Análisis de sonido

In [ ]:
```
import librosa
```

```
/usr/local/lib/python3.7/dist-packages/resampy/interpn.py:114: NumbaWarning: The TBB threading layer requires TBB version 2019.5 or later i.e., TBB_INTERF
ACE_VERSION >= 11005. Found TBB_INTERFACE_VERSION = 9107. The TBB threading layer is disabled.
  _resample_loop_p(x, t_out, interp_win, interp_delta, num_table, scale, y)
```

In [ ]:
```
# Se importa el archivo muestra
audio_data = 'audio_muestra.mp3'
x , sr = librosa.load(audio_data)
print(type(x), type(sr))
```

```
/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
  warnings.warn("PySoundFile failed. Trying audioread instead.")
<class 'numpy.ndarray'> <class 'int'>
```

In [ ]:
```
librosa.load(audio_data, sr=44100)
```

```
/usr/local/lib/python3.7/dist-packages/librosa/core/audio.py:165: UserWarning: PySoundFile failed. Trying audioread instead.
  warnings.warn("PySoundFile failed. Trying audioread instead.")
```

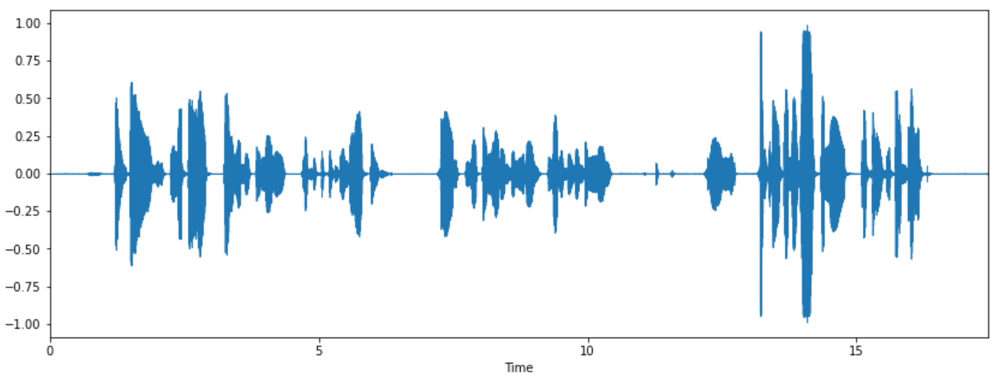
Out[ ]:
```
(array([ 0.0000000e+00,  0.0000000e+00,  0.0000000e+00, ...,
         4.4294598e-06, -3.0724789e-06,  0.0000000e+00], dtype=float32), 44100)
```

In [ ]:
```
import IPython.display as ipd
ipd.Audio(audio_data)
```

Out[ ]:

0:00 / 0:17

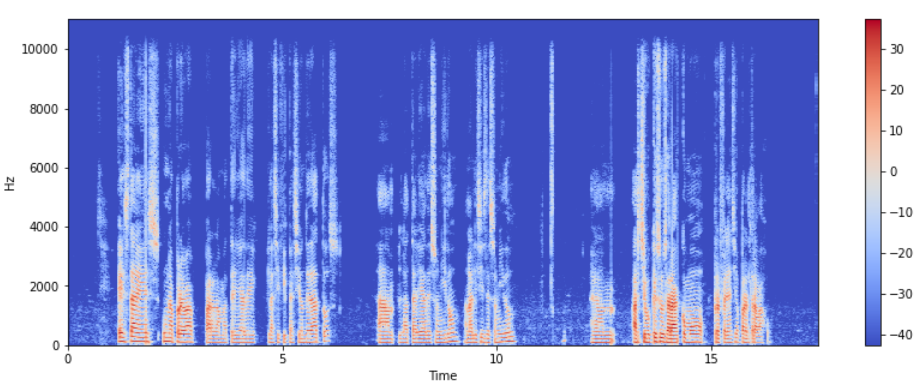
In [ ]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import librosa.display

plt.figure(figsize=(14,5))
librosa.display.waveplot(x, sr=sr)
plt.show()
```



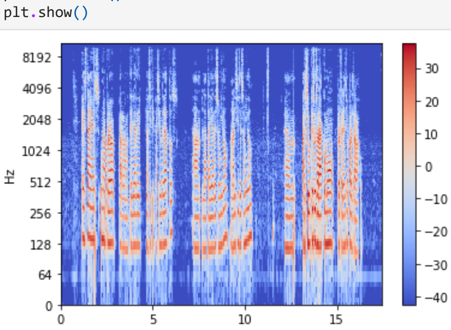
Espectrograma

In [ ]:
```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

Out[ ]:
```
<matplotlib.colorbar.Colorbar at 0x7f30ff35fa90>
```



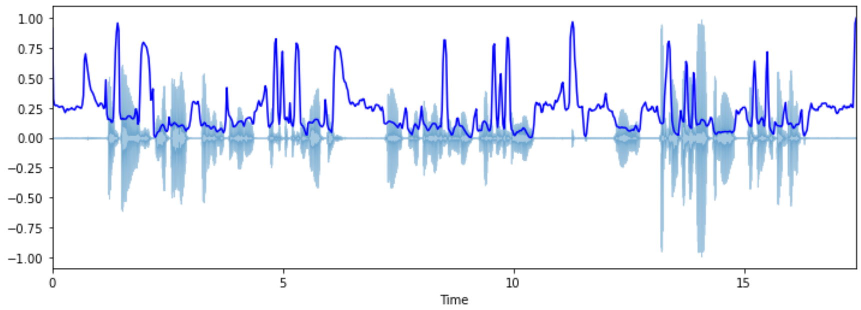
In [ ]:
```
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
plt.show()
```



Centroide Espectral

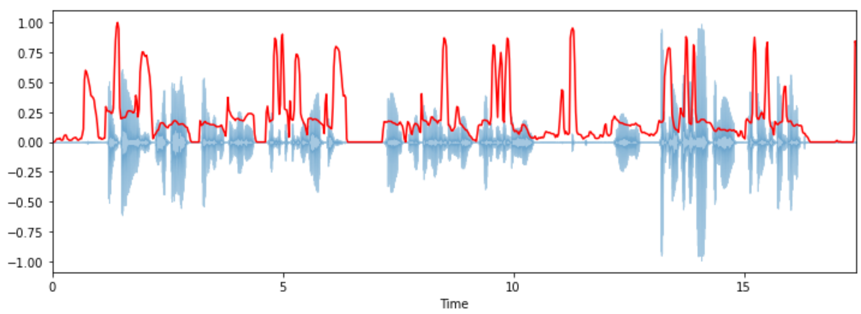
In [ ]:
```
import sklearn
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
(775,)
# Computing the time variable for visualization
plt.figure(figsize=(12, 4))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)
```

```
# Normalising the spectral centroid for visualisation
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
#Plotting the Spectral Centroid along the waveform
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='b')
plt.show()
```



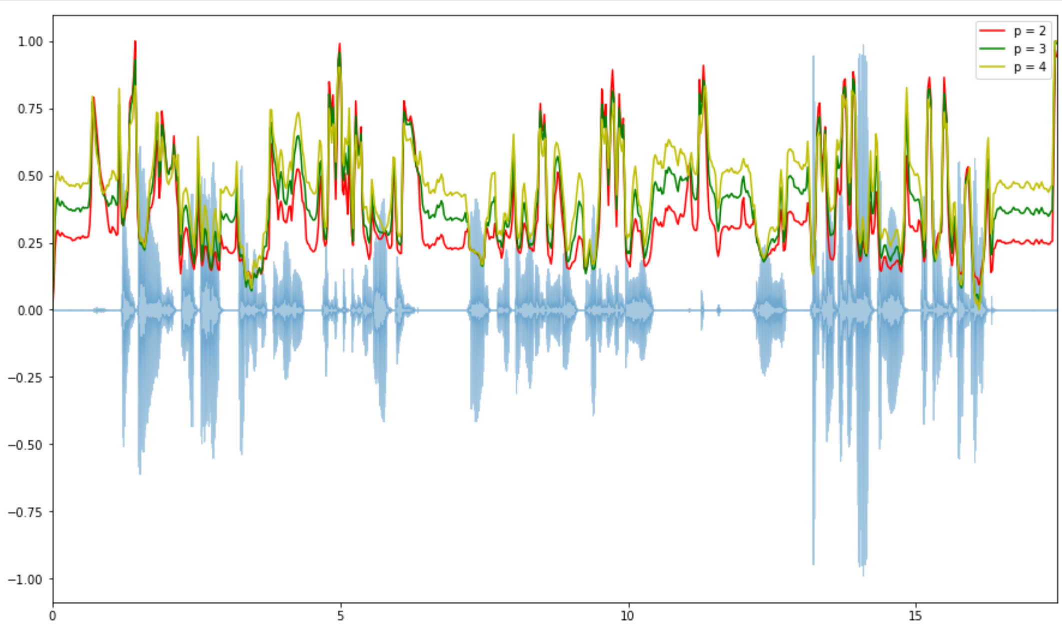
Rolloff Espectral

In [ ]:
```
spectral_rolloff = librosa.feature.spectral_rolloff(x+0.01, sr=sr)[0]
plt.figure(figsize=(12, 4))
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.show()
```



Ancho de Banda Espectral

In [ ]:
```
spectral_bandwidth_2 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr)[0]
spectral_bandwidth_3 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=3)[0]
spectral_bandwidth_4 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=4)[0]
plt.figure(figsize=(15, 9))
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_bandwidth_2), color='r')
plt.plot(t, normalize(spectral_bandwidth_3), color='g')
plt.plot(t, normalize(spectral_bandwidth_4), color='y')
plt.legend(('p = 2', 'p = 3', 'p = 4'))
plt.show()
```



Clasificador de Géneros Musicales

In [ ]:
```
import librosa
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
from PIL import Image
import pathlib
import csv
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

import keras
from keras import layers
from keras import layers
import keras
from keras.models import Sequential
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
cmap = plt.get_cmap('inferno')
plt.figure(figsize=(8,8))
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    pathlib.Path(f'img_data/{g}').mkdir(parents=True, exist_ok=True)
```

```
    for filename in os.listdir(f'/content/drive/MyDrive/Colab Notebooks/Data/genres_original/{g}'):
        songname = f'/content/drive/MyDrive/Colab Notebooks/Data/genres_original/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=5)
        plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode='default', scale='dB');
        plt.axis('off');
        plt.savefig(f'img_data/{g}/{filename[:-3].replace(".", "")}.png')
        plt.clf()
```

```
<Figure size 576x576 with 0 Axes>
```

In [ ]:
```
header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff zero_crossing_rate'
for i in range(1, 21):
    header += f' mfcc{i}'
header += ' label'
header = header.split()
```

In [ ]:
```
file = open('dataset.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    for filename in os.listdir(f'/content/drive/MyDrive/Colab Notebooks/Data/genres_original/{g}'):
        songname = f'/content/drive/MyDrive/Colab Notebooks/Data/genres_original/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30)
        rmse = librosa.feature.rms(y=y)[0]
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        to_append += f' {g}'
        file = open('dataset.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

In [ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data/features_30_sec.csv')
data.head()# Dropping unneccesary columns
data = data.drop(['filename'],axis=1)#Encoding the Labels
genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(genre_list)#Scaling the Feature columns
scaler = StandardScaler()
X = scaler.fit_transform(np.array(data.iloc[:, :-1], dtype = float))#Dividing data into training and Testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [ ]:
```
model = Sequential()
model.add(layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
classifier = model.fit(X_train,
                    y_train,
                    validation_split=0.20,
                    epochs=50,
                    batch_size=32)
```

```
Epoch 1/50
20/20 [==============================] - 1s 11ms/step - loss: 1.9457 - accuracy: 0.3250 - val_loss: 1.5992 - val_accuracy: 0.4563
Epoch 2/50
20/20 [==============================] - 0s 5ms/step - loss: 1.3216 - accuracy: 0.5500 - val_loss: 1.1856 - val_accuracy: 0.5625
Epoch 3/50
20/20 [==============================] - 0s 5ms/step - loss: 0.9691 - accuracy: 0.7016 - val_loss: 1.0044 - val_accuracy: 0.6625
Epoch 4/50
20/20 [==============================] - 0s 4ms/step - loss: 0.7435 - accuracy: 0.7516 - val_loss: 0.9021 - val_accuracy: 0.7312
Epoch 5/50
20/20 [==============================] - 0s 4ms/step - loss: 0.5942 - accuracy: 0.8219 - val_loss: 0.8647 - val_accuracy: 0.7563
Epoch 6/50
20/20 [==============================] - 0s 5ms/step - loss: 0.4780 - accuracy: 0.8391 - val_loss: 0.8852 - val_accuracy: 0.7000
Epoch 7/50
20/20 [==============================] - 0s 5ms/step - loss: 0.3802 - accuracy: 0.8891 - val_loss: 0.8935 - val_accuracy: 0.7250
Epoch 8/50
20/20 [==============================] - 0s 4ms/step - loss: 0.3122 - accuracy: 0.9141 - val_loss: 0.9130 - val_accuracy: 0.7312
Epoch 9/50
20/20 [==============================] - 0s 5ms/step - loss: 0.2465 - accuracy: 0.9359 - val_loss: 0.9067 - val_accuracy: 0.7250
Epoch 10/50
20/20 [==============================] - 0s 4ms/step - loss: 0.2007 - accuracy: 0.9563 - val_loss: 0.9137 - val_accuracy: 0.7312
Epoch 11/50
20/20 [==============================] - 0s 4ms/step - loss: 0.1564 - accuracy: 0.9703 - val_loss: 0.9025 - val_accuracy: 0.7500
Epoch 12/50
20/20 [==============================] - 0s 4ms/step - loss: 0.1280 - accuracy: 0.9797 - val_loss: 0.9538 - val_accuracy: 0.7437
Epoch 13/50
20/20 [==============================] - 0s 4ms/step - loss: 0.1176 - accuracy: 0.9750 - val_loss: 0.9786 - val_accuracy: 0.7125
Epoch 14/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0892 - accuracy: 0.9875 - val_loss: 1.0741 - val_accuracy: 0.7563
Epoch 15/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0641 - accuracy: 0.9953 - val_loss: 1.0864 - val_accuracy: 0.7375
Epoch 16/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0546 - accuracy: 0.9984 - val_loss: 1.0791 - val_accuracy: 0.7437
Epoch 17/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0402 - accuracy: 0.9984 - val_loss: 1.0904 - val_accuracy: 0.7437
Epoch 18/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0339 - accuracy: 1.0000 - val_loss: 1.1256 - val_accuracy: 0.7437
Epoch 19/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0317 - accuracy: 1.0000 - val_loss: 1.1208 - val_accuracy: 0.7563
Epoch 20/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0299 - accuracy: 0.9984 - val_loss: 1.1462 - val_accuracy: 0.7375
Epoch 21/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0202 - accuracy: 1.0000 - val_loss: 1.2001 - val_accuracy: 0.7437
Epoch 22/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0193 - accuracy: 1.0000 - val_loss: 1.2206 - val_accuracy: 0.7437
Epoch 23/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0169 - accuracy: 1.0000 - val_loss: 1.2132 - val_accuracy: 0.7563
Epoch 24/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0136 - accuracy: 1.0000 - val_loss: 1.2295 - val_accuracy: 0.7563
Epoch 25/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0125 - accuracy: 1.0000 - val_loss: 1.2352 - val_accuracy: 0.7563
Epoch 26/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0121 - accuracy: 1.0000 - val_loss: 1.2644 - val_accuracy: 0.7500
Epoch 27/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0103 - accuracy: 1.0000 - val_loss: 1.2894 - val_accuracy: 0.7312
Epoch 28/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0090 - accuracy: 1.0000 - val_loss: 1.2934 - val_accuracy: 0.7437
Epoch 29/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0088 - accuracy: 1.0000 - val_loss: 1.3306 - val_accuracy: 0.7250
Epoch 30/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0081 - accuracy: 1.0000 - val_loss: 1.3185 - val_accuracy: 0.7437
Epoch 31/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0071 - accuracy: 1.0000 - val_loss: 1.3306 - val_accuracy: 0.7563
Epoch 32/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0060 - accuracy: 1.0000 - val_loss: 1.3417 - val_accuracy: 0.7500
Epoch 33/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0054 - accuracy: 1.0000 - val_loss: 1.3446 - val_accuracy: 0.7437
Epoch 34/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0049 - accuracy: 1.0000 - val_loss: 1.3741 - val_accuracy: 0.7312
Epoch 35/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0046 - accuracy: 1.0000 - val_loss: 1.3644 - val_accuracy: 0.7437
Epoch 36/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 1.3867 - val_accuracy: 0.7250
Epoch 37/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 1.3874 - val_accuracy: 0.7437
Epoch 38/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 1.3937 - val_accuracy: 0.7375
Epoch 39/50
20/20 [==============================] - 0s 6ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 1.4177 - val_accuracy: 0.7312
Epoch 40/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0033 - accuracy: 1.0000 - val_loss: 1.4237 - val_accuracy: 0.7437
Epoch 41/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0031 - accuracy: 1.0000 - val_loss: 1.4260 - val_accuracy: 0.7375
Epoch 42/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 1.4370 - val_accuracy: 0.7375
Epoch 43/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 1.4453 - val_accuracy: 0.7375
Epoch 44/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 1.4611 - val_accuracy: 0.7375
Epoch 45/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 1.4620 - val_accuracy: 0.7375
Epoch 46/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 1.4746 - val_accuracy: 0.7375
Epoch 47/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 1.4782 - val_accuracy: 0.7375
Epoch 48/50
20/20 [==============================] - 0s 4ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 1.4867 - val_accuracy: 0.7375
Epoch 49/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 1.4977 - val_accuracy: 0.7375
Epoch 50/50
20/20 [==============================] - 0s 5ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 1.5032 - val_accuracy: 0.7375
```