In [2]:
```
# Importacoes


import numpy as np
import pandas as pd
import scipy.io.wavfile as wavfile
import matplotlib.pyplot as plt
import warnings

from scipy.fft import rfft, rfftfreq
from scipy.signal import butter, filtfilt
from IPython.display import Audio
```

In [3]:
```
# Ajustes no Notebook


%matplotlib inline
warnings.filterwarnings('ignore')
```

In [48]:
```
# Funcao para visualizar analises


def plot_analise(x, y, titulo, titulo_eixo_x, ver_eixo_y=False):
    plt.figure(figsize=(24, 8))

    plt.title(f'{titulo}')

    plt.xlabel(f'{titulo_eixo_x}')

    figure = plt.gca()
    y_axis = figure.axes.get_yaxis()
    y_axis.set_visible(False)

    plt.plot(x, y)

    plt.show()
```

In [43]:
```
# Leitura do arquivo .wav


microondas = 'Microondas.wav'

taxa, som = wavfile.read(microondas)
sinal = som[:, 0]
duracao = len(sinal) / freq

print(f'Taxa de Amostragem: {taxa} Hz')
print(f'Duracao: {duracao:.2f}s')
```

```
Taxa de Amostragem: 44100 Hz
Duracao: 10.27s
```

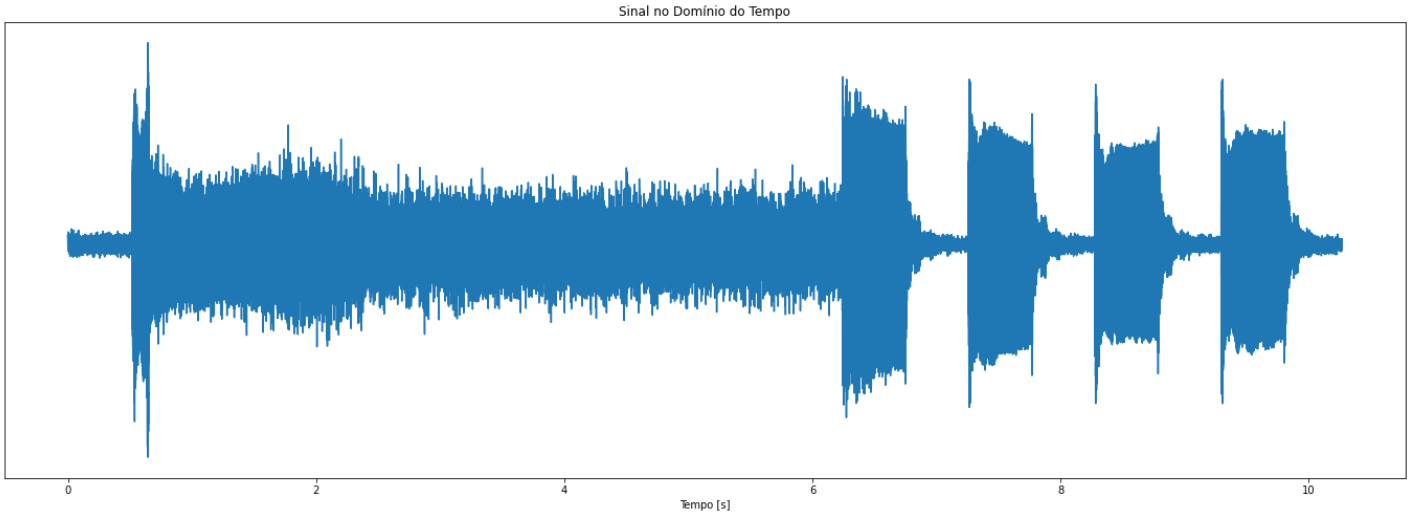
In [8]:
```
# Testando execucao
# Recomenda-se abaixar o volume


Audio(microondas)
```

Out[8]:

0:00 / 0:10

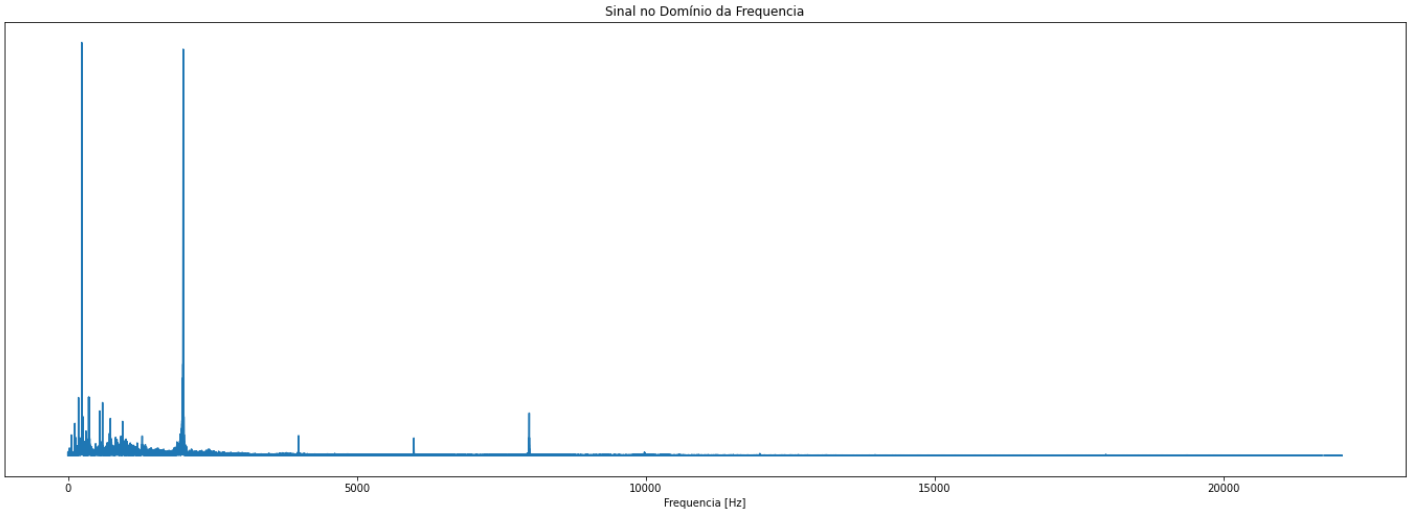
In [51]:

```
# Visualizacao do sinal sonoro no dominio do tempo


x = np.linspace(0, duracao, int(taxa * duracao), endpoint=False)
plot_analise(x, sinal, 'Sinal no Domínio do Tempo', 'Tempo [s]')
```



In [53]:

```
# Aplicacao do Algoritmo FFT (Fast Fourier Transform) para obter Sinal no Dominio da Frequ


amplitude = rfft(sinal)
frequencia = rfftfreq(int(taxa * duracao), 1 / taxa)

plot_analise(frequencia, np.abs(amplitude), 'Sinal no Domínio da Frequencia', 'Frequencia
```



In [70]:

```
# Criando tabela de frequencias e suas respectivas amplitudes


amplitude_db = 20 * np.log10(abs(amplitude))

df_frequencias = pd.DataFrame(data={'Frequencia (Hz)': frequencia, 'Amplitude (dB)': ampli
df_frequencias = df_frequencias.round(2)
df_frequencias.head()
```

Out[70]:

	Frequencia (Hz)	Amplitude (dB)
0	0.00	111.41
1	0.10	113.92
2	0.19	117.60
3	0.29	116.24
4	0.39	117.98

In [73]:

```
# Ordenando tabela: 5 maiores amplitudes


df_frequencias_desc = df_frequencias.sort_values(by='Amplitude (dB)', ascending=False)
df_frequencias_desc = df_frequencias_desc.reset_index(drop=True)
df_frequencias_desc.head()
```

Out[73]:

	Frequencia (Hz)	Amplitude (dB)
0	239.97	172.65
1	1995.98	172.51
2	1996.18	172.43
3	1996.28	171.73
4	1995.89	171.45

In [76]:

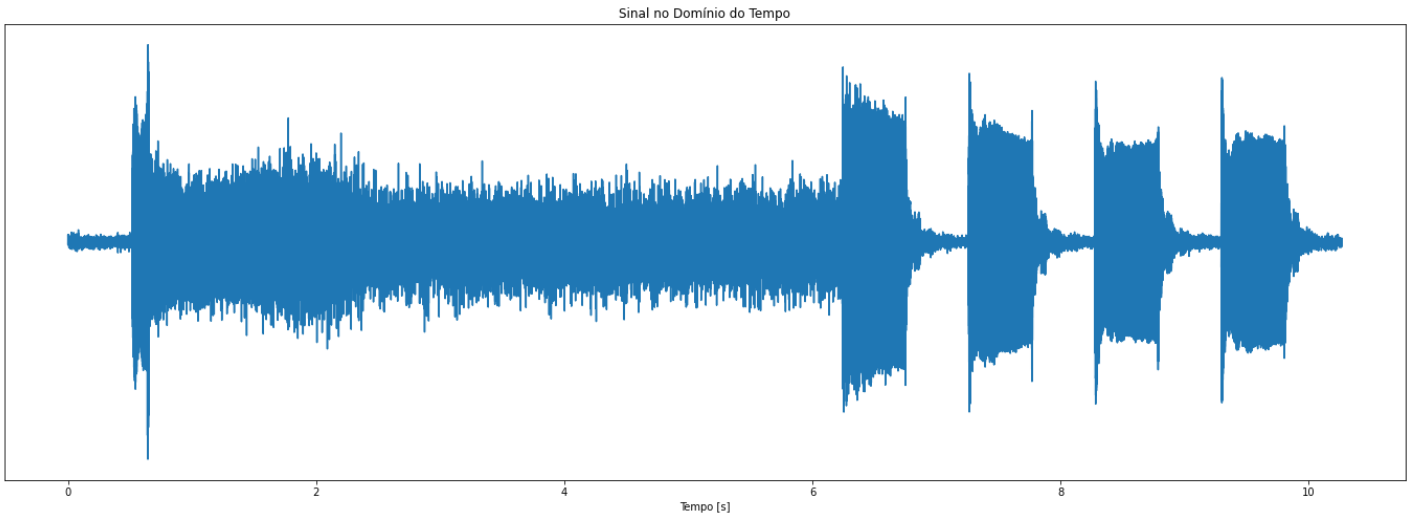
```
# Funcao responsavel pela aplicacao do filtro passa baixa


def filtro_passa_baixa(sinal, corte, nyq, taxa_amostragem, ordem):
    corte_normalizado = corte / nyq

    b, a = butter(ordem, corte_normalizado, btype='low', analog=False)
    sinal_filtrado = filtfilt(b, a, sinal)

    return sinal_filtrado
```

In [93]:

```
# Configurando Filtro


periodo = 5.0
taxa_amostragem = 30.0
corte = 2
nyq = 0.5 * taxa_amostragem
ordem = 5
num_amostras = int(periodo * taxa_amostragem)
```

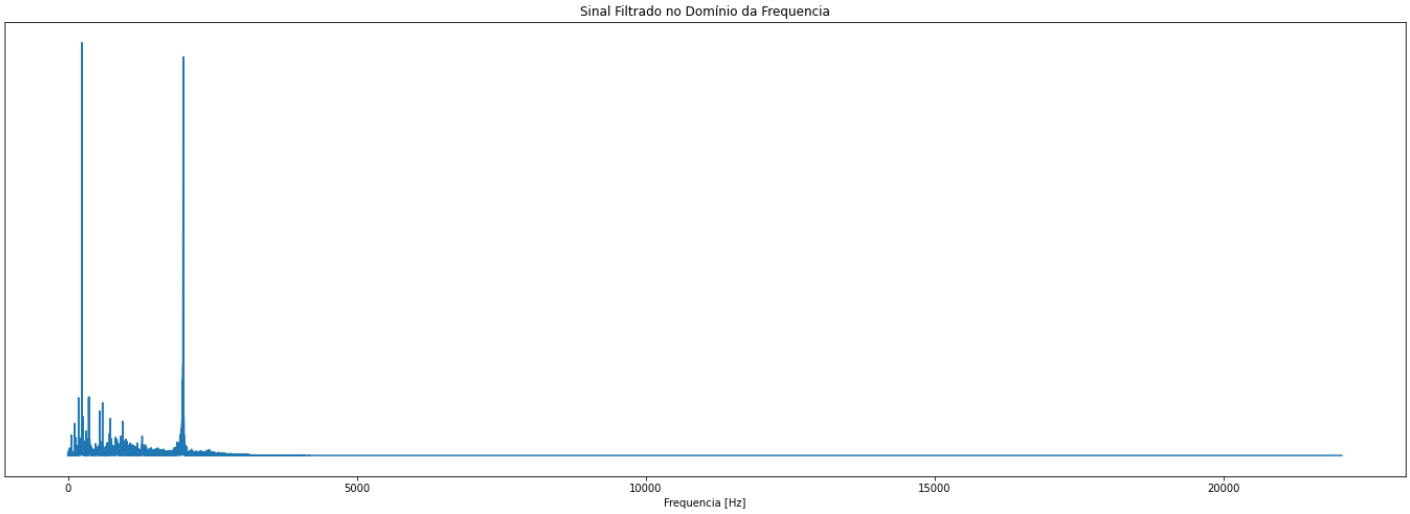
In [94]:

```
# Sinal resultante do filtro passa baixa


sinal_filtrado = filtro_passa_baixa(sinal, corte, nyq, taxa_amostragem, ordem)

plot_analise(x, sinal_filtrado, 'Sinal no Domínio do Tempo', 'Tempo [s]')
```



In [99]:

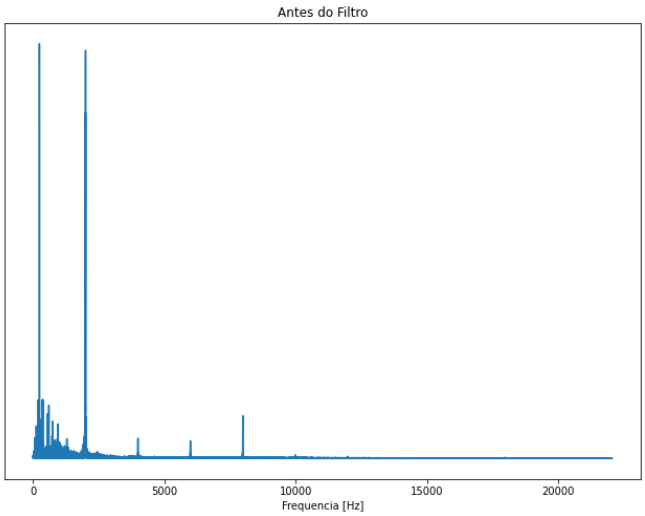
```
# Aplicacao do Algoritmo FFT (Fast Fourier Transform) para obter o Sinal Filtrado no Domi

amplitude_filtro = rfft(sinal_filtrado)
frequencia_filtro = rfftfreq(int(taxa * duracao), 1 / taxa)

plot_analise(frequencia_filtro, np.abs(amplitude_filtro), 'Sinal Filtrado no Domínio da Fr
```

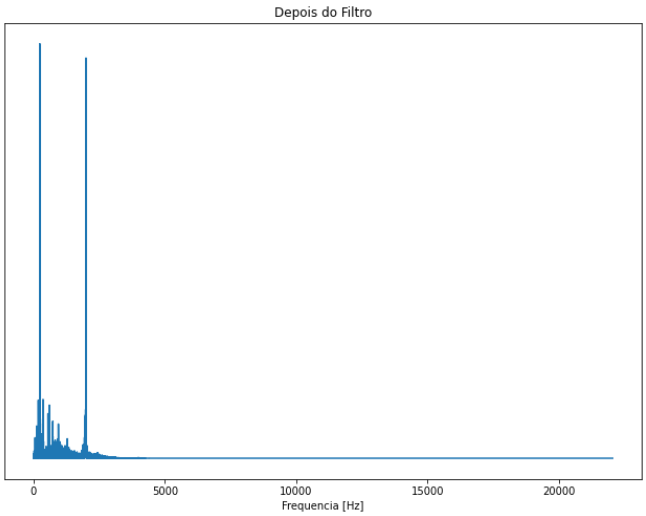


In [140…

```
# Construindo visualizacao para comparar efeito do filtro no som analisado

plt.figure(figsize=(24, 8))

plt.subplot(1, 2, 1)
plt.title('Antes do Filtro')
plt.xlabel('Frequencia [Hz]')
plt.plot(frequencia, np.abs(amplitude))

figure = plt.gca()
y_axis = figure.axes.get_yaxis()
y_axis.set_visible(False)

plt.subplot(1, 2, 2)
plt.title('Depois do Filtro')
plt.xlabel('Frequencia [Hz]')
plt.plot(frequencia_filtro, np.abs(amplitude_filtro))

figure = plt.gca()
y_axis = figure.axes.get_yaxis()
```

```
y_axis.set_visible(False)

plt.show()
```





In [143…

```
# Exportando tabelas


df_frequencias_desc.to_csv('tabela_frequencias.csv', index=False)
df_frequencias_desc.to_excel('tabela_frequencias.xlsx', index=False)
```

In [96]:

```
# Exportando som resultante


wavfile.write('.microondas_passa_baixa.wav', taxa, sinal_filtrado.astype(np.int16))
```