

```
# -*- coding: utf-8 -*-
"""
Sistemas de Adquisición y Procesamiento de Señales
Facultad de Ingeniería - UNER

Resolución del examen parcial 03/11/2025

Autor: Narella Corona
Fecha: Noviembre 2025
"""

#-----DATOS-----
"""

Señal del fotodetector: CAD de 14 bits (Vref: 3.3V) muestreando a 30 Hz
ACTIVIDAD 1: filtro antialiasing, respuesta máximamente plana en la banda de paso
ACTIVIDAD 2: analog y LTspice
ACTIVIDAD 3: separar las componentes relacionadas a la FC de las relacionadas a la FR
    Tipo: IIR
    Banda de paso: 1.5 Hz a 2.3 Hz
    Atenuación en la banda de paso: menor a 0.5 dB
    Atenuación para las componentes relacionadas a la FR: mayor a 30 dB

ACTIVIDAD 4: separar las componentes relacionadas a la FC
    Tipo: FIR
    Banda de paso: 0.05 Hz a 0.55 Hz
    Atenuación en la banda de paso: menor a 1 dB
    Atenuación para las componentes relacionadas a la FC: mayor a 30 dB
ACTIVIDAD 5: probar los filtros con las señales remuestreadas a 30Hz. Graficar.
ACTIVIDAD 6: determine la frecuencia respiratoria de cada señal

"""

"""

import numpy as np
import matplotlib.pyplot as plt
from funciones_fft import fft_mag
from import_ltspace import import_AC_LTSpice
from import_analogfilterwizard import import_AnalogFilterWizard
import filter_parameters
from scipy import signal

#-----
#ACTIVIDAD 1
#-----

archivo_1 = 'dataset/pleth_72lpm_500hz.txt' # nombre de archivo
archivo_2 = 'dataset/pleth_90lpm_500hz.txt' # nombre de archivo
archivo_3 = 'dataset/pleth_120lpm_500hz.txt' # nombre de archivo

senal_1 = np.loadtxt(archivo_1) # La amplitud de la señal se encuentra en mV
senal_2 = np.loadtxt(archivo_2) # La amplitud de la señal se encuentra en mV
senal_3 = np.loadtxt(archivo_3) # La amplitud de la señal se encuentra en mV

fs = 500 # frecuencia de muestreo 500Hz

ts = 1 / fs # tiempo de muestreo

N_1 = len(senal_1) # número de muestras en el archivo de
N_2 = len(senal_2) # número de muestras en el archivo de
N_3 = len(senal_3) # número de muestras en el archivo de

t_1 = np.linspace(0, N_1 * ts, N_1) # vector de tiempo -> la función la da x es
```

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js> La da x es

```

# Grafico las señales en el tiempo
fig1, ax1 = plt.subplots(3, 1, figsize=(20, 20))
fig1.suptitle("Senales en el tiempo", fontsize=20)
ax1[0].set_xlabel('Tiempo [s]', fontsize=15)
ax1[0].set_ylabel('Magnitud [mV]', fontsize=15)
ax1[0].grid(True, which="both")
ax1[0].plot(t_1, senial_1, color = 'blue')

ax1[1].set_xlabel('Tiempo [s]', fontsize=15)
ax1[1].set_ylabel('Magnitud [mV]', fontsize=15)
ax1[1].grid(True, which="both")
ax1[1].plot(t_2, senial_2, color = 'purple')

ax1[2].set_xlabel('Tiempo [s]', fontsize=15)
ax1[2].set_ylabel('Magnitud [mV]', fontsize=15)
ax1[2].grid(True, which="both")
ax1[2].plot(t_3, senial_3, color = 'magenta')
plt.show()

# Calculo las transformadas de las señales
frec_1, fft_senial_1 = fft_mag(senial_1, fs)
frec_2, fft_senial_2 = fft_mag(senial_2, fs)
frec_3, fft_senial_3 = fft_mag(senial_3, fs)

# Grafico las señales en frecuencia
fig2, ax2 = plt.subplots(3, 1, figsize=(20, 20))
fig2.suptitle("Senales en la frecuencia", fontsize=20)
ax2[0].set_xlabel('Frecuencia [Hz]', fontsize=15)
ax2[0].set_ylabel('Magnitud [mV]', fontsize=15)
ax2[0].grid(True, which="both")
ax2[0].plot(frec_1, fft_senial_1, color = 'blue')

ax2[1].set_xlabel('Frecuencia [Hz]', fontsize=15)
ax2[1].set_ylabel('Magnitud [mV]', fontsize=15)
ax2[1].grid(True, which="both")
ax2[1].plot(frec_2, fft_senial_2, color = 'purple')

ax2[2].set_xlabel('Frecuencia [Hz]', fontsize=15)
ax2[2].set_ylabel('Magnitud [mV]', fontsize=15)
ax2[2].grid(True, which="both")
ax2[2].plot(frec_3, fft_senial_3, color = 'magenta')

plt.show()

"""

La frecuencia de muestreo es 30Hz por lo que la redefino para encontrar los req
antialias, busco maximos en 15Hz (Frecuencia de Nyquist) y en valores mayores a
"""

# Determino los indices validos (los que corresponden a frecuencias mayores a FS
FS_resample = 30
indices_validos_1 = np.where(frec_1 >= FS_resample/2)[0]
indices_validos_2 = np.where(frec_2 >= FS_resample/2)[0]
indices_validos_3 = np.where(frec_3 >= FS_resample/2)[0]

# Recorto el vector de busqueda, las magnitudes y frecuencias correspondientes
magnitudes_recortadas_1 = fft_senial_1[indices_validos_1]
magnitudes_recortadas_2 = fft_senial_2[indices_validos_2]
magnitudes_recortadas_3 = fft_senial_3[indices_validos_3]

```

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js>
`frecuencias_recortadas_1 = frec_1[indices_validos_1]`

```

frecuencias_recortadas_2 = frec_2[indices_validos_2]
frecuencias_recortadas_3 = frec_3[indices_validos_3]

# Buscar el índice del máximo dentro de cada subconjunto
indice_max_1 = np.argmax(magnitudes_recortadas_1)
indice_max_2 = np.argmax(magnitudes_recortadas_2)
indice_max_3 = np.argmax(magnitudes_recortadas_3)

# Con el indice busco a que amplitud corresponde en cada senial
amp_max_1 = magnitudes_recortadas_1[indice_max_1]
amp_max_2 = magnitudes_recortadas_2[indice_max_2]
amp_max_3 = magnitudes_recortadas_3[indice_max_3]

frecuencia_max_1 = frecuencias_recortadas_1[indice_max_1]
frecuencia_max_2 = frecuencias_recortadas_2[indice_max_2]
frecuencia_max_3 = frecuencias_recortadas_3[indice_max_3]

fs2_1 = freq_1[np.where(freq_1>=(FS_resample/2))][0] #valor mas cercano a la fm
fs2_2 = freq_2[np.where(freq_2>=(FS_resample/2))][0] #valor mas cercano a la fm
fs2_3 = freq_3[np.where(freq_3>=(FS_resample/2))][0] #valor mas cercano a la fm

amp_fm_2_1 = np.max( fft_senial_1 [np.where(freq_1 == fs2_1) ] )
amp_fm_2_2 = np.max( fft_senial_2 [np.where(freq_2 == fs2_2) ] )
amp_fm_2_3 = np.max( fft_senial_3 [np.where(freq_3 == fs2_3) ] )

freq_fm_2_1 = freq_1[np.argmax(fft_senial_1[np.where(freq_1 == fs2_1) ] ) ] + f
freq_fm_2_2 = freq_2[np.argmax(fft_senial_2[np.where(freq_2 == fs2_2) ] ) ] + f
freq_fm_2_3 = freq_3[np.argmax(fft_senial_3[np.where(freq_3 == fs2_3) ] ) ] + f

print("ACTIVIDAD 1")
print("En la senial 1")
print(f"Máximo valor: {amp_max_1} a {frecuencia_max_1} Hz")
print(f"Valor en fs/2: {amp_fm_2_1} a {freq_fm_2_1} Hz \n")

print("En la senial 2")
print(f"Máximo valor: {amp_max_2} a {frecuencia_max_2} Hz")
print(f"Valor en fs/2: {amp_fm_2_2} a {freq_fm_2_2} Hz \n")

print("En la senial 3")
print(f"Máximo valor: {amp_max_3} a {frecuencia_max_3} Hz")
print(f"Valor en fs/2: {amp_fm_2_3} a {freq_fm_2_3} Hz \n")

freq_max = 0

```

Bien la búsqueda de requisitos.

```

#Busco el maximo de Las fs/2
max_fs2 = 0
if(max_fs2 < amp_fm_2_1):
    max_fs2 = amp_fm_2_1

if(max_fs2 < amp_fm_2_2):
    max_fs2 = amp_fm_2_2

if(max_fs2 < amp_fm_2_3):
    max_fs2 = amp_fm_2_3

#Ahora calculo la atenuacion que necesito cumplir con el filtro antialias
V_REF = 3300 # Tensión de referencia en mV
N_BITS = 14 # Resolución en bits

RES = V_REF/(2**N_BITS - 1) # Resolución en V

```

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js>
at_fs2_max = 20*np.log10(max_fs2/RES)

```

at_fs2_1 = 20*np.log10(amp_fm_2_1/RES)
at_fs2_2 = 20*np.log10(amp_fm_2_2/RES)
at_fs2_3 = 20*np.log10(amp_fm_2_3/RES)

at_1 = 20*np.log10(amp_max_1/RES)
at_2 = 20*np.log10(amp_max_2/RES)
at_3 = 20*np.log10(amp_max_3/RES)

at_max = 0

if(at_max < at_1):
    at_max = at_1
    frec_at_max = frecuencia_max_1

if(at_max < at_2):
    at_max = at_2
    frec_at_max = frecuencia_max_2

if(at_max < at_3):
    at_max = at_3
    frec_at_max = frecuencia_max_3

print(f"Resolucion del conversor {RES} \n")

print("Los requisitos para el filtro antialisa son: ")
print(f"Atenuación maxima en senial 1: {at_1:.2f}dB en {frecuencia_max_1:.3f}Hz")
print(f"Atenuación maxima en senial 2: {at_2:.2f}dB en {frecuencia_max_2:.3f}Hz")
print(f"Atenuación maxima en senial 3: {at_3:.2f}dB en {frecuencia_max_3:.3f}Hz

print(f"Atenuación fs/2 a {at_fs2_1:.2f}dB en {frec_fm_2_1}Hz")
print(f"Atenuación fs/2 a {at_fs2_2:.2f}dB en {frec_fm_2_2}Hz")
print(f"Atenuación fs/2 a {at_fs2_3:.2f}dB en {frec_fm_2_3}Hz\n")

print(f"Atenuación max en fs/2 a {at_fs2_max:.2f}dB en {frec_fm_2_3}Hz")
print(f"Atenuación max despues de fs/2 es {at_max:.2f}dB en {frecuencia_max_1}H

#-----
#ACTIVIDAD 2
#-----
```

Bien el filtro analógico.

```

freq_dis, mag_design = import_AnalogFilterWizard('DesignFiles/Data Files/Magnit
f_design, mag_sim, _ = import_AC_LTSpice('DesignFiles/SPICE Files/LTspice/ACAna

# Análisis de la atenuación del filtro simulado en las frecuencias de interés
F_AT1 = FS_resample/2
F_AT2 = freq_at_max
# se calcula la atenuación en el punto mas cercano a la frecuencia de interés
at1 = mag_sim[np.argmin(np.abs(f_design-F_AT1))]
at2 = mag_sim[np.argmin(np.abs(f_design-F_AT2))]

print("ACTIVIDAD 2")
print("La atenuación del filtro simulado en {}Hz es de {:.2f}dB".format(F_AT1,
print("La atenuación del filtro simulado en {}Hz es de {:.2f}dB".format(F_AT2,
print("\r")

# Importar datos de herramientas de diseño y simulación.
f_design, mag_design = import_AnalogFilterWizard('DesignFiles/Data Files/Magnit
f_sim, mag_sim, _ = import_AC_LTSpice('DesignFiles/SPICE Files/LTspice/ACAnalys
```

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js>

```
fig_filtro, ax_filtro = plt.subplots(1, 1, figsize=(12, 10))
```

```

ax_filtro.set_title('Filtro Antialiasing - Análisis de Requisitos', fontsize=18)
ax_filtro.set_xlabel('Frecuencia [Hz]', fontsize=15)
ax_filtro.set_ylabel('|H(jw)|² [dB]', fontsize=15)
ax_filtro.grid(True, which="both")

# Graficar la respuesta ideal y la simulada.
ax_filtro.plot(f_design, mag_design, label='Filtro Diseñado', linewidth=2)
ax_filtro.plot(f_sim, mag_sim, label='Filtro Simulado', linewidth=2)

# Marcar los puntos de requisito mínimo de atenuación.
ax_filtro.plot(F_AT1, -at_fs2_max, marker='X', markersize=15, color='red',
                label=f'Requisito en fs/2 a {at_fs2_max:.2f}dB en {frec_fm_2_3}Hz')

ax_filtro.plot(F_AT2, -at_max, marker='X', markersize=15, color='orange',
                label=f'Requisito después de fs/2 es {at_max:.2f}dB en {frecuenci

ax_filtro.legend(loc="lower left", fontsize=12)
ax_filtro.set_xlim(1, 100)
ax_filtro.set_ylim(-100, 10)
ax_filtro.set_xscale('log')

plt.show()

#-----
#ACTIVIDAD 5
#-----

print('ACTIVIDAD 5')
archivo_1_rem = 'dataset/pleth_72lpm_30hz.txt' # nombre de archivo
archivo_2_rem = 'dataset/pleth_90lpm_30hz.txt' # nombre de archivo
archivo_3_rem = 'dataset/pleth_120lpm_30hz.txt' # nombre de archivo

senial_1_rem = np.loadtxt(archivo_1_rem) # La amplitud de la señal se encuentra
senial_2_rem = np.loadtxt(archivo_2_rem) # La amplitud de la señal se encuentra
senial_3_rem = np.loadtxt(archivo_3_rem) # La amplitud de la señal se encuentra

ts_rem = 1 / FS_resample # tiempo de muestreo
N_1_rem = len(senial_1_rem) # número de muestras
t_1_rem = np.linspace(0, N_1_rem * ts_rem, N_1_rem) # vector de tiempo -> la fu

N_2_rem = len(senial_2_rem) # número de muestras
t_2_rem = np.linspace(0, N_2_rem * ts_rem, N_2_rem) # vector de tiempo -> la fu

N_3_rem = len(senial_3_rem) # número de muestras
t_3_rem = np.linspace(0, N_3_rem * ts_rem, N_3_rem) # vector de tiempo -> la fu

#Primero aplico el filtro IIR a las 3 señales y después el filtro FIR

filtro_iir = np.load('Filtro_IIR_Chebyshev.npz', allow_pickle=True)
print("Filtro IIR: ")
filter_parameters.filter_parameters('Filtro_IIR_Chebyshev.npz')
print("\r\n")

# Se extraen los coeficientes de numerador y denominador
Num_iir, Den_iir = filtro_iir['ba']

senial_iir_1 = signal.lfilter(Num_iir, Den_iir, senial_1_rem)
senial_iir_2 = signal.lfilter(Num_iir, Den_iir, senial_2_rem)
senial_iir_3 = signal.lfilter(Num_iir, Den_iir, senial_3_rem)

```

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js>

```
print("Filtro FIR: ")
```

```

filter_parameters.filter_parameters('Filtro_FIR_Equiripple.npz')
print("\r\n")

# Se extraen Los coeficientes de numerador y denominador
Num_fir, Den_fir = filtro_fir['ba']

ceros_agregar = filtro_fir['N']//2

print(f"La mitad del Orden del filtro FIR {ceros_agregar} \n")

senial_fir_1 = senial_1_rem
senial_fir_2 = senial_2_rem
senial_fir_3 = senial_3_rem

senial_fir_1 = np.pad(senial_1_rem,(0,ceros_agregar),mode='constant')
senial_fir_2 = np.pad(senial_2_rem,(0,ceros_agregar),mode='constant')
senial_fir_3 = np.pad(senial_3_rem,(0,ceros_agregar),mode='constant')

senial_fir_1 = signal.lfilter(Num_fir, Den_fir, senial_fir_1)
senial_fir_2 = signal.lfilter(Num_fir, Den_fir, senial_fir_2)
senial_fir_3 = signal.lfilter(Num_fir, Den_fir, senial_fir_3)

senial_fir_1 = senial_fir_1[ceros_agregar:]
senial_fir_2 = senial_fir_2[ceros_agregar:]
senial_fir_3 = senial_fir_3[ceros_agregar:]

fig4, ax4 = plt.subplots(3, 1, figsize=(22, 15), sharex=True)
fig4.suptitle("Senial 1 remuestreada y las filtradas", fontsize=18)
ax4[0].plot(t_1_rem, senial_1_rem, label='Señal Original', color='blue')
ax4[0].legend(loc="upper right", fontsize=15)
ax4[0].set_ylabel('Magnitud [mV]', fontsize=15)
ax4[0].set_xlabel('Tiempo [s]', fontsize=15)
ax4[0].grid()
ax4[1].plot(t_1_rem, senial_iir_1, label='Señal filtrada iir', color='blue')
ax4[1].legend(loc="upper right", fontsize=15)
ax4[1].set_ylabel('Magnitud [mV]', fontsize=15)
ax4[1].set_xlabel('Tiempo [s]', fontsize=15)
ax4[1].grid()
ax4[2].plot(t_1_rem, senial_fir_1, label='Señal filtrada fir', color='blue')
ax4[2].legend(loc="upper right", fontsize=15)
ax4[2].set_ylabel('Magnitud [mV]', fontsize=15)
ax4[2].set_xlabel('Tiempo [s]', fontsize=15)
ax4[2].grid()

plt.show()

fig5, ax5 = plt.subplots(3, 1, figsize=(22, 15), sharex=True)
fig5.suptitle("Senial 2 remuestreada y las filtradas", fontsize=18)
ax5[0].plot(t_2_rem, senial_2_rem, label='Señal Original', color='purple')
ax5[0].legend(loc="upper right", fontsize=15)
ax5[0].set_ylabel('Magnitud [mV]', fontsize=15)
ax5[0].set_xlabel('Tiempo [s]', fontsize=15)
ax5[0].grid()
ax5[1].plot(t_2_rem, senial_iir_2, label='Señal filtrada iir', color='purple')
ax5[1].legend(loc="upper right", fontsize=15)
ax5[1].set_ylabel('Magnitud [mV]', fontsize=15)
ax5[1].set_xlabel('Tiempo [s]', fontsize=15)
ax5[1].grid()
ax5[2].plot(t_2_rem, senial_fir_2, label='Señal filtrada fir', color='purple')
ax5[2].legend(loc="upper right", fontsize=15)
ax5[2].set_ylabel('Magnitud [mV]', fontsize=15)
ax5[2].set_xlabel('Tiempo [s]', fontsize=15)

```

Bien el IIR. El FIR debería ser pasa-banda, habría que haber analizado las FFts para los requisitos:

"Atenuación para las componentes relacionadas a la FR: mayor a 30 dB" -> sería la componente de 0.33Hz

"Atenuación para las componentes relacionadas a la FC: mayor a 30 dB"->sería la componente de 0.83Hz

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js>

```

plt.show()

fig6, ax6 = plt.subplots(3, 1, figsize=(22, 15), sharex=True)
fig6.suptitle("Senial 3 remuestreada y las filtradas", fontsize=18)
ax6[0].plot(t_3_rem, senial_3_rem, label='Señal Original', color='magenta')
ax6[0].legend(loc="upper right", fontsize=15)
ax6[0].set_ylabel('Magnitud [mV]', fontsize=15)
ax6[0].set_xlabel('Tiempo [s]', fontsize=15)
ax6[0].grid()
ax6[1].plot(t_3_rem, senial_iir_3, label='Señal filtrada iir', color='magenta')
ax6[1].legend(loc="upper right", fontsize=15)
ax6[1].set_ylabel('Magnitud [mV]', fontsize=15)
ax6[1].set_xlabel('Tiempo [s]', fontsize=15)
ax6[1].grid()
ax6[2].plot(t_3_rem, senial_fir_3, label='Señal filtrada fir', color='magenta')
ax6[2].legend(loc="upper right", fontsize=15)
ax6[2].set_ylabel('Magnitud [mV]', fontsize=15)
ax6[2].set_xlabel('Tiempo [s]', fontsize=15)
ax6[2].grid()

plt.show()

#-----
#ACTIVIDAD 6
#-----
print('ACTIVIDAD 6')

"""
Para encontrar La frecuencia respiratoria, calculo el espectro de las
señales filtradas en la actividad 5 y busco la frecuencia del pico de máxima en
"""

```

Bien la obtención de FR

```

# Señal 1
frec_fr_1, mag_fr_1 = fft_mag(senial_fir_1, FS_resample)
# Se busca el índice del máximo en la magnitud
idx_pico_1 = np.argmax(mag_fr_1[1:]) + 1
frec_respiratoria_1 = frec_fr_1[idx_pico_1]
print(f"Frecuencia Respiratoria (Señal 1): {frec_respiratoria_1:.3f} Hz")

# Señal 2
frec_fr_2, mag_fr_2 = fft_mag(senial_fir_2, FS_resample)
idx_pico_2 = np.argmax(mag_fr_2[1:]) + 1
frec_respiratoria_2 = frec_fr_2[idx_pico_2]
print(f"Frecuencia Respiratoria (Señal 2): {frec_respiratoria_2:.3f} Hz")

# Señal 3
frec_fr_3, mag_fr_3 = fft_mag(senial_fir_3, FS_resample)
idx_pico_3 = np.argmax(mag_fr_3[1:]) + 1
frec_respiratoria_3 = frec_fr_3[idx_pico_3]
print(f"Frecuencia Respiratoria (Señal 3): {frec_respiratoria_3:.3f} Hz")

```

ACTIVIDAD 1

En la senial 1

Máximo valor: 20.559573176154675 a 50.0 Hz

Valor en fs/2: 2.6734191750275778 a 15.0 Hz

En la senial 2

Máximo valor: 30.594319769037273 a 50.0 Hz

Valor en fs/2: 2.6843364799963676 a 15.0 Hz

En la senial 3

Máximo valor: 25.505270151056102 a 50.0 Hz

File failed to load: https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/TeX/config.js

Resolucion del conversor 0.20142830983336385

Los requisitos para el filtro antialisa son:

Atenuación maxima en senial 1: 40.18dB en 50.000Hz

Atenuación maxima en senial 2: 43.63dB en 50.000Hz

Atenuación maxima en senial 3: 42.08dB en 50.000Hz

Atenuación fs/2 a 22.46dB en 15.0Hz

Atenuación fs/2 a 22.49dB en 15.0Hz

Atenuación fs/2 a 21.67dB en 15.0Hz

Atenuación max en fs/2 a 22.49dB en 15.0Hz

Atenuación max despues de fs/2 es 43.63dB en 50.0Hz

ACTIVIDAD 2

La atenuación del filtro simulado en 15.0Hz es de -32.44dB

La atenuación del filtro simulado en 50.0Hz es de -53.59dB

ACTIVIDAD 5

Filtro IIR:

Frecuencia de muestreo: 30.0 Hz

Aproximación utilizada:pyfda.filter_widgets.cheby1

Orden del filtro: 10

Filtro Pasa-Banda

Frecuencia de corte banda de rechazo inferior: 1.20 Hz

Frecuencia de corte inferior banda de paso : 1.50 Hz

Frecuencia de corte superior banda de paso : 2.30 Hz

Frecuencia de corte banda de rechazo superior: 15.00 Hz

Filtro FIR:

Frecuencia de muestreo: 30.0 Hz

Aproximación utilizada:pyfda.filter_widgets.equiripple

Orden del filtro: 107

Filtro Pasa-Bajos

Frecuencia de corte banda de paso : 0.05 Hz

Frecuencia de corte banda de rechazo: 0.55 Hz

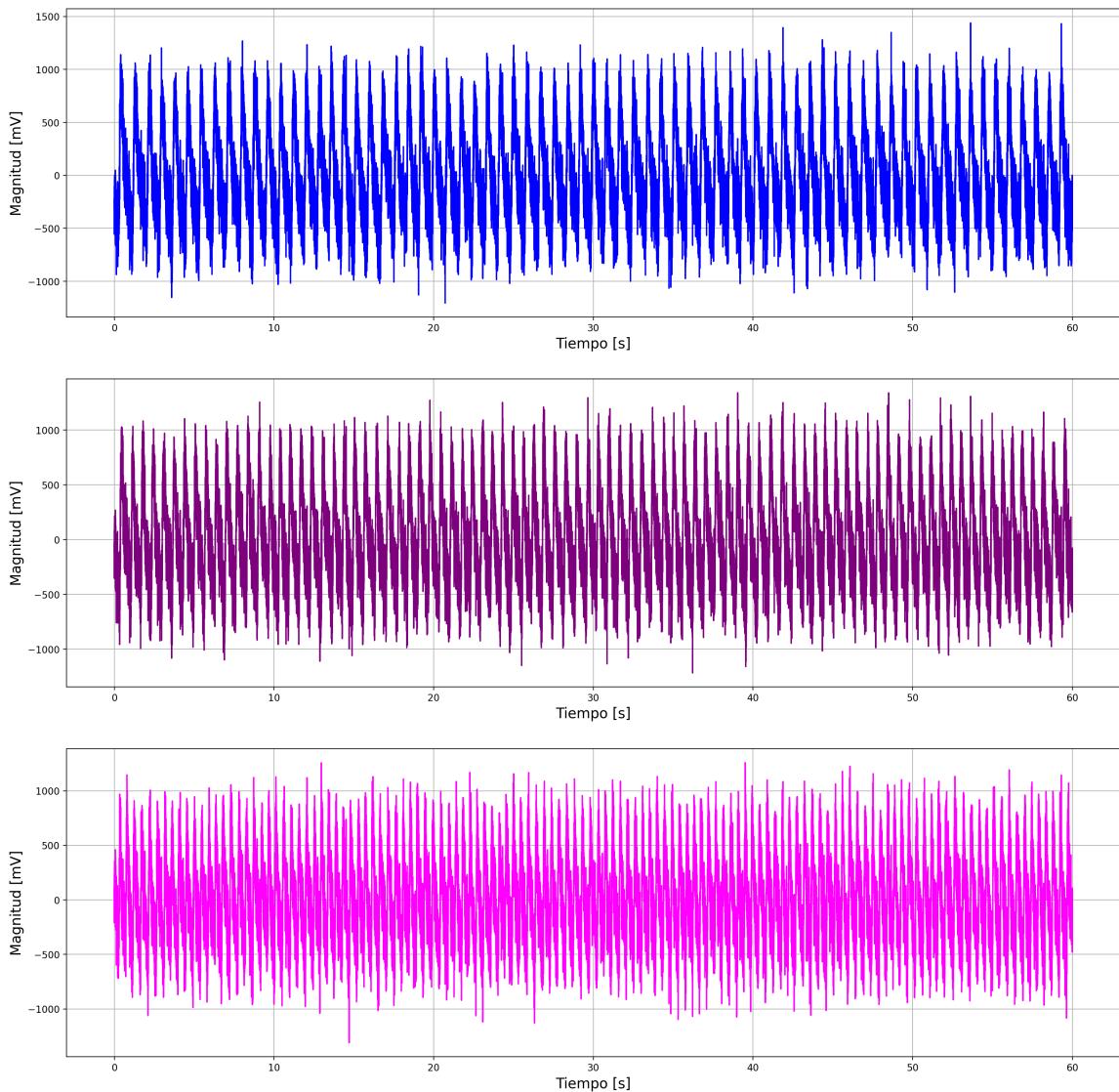
La mitad del Orden del filtro FIR 53

ACTIVIDAD 6

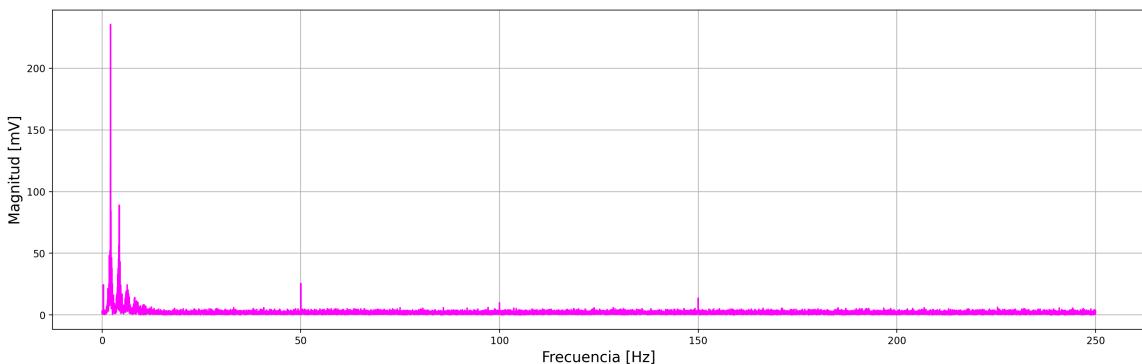
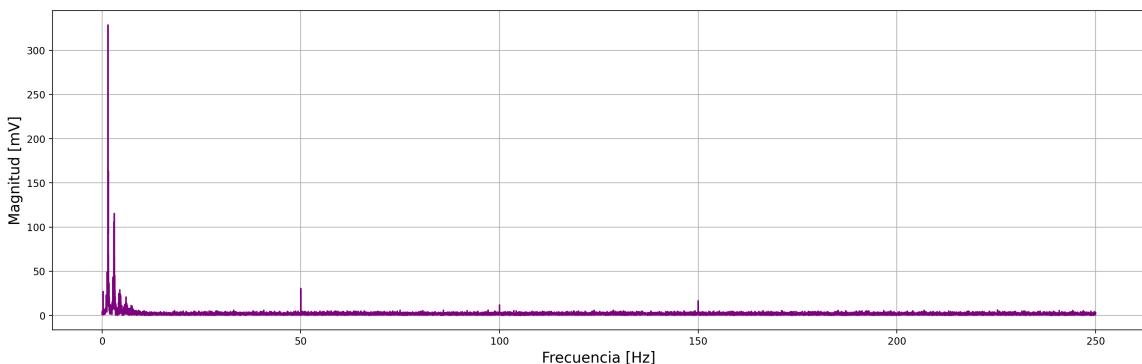
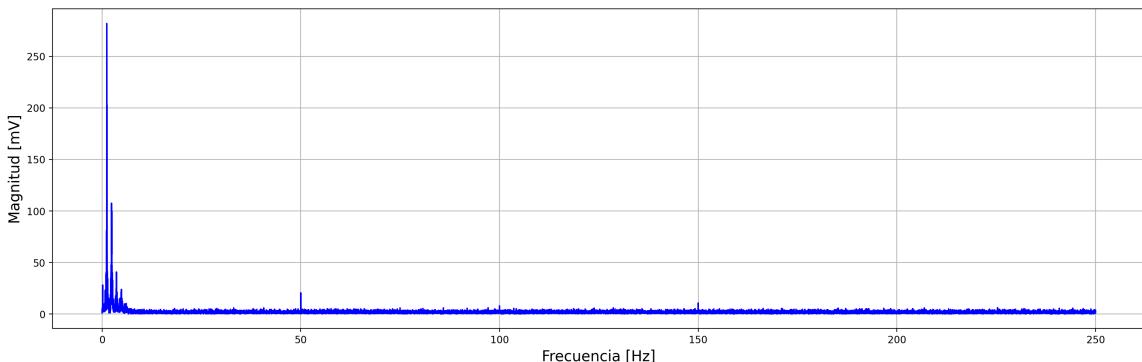
Frecuencia Respiratoria (Señal 1): 0.167 Hz

Frecuencia Respiratoria (Señal 2): 0.267 Hz

Frecuencia Respiratoria (Señal 3): 0.333 Hz

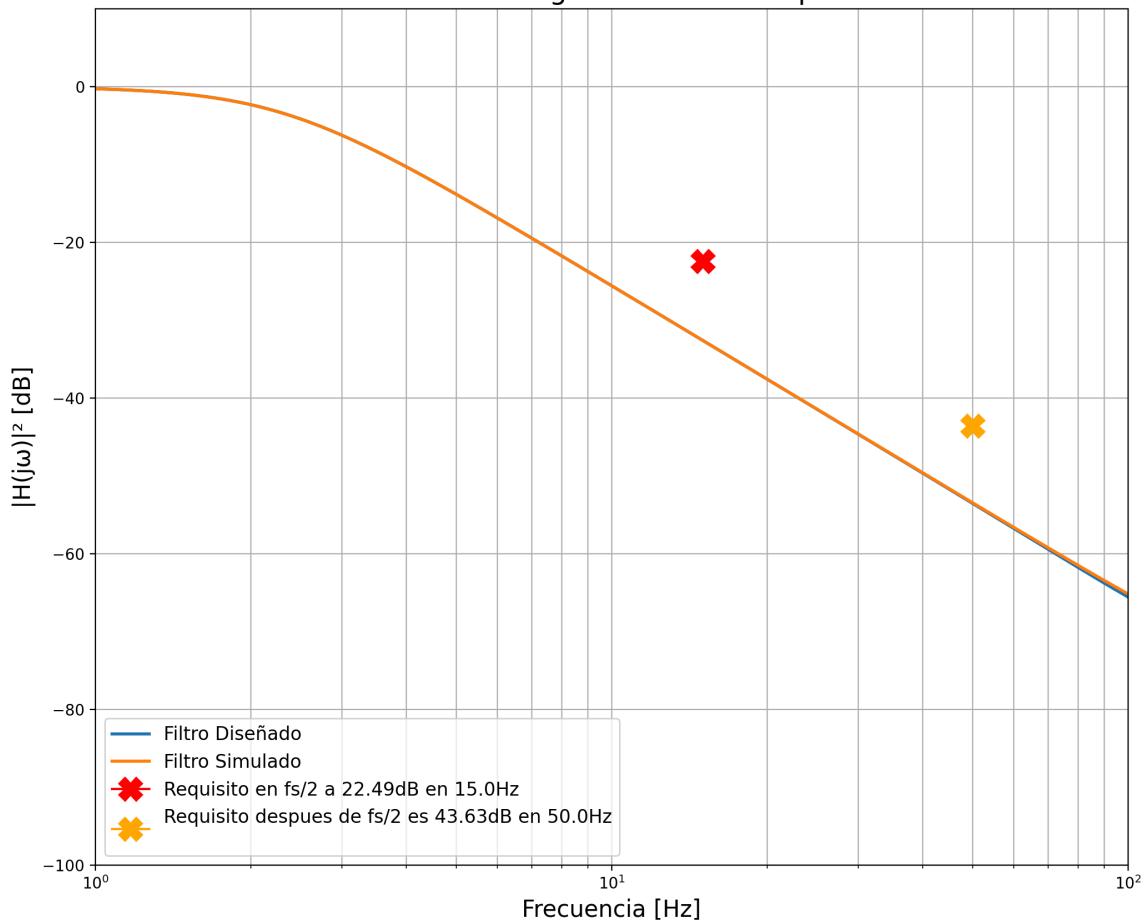


File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/LaTeX/config.js>

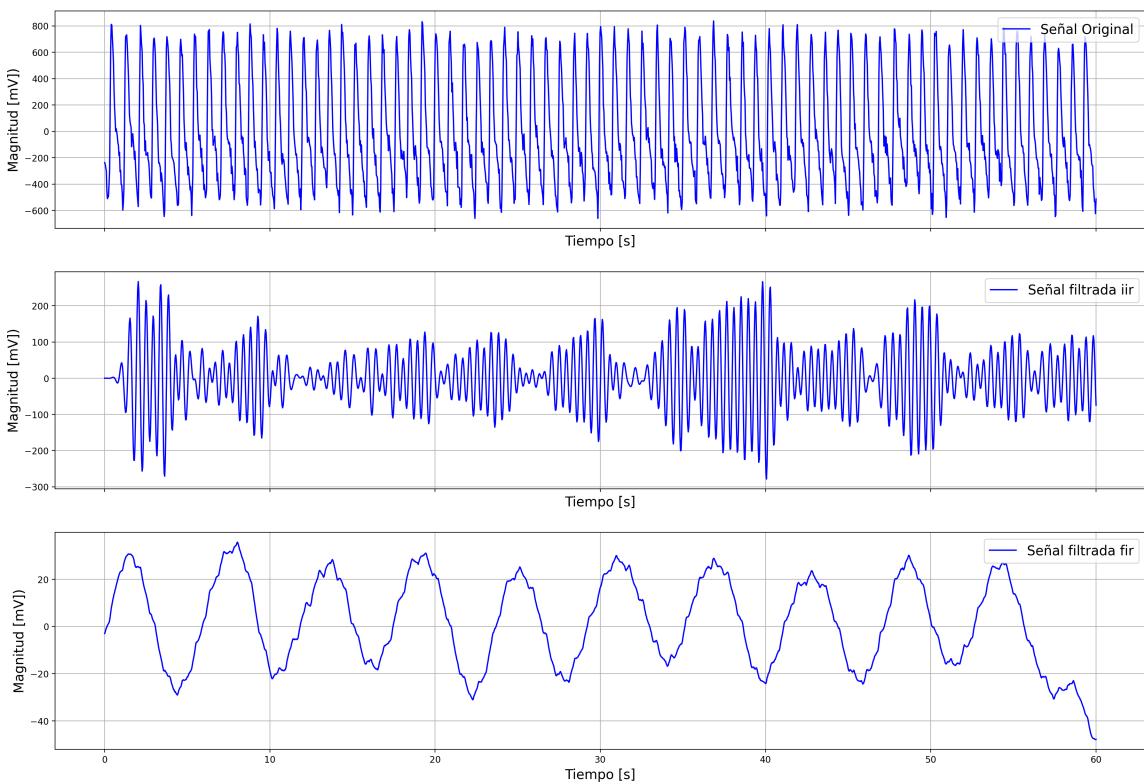


File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/LaTeX/config.js>

Filtro Antialiasing - Análisis de Requisitos

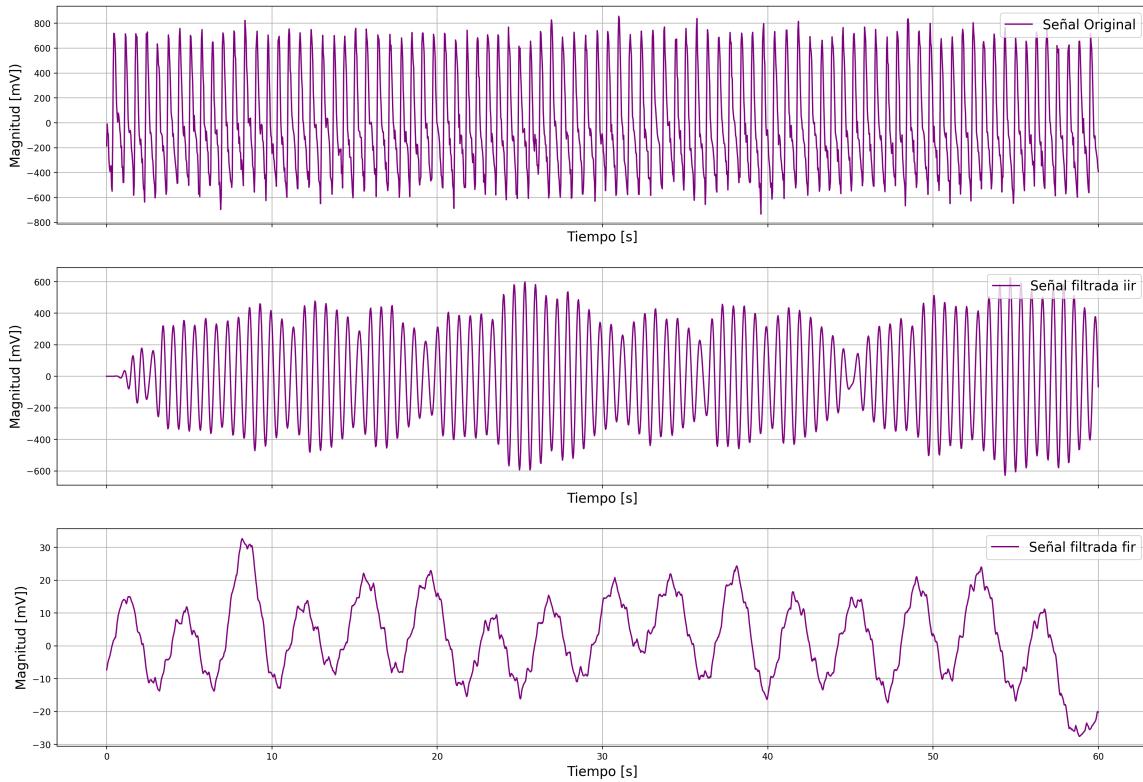


Señal 1 remuestreada y las filtradas

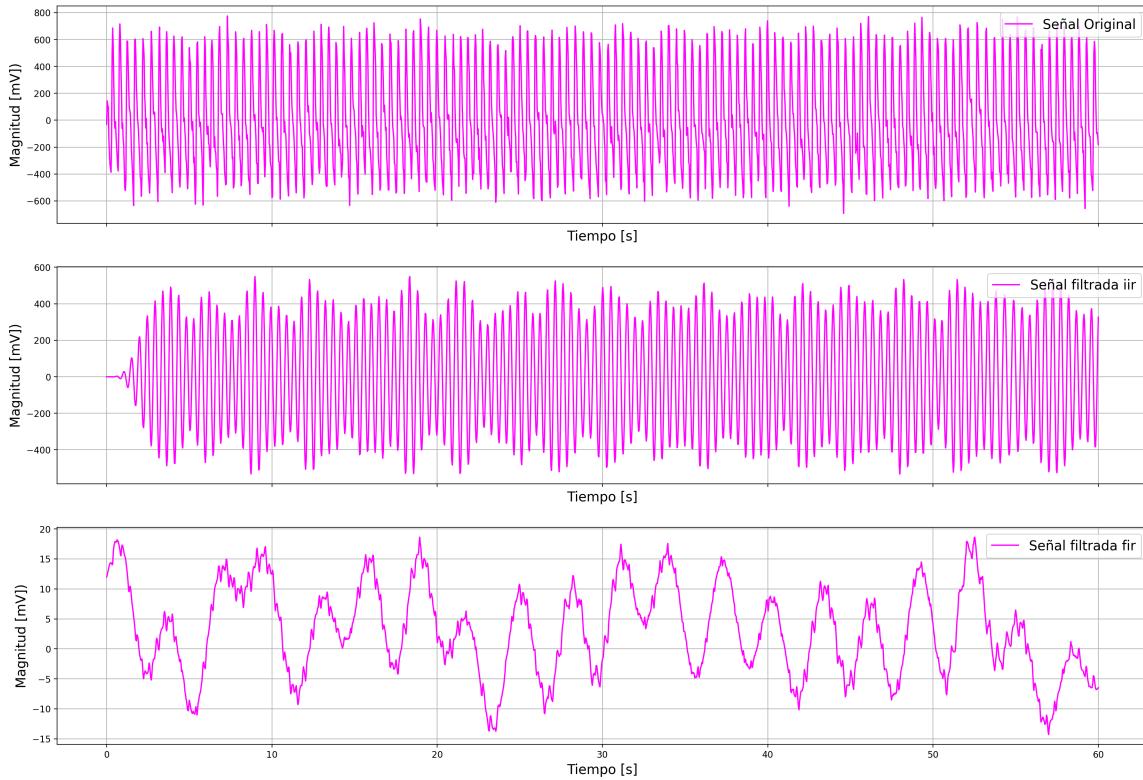


File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/LaTeX/config.js>

Señal 2 remuestreada y las filtradas



Señal 3 remuestreada y las filtradas



Published from [Examen_Corona.py](#) using [Pweave](#) 0.30.3 on 03-11-2025.

File failed to load: <https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/jax/input/LaTeX/config.js>