	Universidade Federal de Uberlândia **Faculdade de Computação** **Bacharelado em Ciência da Computação**
	GBC213 – Multimídia
Laboratório 2	**Áudio**
Prof. Dr. Marcelo Zanchetta do Nascimento	

Informações:

- Deve ser elaborado um arquivo no CoLab (Google – arquivo extensão .ipynb) para resolução dos exercícios desse laboratório. Deve ser inserido comentários nos programas desenvolvidos (use o símbolo #);
- Os exercícios 5 e 6 devem ser entregues pelo Moodle-UFU. O arquivo *.ipynb deve ser comprimido em conjunto com os arquivos de áudio e enviado ao professor pelo Moodle-UFU até a data máxima de entrega. Posteriormente será publicado o gabarito para os demais exercícios dessa atividade;
- Colocar um cabeçalho do arquivo os nomes, números de RA dos membros da equipe. Inicie o arquivo com o cabeçalho:

```
#Nome dos alunos:
#RA:
#Laboratório: <inserir o número e assunto>
```

Introdução

A linguagem Python possui diversas bibliotecas para manipulação da mídia de representação “áudio”. Segue alguns exemplos de bibliotecas:

- PySoundFile
- scipy.io.wavfile (from scipy)
- wave(to read streams. Included in python 2 and 3)
- scikits.audiolab (that seems unmaintained)
- sounddevice(play and record sounds, good for streams and real-time)
- pyglet
- Neste laboratório vamos trabalhar com algumas dessas bibliotecas para manipulação e processamento de áudio.

Para ter acesso aos arquivos na CoLab, você deve importar seu drive dentro do notebook. Então, devemos usar:

```
1. # acesso ao google drive para importar imagens, áudio e dados
2. from google.colab import drive
3. drive.mount('/content/gdrive')
```

Exercícios

1) A biblioteca “scipy” permite ler um arquivo de som em formato “wav” do Windows.

```
1. from scipy.io.wavfile import read, write
2. from IPython.display import Audio

3. root_path = '/content/gdrive/My Drive/Colab Notebooks/'
4. Fs, data = read(root_path + 'Lab2_Audio/test.wav')
5. data = data[:,0]
6. Audio(data,rate=Fs)
```

a) Implemente o código e análise os arquivos disponíveis: test.wav e bird.wav. Implemente o comando **`Audio`** (player) para que possa ouvir os arquivos de áudio.

b) Crie um novo programa para leitura dos arquivos (test.wav e bird.wav) e faça a representação gráfica da forma de onda dos arquivos de áudio. As funções necessárias para plotar o gráfico de áudio são apresentadas abaixo.

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
    ...

plt.figure()
plt.plot(data)
plt.xlabel('Amostra')
plt.ylabel('Amplitude')
plt.title('Teste de Áudio')
plt.show()
```

c) Mostre os dados desse arquivos (vetor), taxa de amostragem, tamanho do áudio, a amplitude mais alta e amplitude mais baixa.

d) Uma outra forma de visualizar os dados de um arquivo de áudio é por meio de um espectrograma. O espectrograma é uma forma de visualizar a intensidade de um sinal através do tempo em várias frequências. Para visualizar o espectrograma de um áudio podemos usar esses comandos:

```
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io.wavfile import read

root_path = '/content/gdrive/My Drive/Colab Notebooks/'
Fs, data = read(root_path + 'Lab2_Audio/bird.wav')

frequencies, times, spectrogram = signal.spectrogram(data, Fs)
```

```
plt.pcolormesh(times, frequencies, spectrogram)
plt.imshow(spectrogram)
plt.ylabel('Frequência [Hz]')
plt.xlabel('Tempo [sec]')
plt.show()
```

2) Implemente o código em um notebook do Google CoLab e análise o que é realizado em cada uma das linhas deste código.

```
from IPython.display import Audio

Fs = 8000
T = 1/Fs
t = 0.1
N = Fs*t

freq = 1000
omega = 2*np.pi*freq

t_vec = np.arange(N)*T
y = np.sin(omega*t_vec)

plt.plot(t_vec,y)
plt.show()
```

a) O que representa o valor 1000 na linha 6? Altere este valor para 500 e mostre o que é possível concluir com essa modificação no código? Construa um gráfico para mostrar o resultado desta alteração em relação aos parâmetros originais.

b) O que representa a variável **`fs`**? Altere os valores da **fs** para 4 kHz, 16 kHz, 22 kHz. O que é possível concluir?

c) Altere a amplitude do sinal (10 vezes superior à amplitude inicial e 1/10 da amplitude inicial). O que é possível observar?

3) Implemente o código abaixo em um notebook do CoLab e análise o que está ocorrendo para responder os questionamentos:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
from scipy.io.wavfile import read, write

b_normalize=True
root_path = '/content/gdrive/My Drive/Colab Notebooks/'
sr, s = read(root_path + 'Lab2_Audio/test.wav')
s = s[:,0]

if b_normalize:
        s = s.astype(np.float32)
        s = (s / np.max(np.abs(s)))
        s -= np.mean(s)
```

a) O que é realizado neste código? Implemente esse código para a realização dessa operação sobre 2 arquivos de áudio. O que ocorre com esses áudios após aplicação desta etapa.

b) Plote os gráficos e análise os arquivos antes e após aplicação dessas etapas sobre os arquivos de áudio. Use o player para ouvir esses sons e análise se ocorreu alguma mudança na representação gráfica ou na qualidade do áudio.

4) O código abaixo mostra a criação de um novo áudio com parâmetros definidos para digitalização deste código. Faça a implementação e novas modificações relacionados às taxas de amostragem e frequência. Salve esses arquivos e observe as variações de acordo com um Player. Neste caso use uma ferramenta de edição de áudio, como por exemplo, o Audacity para uma análise mais detalhada desses arquivos.

```
from scipy.io.wavfile import write
import numpy as np
import matplotlib.pyplot as plt

samplerate = 44100
fs = 100
t = np.linspace(0., 1., samplerate)
amplitude = np.iinfo(np.int16).max
data = amplitude * np.sin(2. * np.pi * fs * t)

root_path = '/content/gdrive/My Drive/Colab Notebooks/'
write(root_path + "Lab2_Audio/test-1.wav", samplerate, data)
```

5) Construa um programa para demonstrar a teoria de amostragem de Nyquist. Use também o modelo para apresentar o problema de *aliasing* quando a amostragem não respeita a teoria de Nyquist.

6) Capture duas gravações curtas de áudio, sendo a primeira com o som 'o' sustentado, falado em tom alto (ou seja, 'ooooooo') e a segunda gravação deve ser o mesmo som, falado em tom baixo. Use o software Audacity para converter os formatos, salvando-os em mono com 8 kHz taxa de amostragem. Então, no CoLab, crie um programa que carregue os dois arquivos e analise os gráficos da forma de onda e espectrograma. Realize uma discussão se ocorre uma diferença entre os dois sinais usando esses gráficos?

Segue alguns datasets com arquivos de áudio:

- https://www.caito.de/2019/01/03/the-m-ailabs-speech-dataset/
- http://www.openslr.org/12/
- https://github.com/soerenab/AudioMNIST/tree/master/data