UNIVERSIDAD NACIONAL DE TUCUMAN

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGIA

PROCESAMIENTO DIGITAL DE SEÑALES

Proyecto Integrador

Grupo: ANCE, GASTON ARMANDO

ARNEDO, EMILIANO

DIAZ, LEONARDO LEANDRO

PEREYRA, FAUSTO HORACIO

Carrera: ING. ELECTRONICA

2021

TEMA: Filtro digital de fase cero

APLICACIÓN: Procesamiento de señal de Electrocardiograma de tres vías

INDICE DE FIGURAS

INDICE

INTRODUCCION

La finalidad del presente trabajo integrador es poner en práctica los conocimientos adquiridos durante el cursado de la asignatura **PROCESAMIENTO DIGITAL DE SEÑALES** para resolver un caso que se presenta en la labor diaria del ingeniero electrónico.

RESUMEN

Con el transcurso de los años, la tecnología ha experimentado un progresivo e indetenible avance. Diariamente se diseñan y desarrollan nuevos productos y servicios orientados al mejoramiento de la calidad de vida del ser humano entre los que se encuentran las soluciones de ingeniería al servicio del cuidado de la salud [18].

Las señales eléctricas producidas por el corazón, vistas a través del empleo de un electrocardiograma (ECG), permiten conocer el comportamiento de dicho órgano. Estas señales indican toda la actividad realizada por el músculo cardiaco y a través de ellas es posible identificar alguna anomalía en el corazón [18].

El electrocardiograma es un registro gráfico de los potenciales eléctricos generados en el corazón durante el ciclo cardíaco. Esta representación consiste en una línea base y varias deflexiones y ondas. La figura [1] ilustra las componentes presentes en un pulso teórico de un electrocardiograma.

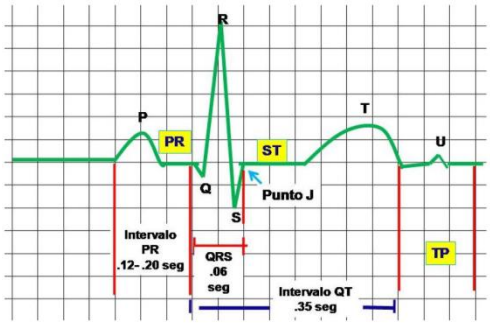


Figura (1) Componentes de un electrocardiograma.

Para obtener las señales cardíacas se colocan electrodos en posiciones específicas del cuerpo, los cuales transmiten las señales eléctricas que se propagan a través de los tejidos corporales desde el corazón, en un ECG de control se fijan tres electrodos.

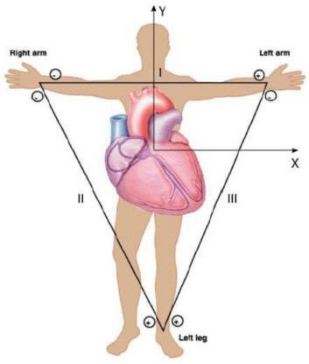


Figura (2) Ubicación de electrodos en el cuerpo humano.

Los potenciales eléctricos son relativamente pequeños, se utiliza un amplificador diferencial como el de la figura (3) para incrementar el nivel de la señal que se obtiene con los electrodos.

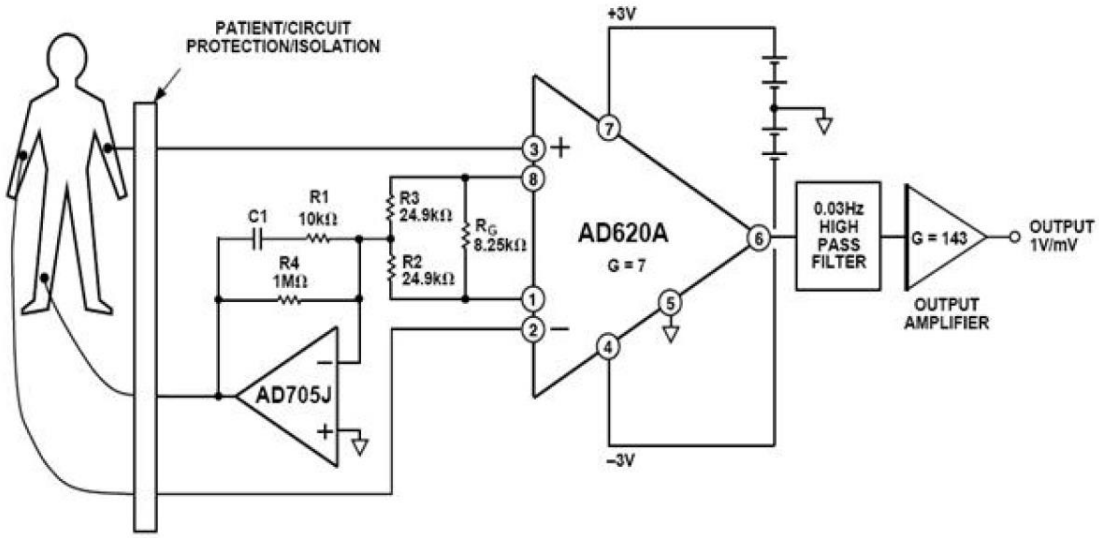


Figura (3) Amplificador diferencial implementado para señales de ECG.

Existen principalmente cuatro tipos de problemas encontrados en las señales de ECG: deriva de la línea de base, interferencia de la línea eléctrica, ruido EMG (causados por actividad muscular) y los causados por movimiento de electrodos.

El efecto de la deriva de línea de base consiste en el movimiento del eje x de la señal hacia arriba y hacia abajo, el contenido de frecuencia de este movimiento está por debajo de los 0.5 [Hz], en el sistema propuesto, para eliminarlo se pasa la señal por un filtro pasa-altos analógico como se ilustra en la figura (4).

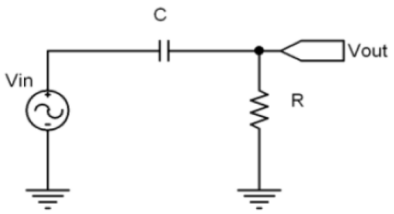


Figura (4) Filtro pasa-altos pasivo.

La interferencia de la línea eléctrica es una señal sinusoidal de 50 [Hz], en el sistema propuesto se elimina con un filtro digital ranura (digital notch filter) cuya respuesta en amplitud puede visualizarse en la figura (5).

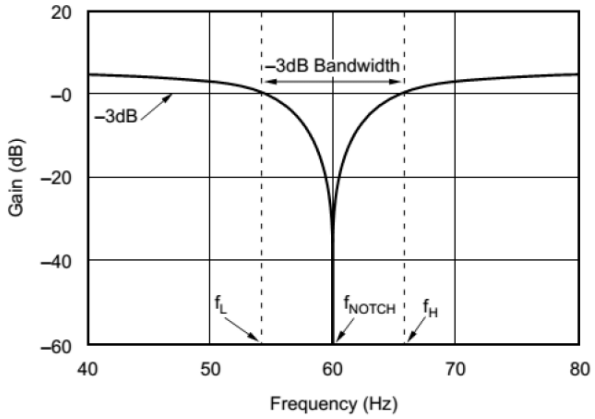


Figura (5) Respuesta en amplitud de un filtro notch.

El contenido de frecuencia del ruido muscular se superpone considerablemente al del complejo PQRST, se elimina con técnicas de procesamiento digital de señales que superan el contenido de este curso.

Los problemas causados por el movimiento del electrodo se asemejan a las características de la deriva de la línea de base, pero su contenido espectral que es en el rango de 1 a 10 [Hz] se superpone considerablemente al del complejo PQRST.

CASO DE ESTUDIO

Para realizar un correcto análisis de las señales cardiacas es necesario proponer un modelo de procesamiento digital de señales cardiacas que permita obtener una representación confiable de la señal eléctrica del corazón con el mínimo ruido posible, para ello nos valdremos de la herramienta de cómputo numérico Matlab [18].

Diseñar un filtro de fase cero, componente de procesamiento digital de señales del sistema propuesto que implementa el filtro pasa bajos y el filtro ranura. En la figura [6] se puede apreciar el bloque correspondiente a desarrollar dentro del subsistema de procesamiento digital de señal.

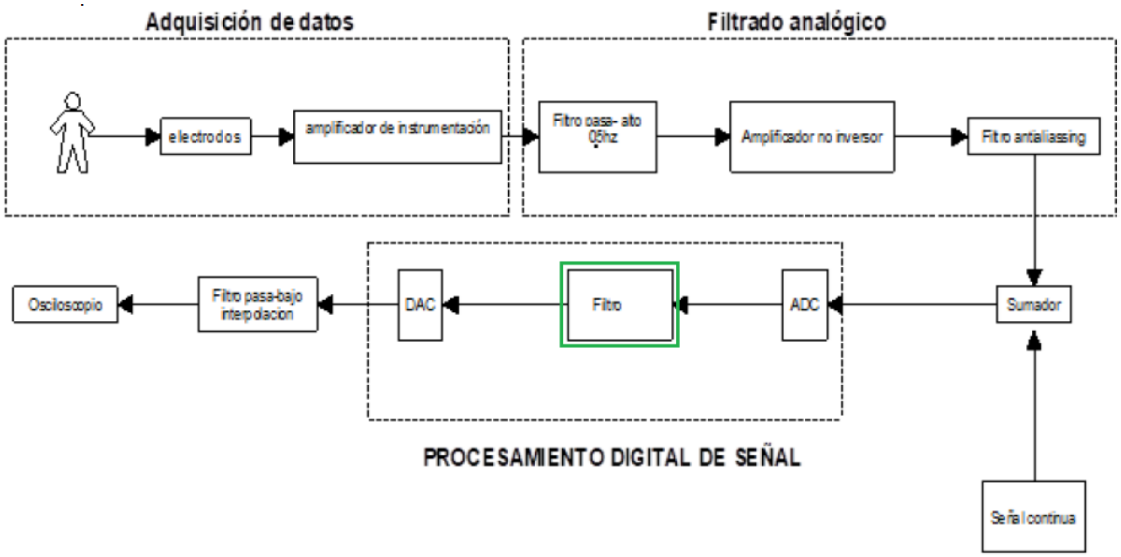


Figura (6) Diagrama de bloques de un electrocardiógrafo.

Dado que el diagnóstico que se realiza utilizando la señal de ECG se basa en la forma de la señal, es importante que la respuesta de los filtros sea de fase lineal, de modo que se implementa un filtro de fase cero.

- La señal de ECG contiene valores bajos de frecuencia, se considerará de interés las componentes de frecuencia por debajo de 120 [Hz].
- La frecuencia de muestreo utilizada en el sistema es de 1.000 [Hz] (modificado a 800 [Hz]).
- El sistema se implementa utilizando un micro DSP de 16 bits.
- Se desea que el ancho de banda del filtro ranura sea lo más chica posible.
- La cátedra pasará dos señales de prueba.

OBJETIVOS

1) Justificar por qué el filtro debe ser de fase lineal para no modificar la forma de la señal.
2) Diseñar el filtro ranura y el filtro pasa bajos para la implementación del filtro de fase cero.
3) Desarrollar una función en Matlab que implemente el filtrado de fase cero, debe procesar la señal muestra a muestra, es decir se llama la función pasándole el valor actual de la señal y devuelve el valor actual de la salida, actualizándose las variables o registros internos del sistema.
4) Procesar las dos señales de entrada entregada por la cátedra.
5) Analizar las componentes de frecuencia de las señales de entrada utilizando la Transformada Rápida de Fourier.
6) Analizar las componentes de frecuencia de las señales de salida utilizando la Transformada Rápida de Fourier.

Para cumplir satisfactoriamente con los objetivos planteados en este proyecto integrador, como equipo llevamos a cabo un estudio exhaustivo del concepto, diseño, implementación y simulación en software de cómputo numérico de un filtro de fase cero.

Se planificó detalladamente la metodología de trabajo, la carga horaria semanal que se le dedicaría al proyecto y las tareas individuales que debía cumplir cada integrante del equipo como así también, las tareas colectivas.

Finalmente, con los recursos disponibles y el abanico de referencias bibliográficas y documentación, se optó por el diseño de un filtro de fase cero basado en las ideas planteadas en [1], [4], [13] y [14].

DESARROLLO

1) Justificar por qué el filtro debe ser de fase lineal para no modificar la forma de la señal.

Los filtros digitales con fase lineal tienen la ventaja de retrasar todos los componentes de frecuencia en la misma cantidad, es decir, conservan las relaciones de fase de la señal de entrada [21]. Esta preservación de la fase significa que la señal filtrada conserva la forma de la señal de entrada original. Esta característica es esencial para las aplicaciones de audio, ya que la forma de la señal es primordial para mantener una alta fidelidad en el audio filtrado. Otra área de aplicación que requiere esto es el análisis de formas de ondas biomédicas de ECG, ya que cualquier artefacto introducido por el filtro puede malinterpretarse como anomalías cardíacas.

Para lograr retrasos temporales iguales para todas las frecuencias, necesitamos que cada frecuencia tenga un cambio de fase diferente, es decir, un cambio de fase que resulte en el mismo retraso para cada frecuencia [21]. Más específicamente, necesitamos una respuesta de cambio de fase que aumente linealmente con la frecuencia; esto tiene sentido, porque a medida que aumenta la frecuencia, un cambio de fase fijo corresponde a un período de tiempo que disminuye gradualmente y, por lo tanto, necesitamos más cambio de fase para compensar.

Entonces, un filtro de fase lineal ideal exhibe un cambio de fase que aumenta linealmente con la frecuencia y, por lo tanto, proporciona un retraso temporal constante (esto se aplica principalmente a las frecuencias dentro de la banda de paso, es decir, las frecuencias de interés). El retardo de grupo es proporcional a la derivada de la respuesta de fase con respecto a la frecuencia; la derivada de una función lineal es una constante, lo que explica por qué una respuesta de fase lineal también se conoce como retardo de grupo constante. Un tipo de filtro bien conocido que está optimizada por la fase lineal es el filtro de Bessel.

Lo expresado anteriormente se justifica matemáticamente utilizando transformada de Fourier [2], [3], [17], la que tiene la propiedad:

Desplazamiento en el tiempo

$$Sea: x_1(t) \overset{TF}{\leftrightarrow} X_1(\omega)$$

$$Entonces: x_2(t) = x_1(t - t_0) \overset{TF}{\leftrightarrow} X_2(\omega) = e^{j\omega t_0} \cdot X_1(\omega)$$

Donde el término $e^{j\omega t_0}$ tiene módulo igual a 1 e introduce un incremento lineal en fase, el incremento lineal es en función de la frecuencia.

2) Diseñar el filtro ranura y el filtro pasa bajos para la implementación del filtro de fase cero.

Se propone como solución implementar filtros recursivos por recomendaciones de los docentes de cátedra [22] y [23] y según exponen los autores en su artículo [1], es decir mediante la ecuación en recurrencia. En este caso, cada filtro se define por los coeficientes de recursión. La salida en cada instante involucra además de muestras de la entrada, muestras previas de la salida como se explica en [21].

Este tipo de filtro tiene una respuesta al impulso infinita siendo su ecuación de diferencia de la forma:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \cdots + b_Mx[n-M] - a_1y[n-1] - a_2y[n-2] - \cdots - a_Ny[n-N]$$

Las constantes $b_i, i = 1, \dots, M$ y $a_i, j = 1, \dots, N$ se llaman coeficientes del filtro. El filtro queda completamente especificado con los valores de todos los coeficientes.

Los valores b_i se llaman coeficientes de prealimentación (feedforward) y los valores a_i se llaman coeficientes de realimentación (feedback).

Observaciones:

- El filtro es recursivo si tiene algún coeficiente de realimentación no nulo. En ese caso, es un filtro IIR. En caso contrario, no hay realimentación y el filtro es FIR, o equivalentemente, no recursivo.
- El retardo máximo usado por la ecuación en recurrencia se llama orden del filtro. El orden es el máximo entre N y M.
- Se requiere que el filtro conjunto sea de fase cero y recursivos por la distorsión que provoca [22], [23].

Diseño de Filtro Ranura

Para llevar a cabo el diseño del filtro ranura o notch desarrollado en [12] se deben tener en cuenta las siguientes especificaciones:

- Frecuencia de muestreo $f_s = 800\ [Hz]$
- Frecuencia central del filtro ranura $f_c = 50\ [Hz]$ correspondiente a la frecuencia de la red eléctrica.
- Ancho de banda $AB = 12\ [Hz]$ correspondiente a la separación entre los dos puntos de media potencia medida en [Hz].
- Distancia de los polos al origen del círculo unitario $r = 0.95$
- Ganancia del filtro medida en veces $K = 1$

1º) Calculamos la frecuencia digital Ω_n del filtro notch que viene dada por la expresión:

$$\Omega_n = \frac{2 \cdot \pi \cdot f_c}{f_s} = \frac{2 \cdot \pi \cdot 50}{800} = 0.39270\ \left[\frac{rad}{s}\right]$$

2º) A continuación ubicamos los ceros y los polos sobre el círculo unitario. Los ceros irán sobre el círculo a la frecuencia Ω_n y los polos sobre el radio que une el centro del círculo con los ceros a una distancia 'r' del origen. Vale aclarar que los polos se agregan con el fin de que el filtro tenga una respuesta lineal y de esta forma la señal no resulte distorsionada. De esta manera, el diagrama de polos y ceros queda de la siguiente manera:

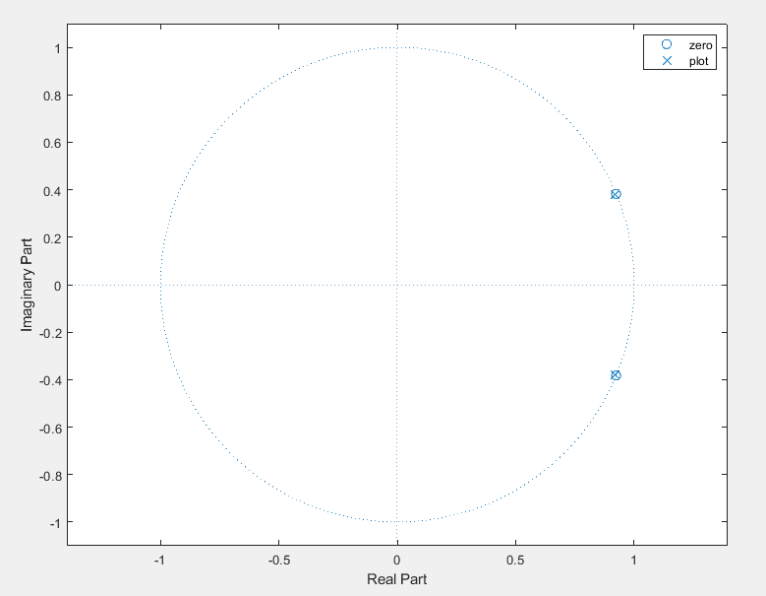


Figura (7) Ubicación de polos y ceros del filtro ranura ($r = 0.99\,; \Omega_P = \Omega_N$).

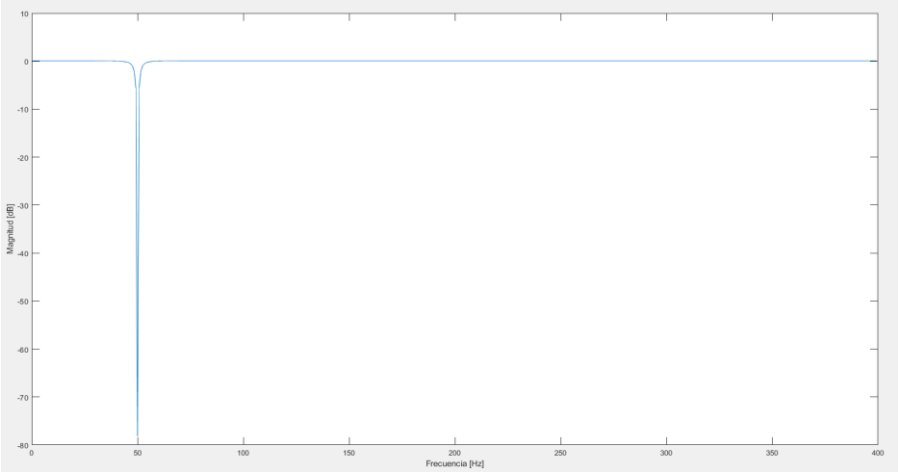


Figura (8) Respuesta en frecuencia del filtro ranura sin corrección.

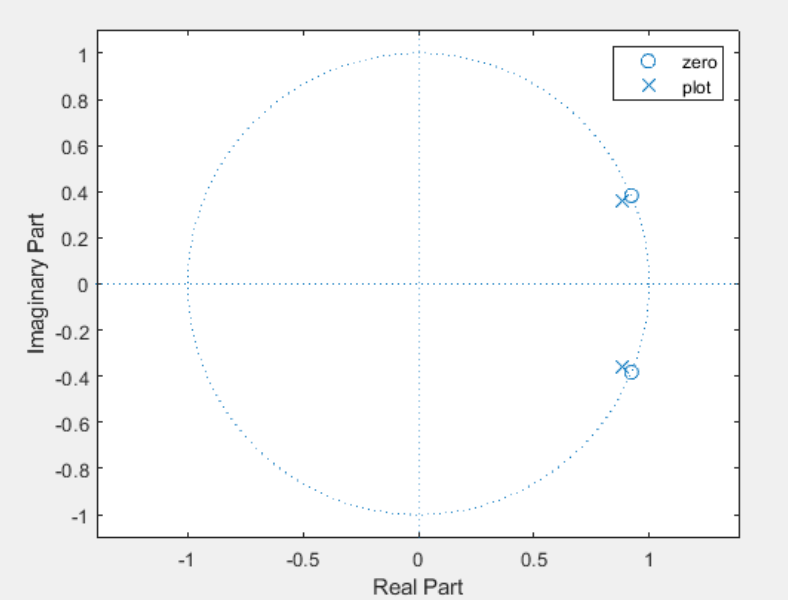


Figura (9) Ubicación de polos y ceros del filtro notch corregido ($r = 0.95$; $\Omega_P \neq \Omega_N$).

Los diagramas de polos y ceros de las figuras (7) y (9) ilustran como cambia la posición de los polos al modificar la distancia $'r'$ y frecuencia $'\Omega_P'$ respecto a la ubicación de los ceros sobre el círculo unitario y frecuencia $'\Omega_N'$. La función de transferencia correspondiente a este filtro posee ganancia en una de las bandas divididas por la frecuencia de corte, como se aprecia en la figura (8). Su ganancia debe ser unitaria en toda la banda de paso [24].

Esto se resuelve trasladando los polos respecto a los ceros en el diagrama de polos y ceros, como se muestra en la figura (9).

3º) Ubicados los polos y los ceros, ya se puede calcular la función de transferencia del filtro que estará dada por:

$$H(z) = K \cdot \frac{\left(z - e^{j\Omega_n}\right) \cdot \left(z - e^{-j\Omega_n}\right)}{\left(z - r \cdot e^{j\Omega_n}\right) \cdot \left(z - r \cdot e^{-j\Omega_n}\right)} \quad con\ K = 1$$

Utilizando la relación de Euler y operando algebraicamente, podemos escribir a la función $H(z)$ como:

$$H(z) = \frac{1 - [2 \cdot \cos(\Omega_n) \cdot z^{-1}] + z^{-2}}{1 - [2 \cdot r \cdot \cos(\Omega_n) \cdot z^{-1}] + (r^2 \cdot z^{-2})}$$

Finalmente, asignando los valores correspondientes, la función de transferencia del filtro ranura queda de la siguiente forma:

$$H(z) = \frac{1 - [1.84776]z^{-1} + z^{-2}}{1 - [1.84776 \cdot \rho]z^{-1} + [0.9025]z^{-2}}$$

Nota: El parámetro 'ρ' varía la posición de los polos y se ajusta de acuerdo a las diferentes necesidades.

Si tomamos $\rho = 0.95$, entonces $H(z)$ queda:

$$H(z) = \frac{1 - [1.84776]z^{-1} + z^{-2}}{1 - [1.7554]z^{-1} + [0.9025]z^{-2}}$$

Realizamos la corrección para el caso $H_{(1)} = 1$ y $H_{(-1)} = 1$ ajustando el ángulo del polo de Ω_N a Ω_P, manteniendo el valor del módulo:

$$\cos(\Omega_P) = \frac{[1 + \rho^2] \cdot \cos(\Omega_N)}{2 \cdot \rho} = \frac{[1 + 0.95^2] \cdot 0.7854}{2 \cdot 0.95}$$

$$\cos(\Omega_P) = 0.3932$$

$$k = \frac{1 + \rho^2}{2} = \frac{1 + 0.95^2}{2}$$

$$k = 0.95125$$

Finalmente la función de transferencia para el filtro ranura queda definida de la siguiente manera:

$$H(z) = 0.95125 \cdot \frac{1 - [1.84776]z^{-1} + z^{-2}}{1 - [0.74708]z^{-1} + [0.9025]z^{-2}}$$

En la figura (10) y (11) se puede apreciar la respuesta en frecuencia del filtro diseñado donde para la frecuencia de 50 [Hz] presenta una respuesta en magnitud de casi -600 [dB] siendo su factor de calidad Q de 35. Mientras que se produce una contrafase en su respuesta en fase para la frecuencia de interés.

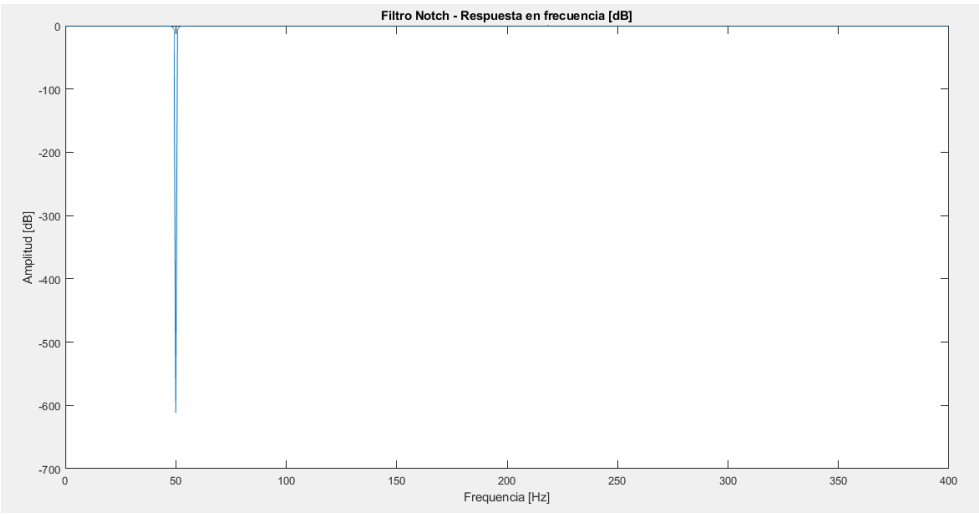


Figura (10) Respuesta en amplitud del filtro notch corregido.

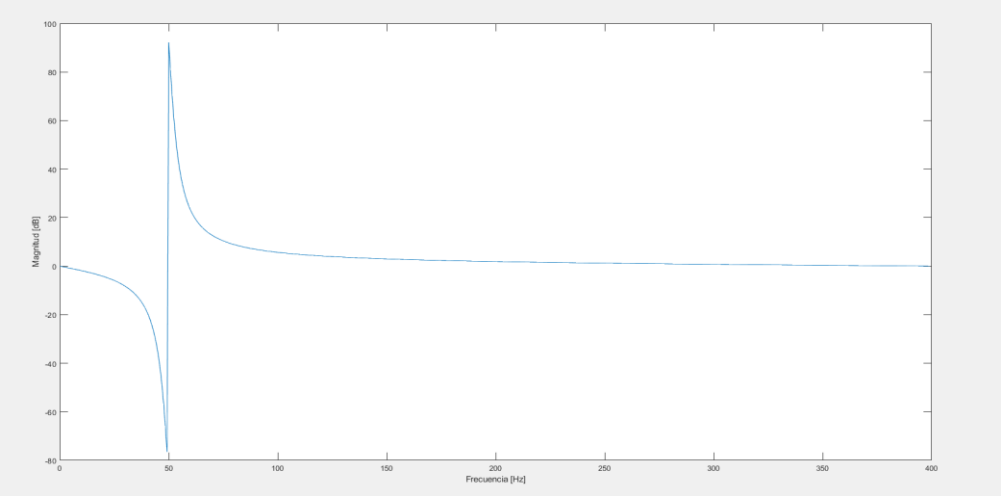


Figura (11) Respuesta en fase del filtro notch diseñado.

Siguiendo un diseño similar pero especificando el ancho de banda en 12 [Hz] en lugar del factor de calidad del filtro se obtuvo la respuesta en frecuencia ilustrada en las figuras (12) y (13).

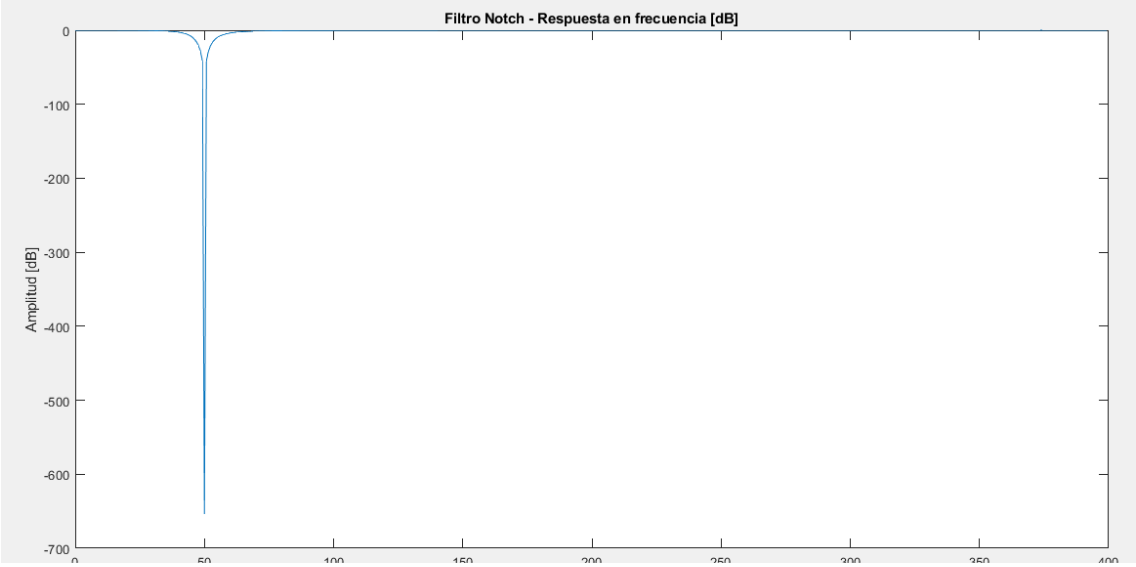


Figura (12) Respuesta en amplitud del filtro notch alternativo.

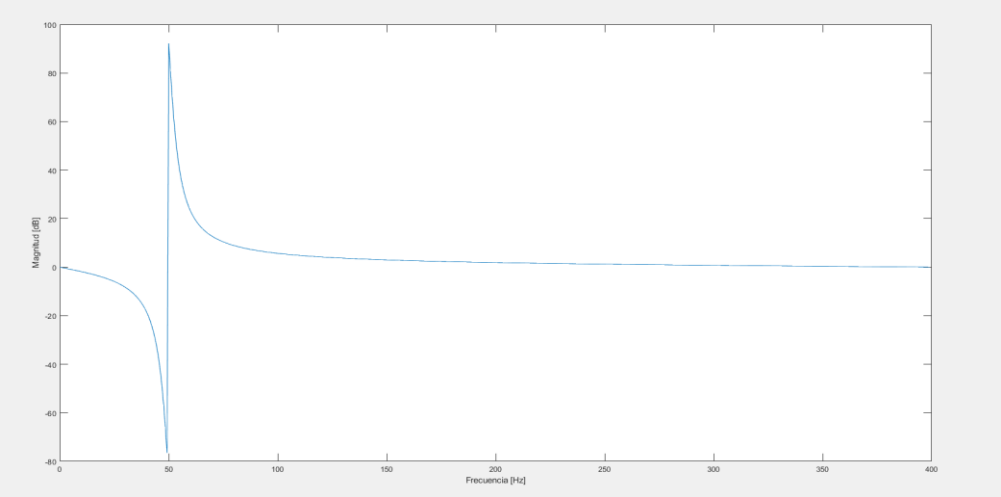


Figura (13) Respuesta en fase del filtro notch alternativo.

Para el segundo diseño se ajustó la distancia entre los polos y ceros para obtener el ancho de banda deseado [24]. A continuación se presenta el desarrollo matemático [24] que sustenta demuestra la relación de dependencia entre la distancia entre polos y ceros con el ancho de banda. En las figuras (14) y (15) se ilustra este concepto.

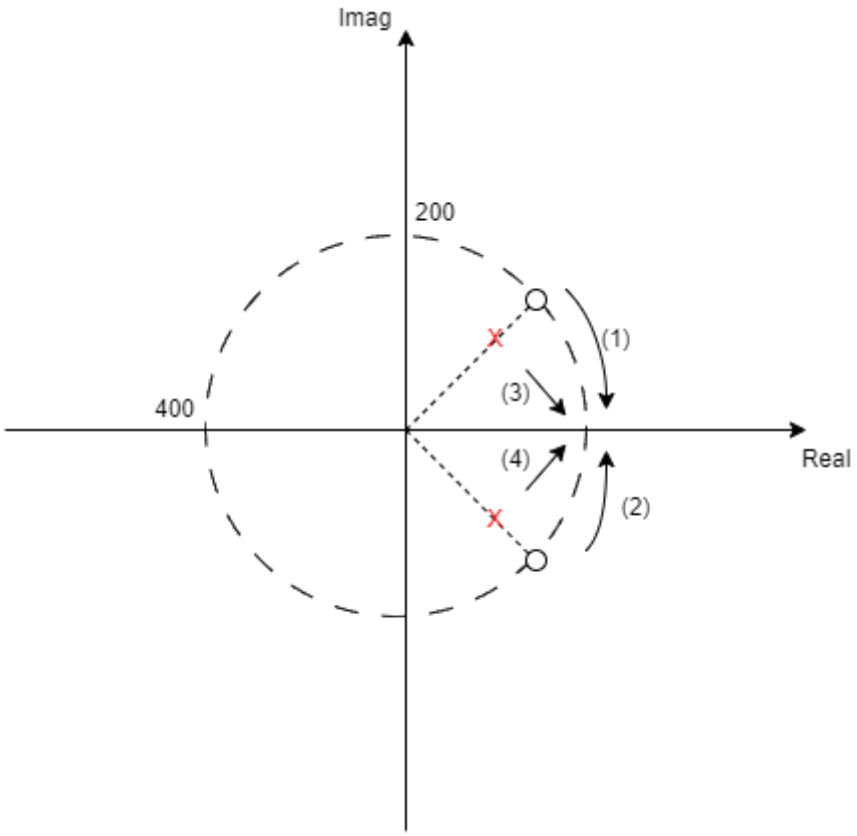


Figura (14) Diagrama de polos y ceros.

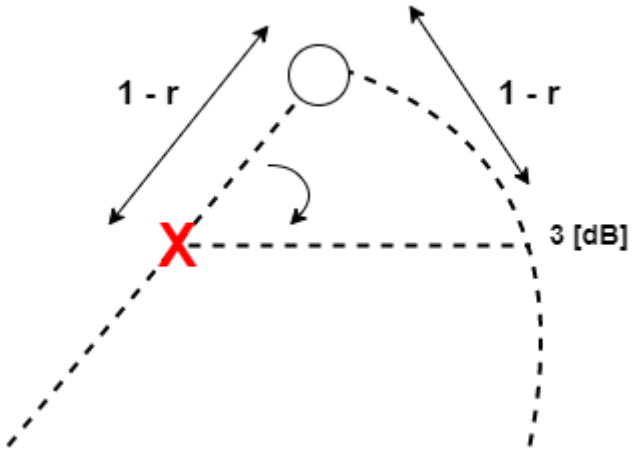


Figura (15) Desplazamiento en un diagrama de polos y ceros.

$$H(\Omega) = K \cdot \frac{\overbrace{(e^{j\Omega} - e^{j\Omega_n})}^{(1)} \cdot \overbrace{(e^{j\Omega} - e^{-j\Omega_n})}^{(2)}}{\underbrace{(e^{j\Omega} - r \cdot e^{j\Omega_n})}_{(3)} \cdot \underbrace{(e^{j\Omega} - r \cdot e^{-j\Omega_n})}_{(4)}}$$

Si el ángulo que se forma es muy pequeño, tenemos:

$$BW \approx \frac{2(1-r)}{2\pi} \cdot f_s = \left(\frac{1}{\pi} - \frac{r}{\pi}\right) \cdot f_s$$

$$\frac{BW \cdot \pi}{f_s} \approx 1 - r \rightarrow r = 1 - \frac{BW \cdot \pi}{f_s}$$

Nota: Para su implementación en Matlab, véase Anexo B.

Diseño de Filtro Pasa Bajos

Tomando como referencia no exclusiva [16], diseñamos un filtro digital pasa bajos LP del tipo Butterworth con frecuencia de corte $f_c = 120\ [Hz]$, que atenúe más de $10\ [dB]$ para frecuencias mayores a $200\ [Hz]$. Consideramos la frecuencia de muestreo asignada $f_s = 800\ [Hz]$.

1º) Pasamos de frecuencias digitales a frecuencias analógicas:

Frecuencia de corte

$$\omega_{ac} = \frac{2}{T_s} \cdot tg\left(\frac{\omega_{dc} \cdot T_s}{2}\right) = 2 \cdot f_s \cdot tg\left(\frac{\omega_{dc}}{2f_s}\right)$$

$$\omega_{ac} = 2 \cdot 800 \cdot tg\left(\frac{2 \cdot \pi \cdot 120}{2 \cdot 800}\right) \rightarrow \omega_{ac} = 815.24\left[\frac{rad}{s}\right]$$

Frecuencia de stop

$$\omega_{as} = \frac{2}{T_s} \cdot tg\left(\frac{\omega_{ds} \cdot T_s}{2}\right) = 2 \cdot f_s \cdot tg\left(\frac{\omega_{ds}}{2f_s}\right)$$

$$\omega_{ac} = 2 \cdot 800 \cdot tg\left(\frac{2 \cdot \pi \cdot 200}{2 \cdot 800}\right) \rightarrow \omega_{ac} = 1.600\left[\frac{rad}{s}\right]$$

2º) A partir de ω_{ac}, ω_{as} y la atenuación deseada determinamos el orden de nuestro filtro:

$$n = \frac{1}{2} \cdot \frac{\log\left(10^{\frac{At}{10}} - 1\right)}{\log\left(\frac{\omega_{as}}{\omega_{ac}}\right)} = \frac{1}{2} \cdot \frac{\log\left(10^{\frac{10}{10}} - 1\right)}{\log\left(\frac{1.600}{815.24}\right)} = 1.629 \rightarrow adoptamos\ n = 2$$

3º) Verificamos la atenuación:

$$At_{dB} = 10 \cdot \log\left(1 + \left[\frac{\omega_{as}}{\omega_{ac}}\right]^{2n}\right) = 10 \cdot \log\left(1 + \left[\frac{1.600}{815.24}\right]^{4}\right) \rightarrow At_{dB} = 11.997[dB]$$

4º) Por tabla obtenemos los coeficientes de Butterworth para la función de transferencia $H(s)$:

$$H(s) = \frac{1}{1 + 1.414s + s^2}$$

5º) Reemplazamos $s = \frac{1-z^{-1}}{K_L(1+z^{-1})}$ donde $K_L = tg\left(\frac{\omega_c \cdot T}{2}\right)$

$$K_L = tg\left(\frac{\omega_c \cdot T}{2}\right) = tg\left(\frac{\omega_c}{2f_s}\right) = tg\left(\frac{2 \cdot \pi \cdot 120}{2 \cdot 800}\right) \rightarrow K_L = 0.509$$

$$s = \frac{1 - z^{-1}}{0.509(1 + z^{-1})}$$

$$H(z^{-1}) = \frac{1}{1 + \sqrt{2}\left(\frac{1 - z^{-1}}{K_L(1 + z^{-1})}\right) + \left(\frac{1 - z^{-1}}{K_L(1 + z^{-1})}\right)^2}$$

$$H(z^{-1}) = \frac{K_L^2(1 + z^{-1})^2}{K_L^2(1 + z^{-1})^2 + \sqrt{2}(1 - z^{-1})K_L(1 + z^{-1}) + (1 - z^{-1})}$$

$$H(z^{-1}) = \frac{K_L^2[1 + 2z^{-1} + z^{-2}]}{K_L^2[1 + 2z^{-1} + z^{-2}] + \sqrt{2}K_L[1 - z^{-2}] + [1 - z^{-1}]}$$

$$H(z^{-1}) = \frac{[K_L^2]z^{-2} + [2K_L^2]z^{-1} + [K_L^2]}{[K_L^2 - \sqrt{2}K_L]z^{-2} + [2K_L^2 - 1]z^{-1} + [K_L^2 + \sqrt{2}K_L + 1]}$$

$$\rightarrow H(z^{-1}) = \frac{(0.259)z^{-2} + (0.518)z^{-1} + (0.259)}{(-0.461)z^{-2} + (-0.481)z^{-1} + (1.979)}$$

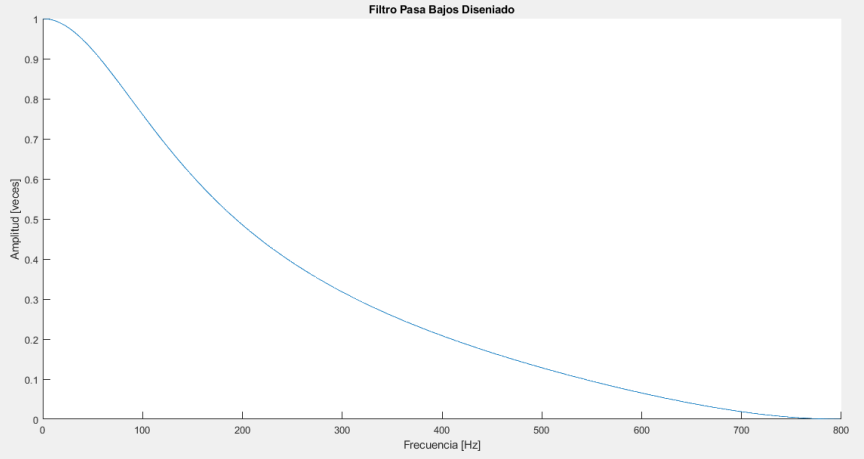


Figura (16) Filtro Pasa-bajos (Diseño 1).

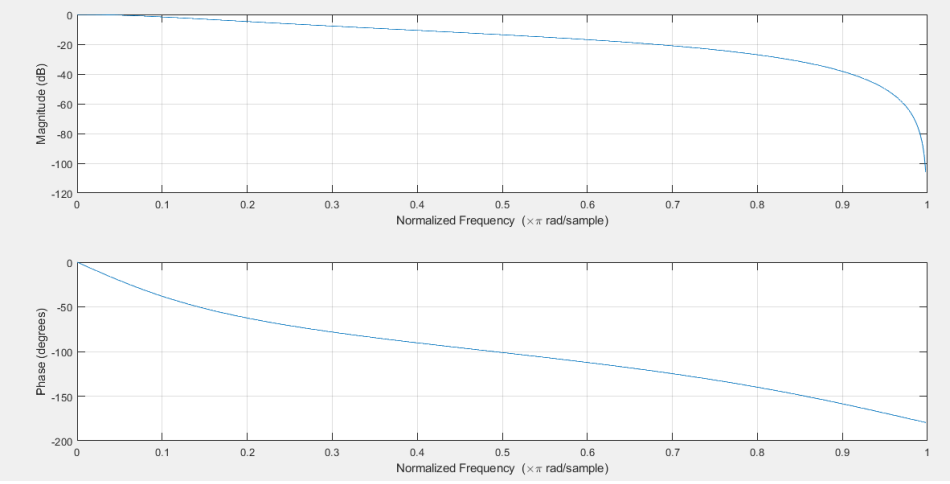


Figura (17) Respuesta en frecuencia del Filtro Pasa-bajos (Diseño 1).

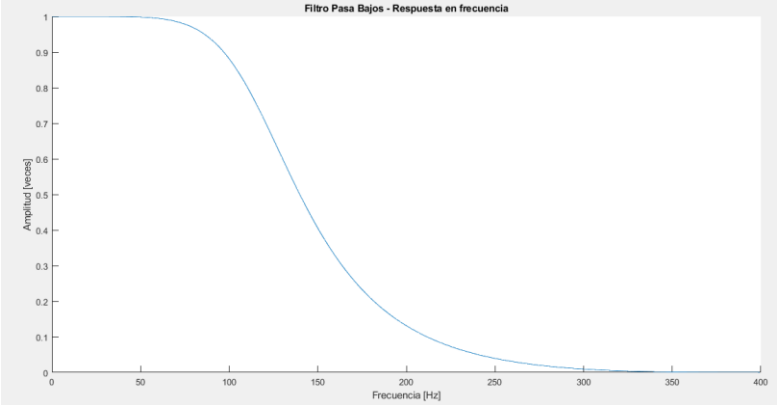


Figura (18) Filtro Pasa-bajos (Diseño 2).

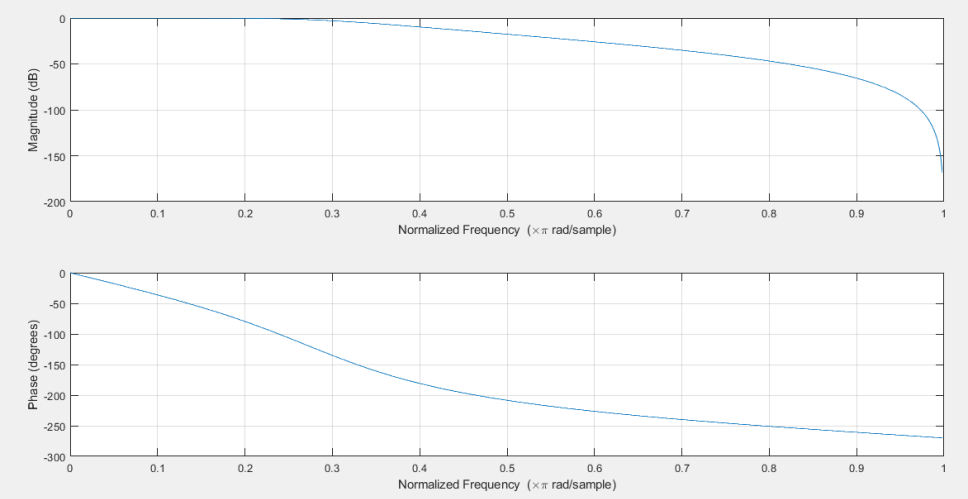


Figura (19) Respuesta en frecuencia del Filtro Pasa-bajos (Diseño 2).

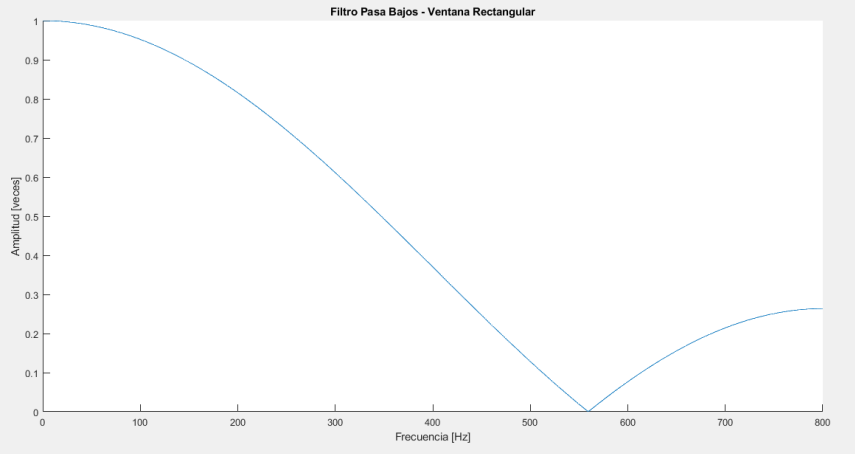


Figura (20) Filtro Pasa-bajos (Diseño 3).

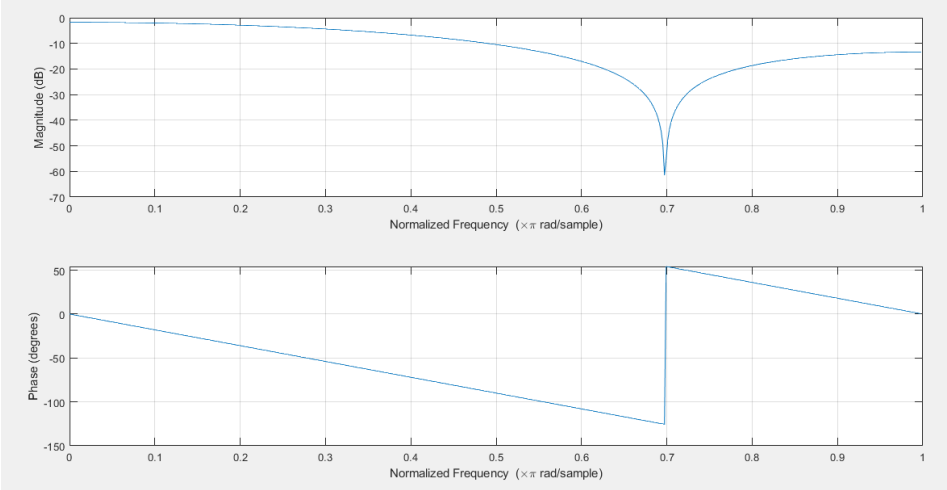


Figura (21) Respuesta en frecuencia del Filtro Pasa-bajos (Diseño 3).

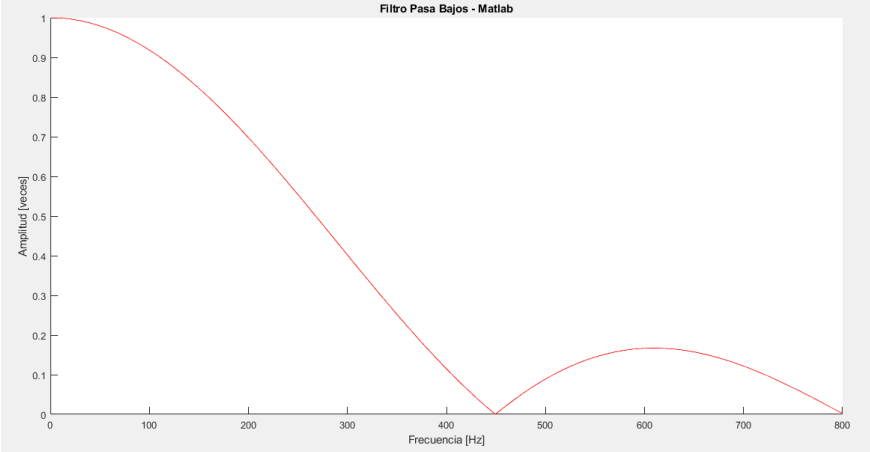


Figura (22) Filtro Pasa-bajos (Diseño 4).

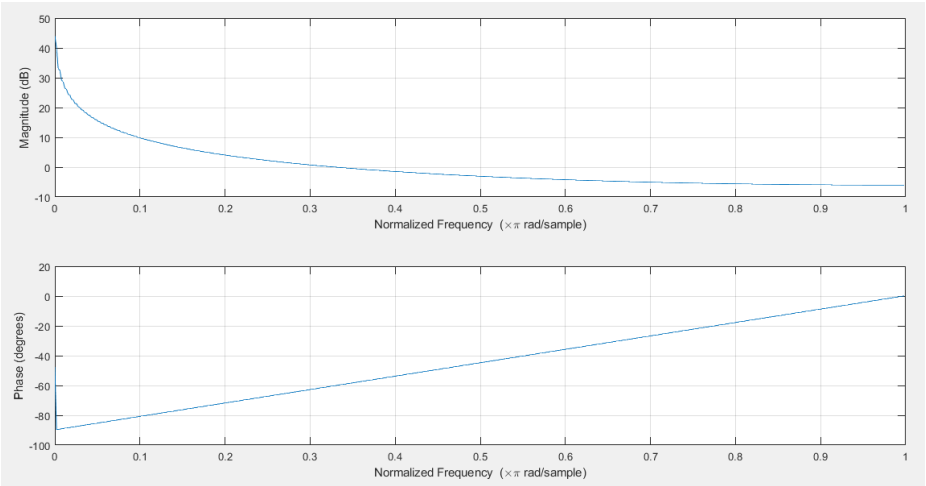


Figura (23) Respuesta en frecuencia del Filtro Pasa-bajos (Diseño 4).

Generalizando para los cuatro filtros diseñados, en las figuras (16), (18), (20) y (22) observamos que la respuesta en amplitud con la frecuencia que ofrece cada filtro, especialmente para el rango de valores de nuestro interés. Se lograron resultados aceptables a partir de los cálculos realizados, obteniendo frecuencias de corte entre 120 [Hz] y 150 [Hz]; esto debido al orden de cada filtro.

Aclaración: Se probaron los cuatro diseños de filtros pasa-bajos y finalmente se optó por el diseño 2, el cual sigue una distribución Butterworth.

Nota: Para sus implementaciones en Matlab, véase Anexo B.

La respuesta de ambos filtros en cascada se muestra a continuación en la figura (24).

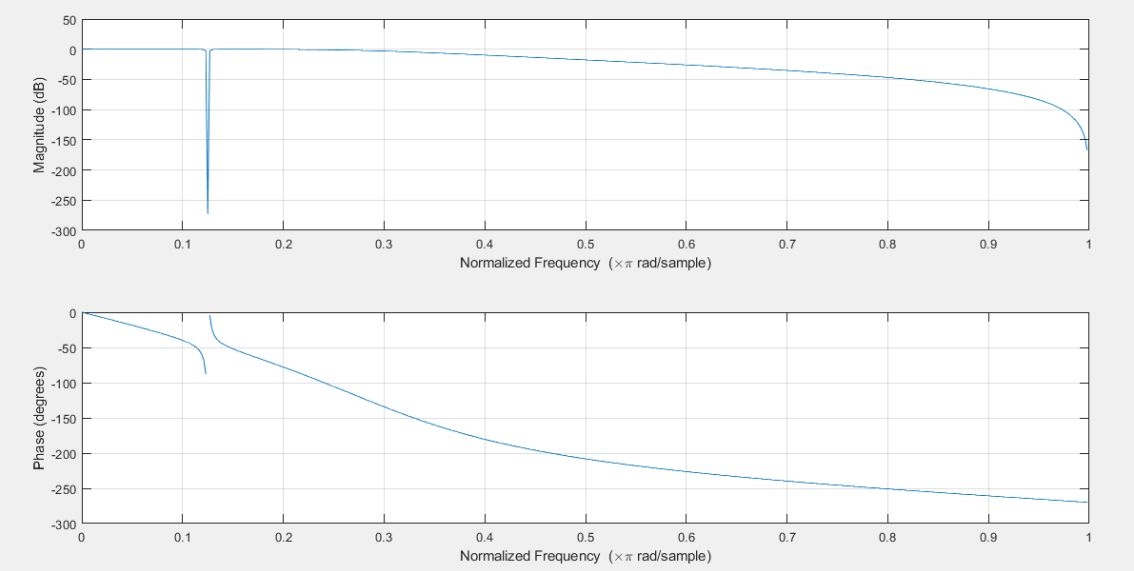


Figura (24) Respuesta en frecuencia de Filtro Notch conectado en cascada con Filtro Pasa-bajos.

Nota: Para sus implementaciones en Matlab, véase Anexo B.

3) Desarrollar una función en Matlab que implemente el filtrado de fase cero, debe procesar la señal muestra a muestra, es decir se llama la función pasándole el valor actual de la señal y devuelve el valor actual de la salida, actualizándose las variables o registros internos del sistema.

Un filtro de fase cero es un filtro para el cual el cambio de fase es cero para todas las frecuencias. Estos filtros son anticipatorios y por lo tanto no son físicamente realizables. Por ejemplo, la mitad de la energía llega antes que el tiempo de referencia, así que recibe su señal de salida antes de que llegue la señal de entrada. Si la señal de entrada a un filtro de fase cero es simétrica, entonces la salida también es simétrica. Los filtrados de fase cero pueden ser aproximados usando un filtro de fase lineal [22], un filtro de fase mixta cambia frecuencias de componentes proporcionales a su frecuencia, y luego retrasando el tiempo de referencia. Un filtro de fase cero no produce distorsión de fase [22], [23].

Si bien, como se explica en [17], un sistema IIR siempre se implementa de forma recursiva pero un sistema recursivo no siempre es de tipo IIR. Como es el caso de la solución que se propone a continuación.

Partiendo de la base teórica desarrollada en [2] y [3], y según la propuesta planteada por los autores Powell y Chau en su artículo científico [1] trabajamos con el siguiente diagrama de bloques para representar la forma en que se estructura nuestro filtro de fase cero:

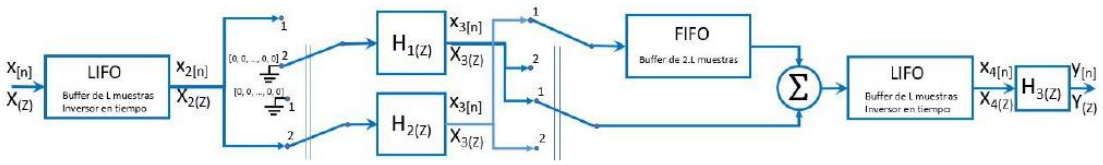


Figura (25) Diagrama de bloques de la implementación de Powell y Chau.

<u>Aclaración</u>: Dada las opciones propuestas en las referencias [1], [13], [14] donde el problema que se nos plantea como equipo puede resolverse por dos caminos igualmente válidos: Implementando un filtro digital de fase lineal en tiempo real o mediante la implementación de un filtro de fase lineal en tiempo diferido.

Optamos por la realización de un filtro de fase lineal en tiempo diferido, principalmente porque la cátedra nos proporciona las señales ECG a través de archivos formato '.m' los cuales presentan una cantidad finita de muestras [17]. En segundo lugar, ya que el proyecto integrador tiene por finalidad didáctica poner en práctica los conocimientos adquiridos y el alcance del mismo no cubre la implementación del filtro en un microcontrolador.

A continuación se describe detalladamente el código de programa del filtro de fase cero implementado en Matlab. Los siguientes diagramas de bloques representan las diferentes etapas del sistema que define a nuestro filtro y en el mismo se pueden visualizar las variables y registros que intervienen en cada etapa.

Nota: La descripción que se realiza a continuación es válida para las tres señales ECG facilitadas por la cátedra ya que las mismas presentan las mismas características (mismo número de muestras e igual frecuencia de muestreo).

Nota: Para llevar a cabo el informe, se tomó como referencia ilustrativa los resultados parciales y finales de procesar la señal 'ECG_senal_1a.mat'.

Primero comenzamos con una limpieza de la ventana de comandos y la inicialización aquellas variables globales y valores constantes que resultan de utilidad para la ejecución del programa principal:

```
% Limpiar la ventada de comandos
close all
clc

%%
%Inicializacion de parametros para las señales ECG
ns=0:length(signal)-1;      %Barrido de tiempo igual al numero de
datos adquiridos
ns=ns*1/sample_frequency;   %Normalizacion por T=1/Fs

figure(1)
Plotear_Signal(ns,signal);   % Gráfico de la señal ECG original
(informe)
%figure(2)
%Plotear_Signal_2(ns,signal); % Gráfico de la señal ECG original
(presentacion)

%Inicializacion de otros parametros
size_impulse=100;   %Tamaño del vector de la senial Impulso
L=500;              %Tamaño de los buffers LIFO
j=1;
j2=1;
flag=1;
flag2=1;
tiempo=linspace(1,L,L);
tiempo2L=linspace(1,2*L,2*L);
fp=1;
fp2=1;
BLOQUE=1; %Revisar si se puede rescatar esto
```

En esta sección del código se grafica la señal ECG que ingresa al programa muestra a muestra; la figura (26) muestra el resultado obtenido.

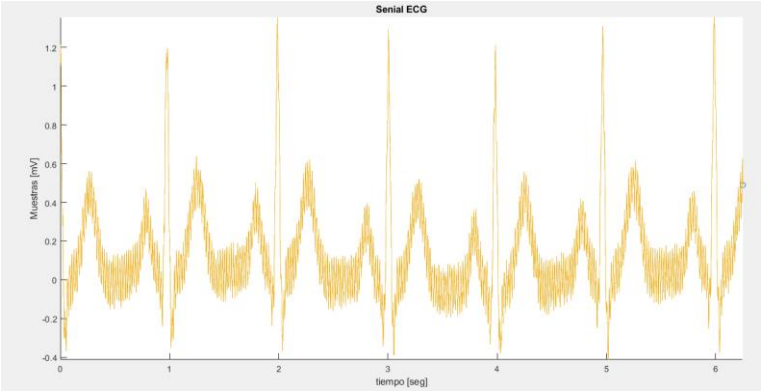


Figura (26) Gráfica de la señal ECG original.

A continuación se definieron los parámetros y se diseñaron los filtros pasa-bajos y filtros ranuras.

```
%Filtro Pasa-bajos
Fc=120;                                 %Frecuencia del filtro Pasa-
bajos
N=3;                                    %Orden del filtro Pasa-bajos
[num_lp,den_lp] = Lowpass_Design(N,Fc,sample_frequency);
%Coeficientes de la Funcion Transferencia
Lowpass_Frequency_Response(num_lp,den_lp,sample_frequency);
%Respuesta en frecuencia del Filtro Pasa-bajos

%Otros disenios del Filtro Pasabajos
Lowpass_Filters(N,Fc,sample_frequency);          %Figura (5)
Filtro_Pasabajos_Diseniado(sample_frequency);    %Figura (6)
Lowpass_Matlab(N,Fc,sample_frequency);           %Figura (7)

%Filtro Notch
CT_notch_freq=50;                          %Frecuencia del Filtro Notch
AB=12;                                     %Ancho de banda
w_n=CT_notch_freq/(sample_frequency/2); %Frecuencia de rechaza banda
Q=35;                                      %Factor de calidad
bw=w_n/Q;
[num_n,den_n] = iirnotch(w_n,bw);          %Coeficientes

figure(8)
Notch_Frequency_Response(num_n,den_n,sample_frequency);  %Figura (8)

% Visualizamos el plano z
figure(9);
zplane(num_n,den_n);legend('zero','plot');

%Otros disenios del Filtro Notch
[num_n2,den_n2]=Notch_Design(CT_notch_freq,sample_frequency,AB);
figure(10)
Notch_Frequency_Response(num_n2,den_n2,sample_frequency);
```

```
% Visualizamos el plano z
figure(11);
zplane(num_n2,den_n2);legend('zero','plot');
```

Para la conexión en cascada de los filtros Pasa-bajos y Notch seleccionados se implementaron las siguientes funciones:

```
%Conexion en cascada
FTlp = tf(num_lp,den_lp);    %Generamos la funcion transferencia del
filtro pasa bajo
FTn = tf(num_n,den_n);       %Generamos la funcion transferencia del
filtro notch

FTcascada=FTlp*FTn;          %Funcion de transferencia del sistema
total
[num,den] = tfdata(FTcascada,'v');


%NOTA: La funcion tfdata genera los vectores que contienen los
coeficientes
%del numerador y denominador de la funcion transferencia
```

Luego, fue necesario establecer el tamaño de los buffers $LIFO$ y $FIFO$ a partir de la elección de un número $'L'$ de muestras apropiado. Para ello, se analizó la respuesta del par de filtros en cascada a la señal impulso. La figura (27) ilustra la respuesta obtenida:

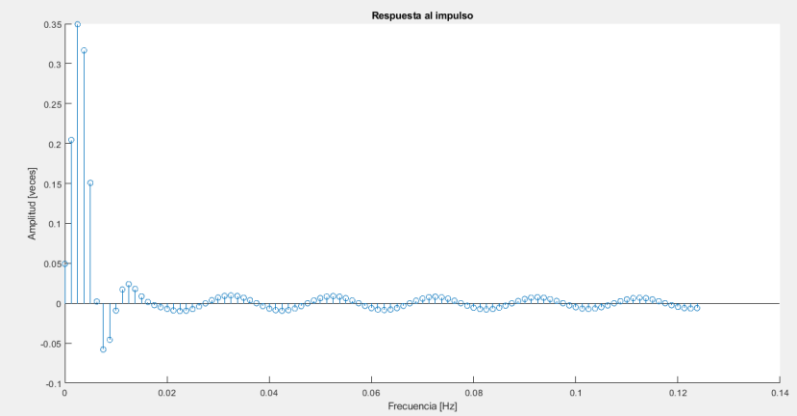


Figura (27) Respuesta del par de filtros en cascada al impulso.

A partir de esta respuesta se determinó que una cantidad apropiada de muestras es $L = 500$ lo cual, siguiendo la propuesta del artículo científico que alienta este trabajo, corresponde a un total de cinco bloques de procesamiento. Se inicializaron los buffers:

```
%Generamos los registros
buffer_LIFO1=zeros(1,L);     %buffer LIFO 1
buffer_LIFO2=zeros(1,L);     %buffer LIFO 2
buffer_FIFO=zeros(1,2*L);    %buffer FIFO
```

```
signal_salida=zeros(1,length(signal));
salida_LIFO1=0;
salida_H1=0;
salida_H2=0;
salida_FIFO=0;
salida_LIFO2=0;
SUMA=0;
salida_H3=0;

%Condiciones iniciales nulas
z1=0;
z2=0;
z3=0;
```

A continuación se adjunta el segmento de código correspondiente al filtro de fase cero siguiendo la propuesta [1] de Powell y Chau:

```
for i=1:length(signal)

[salida_LIFO1,buffer_LIFO1,j,flag,BLOQUE]=LIFO(i,j,flag,buffer_LIFO1,s
ignal,BLOQUE);
    if(flag==1)
        [salida_H1,z1]=Filtrar(num,den,0,z1);
        [salida_H2,z2]=Filtrar(num,den,salida_LIFO1,z2);
        [salida_FIFO,buffer_FIFO]=FIFO(L,i,buffer_FIFO,salida_H2);
        SUMA=salida_FIFO+salida_H1;
    else
        [salida_H1,z1]=Filtrar(num,den,salida_LIFO1,z1);
        [salida_H2,z2]=Filtrar(num,den,0,z2);
        [salida_FIFO,buffer_FIFO]=FIFO(L,i,buffer_FIFO,salida_H1);
        SUMA=salida_FIFO+salida_H2;
    end

[salida_LIFO2,buffer_LIFO2,j2,flag2]=LIFO2(j2,flag2,buffer_LIFO2,SUMA)
;
    [salida_H3,z3]=Filtrar(num,den,salida_LIFO2,z3);
    signal_salida(i)=salida_H3;
```

Nota: El código de programa correspondiente a las funciones internas implementadas se adjunta en el anexo B de este informe.

Una vez procesada la señal en su totalidad y almacenada en un vector de tamaño igual a la cantidad de muestras que conforman la señal ECG original se procede a graficar la señal filtrada obtenida y, a continuación se lleva a cabo el análisis del espectro de frecuencia, tanto de la señal original como filtrada.

```
figure(13)
Plotear_Senial_3(ns,signal_salida); %Gráfico de la señal ECG filtrada
(informe)
%Plotear_Signal_2(ns,signal_salida); %Gráfico de la señal ECG filtrada
(presentacion)
```

```
%Analisis del espectro de frecuencia

%Senial ECG Original
u_out2=fft(signal);
f_out2=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
P_out2=2.*(abs(u_out2(1:length(signal)/2)/length(signal)));

%Senial ECG Filtrada
u_out1=fft(signal_salida);
f_out1=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
P_out1=2.*(abs(u_out1(1:length(signal)/2)/length(signal)));

figure(14)
subplot(2,1,1);
plot(f_out2,P_out2);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Original');
subplot(2,1,2);
plot(f_out1,P_out1);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada');
```

La señal ECG resultante de ser procesada por el filtro de fase cero diseñado por el equipo de trabajo puede visualizarse en la figura (28).

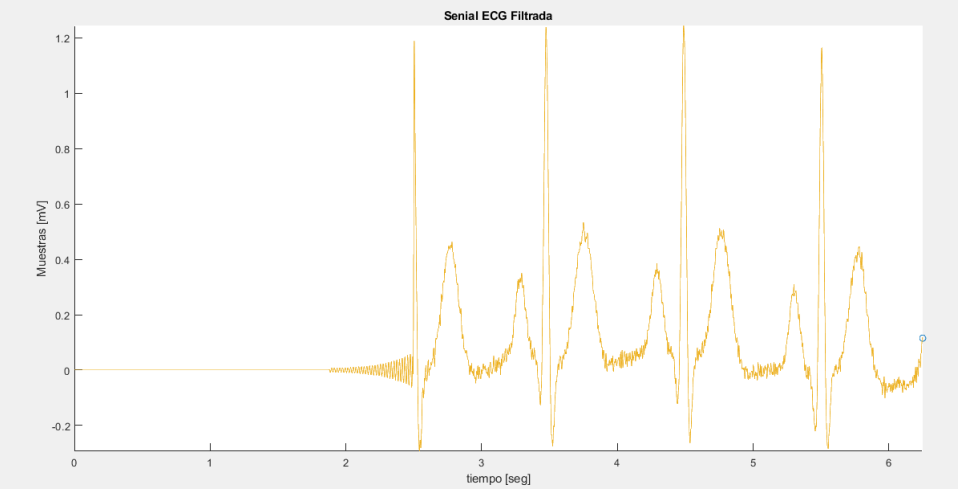


Figura (28) Gráfica de la señal ECG Filtrada.

Nota: Las observaciones y conclusiones pertinentes a cada resultado obtenido se detallan en las secciones futuras correspondientes.

Respecto al análisis del espectro de frecuencias, se grafica el espectro de frecuencia de la señal ECG original y el de la señal ECG filtrada; ver figura (29).

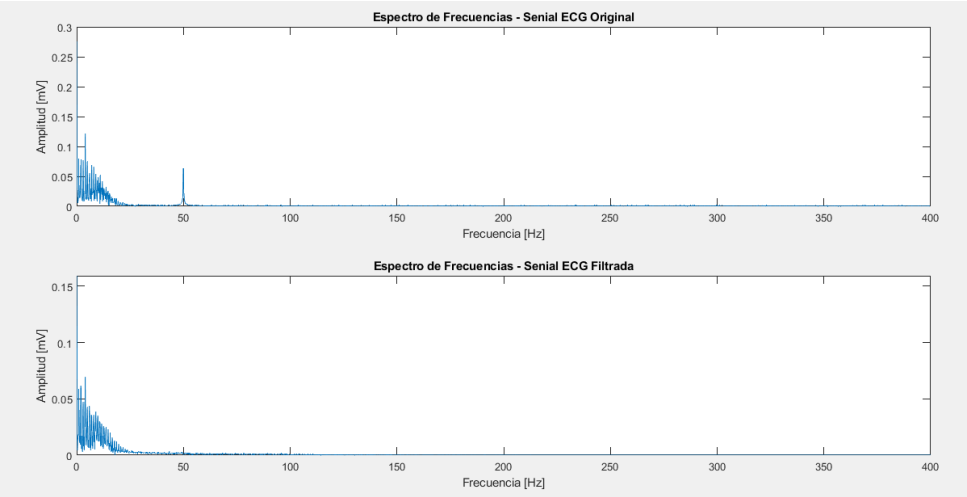


Figura (29) Espectro de frecuencias de la señal ECG original y de la filtrada.

Por último, se propone una breve comparación entre los resultados obtenidos con el filtro de fase cero diseñado y los resultados que se obtienen aplicando la función $'filtfilt'$ que nos ofrece Matlab.

```
%Comparamos los resultados obtenidos con la funcion 'filtfilt' de
Matlab
signal_salida_filtfilt = filtfilt(num,den,signal);

figure(15)
subplot(2,1,1);
plot(ns,signal_salida);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Senial ECG Filtrada');
subplot(2,1,2);
plot(ns,signal_salida_filtfilt);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Senial ECG Filtrada - filtfilt');

u_out3=fft(signal_salida_filtfilt);
f_out3=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
P_out3=2.*(abs(u_out3(1:length(signal)/2)/length(signal)));
```

```
figure(16)
subplot(2,1,1);
plot(f_out1,P_out1);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada');
subplot(2,1,2);
plot(f_out3,P_out3);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada (filtfilt)');
```

La figura (30) ilustra la señal ECG filtrada mediante el filtro de fase cero diseñado y la función que ofrece Matlab, mientras que la figura (31) muestra el espectro de frecuencias de las señales mencionadas anteriormente.

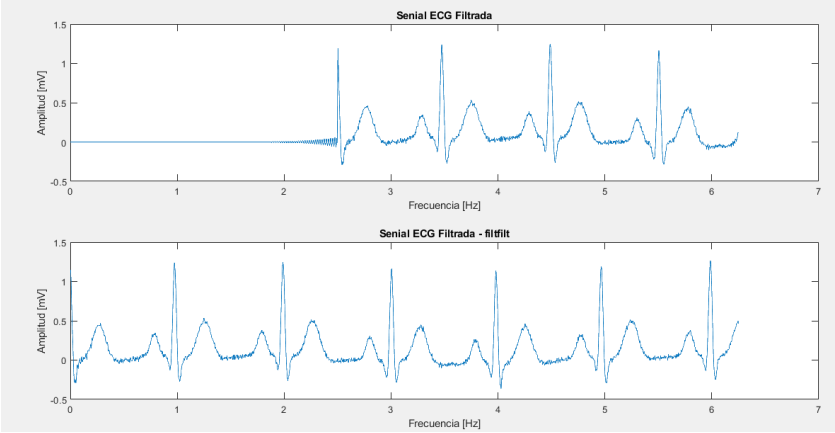


Figura (30) Señal ECG filtrada mediante el filtro diseñado y con la función de Matlab.

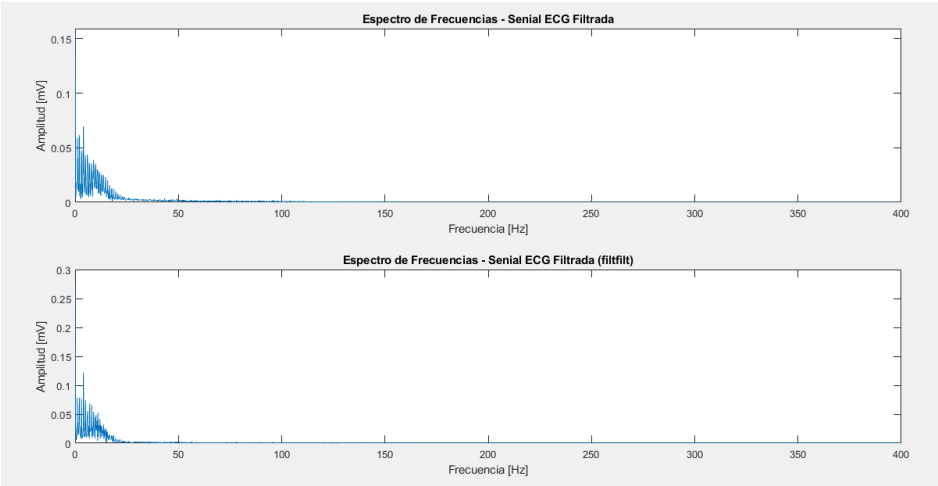


Figura (31) Espectro de frecuencias de la señal ECG filtrada mediante el filtro diseñado y con la función de Matlab.

Funcionamiento

La figura (32) muestra un diagrama de bloques conceptual de cómo se estructura el código de programa a nivel de registros y a continuación se describe el recorrido que realiza un segmento de la señal para ser procesada y presentada a la salida.

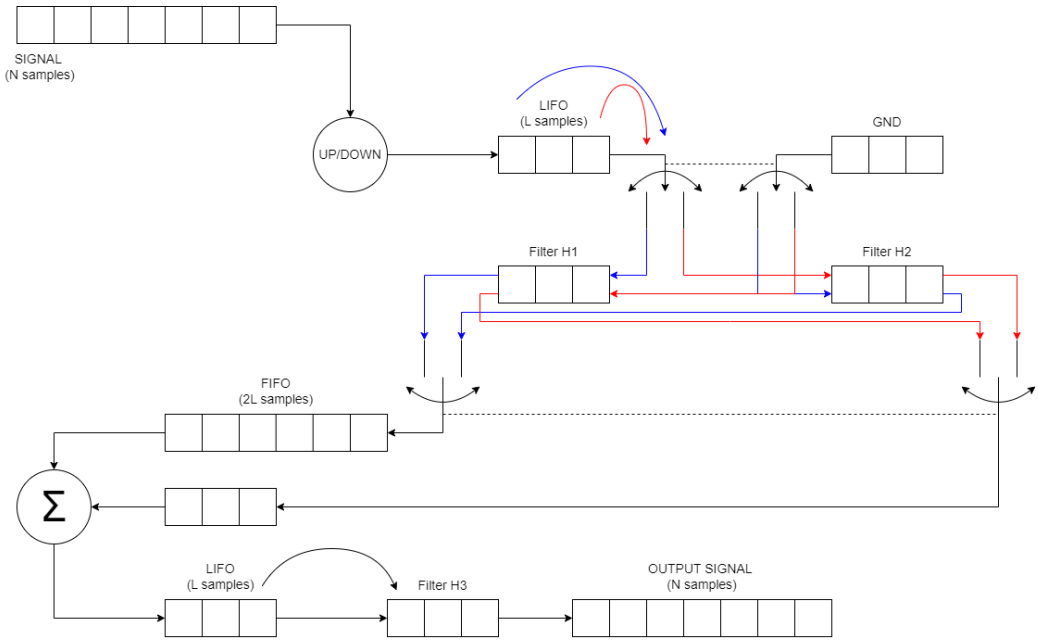


Figura (32) Diagrama de bloques conceptual de la implementación de Powell y Chau.

Inicialmente, el filtro de fase cero tiene todos sus registros vacíos y las llaves en la posición ‘1’. La señal ECG aún no ha ingresado. La figura (33) representa este estado inicial.

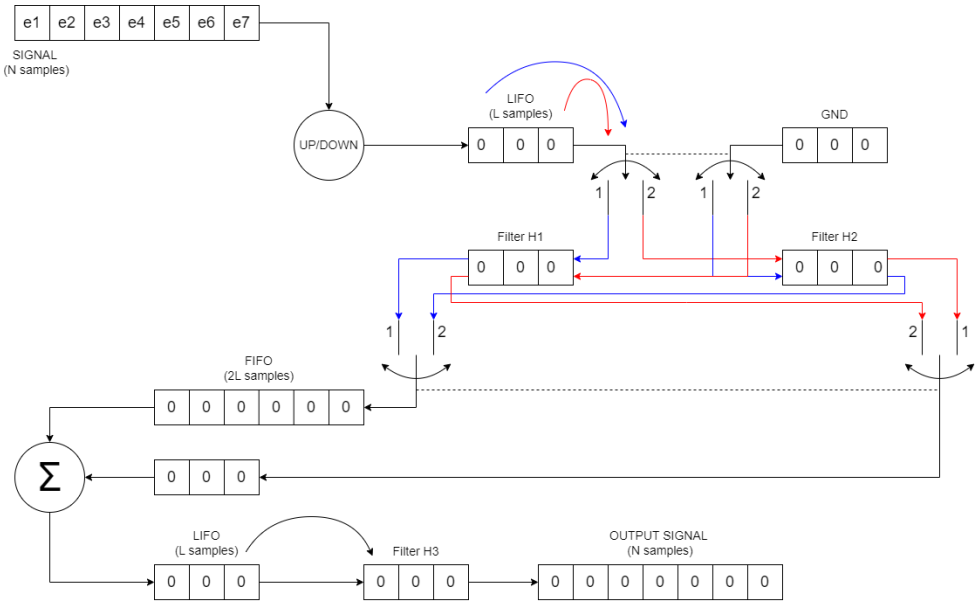


Figura (33) Diagrama de bloques del sistema en su estado incial.

Cada pulso de reloj en el microprocesador se implementó como un ciclo de bucle 'for' en el código de programa. En cada ciclo, se toma una muestra de la señal y se comienza a almacenar en el primer registro LIFO. Tal almacenamiento demanda la liberación de un espacio en dicho registro lo cual desencadena una serie de desplazamiento en los registros consecutivos, de manera que la información almacenada sea procesada y presentada a la salida. Esto significa, se procesa el dato anterior antes de almacenar uno nuevo.

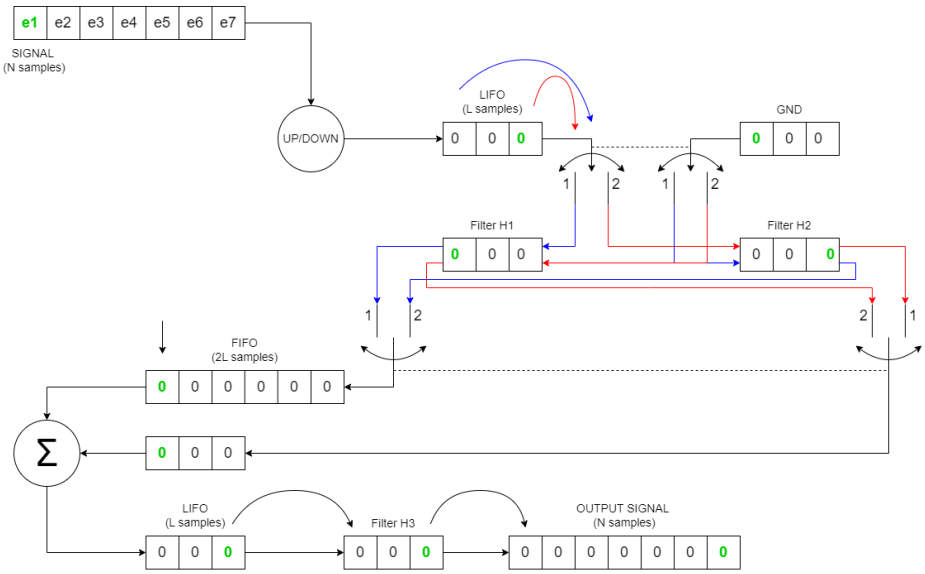


Figura (34) Desplazamiento de una muestra a través de los buffers.

Para tomar el primer elemento (e1) de la señal, es necesario liberar un espacio en cada uno de los registros. En la figura (34) se señalan de color verde aquellos elementos que sufren un desplazamiento a causa del elemento e1 de la señal que, a su vez, implica el filtrado de una muestra por los filtros H1, H2 y H3. Por otro lado, el operador sumador sumará el valor recibido del filtro en estado de relajación con el primer valor que ingresó al registro FIFO para continuar con el procesamiento. La señal de salida (output signal) presenta en pantalla, el segmento de la señal de tamaño 'L' que fue procesado en un estado anterior.

Nota: Recordar que en los buffers LIFO, el último elemento en ingresar es el primero en salir. Es decir, el segmento de señal almacenado en este registro experimenta una inversión en el tiempo. En el caso del buffer FIFO, el primer elemento en ingresar es el primero en salir. En este caso, no hay inversión en el tiempo.

Durante el primer ciclo, asumimos que las llaves se encuentran en la posición '1'. Cada elemento de la señal almacenada de forma ascendente en el buffer LIFO es filtrada por el filtro H1 mientras que el filtro H2 se encuentra en relajación, es decir, sus entradas en cada ciclo de reloj son nulas. Esto podemos apreciarlo en la figura (35).

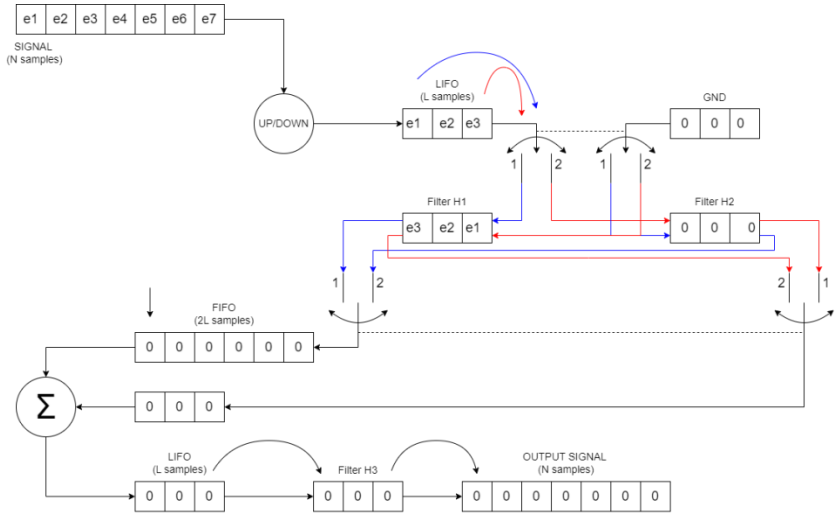


Figura (35) Ciclo de operación de los filtros H1 y H2.

El primer registro LIFO almacena los primeros ‘L’ los cuales sufren una inversión en el tiempo al ser tomados muestra a muestra para ser filtrados por H1. En la figura (35) vemos que se almacenaron de forma ascendente del elemento ‘e1’ al elemento ‘e3’. Luego el filtro H1, toma la muestra ‘e3’ y la filtra, a continuación repite el proceso para la muestra ‘e2’ y así sucesivamente. Esto se representa por el recorrido azul en el diagrama de la figura (35).

Una vez que el buffer LIFO ocupó toda su memoria, se invierte la forma de almacenamiento, es decir, las nuevas muestras de la señal ECG comienzan a llevar los espacios que se llenaron al último debido a que estos elementos son los primeros en ser procesados, es decir, para el ejemplo implementado, la muestra ‘e4’ ocupará el lugar del elemento ‘e3’ y no la posición de la muestra ‘e1’. La reescritura en la memoria del buffer LIFO en el nuevo ciclo se ilustra en la figura (36).

Esta técnica de almacenamiento disminuye el tiempo de procesamiento y agiliza la reescritura de memoria haciendo un mejor uso de esta [25].

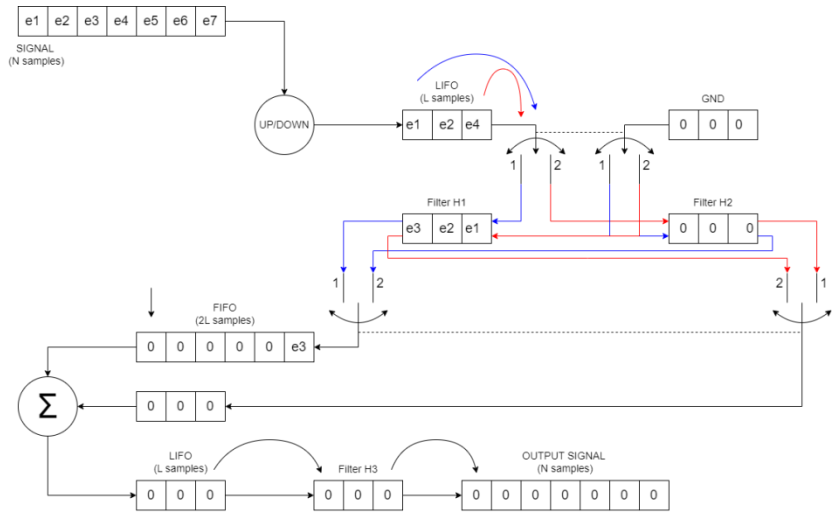


Figura (36) Modo de reescritura del buffer LIFO.

En esta instancia de procesamiento, los demás registros permanecen vacíos. El almacenamiento en el primer buffer LIFO se lleva a cabo de forma descendente. La muestra ‘e3’ filtrada por H1 continua su recorrido hacia el registro FIFO que tiene el doble de tamaño que el resto de los buffers, esto es para introducir un retardo a la señal que permita llevar a cabo el filtrado de los segmentos de la señal almacenados en el primer registro LIFO. La señal de salida del filtro de fase cero continúa siendo nula.

Luego de un par de ciclos de reloj, cuando el buffer LIFO se llenó por segunda vez, las llaves conmutan a su posición ‘2’. Esto produce que el filtro H1 entre en su estado de relajación con muestras nulas a su entrada y sea el filtro H2 el que procese el segundo segmento de la señal ECG. Las primeras tres muestras ‘e1’, ‘e2’ y ‘e3’ continúan su recorrido a través de los registros. La señal de salida permanece nula. Ver la figura (37) siguiendo el recorrido de color rojo.

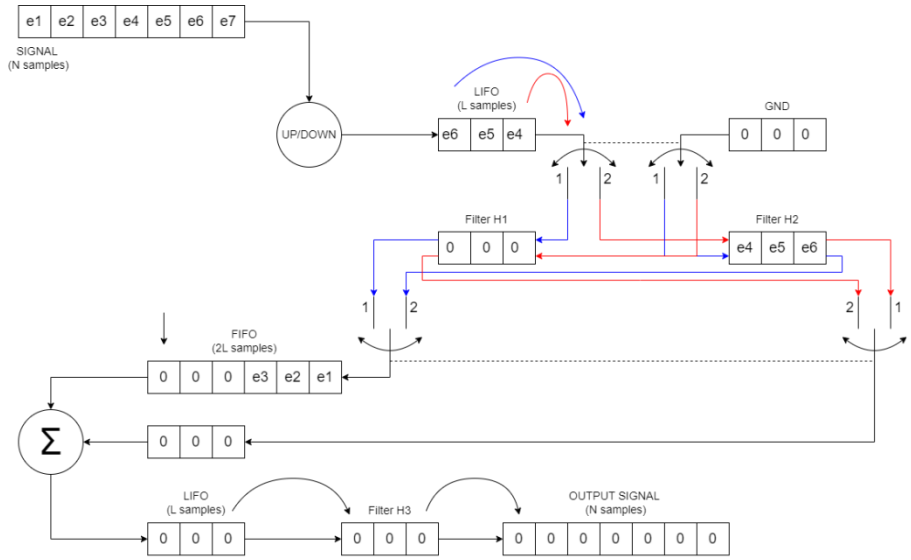


Figura (37) Segundo ciclo de procesamiento.

Este proceso se repite hasta completar el procesamiento de todas las muestras que conforman la señal ECG.

Analizando lo que ocurre a partir de la salida de los filtros H1 y H2 en adelante luego de un par de ciclos de reloj, los segmentos del primer buffer LIFO se acumulan y desplazan dentro del registro FIFO. A continuación muestra a muestra se suma el resultado de la señal de relajación del filtro correspondiente a cada ciclo con las muestras almacenadas secuencialmente en el registro FIFO. Tales resultados son almacenados en un segundo buffer LIFO. Las figuras (38) y (39) representa esta etapa del procesamiento.

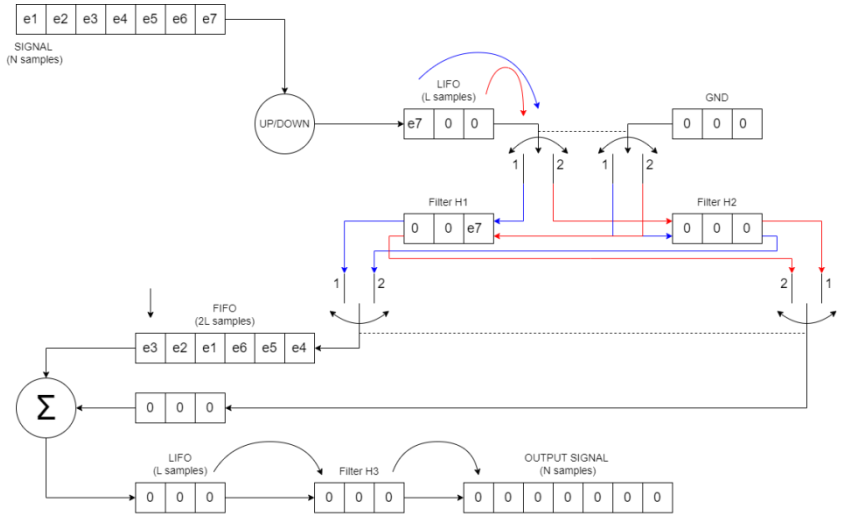


Figura (38) Funcionamiento del buffer FIFO (Parte 1).

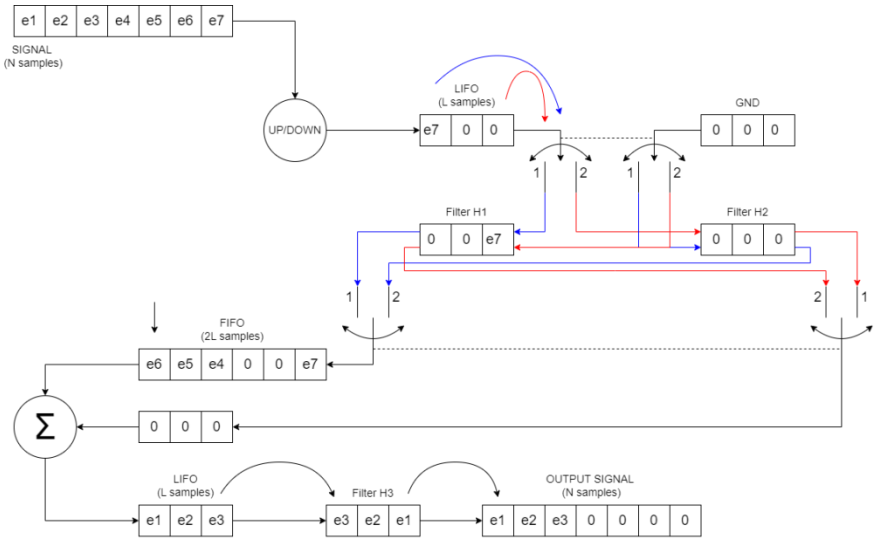


Figura (39) Funcionamiento del buffer FIFO (Parte 2).

En la figura (39), luego de un par de bloques de procesamiento, a la salida del filtro de fase cero empezamos a tener señal a la salida que corresponde al régimen transitorio.

Nota: Observar que el primer buffer LIFO invirtió la señal en el tiempo y luego, el segundo registro LIFO la invierte nuevamente regresándola a su estado original.

El filtro H3 es idéntico a los filtros H1 y H2. Esto quiere decir que cada segmento de la señal ECG que ingresa al filtro de fase cero es filtrada dos veces. La primera vez lo hace invertida en el tiempo y la segunda, en su estado inicial.

Finalmente, luego del quinto bloque y coincidiendo con lo planteado en el artículo [1] de Powell y Chau, el sistema entra en régimen permanente y a la salida ya se puede visualizar la señal filtrada. Esto se ilustra en la figura (40).

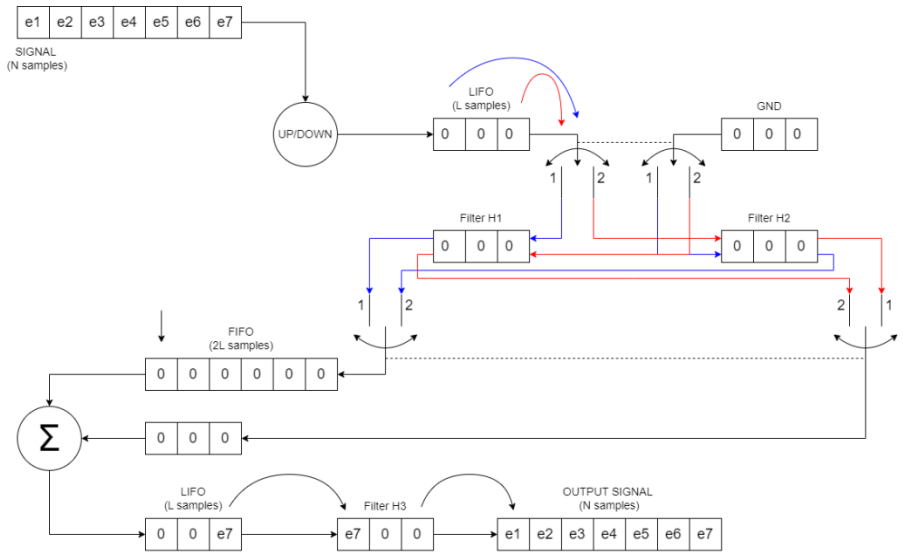


Figura (40) Señal de salida del sistema en régimen permanente.

En la figura (41) se muestra el resultado obtenido luego de ejecutar el código de programa en Matlab para la señal ECG ‘1a’. En la misma se puede apreciar los primeros ciclos de procesamiento donde la señal de salida del filtro de fase cero es nula, el régimen transitorio y posteriormente, parte de la señal ECG procesada cuando el sistema entró en régimen permanente.

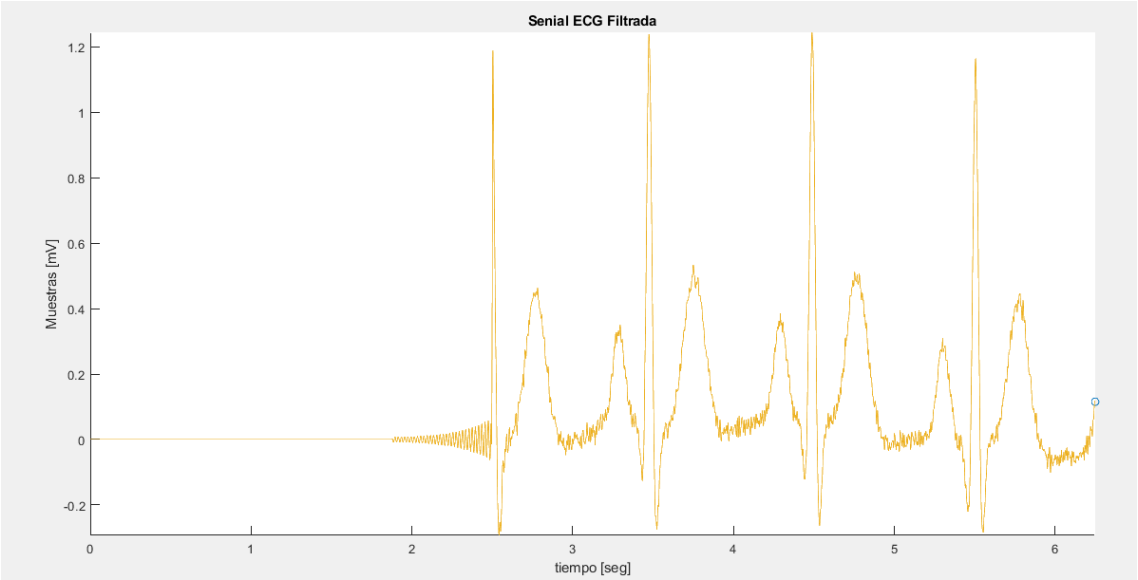


Figura (41) Visualización de la señal ECG filtrada a la salida del sistema.

4) Procesar las dos señales de entrada entregada por la cátedra.

En el siguiente apartado pusimos a prueba el filtro de fase cero diseñado utilizando las señales ECG proporcionadas por la cátedra como entradas para la función desarrollada en Matlab.

Aclaración: En este apartado solo se adjuntarán las gráficas de los resultados finales obtenidos y se detallarán las observaciones pertinentes y las conclusiones obtenidas en apartados siguientes de este informe con la intención de recopilar todo lo aprendido en este proyecto integrador en secciones dedicadas a tal fin y cumpliendo con las consignas asignadas.

Para la señal 'ECG_senal_1a.mat'

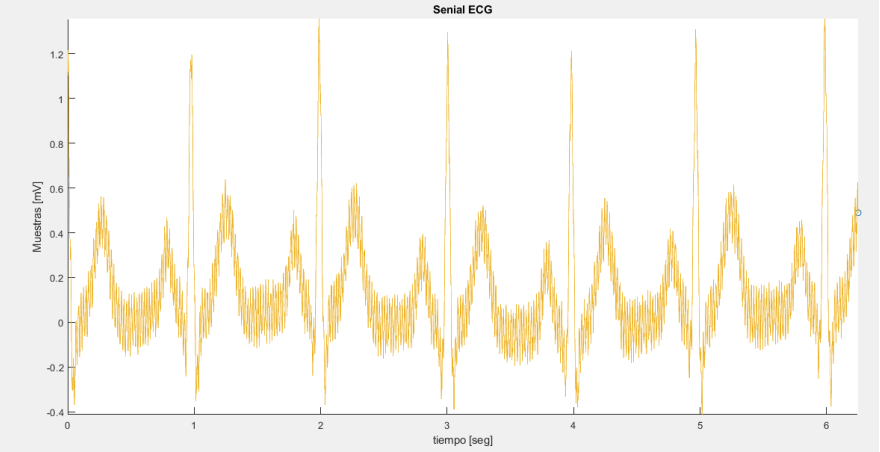


Figura (42) Señal ECG '1a' original.

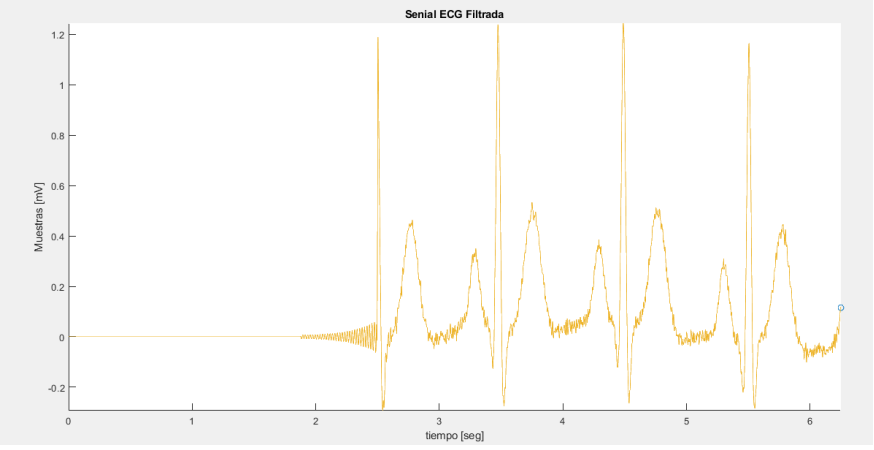


Figura (43) Señal ECG '1a' filtrada.

Para la señal ‘ECG_senal_2a.mat’

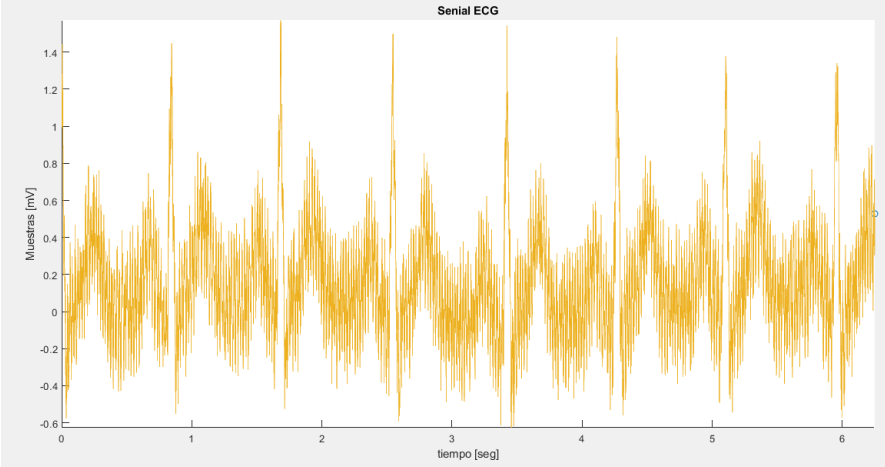


Figura (44) Señal ECG ‘2a’ original.

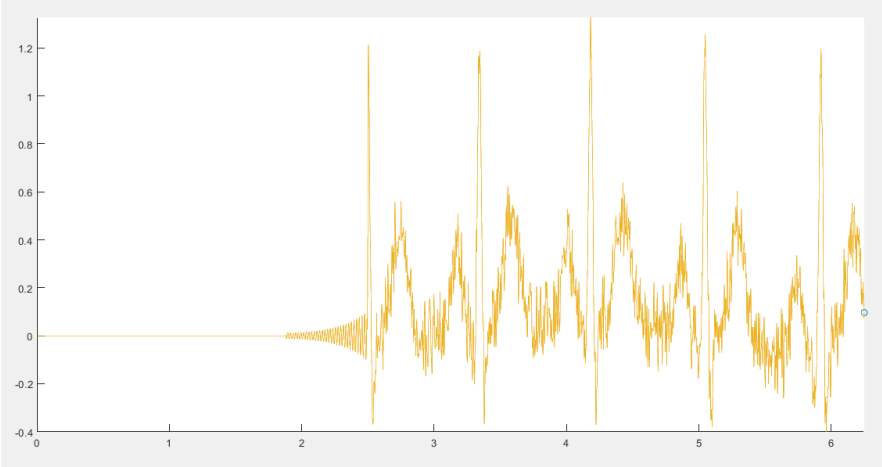


Figura (45) Señal ECG ‘2a’ filtrada.

Para la señal ‘ECG_senal_3a.mat’

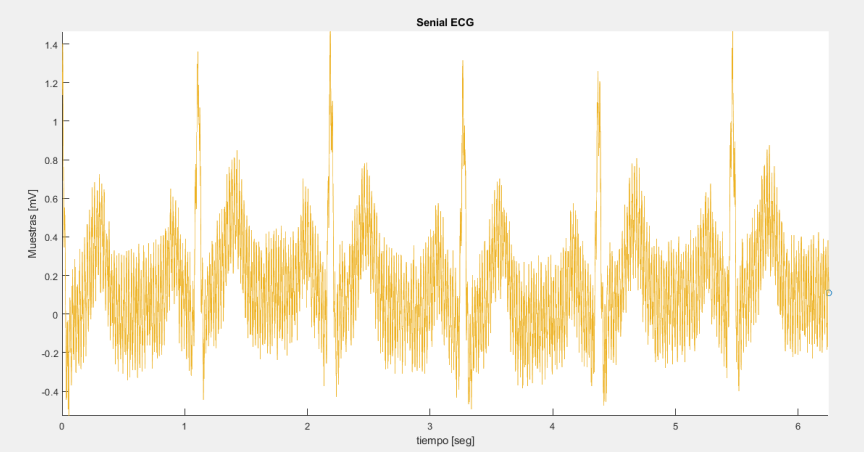


Figura (46) Señal ECG ‘3a’ original.

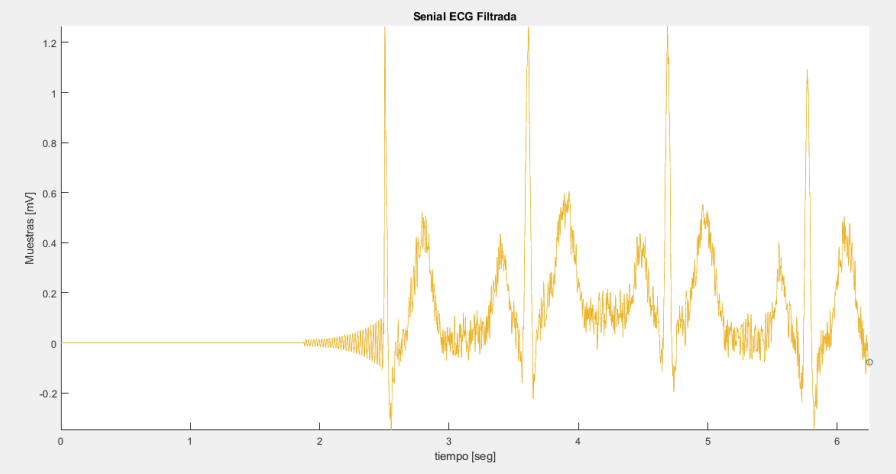


Figura (47) Señal ECG ‘3a’ filtrada.

5) Analizar las componentes de frecuencia de las señales de entrada utilizando la Transformada Rápida de Fourier.

Aclaración: De igual manera que en el apartado anterior, solo se adjuntarán las gráficas de los resultados finales obtenidos y se detallarán las observaciones pertinentes y las conclusiones obtenidas en apartados siguientes de este informe.

Para la señal 'ECG_senal_1a.mat'

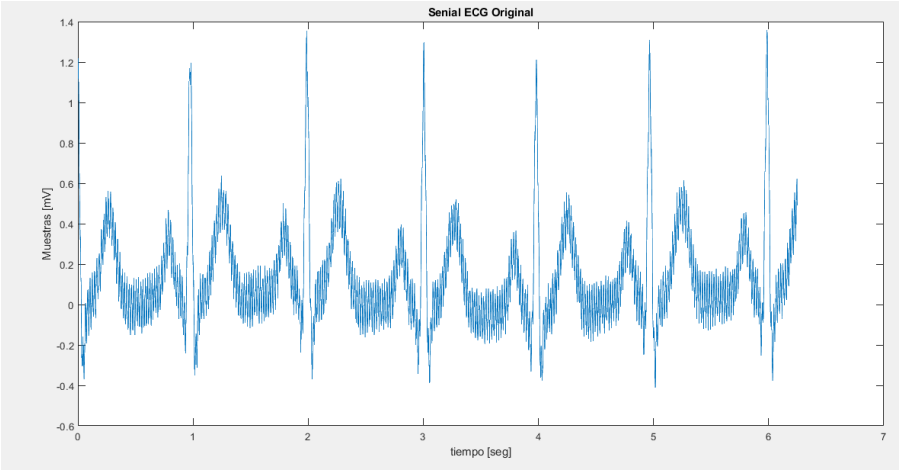


Figura (48) Señal ECG '1a' original.

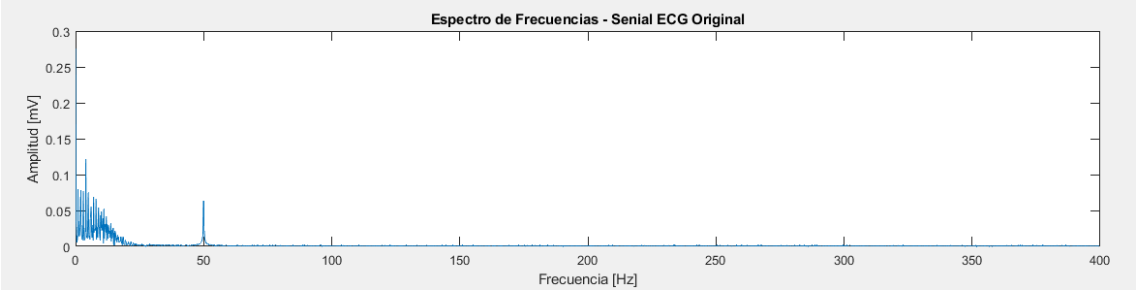


Figura (49) Espectro de frecuencia de la señal ECG '1a' original.

Para la señal ‘ECG_senal_2a.mat’

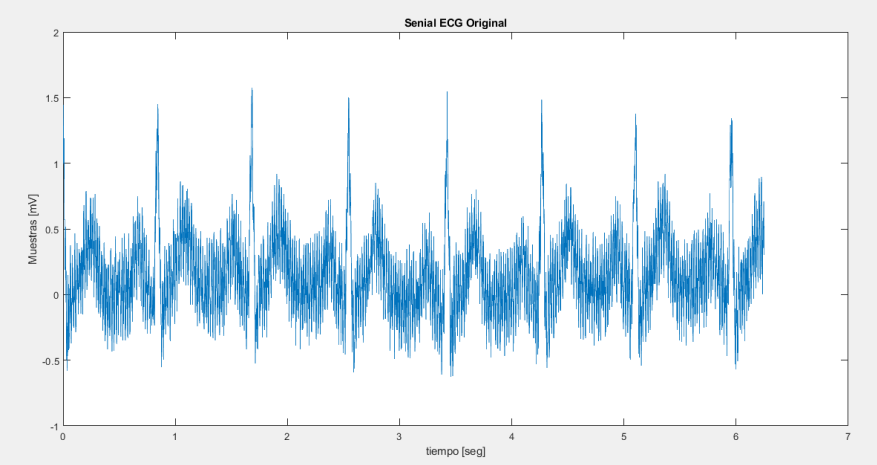


Figura (50) Señal ECG ‘2a’ original.

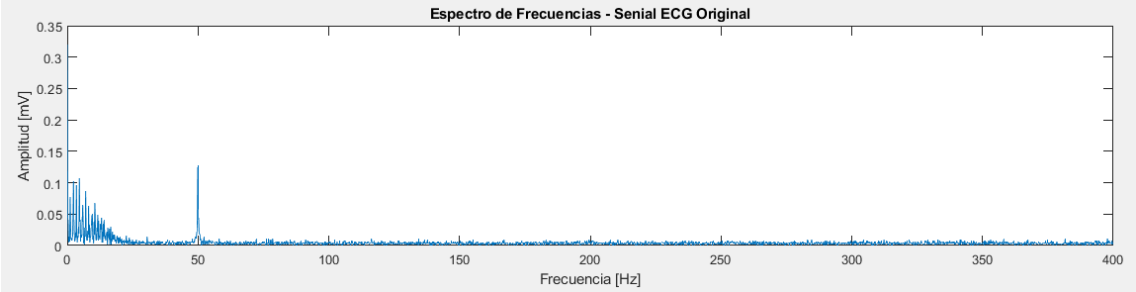


Figura (51) Espectro de frecuencia de la señal ECG ‘2a’ original.

Para la señal ‘ECG_senal_3a.mat’

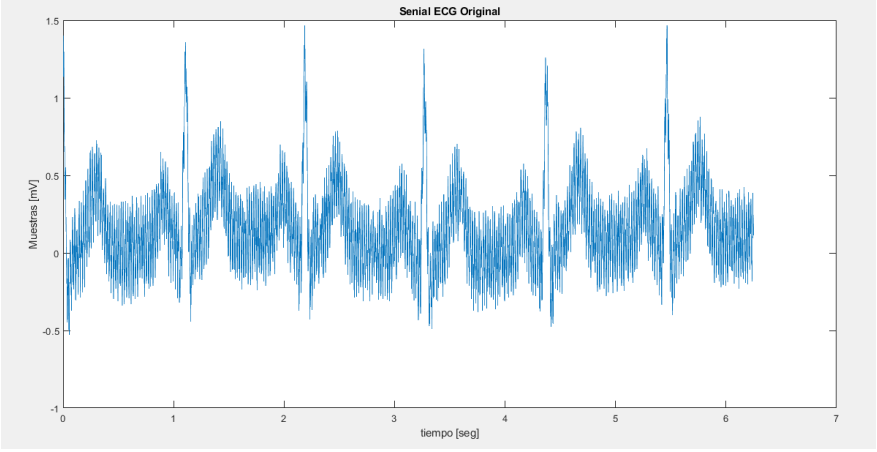


Figura (52) Señal ECG ‘3a’ original.

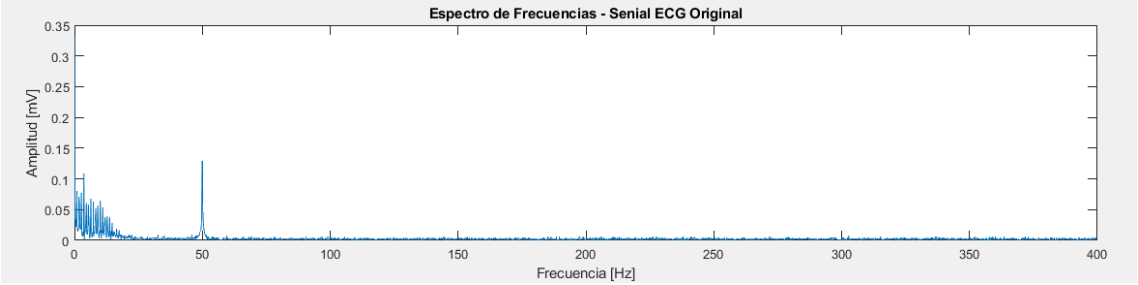


Figura (53) Espectro de frecuencia de la señal ECG ‘3a’ original.

6) Analizar las componentes de frecuencia de las señales de salida utilizando la Transformada Rápida de Fourier.

Aclaración: Siguiendo la metodología de los apartados 4) y 5), solo se adjuntarán las gráficas de los resultados finales obtenidos y se detallarán las observaciones pertinentes y las conclusiones obtenidas en apartados siguientes de este informe.

Para la señal 'ECG_senal_1a.mat'

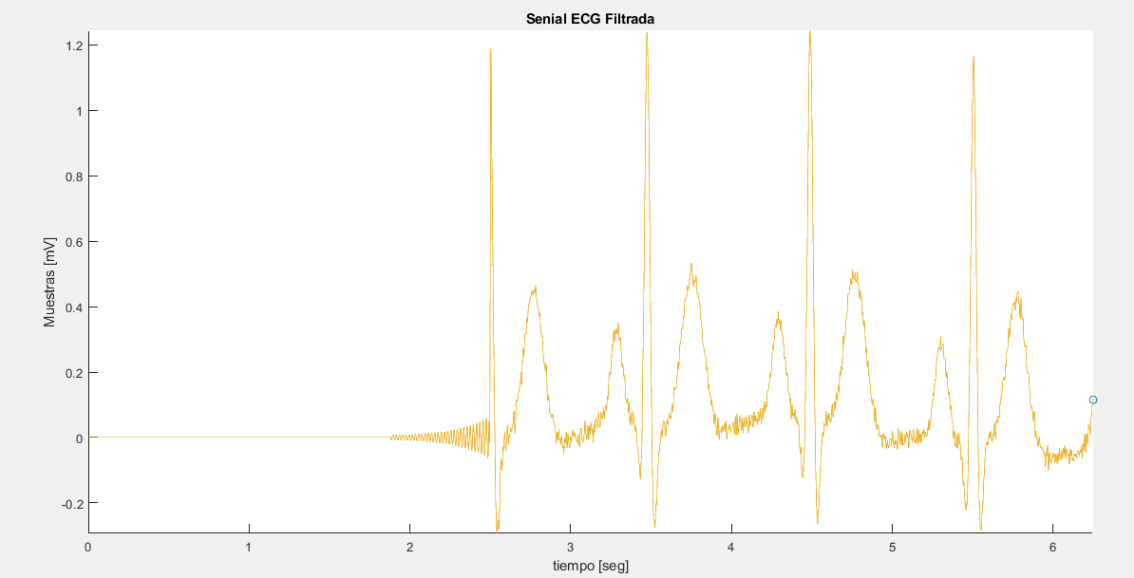


Figura (54) Señal ECG '1a' filtrada.

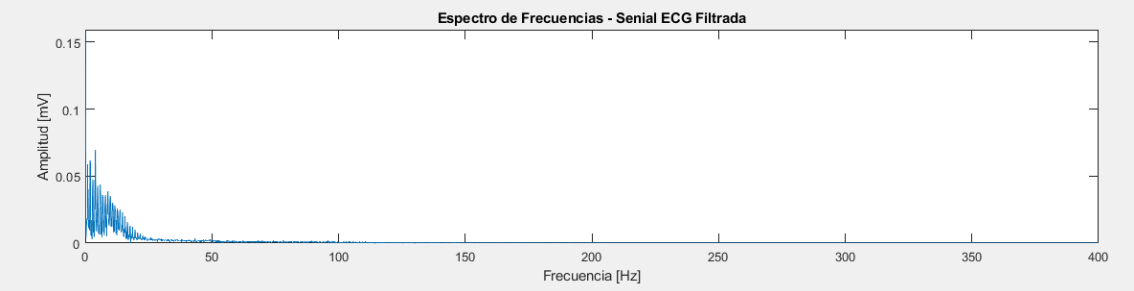


Figura (55) Espectro de frecuencia de la señal ECG '1a' filtrada.

Para la señal ‘ECG_senal_2a.mat’

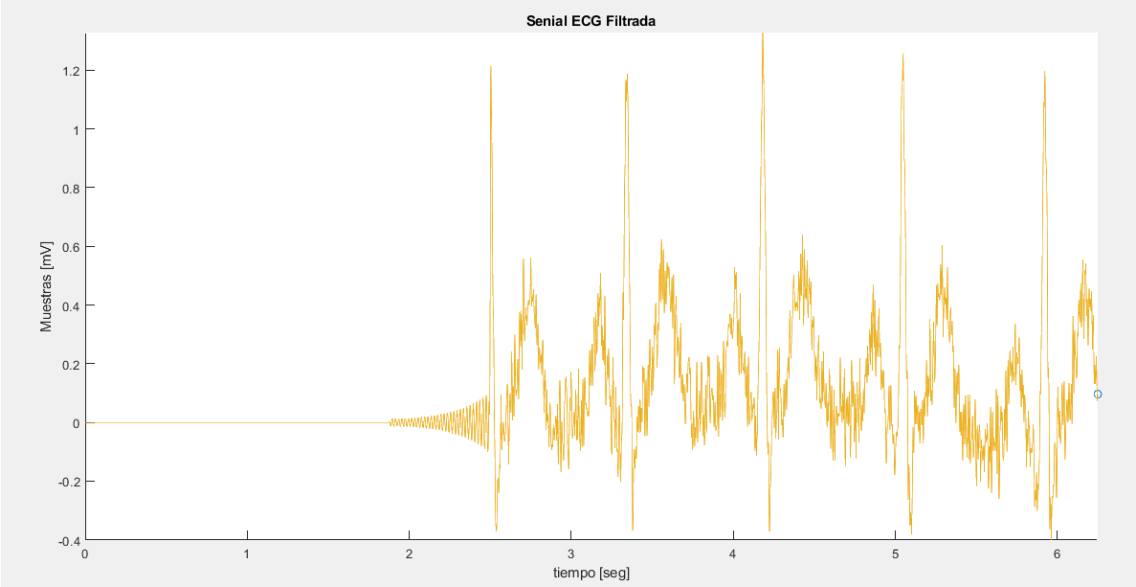


Figura (56) Señal ECG ‘2a’ filtrada.

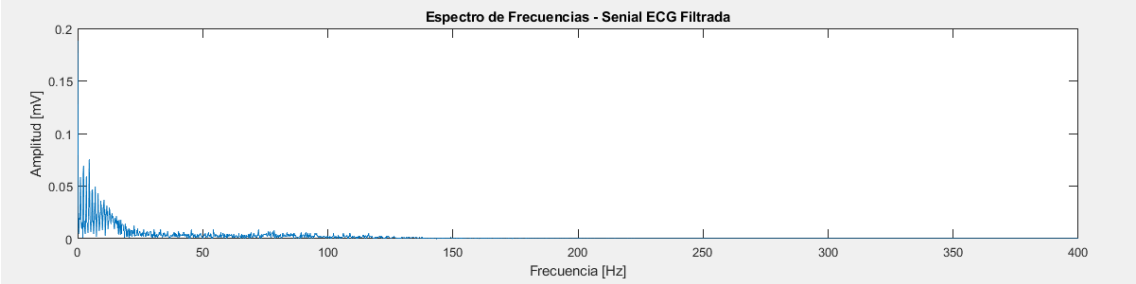


Figura (57) Espectro de frecuencia de la señal ECG ‘2a’ filtrada.

Para la señal ‘ECG_senal_3a.mat’

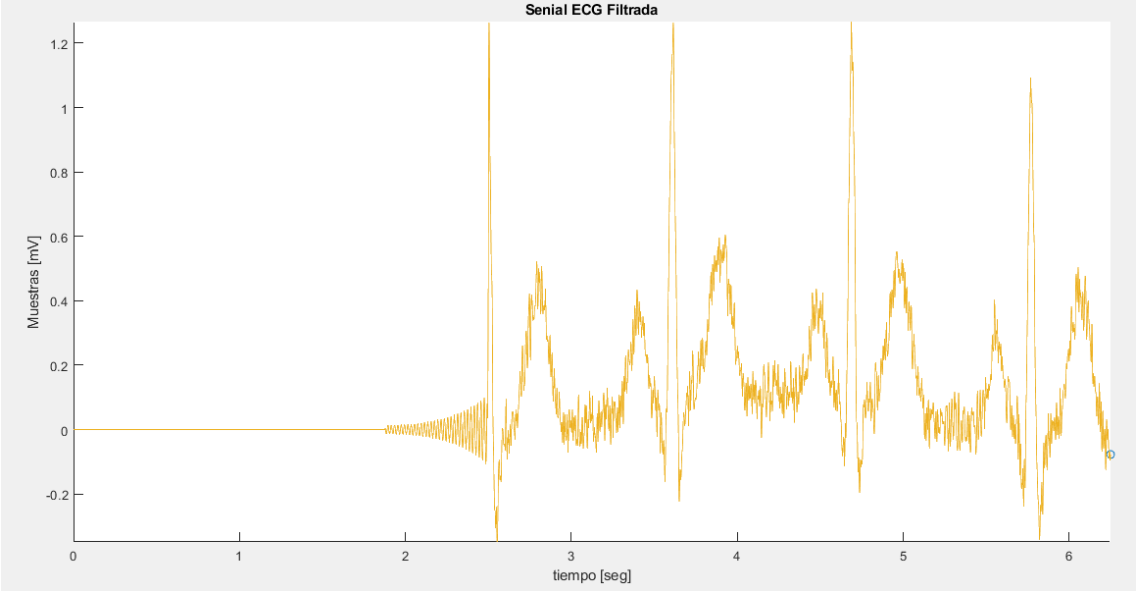


Figura (58) Señal ECG ‘3a’ filtrada.

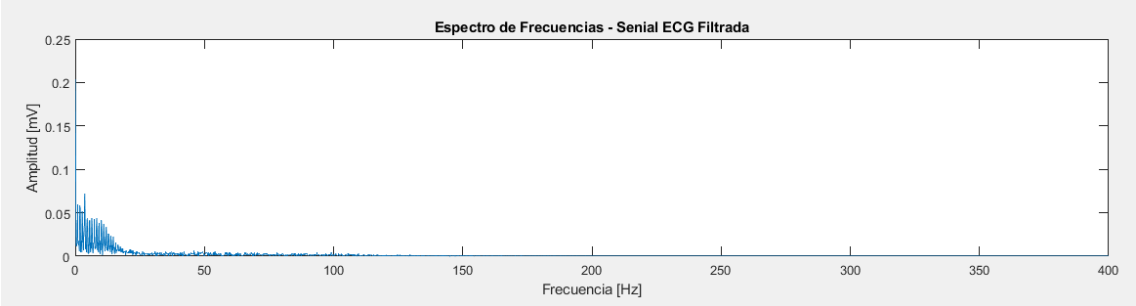


Figura (59) Espectro de frecuencia de la señal ECG ‘3a’ filtrada.

OBSERVACIONES

Cuestiones Teóricas

- Trabajamos con filtros digitales porque operamos sobre señales digitales. Es una operación matemática que toma una secuencia de números (señal de entrada) y la modifica produciendo otra secuencia de números (señal de salida) con el objetivo de resaltar o atenuar ciertas características. Pueden existir como una fórmula en un papel, un loop en un programa de computadora o como un circuito integrado en un chip. [21]
- La componente eléctrica de $50\ [Hz]$ se elimina con un filtro ranura, el cual se puede lograr con un filtro FIR (fase lineal) clásico de orden 70, 80 o 100 pero no es conveniente para nuestro caso debido a la carga computacional que se le añade al procesamiento digital. Como solución alternativa, diseñamos el filtro ranura utilizando el método directo de ubicación de polos y ceros resultando un filtro tipo IIR [22].
- Diseñamos un filtro pasa bajos del tipo IIR porque nos garantiza una fase lineal mientras los coeficientes sean simétricos. Recordemos que un filtro de fase lineal no produce distorsión en la forma de onda de la señal [22], [23].
- Un filtro FIR solo tiene numerador. Tiene una cuenta finita. Agregar el denominador es equivalente a colocar realimentación (colocamos polos) convirtiéndolo en un filtro IIR. Ahora, el problema de añadir polos y que no se encuentren en el origen es que introducen oscilaciones y lo vuelven inestable al sistema. Producen deformación, lo ideal es que estén cercanos al centro. Al ubicar los polos en un lugar diferente al origen, ya no podemos cumplir con la condición de fase lineal. Pero trabajando en conjunto con el filtro pasa bajos (orden entre 10 y 20) lograríamos la respuesta deseada.

Sobre Matlab

- Uno de los inconvenientes que se presentó durante el desarrollo del filtro de fase cero como función de Matlab fue querer implementar el diseño modular de los filtros pasa-bajos y notch. Particularmente, la forma en que se quería llevar a cabo resultaba contraproducente y generaba una carga computacional inmensa a la hora de correr el programa lo cual implicaba retardo en el procesamiento. Esto se debió a un mal diseño de algoritmo ya que existía una mala interpretación de la propuesta plasmada por Powell y Chau en su artículo científico [1]. Se solucionó estudiando con más detalle la configuración propuesta por Powell y Chau y recibiendo asesoría por parte del docente de cátedra [24], [25].
- Implementando un filtro pasa bajos de orden 3, la respuesta no es muy buena. La frecuencia de corte está desplazada hacia la derecha en un valor de $181\ [Hz]$ aproximadamente. El problema radicaba en que la señal de ECG estaba muestreada a una frecuencia de $800\ [Hz]$ mientras que el filtro fue diseñado para una frecuencia de $1000\ [Hz]$.
- Se implementó un filtro ranura con la posibilidad de probar diferentes valores de r.
- La implementación del filtro de fase cero se llevó a cabo el programa de cómputo numérico en sus versiones R2017b y R2021a. Si la capacidad de procesamiento de la computadora donde se corre el programa es limitada, se recomienda visualizar las gráficas de forma individual y no en su totalidad debido a la carga computacional que implica.

- Al aplicar la transformada de Fourier a una señal obtenemos un espectro bilateral, tanto el espectro de amplitud (simetría par) como de fase (simetría impar), que se compone de frecuencias positivas y negativas. Debido a esto, graficamos los espectros de frecuencia hasta el valor de $f_s/2$ para llevar a cabo el análisis.
- Si se desea visualizar la forma en que se actualizan los registros internos de nuestro filtro de fase cero simplemente basta con habilitar la sección del código de programa que se encuentra comentada. Ver anexo B.

Apartado 4

Señal ECG '1a'

1. Observando las gráficas de las figuras (42) y (43), podemos comparar la señal ECG '1a' de entrada con la señal ECG '1a' filtrada donde claramente notamos que el ruido aditivo queda suprimido. También se observa en la figura (43) las tres etapas del procesamiento digital: la salida nula, el régimen transitorio y el régimen permanente de la señal filtrada.

2. Se pueden distinguir sin mayores complicaciones los puntos PQRST que son de interés en las señales ECG.

Señal ECG '2a'

1. Observando las gráficas de las figuras (44) y (45), podemos comparar la señal ECG '2a' de entrada con la señal ECG '2a' filtrada donde se observa que si bien logró eliminarse un gran contenido de frecuencias no deseadas, la señal ECG '2a' filtrada todavía presenta ruido y la identificación de los puntos PQRTS no es tan clara como el caso de la señal ECG '1a'.

2. También se observa en la figura (45) las tres etapas del procesamiento digital: la salida nula, el régimen transitorio y el régimen permanente de la señal filtrada.

Señal ECG '3a'

1. Observando las gráficas de las figuras (46) y (47), podemos comparar la señal ECG '3a' de entrada con la señal ECG '3a' filtrada donde se observa claramente que el ruido aditivo queda suprimido. También se observa en la figura (43) las tres etapas del procesamiento digital: la salida nula, el régimen transitorio y el régimen permanente de la señal filtrada.

2. Respecto al ruido, comparando las gráficas de las figuras (43), (45) y (47) podemos decir que la señal ECG '3a' sería el resultado intermedio entre las señales ECG '1a' y ECG '2a'. Todavía existe una componente de ruido desagradable a la vista pero los puntos PQRTS se pueden divisar con cierta facilidad.

Apartado 5

Dada la consigna de este apartado se darán las dos observaciones alcanzadas por el equipo de trabajo respecto a las señales ECG proporcionadas por la cátedra:

1. Las tres señales ECG poseen un elevado nivel de ruido acoplado al conjunto de frecuencias de interés. Siendo la señal ECG '2a' la que presenta mayor contaminación en su espectro de frecuencia. La señal ECG '1a' es la que presenta menor cantidad de ruido. Esta apreciación puede hacerse a través de las gráficas de cada señal observando las figuras (48), (50) y (52) o por medio de sus respectivos espectros de frecuencias en las figuras (49), (51) y (53). Recordar que cualquier frecuencia por encima de los 120 [Hz] es considerada ruido.

2. Las tres señales ECG presentan una componente fuerte en 50 [Hz] introducida por la red eléctrica que alimenta al equipo electrocardiógrafo.

Apartado 6

Dada la consigna de este apartado se darán las dos observaciones alcanzadas por el equipo de trabajo respecto a las señales ECG procesadas por el filtro diseñado:

1. En el espectro de frecuencia de las tres señales ECG podemos apreciar que cualquier componente de frecuencia por encima de 120 [Hz] está fuertemente atenuada, ver figuras (55), (57) y (59).

2. En el espectro de frecuencia de las tres señales ECG podemos apreciar que la componente de 50 [Hz] introducida por la red eléctrica fue suprimida exitosamente, ver figuras (55), (57) y (59).

CONCLUSIONES

Comparación con la función 'filtfilt()' de Matlab

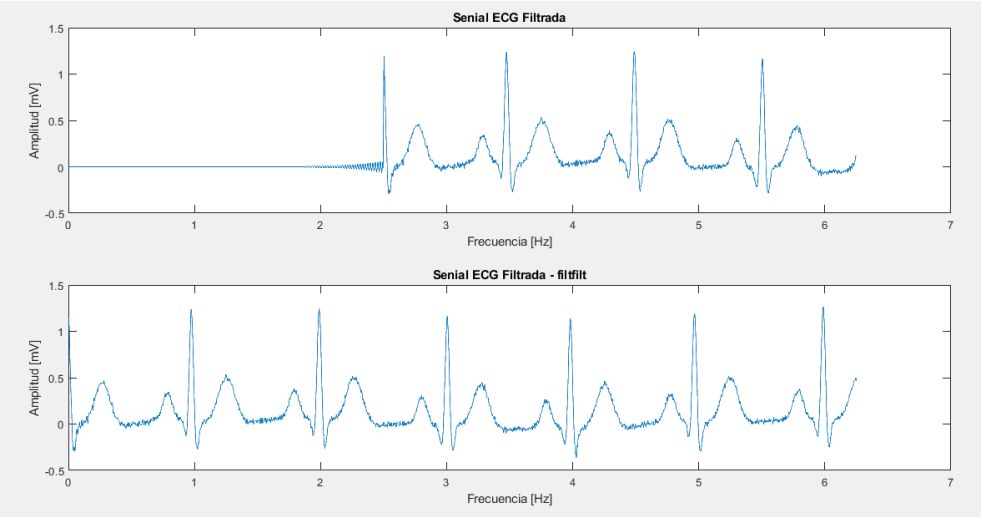


Figura (60) Señal '1a' filtrada (Filtro Fase Cero) vs. Señal '1a' filtrada (Filtfilt).

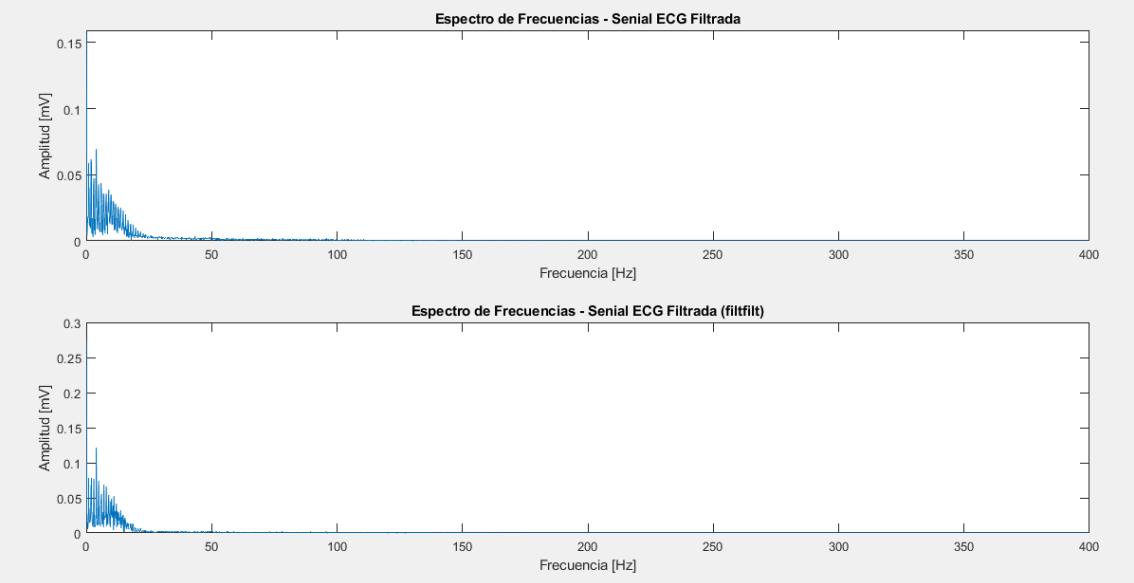


Figura (61) Espectro de frecuencia de señal '1a' filtrada (Filtro Fase Cero) vs. Espectro de frecuencia de señal '1a' filtrada (Filtfilt).

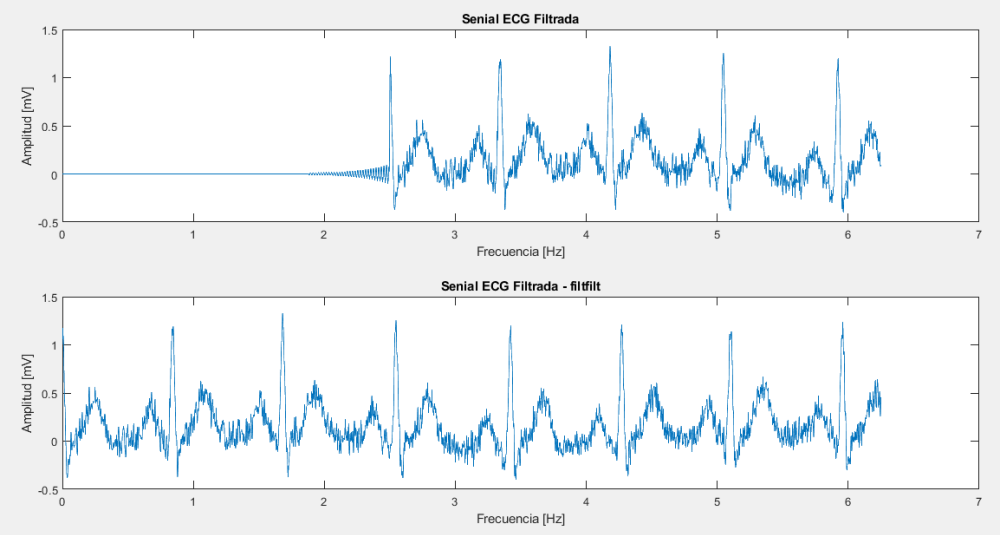


Figura (62) Señal ‘2a’ filtrada (Filtro Fase Cero) vs. Señal ‘2a’ filtrada (Filtfilt).

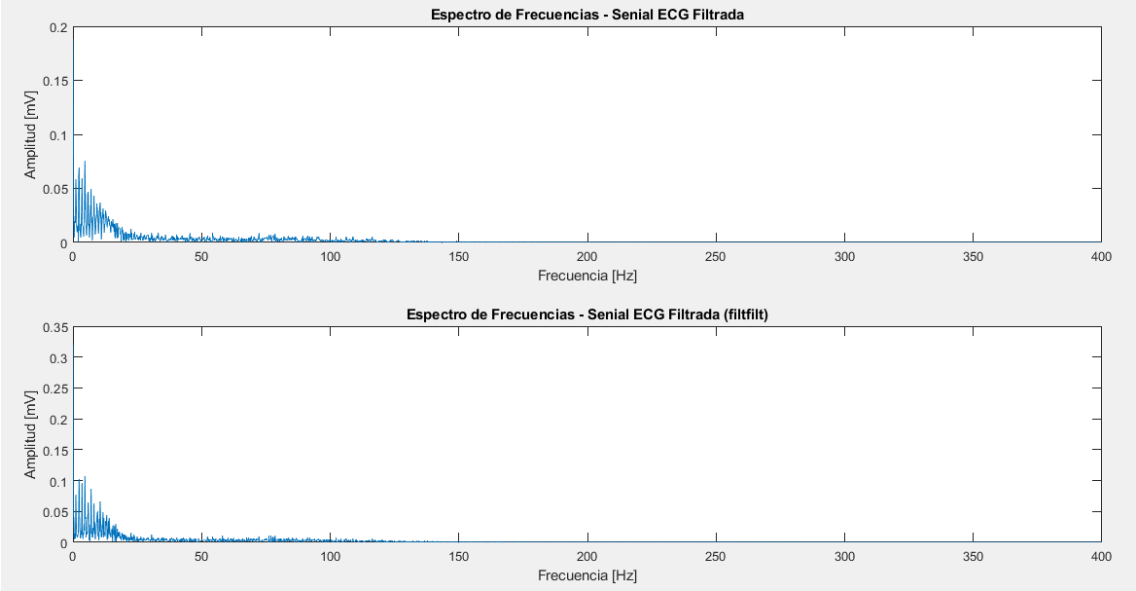


Figura (63) Espectro de frecuencia de señal ‘2a’ filtrada (Filtro Fase Cero) vs. Espectro de frecuencia de señal ‘2a’ filtrada (Filtfilt).

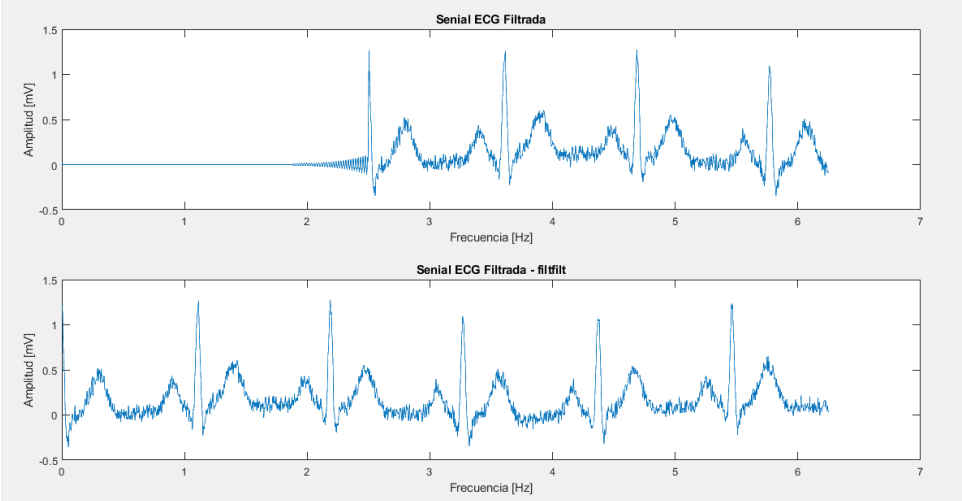


Figura (64) Señal ‘3a’ filtrada (Filtro Fase Cero) vs. Señal ‘3a’ filtrada (Filtfilt).

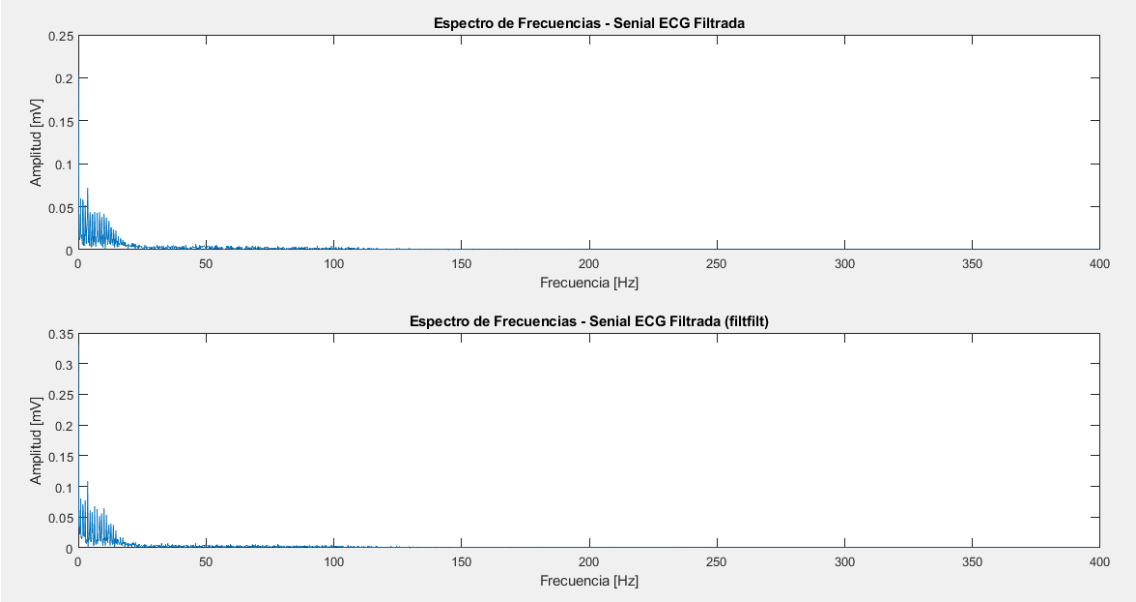


Figura (65) Espectro de frecuencia de señal ‘3a’ filtrada (Filtro Fase Cero) vs. Espectro de frecuencia de señal ‘3a’ filtrada (Filtfilt).

Una forma de evaluar el filtro de fase cero diseñado es comparándolo con la función filtfilt() de Matlab que realiza una operación similar pero no igual. La principal diferencia radica en que la función diseñada filtra muestra por muestra a la señal de entrada (un elemento del arreglo) mientras que la función que implementa Matlab toma todas las muestras de la señal (arreglo completo) para realizar el filtrado [4].

En las gráficas de las figuras (60), (62) y (64) se muestran las señales ECG filtradas con el filtro de fase cero diseñado y con la función filtfilt() de Matlab. Se observa una gran similitud entre los resultados de un proceso y otro, una vez que se estableció el régimen permanente en filtro de fase cero. Esta conclusión aplica a las tres señales ECG.

Con respecto a los espectros de frecuencia, representados en las gráficas de las figuras (61), (63) y (65), se observan diferencias sutiles entre ambas funciones pero debido al hecho que para el filtro de fase cero diseñado se contempla el régimen transitorio causado por el procesamiento de la señal. Si sólo consideramos el régimen permanente de la señal filtrada obtenemos el mismo espectro de frecuencia que para la función filtfilt() de Matlab.

Específicas de Diseño

- Consideramos que el filtro de fase lineal diseñado como aproximación del filtro de fase cero cumple los requisitos de forma satisfactoria y brinda resultados aceptables.
- Sin embargo creemos que el mismo puede mejorar en algunos aspectos como los que se enlistan a continuación:
    - Incrementar el orden del filtro pasa-bajos para lograr una respuesta más abrupta a la frecuencia de corte. Esto permitiría atenuar mejor las componentes por encima de 120 [Hz] que para la señal ECG se considera ruido.
    - Optimizar el script para reducir la carga computacional consecuencia de su ejecución en un ordenador haciendo foco sobre todo en los procesos iterativos involucrados.

Generales del Proyecto Integrador

- La realización de este proyecto integrador nos permitió demostrar, que teóricamente es posible generar un modelo de procesamiento de señales cardiacas escalable a cualquier señal de tipo biomédica, es decir que es posible analizar bajo estos criterios de modelado la temperatura corporal, el nivel de oxígeno en la sangre entre otras ondas biomédicas, lo cual permitiría generar una poderosa herramienta de simulación que estaría al servicio de la salud y la formación de personal médico asistencial [18].
- Con la culminación de este proyecto integrador logramos afianzar las competencias y habilidades adquiridas durante el cursado de la asignatura. Resultó ser un desafío interesante ya que puso a prueba nuestras habilidades para la investigación y nuestra destreza en la programación y manejo de MATLAB.
- Consideramos que fue una experiencia de aprendizaje constante, enriquecedora y sumamente satisfactoria gracias al trabajo en equipo lo cual es una capacidad muy valorada en el mercado laboral.

REFERENCIAS

Artículos Científicos

[1] Scott R. Powell and Paul M. Chau, "A Technique for Realizing Linear Phase IIR Filters", IEEE Transactions on Signal Processing, Vol. 39, No. 11, November 1991.

Libros

[2] *Digital Signal Processing: Principles, Algorithms, and Applications*. John G. Proakis & Dimitris G. Manolakis. Prentice-Hall. Third Edition. 1996.

[3] *Tratamiento de señales en tiempo discreto.* Alan V. Oppenheim & Ronald W. Schafer. Pearson. 3ª Edición. 2011.

Sitios web

[4] MathWorks. *Filtrado digital de fase cero*. [En línea] Disponible: https://la.mathworks.com/help/signal/ref/filtfilt.html

[5] MathWorks. *Power bandwidth*. [En línea] Disponible: https://la.mathworks.com/help/signal/ref/powerbw.html

[6] MathWorks. *Notch filter specification*. [En línea] Disponible: https://la.mathworks.com/help/dsp/ref/fdesign.notch.html

[7] MathWorks. *Second-order IIR notch filter*. [En línea] Disponible: https://la.mathworks.com/help/dsp/ref/iirnotch.html

[8] MathWorks. *Implementar filtros.* [En línea] Disponible: https://la.mathworks.com/help/signal/ug/filter-implementation-and-analysis_es.html

[9] Youtube. Barry Van Veen. *Zero-Phase Filtering*. [En línea] Disponible: https://www.youtube.com/watch?v=ue4ba_wXV6A

[10] Youtube. Guillermo Cijanes. *Implementación de filtros digitales (Notch) – Matlab*. [En línea] Disponible: https://www.youtube.com/watch?v=Ow7hpZDLml4

[11] Youtube. Study & Tutor. *Notch filter design in Matlab*. [En línea] Disponible: https://www.youtube.com/watch?v=ke4i9NsTOyQ

Apuntes de clases

[12] "Diseño de Filtro Digital Recursivo Ranura de Segundo Orden", Procesamiento Digital de Señales, Departamento de Eléctrica, Electrónica y Computación, Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, 2º Cuatrimestre 2021.

[13] "Implementación de un filtro de fase cero en tiempo real", Procesamiento Digital de Señales, Departamento de Eléctrica, Electrónica y Computación, Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, 2º Cuatrimestre 2021.

[14] "Implementación de un filtro digital de fase lineal en tiempo real basado en filtro recursivo", Procesamiento Digital de Señales, Departamento de Eléctrica, Electrónica y Computación, Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, 2º Cuatrimestre 2021.

[15] "Diseño de Filtro Digital Recursivo Ranura de Segundo Orden", Procesamiento Digital de Señales, Departamento de Eléctrica, Electrónica y Computación, Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, 2º Cuatrimestre 2021.

[16] "Filtro FIR", Procesamiento Digital de Señales, Departamento de Eléctrica, Electrónica y Computación, Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, 2º Cuatrimestre 2021.

[17] "Tema 3. Señales y sistemas en tiempo discreto". Open Course Ware. Departamento de Ingeniería Electrónica. Escuela Técnica Superior de Ingeniería. Universidad de Valencia.

[18] "Modelo de Procesamiento Digital de Señales Cardíacas Desarrollado en Matlab". Revista Electrónica de Estudios Telemáticos, TELEMATIQUE. Universidad Privada Dr. Rafael Belloso Chacín. 2013.

[19] "Síntesis de Filtros Digitales". Tratamiento Digital de Señales. Fernando Cruz Roldán.

[20] "Introducción a Matlab para estudiantes de ingeniería". Laboratorio de Materiales y Estructuras Aeroespaciales. Facultad de Ciencias Exactas y Tecnología. Universidad Nacional de Tucumán. 2º Cuatrimestre 2021.

[21] "Introducción a los Filtros Digitales – clase 10" Escuela Universitaria de Música. Facultad de Artes. Universidad de la República Uruguay.

Consultas con docentes

[22] Ing. J. O. Pérez (Consulta pública) Diciembre 2022.

[23] Ing. F. E. Pacheco (Consulta pública) Diciembre 2022.

[24] Ing. F. E. Pacheco (Consulta pública) Marzo 2023.

[25] Ing. F. E. Pacheco (Consulta pública) Abril 2023.

ANEXOS

Anexo A: Enunciado del proyecto integrador

Anexo B: Código de programa implementado en Matlab

INGENIERÍA ELECTRÓNICA
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ANEXO A

Enunciado del Proyecto Integrador

INGENIERÍA ELECTRÓNICA
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ANEXO A

Enunciado del Proyecto Integrador

CATEDRA PROCESAMIENTO DE SEÑALES

"PROCESAMIENTO DIGITAL DE SEÑALES" *(E7S)*

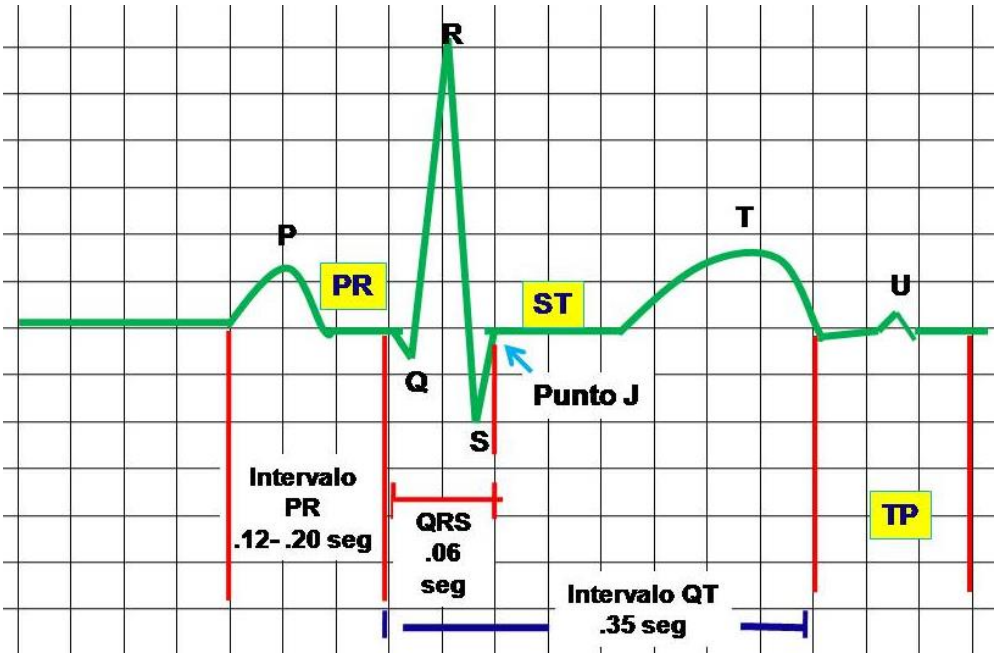
Año 2021

Proyecto Integrador

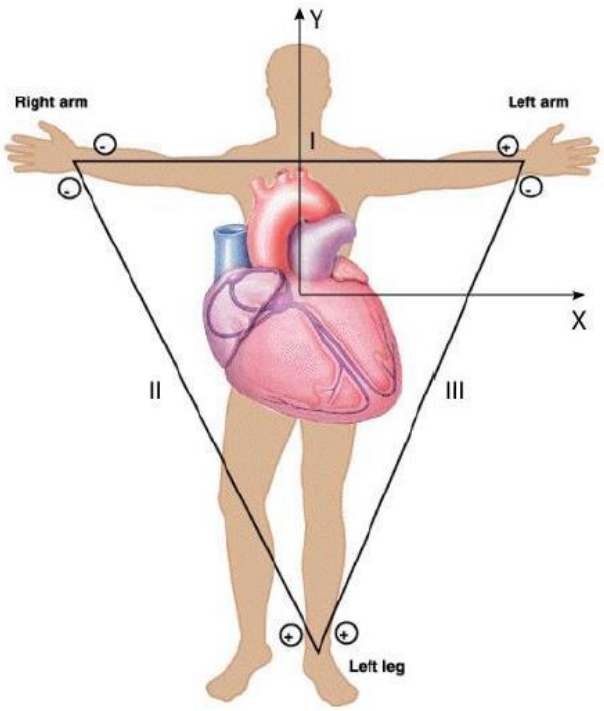
Tema: Filtro digital de fase cero

Aplicación: Procesamiento de señal de Electrocardiograma de tres vías

El electrocardiograma es un registro grafico de los potenciales eléctricos generados en el corazón durante el ciclo cardiaco. Esta representación consiste en una línea base y varias deflexiones y ondas.



Para obtener las señales cardiacas se colocan electrodos en posiciones específicas del cuerpo, los cuales transmiten las señales eléctricas que se propagan a través de los tejidos corporales desde el corazón, en un ECG de control se fijan tres electrodos.



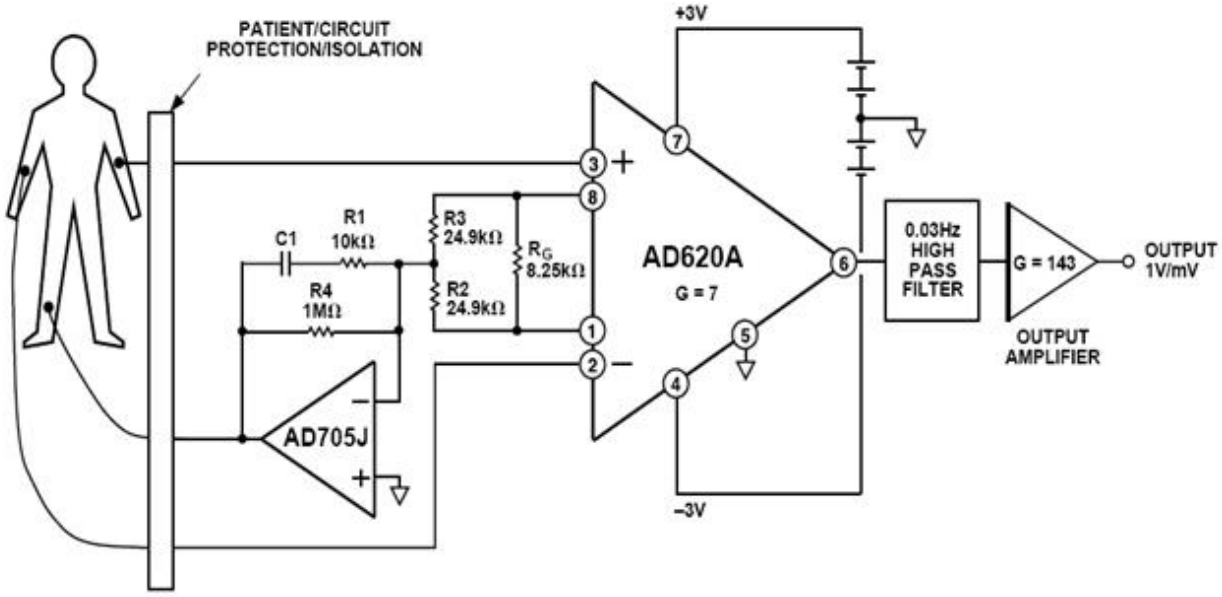
Los potenciales eléctricos son relativamente pequeños, se utiliza un amplificador diferencial para incrementar el nivel de la señal que se obtiene con los electrodos.

Se adjunta la "Reglamentación para Trabajos de Integración" **Fecha de Presentación:……**

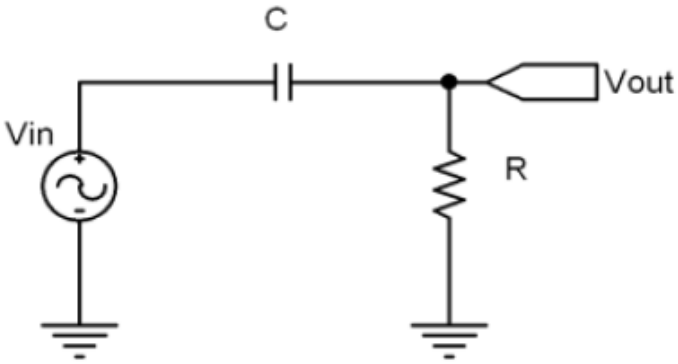
CATEDRA PROCESAMIENTO DE SEÑALES

"PROCESAMIENTO DIGITAL DE SEÑALES" ***(E7S)***

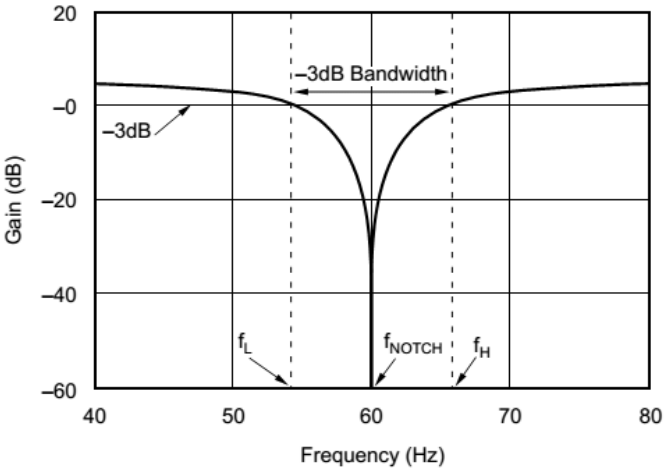
Año 2021



Existen principalmente cuatro tipos de problemas encontrados en las señales de ECG: deriva de la línea de base, interferencia de la línea eléctrica, ruido EMG (causados por actividad muscular) y los causados por movimiento de electrodos.



El efecto de la deriva de línea de base consiste en el movimiento del eje x de la señal hacia arriba y hacia abajo, el contenido de frecuencia de este movimiento está por debajo de los 0,5 Hz, en el sistema propuesto, para eliminarlo se pasa la señal por un filtro pasa altos analógico.



La interferencia de la línea eléctrica es una señal sinusoidal de 50 Hz, en el sistema propuesto se elimina con un filtro digital ranura (digital notch filter).

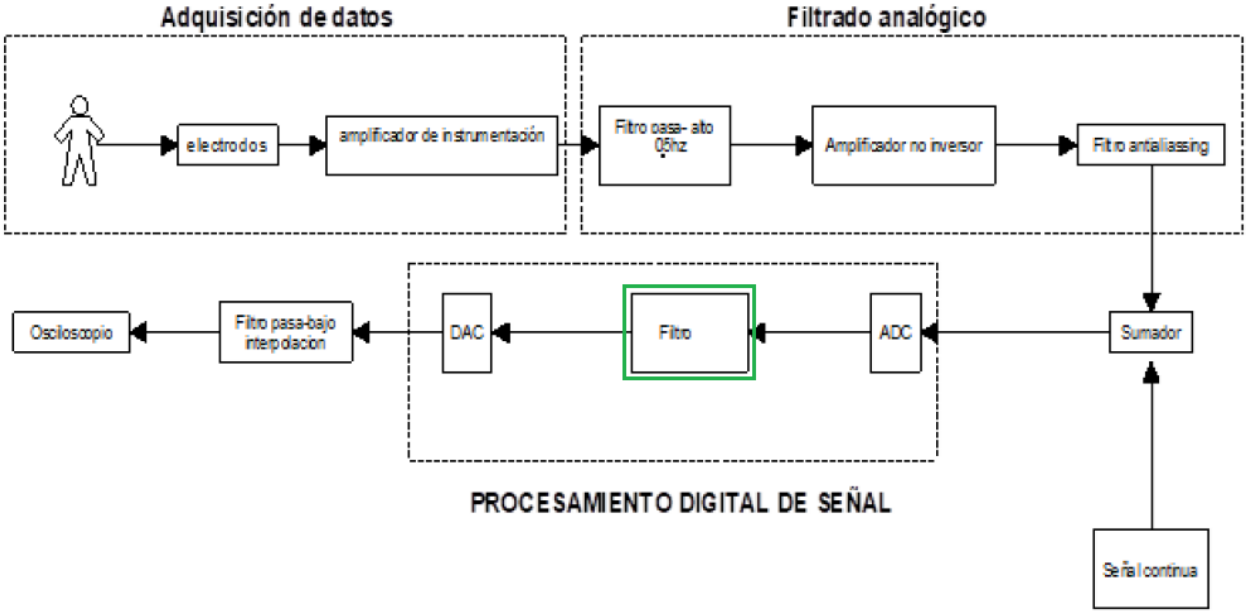
El contenido de frecuencia del ruido muscular se superpone considerablemente al del complejo PQRST, se elimina con técnicas de procesamiento digital de señales que superan el contenido de este curso.

Se adjunta la "Reglamentación para Trabajos de Integración" **Fecha de Presentación:……**

Los problemas causados por el movimiento del electrodo se asemejan a las características de la deriva de la línea de base, pero su contenido espectral que es en el rango de 1 a 10 Hz se superpone considerablemente al del complejo PQRST.

Plan de trabajo

Realizar el diseño del componente de procesamiento digital de señales del sistema propuesto que implementa el filtro pasa bajos y el filtro ranura.



Dado que el diagnóstico que se realiza utilizando la señal de ECG se basa en la forma de la señal, es importante que la repuesta de los filtros sea de fase lineal, de modo que se implementa un filtro de fase cero.

- La señal de ECG contiene valores bajos de frecuencia, se considerará de interés las componentes de frecuencia por debajo de 120 Hz.
- La frecuencia de muestreo utilizada en el sistema es de 1.000 Hz.
- El sistema se implementa utilizando un micro DSP de 16 bits.
- Se desea que el ancho de banda del filtro ranura sea lo más chica posible.
- La cátedra pasará dos señales de prueba.

1) Justificar por qué el filtro debe ser de fase lineal para no modificar la forma de la señal.
2) Diseñar el filtro ranura y el filtro pasa bajos para la implementación del filtro de fase cero.
3) Desarrollar una función en Matlab que implemente el filtrado de fase cero, debe procesar la señal muestra a muestra, es decir se llama la función pasándole el valor actual de la señal y devuelve el valor actual de la salida, actualizándose las variables o registro internos del sistema.
4) Procesar las dos señales de entrada entregada por la cátedra.
5) Analizar las componentes de frecuencia de las señales de entrada utilizando la Transformada Rápida de Fourier.
6) Analizar las componentes de frecuencia de las señales de salida utilizando la Transformada Rápida de Fourier.

CATEDRA PROCESAMIENTO DE SEÑALES

"PROCESAMIENTO DIGITAL DE SEÑALES" ***(E7S)***

Año 2021

Reglamento para realizar el Informe y la Presentación Oral

Al Proyecto Integrador comprende un informe escrito y una presentación oral, y debe ser realizado en grupo, con la participación de todos sus integrantes.

Para aprobar es obligatoria la participación del alumno en la presentación, ya que la calificación es individual.

Informe Escrito:

1. El informe debe ser escrito en formato A4 y encarpetado. Incluir el enunciado.
2. Antes de la presentación cada grupo deberá entregar un ejemplar impreso del informe escrito.
3. Contenido del informe escrito:

   Carátula (Asignatura, Grupo y Nombre de los integrantes, Año de cursado) Enunciado.

   Resumen (Explicación del caso de estudio)

   Desarrollo del tema. Detallar los procedimientos utilizados. Incluir sus apreciaciones y comentarios sobre los resultados obtenidos.

   Conclusiones

   Referencias

4. Junto al Informe entregar un CD con todo lo concerniente al Proyecto Integrador grabado (Incluir informe, programas y pruebas desarrolladas, y la presentación).

Presentación Oral:

5. Preparar una presentación oral para exponer el trabajo en forma grupal (preferentemente desarrollada en Power Point).
6. En la presentación oral deberán estar presentes y participar todos los miembros del grupo, ya que la nota final es individual.

Se adjunta la "Reglamentación para Trabajos de Integración" **Fecha de Presentación:……**

INGENIERÍA ELECTRÓNICA
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

ANEXO B

Código de programa implementado en Matlab

Zero_Phase_Filter.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     UNIVERSIDAD NACIONAL DE TUCUMAN                             %
%                FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGIA                        %
%                                                                                 %
%   Asignatura: PROCESAMIENTO DIGITAL DE SENIALES                                 %
%                                                                                 %
%   Equipo de trabajo:  ANCE, GASTON ARMANDO                                      %
%                       ARNEDO, EMILIANO                                          %
%                       DIAZ, LEONARDO LEANDRO                                    %
%                       PEREYRA, FAUSTO HORACIO                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
function[]=Zero_Phase_Filter(sample_frequency,signal)

% Limpiar la ventada de comandos
close all
clc

%%
%Inicializacion de parametros para las señales ECG
ns=0:length(signal)-1;      %Barrido de tiempo igual al numero de datos
adquiridos
ns=ns*1/sample_frequency;   %Normalizacion por T=1/Fs

figure(1)
Plotear_Signal(ns,signal);   % Gráfico de la señal ECG original (informe)
%figure(2)
%Plotear_Signal_2(ns,signal); % Gráfico de la señal ECG original
(presentacion)

%Inicializacion de otros parametros
size_impulse=100;   %Tamaño del vector de la senial Impulso
L=500;              %Tamaño de los buffers LIFO
j=1;
j2=1;
flag=1;
flag2=1;
tiempo=linspace(1,L,L);
tiempo2L=linspace(1,2*L,2*L);
fp=1;
fp2=1;
BLOQUE=1; %Revisar si se puede rescatar esto

%%
%Filtro Pasa-bajos
Fc=120;                                 %Frecuencia del filtro Pasa-bajos
N=3;                                    %Orden del filtro Pasa-bajos
[num_lp,den_lp] = Lowpass_Design(N,Fc,sample_frequency);     %Coeficientes
de la Funcion Transferencia
Lowpass_Frequency_Response(num_lp,den_lp,sample_frequency);  %Respuesta en
frecuencia del Filtro Pasa-bajos
```

```
%Otros disenios del Filtro Pasabajos
Lowpass_Filters(N,Fc,sample_frequency);          %Figura (5)
Filtro_Pasabajos_Diseniado(sample_frequency);    %Figura (6)
Lowpass_Matlab(N,Fc,sample_frequency);           %Figura (7)

%Filtro Notch
CT_notch_freq=50;                          %Frecuencia del Filtro Notch
AB=12;                                     %Ancho de banda
w_n=CT_notch_freq/(sample_frequency/2); %Frecuencia de rechaza banda
Q=35;                                      %Factor de calidad
bw=w_n/Q;
[num_n,den_n] = iirnotch(w_n,bw);          %Coeficientes

figure(8)
Notch_Frequency_Response(num_n,den_n,sample_frequency);  %Figura (8)

% Visualizamos el plano z
figure(9);
zplane(num_n,den_n);legend('zero','plot');

%Otros disenios del Filtro Notch
[num_n2,den_n2]=Notch_Design(CT_notch_freq,sample_frequency,AB);
figure(10)
Notch_Frequency_Response(num_n2,den_n2,sample_frequency);
% Visualizamos el plano z
figure(11);
zplane(num_n2,den_n2);legend('zero','plot');

%Conexion en cascada
FTlp = tf(num_lp,den_lp);    %Generamos la funcion transferencia del filtro
pasa bajo
FTn = tf(num_n,den_n);       %Generamos la funcion transferencia del filtro
notch

FTcascada=FTlp*FTn;          %Funcion de transferencia del sistema total
[num,den] = tfdata(FTcascada,'v');
%freqz(num,den)              %Descomentar para ver respuesta en frecuencia

%NOTA: La funcion tfdata genera los vectores que contienen los coeficientes
%del numerador y denominador de la funcion transferencia

%Determinamos el tamanio de los buffers
[L]=Calcule_buffer_size(size_impulse,num_n,den_n,num_lp,den_lp,sample_freque
ncy);  %Figura (12)

%Generamos los registros
buffer_LIFO1=zeros(1,L);     %buffer LIFO 1
buffer_LIFO2=zeros(1,L);     %buffer LIFO 2
buffer_FIFO=zeros(1,2*L);    %buffer FIFO
```

```
signal_salida=zeros(1,length(signal));
salida_LIFO1=0;
salida_H1=0;
salida_H2=0;
salida_FIFO=0;
salida_LIFO2=0;
SUMA=0;
salida_H3=0;

%Condiciones iniciales nulas
z1=0;
z2=0;
z3=0;

for i=1:length(signal)

[salida_LIFO1,buffer_LIFO1,j,flag,BLOQUE]=LIFO(i,j,flag,buffer_LIFO1,signal,
BLOQUE);
    if(flag==1)
        [salida_H1,z1]=Filtrar(num,den,0,z1);
        [salida_H2,z2]=Filtrar(num,den,salida_LIFO1,z2);
        [salida_FIFO,buffer_FIFO]=FIFO(L,i,buffer_FIFO,salida_H2);
        SUMA=salida_FIFO+salida_H1;
    else
        [salida_H1,z1]=Filtrar(num,den,salida_LIFO1,z1);
        [salida_H2,z2]=Filtrar(num,den,0,z2);
        [salida_FIFO,buffer_FIFO]=FIFO(L,i,buffer_FIFO,salida_H1);
        SUMA=salida_FIFO+salida_H2;
    end
    [salida_LIFO2,buffer_LIFO2,j2,flag2]=LIFO2(j2,flag2,buffer_LIFO2,SUMA);
    [salida_H3,z3]=Filtrar(num,den,salida_LIFO2,z3);
    signal_salida(i)=salida_H3;

%%
%     if i<=L
%         fp=i;
%     else
%         fp=1;
%     end
%
%      if i<=2*L
%         fp2=i;
%     else
%         fp2=1;
%     end
%
%     figure(BLOQUE)
%
%     subplot(2,2,1)
%     plot(tiempo(1:fp),buffer_LIFO1(1:fp))
%     title('LIFO1')
%     axis([1 L -0.6 1.5])
%
%     subplot(2,2,2)
%     plot(tiempo2L(1:fp2),buffer_FIFO(1:fp2))
%     title('FIFO')
%     axis([1 2*L -0.6 1.5])
```

```
%
%      subplot(2,2,3)
%      plot(tiempo(1:fp),buffer_LIFO2(1:fp))
%      title('LIFO2')
%      axis([1 L -0.6 1.5])
%
%      pause(0.00001)
%
%      subplot(2,2,4)
%      plot(tiempo(1:fp),signal_salida(i-(L-1):i))
%      title('SALIDA')
%
%    if (j>L || j<1) && BLOQUE<10
%      figure('Name',['BLOQUE ' num2str(BLOQUE)],'NumberTitle','off')
%      subplot(2,2,1)
%      plot(tiempo,buffer_LIFO1)
%      title('LIFO1')
%      axis([1 L -0.6 1.5])
%
%      subplot(2,2,2)
%      plot(tiempo2L,buffer_FIFO)
%      title('FIFO')
%      axis([1 2*L -0.6 1.5])
%
%      subplot(2,2,3)
%      plot(tiempo,buffer_LIFO2)
%      title('LIFO2')
%      axis([1 L -0.6 1.5])
%
%      subplot(2,2,4)
%      plot(tiempo,signal_salida(i-(L-1):i))
%      title('SALIDA')
%      axis([1 L -0.6 1.5])
%
%      pause(1)
%    end

%NOTA: Mediante el algoritmo comentado en la parte superior de esta nota
%se puede ir visualizando como se actualizan los buffers de una manera
%didactica.
end

figure(13)
Plotear_Senial_3(ns,signal_salida); %Gráfico de la señal ECG filtrada
(informe)
%Plotear_Signal_2(ns,signal_salida); %Gráfico de la señal ECG filtrada
(presentacion)

%Analisis del espectro de frecuencia

%Senial ECG Original
u_out2=fft(signal);
f_out2=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
P_out2=2.*(abs(u_out2(1:length(signal)/2)/length(signal)));

%Senial ECG Filtrada
u_out1=fft(signal_salida);
f_out1=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
```

```
P_out1=2.*(abs(u_out1(1:length(signal)/2)/length(signal)));

figure(14)
subplot(2,1,1);
plot(f_out2,P_out2);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Original');
subplot(2,1,2);
plot(f_out1,P_out1);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada');

%Comparamos los resultados obtenidos con la funcion 'filtfilt' de Matlab
signal_salida_filtfilt = filtfilt(num,den,signal);

figure(15)
subplot(2,1,1);
plot(ns,signal_salida);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Senial ECG Filtrada');
subplot(2,1,2);
plot(ns,signal_salida_filtfilt);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Senial ECG Filtrada - filtfilt');

u_out3=fft(signal_salida_filtfilt);
f_out3=sample_frequency.*(0:(length(signal)/2)-1)./length(signal);
P_out3=2.*(abs(u_out3(1:length(signal)/2)/length(signal)));

figure(16)
subplot(2,1,1);
plot(f_out1,P_out1);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada');
subplot(2,1,2);
plot(f_out3,P_out3);
hold on
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [mV]')
title('Espectro de Frecuencias - Senial ECG Filtrada (filtfilt)');
```

Plotear_Signal.m

```
function[]=Plotear_Senial(axis_x,axis_y)

%Visualizamos la senial ECG
comet(axis_x,axis_y,0);
title('Senial ECG')
xlabel('tiempo [seg]')
ylabel('Muestras [mV]')
hold on

end
```

Plotear_Signal_2.m

```
function[]=Plotear_Signal_2(axis_x,axis_y)

title('Senial ECG')
xlabel('tiempo [seg]')
ylabel('Muestras [mV]')
Fs=800;
umbral_y=6*mean(abs(axis_y));
umbral_x=0.3*Fs;

%Find peaks
[pks,locs]=findpeaks(axis_y,'MinPeakHeight',umbral_y,'MinPeakDistance',umbra
l_x);

a=tic;

h=animatedline('Color','red','Linewidth',2);

%Tones for each signal levels
A=440;
FF=5000;
T=0:1/FF:0.3;
X=sin(2*pi*A*T);
T2=0:1/FF:20;
X2=sin(2*pi*A*T2);

if(axis_y(1)==0 && axis_y(2)==0 && axis_y(3)==0 )
    sound(X2)
end
flag=0;
for k= 1:length(axis_y)
    if axis_y(k) ~= 0 && flag==0
        clear sound;
        flag=1;
    end
    if ismember(k,locs)
       sound(X)
    else
       addpoints(h,axis_x(k),axis_y(k));
    if k<=600
        axis([0,0.7488,-0.4,1.4])
    else
        axis([axis_x(k-600),axis_x(k),-0.4,1.4])
    end
    b=toc(a);
    if b>(1/15)
        drawnow
        a=tic;
    end
    end
end
drawnow
end
```

Plotear_Senial_3.m

```
function[]=Plotear_Senial_3(axis_x,axis_y)

%Visualizamos la senial ECG
comet(axis_x,axis_y,0);
title('Senial ECG Filtrada')
xlabel('tiempo [seg]')
ylabel('Muestras [mV]')
hold on

end
```

Notch_Design.m

```
function[num,den]=Notch_Design(Fc_analog,sample_freq,AB)

% Calculo de la Frecuencia de corte digital
DT_notch_freq_zero=2*pi*Fc_analog/sample_freq;

% Calculo del radio para un ancho de banda AB
r=1-(AB*pi/sample_freq);

DT_notch_freq_pole=acos(((1+r.^2)*cos(DT_notch_freq_zero))/(2*r));
notchzeros = [exp(1i*DT_notch_freq_zero) exp(-1i*DT_notch_freq_zero)];
notchpoles = [r*exp(1i*DT_notch_freq_pole) r*exp(-1i*DT_notch_freq_pole)];

b=poly(notchzeros);
K=(1+r.^2)/2;
num=K*b;
den=poly(notchpoles);

end
```

Notch_Frequency_Response.m

```
function[]=Notch_Frequency_Response(num,den,sample_freq)

% Visualizamos la respuesta en frecuencia
[h,freq] = freqz(num,den);
%freqz(num,den)          %Descomentar para ver respuesta en frecuencia
plot(freq*sample_freq/(2*pi),20*log(abs(h)));
xlabel('Frequencia [Hz]');
ylabel('Amplitud [dB]');
title('Filtro Notch - Respuesta en frecuencia [dB]');

end
```

Lowpass_Filters.m

```
function[]=Lowpass_Filters(order,Fc_analog,sample_frequency)
wc=(2*pi*Fc_analog)./sample_frequency; % Frecuencia de corte digital

% Funcion de sistema h(n) para una respuesta FIR
n=-(order-1)/2:(order-1)/2;
hpb=(sin(wc*n))./(pi*n);

% Termino central del Filtro FIR Pasa-bajos
hpb((order+1)/2)=wc/pi;

% Visualizamos la respuesta en frecuencia
[hfpb,w]=freqz(hpb,1); % En este caso, filtro FIR con numerador b=1
%freqz(hpb,1)
figure(5)
hold on
f=w/2*pi;
plot(f*sample_frequency/max(f),abs(hfpb)/abs(hfpb(1)))
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [veces]')
title('Filtro Pasa Bajos - Ventana Rectangular')
end
```

Lowpass_Matlab.m

```
function[]=Lowpass_Matlab(order,Fc_analog,sample_frequency)
% Calculo de la Frecuencia de Muestreo
Ws=2*pi*sample_frequency;
% Calculo de la Frecuencia de Corte Digital
Wc=2*pi*Fc_analog;

% Genera ventana Hamming (ancho filtro FIR)
Wrect=ones(order+1,1);
hpbmatlab = fir1(order,Wc/(Ws/2),Wrect); %Funcion de Matlab

% Visualizamos la respuesta en frecuencia
[hpbmatlab,w1]=freqz(hpbmatlab,1);
%freqz(hpbmatlab,1)          %Descomentar para ver respuesta en frecuencia
f=w1/2*pi;
figure(7)
hold on
plot(f*sample_frequency/max(f),abs(hpbmatlab),'r')
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [veces]')
title('Filtro Pasa Bajos - Matlab')
end
```

Filtro_Pasabajos_Diseniado.m

```
function[]=Filtro_Pasabajos_Diseniado(sample_frequency)
N=[0.259 0.518 0.259];      %Numerador
D=[1.979 -0.481 -0.461];    %Denominador

% Visualizamos la respuesta en frecuencia
[h,w]=freqz(N,D);
%freqz(N,D)                 %Descomentar para ver respuesta en frecuencia
figure(6)
hold on
plot(w*sample_frequency/pi,abs(h)/max(abs(h)))
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [veces]')
title('Filtro Pasa Bajos Diseniado')
end
```

Lowpass_Design.m

```
function[num,den]=Lowpass_Design(order,Fc_analog,sample_freq)

% Calculo de la Frecuencia de corte digital
wn=Fc_analog/(sample_freq/2);

[num,den] = butter(order,wn);

end
```

Lowpass_Frequency_Response.m

```
function[]=Lowpass_Frequency_Response(num,den,sample_freq)

[h,w]=freqz(num,den);
%freqz(num,den)      %Descomentar para graficar la respuesta en frecuencia

% Visualizamos la respuesta en frecuencia
figure(3);
hold on
plot(w*sample_freq/(2*pi),abs(h));
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [veces]')
title('Filtro Pasa Bajos - Respuesta en frecuencia');

figure(4)
hold on
f=w/2*pi;
plot(f*sample_freq/max(f),20*log10(abs(h)/abs(h(1))))
xlabel('Frequencia [Hz]');
ylabel('Amplitud [dB]');
title('Filtro Pasa Bajos - Respuesta en frecuencia [dB]')

end
```

Calculate_buffer_size.m

```
function[buffer_size]=Calcule_buffer_size(size_impulse,num1,den1,num2,den2,s
ample_freq)

impulse=zeros(1,size_impulse);
impulse(1)=1;

y1=filter(num1,den1,impulse);
y2=filter(num2,den2,y1);

n1=0:length(y2)-1;
% Afectamos por el valor del muestreo T=1/Fs
n1=n1*1/sample_freq;
figure(12)
hold on
stem(n1,y2)
xlabel('Frecuencia [Hz]')
ylabel('Amplitud [veces]')
title('Respuesta al impulso')

% index=1;
%
% while(abs(y2(index))>=1/(1000000.^2) && index<=length(y2))
%     index=index+1;
% end
%
% buffer_size=index
buffer_size=500;
end
```

LIFO.m

```
function[out,buffer_LIFO,j,flag,BLOQUE]=LIFO(i,j,flag,buffer_LIFO,signal,BLO
QUE)
   if(j<1)
        flag=1;
        j=j+1;                    %j=1
        BLOQUE=BLOQUE+1;

    end
    if(j>length(buffer_LIFO))
        flag=0;
        j=j-1;                    %j=L
        BLOQUE=BLOQUE+1;
    end
    if(flag==1)                   %Ascendente
        out=buffer_LIFO(j);
        buffer_LIFO(j)=signal(i);
        j=j+1;
    end
    if(flag==0)                   %Descendente
        out=buffer_LIFO(j);
        buffer_LIFO(j)=signal(i);
        j=j-1;
    end
end
```

LIFO2.m

```
function[out,buffer_LIFO2,j2,flag2]=LIFO2(j2,flag2,buffer_LIFO2,in)
   if(j2<1)
        flag2=1;
        j2=j2+1;                        %j=1
   end
    if(j2>length(buffer_LIFO2))
        flag2=0;
        j2=j2-1;                        %j=L
    end
    if(flag2==1)                        %Ascendente
        out=buffer_LIFO2(j2);
        buffer_LIFO2(j2)=in;
        j2=j2+1;
    end
    if(flag2==0)                        %Descendente
        out=buffer_LIFO2(j2);
        buffer_LIFO2(j2)=in;
        j2=j2-1;
    end
end
```

FIFO.m

```
function[out,buffer_FIFO]=FIFO(L,i,buffer_FIFO,in)
   out=buffer_FIFO(2*L);
   for j=0:L
      buffer_FIFO(2*L-j)=buffer_FIFO(2*L-(j+1));
   end
   for j=1:L-1
      buffer_FIFO(L-(j-1))=buffer_FIFO(L-j);
   end
  buffer_FIFO(1)=in;
end
```

FILTRADO.m

```
function[out_sample,z]=FILTRADO(num,den,in_sample,z)
    out_sample=num(1)*in_sample+z(1);
    for i=2:n
        z(i-1)=num(i)*in_sample+z(i)-den(i)*out_sample;
    end
end
```

Filtrar.m

```
function[muestra_salida,z]=Filtrar(b,a,muestra_entrada,z)
    n=length(a);
    z(n)=0;

%Normalizamos los polinomios numerador y denominador
    b=b/a(1);
    a=a/a(1);

    muestra_salida=b(1)*muestra_entrada+z(1);
    for i=2:n
      z(i-1)=b(i)*muestra_entrada+z(i)-a(i)*muestra_salida;
    end
    z=z(1:n - 1);
end
```