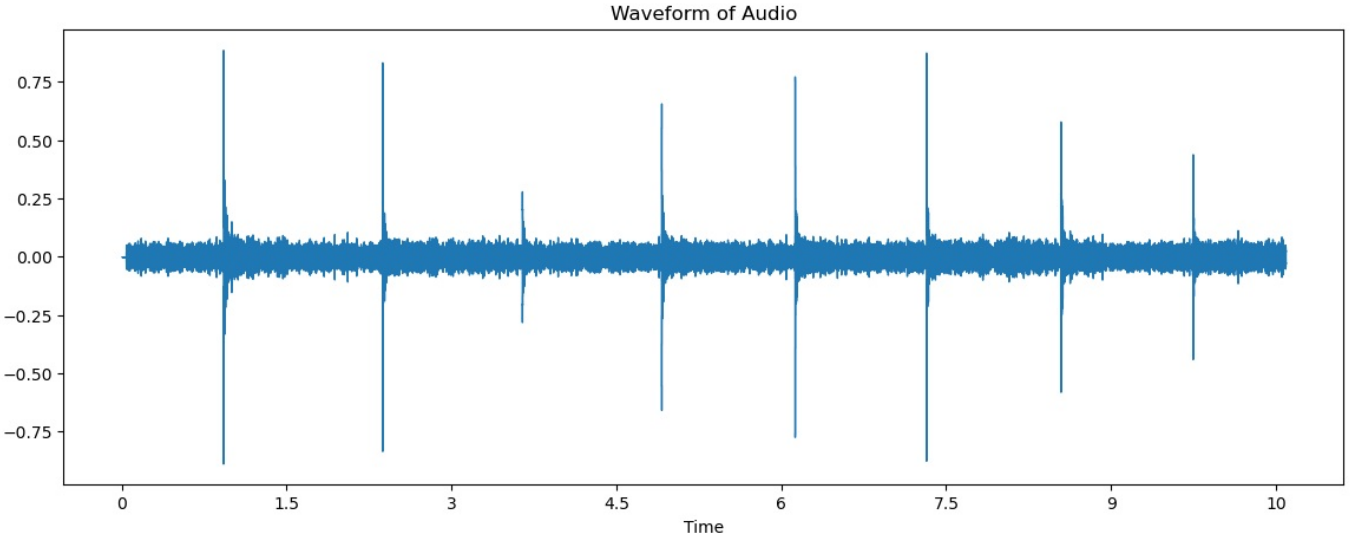
In [15]:
```
from IPython.display import Audio

# Play the entire audio
Audio(y, rate=sr)

# Play the first detected onset segment
Audio(onset_segment, rate=sr)
```

Out[15]: Your browser does not support the audio element.

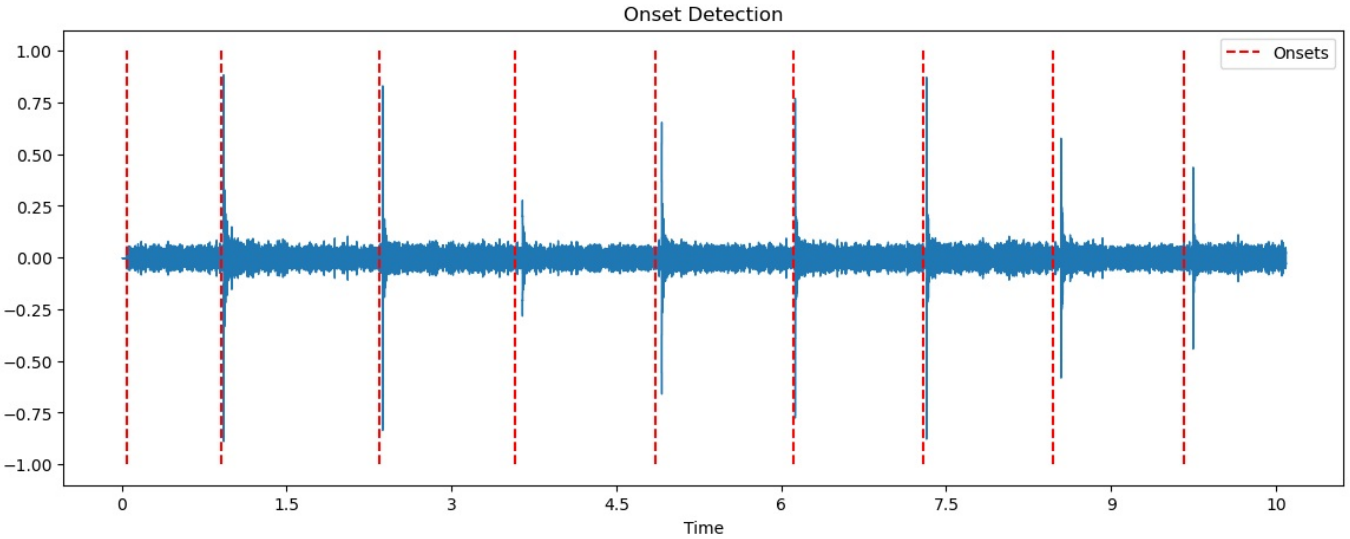
In [ ]:

In [1]:
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
audio_path = r'C:/Users/ASUS/switches.wav'
y, sr = librosa.load(audio_path)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [5]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True)

onset_times = librosa.frames_to_time(onset_frames, sr=sr)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.vlines(onset_times, ymin=-1, ymax=1, color='r', linestyle='--', label='Onsets')
plt.title('Onset Detection')
plt.legend()
plt.show()

print("Detected onset times (in seconds):")
print(onset_times)
```

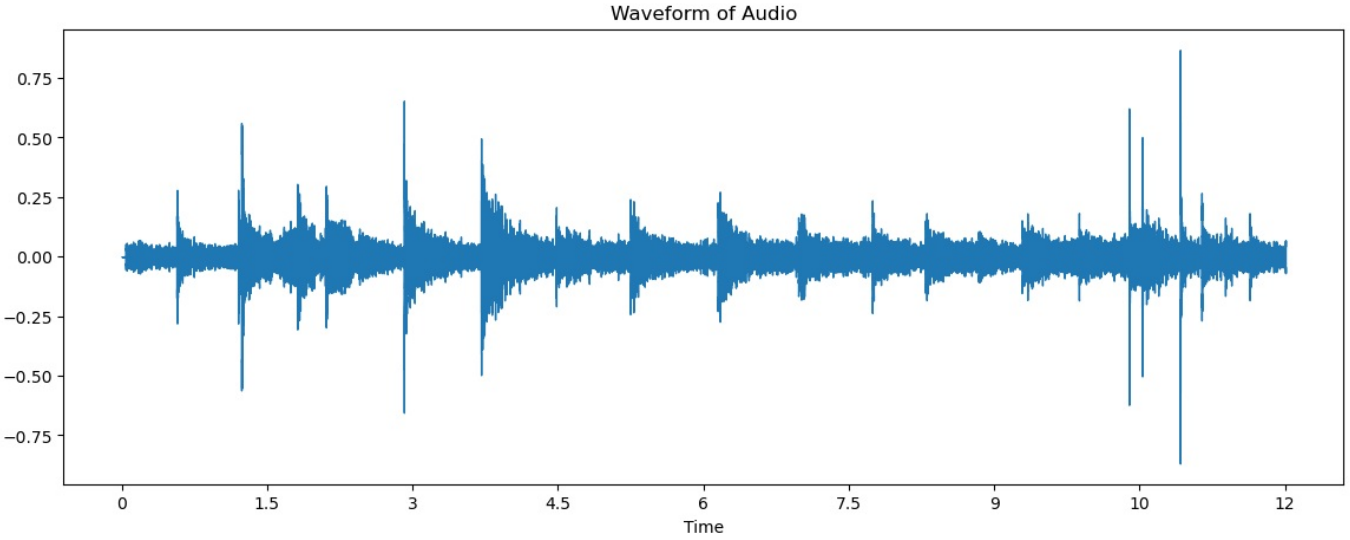



```
Detected onset times (in seconds):
[0.04643991 0.90557823 2.34521542 3.57587302 4.85297052 6.10684807
 7.29106576 8.47528345 9.65950113]
```

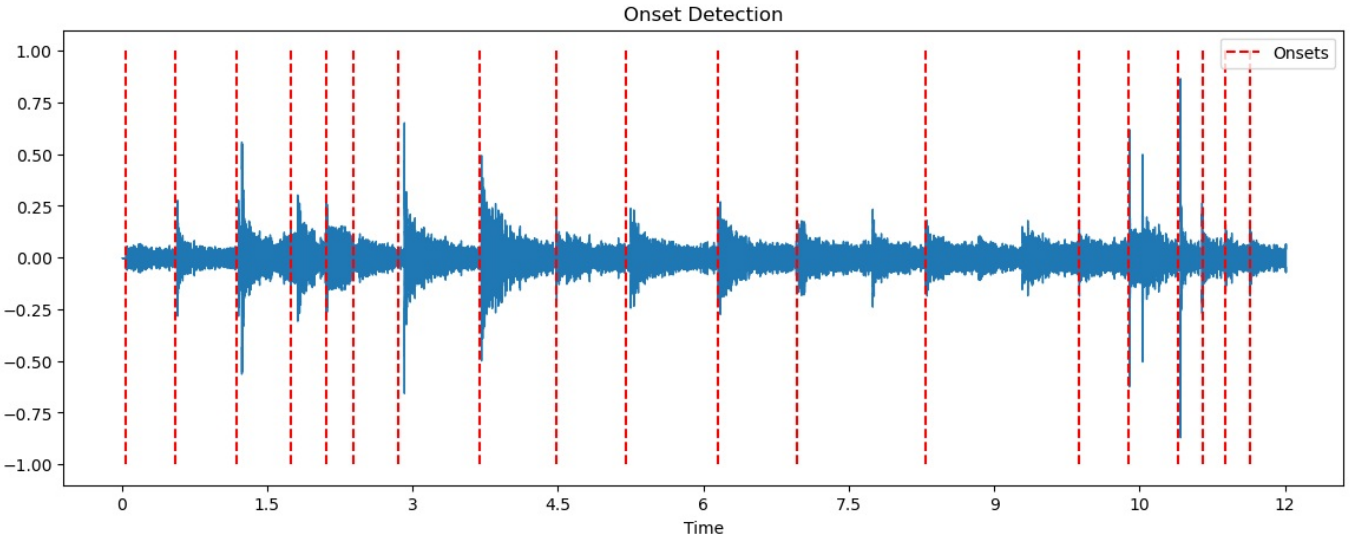
In [ ]:

In [1]:
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
audio_path = r'C:/Users/ASUS/walking.wav'
y, sr = librosa.load(audio_path)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform of Audio')
plt.show()
```



In [5]:
```
onset_frames = librosa.onset.onset_detect(y=y, sr=sr, backtrack=True)

onset_times = librosa.frames_to_time(onset_frames, sr=sr)

plt.figure(figsize=(14, 5))
librosa.display.waveshow(y, sr=sr)
plt.vlines(onset_times, ymin=-1, ymax=1, color='r', linestyle='--', label='Onsets')
plt.title('Onset Detection')
plt.legend()
plt.show()

print("Detected onset times (in seconds):")
print(onset_times)
```




```
Detected onset times (in seconds):
[ 0.04643991  0.55727891  1.18421769  1.7414966   2.11301587  2.39165533
  2.85605442  3.69197279  4.48145125  5.20126984  6.15328798  6.96598639
  8.28952381  9.86848073 10.37931973 10.89015873 11.14557823 11.37777778
 11.63319728]
```

In [7]:
```
import soundfile as sf

# Extract and save each onset as a separate audio file
for i, onset_time in enumerate(onset_times):
    onset_sample = int(onset_time * sr)

    # Extract 0.5 seconds around the onset
```

```
    start_sample = max(0, onset_sample - int(0.25 * sr))  # 0.25 seconds before the onset
    end_sample = min(len(y), onset_sample + int(0.25 * sr))  # 0.25 seconds after the onset

    onset_segment = y[start_sample:end_sample]

    # Save the segment using soundfile
    filename = f'onset_{i+1}.wav'
    sf.write(filename, onset_segment, sr)
    print(f"Saved: {filename}")
```

```
Saved: onset_1.wav
Saved: onset_2.wav
Saved: onset_3.wav
Saved: onset_4.wav
Saved: onset_5.wav
Saved: onset_6.wav
Saved: onset_7.wav
Saved: onset_8.wav
Saved: onset_9.wav
Saved: onset_10.wav
Saved: onset_11.wav
Saved: onset_12.wav
Saved: onset_13.wav
Saved: onset_14.wav
Saved: onset_15.wav
Saved: onset_16.wav
Saved: onset_17.wav
Saved: onset_18.wav
Saved: onset_19.wav
```

In [9]:
```
from IPython.display import Audio

# Play the entire audio
Audio(y, rate=sr)

# Play the first detected onset segment
Audio(onset_segment, rate=sr)
```

Out[9]: Your browser does not support the audio element.

In [ ]:

In [16]:
```
!pip install librosa
```

```
Requirement already satisfied: librosa in c:\users\asus\anaconda3\lib\site-packages (0.10.2.post1)
Requirement already satisfied: audioread>=2.1.9 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (3.0
.1)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in c:\users\asus\anaconda3\lib\site-pack
ages (from librosa) (1.26.4)
Requirement already satisfied: scipy>=1.2.0 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (1.13.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\asus\anaconda3\lib\site-packages (from librosa)
(1.4.2)
Requirement already satisfied: joblib>=0.14 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (1.4.2)
Requirement already satisfied: decorator>=4.3.0 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (5.1
.1)
Requirement already satisfied: numba>=0.51.0 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (0.59.1
)
Requirement already satisfied: soundfile>=0.12.1 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (0.
12.1)
Requirement already satisfied: pooch>=1.1 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (1.8.2)
Requirement already satisfied: soxr>=0.3.2 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (0.5.0.po
st1)
Requirement already satisfied: typing-extensions>=4.1.1 in c:\users\asus\anaconda3\lib\site-packages (from libro
sa) (4.11.0)
Requirement already satisfied: lazy-loader>=0.1 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (0.4
)
Requirement already satisfied: msgpack>=1.0 in c:\users\asus\anaconda3\lib\site-packages (from librosa) (1.0.3)
Requirement already satisfied: packaging in c:\users\asus\anaconda3\lib\site-packages (from lazy-loader>=0.1->li
brosa) (23.2)
Requirement already satisfied: llvmlite<0.43,>=0.42.0dev0 in c:\users\asus\anaconda3\lib\site-packages (from num
ba>=0.51.0->librosa) (0.42.0)
Requirement already satisfied: platformdirs>=2.5.0 in c:\users\asus\anaconda3\lib\site-packages (from pooch>=1.1
->librosa) (3.10.0)
Requirement already satisfied: requests>=2.19.0 in c:\users\asus\anaconda3\lib\site-packages (from pooch>=1.1->l
ibrosa) (2.32.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\asus\anaconda3\lib\site-packages (from scikit-le
arn>=0.20.0->librosa) (2.2.0)
Requirement already satisfied: cffi>=1.0 in c:\users\asus\anaconda3\lib\site-packages (from soundfile>=0.12.1->l
ibrosa) (1.16.0)
Requirement already satisfied: pycparser in c:\users\asus\anaconda3\lib\site-packages (from cffi>=1.0->soundfile
>=0.12.1->librosa) (2.21)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\asus\anaconda3\lib\site-packages (from reque
sts>=2.19.0->pooch>=1.1->librosa) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\anaconda3\lib\site-packages (from requests>=2.19.0-
>pooch>=1.1->librosa) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\asus\anaconda3\lib\site-packages (from requests>=2
.19.0->pooch>=1.1->librosa) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\asus\anaconda3\lib\site-packages (from requests>=2
.19.0->pooch>=1.1->librosa) (2024.7.4)
```

In [2]:
```
import librosa
print("Librosa installed successfully!")
```

```
Librosa installed successfully!
```

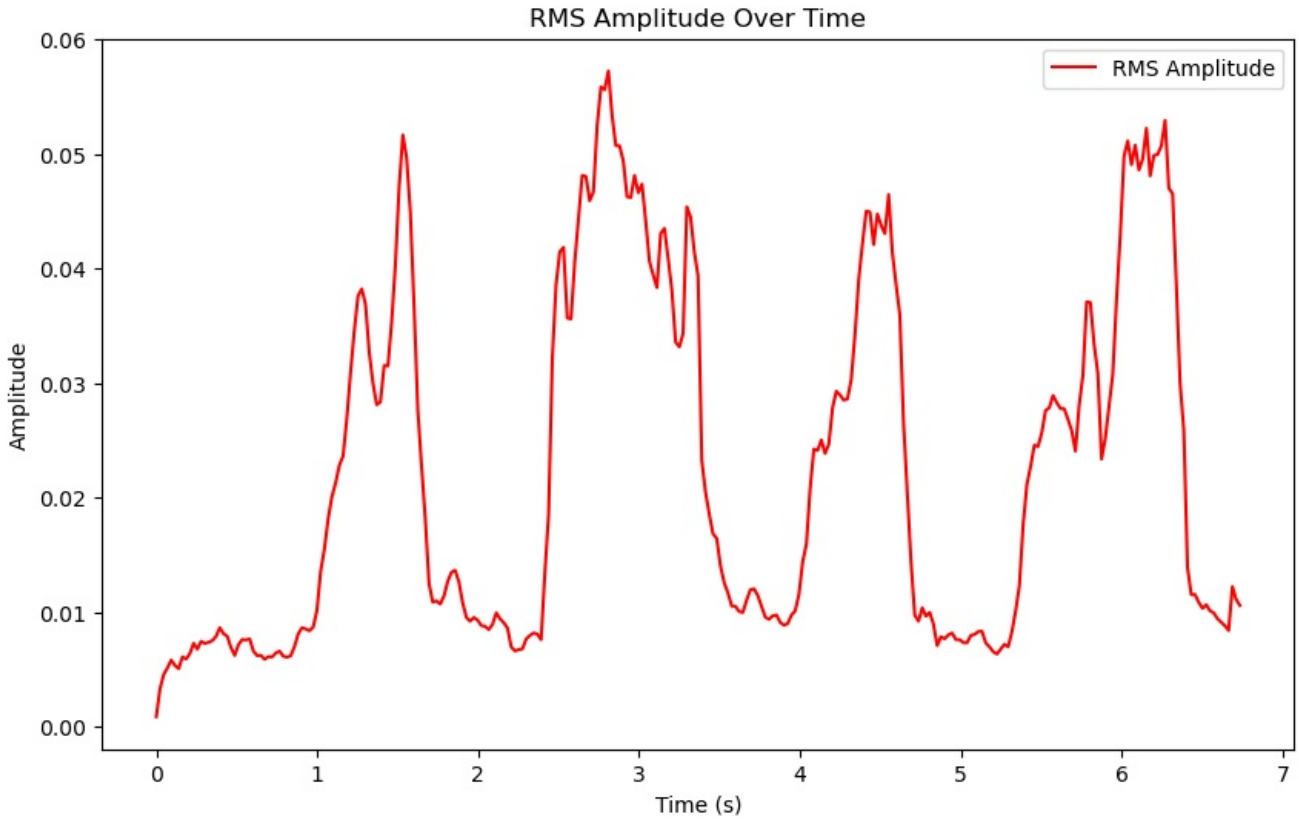
In [4]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [9]:
```
# Load the audio file (replace 'your_audio_file.wav' with your actual file path)
audio_path = r'C:\Users\ASUS\bagzip.wav'
y, sr = librosa.load(audio_path)
```

In [13]:
```
# Calculate the RMS amplitude
rms_amplitude = librosa.feature.rms(y=y)[0]
```

In [15]:
```
# Generate time axis
times = librosa.times_like(rms_amplitude, sr=sr)
```

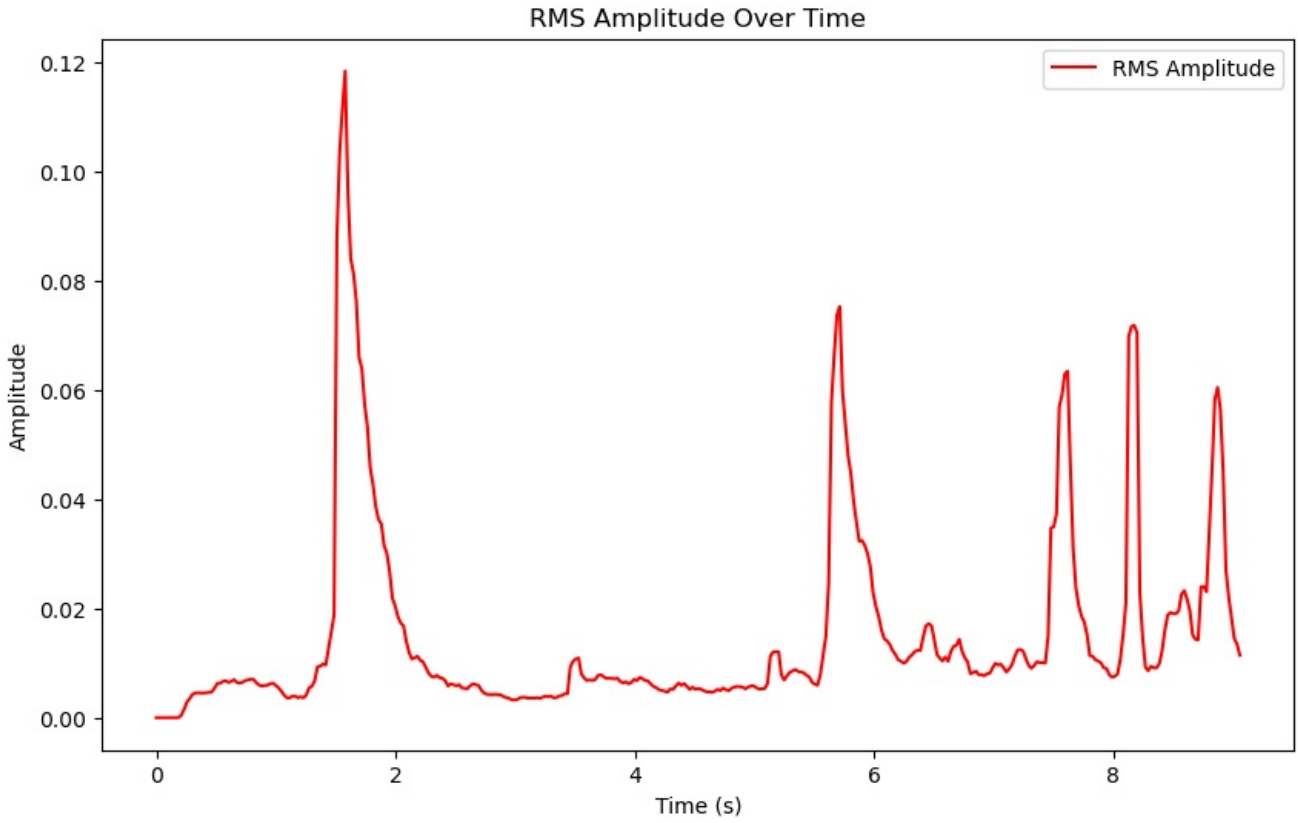
In [17]:
```
# Plot the RMS amplitude
plt.figure(figsize=(10, 6))
plt.plot(times, rms_amplitude, label='RMS Amplitude', color='r')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('RMS Amplitude Over Time')
plt.legend()
plt.show()
```



In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
audio_path = r'C:\Users\ASUS\doorclosing.wav'
y, sr = librosa.load(audio_path)
rms_amplitude = librosa.feature.rms(y=y)[0]
times = librosa.times_like(rms_amplitude, sr=sr)
```

In [5]:
```
plt.figure(figsize=(10, 6))
plt.plot(times, rms_amplitude, label='RMS Amplitude', color='r')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('RMS Amplitude Over Time')
plt.legend()
plt.show()
```



In [ ]:

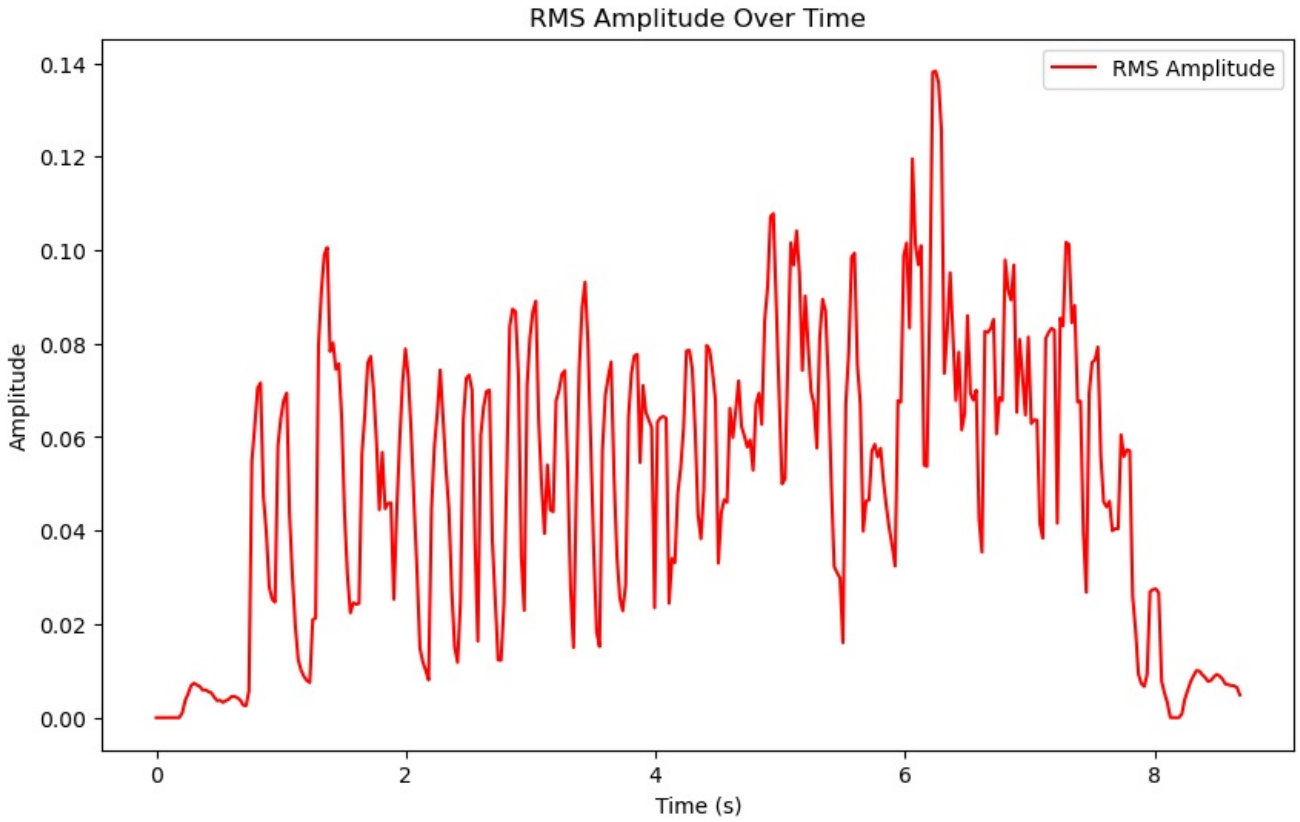
In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
audio_path = r'C:\Users\ASUS\keyboard.wav'
y, sr = librosa.load(audio_path)
```

In [5]:
```
rms_amplitude = librosa.feature.rms(y=y)[0]
```

In [7]:
```
times = librosa.times_like(rms_amplitude, sr=sr)
```

In [9]:
```
plt.figure(figsize=(10, 6))
plt.plot(times, rms_amplitude, label='RMS Amplitude', color='r')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('RMS Amplitude Over Time')
plt.legend()
plt.show()
```



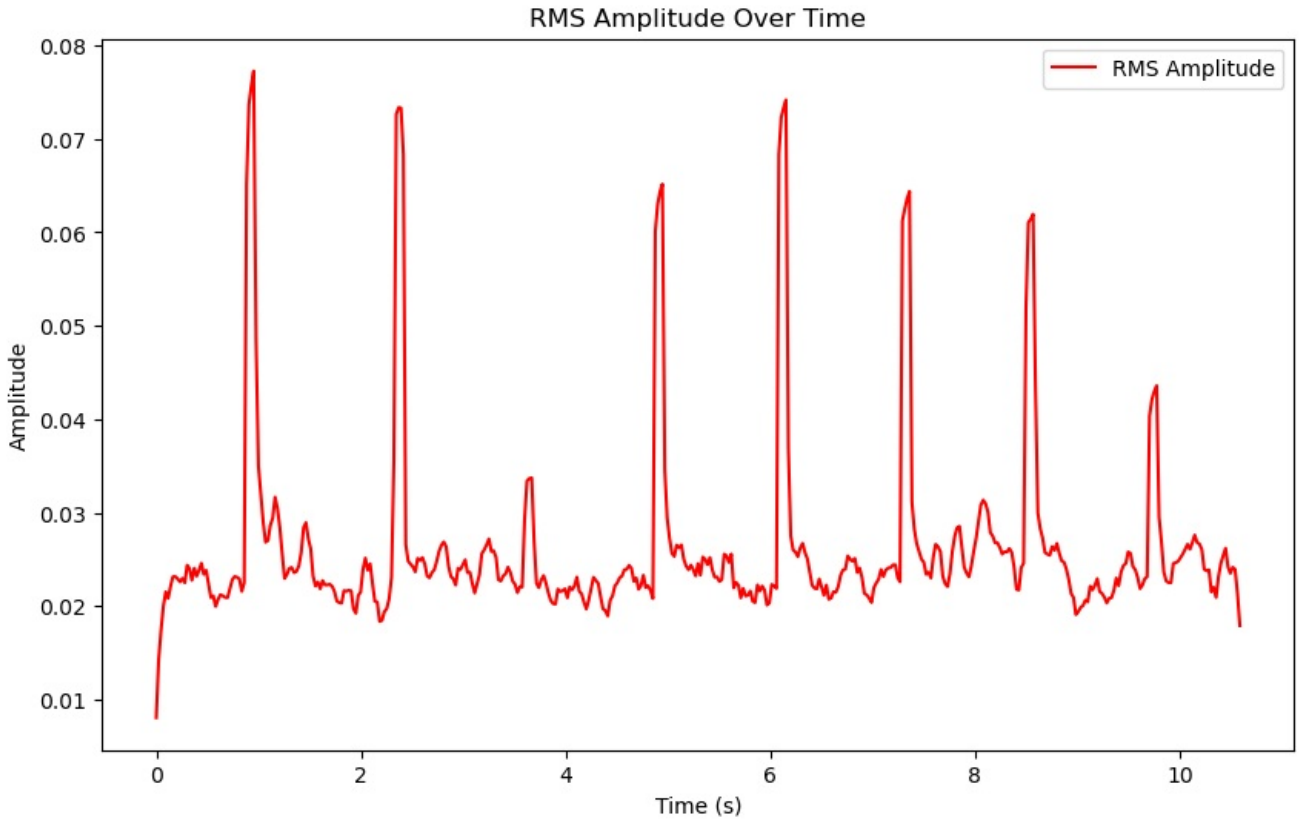
In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
audio_path = r'C:\Users\ASUS\switches.wav'
y, sr = librosa.load(audio_path)
rms_amplitude = librosa.feature.rms(y=y)[0]
```

In [5]:
```
times = librosa.times_like(rms_amplitude, sr=sr)
```

In [7]:
```
plt.figure(figsize=(10, 6))
plt.plot(times, rms_amplitude, label='RMS Amplitude', color='r')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('RMS Amplitude Over Time')
plt.legend()
plt.show()
```

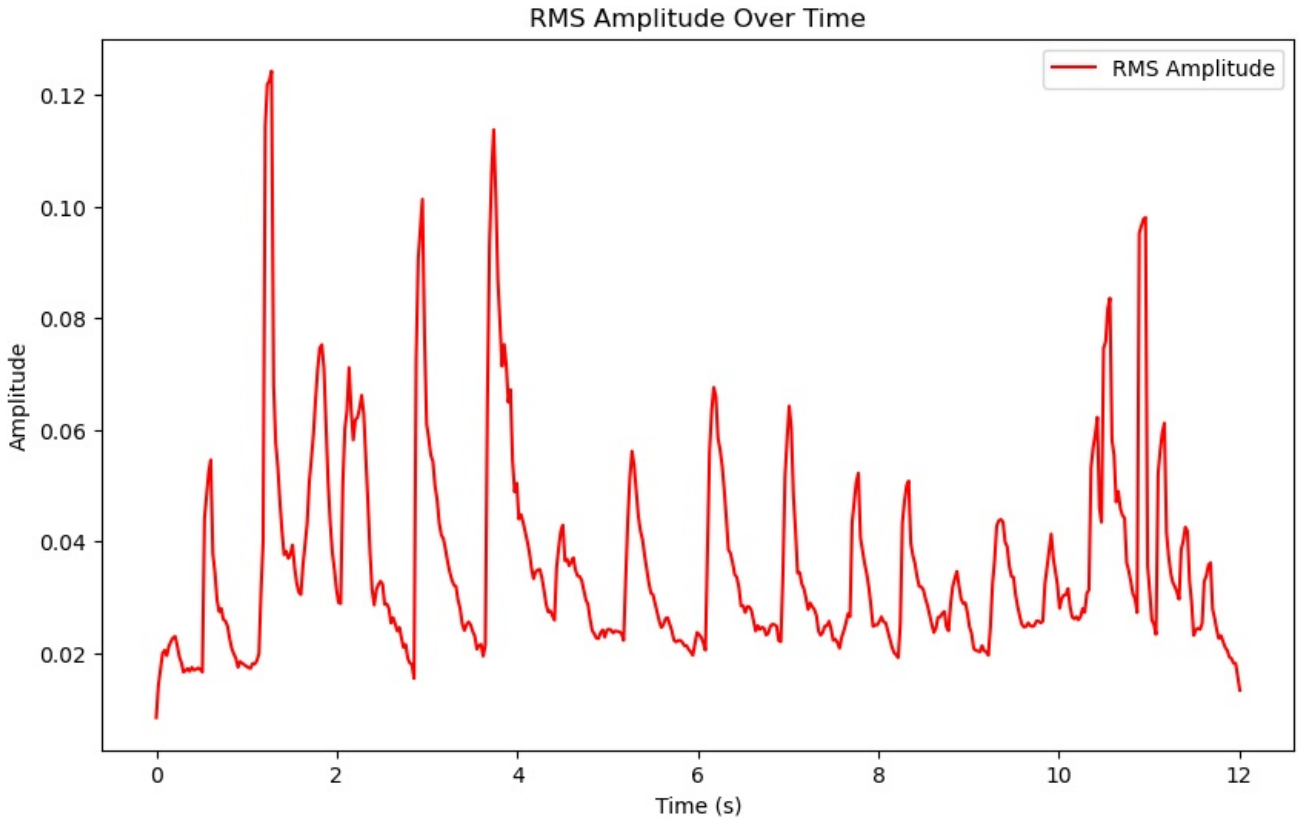


In [ ]:

In [1]:
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
audio_path = r'C:\Users\ASUS\walking.wav'
y, sr = librosa.load(audio_path)
rms_amplitude = librosa.feature.rms(y=y)[0]
times = librosa.times_like(rms_amplitude, sr=sr)
```

In [5]:
```
plt.figure(figsize=(10, 6))
plt.plot(times, rms_amplitude, label='RMS Amplitude', color='r')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('RMS Amplitude Over Time')
plt.legend()
plt.show()
```



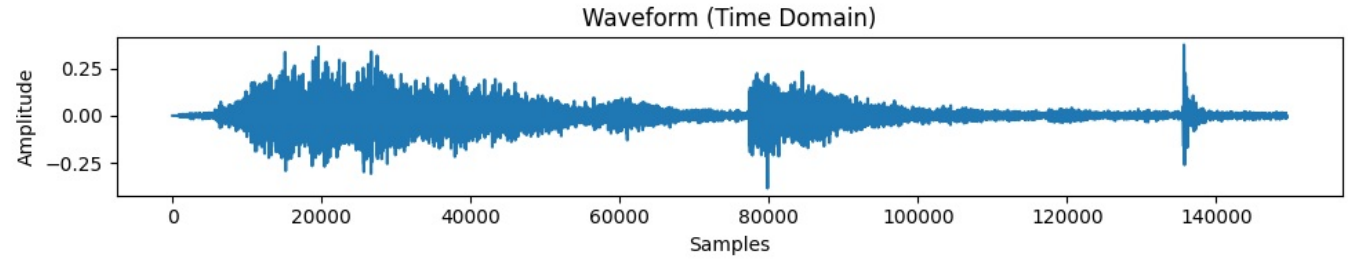
In [ ]:

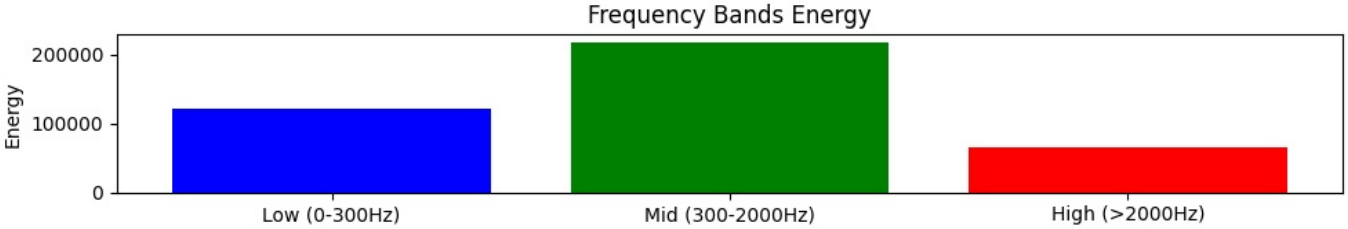
In [9]:

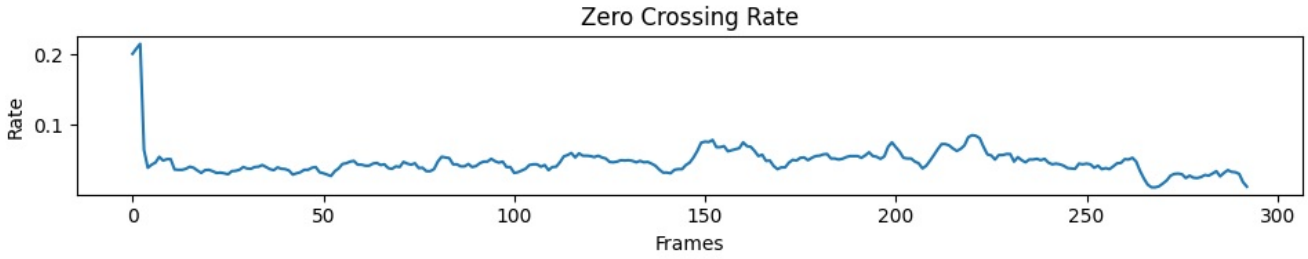
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'chairscraping.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

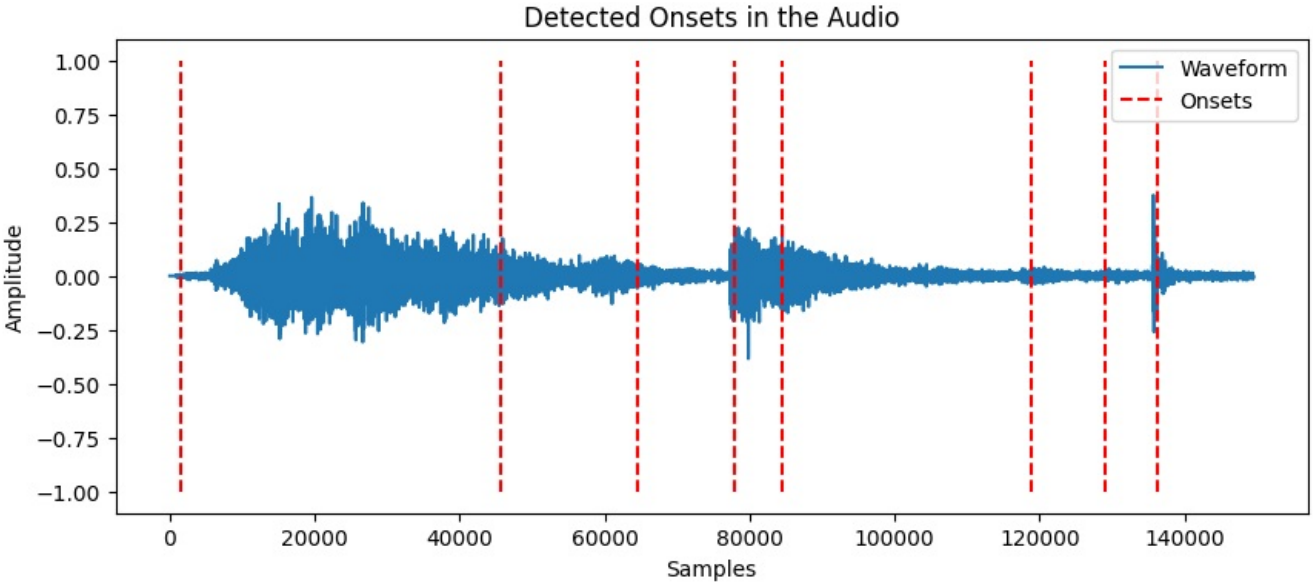
```
plt.show()
```

```
RMS Amplitude: 0.047595166
Low Frequency Energy: 120725.234
Mid Frequency Energy: 217681.5
High Frequency Energy: 65156.74
Spectral Flux: 15.410598
Zero Crossing Rate: 0.0482863334760274
Onset Times: [0.06965986 2.06657596 2.92571429 3.52943311 3.83129252 5.38702948
 5.85142857 6.17650794]
```









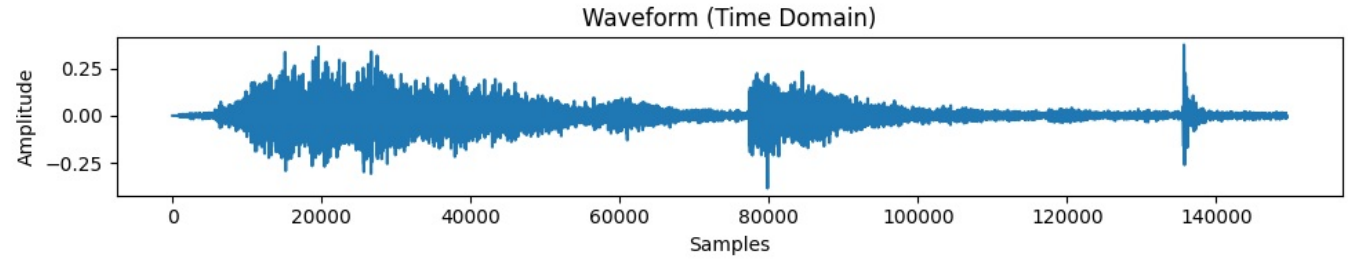
In [ ]:

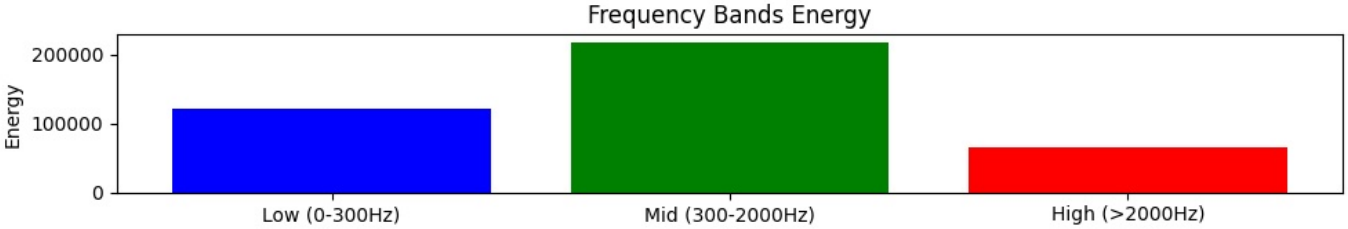
In [9]:

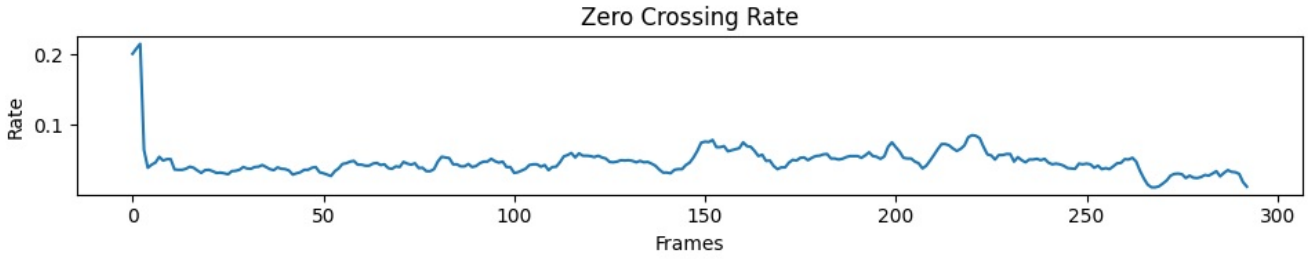
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'chairscraping1.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

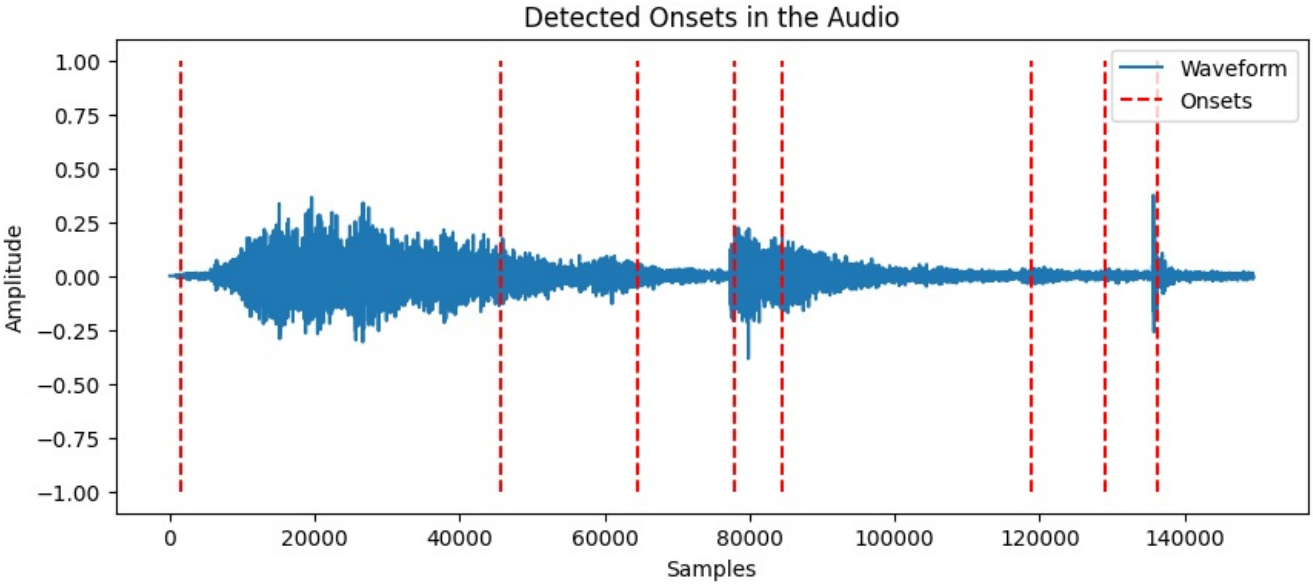
```
plt.show()
```

```
RMS Amplitude: 0.047595166
Low Frequency Energy: 120725.234
Mid Frequency Energy: 217681.5
High Frequency Energy: 65156.74
Spectral Flux: 15.410598
Zero Crossing Rate: 0.0482863334760274
Onset Times: [0.06965986 2.06657596 2.92571429 3.52943311 3.83129252 5.38702948
 5.85142857 6.17650794]
```









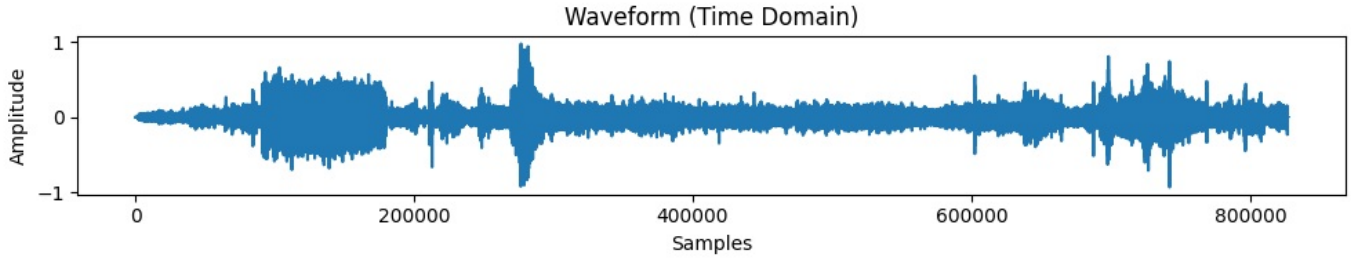
In [ ]:

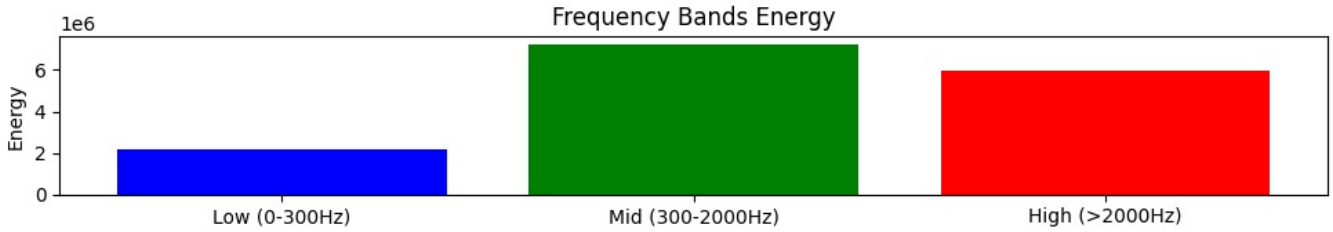
In [7]:

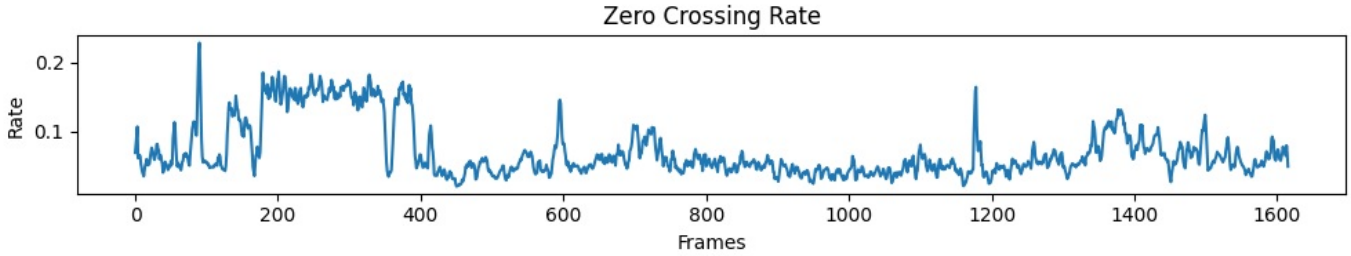
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'coffee.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

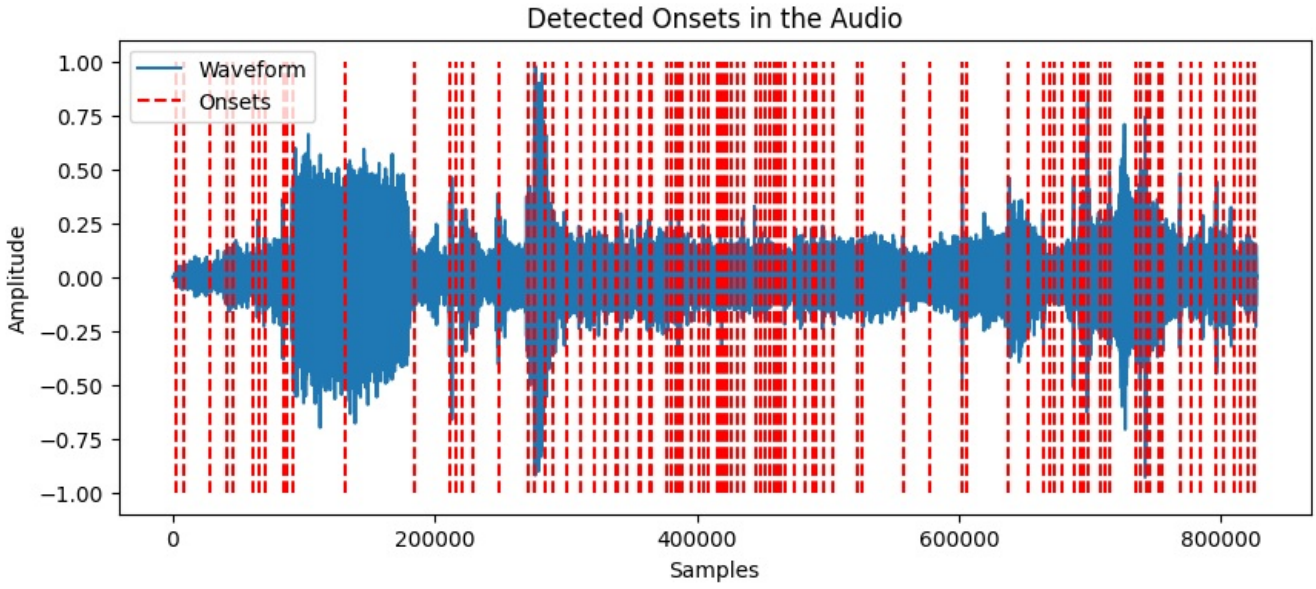
```
plt.show()
```

```
RMS Amplitude: 0.090381086
Low Frequency Energy: 2175649.0
Mid Frequency Energy: 7238371.0
High Frequency Energy: 5988051.5
Spectral Flux: 30.360418
Zero Crossing Rate: 0.07119052340851596
Onset Times: [ 0.09287982  0.37151927  1.30031746  1.85759637  2.06657596  2.7631746
  2.9721542   3.18113379  3.85451247  3.97061224  4.13315193  5.96752834
  8.38240363  9.61306122  9.82204082 10.00780045 10.37931973 11.28489796
 12.28335601 12.53877551 12.86385488 13.14249433 13.63011338 14.11773243
 14.58213152 14.95365079 15.32517007 15.37160998 15.7199093  16.11464853
 16.20752834 16.50938776 16.57904762 17.08988662 17.27564626 17.43818594
 17.5078458  17.62394558 17.94902494 18.2276644  18.39020408 18.55274376
 18.85460317 18.97070295 19.04036281 19.15646259 19.34222222 19.55120181
 19.73696145 20.15492063 20.31746032 20.48       20.64253968 20.80507937
 20.94439909 21.01405896 21.19981859 21.501678   21.89641723 22.15183673
 22.24471655 22.50013605 22.82521542 23.7075737  23.84689342 25.30975057
 26.19210884 27.32988662 27.46920635 28.93206349 29.58222222 30.13950113
 30.37170068 30.53424036 30.78965986 31.20761905 31.41659864 31.53269841
 31.67201814 32.11319728 32.25251701 32.43827664 33.34385488 33.50639456
 33.6921542  33.78503401 34.11011338 34.24943311 34.87637188 35.27111111
 35.54975057 36.10702948 36.36244898 36.75718821 36.9661678  37.19836735
 37.40734694]
```









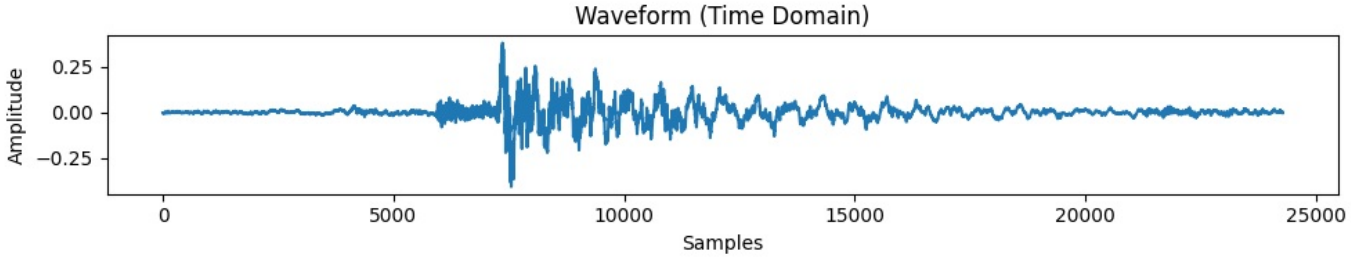
In [ ]:

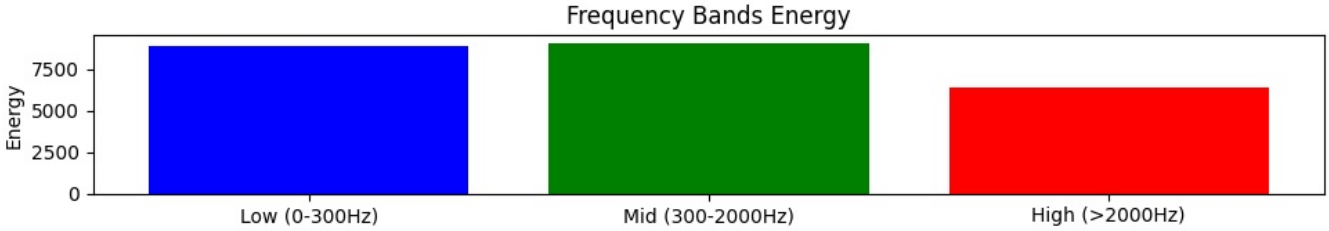
In [11]:

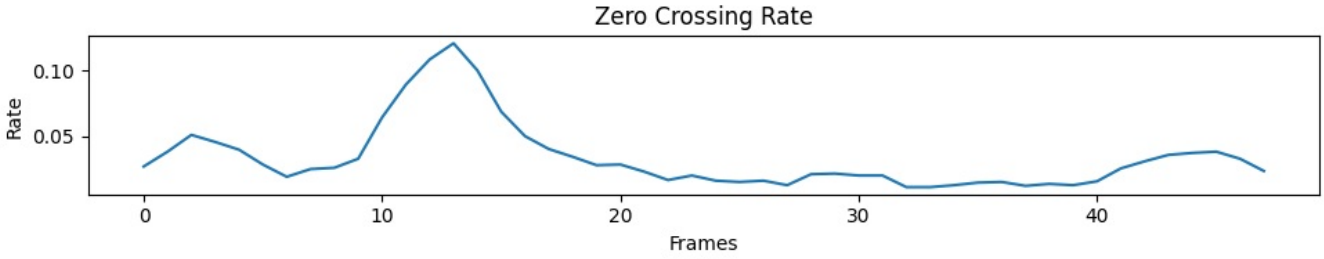
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'doorclosing.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

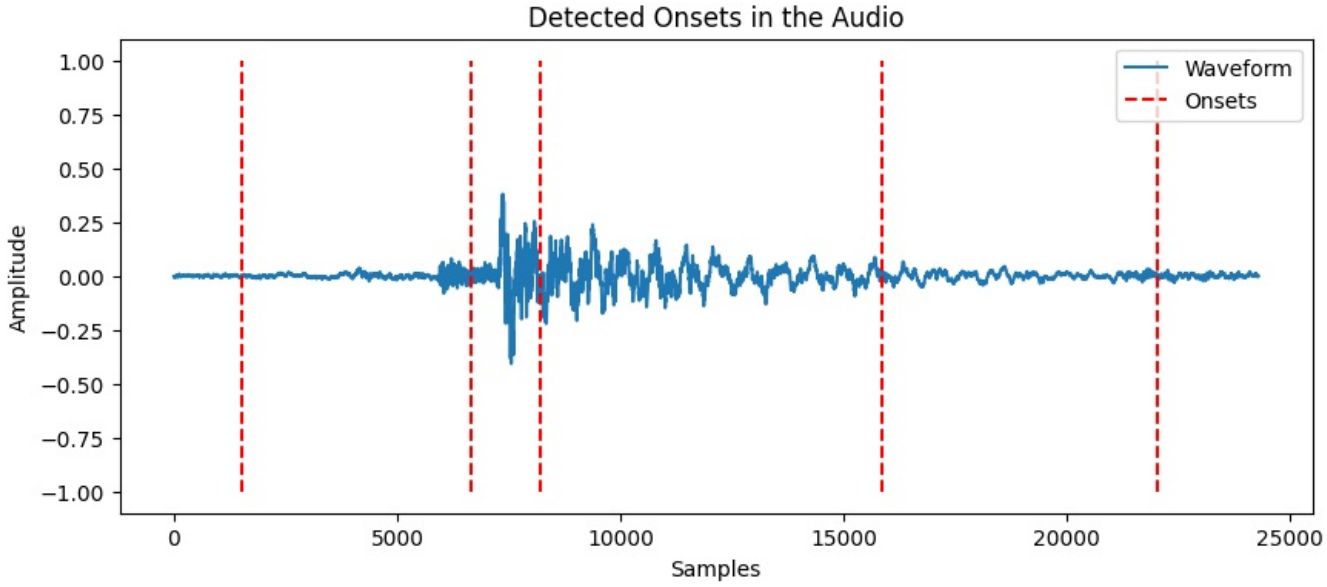
```
plt.show()
```

```
RMS Amplitude: 0.04654492
Low Frequency Energy: 8880.289
Mid Frequency Energy: 9099.405
High Frequency Energy: 6393.6606
Spectral Flux: 13.102825
Zero Crossing Rate: 0.03513179571663921
Onset Times: [0.06965986 0.30185941 0.37151927 0.71981859 0.99845805]
```









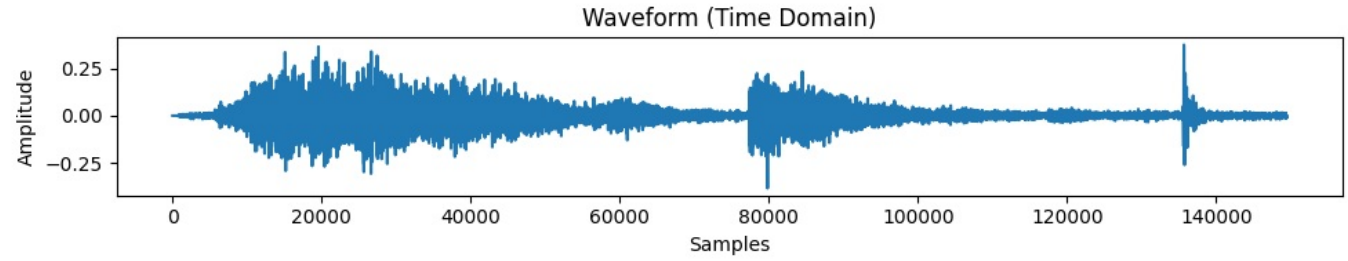
In [ ]:

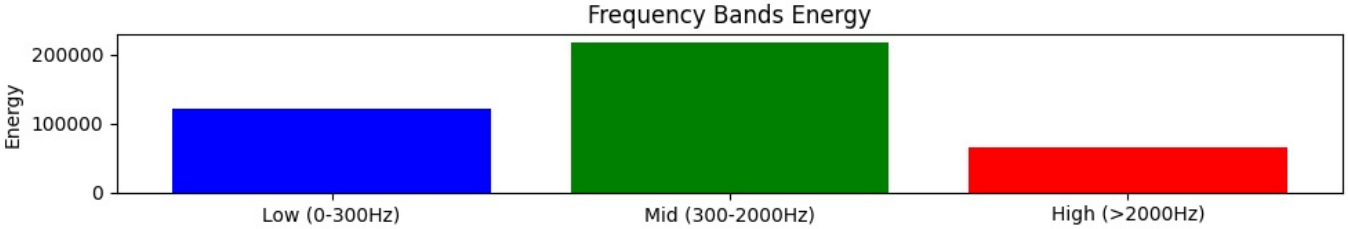
In [9]:

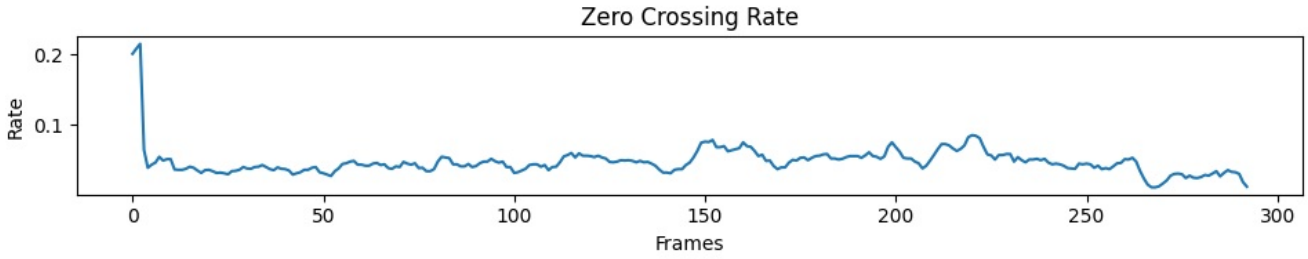
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'fan.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

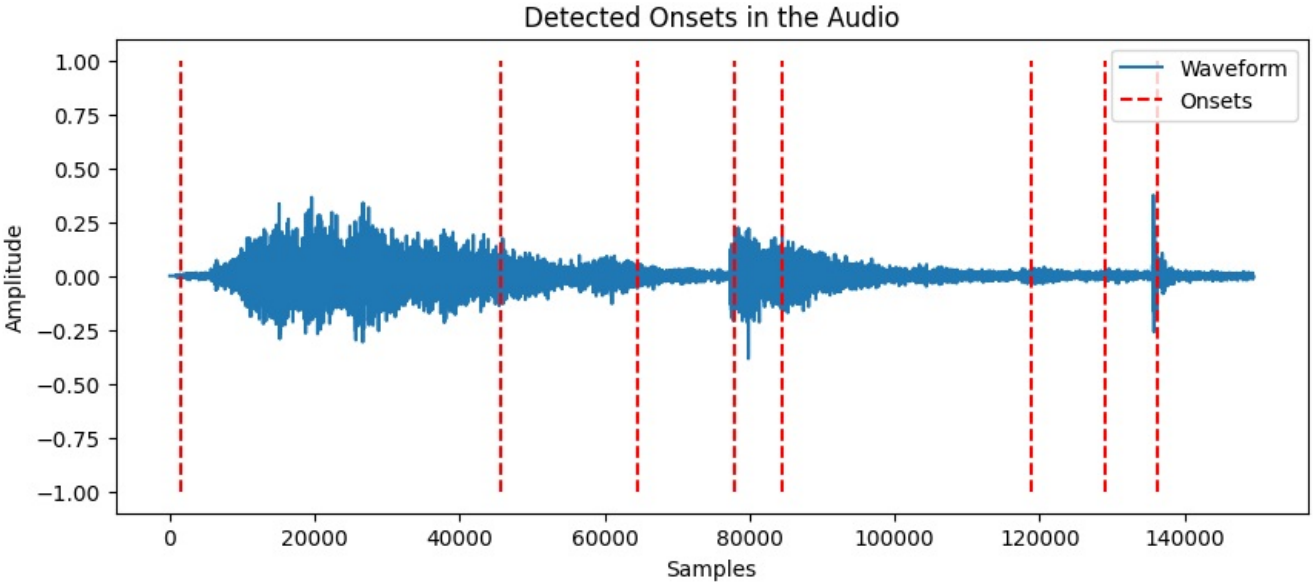
```
plt.show()
```

```
RMS Amplitude: 0.047595166
Low Frequency Energy: 120725.234
Mid Frequency Energy: 217681.5
High Frequency Energy: 65156.74
Spectral Flux: 15.410598
Zero Crossing Rate: 0.0482863334760274
Onset Times: [0.06965986 2.06657596 2.92571429 3.52943311 3.83129252 5.38702948
 5.85142857 6.17650794]
```









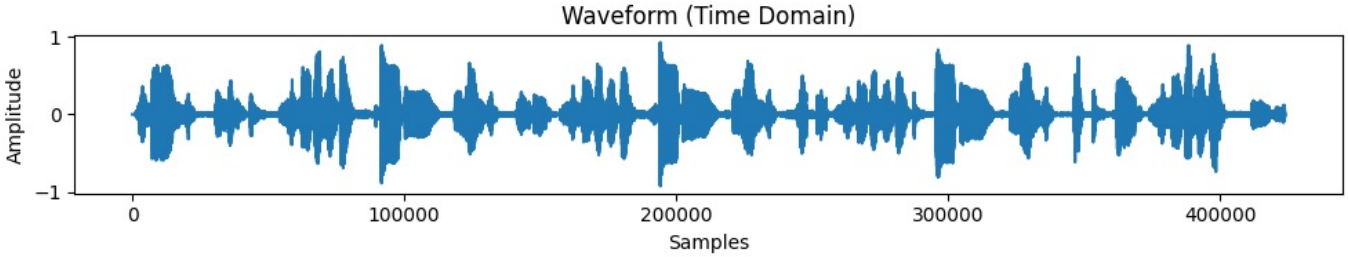
In [ ]:

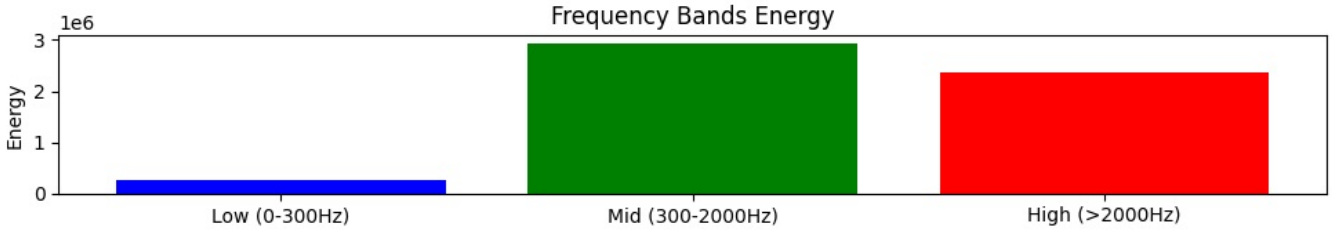
In [10]:

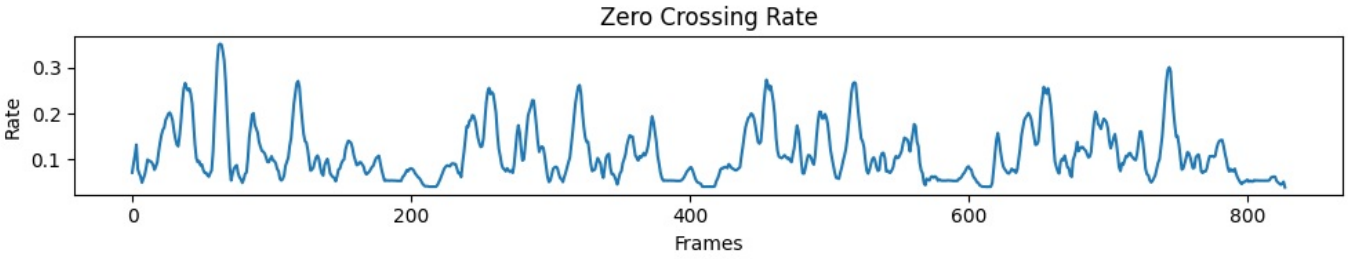
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'paytm.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

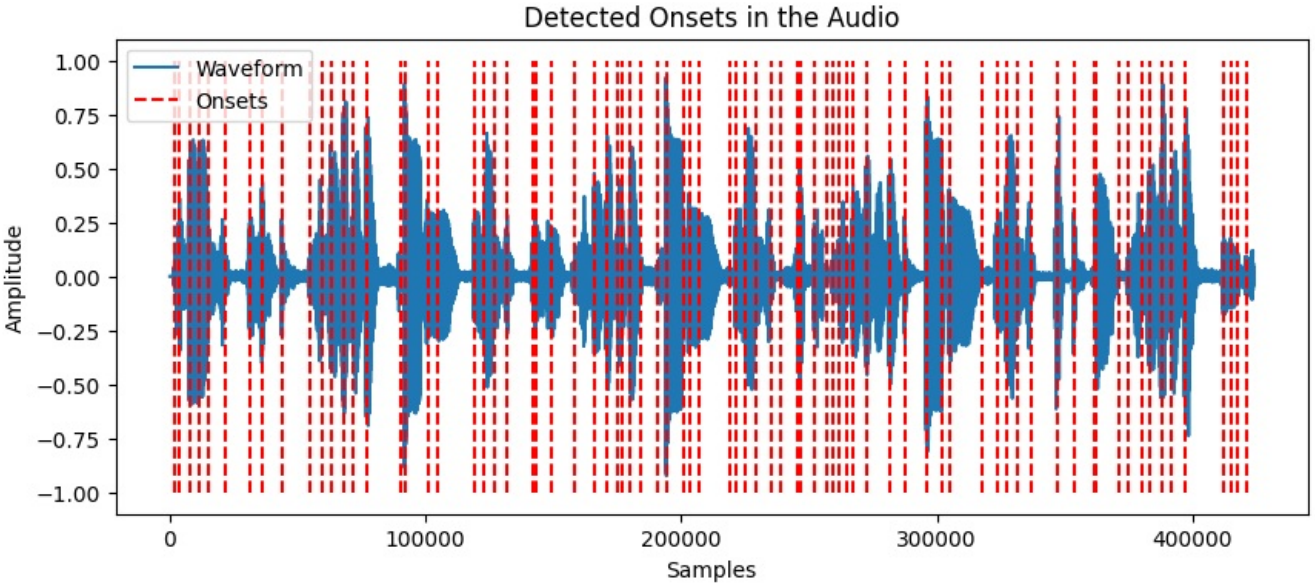
```
plt.show()
```

```
RMS Amplitude: 0.13456026
Low Frequency Energy: 274403.47
Mid Frequency Energy: 2937488.2
High Frequency Energy: 2359341.8
Spectral Flux: 38.38438
Zero Crossing Rate: 0.11195028206997601
Onset Times: [ 0.09287982  0.16253968  0.37151927  0.53405896  0.69659864  0.9752381
  1.41641723  1.64861678  1.9969161   2.48453515  2.71673469  2.85605442
  3.08825397  3.25079365  3.50621315  4.08671202  4.17959184  4.57433107
  4.73687075  5.41024943  5.57278912  5.75854875  5.9907483   6.43192744
  6.5015873   6.7570068   7.17496599  7.54648526  7.75546485  7.94122449
  8.01088435  8.17342404  8.35918367  8.66104308  8.82358277  9.10222222
  9.218322    9.38086168  9.93814059 10.05424036 10.21678005 10.40253968
 10.65795918 10.84371882 11.12235828 11.19201814 11.44743764 11.65641723
 11.77251701 11.86539683 12.00471655 12.12081633 12.35301587 12.77097506
 13.04961451 13.42113379 13.69977324 13.83909297 14.41959184 14.67501134
 14.83755102 15.02331066 15.27873016 15.74312925 16.04498866 16.39328798
 16.43972789 16.83446712 16.9970068  17.2524263  17.36852608 17.60072562
 17.76326531 18.01868481 18.69206349 18.83138322 18.92426304 19.11002268]
```









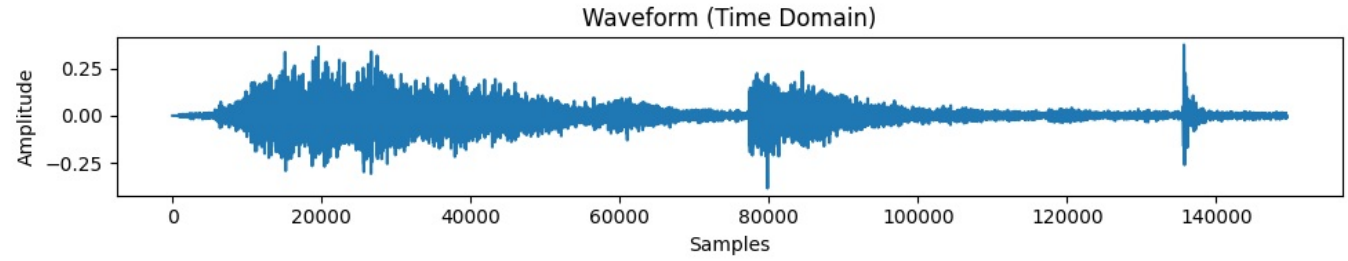
In [ ]:

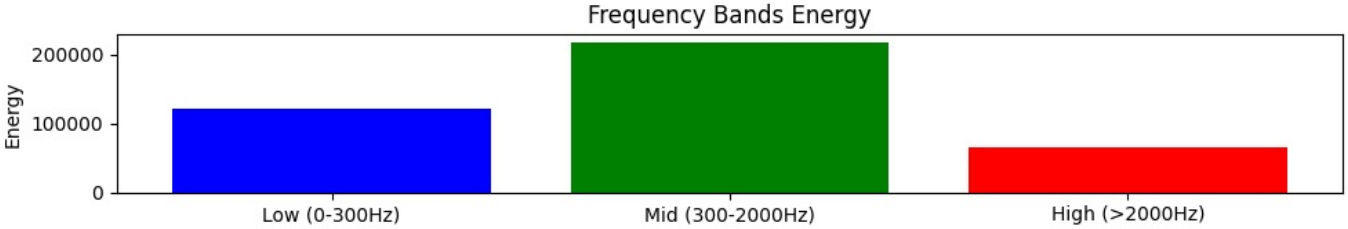
In [9]:

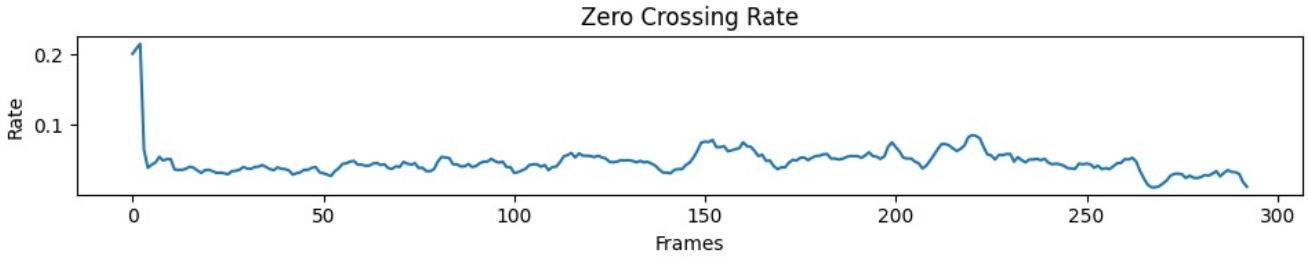
```
import librosa
import matplotlib.pyplot as plt
import numpy as np
import scipy

def calculate_rms_amplitude(audio):
    return np.sqrt(np.mean(np.square(audio)))

def calculate_frequency_bands(audio, sr):
    fft_result = np.fft.fft(audio)
    fft_magnitude = np.abs(fft_result)
    freqs = np.fft.fftfreq(len(fft_magnitude), 1/sr)
    low_freq = np.sum(fft_magnitude[(freqs >= 0) & (freqs <= 300)])
    mid_freq = np.sum(fft_magnitude[(freqs > 300) & (freqs <= 2000)])
    high_freq = np.sum(fft_magnitude[(freqs > 2000)])
    return low_freq, mid_freq, high_freq

def detect_onset(y, sr):
    onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
    onset_times = librosa.frames_to_time(onset_frames, sr=sr)
    return onset_times

def calculate_spectral_flux(audio, sr):
    stft = np.abs(librosa.stft(audio, n_fft=2048, hop_length=512))
    flux = np.sqrt(np.sum(np.diff(stft)**2, axis=0))
    return np.mean(flux)

def calculate_zero_crossing_rate(audio):
    return np.mean(librosa.zero_crossings(audio, pad=False))

# Load the audio file
file_path = 'peopletalking.wav'
audio, sr = librosa.load(file_path)

# Calculate features
rms_amplitude = calculate_rms_amplitude(audio)
low_freq, mid_freq, high_freq = calculate_frequency_bands(audio, sr)
spectral_flux = calculate_spectral_flux(audio, sr)
zero_crossing_rate = calculate_zero_crossing_rate(audio)
onset_times = detect_onset(audio, sr)

# Print results
print("RMS Amplitude:", rms_amplitude)
print("Low Frequency Energy:", low_freq)
print("Mid Frequency Energy:", mid_freq)
print("High Frequency Energy:", high_freq)
print("Spectral Flux:", spectral_flux)
print("Zero Crossing Rate:", zero_crossing_rate)
print("Onset Times:", onset_times)

# Plotting the graphs
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(audio)
plt.title("Waveform (Time Domain)")
plt.xlabel("Samples")
plt.ylabel("Amplitude")

plt.subplot(3, 1, 2)
freq_labels = ['Low (0-300Hz)', 'Mid (300-2000Hz)', 'High (>2000Hz)']
freq_values = [low_freq, mid_freq, high_freq]
plt.bar(freq_labels, freq_values, color=['blue', 'green', 'red'])
plt.title("Frequency Bands Energy")
plt.ylabel("Energy")

plt.subplot(3, 1, 3)
zcr = librosa.feature.zero_crossing_rate(audio)
plt.plot(zcr[0])
plt.title("Zero Crossing Rate")
plt.xlabel("Frames")
plt.ylabel("Rate")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 4))
plt.plot(audio, label='Waveform')
plt.vlines(onset_times * sr, -1, 1, color='r', linestyle='--', label='Onsets')
plt.title('Detected Onsets in the Audio')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()
```

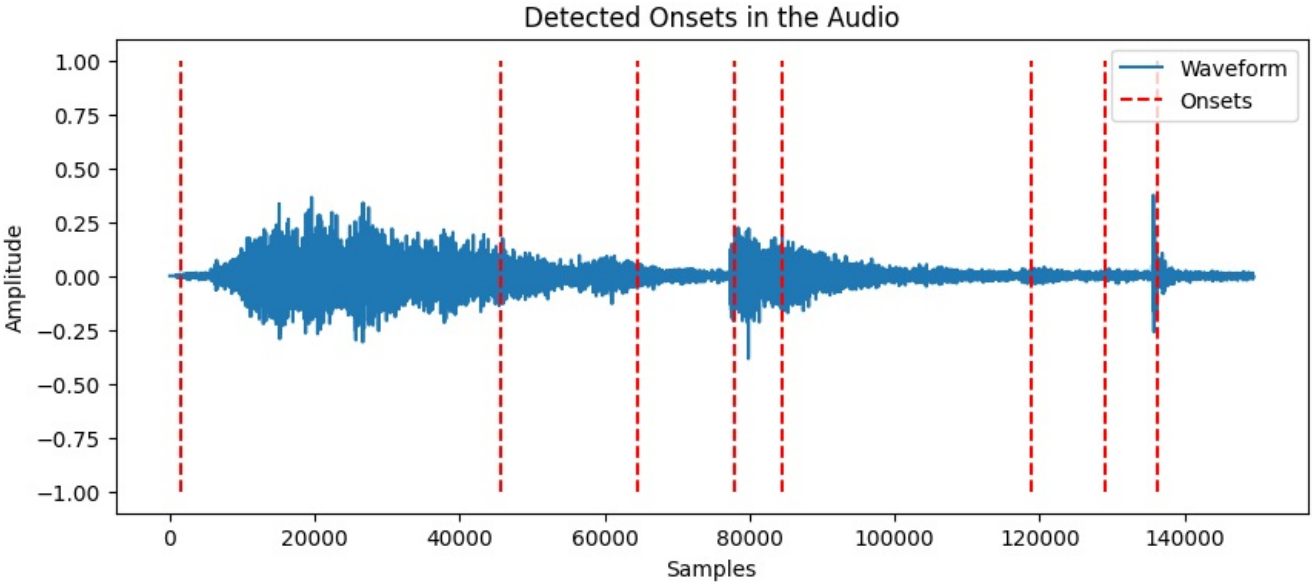
```
plt.show()
```

```
RMS Amplitude: 0.047595166
Low Frequency Energy: 120725.234
Mid Frequency Energy: 217681.5
High Frequency Energy: 65156.74
Spectral Flux: 15.410598
Zero Crossing Rate: 0.0482863334760274
Onset Times: [0.06965986 2.06657596 2.92571429 3.52943311 3.83129252 5.38702948
 5.85142857 6.17650794]
```









In [ ]: