Procesamiento Digital De Señales

Alumno: Pose, Fernando Ezequiel

Legajo: 143.791-4

Ciclo lectivo: 2° Cuatrimestre 2015

Profesor: Dr. Ing. Mariano Llamedo Soria

Introducción

En muchas aplicaciones prácticas de tratamiento digital de señales, hay que enfrentarse con el problema de cambiar la tasa o frecuencia de muestreo de una señal, aumentándola o disminuyéndola en cierta cantidad. Existe un requisito para procesar las diversas señales a diferentes frecuencias asociadas a los correspondientes anchos de banda de las señales. El proceso de convertir una señal de una tasa dada a una tasa diferente se denomina *conversión de la tasa* o *frecuencia de muestreo*. A su vez, los sistemas que emplean múltiples tasas de muestreo en el procesamiento de señales digitales se conocen como *sistemas de tratamiento digital de señales de tasa múltiple*.

En este trabajo práctico se describe y realiza la conversión de la tasa o frecuencia de muestreo en el dominio digital de la señal ECG utilizada en el trabajo práctico de la materia: Procesamiento Digital de Señales, materia correspondiente al departamento de electrónica de la Universidad Tecnológica Nacional – Facultad Regional Buenos Aires.

Existen dos métodos de conversión de la tasa de muestreo de una señal digital.

- El primer método consiste en pasar la señal digital a través de un convertidor D/A, filtrarla si fuera necesario y luego volver a muestrear la señal analógica resultante a la frecuencia deseada.
- El segundo método consiste en realizar toda la conversión de la frecuencia de muestreo en el dominio digital.

En este trabajo práctico se aborda el segundo método.

Trabajo práctico 6 - Ejercicio 1

Implementar las funciones que realicen la interpolación por L, el diezmado por M y el cambio de frecuencia
de muestreo de una señal. Probar su funcionamiento con una de las señales de ECG provistas en el TP4, cambiando su frecuencia de muestreo a:

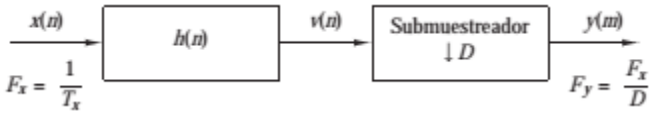
- fs1 = 250 Hz
- fs1 = 2000 Hz
- fs1 = 1100 Hz
- fs1 = 180 Hz
- fs1 = 999 Hz

Recordar que la fs del ECG es de 1000Hz. Para cada situación mostrar un segmento de 2 segundos del principio, parte media y final de cada registro. Comparar las 5 situaciones evidenciando la densisdad de muestras en cada señal.

Introducción

Diezmado por un factor D

Supongamos que la señal x(n) con espectro X(w) va a submuestrearse por un factor entero D. Se supone que el espectro X(w) es distinto de cero en el intervalo de frecuencias 0-PI. Es conocido que si reducimos la frecuencia de muestreo seleccionando simplemente uno de cada D valores de x(n), la señal resultante será una versión con aliasing de x(n). Para evitar el aliasing, en primer lugar se debe reducir el ancho de banda y luego se podrá submuestrear por D y evitar el aliasing.

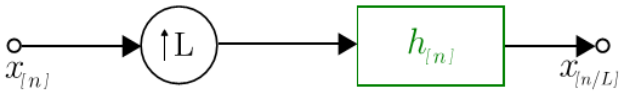


A continuación se desarrolla la función de diezmado.

```
%**
% \fn [Signal_Diez] = func_diez(signal, M)
% \brief Elimina M-1 muestras entre valores sucesivos de la señal Signal
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal     - Señal a decimar
%   \param  I          - Muestras a eliminar
% \return Signal_Int - Señal diezmada

function [Signal_Diez] = func_diez(signal, M)
    if M == 1
        Signal_Diez = signal;
    else
        Signal_Diez = signal(1:M:end);
    end
end
```

Interpolación por un factor I

Un incremento en la frecuencia de muestreo por un factor entero I puede conseguirse interpolando I-1 nuevas muestras entre valores sucesivos de la señal. El proceso de interpolación puede llevarse a cabo de diversas formas. Sea v(m) una secuencia con una frecuencia Fy = I Fx, la cual se obtiene a partir de x(n) añadiendo I-1 ceros entre valores sucesivos de x(n)



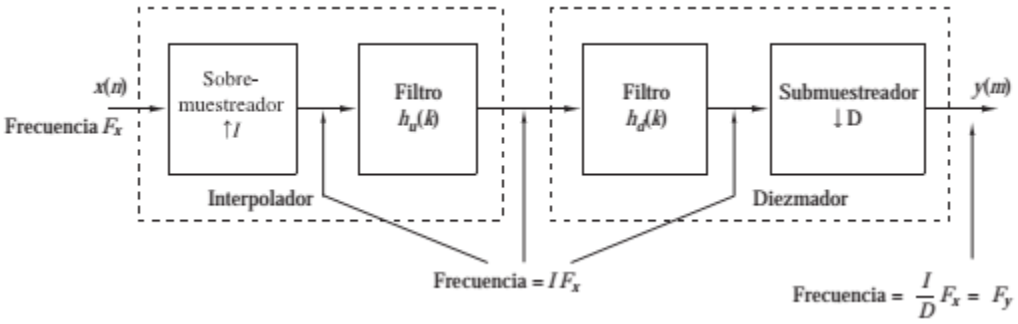
A continuación se desarrolla la función de interpolación.

```
%**
% \fn [Signal_Int] = func_interpol(Signal,I)
% \brief Agrega I-1 muestras entre valores sucesivos de la señal Signal
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal     - Señal a interpolar
%   \param  I          - Muestras a ingresar por muestra de Signal
% \return Signal_Int - Señal interpolada

function [Signal_Int] = func_interpol(Signal,I)

    if I == 1
        Signal_Int = Signal;
        return;
    else
        Signal_Int = zeros(1,length(Signal)*I);
        Signal_Int(1:I:end) = Signal;
    end
end
```

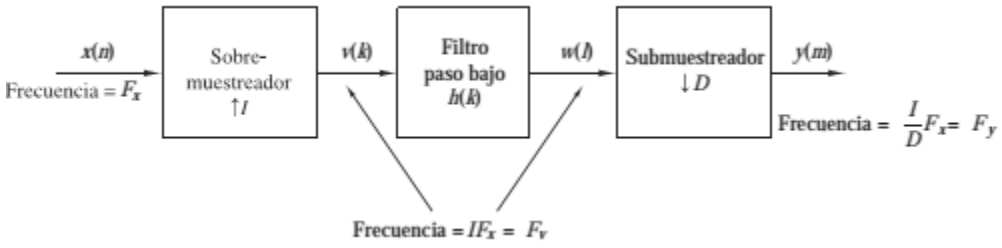
Conversión de la frecuencia de muestreo por un factor racional I/D

Dada una introducción a diezmado y interpolación, se considera ahora el caso general de conversión de la frecuencia de muestreo por un factor racional I/D. Básicamente, se puede conseguir esta conversión de la frecuencia de muestreo interpolando en primer lugar por el factor I y luego submuestreando la salida del interpolador por el factor D. En otras palabras, una conversión de la frecuencia de muestreo por un factor racional I/D se consigue conectando en cascada un interpolador y un diezmador, como se ilustra a continuación:

Método para llevar a cabo la conversión de la frecuencia de muestreo por un factor I/D:



Método para la conversión de la frecuencia de muestreo por un factor I/D:



A continuación se desarrolla la función de Resampleo.

```
%**
% \fn [Signal_Resample] = Resample1(Signal,fs,fsx,tol)
% \brief Función que resamplea (ejercicio: 1)
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal a a trabajar
%   \param  fs              - Frecuencia de la señal de entrada
%   \param  fsx             - Frecuencia nueva de sampleo
% \param tol              - Tolerancia de aproximacion
%   \return Signal_Resample - Señal resampleada

function [Signal_Resample] = Resample1(Signal,fs,fsx,tol)

    Coef_Rel = fsx/fs;

    % [N,D] = rat(X,tol) returns N./D approximating X to within tol.
    [I,M] = rat(Coef_Rel, tol * Coef_Rel);
    CoefMax = max(I,M);

    % Para el informe ver: Cap.11 - Pag.681 - Proakis.

    % Realizo el upsample
    out = func_interpol(Signal,I);

    % Realizo el filtrado
    SignalFilter = I * filterLp(out,1/CoefMax);

    % Realizo el downsample
    out = func_diez(SignalFilter, M);

    Signal_Resample = out;

end
```

Observación: En el código se puede observar que primero se realiza la interpolación (upsample) y luego el diezmado (downsample) esto se realiza de modo tal de ahorrar un filtrado al realizar ambas funciones.

A continuación se desarrolla la función del filtro pasa bajos.

```
%**
% \fn [SignalFilter] = filterLp(Signal,fc)
% \brief Filtro pasabajos
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal       - Señal a filtrar
%   \param  fc           - Frecuencia de corte del filtro pasa bajos
% \return SignalFilter - Señal filtrada

function [SignalFilter] = filterLp(Signal,fc)

    Fwpass = fc*0.9;          % Passband Frequency
    Fwstop = fc*1.1;          % Stopband Frequency
    Dpass = 0.001;            % Passband Attenuation
    Dstop = 80;               % Stopband Attenuation

    if fc == 1
        SignalFilter = Signal;
        return;

    else

    % Diseño del filtro pasa bajos
    d=fdesign.lowpass('Fp,Fst,Ap,Ast',Fwpass,Fwstop,Dpass,Dstop);
    Hd = design(d);

    SignalFilter = filtfilt(Hd.Numerator,1,Signal);

    end
end
```

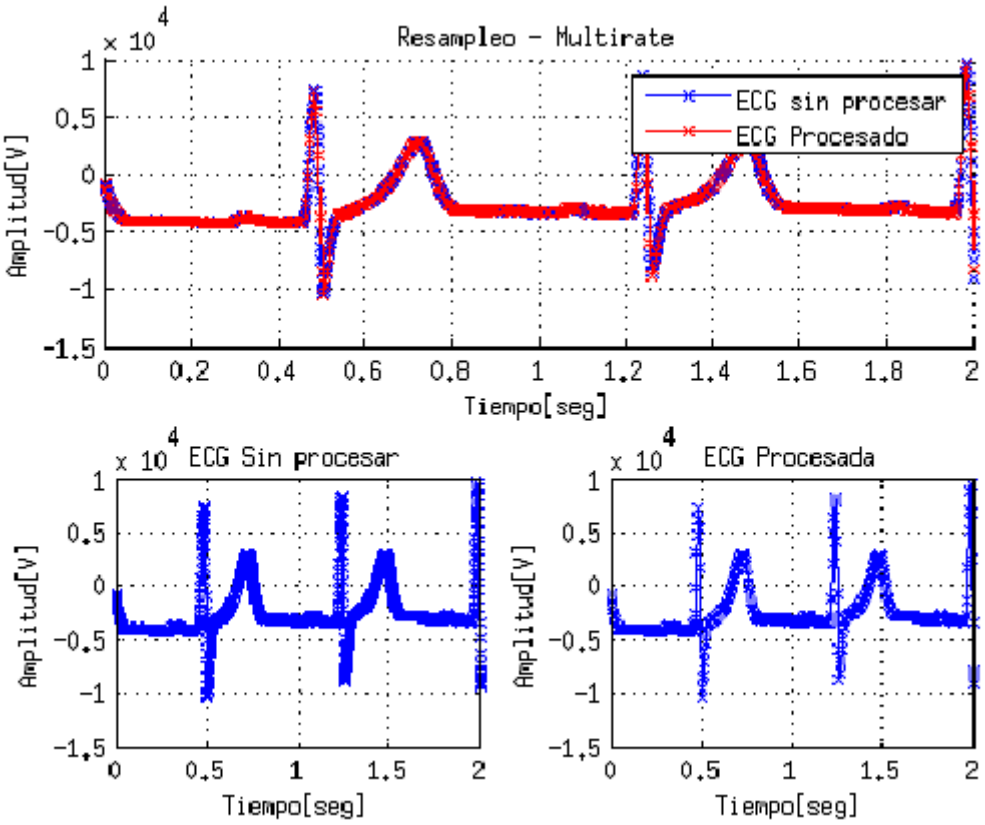
Observación: Para realizar el filtro se utilizan las peores condiciones (respecto al diezmado y la interpolación). Como en ambos casos las atenuaciones en la banda de paso y banda de atenuación, las especificaciones respecto a la amplitud son los mismos se utiliza un solo filtro donde la frecuencia de corte esta limitada por el mayor del cociente entre el numerador y denominador (I y M)

A continuación utilizando la señal ECG incluida con el informe se realiza los cambios de frecuencia dados como consigna.
Para evidenciar los cambios de frecuencia de muestreo se exponen tres partes de la grabación. Estas partes son: El inicio, el medio y fin de la grabación.

Finalmente se desarrollan las conclusiones pertinentes a los gráficos expuestos y junto a la exposición de una tabla de tiempos de operación.

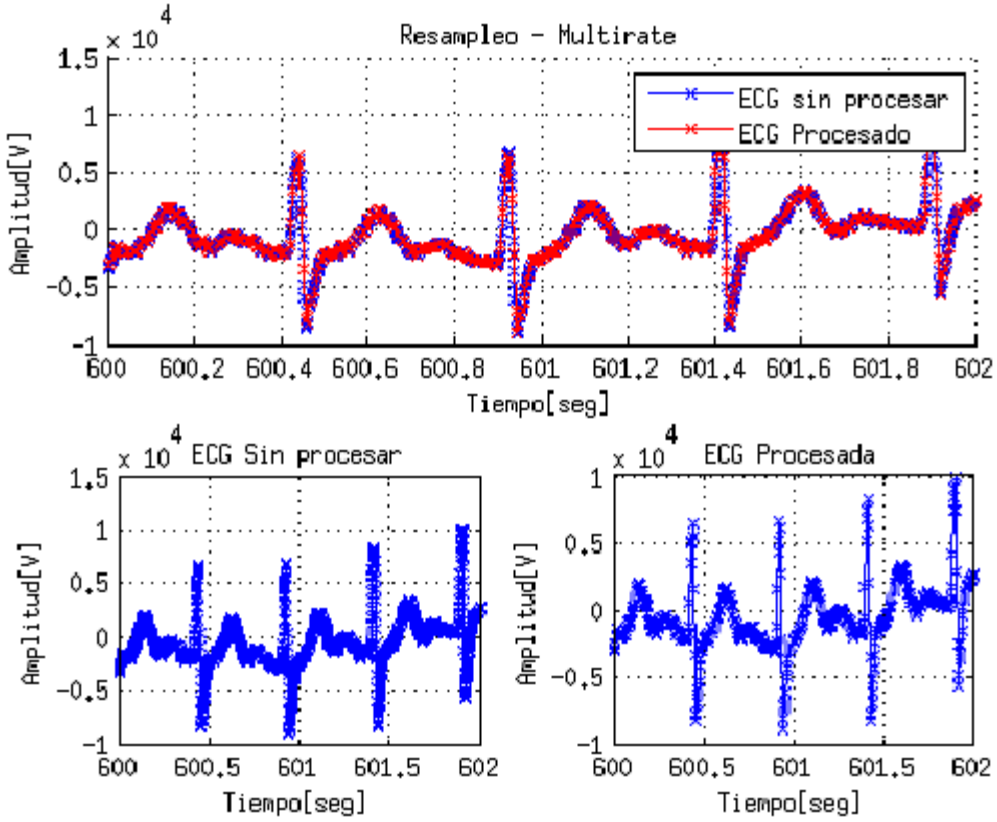
Señal de Fs: 1000Hz re-muestreada con Fs = 250Hz

Inicio de la grabación



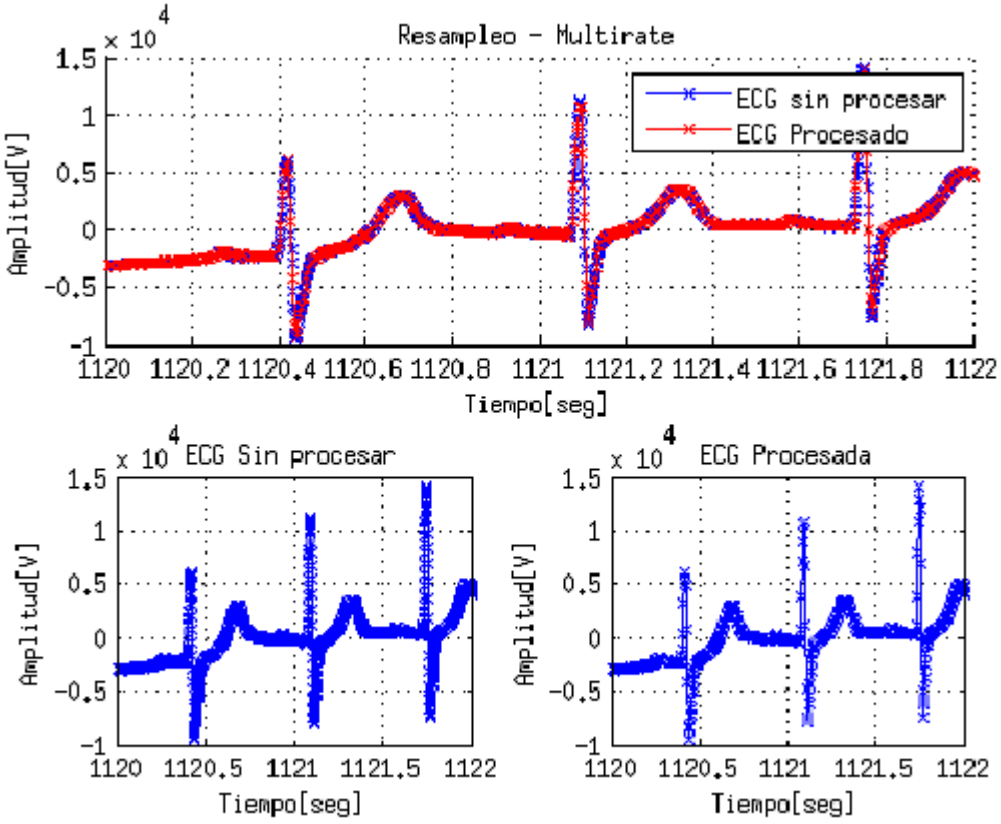
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



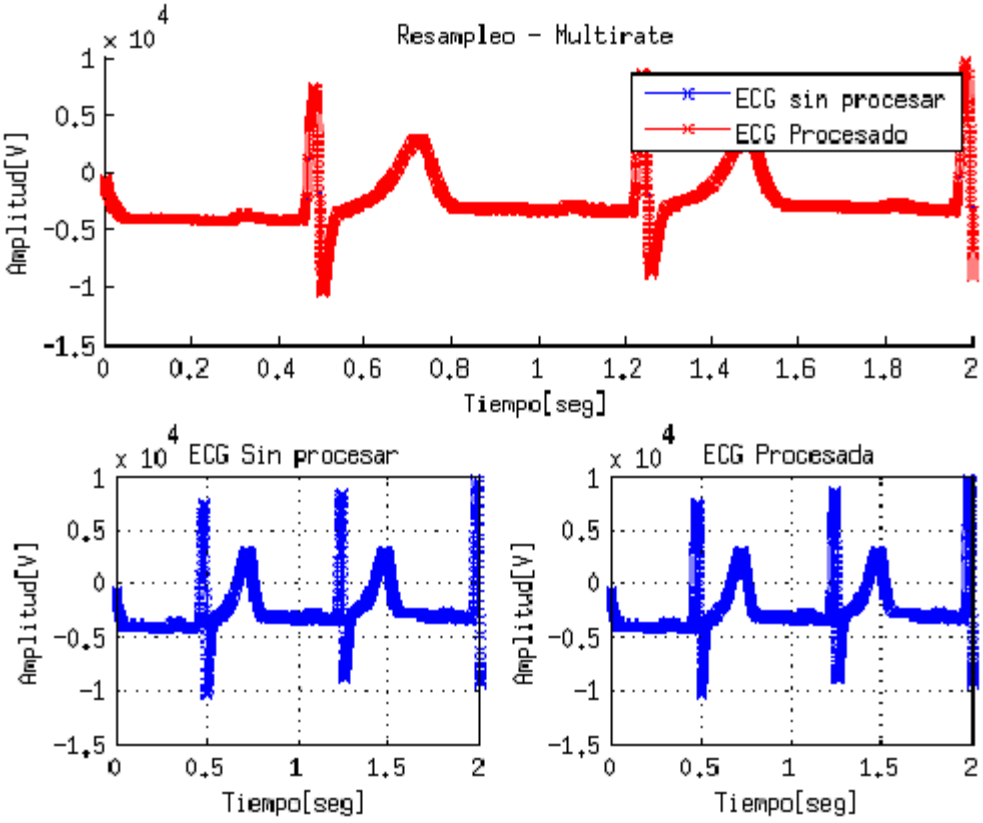
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

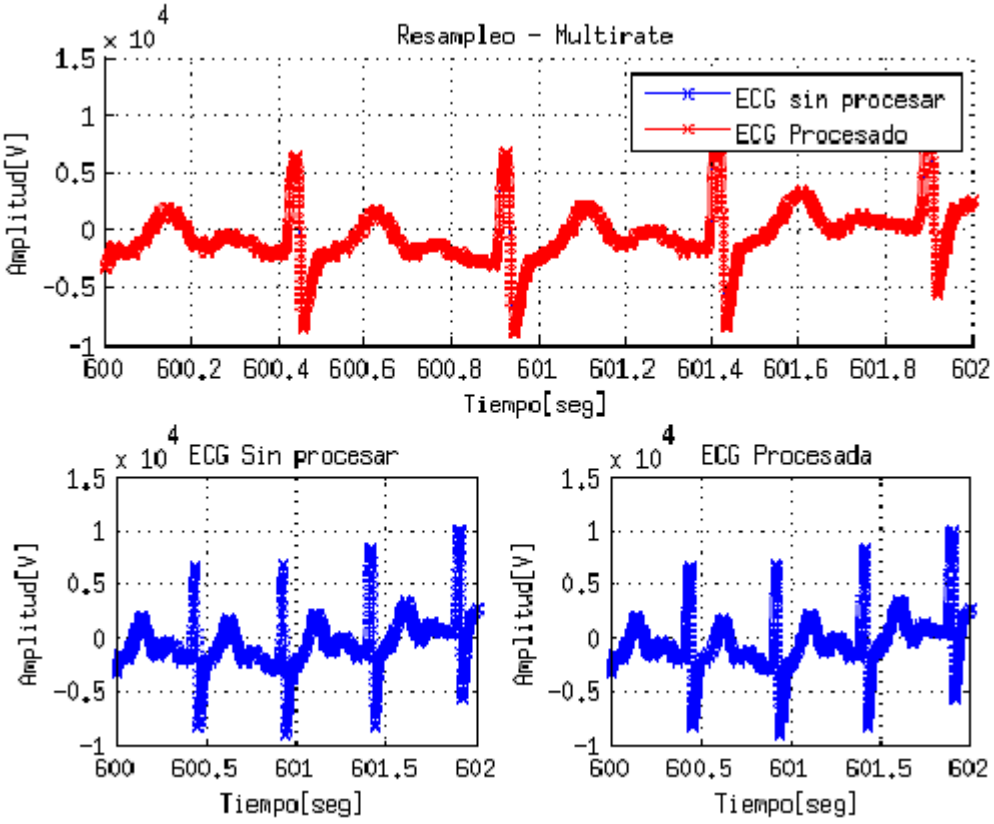
Señal de Fs: 1000Hz re-muestreada con Fs = 2000Hz

Inicio de la grabación



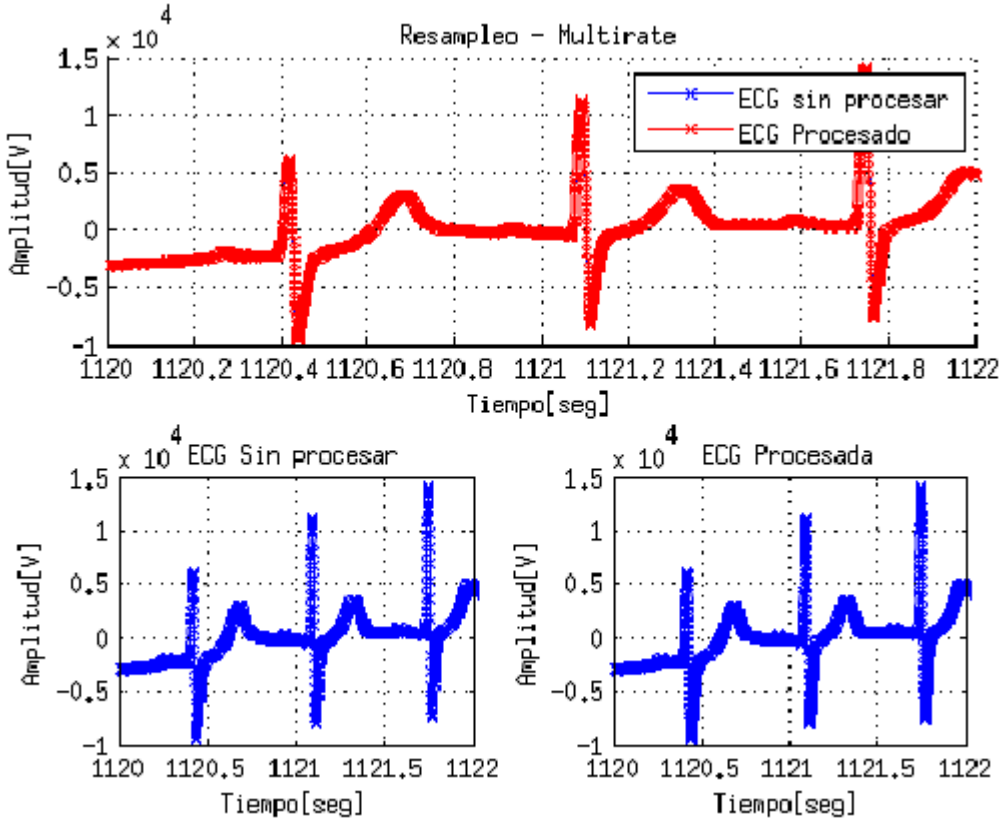
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Finde la grabación



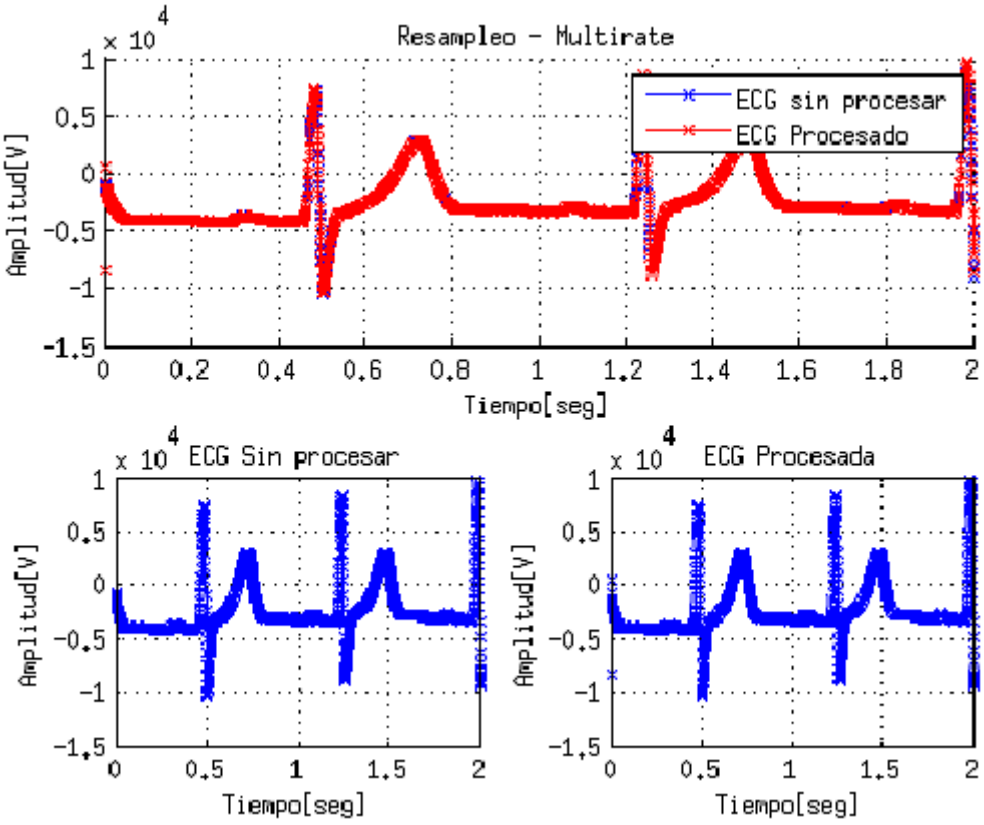
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

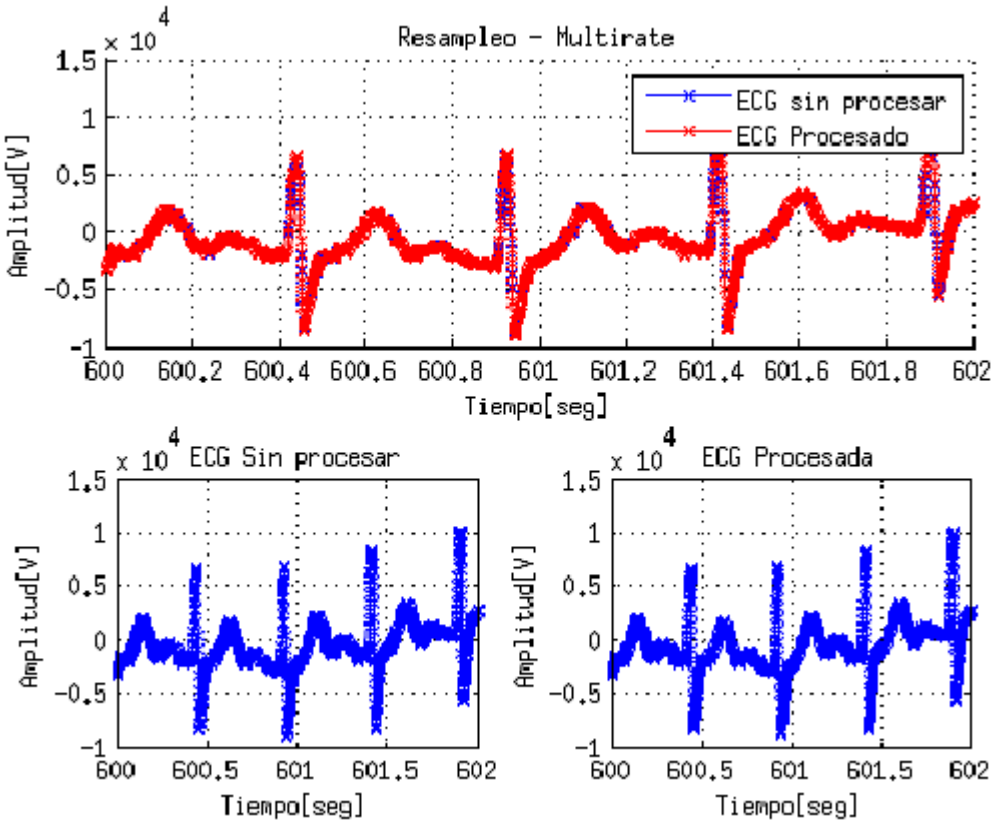
Señal de Fs: 1000Hz re-muestreada con Fs = 1100Hz

Inicio de la grabación



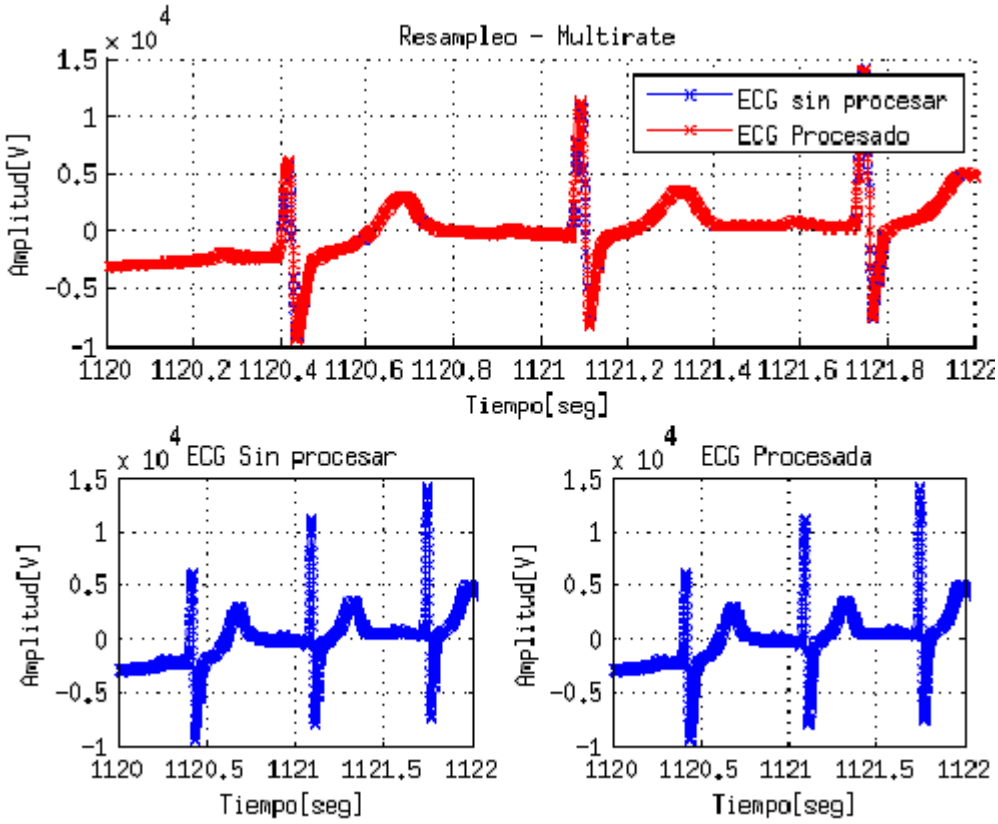
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

Tanto para la conversión de la frecuencia de muestro de 1 Khz (fs inicial) a 180 Hz como a 900 Hz el Matlab arrojó un problema de memoria. Por lo que se limito el trabajo práctico al cambio de la frecuencia de muestreo expuestos.

A continuación se exponen los tiempos de operación obtenidos al realizar la conversión de la tasa o frecuencia de muestreo de la señal adjuntada al informe.

Tabla comparativa de tiempos de operación.

Fs' [Hz]	Rango [Segundos]	Tiempo de Operación [Segundos]
250	0 – 2	1.5448
2000	0 – 2	1.5737
1100	0 – 2	26.9900
250	600 – 602	1.5351
2000	600 – 602	1.5524
1100	600 – 602	24.3768
250	1120 – 1122	1.5789
2000	1120 – 1122	1.5263
1100	1120 – 1122	25.1118

Nota: Esta tabla se utilizará para desarrollar las conclusiones pertinentes al ejercicio 3 (tres) del presente trabajo práctico por lo que se pospondrán las conclusiones pertinentes.

Ejercicio 1 – Códigos

A continuación se expone el código principal generado para la resolución del ejercicio

Ejercicio 1 - Código: Función manejadora del ejercicio 1.

```
%**
% \fn [fp] = FuncEjerciciol(Signal,fs,fsx,tol,fp)
% \brief Función manejadora de ejercicio 1
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal original
%   \param  Signal_Resample - Frecuencia de muestreo de la señal original
% \param  ts              - Frecuencia de muestreo de la señal modificada
%   \param  tol             - Tolerancia
%   \param  fp              - Número de imagen
%   \return fp              - Nuevo número de imagen

function [fp] = FuncEjerciciol(Signal,fs,fsx,tol,fp)

Signal_Resample = Resamplel(Signal,fs,fsx,tol);

N  = length(Signal);
Nx = length(Signal_Resample);

ts  = 0:1/fs:(N-1)*1/fs;
tsx = 0:1/fsx:(Nx-1)*1/fsx;
fp = Ploteol(Signal,Signal_Resample,ts,tsx,fp);

fp = fp+1;

end
```

Trabajo práctico 6 - Ejercicio 2

Repetir el punto 1 de manera tal que la función de remuestreo elija factorización de M y L más adecuada en tanto a:

1. Costo computacional. Proporcional a la cantidad de coeficientes empleada por los filtros antialias (ver la función firpmord, rat y factor).
2. Memoria intermedia máxima requerida. El costo computacional es proporcional a la longitud de cada filtro (que predice firmord). Es decir que habiendo M caminos posibles para hacer el cambio de frecuencia de muestreo, por ejemplo:
   L: 3 3 5 (45)
   M: 2 7 (14) Se podria pensar también por medio de los siguientes caminos (ver enunciado) y en esos casos elegir por el de menor costo.

   - fs1 = 250 Hz
   - fs1 = 2000 Hz
   - fs1 = 1100 Hz
   - fs1 = 180 Hz
   - fs1 = 999 Hz

Recordar que la fs del ECG es de 1000Hz. Para cada situación mostrar un segmento de 2 segundos del principio, parte media y final de cada registro. Comparar las 5 situaciones evidenciando la densisdad de muestras en cada señal.

Para la realización de este ejercicio se implementaron las mejoras pedidas sobre el algoritmo de resampleo.

A continuación se desarrolla la función de resampleo mejorada.

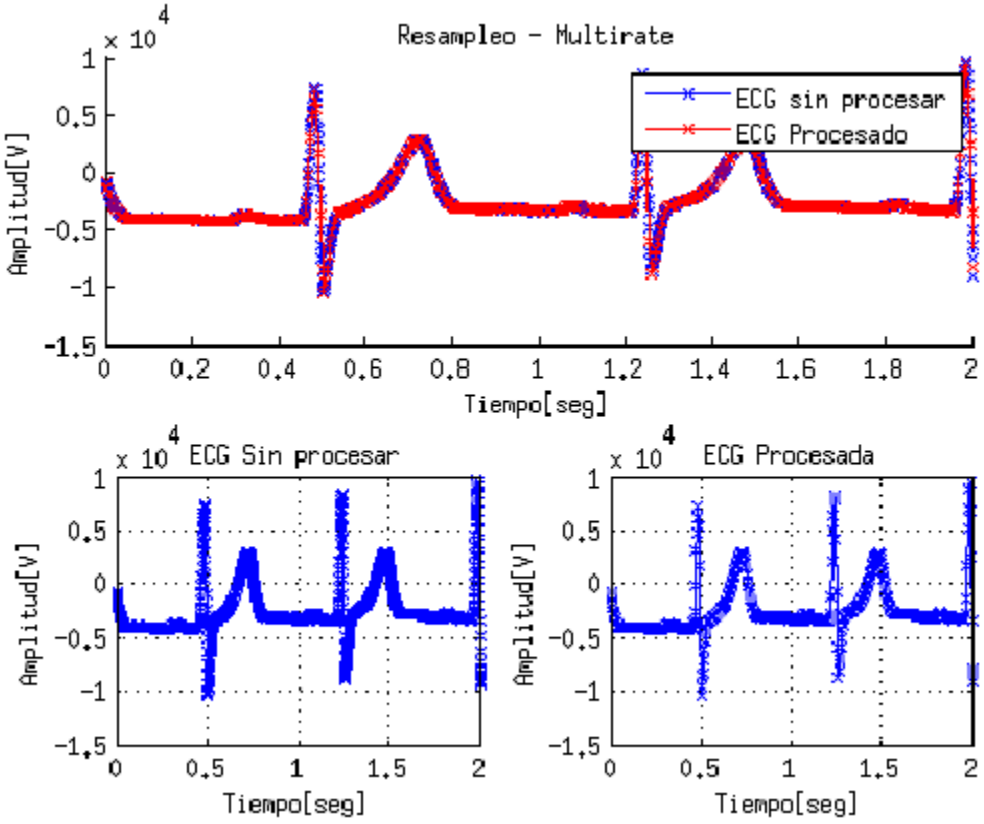
```
%**
% \fn [Signal_Resample] = Resample2(Signal,fs,fsx,tol)
% \brief Función que resamplea (ejercicio: 2)
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal a a trabajar
%   \param  fs              - Frecuencia de la señal de entrada
%   \param  fsx             - Frecuencia nueva de sampleo
% \param tol              - Tolerancia de aproximacion
%   \return Signal_Resample - Señal resampleada

function [Signal_Resample] = Resample2(Signal,fs,fsx,tol)

    Coef_Rel = fsx/fs;

    % [N,D] = rat(X,tol) returns N./D approximating X to within tol.
    [I,M] = rat(Coef_Rel, tol * Coef_Rel);

    % Factorizo numerador y denominador.

    Ifact = factor(I);
    Mfact = factor(M);

    % Completo con 1s si alguno de los dos vectores obtenidos es menor
    % que el otro vector obtenido.

    if length(Ifact) > length(Mfact)
        Mfact = [Mfact ones(1,length(Ifact) - length(Mfact))];
    else
        Ifact = [Ifact ones(1,length(Mfact) - length(Ifact))];
    end

    % Ordeno de mayor a menor los vectores.

    Ifact = sort(Ifact);
    Mfact = sort(Mfact);

    [~,IX] = sort(Ifact./Mfact);

    Signal_Resample = Signal;

    for i = 1: length(Ifact)

        CoefMax = max(Ifact(IX(i)),Mfact(IX(i)));

        % Realizo el upsample
        out = func_interpol(Signal_Resample,Ifact(IX(i)));

        % Realizo el filtrado
        Signal_filter = Ifact(IX(i)) * filterLp(out,1/CoefMax);

        % Realizo el downsample

       Signal_Resample = func_diez(Signal_filter, Mfact(IX(i)));
    end
end
```

Este algoritmo luego de realizar el cociente de frecuencias y obner un numerador y denominador enteros (I y M) se descompuso a los mismos en factores primos. De esta forma el numerador y denominador se transformaron en vectores compuestos por los valores calculados. A continuación se igualan los largos de los vectores (en caso de que los mismos sean desiguales) obteniendo dos vectores (I y M) de la misma longitud (complentando a los mismos con 1's en caso de ser necesario). A partir de estos cambios se logra una mejora en lo que respecta al costo computacional vista en clase y pedido por el ejercicio.

A continuación utilizando la señal ECG incluida con el informe se realiza los cambios de frecuencia dados como consigna.
Para evidenciar los cambios de frecuencia de muestreo se exponen tres partes de la grabación. Estas partes son: El inicio, el medio y fin de la grabación.
Finalmente se desarrollan las conclusiones pertinentes a los gráficos expuestos y junto a la exposición de una tabla de tiempos de operación.

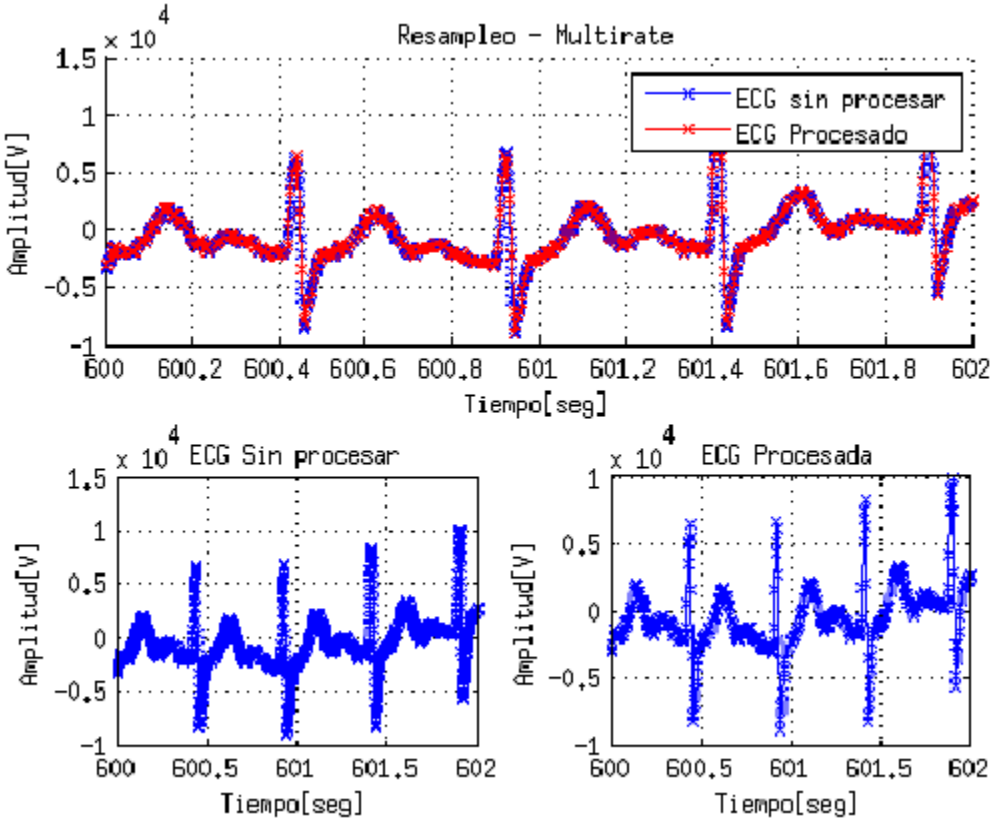
Señal de Fs: 1000Hz re-muestreada con Fs = 250Hz

Inicio de la grabación



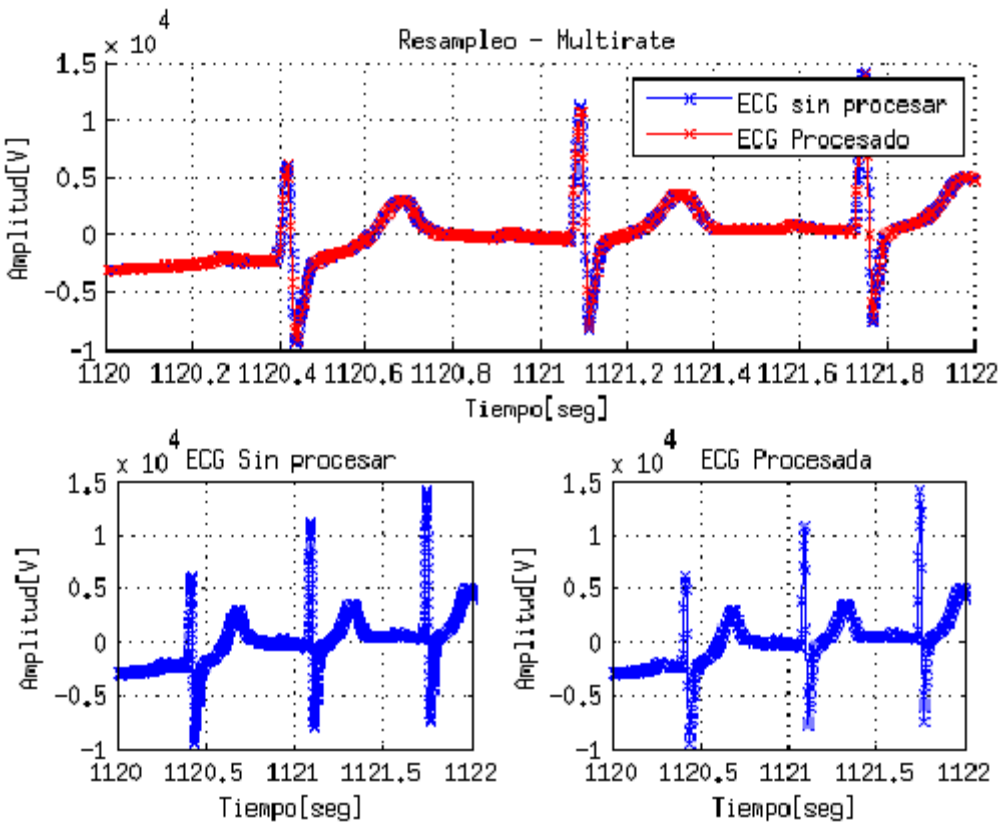
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



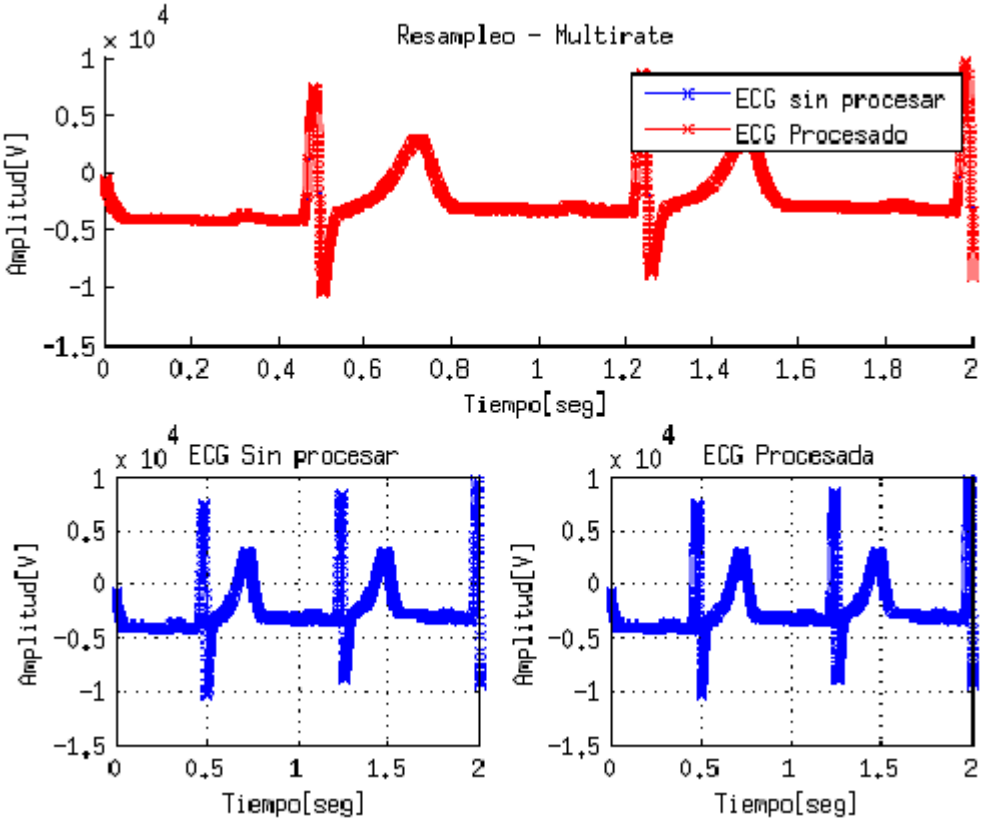
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

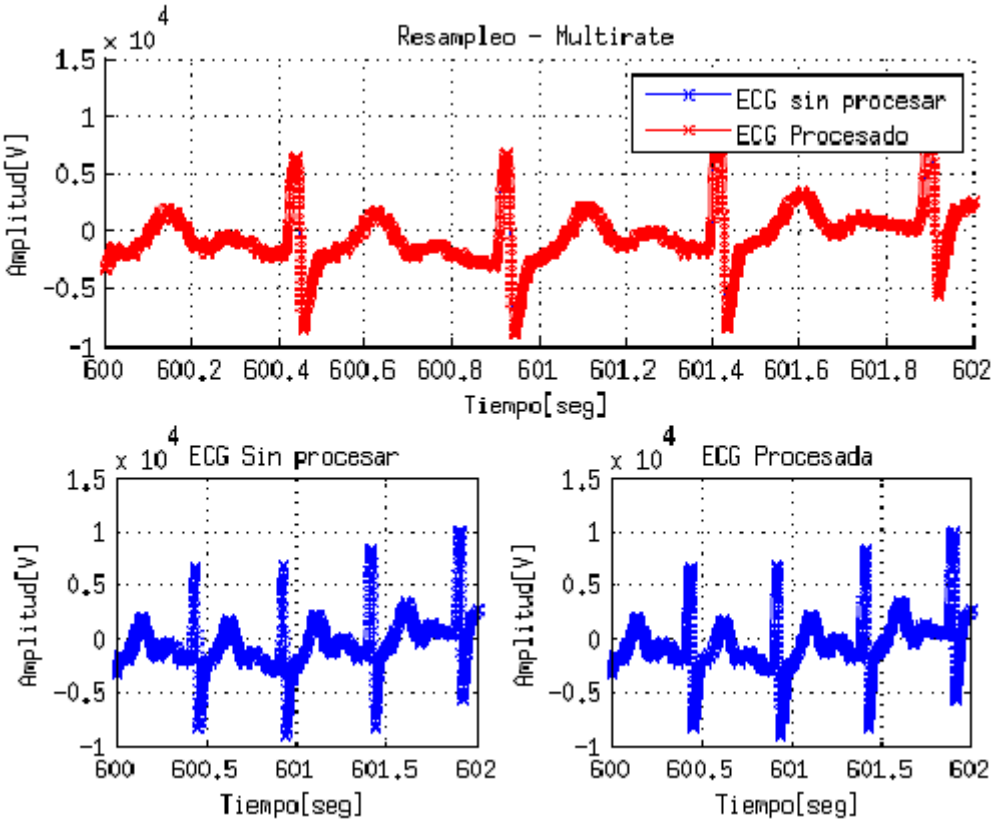
Señal de Fs: 1000Hz re-muestreada con Fs = 2000Hz

Inicio de la grabación



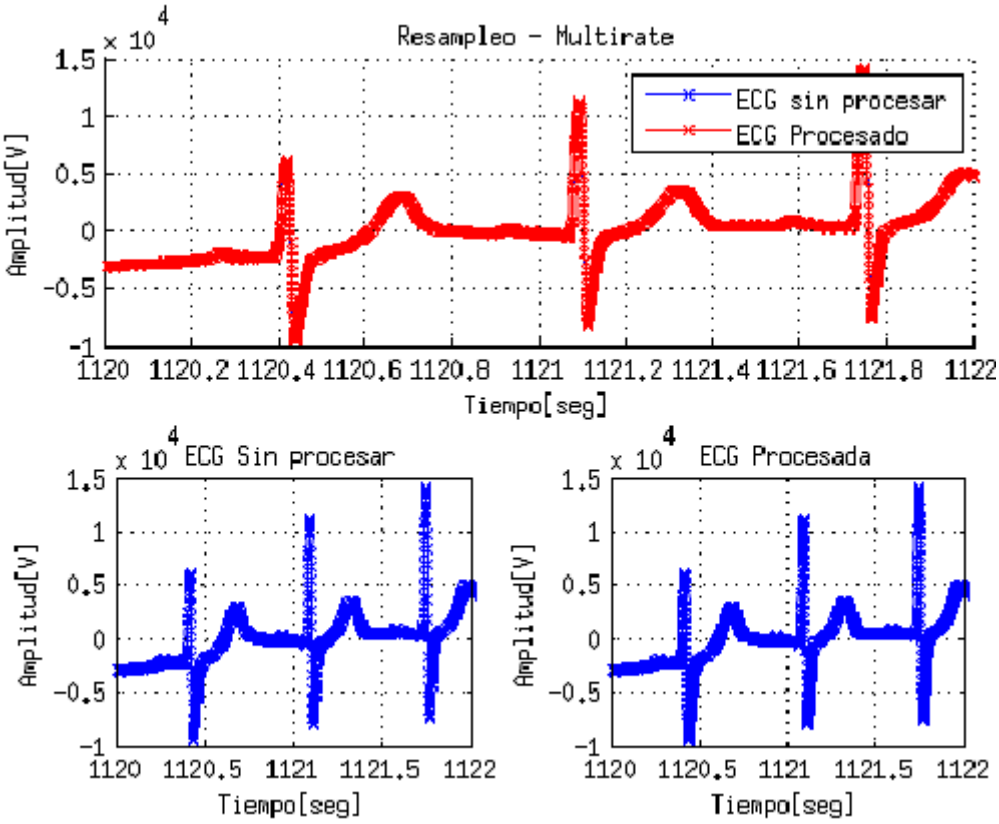
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



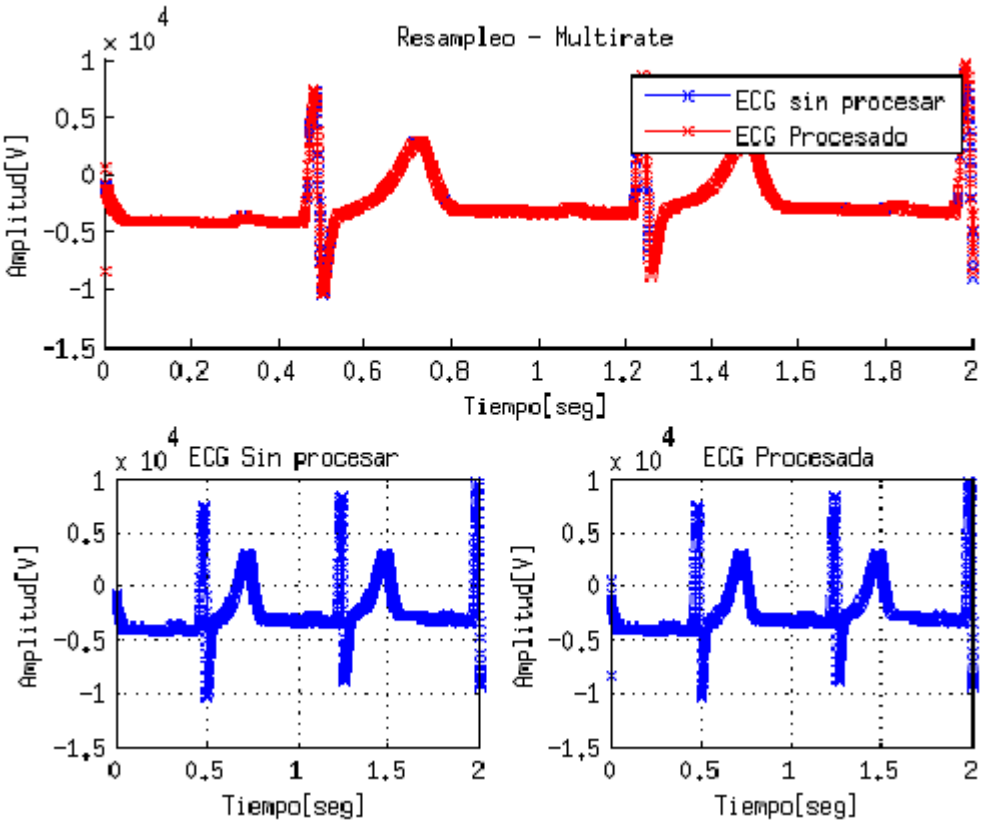
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

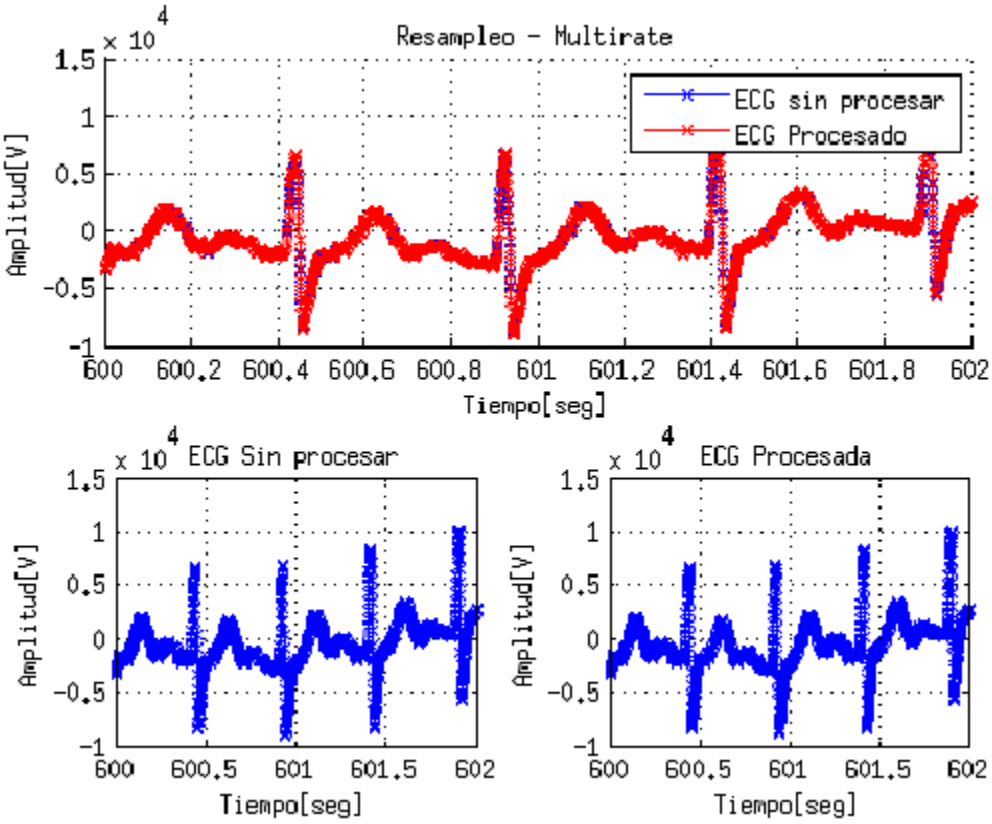
Señal de Fs: 1000Hz re-muestreada con Fs = 1100Hz

Inicio de la grabación



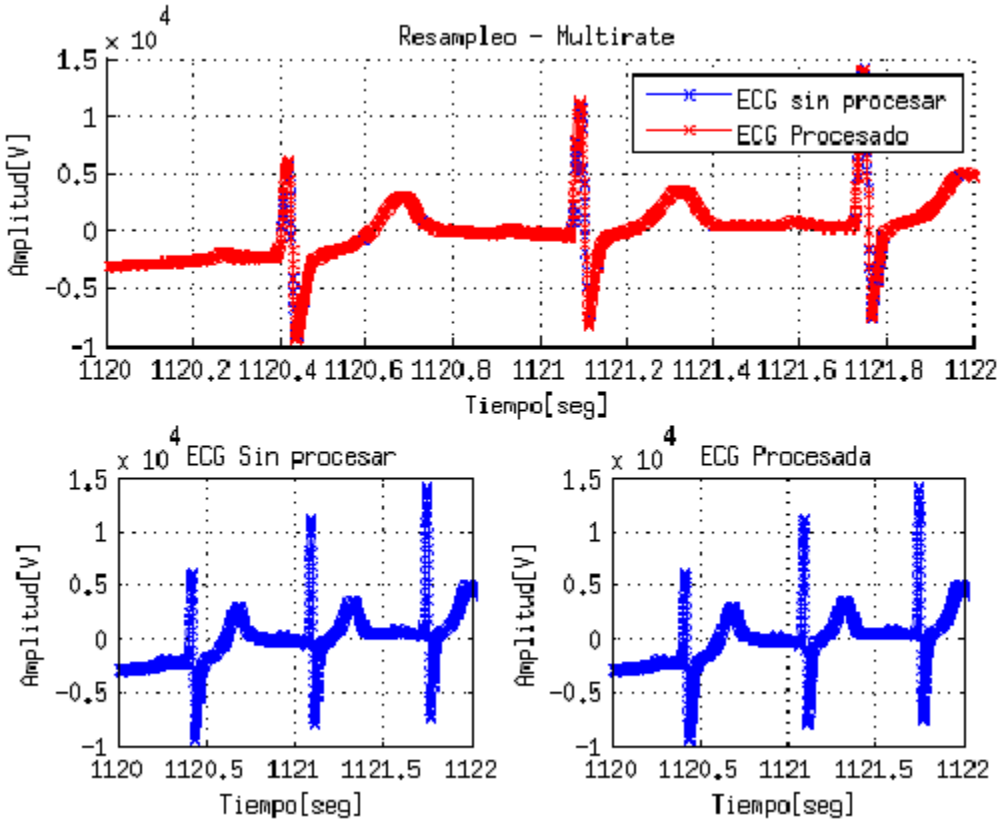
Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



Observación: Del gráfico se observa que no se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

A continuación se exponen los tiempos de operación obtenidos al realizar la conversión de la tasa o frecuencia de muestreo de la señal adjuntada al informe.

Tabla comparativa de tiempos de operación.

Fs' [Hz]	Rango [Segundos]	Tiempo de Operación [Segundos]
250	0 – 2	1.4572
2000	0 – 2	1.4682
1100	0 – 2	24.7439
250	600 – 602	1.5514
2000	600 – 602	1.5622
1100	600 – 602	30.5501
250	1120 – 1122	1.4670
2000	1120 – 1122	1.5241
1100	1120 – 1122	24.8659

Nota: Esta tabla se utilizará para desarrollar las conclusiones pertinentes al ejercicio 3 (tres) del presente trabajo práctico por lo que se pospondrán las conclusiones pertinentes.

Observación: El script realizado para el ploteo de gráficos es nuevamente el utilizado en el ejercicio 1 del presente trabajo práctico.

Ejercicio 2 – Códigos

A continuación se expone el código principal generado para la resolución del ejercicio

Ejercicio 2 - Código: Función manejadora del ejercicio 2.

```
%**
% \fn [fp] = FuncEjercicio2(Signal,fs,fsx,tol,fp)
% \brief Función manejadora de ejercicio 2
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal original
%   \param  Signal_Resample - Frecuencia de muestreo de la señal original
% \param  ts              - Frecuencia de muestreo de la señal modificada
%   \param  tol             - Tolerancia
%   \param  fp              - Número de imagen
%   \return fp              - Nuevo número de imagen

function [fp] = FuncEjercicio2(Signal,fs,fsx,tol,fp)

Signal_Resample = Resample2(Signal,fs,fsx,tol);

N  = length(Signal);
Nx = length(Signal_Resample);

ts  = 0:1/fs:(N-1)*1/fs;
tsx = 0:1/fsx:(Nx-1)*1/fsx;
fp = Ploteo1(Signal,Signal_Resample,ts,tsx,fp);

fp = fp+1;

end
```

Trabajo práctico 6 - Ejercicio 3

Realizar las optimizaciones computacionales polyphase para las 3 funciones del punto 2. Comparar el tiempo de ejecución para cada situación del punto 1 y del punto 2.

Introducción

Estructuras de los filtros polifásicos

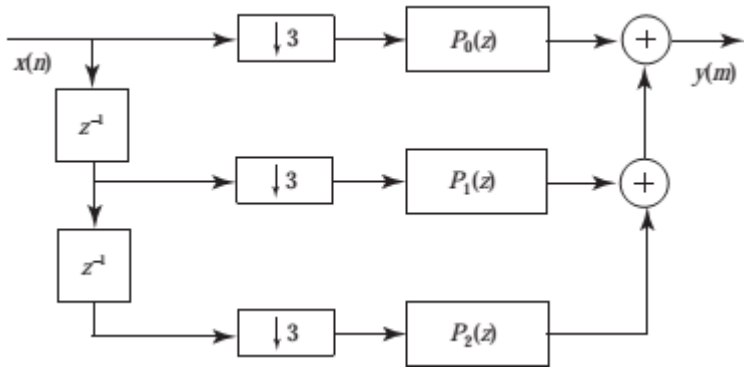
Las estructuras polifásicas de los filtros FIR se desarrollaron para permitir la implementación eficiente de los convertidores de la frecuencia de muestreo; sin embargo, pueden emplearse también en otras aplicaciones. La estructura polifásica se basa en el hecho de que ualquier función de sistema puede separarse. *(Para más información referirse a la bibliografía recomendada al final del informe)*

Estructuras polifásicas para filtros de diezmado y interpolación

Implementación eficiente del diezmado:

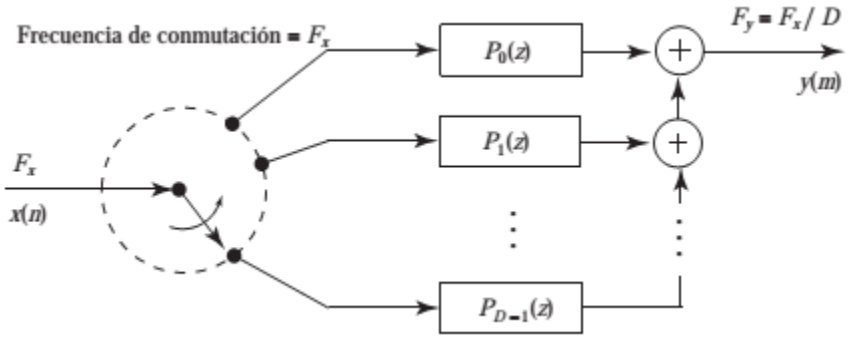
Para desarrollar una estructura polifásica para diezmado, se parte de la implementación directa del proceso de diezmado. La secuencia diezmada se obtiene pasando la secuencia de entrada x(n) a través de un filtro lineal y luego submuestreando la salida del filtro por un factor D. En esta configuración, el filtro opera a la frecuencia de muestreo superior Fx mientras que solo se precisa una de cada D muestras de salida. Una solución lógica sería encontrar una estructura qen la que sólo se calcuen las muestras necesarias.

Se desarrolla dichaimplementación utilizando la estructura polifásica. Puesto que el submuestreo conmuta con la suma, combinando las estructuras se obtiene la siguiente estructura:



En esta estructura de filtrado, solo se calculan las muestras necesarias y todas las multiplicaciones y sumas se realizan a la frecuencia de muestreo inferior Fx/D. Así hemos obtenido la eficiencia deseada. Puede conseguirse una reducción adicional de los cálculos utilizando un filtro FIR de fase lineal y aplicando la propiedad de simetría de su respuesta al impulso.

En la práctica es más conveniente implementar el diezmado polifásico utilizando un modelo de conmutador como el que se muestra en la siguiente figura:



El conmutador gira en sentido horario a partir del instante n = 0 y distribuye un bloque de D muestras de entrada a los filtros polifásicos comenzando por el filtro i = D – 1 y continuando en orden inverso hasta i = 0.

Por cada bloque de D muestras de entrada, los filtros polifásicos reciben una nueva entrada y sus salidas se calculan y suman para generar una muestra de la señal de salida y(m).

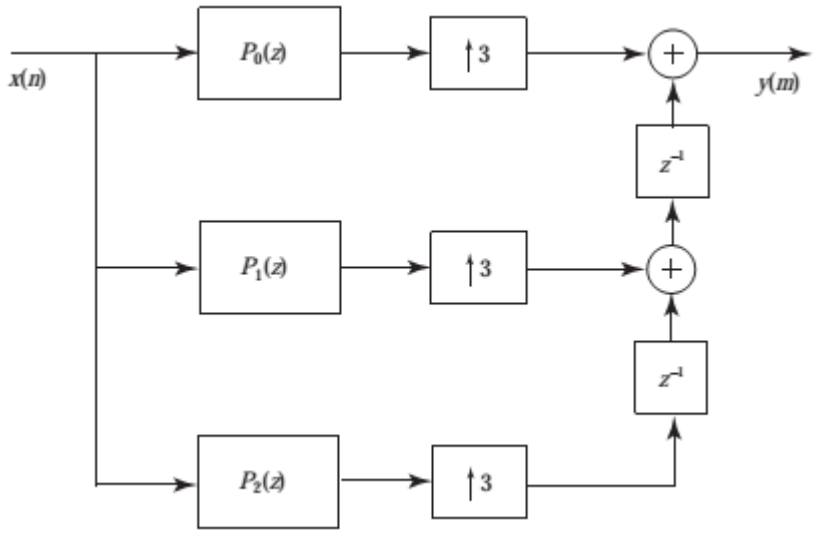
A continuación se desarrolla la función de diezmado polifásico

```
%**
% \fn [Signal_Diez] = FuncDiezPol(Signal,M)
% \brief Elimina M-1 muestras entre valores sucesivos de la señal Signal
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal     - Señal a interpolar
%   \param  I          - Muestras a ingresar por muestra de Signal
% \return Signal_Int - Señal interpolada

function [Signal_Diez] = FuncDiezPol(Signal,M)

    Order = 2;
    RollOff = 0.5;
    Scale = max(Signal);

    if M == 1
        Signal_Diez = Signal;
        return;
    end

    nS = length(Signal);
    nZ = ceil(nS/M) * M - nS;

    Signal = [Signal zeros(1,nZ)];
    % Creacion del filtro
    OrderFilter = (Order * M);
    Filtro = firnyquist(OrderFilter,M,RollOff);

    % Creacion del banco de filtros (Matriz de filtros)
    h(1,:) = downsample(Filtro,M,OrderFilter);
    for i = 1:1:M-1
        h(i+1,:) = [0 downsample(Filtro,M,i)];
    end

    % Obtengo las señales a ser filtradas
    for j = 0:1:M-1
        SignalAFilter(j+1,:) = downsample(Signal,M,j);
    end

    % Filtro las señales
    for j = 0:1:M-1
        SignalFilter(j+1,:) = filtfilt(h(j+1,:), 1, SignalAFilter(j+1,:));
    end

    %Suma
    SignalSumFilter =  sum(SignalFilter,1);

    Signal_Diez = SignalSUmFilter;

    Signal_Diez = SignalSumFilter;
    Signal_Diez = (Signal_Diez / max(Signal_Diez)) * Scale;

end
```

Implementación eficiente del interpolador:

Se realiza insertando en primer lugar I – 1 ceros entre muestras sucesivas de x(n) y filtrando a continuación la secuencia resultante. El principal problema con esta estructura qes que los cálculos para el filtro se efectúan a la frecuencia de muestreo superior I Fx. La simplifacipon deseada se consigue reemplazando en primer lugar el filtro por la estructura polifásica trasnpuesta. *(Para más información referirse a la bibliografía recomendada al final del informe)*

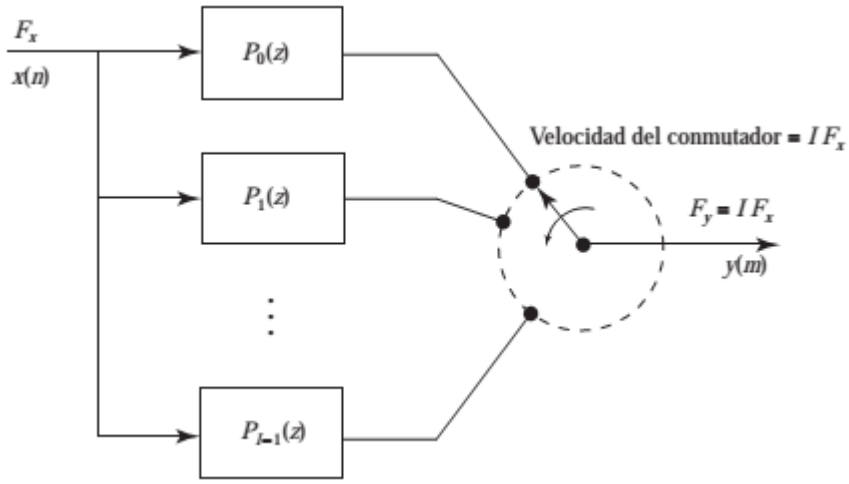
Utilizando las identidades vistas en clase (identidad noble) se obiene la siguiente estructura:



Así, todas las multiplicaciones de la operación de filtrado se efectúan a la frecuencia Fx. Es interesante destacar que la estructura de un interpolador se puede obtener transponiendo la estructura de un diezmador, y viceversa.

Para cada muestra de entrada, los filtros polifásicos producen I muestras a la salida. Dado que la salida $y_i(n)$ del filtro i va seguida por (I – 1) ceros y estpa retardada i muestras, los filtros polifásicos contribuyen con muestras distintas de cero en diferentes franjas temporales.

En la práctica es más conveniente implementar el interpolador polifásico utilizando un modelo de conmutador como el que se muestra en la siguiente figura:



El conmutador gira en sentido antihorario empezando en el instante n = 0 en la rama i = 0. Por cada muestra de entrada x(n), el conmutador lee la salida de los filtros polifásicos para obtener I muestras de la señal de salida (interpolada) y(m). Cada filtro polifásico oepra sobre la misma entrada utilizando su conjunto unpivoco de coeficientes. Por tanto, podemos obtener los mismos resultados empleando un solo filtro y cargando secuencialmente un conjunto diferente de coeficientes.

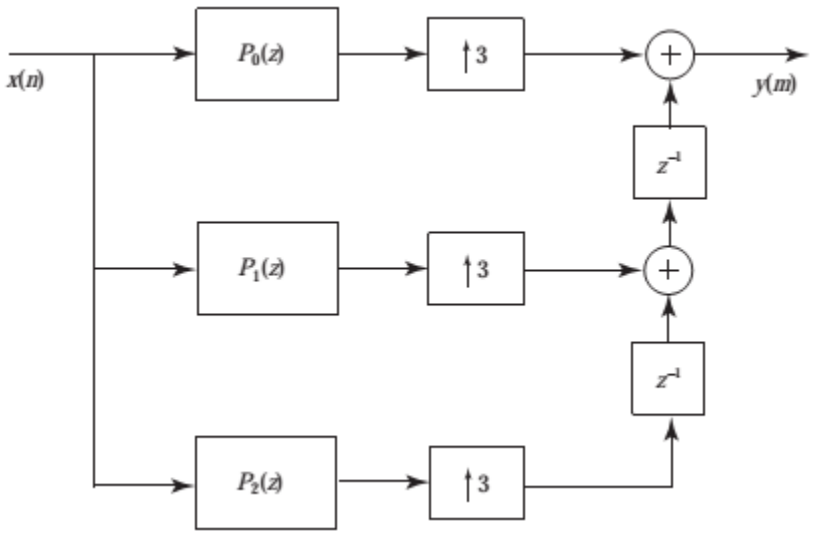
A continuación se desarrolla la función de interpolación polifásico

```
%**
% \fn [Signal_Int] = FuncIntPol(Signal,I)
% \brief Agrega I-1 muestras entre valores sucesivos de la señal Signal
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal     - Señal a interpolar
%   \param  I          - Muestras a ingresar por muestra de Signal
% \return Signal_Int - Señal interpolada

function [Signal_Int] = FuncIntPol(Signal,I)

    Order = 2;
    RollOff = 0.5;
    Scale = max(Signal);

    if I == 1
        Signal_Int = Signal;
        return;
    end

        Error using FuncIntPol (line 14)
        Not enough input arguments.



    % Creacion del filtro
    OrderFilter = (Order * I);
    Filtro = firnyquist(OrderFilter,I,RollOff);


    % Creacion del banco de filtros (Matriz de filtros)
    h(1,:) = downsample(Filtro,I,OrderFilter);
    for i = 1:1:I-1
        h(i+1,:) = [0 downsample(Filtro,I,i)];
    end

    % Filtro las señales
    for j = 0:1:I-1
        SignalFilter(j+1,:) = filtfilt(h(j+1,:), 1, Signal);
    end

    % Obención de Signal_Int (Salida ó retorno de la función)

    Signal_Int = [ SignalFilter(:,1)' ];

    for i = 2:1:length(SignalFilter)
        Signal_Int =  [ Signal_Int SignalFilter(:,i)' ];
    end

    %Escala
    Signal_Int = (Signal_Int / max(Signal_Int)) * Scale;

end
```

Estructura para la conversión de la frecuencia de meustreo racional.

Un convertidor de frecuencia de muestreo con una relación I/D se puede implementar de forma eficiente utilizando un interpolador polifásico seguido de un sumbmuestreador. Sin embargo, puesto que el submuestreador sólo conserva D salidas del subfiltro polifásico, no es necesario calcular los I valores interpolados entre muestras de entrada sucesivas.

Implementación de un sistema de interpolación utilizando una estructura polifásica y utilizando la identidad noble:



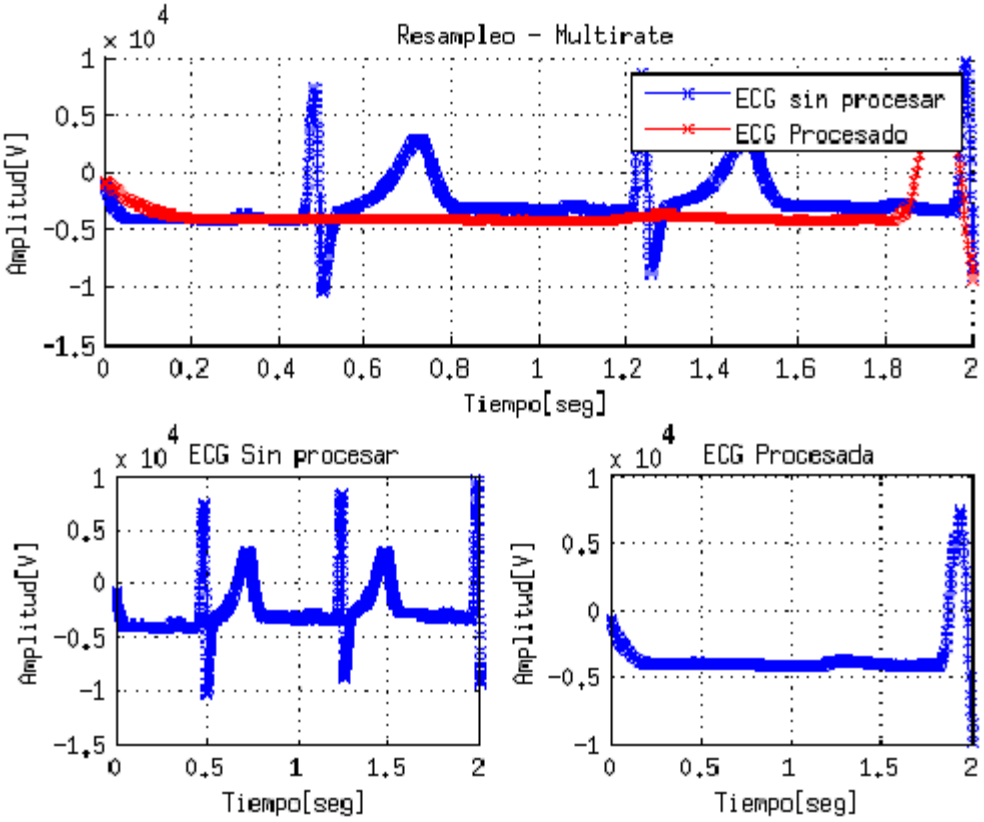
A continuación se desarrolla la función de Resampleo.

```
%**
% \fn [Signal_Resample] = Resample3(Signal,fs,fsx,tol)
% \brief Función que resamplea (ejercicio: 3)
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal a a trabajar
%   \param  fs              - Frecuencia de la señal de entrada
%   \param  fsx             - Frecuencia nueva de sampleo
% \param tol              - Tolerancia de aproximacion
%   \return Signal_Resample - Señal resampleada

function [Signal_Resample] = Resample3(Signal,fs,fsx,tol)

    Coef_Rel = fsx/fs;

    % [N,D] = rat(X,tol) returns N./D approximating X to within tol.
    [I,M] = rat(Coef_Rel, tol * Coef_Rel);

    % Factorizo numerador y denominador.

    Ifact = factor(I);
    Mfact = factor(M);

    % Completo con 1s si alguno de los dos vectores obtenidos es menor
    % que el otro vector obtenido.

    if length(Ifact) > length(Mfact)
        Mfact = [Mfact ones(1,length(Ifact) - length(Mfact))];
    else
        Ifact = [Ifact ones(1,length(Mfact) - length(Ifact))];
    end

    % Ordeno de mayor a menor los vectores.

    Ifact = sort(Ifact);
    Mfact = sort(Mfact);

    Signal_Resample = Signal;

    nI = length(Ifact);
    nM = length(Mfact);
    i  = 1; j = 1;

    while (i <= nI && j <= nM)
        I = Ifact(i); M = Mfact(j);
        if I >= M
            Signal_Resample = FuncDiezPol(Signal,M);
            i = i+1;
        elseif M > I
            Signal_Resample = FuncIntPol(Signal,I);
            j = j+1;
        end
    end
end
```

A continuación utilizando la señal ECG incluida con el informe se realiza los cambios de frecuencia dados como consigna.
Para evidenciar los cambios de frecuencia de muestreo se exponen tres partes de la grabación. Estas partes son: El inicio, el medio y fin de la grabación.
Finalmente se desarrollan las conclusiones pertinentes a los gráficos expuestos y junto a la exposición de una tabla de tiempos de operación.

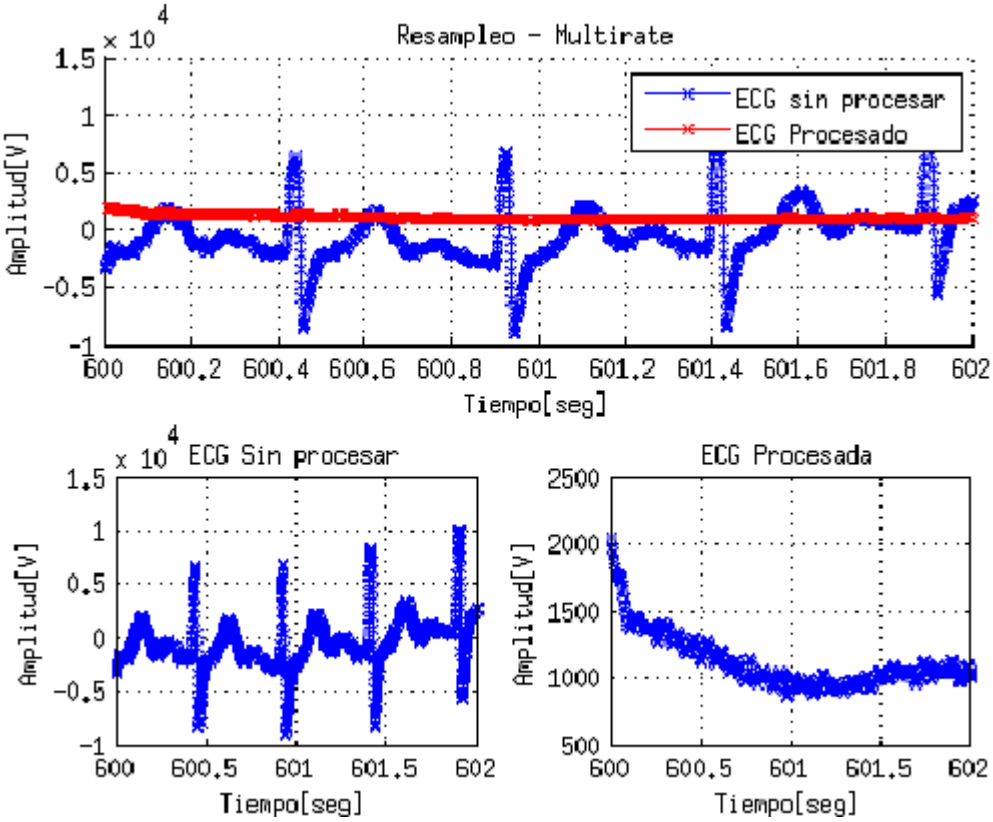
Señal de Fs: 1000Hz re-muestreada con Fs = 250Hz

Inicio de la grabación



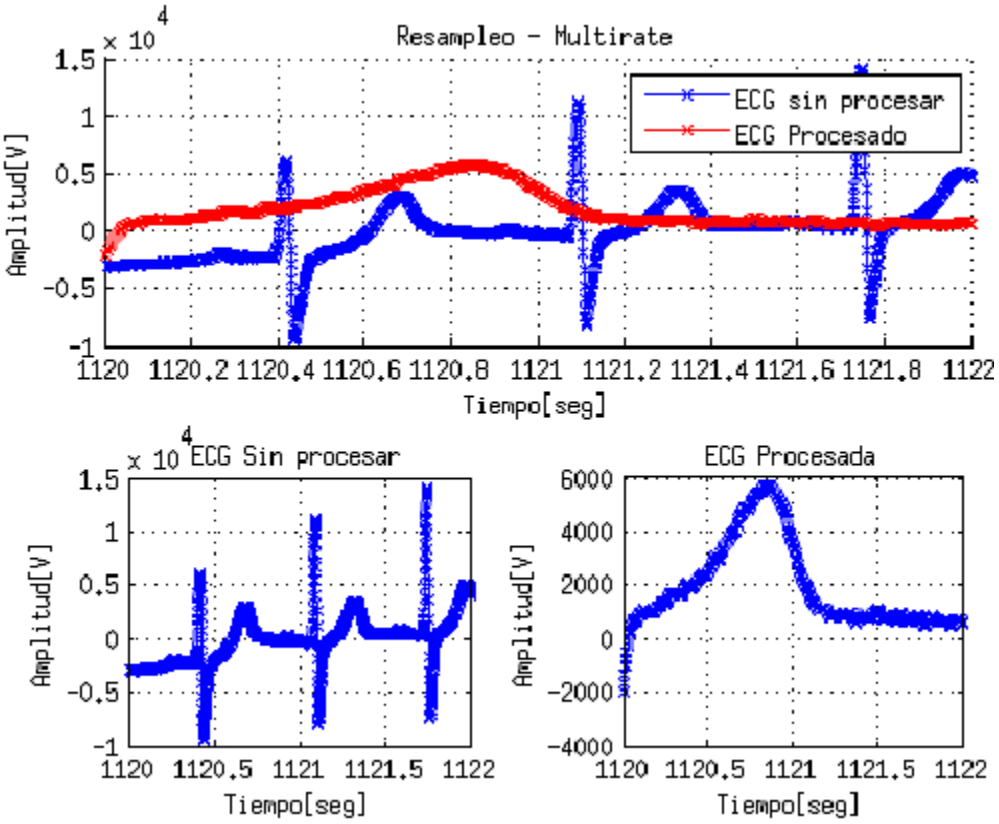
Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



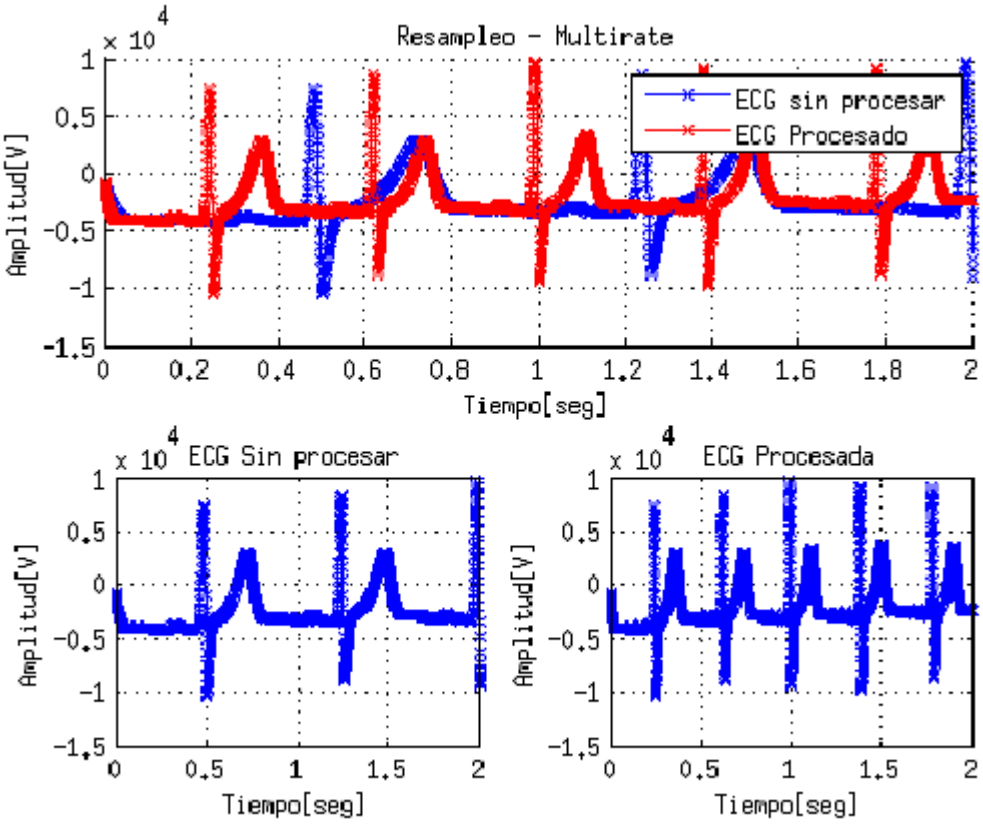
Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

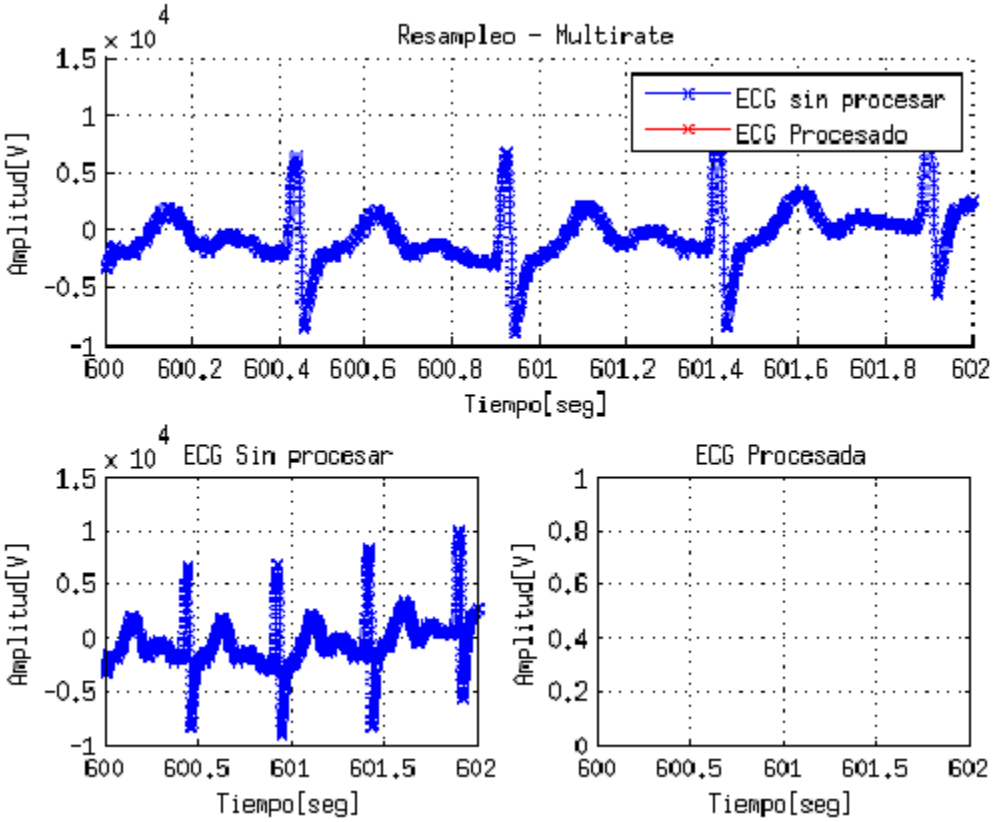
Señal de Fs: 1000Hz re-muestreada con Fs = 2000Hz

Inicio de la grabación



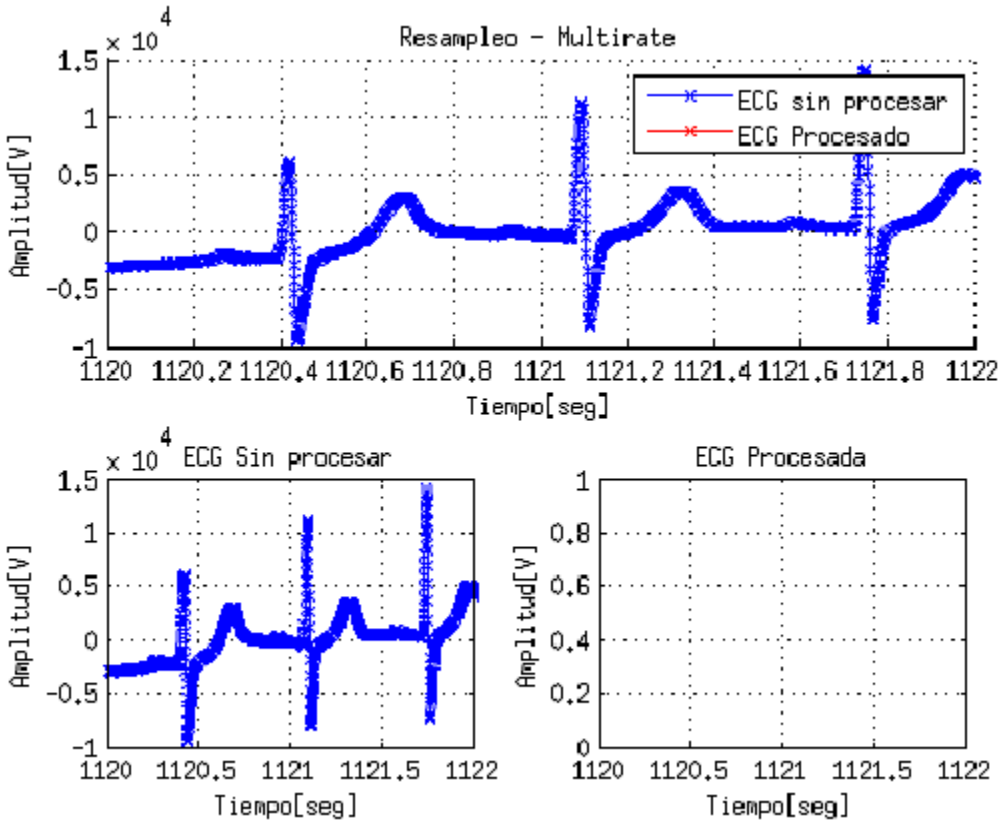
Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación



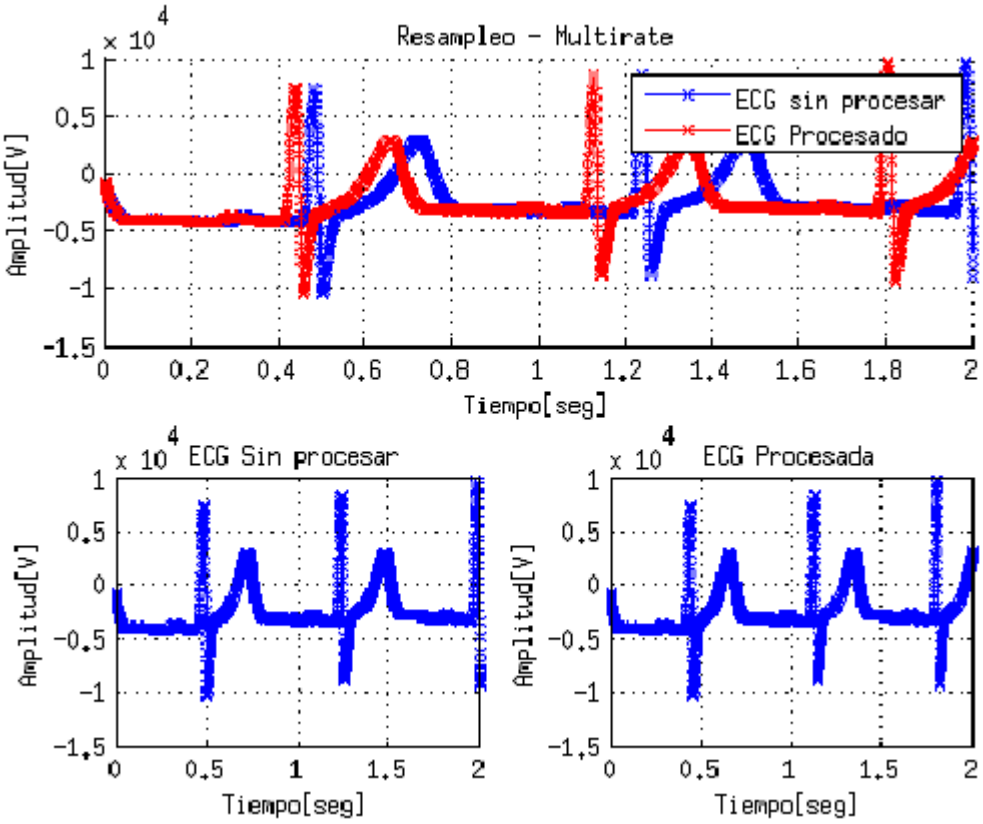
Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

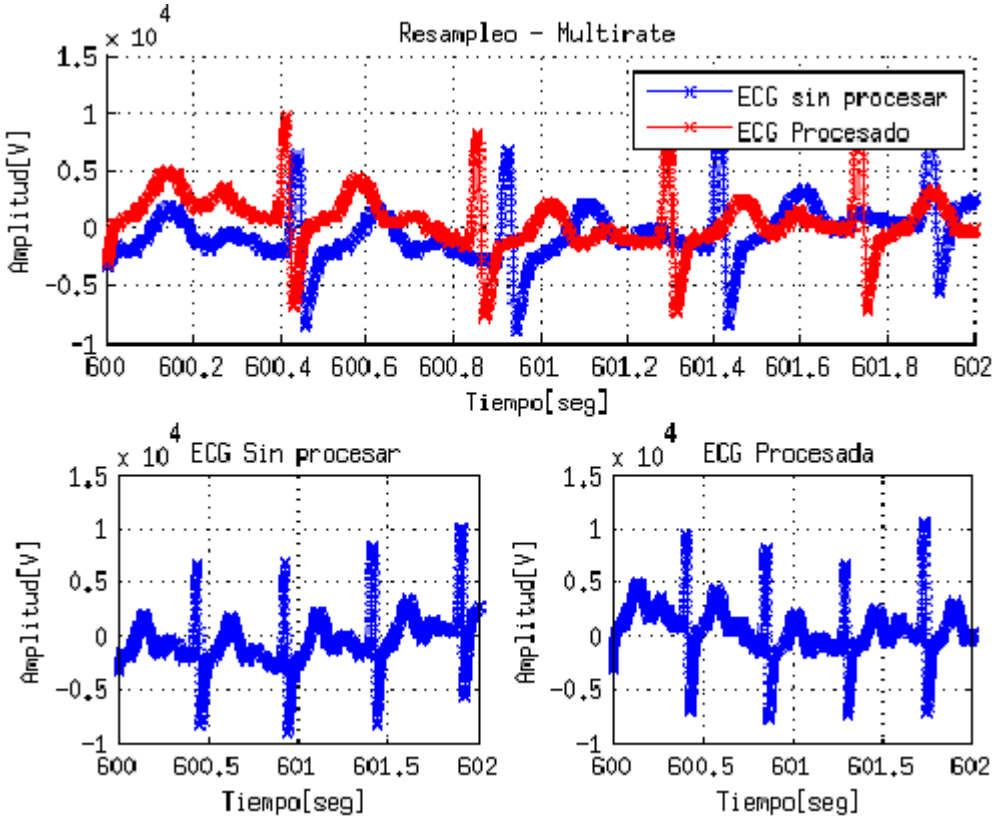
Señal de Fs: 1000Hz re-muestreada con Fs = 1100Hz

Inicio de la grabación



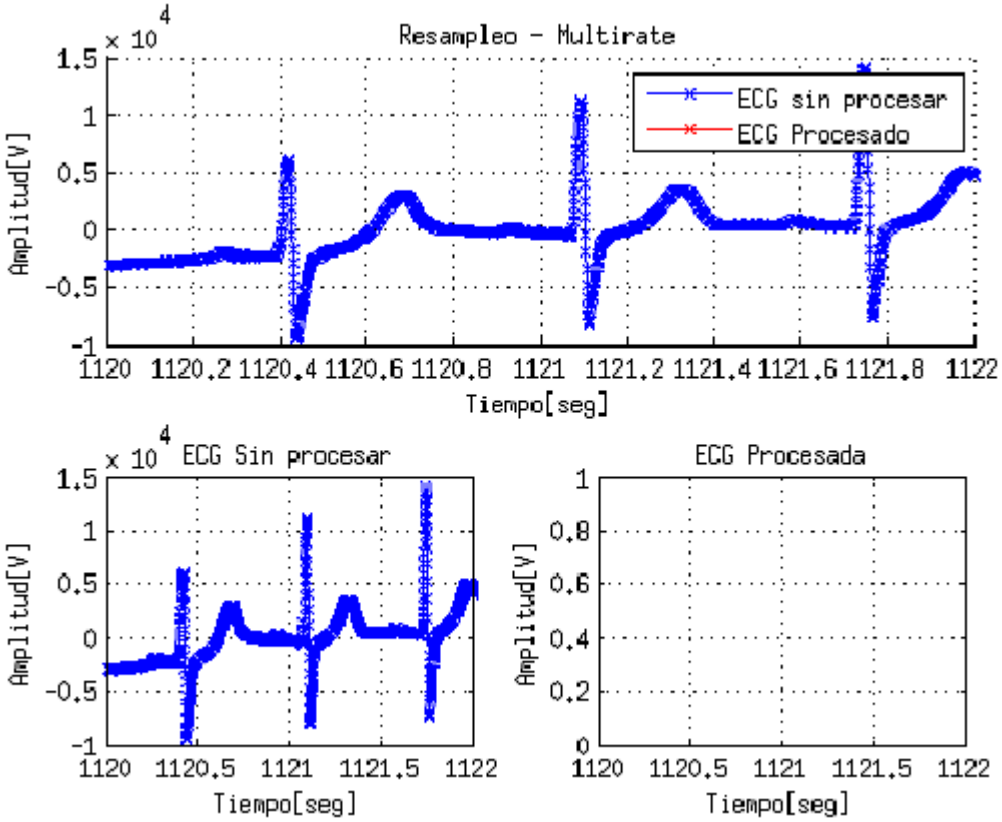
Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

En el medio de la grabación



Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Fin de la grabación

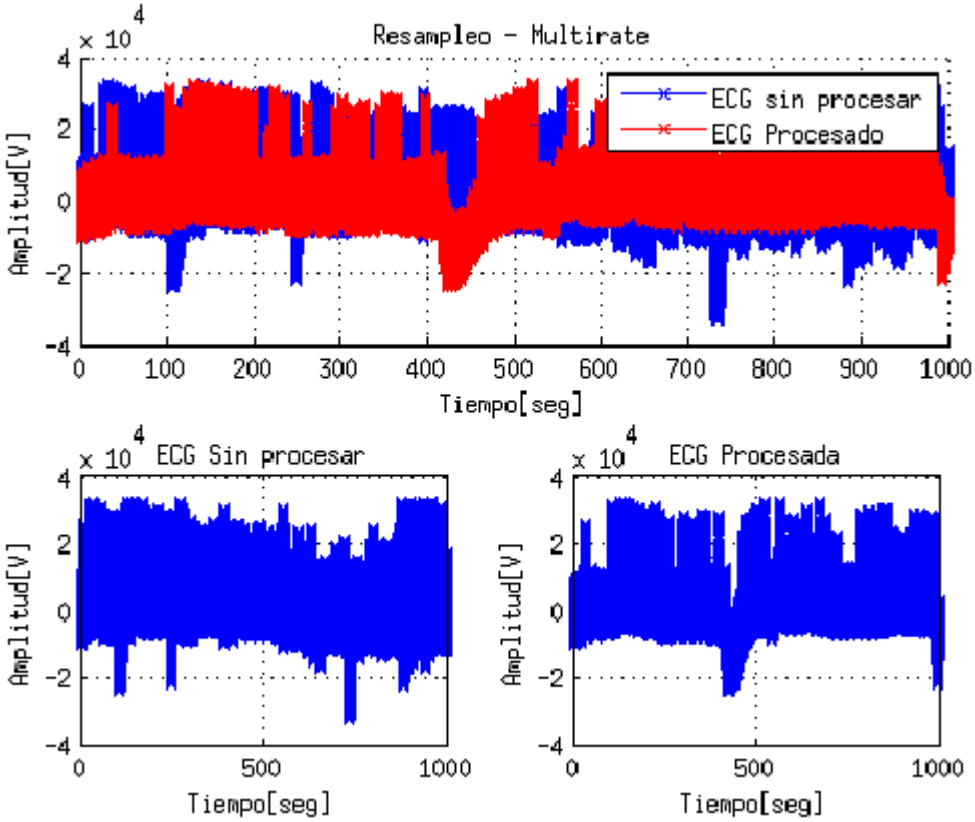


Observación: Del gráfico se observa que se encuentran variaciones significativas en la morfología de la señal.

Para los tres instentes de la grabación se tomaron 2 (dos) segundos, por lo que se tomaron los instentes de tiempo:

- Inicio de la grabación (instante 0) al segundo dos.
- Segundo 600 de la grabación al segundo 602.
- Segundo 1120 de la grabación al segundo 1122.

De los gráficos expuestos y el expuesto a continuación:



Se puede observar que existen variaciones en la morfología de la señal junto a un desfasaje. Esto es producto de las aproximaciones realizadas en cuanto al diezmado y la interpolación junto a la cantidad de filtros por los que la señal debe pasar.

Para observar el correcto funcionamiento del algoritmo planteado se procedio a tomar los valores arrojados a la salida del filtro, luego se comparo dichos resultados con los obtenidos a la salida del filtro en el ejercicio 1 del presente trabajo práctico obteniendo los siguientes resultados:

Media	2.821387
Desvio	3.434437

Estos valores son el error en las amplitudes de las muestras respecto de las primeras 456 muestras de la grabación.
Del anterior análisis se desprende como conclusión que el algoritmo fue desarrollado de forma correcta.

A continuación se exponen los tiempos de operación obtenidos al realizar la conversión de la tasa o frecuencia de muestreo de la señal adjuntada al informe.

Tabla comparativa de tiempos de operación

Fs' [Hz]	Rango [Segundos]	Tiempo de Operación [Segundos]
250	0 – 2	0.4193
2000	0 – 2	0.4176
1100	0 – 2	0.4091
250	600 – 602	0.4325
2000	600 – 602	0.4036
1100	600 – 602	0.4159
250	1120 – 1122	0.4343
2000	1120 – 1122	0.4170
1100	1120 – 1122	0.4062

Nota: Esta tabla se utilizará como soporte para desarrollar las conclusiones pertinentes a la siguiente tabla por lo que se pospondrán las conclusiones pertinentes.

Tabla comparativa de tiempos de operación de los tres ejercicios realizados

		Ejercicio 1	Ejercicio 2	Ejercicio 3
Fs' [Hz]	Rango [Segundos]	Tiempo de Operación [Segundos]	Tiempo de Operación [Segundos]	Tiempo de Operación [Segundos]
250	0 – 2	1.5448	1.4572	0.4193
2000	0 – 2	1.5737	1.4682	0.4176
1100	0 – 2	26.9900	24.7439	0.4091
250	600 – 602	1.5351	1.5514	0.4325
2000	600 – 602	1.5524	1.5622	0.4036
1100	600 – 602	24.3768	30.5501	0.4159
250	1120 – 1122	1.5789	1.4670	0.4343
2000	1120 – 1122	1.5263	1.5241	0.4170
1100	1120 – 1122	25.1118	24.8659	0.4062

A través del cuadro presentado se puede concluir que los tiempos de operación cuando se trabaja con estructuras polifásicas disminuye notablemente respecto a cuando no se utiliza estas.

Si bien éste método presento un desfasaje y una variación en la morfología de la señal original se puede ver una disminución el el tiempo de operación.

A partir de esta conclusión se puede observar un compromiso entre tiempo de operación y cambios en la morfología de la señal, motivo por el cual según el tipo de estudio que se desee llevar a cabo sera conveniente uno de los tres métodos expuestos.

Ejercicio 3 – Códigos

A continuación se expone el código principal generado para la resolución del ejercicio

Ejercicio 3 - Código: Función manejadora del ejercicio 3.

```
%**
% \fn [fp] = FuncEjercicio3(Signal,fs,fsx,tol,fp)
% \brief Función manejadora de ejercicio 3
% \author Pose, Fernando Ezequiel (fernandoepose@gmail.com)
% \date 2015.10.17
% \param  Signal          - Señal original
%   \param  Signal_Resample - Frecuencia de muestreo de la señal original
% \param  ts              - Frecuencia de muestreo de la señal modificada
%   \param  tol             - Tolerancia
%   \param  fp              - Número de imagen
%   \return fp              - Nuevo número de imagen

function [fp] = FuncEjercicio3(Signal,fs,fsx,tol,fp)

Signal_Resample = Resample3(Signal,fs,fsx,tol);

%N  = length(Signal);
%Nx = length(Signal_Resample);

%ts  = 0:1/fs:(N-1)*1/fs;
%tsx = 0:1/fsx:(Nx-1)*1/fsx;
%fp = Ploteo1(Signal,Signal_Resample,ts,tsx,fp);

%fp = fp+1;

end
```

Bibliografía utilizada

[1] Tratamiento de señales en tiempo discreto. Alan V. Oppenheim, Ronald W. Schafer. Ed. Pearson. Tercera edición.

[3] Understanding Digital Signal Processing. Richard G. Lyons. Ed. Prentice Hall PTR

[4] Tratamiento digital de señales. Johm G. Proakis y Dimitris G. Manolakis. 4°Ed. Pearson Prentice Hall.