# Lab 06 - Telecomunicações
# Tales Araujo Kodama - 2023001218

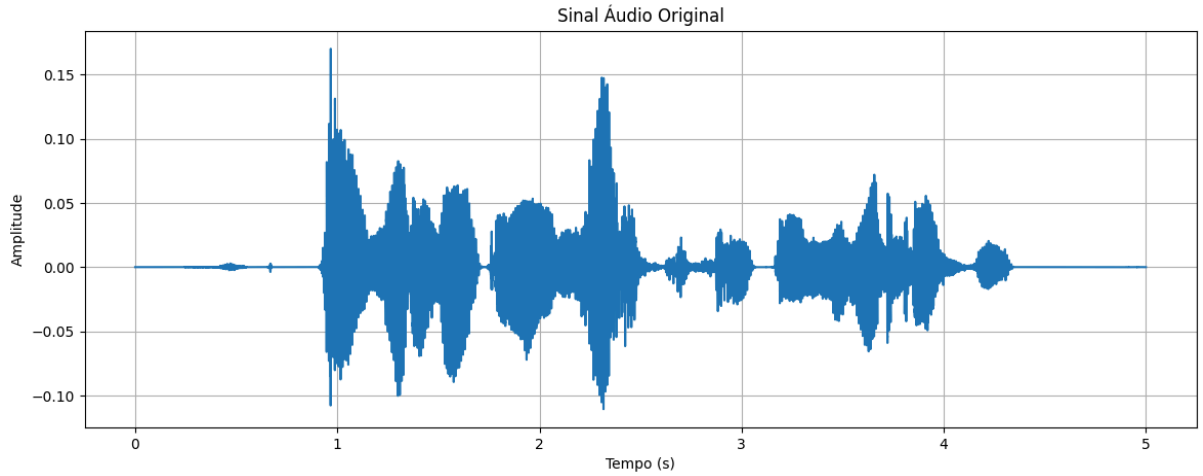
## Experimento 1 -

```
import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd
from scipy.io.wavfile import write, read
from IPython.display import Audio
```

## Item a -
## Código:

```
fs_original = 8000
duracao = 5

print(f"Iniciando a gravação por 5 segundos...")

my_audio = sd.rec(int(duracao * fs_original), samplerate=fs_original,
channels=1, dtype='float32')
sd.wait()

print("Gravação finalizada.")
write('audio_original.wav', fs_original, my_audio)
```

```
t_original = np.linspace(0, duracao, len(my_audio), endpoint=False)

plt.figure(figsize=(14, 5))
plt.plot(t_original, my_audio)
plt.title('Sinal Áudio Original')
plt.xlabel('Tempo (s)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()

print("Aúdio gerado: ")
Audio(data=my_audio.flatten(), rate=fs_original)
```




Acima temos o gráfico do sinal do áudio gravado. Foi dita a frase: Palmeiras não tem mundial e São Paulo maior do Brasil.

**Item b -**

**Código:**

```
print(f"Taxa de amostragem escolhida: {fs_original} Hz")
```

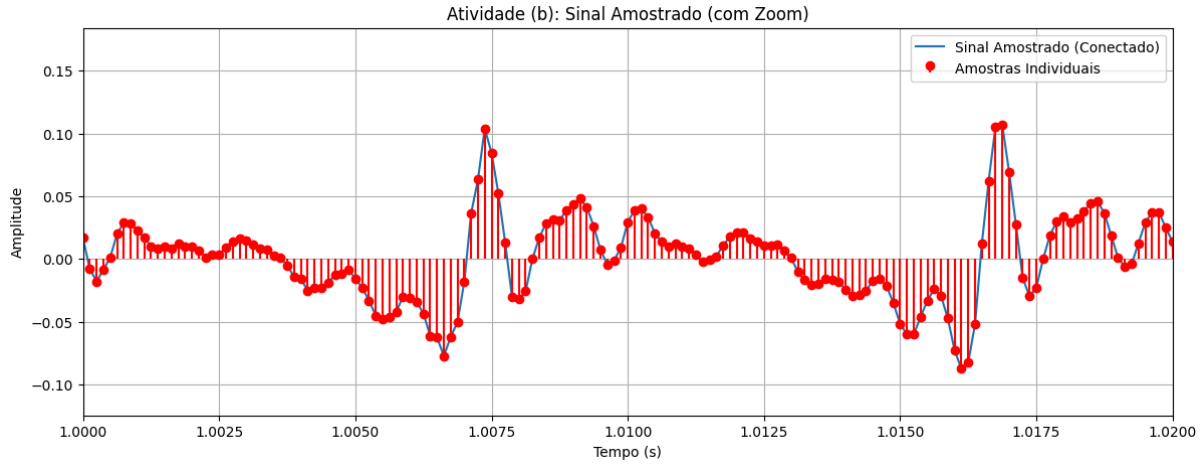
```
Taxa de amostragem escolhida: 8000 Hz
```

A taxa de amostragem de 8000 Hz foi escolhida para digitalizar a voz humana (frequência máxima de 3400 Hz) sem *aliasing*. Conforme o Teorema de Nyquist, a amostragem mínima necessária é de 6800 Hz (o dobro da frequência máxima). A taxa de 8000 Hz, padrão na telefonia, atende a esse requisito.

**Código:**

```
plt.figure(figsize=(14, 5))
plt.plot(t_original, my_audio, label='Sinal Amostrado (Conectado)')
plt.stem(t_original, my_audio, linefmt='r-', markerfmt='ro', basefmt='
', label='Amostras Individuais')
plt.title('Atividade (b): Sinal Amostrado (com Zoom)')
plt.xlabel('Tempo (s)')
plt.ylabel('Amplitude')
plt.grid(True)

plt.xlim(1.0, 1.02)
plt.legend()
plt.show()
```

## Gráfico obtido :

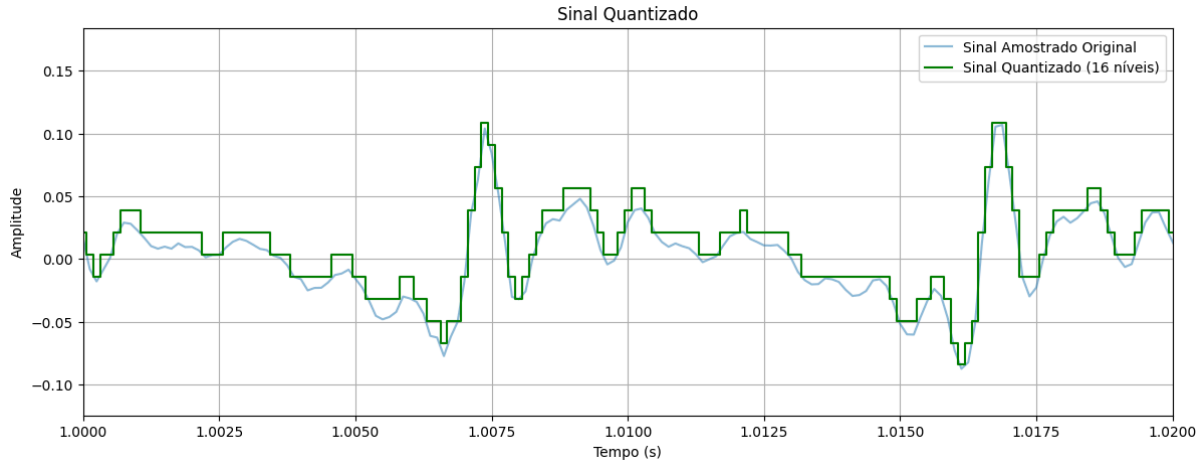



## Item c -

## Código:

```
n_bits = 4
M = 2**n_bits

V_min = np.min(my_audio)
V_max = np.max(my_audio)

delta_v = (V_max - V_min) / M

indices = np.round((my_audio - V_min) / delta_v)
indices = np.clip(indices, 0, M - 1)
audio_quantizado = V_min + delta_v/2 + indices * delta_v

plt.figure(figsize=(14, 5))
plt.plot(t_original, my_audio, label='Sinal Amostrado Original',
alpha=0.5)
plt.step(t_original, audio_quantizado, where='mid', label=f'Sinal
Quantizado ({M} níveis)', color='green')
plt.title('Sinal Quantizado')
plt.xlabel('Tempo (s)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.xlim(1.0, 1.02)

plt.legend()
plt.show()
```

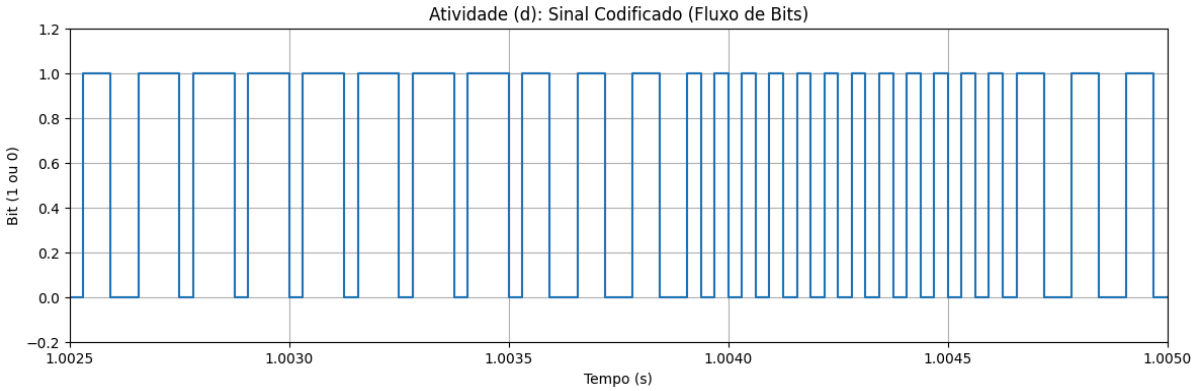
## Gráfico:




Aqui foi utilizado 4 bits, o que resulta em 2^4 = 16 níveis. Podemos visualizar o sinal original em azul e o sinal quantizado em verde.

## Item d -
## Código:

```
fluxo_de_bits = []
for indice in indices:

    codigo_binario = format(int(indice), f'0{n_bits}b')

    fluxo_de_bits.extend([int(b) for b in codigo_binario])

sinal_codificado = np.array(fluxo_de_bits)
num_total_de_bits = len(sinal_codificado)
t_codificado = np.linspace(0, duracao, num_total_de_bits,
endpoint=False)

plt.figure(figsize=(14, 4))
plt.step(t_codificado, sinal_codificado, where='post')
plt.title('Atividade (d): Sinal Codificado (Fluxo de Bits)')
plt.xlabel('Tempo (s)')
plt.ylabel('Bit (1 ou 0)')
plt.grid(True)
plt.ylim(-0.2, 1.2)

plt.xlim(1.0025, 1.0050)

plt.show()
```




Aqui podemos visualizar um trecho da codificação, foi agrupado com 4 bits, assim podemos ter 16 combinações de 0000 a 1111, no gráfico podemos visualizar a variação dos códigos.

**Item e -**

**Código:**

```
audio_quantizado_float = audio_quantizado.astype('float32')
write('audio_quantizado_4bit.wav', fs_original, audio_quantizado_float)

print("Áudio Original: ")
display(Audio(data=my_audio.flatten(), rate=fs_original))
```

**Código:**

```
print("Áudio Final: ")
display(Audio(data=audio_quantizado_float.flatten(), rate=fs_original))
```

Acima temos os códigos que geram os áudios original e o quantizado. Quando escutamos os dois áudios, notamos que o áudio quantizado fica meio robotizado, com ruído.
Isso ocorre por conta do erro de quantização, que é grande pois utilizamos 4 bits. Mas se aumentarmos o número de bits, o passo de quantização fica menor, logo o erro de quantização também fica menor. Assim temos um áudio de melhor qualidade e menos robotizado.