In [18]:
```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.signal import welch, lfilter
from scipy import fftpack
```

In [19]:
```
def plot_time_domain_channels_stereo(time, lc, rc):
    #Plota as figuras ao longo do tempo
    #Plota os canais esquerdo e direito
    plt.figure(1,figsize=(20, 5))
    plt.plot(time, lc, label="Canal esquerdo")
    plt.legend()
    plt.xlabel("Tempo [s]")
    plt.ylabel("Amplitude")
    plt.show()

    plt.figure(2,figsize=(20, 5))
    plt.plot(time, rc, color="red",label="Canal direito")
    plt.legend()
    plt.xlabel("Tempo [s]")
    plt.ylabel("Amplitude")
    plt.show()
```

In [20]:
```
def plot_spect_welch_channels_stereo(lc, rc, fs):
    #Sample Frequencies, Power Spectral Density
    sf_lc, psd_lc = welch(
        x=lc,
        fs=fs,
        window='flattop',
        nperseg=512,
        scaling='spectrum'
    )
    sf_rc, psd_rc = welch(
        x=rc,
        fs=fs,
        window='flattop',
        nperseg=512,
        scaling='spectrum'
    )

    #Plota o espectro do sinal para frequencias normalizadas entre 0 1 pi
    #(frequencias positivas)
    plt.subplots(figsize=(15,5))
    plt.subplot(1, 2, 1)
    plt.semilogy(sf_lc, psd_lc, label="Canal esquerdo")
    plt.legend()
    plt.xlabel('Frequencia [rad]')
    plt.ylabel('Espectro')

    plt.subplot(1, 2, 2)
    plt.semilogy(sf_rc, psd_rc, color="red", label="Canal direito")
    plt.legend()
    plt.xlabel('Frequencia [rad]')
    plt.ylabel('Espectro')
    plt.show()
```

In [21]:
```
def plot_spect_fft_channels_stereo(lc,rc, sampling_rate, nfft):
    freq_lc = np.linspace(0., sampling_rate, nfft) #Interpola para determinar
```

```
    sig_fft_lc = fftpack.rfft(lc,nfft)
    plt.subplots(figsize=(15,5))
    plt.subplot(1, 2, 1)
    plt.title("Canal esquerdo")
    plt.plot(freq_lc, np.abs(sig_fft_lc), label="fft")
    plt.legend()
    plt.xlabel('Frequencia [Hz]')
    plt.ylabel('Espectro de amplitudes')
    #plt.plot(freq_lc, np.abs(fftpack.fftshift(sig_fft_lc)), label="fftshift'
    plt.legend()

    freq_rc = np.linspace(0., sampling_rate, nfft) #Interpola para determinar
    sig_fft_rc = fftpack.rfft(rc,nfft)
    plt.subplot(1, 2, 2)
    plt.title("Canal direito")
    plt.plot(freq_rc, np.abs(sig_fft_rc), color="red", label="fft")
    plt.legend()
    plt.xlabel('Frequencia [Hz]')
    plt.ylabel('Espectro de amplitudes')
    #plt.plot(freq_rc, np.abs(fftpack.fftshift(sig_fft_rc)), color="green", l
    plt.legend()
    plt.show()
```

In [22]:

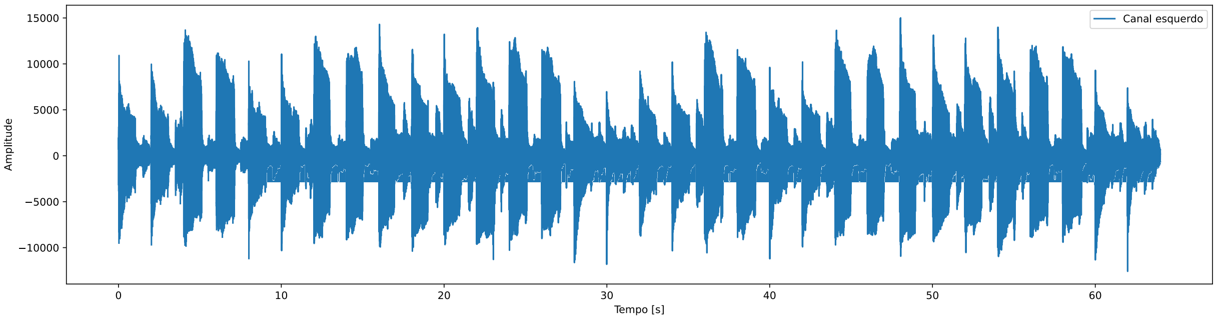
```
#Carrega o arquivo
sampling_rate, data = wavfile.read('569127__josefpres__dark-loops-201-simple-
#sampling_rate, data = wavfile.read('581010__xcreenplay__smoking-in-the-angel

number_of_samples = data.shape[0]
number_of_channels = data.shape[1]

#Tempo total = numero de amostras / fs
duration = number_of_samples / sampling_rate

#Carrega o arquivo em dois canais (audio estereo)
left_channel = data[:, 0]
right_channel  = data[:, 1]

print(f"Numero de canais = {number_of_channels}")
print(f"Duracao = {duration}s")
print(f'Numero de amostras: {number_of_samples}')
print(f"Amostras por segundo: {sampling_rate}Hz")
```

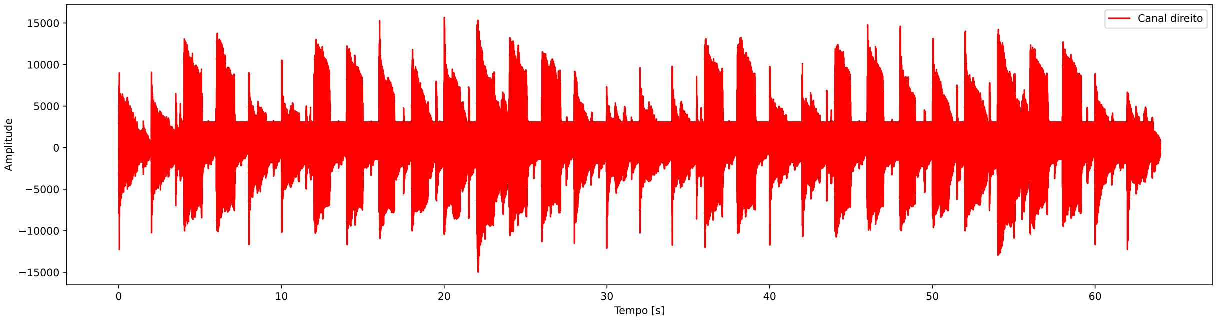
```
Numero de canais = 2
Duracao = 64.0s
Numero de amostras: 2822400
Amostras por segundo: 44100Hz
```

```
/tmp/ipykernel_359008/1384073729.py:2: WavFileWarning: Chunk (non-data) not u
nderstood, skipping it.
  sampling_rate, data = wavfile.read('569127__josefpres__dark-loops-201-simpl
e-mix-2-short-loop-60-bpm.wav')
```

In [23]:

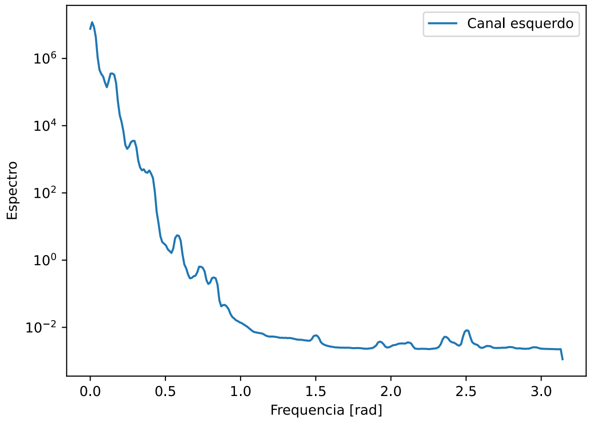
```
#Interpola para determinar eixo do tempo
time = np.linspace(0., duration, number_of_samples)
plot_time_domain_channels_stereo(
    time=time,
    lc=left_channel,
    rc=right_channel
)
```

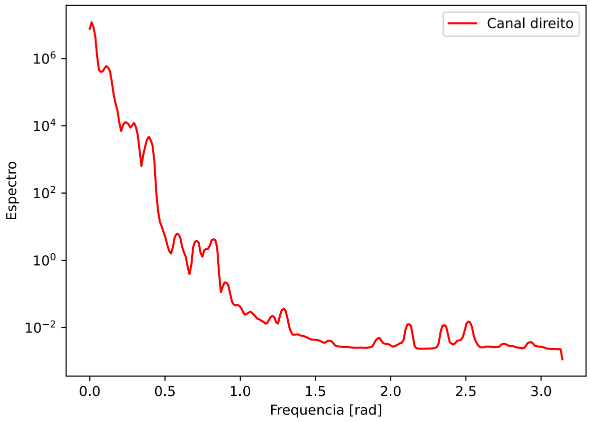


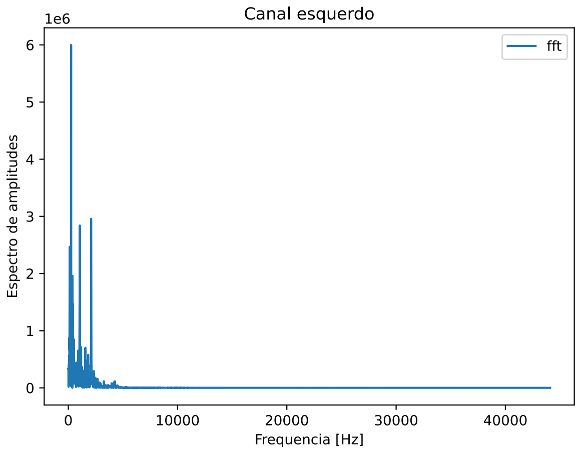


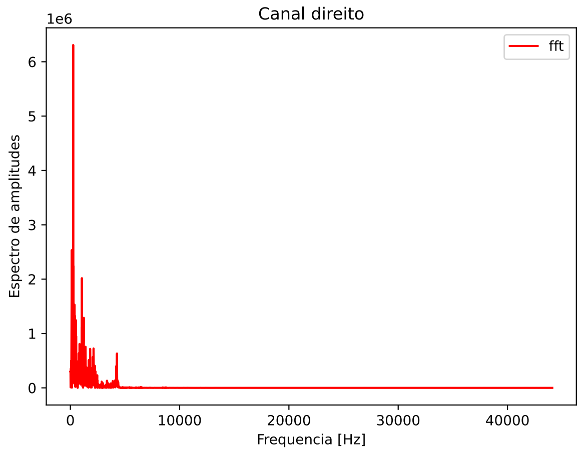
In [24]:

```
plot_spect_welch_channels_stereo(
    lc  = left_channel,
    rc  = right_channel,
    fs = 2*np.pi
)

plot_spect_fft_channels_stereo(
    lc  = left_channel,
    rc  = right_channel,
    sampling_rate=sampling_rate,
    nfft=4096
)
```



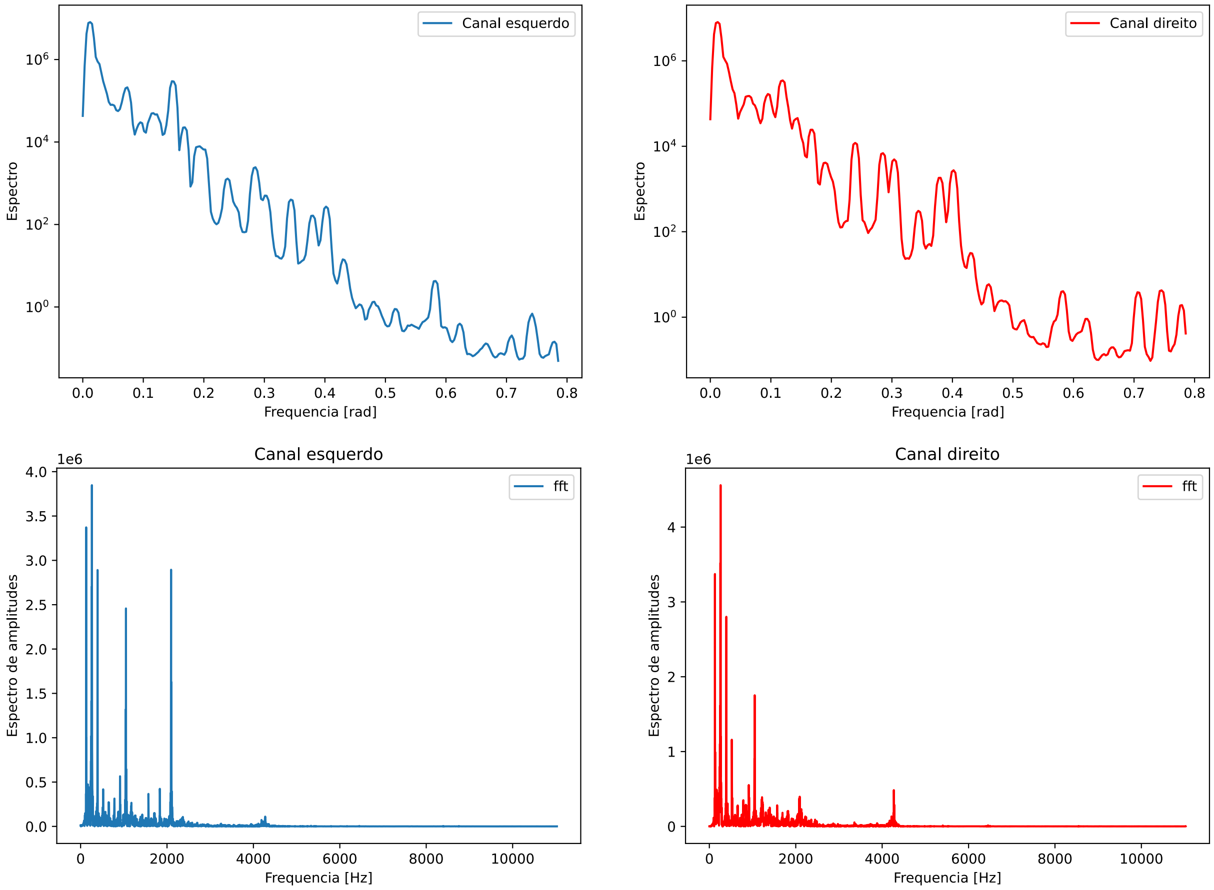






In [25]:
```
#Dizimando o sinal pelo fator M
M=4

decimated_lc = left_channel[0:-1:M]
decimated_rc = right_channel[0:-1:M]
```

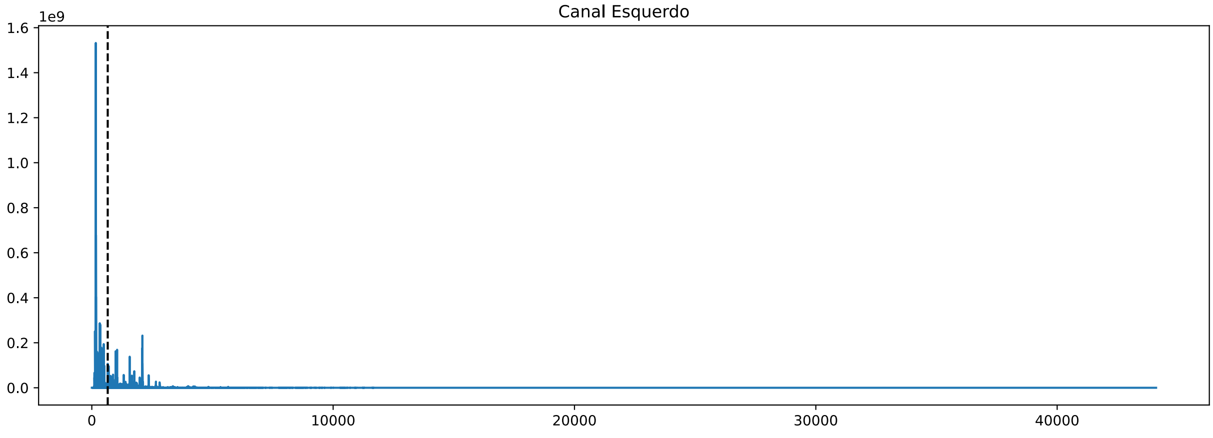
In [26]:
```
plot_spect_welch_channels_stereo(
    lc  = decimated_lc,
    rc  = decimated_rc,
    fs = (2*np.pi)/M
)

plot_spect_fft_channels_stereo(
    lc  = decimated_lc,
    rc  = decimated_rc,
    sampling_rate=sampling_rate//M,
    nfft=4096
)
```



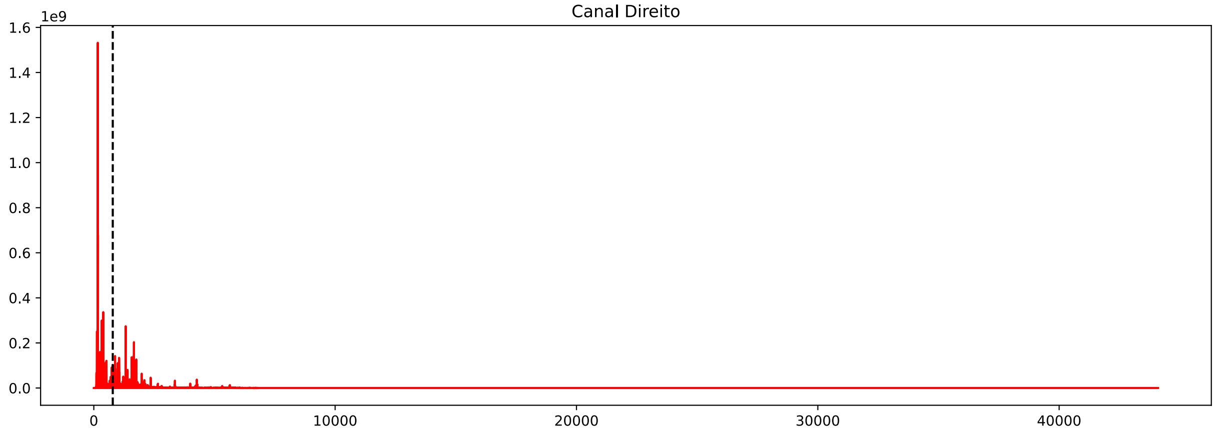
In [27]:
```
freq_lc = np.linspace(0., sampling_rate, number_of_samples) #Interpola para c
sig_fft_lc = fftpack.rfft(left_channel,number_of_samples)

plt.figure(1,figsize=(15, 5))
plt.plot(freq_lc, np.abs(sig_fft_lc[:number_of_samples]))
#"centro de massa" da função no domínio da frequência
peso_lc = 0.0 #porcentagem do sinal acumulado
rangeData_lc = len(sig_fft_lc) #frequências positivas da FFT
sc_lc = sum(abs(sig_fft_lc[:rangeData_lc])) #todo o sinal acumulado
count_lc = 0 # quantidade de amostras somadas

for i in abs(sig_fft_lc[:rangeData_lc]):
  peso_lc = peso_lc + i/sc_lc
  count_lc = count_lc + 1
```

```
  if(peso_lc >= 0.5):
    print(peso_lc)
    break
print(freq_lc[count_lc]) #frequência de corte para o primeiro canal

plt.title("Canal Esquerdo")
plt.axvline(x = freq_lc[count_lc], color = 'black', ls = '--')
plt.show()


freq_rc = np.linspace(0., sampling_rate, number_of_samples) #Interpola para o
sig_fft_rc = fftpack.rfft(right_channel,number_of_samples)

plt.figure(2,figsize=(15, 5))
plt.plot(freq_rc, np.abs(sig_fft_rc[:number_of_samples]), color='red')

peso_rc = 0.0
rangeData_rc = len(sig_fft_rc)
sc_rc = sum(abs(sig_fft_rc[:rangeData_rc]))
count_rc = 0

for i in abs(sig_fft_rc[:rangeData_rc]):
  peso_rc = peso_rc + i/sc_rc
  count_rc = count_rc + 1
  if(peso_rc >= 0.5):
    print(peso_rc)
    break
print(freq_rc[count_rc])

plt.title("Canal Direito")
plt.axvline(x = freq_rc[count_rc], color = 'black', ls = '--')
plt.show()
```

```
0.5000133156517214
659.5002336664661
```




```
0.5000178643250714
785.3284032484422
```



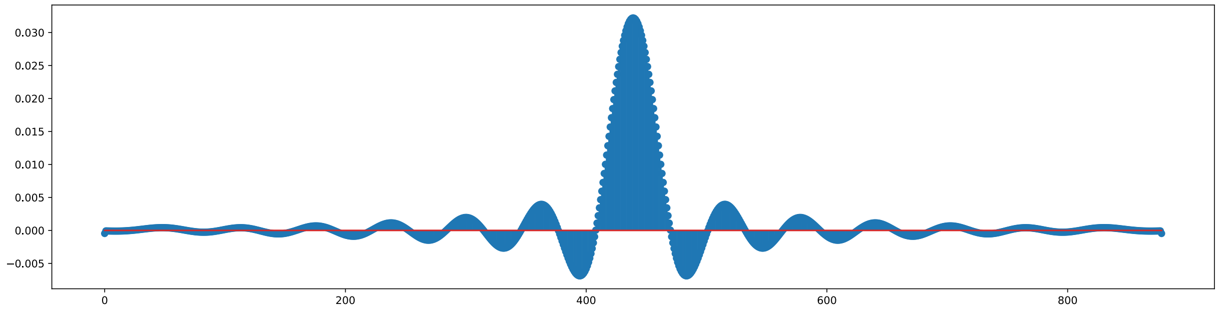
In [28]:

```
aux = np.abs(sig_fft_rc).cumsum()
meio = aux[-1] / 2
valorMeio = aux[aux >= meio][0]
freqMetade = freq_lc[aux >= valorMeio][0]
print("Frequência de metade do conteúdo espectral: ", freqMetade, " Hz")
```

```
Frequência de metade do conteúdo espectral:  785.3127782429062  Hz
```

In [29]:

```
#Carrega os coeficientes do filtro
b = np.genfromtxt('coeffs_pyfda_lp.csv', delimiter=',')
#Plota coeficientes do filtro FIR
plt.figure(7, figsize=(20, 5))
plt.stem(b)
plt.show()
```

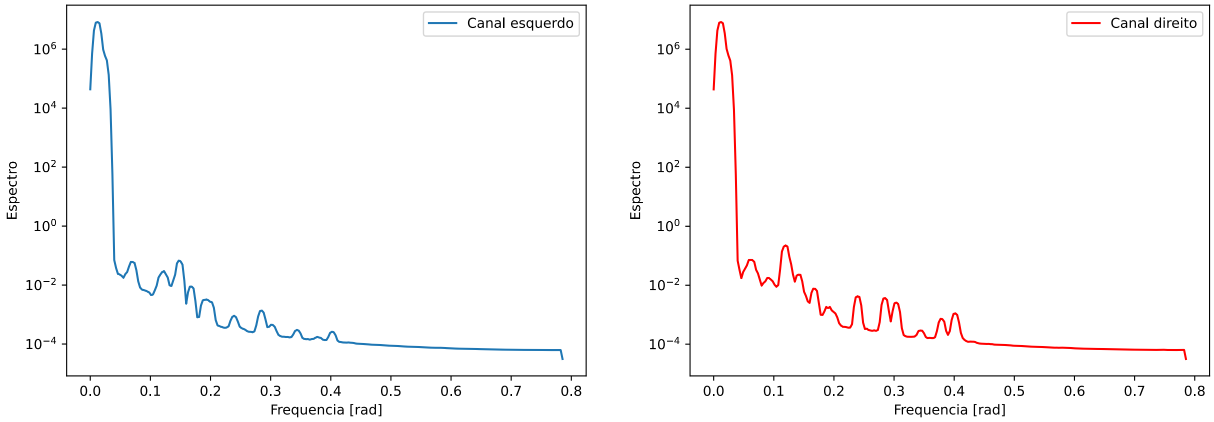


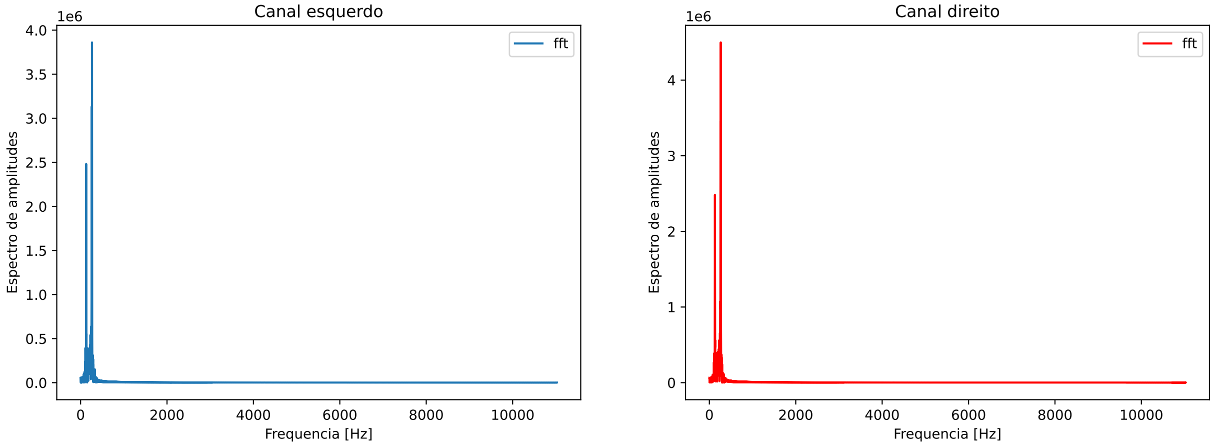
In [30]:

```
#Filtra os dados dos canais esquerdo e direito
decimated_filtered_lc = lfilter(b, 1, decimated_lc)
decimated_filtered_rc = lfilter(b, 1, decimated_rc)
```

In [31]:

```
plot_spect_welch_channels_stereo(
    lc  = decimated_filtered_lc,
    rc  = decimated_filtered_rc,
    fs = (2*np.pi)/M
)

plot_spect_fft_channels_stereo(
    lc  = decimated_filtered_lc,
    rc  = decimated_filtered_rc,
    sampling_rate=sampling_rate//M,
    nfft=4096
)
```





In [32]:

```
# Escrita de arquivo dizimado
audio = np.array([decimated_lc, decimated_rc]).T
scaled = np.int16(audio/np.max(np.abs(audio)) * 32767)
filename = 'signal_decimated_' + str(M) + '.wav'
wavfile.write(filename, sampling_rate//M, scaled)
```

In [33]:

```
# Escrita de arquivo dizimado e filtrado
audio = np.array([decimated_filtered_lc, decimated_filtered_rc]).T
scaled = np.int16(audio/np.max(np.abs(audio)) * 32767)
filename = 'signal_decimated_' + str(M) + '_filted.wav'
wavfile.write(filename, sampling_rate//M, scaled)
```

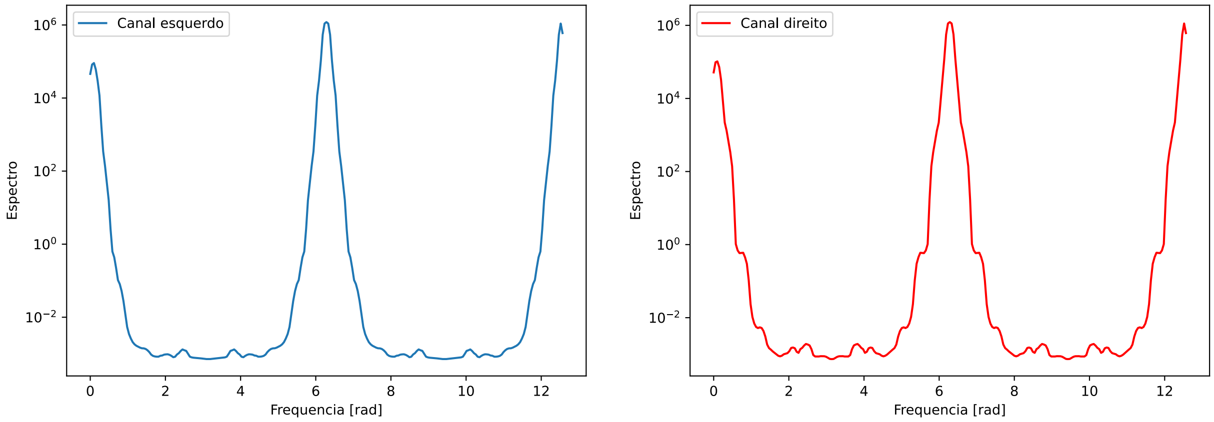
In [34]:

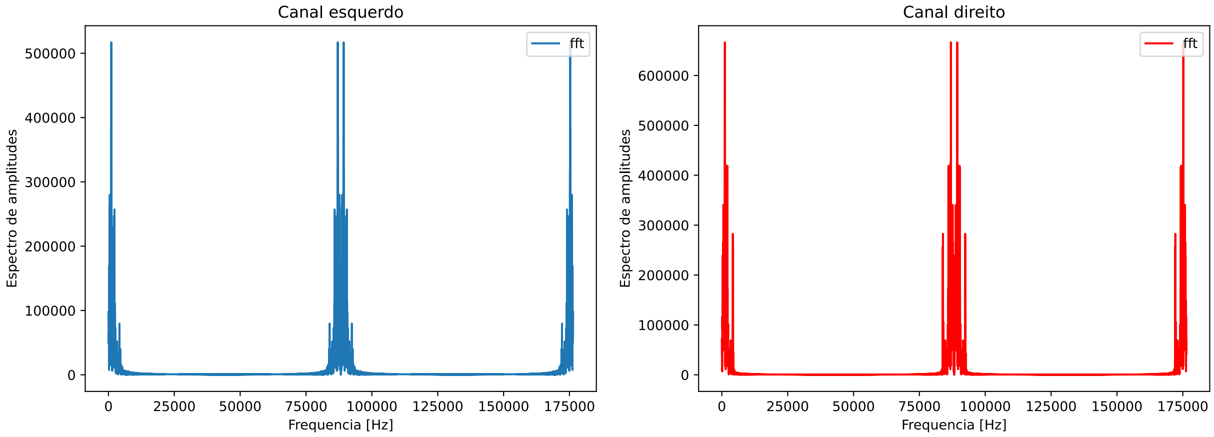
```
# Fator de interpolação
L = 4
interpolated_lc = np.zeros(L * len(left_channel))
interpolated_lc[::L] = left_channel

interpolated_rc = np.zeros(L * len(right_channel))
interpolated_rc[::L] = right_channel
```

In [35]:

```
plot_spect_welch_channels_stereo(
    lc  = interpolated_lc,
    rc  = interpolated_rc,
    fs = (2*np.pi)*L
)

plot_spect_fft_channels_stereo(
    lc  = interpolated_lc,
    rc  = interpolated_rc,
    sampling_rate=sampling_rate*L,
```

```
    nfft=4096
)
```



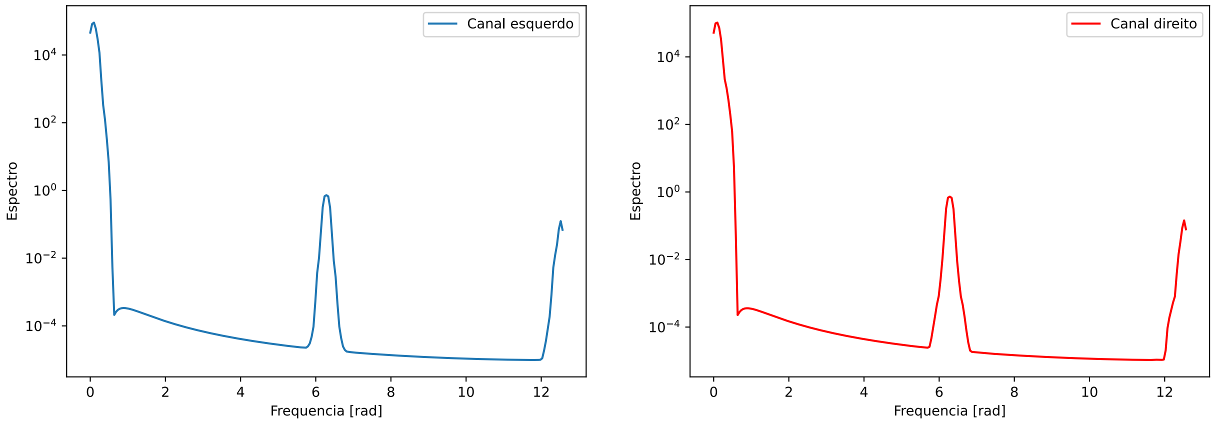


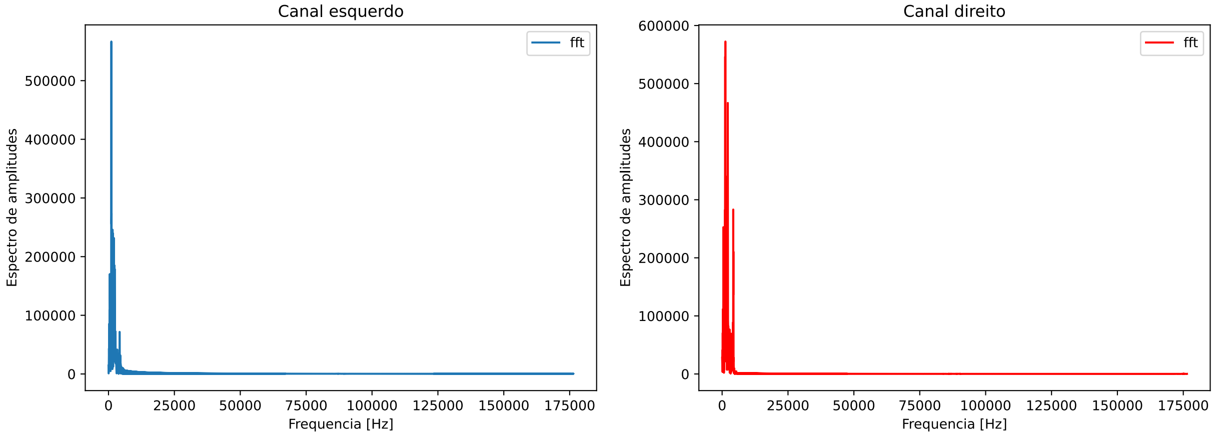
In [36]:

```
#Filtra os dados dos canais esquerdo e direito
interpolated_filtered_lc = lfilter(b, 1, interpolated_lc)
interpolated_filtered_rc = lfilter(b, 1, interpolated_rc)
```

In [37]:

```
plot_spect_welch_channels_stereo(
    lc  = interpolated_filtered_lc,
    rc  = interpolated_filtered_rc,
    fs = (2*np.pi)*L
)

plot_spect_fft_channels_stereo(
    lc  = interpolated_filtered_lc,
    rc  = interpolated_filtered_rc,
    sampling_rate=sampling_rate*L,
    nfft=4096
)
```





In [38]:
```
# Escrita de arquivo interpolado
audio = np.array([interpolated_lc, interpolated_rc]).T
scaled = np.int16(audio/np.max(np.abs(audio)) * 32767)
filename = 'signal_interpolated_' + str(L) + '.wav'
wavfile.write(filename, sampling_rate*L, scaled)
```

In [39]:
```
# Escrita de arquivo interpolado e filtrado
audio = np.array([interpolated_filtered_lc, interpolated_filtered_rc]).T
scaled = np.int16(audio/np.max(np.abs(audio)) * 32767)
filename = 'signal_interpolated_' + str(L) + '_filted.wav'
wavfile.write(filename, sampling_rate*L, scaled)
```