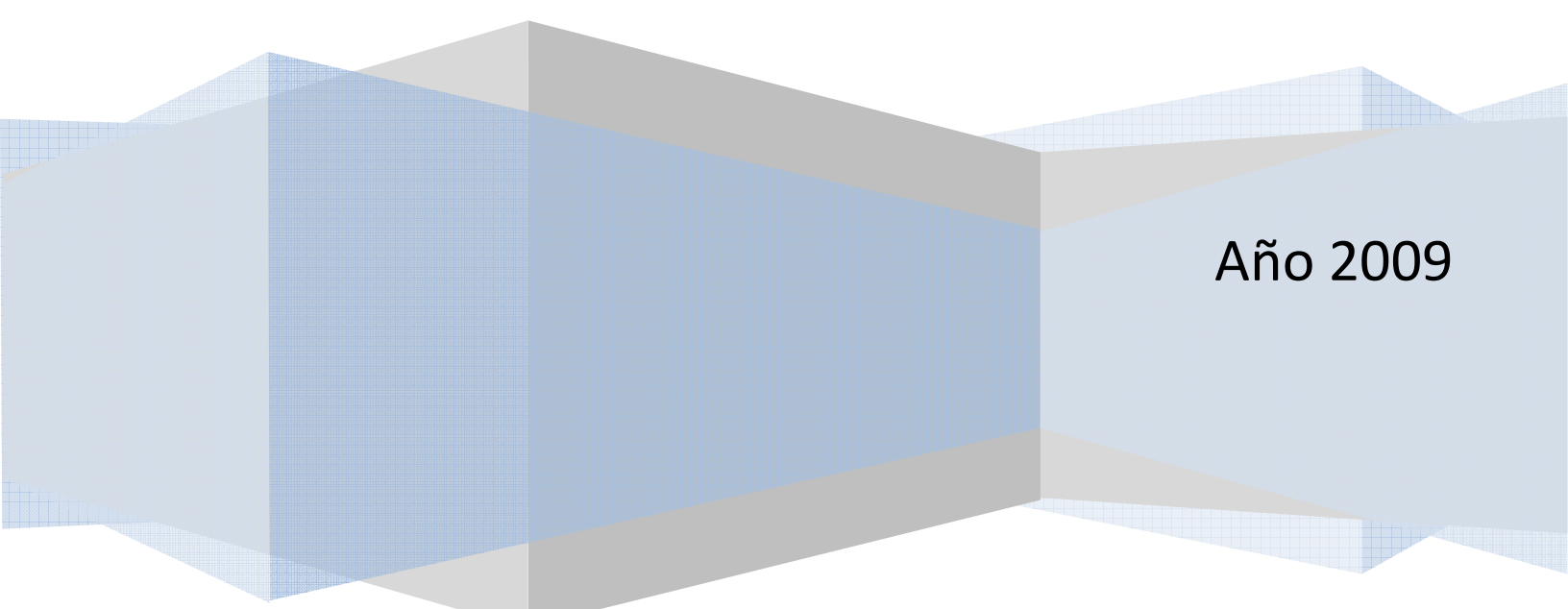


Procesamiento Digital de Señales en Tiempo Real

Análisis e implementación de algoritmo de detección de complejos QRS en tiempo Real

Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

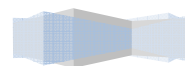
Jorge Escolá



Año 2009

Tabla de contenido

Objetivo:.....	3
Introducción:	3
Investigación y diseño del algoritmo ⁽¹⁾ :.....	4
Implementación en Matlab:.....	6
Filtrado:	7
Derivación:	8
Filtrado no lineal:	9
Integración:	9
Toma de decisión:	11
Resultados:.....	12
Implementación para BlackFin:.....	13
Diseño del algoritmo en tiempo real:	13
Adquisición de la señal:.....	13
Filtrado:	15
Procesamiento de la señal y toma de decisión:.....	17
Diagrama del proceso completo:	20
Análisis de amplitudes en formato de punto fijo:.....	20
Análisis de tiempos:	27
Referencias:.....	29



Objetivo:

El objetivo que se busca alcanzar en el presente trabajo es la implementación en una arquitectura BlackFin BF537 de un algoritmo de reconocimiento de complejos QRS en una señal de electrocardiograma (Figura 1), como punto de partida para trabajos posteriores, como ser la obtención del período RR en tiempo real, la señal de variabilidad de ritmo cardíaco, el reconocimiento de arritmias o la detección de los complejos P y T, entre otros.

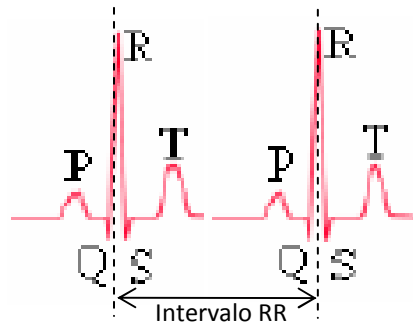


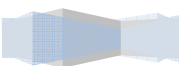
Figura1 : Modelo de la señal cardíaca (ECG)

Introducción:

La señal de ECG se obtiene colocando electrodos sobre la piel del paciente, en diferentes puntos de referencia, que se conocen como derivaciones. Los electrodos miden, por lo tanto, los potenciales que se generan a lo largo del tiempo en esos puntos específicos. Dichos potenciales se condicionan con la despolarización y repolarización de las células musculares a lo largo del tiempo. Esta señal es por lo tanto una mezcla de una gran variedad de señales, como pueden ser las señales que producen los músculos de la respiración al contraerse, el movimiento de diferentes partes del cuerpo, y otros potenciales que se inducen en el circuito de adquisición.

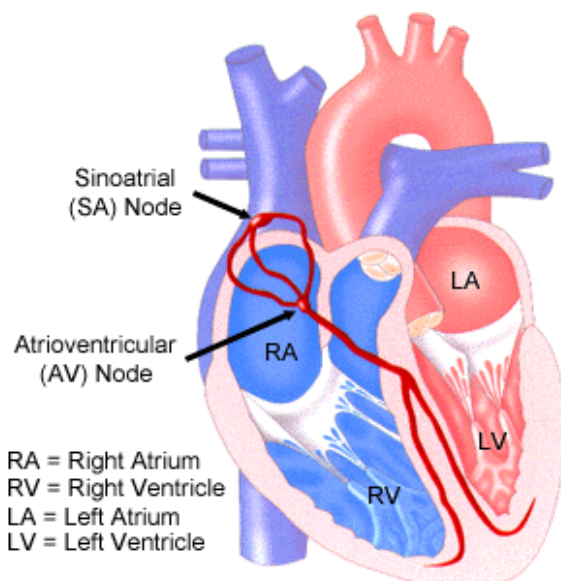
Se busca por lo tanto un algoritmo que sea capaz de separar las señales interferentes de la señal de interés, que tiene una forma similar a la mostrada en la figura 1, y poder luego detectar cuando se produce el complejo que se conoce como QRS, que no es otra cosa que el potencial eléctrico que se genera durante la despolarización de las células ventriculares (sístole).

A partir de esta señal pueden obtenerse diversas características del funcionamiento del corazón, y detectarse arritmias y cardiopatías variadas.



Investigación y diseño del algoritmo⁽¹⁾:

Para el correcto diseño del programa, se necesita primero entender conceptualmente en qué consiste cada segmento de la señal, cómo se genera y que diferencia un segmento del siguiente. A continuación se agrega, por lo tanto, una breve explicación del funcionamiento que permitirá luego extraer conclusiones sobre la mejor forma de discriminar entre los diferentes complejos.

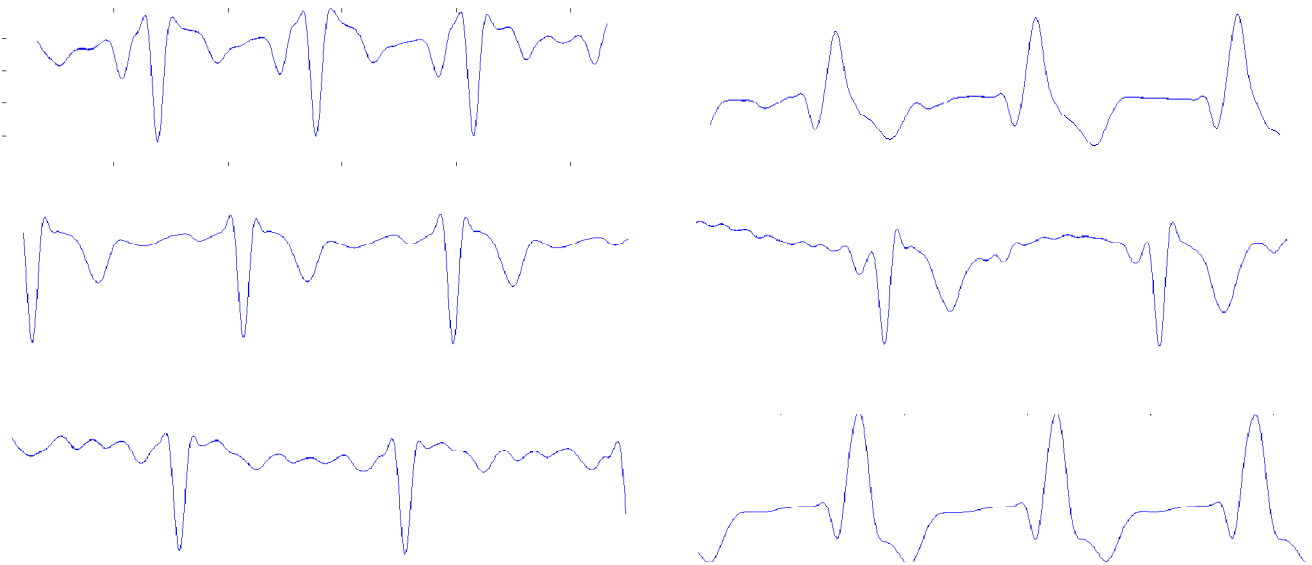


El corazón se contrae y relaja de acuerdo a un ritmo que viene dado por el nódulo sinusal, cuando funciona correctamente. Este nódulo es un conjunto de células (llamadas células marcapaso), que poseen la característica de despolarizarse (esto es, cambiar la polaridad de negativa a positiva en su interior) en forma espontánea. Esta despolarización es transmitida luego a través de todas las células cardíacas de las aurículas (cavidades superiores del corazón, encargadas de llenar de sangre las cavidades inferiores, o ventrículos), es retrasada por el nódulo auriculoventricular, y luego llega a las células de los ventrículos, generando su contracción y la consecuente expulsión de la sangre de su interior. Esta contracción debe generar un impulso muy grande, de manera que la sangre pueda alcanzar todos los capilares del cuerpo. El período durante el que transcurre se denomina sístole. Una vez terminada la sístole, las células vuelven a recobrar su potencial anterior (en forma más lenta), y el músculo su tono de reposo. El período en el que el corazón se encuentra en reposo se denomina diástole.

Ahora bien, la señal cardíaca (tal como se la muestra en la figura 1) refleja estas polarizaciones y despolarizaciones. La onda que se destaca con la letra P se condice con la despolarización de las aurículas, y es más pequeño que el complejo QRS por ser generado por una superficie menor. La onda que se destaca con las letras QRS es justamente la despolarización de los ventrículos, y se destaca por ser un pulso de muy alta energía (en comparación con el resto) y muy veloz, ya que el mismo debe llegar a toda la superficie ventricular al mismo tiempo (con una

demora muy pequeña) para evitar turbulencias en el flujo de la sangre. Por último, la repolarización de las células está dada por la onda T, que como se ve puede ser grande en amplitud, como es grande la excursión de tensiones, pero se caracteriza por ser una onda lenta.

Basados en la explicación anterior, y teniendo en cuenta que la onda puede presentar una gran cantidad de variaciones al obtenerse en pacientes en forma práctica (inversión de la onda R, desaparición de las ondas P o T, mayor amplitud en la onda T –ver figura 2 –), podemos llegar a la conclusión de que la mejor forma de diferenciar al complejo QRS del resto de las variaciones de la señal cardíaca es su alta velocidad y energía, expresadas en la onda como una pendiente y amplitud elevadas.

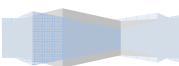


**Figura2 : Diferentes formas de señales de ECG reales
(Obtenidas de la base de datos de arritmias de MIT/BIH)**

El algoritmo deberá entonces analizar la pendiente (variación de la amplitud en función del tiempo) de la señal, así como también la energía (amplitud acumulada a lo largo de una ventana de tiempo) de dicha señal, para luego poder decidir si el segmento de la señal analizado corresponde o no a un complejo QRS⁽²⁾.

Sin embargo, queda por considerar también el ruido y las señales que pueden interferir a la correspondiente al ECG. Las mismas no aportan información sobre la posición del complejo que se desea detectar, ni tampoco de la actividad cardíaca en general, por lo que las mismas deberán ser eliminadas (filtradas) al comienzo del algoritmo.

Un esquema genérico de un algoritmo de detección de QRS quedaría entonces como se muestra:



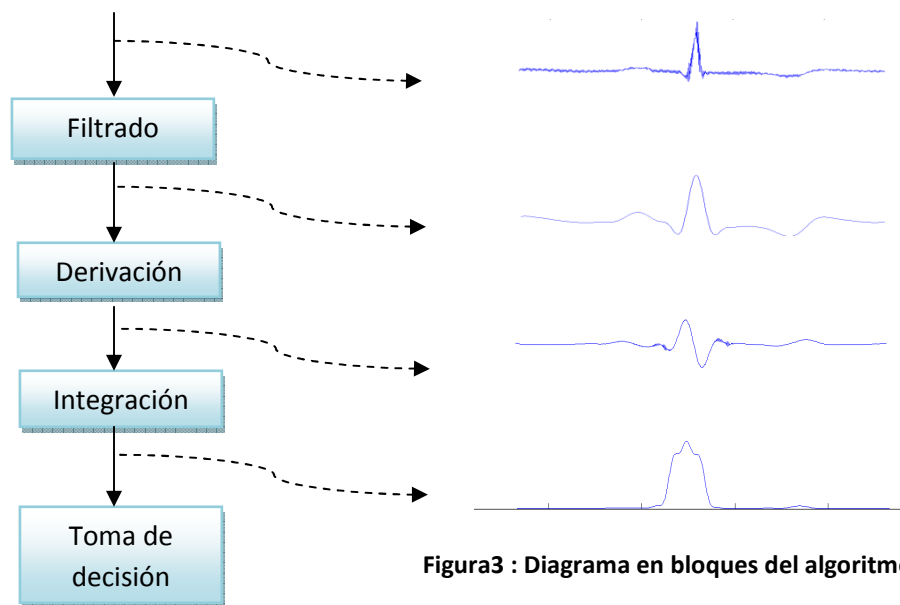


Figura3 : Diagrama en bloques del algoritmo

Implementación en Matlab:

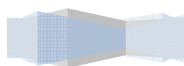
La implementación en Matlab difiere principalmente de la aplicación final en dos aspectos, a saber:

- 1) El desarrollo del programa no es en tiempo real, sino que es un análisis de una señal almacenada previamente en memoria, por lo que no hay limitaciones de tiempo.
- 2) La arquitectura de la PC soporta una codificación en punto flotante, por lo que desaparecen las restricciones de rango que presenta el punto fijo.*

El desarrollo del algoritmo en Matlab tendrá como objetivo el análisis de las formas de la señal y la eficacia del algoritmo, siendo la plataforma muy versátil y cómoda para realizar un gran número de pruebas con diferentes señales en un corto tiempo.

*Esta restricción solo será tomada en cuenta para las primeras evaluaciones, ya que si bien es cierto que la arquitectura soporta la operación en punto flotante, se restringirán los tipos de datos del Matlab a objetos similares a los de la arquitectura Blackfin en pruebas posteriores para verificar la eficacia del programa en cuanto a la utilización del rango dinámico del procesador.

A continuación se detallan los diferentes bloques del programa, las consideraciones que se tuvieron con los mismos y los resultados obtenidos:



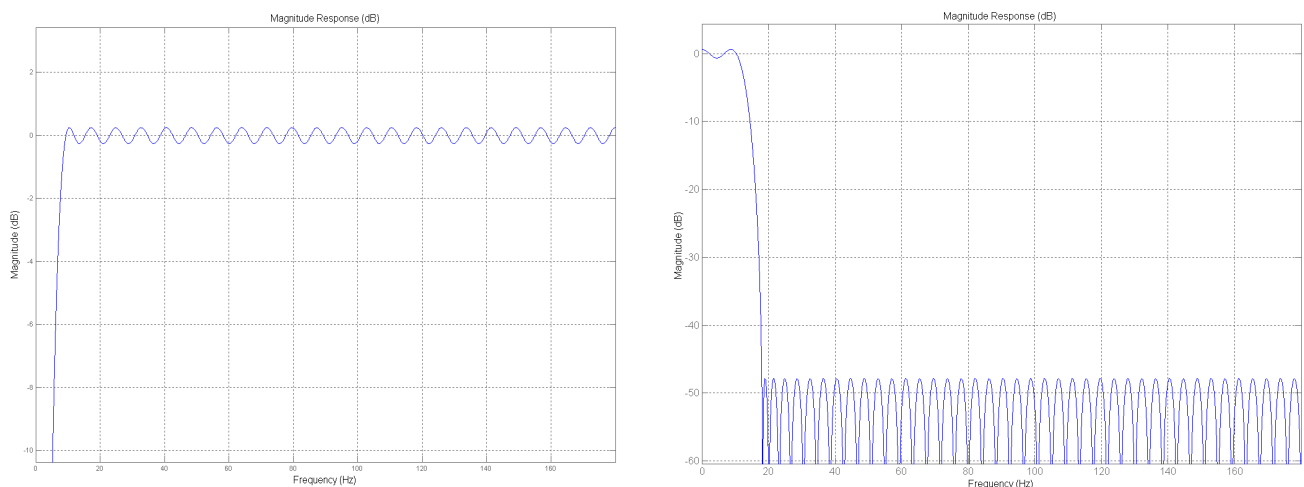
Filtrado:

De acuerdo a los trabajos de Thakor⁽³⁾, el rango de frecuencias en donde se encuentra la mayor densidad de energía para el complejo QRS es se encuentra centrada en 10Hz. Se consideró inicialmente este valor para diseñar un filtro pasabandas, que permitiese eliminar el ruido de altas y bajas frecuencias. El mismo se realizó en base a un filtro pasabajos seguido por un pasaaltos (para conseguir una mayor planicidad dada la estrecha banda de paso del mismo, de acuerdo a los trabajos de Rafael Gonzalez Díaz y Juan Antonio Quintana Silva).

El filtro pasabajos fue diseñado como un filtro FIR cuya frecuencia de corte fuese de 12Hz y su atenuación de 60dB. Se decidió comenzar con un filtrado de estas características para probar en primera instancia la lógica del algoritmo y luego el valor óptimo del filtro.

A continuación se diseñó también un filtro pasaaltos, cuya tarea era de eliminar el posible ruido de bajas frecuencias (la respiración, por ejemplo), y la componente de continua sobre la que se encuentra montada la señal.

Una vez probado el algoritmo y verificado su funcionamiento, se analizaron diferentes frecuencias de corte para reducir ruido en señales como las que se muestran en la figura 4, que perjudicaban el funcionamiento del algoritmo. Se modificaron también los parámetros de la frecuencia de corte y atenuación para lograr un filtro de menor cantidad de coeficientes, considerando que a mayor número de coeficientes del filtro mayor sería el tiempo de procesamiento de la señal. El filtro diseñado para alcanzar los resultados expuestos más adelante (incluido un rediseño final que se realizó para disminuir el orden del filtro y así mejorar el rendimiento del programa) es un filtro FIR pasabandas implementado a partir de un filtro pasaaltos y uno pasabajos, con frecuencias de corte en 5Hz y 15Hz respectivamente, de orden 90 en ambos casos. Su respuesta en frecuencia puede verse en la figura siguiente:



Tanto las frecuencias de corte como el orden de los filtros fueron importantes en el desarrollo del algoritmo, como se explicará más adelante.

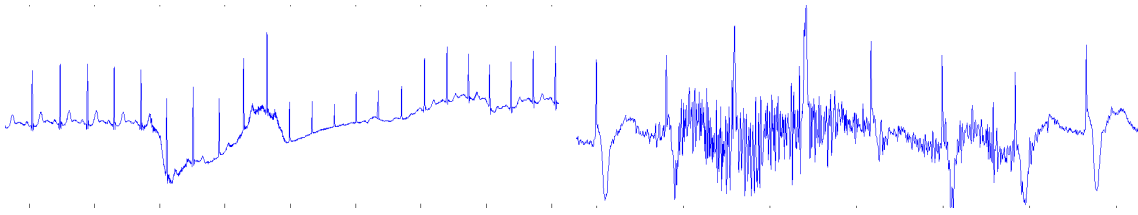


Figura4 : Ruido en las señales de ECG

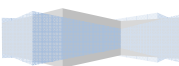
Derivación:

Una vez eliminadas las componentes de alta y baja frecuencia que no aportan información sobre el trabajo que se está realizando, se procede a encontrar una señal que refleje no la amplitud (energía) de la señal, sino la velocidad de variación de la misma con respecto al tiempo. La velocidad de variación (o derivada) de una señal es, por definición, la variación de amplitud de la señal en un determinado intervalo de tiempo. Mientras más pequeño pueda hacerse el intervalo de tiempo analizado, mayor será la semejanza de la señal obtenida a la velocidad instantánea de la señal.

En este punto nos encontramos con una limitación natural al estar trabajando con señales discretizadas en el tiempo, que es que el menor intervalo de tiempo está limitado a la inversa de la frecuencia de muestreo con la que se obtuvo la señal. Aplicando la definición enunciada anteriormente, la derivada de la señal (o su equivalente más próximo) puede obtenerse como:

$$\frac{dF(t)}{dt} \sim \frac{(F(t+\Delta t) - F(t))}{\Delta t} = (senial(i + 1) - senial(i)) * fs$$

Este es el procedimiento que se utilizará para obtener la derivada de la señal original, con la salvedad de la multiplicación por la frecuencia de muestreo, que si bien permite dimensionar correctamente la escala de la derivada, no aporta información a la forma de la señal, y será descartada (más adelante se analizarán los rangos numéricos para la implementación en una arquitectura de punto fijo, y los cambios de escala necesarios).



Filtrado no lineal:

Este paso, omitido en la descripción general del algoritmo, surge como una necesidad de discriminar en forma sencilla los cambios de pendiente abruptos pero de corta duración (como ser ruido en la señal), o bien señales de pendientes menores a las buscadas, pero aún así significativas (ondas T o P, típicamente) de la señal obtenida luego del paso anterior.

Dadas las características de la señal que se analiza, una operación sencilla que permite alejar en forma relativa los picos de gran magnitud de aquellos de pequeño valor es la elevación al cuadrado. Esta operación tiene además la ventaja (para el procesamiento posterior) que convierte cualquier valor en un valor positivo, lo cual es deseable para el algoritmo que se plantea, ya que a continuación se realizará una integración para analizar la energía de cada uno de estos complejos, para lo cual no influye el signo de la pendiente sino simplemente su magnitud.

En la figura siguiente pueden verse los resultados de la elevación al cuadrado, en donde queda de manifiesto la utilidad del bloque.

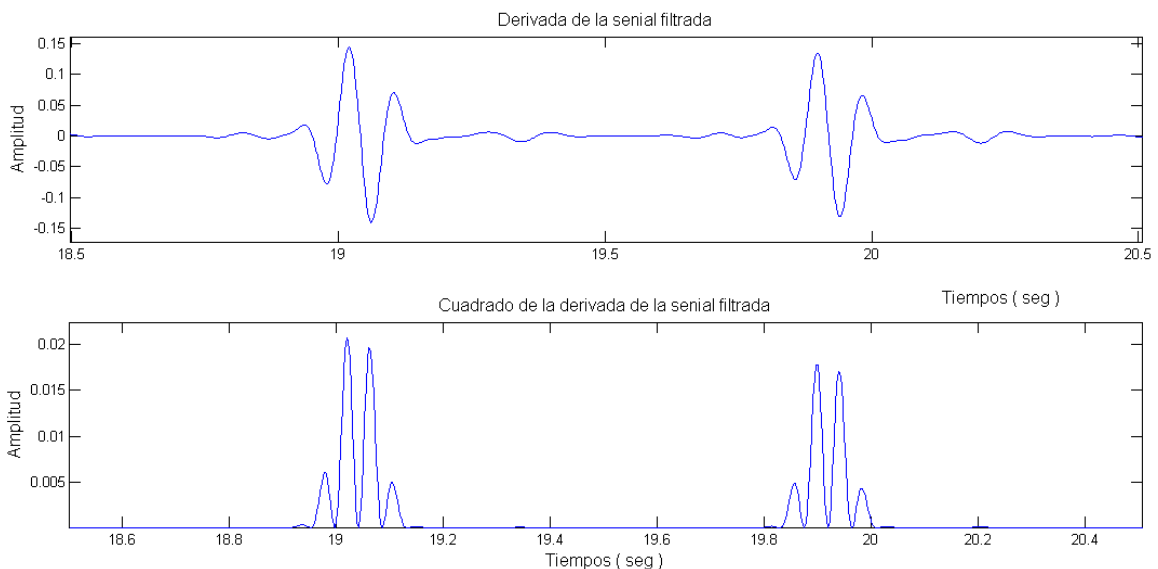


Figura5 : Filtrado no lineal

Integración:

Luego de encontrar la velocidad de variación de la señal, hay que hallar una forma de poder distinguir aquellos puntos en los que la pendiente es mayor, que nos oriente sobre la posición aproximada del pico del complejo QRS. Si bien podría simplemente implementar un algoritmo que busque los máximos de esta señal como una aproximación tentativa, se debe recordar que la señal puede contener ruido o interferencias de otras

señales que puedan generar también una derivada de un valor importante. Sin embargo, una característica de la señal buscada es que la misma posee no solo una alta pendiente ascendente, sino una también elevada pendiente negativa a continuación. Esto significa que se generará no uno, sino dos picos en la señal analizada, y el punto medio entre ambos (el cero de la derivada de la función original) se correspondería con el máximo del complejo QRS.

Dadas las características enunciadas en el párrafo anterior, se puede implementar de forma sencilla la detección de estos fenómenos integrando en un intervalo de tiempo (integración por ventana móvil) la señal. Esto significa analizar cualitativamente la energía de un determinado sector, o ventana, de la señal, para luego mover en forma lineal la ventana y realizar el mismo procedimiento. Si se elige un intervalo de tiempo adecuado, o ancho de ventana, tal que en el punto entre los picos antes descritos los mismos sean parte de la integración, se obtendrá un máximo justamente en este punto, y será nuestra guía para la detección del punto R. De esta manera, se vuelve importante la elección de una ventana de integración de un ancho apropiado, que debe condecirse con el ancho promedio del complejo QRS para evitar captar fenómenos indeseados. De aquí que este valor debe ser adoptado de en un rango de 50 a 110mseg⁽⁵⁾. Se optó por un valor de 80mseg de ancho de la ventana de integración, siendo este valor el promedio de los antes enunciados, y por lo tanto aquel valor que asegurará el menor alejamiento de la condición ideal de la ventana para la mayoría de las situaciones, lo que resulta útil para un tipo de análisis cualitativo como es el presente.

El funcionamiento del algoritmo de integración por ventana móvil y sus resultados pueden verse en las figuras siguientes:

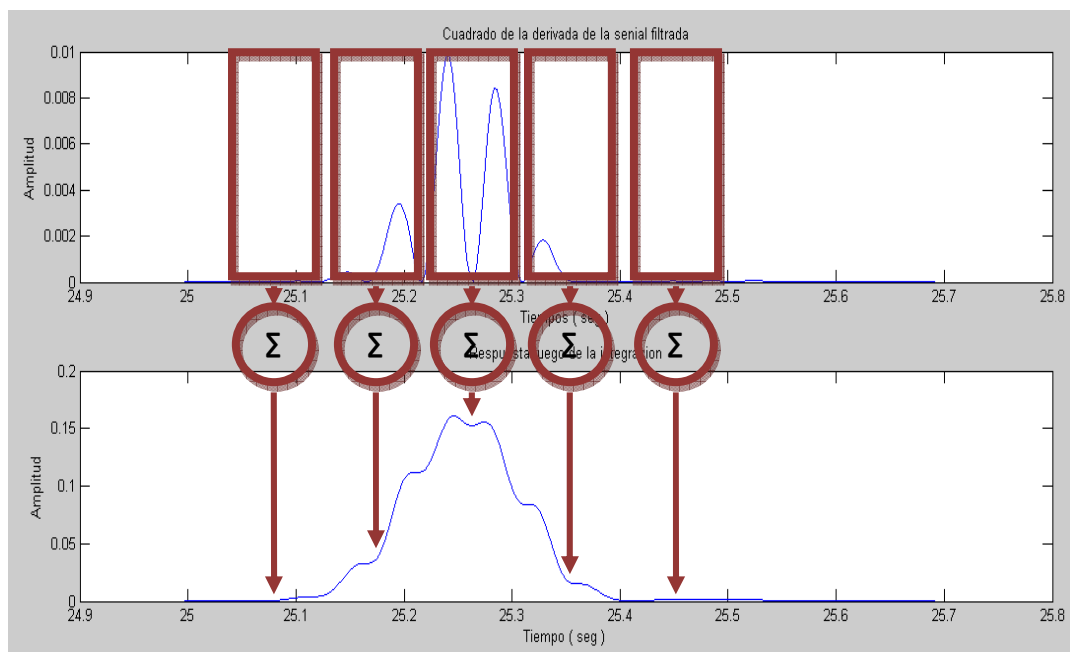


Figura6 : Funcionamiento y resultado de integración por ventana móvil

Toma de decisión:

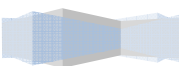
Finalizado el procesamiento de la señal inicial se debe proceder a distinguir los picos de la respuesta a la integral, para luego identificar finalmente el valor correspondiente al punto R del complejo QRS con la mayor precisión posible.

Si bien parecería trivial con una señal como la mostrada en la figura 1, el electrocardiograma puede variar mucho de paciente a paciente, y particularmente la zona del complejo QRS es muy inestable, pudiendo haber 2 picos en lugar de uno, o un solapamiento con la onda T, una inversión del pico o la intensificación de los valles indicados como Q y S en la figura 1.

Todo lo anterior lleva a que no alcance con detectar el máximo valor de la respuesta a la integración, sino que se debe hacer un análisis del entorno a dicho pico para analizar tanto la pendiente de la señal, como la amplitud (energía) de la misma en ese intervalo, y finalmente poder elegir aquel punto que más se aproxime a la definición del punto R de una señal cardíaca, considerando como tal al valor máximo (en módulo y con respecto al nivel de continua de la señal) de la señal cardíaca con una pendiente mayor a las de las ondas P y T, y que se encuentre en medio de estos complejos.

Este algoritmo de toma de decisión se realizó de la siguiente manera:

- Se buscan en primera instancia los picos de la señal de integración por ventana móvil, como aquellas zonas en las que se supera un cierto nivel de threshold, que es evaluado en primera instancia como un valor proporcional al valor medio de la señal (que tiene también en cuenta el ruido, de acuerdo al procedimiento que se describe en los trabajos de Brij N. Singh y Arvind K. Tiwari⁽⁶⁾ y Fayyaz A. Afasar y M. Arif⁽⁷⁾). En caso de no encontrarse un pico, se repite la operación con un threshold menor, hasta alcanzar un valor de threshold suficientemente bajo como para asegurar que no hay un pico en la señal analizada.
- Una vez encontrados los picos, se hayan los instantes de tiempos correspondientes a los valores máximos de la señal (evaluada en módulo) en la mitad inferior y superior del nódulo.
- Finalmente se comparan las pendientes correspondientes a estos picos, para evaluar si se trata de un pico de gran amplitud correspondiente a un complejo T, o un ruido de la señal.
- Una vez establecido el lugar del posible complejo QRS, se evalúa la diferencia de tiempo con respecto al último pico detectado, y en relación al promedio de pulsaciones por minuto que se actualiza en tiempo real con cada pico detectado, se evalúa si el pico se corresponde con un complejo normal o bien si se trata de una arritmia o un ruido en la banda de paso del filtro. Los tiempos máximos y mínimos fueron analizados de acuerdo a la publicación de Carlos Serranos⁽⁵⁾.



Resultados:

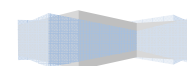
Con el objetivo de probar el funcionamiento del algoritmo y contrastar los resultados obtenidos contra una referencia, se utilizaron las señales 100m a 115m de la base de datos del MIT/BIH, con sus respectivas anotaciones (marcas de latidos, arritmias y anomalías).

Para medir la eficacia del algoritmo en el reconocimiento de complejos QRS, se adoptó como margen de error un intervalo de +/- 5 muestras (14mseg)^(*) en relación con los resultados de la base de datos. Se considerará por lo tanto que un pico está detectado correctamente si se encuentra en un entorno dentro del margen de error permitido del resultado de la base de datos.

A continuación se muestran los resultados de los relevamientos de 5 minutos de cada una de las muestras (entre 300 y 450 latidos por muestra), detallando los errores en la detección:

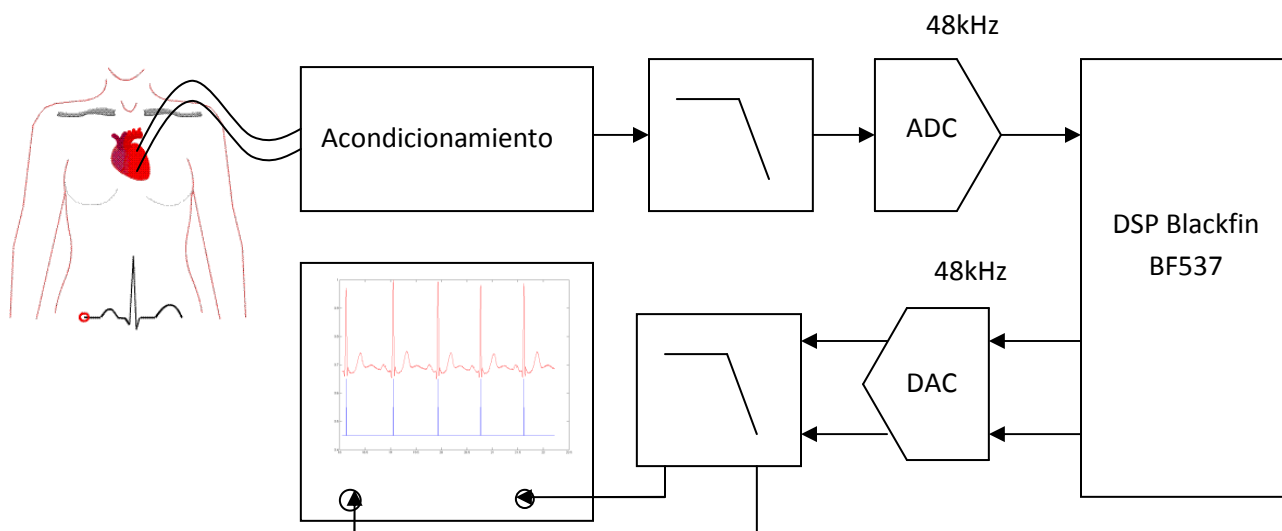
<i>Señal</i>	<i>Cantidad de errores</i>	<i>Rango de diferencias</i>	<i>Observaciones</i>	<i>Eficacia</i>
100m	0	0 - 3		100,00%
101m	1	18		99,67%
102m	1	14		99,72%
103m	0	0 - 3		100,00%
104m	12	22 - 46		96,76%
105m	0	0 - 3	Se detectó una arritmia. Muestra 119988	100,00%
106m	2	14	Los 2 errores detectados son arritmias	99,36%
107m	5	30 - 37		98,58%
108m	12	13 - 14		95,72%
109m	5	12 - 17		98,82%
111m	21	13 - 16		94,00%
112m	5	11 - 12		98,82%
113m	0	0 - 2		100,00%
114m	0	0 - 5	Muchas arritmias en la señal	100,00%
115m	0	0 - 3		100,00%

^(*) Este valor fue seleccionado de acuerdo a la variación promedio que poseen las muestras generadas en la base de datos entre complejos de morfologías similares.



Implementación para BlackFin:

Posteriormente a las pruebas realizadas tal como fue descrito en las secciones anteriores, se procedió a la implementación del sistema en tiempo real. Para esto, se contó en primera instancia con un generador de señales cardíacas (luego se utilizó un circuito de acondicionamiento para obtener las señales del cuerpo del paciente), el cual fue conectado a la entrada analógica del kit de desarrollo ADSP-BF537, que utiliza como núcleo un DSP de Blackfin. La señal es adquirida por uno de los canales del ADC, a una frecuencia de muestreo de 48kHz. Esta señal es luego procesada como se describirá en las secciones siguientes, y el resultado del procesamiento se obtiene a través de los dos canales del DAC del kit de desarrollo, los cuales poseen idéntica frecuencia de muestreo que la entrada. En uno de los mismos se ve la entrada reflejada, mientras que en el otro se envía una señal que representa los puntos en los que se detectó un complejo QRS mediante un pulso, de manera de poder contrastar los resultados con la señal original, utilizando un osciloscopio.



Diseño del algoritmo en tiempo real:

Como fue explicado anteriormente, se necesitó de un rediseño en el esquema del algoritmo para que el mismo pueda funcionar con una señal que se recibe en tiempo real y no se encuentra previamente almacenada en la memoria del dispositivo. El esquema del algoritmo final se detallará para cada bloque del programa por separado, indicando las diferencias con lo explicado hasta este punto.

Adquisición de la señal:

La adquisición de la señal cardíaca dependerá enteramente del circuito que se use para tal fin. Es necesario tener ciertas consideraciones con respecto al mismo, siendo que el hardware que se utilice cambiará las características de la señal que llega al DSP para su

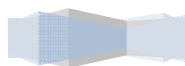
análisis. El mismo deberá entregar una señal que excursione entre $-3.3V$ y $+3.3V$ como valores mínimo y máximo respectivamente, no siendo necesario un filtro pasabajos a la salida del circuito gracias al filtro antialiasing que posee incorporado el circuito de adquisición del hardware utilizado.

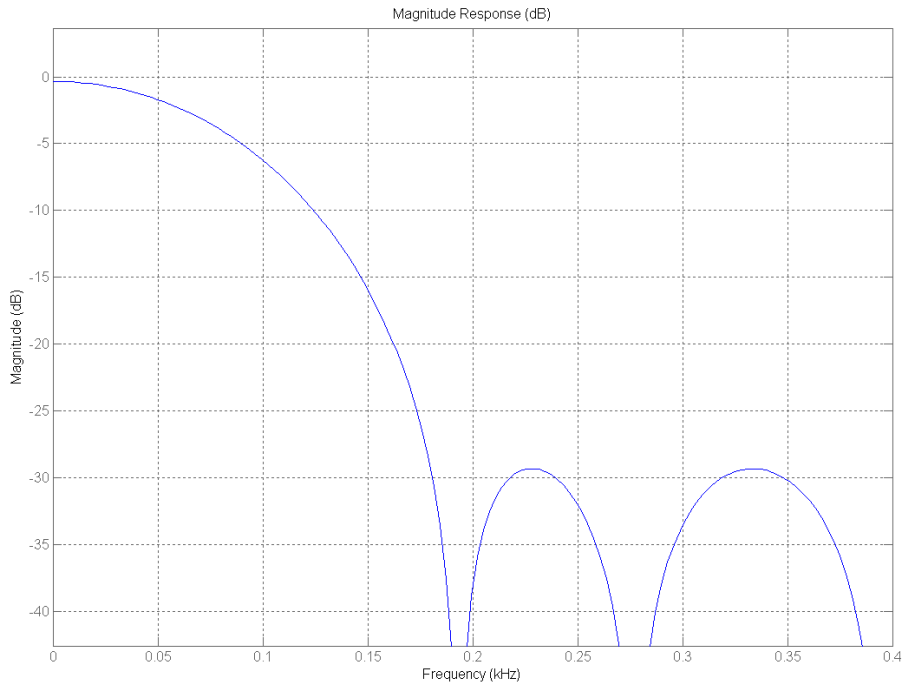
Por otro lado, y considerando que las señales de prueba que se utilizaron poseen una frecuencia de muestreo de $360Hz$, y que la frecuencia de muestreo impacta necesariamente en el diseño de los filtros y los tiempos de procesamiento del programa, se necesita llevar la señal adquirida, ya muestreada por el conversor analógico digital (que en el caso del BlackFin BF537 posee una frecuencia de muestreo fija de $48kHz$) a una frecuencia de $360Hz$. Esto se realizó mediante una etapa de downsampling previa al almacenamiento de la señal, y una etapa de upsampling que eleva la frecuencia de muestreo de la señal previa a enviarla al conversor digital analógico a la salida.

Estos bloques consisten, en el caso del downsampling, de un bloque que dejará pasar una de cada N muestras (donde $N=133$, para una frecuencia de muestreo de $48kHz$ y una frecuencia de procesamiento de $360Hz$). Sin embargo, teniendo en cuenta que esto supone volver a muestrear la señal a una frecuencia diferente, se necesita un filtro pasabajos antialiasing, con frecuencia de corte en la mitad de la frecuencia de muestreo ($180Hz$), para evitar que aparezcan fenómenos de alias o solapamiento de espectro.

El diseño de este filtro, que debe cortar cualquier componente por encima de $180Hz$, y posee una frecuencia de muestreo de $48kHz$, se realizó con una pendiente baja de disminución de la amplitud en función de la frecuencia, ya que de otra manera resultaría en un filtro de un gran número de coeficientes (debe bajar muy abruptamente teniendo en cuenta la frecuencia de muestreo).

Se diseñó un filtro FIR (que se utilizará también posteriormente en el bloque de upsampling) utilizando la herramienta fdatool, de Matlab, cuya frecuencia de paso sea ligeramente mayor a la mayor frecuencia de la señal (que se estableció en $20Hz$, teniendo en cuenta que el filtrado posterior posee una frecuencia de corte de $12Hz$), y frecuencia de stop en $180Hz$, con una atenuación de $30dB$ (teniendo en cuenta que las componentes en alta frecuencia en un circuito de adquisición de la señal cardíaca no son de amplitud comparable a la señal cardíaca puede optarse por una atenuación de una magnitud como la que se indica), cuyo espectro puede verse en la figura siguiente:

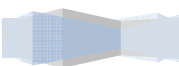




El bloque de upsampling, por otro lado, tiene en cuenta el hecho de que para enviar una señal muestreada a 360Hz por un DAC a 48kHz, se deben generar las muestras de más que se envían al convertor, en función de un criterio, que puede ser el de realizar una interpolación lineal, logarítmica, o polinómica. En cualquier caso, se generan componentes de alta frecuencia dados por el cambio abrupto de un punto al siguiente, deformando la señal a la salida. Dichas componentes deben ser eliminadas mediante un filtro pasabajos, que puede ser de características similares al filtro de downsampling, teniendo en cuenta que la máxima frecuencia que saldrá por el DAC será la máxima frecuencia de la señal procesada (que se corresponde con la frecuencia de paso del filtro diseñado), y de esta manera se evita la realización de un nuevo filtro, mejorando la eficacia del algoritmo al no tener que almacenar el doble de coeficientes y pudiendo realizar los 2 filtrados en paralelo gracias a la doble MAC que posee la arquitectura.

Filtrado:

A continuación debe considerarse una manera de implementar el filtrado de la señal. El inconveniente de realizar el filtrado de manera convencional, es que el mismo se implementa mediante una convolución de la señal a filtrar contra el kernel de un filtro. Una convolución trabaja realizando una operación matemática entre la señal y las N muestras anteriores de la misma, siendo N el tamaño del kernel del filtro. Esto implica que para obtener la señal correspondiente al filtrado de un vector de M muestras, se tendrá un vector de $N + M + 1$ muestras, en donde las primeras N muestras serán los resultados de la



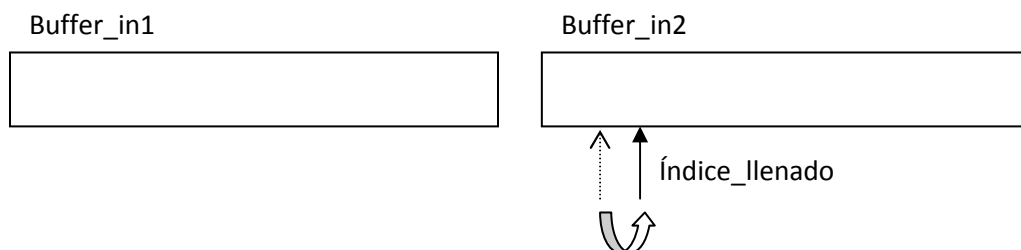
convolución contra valores anteriores de la señal (o si se trabaja con una señal sin memoria, será la convolución contra 0, tomando condiciones iniciales nulas), y no serán de utilidad para el resultado que se busca, sino simplemente para la correcta realización del método (se referirá en lo posterior a esto como “el transitorio” del filtrado).

Existen estrategias para evitar estos problemas en el filtrado en tiempo real, siendo la más común el almacenamiento de las N muestras anteriores en un vector (que se denominará “vector de delay”), de manera de que al realizarse el filtrado de una muestra se posea almacenado el valor de las N muestras anteriores y se evite la generación de un transitorio previo al filtrado, con el inconveniente de tiempos y memoria que esto genera.

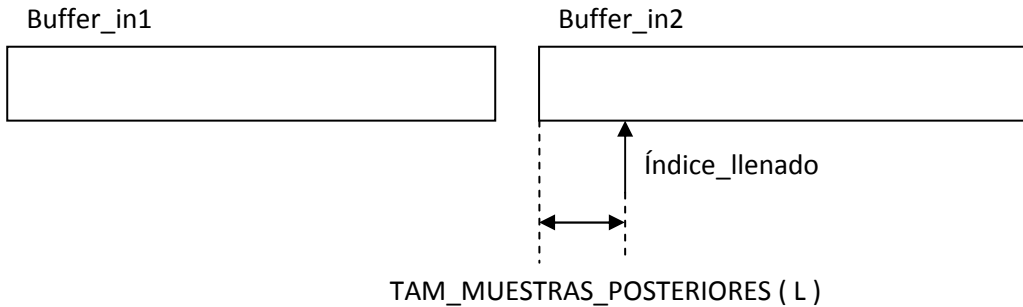
Un enfoque similar, pero más cómodo para la implementación posterior del algoritmo en cuestión, es poseer 2 vectores en los que progresivamente se irán cargando las muestras. Una vez completada la carga de un vector, se procederá a cargar las muestras en el siguiente, mientras se procesan las almacenadas en el primero de ellos. Esta técnica tiene la ventaja de que no hace necesaria la inicialización de todos los registros involucrados en la realización de la convolución cada vez que llega una muestra, sino al llegar las M muestras que posee el buffer. Por otro lado, siendo las muestras cargadas alternadamente en los buffers, se poseen los N valores anteriores de la señal en el buffer que se está cargando de muestras, que se utiliza de una forma similar al buffer de delay, para conocer las condiciones iniciales del filtro al momento de realizar la convolución.

Como será explicado posteriormente, serán necesarias también una cantidad (que llamaremos L) de muestras posteriores para el algoritmo de toma de decisión, por lo que la mecánica del programa será la de esperar al llenado de M muestras en un buffer para dar comienzo al procesamiento de la señal en el otro buffer, de acuerdo a como se ve en el siguiente diagrama:

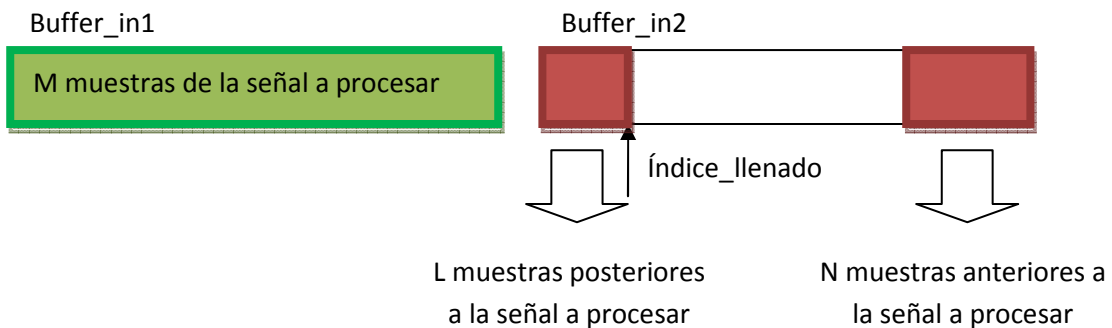
1. Cada vez que llega un dato, el mismo es colocado secuencialmente en un buffer (cuya posición se encuentra indicada por un puntero que llamaremos índice_llenado). Cuando el buffer en el que se están ingresando muestras se llena, se procede a llenar el siguiente. Al finalizar el llenado de este segundo buffer, se procede a llenar nuevamente el primero. Una vez que el dato fue colocado en la posición correcta, el puntero es incrementado, dejando la situación como se ve en la siguiente figura, preparada para la llegada del siguiente dato:



2. Cuando el puntero alcanza a estar a una distancia igual a la cantidad de muestras posteriores (cuya utilidad será explicada en el apartado siguiente) desde el comienzo del buffer que se está llenando, se comenzará el procesamiento, de acuerdo a lo que muestra la figura siguiente:



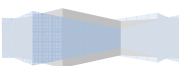
3. Finalmente, en el gráfico siguiente pueden verse cuáles son y donde se encuentran las señales de interés en los buffers de acuerdo a los diagramas anteriores:



Procesamiento de la señal y toma de decisión:

El procesamiento posterior al filtrado no posee mayores inconvenientes para la implementación en tiempo real del algoritmo. Resulta si importante tener en cuenta que, de acuerdo a la forma de reconocimiento que se plantea, que es la de encontrar un punto de las características enunciadas en el apartado *Implementación en Matlab / Toma de decisión*, alrededor de un entorno en el cual la energía obtenida posteriormente a la integración en un sector de la señal sea mayor a la de un nivel establecido (threshold), lo que se está haciendo es en realidad encontrar puntos que cumplan ciertas condiciones alrededor de “entornos críticos” en los que se presume hay complejos QRS, de acuerdo al procesamiento anterior.

Lo anterior posee un problema cuando se analiza la señal por bloques, que radica en que en los bordes de dichos bloques pueden darse las condiciones del entorno del punto en cuestión, pero sin embargo dicho punto puede encontrarse en el bloque siguiente o anterior. Este problema, que indicaré como “condición de borde del algoritmo” necesita por ende considerar las muestras posteriores y anteriores al bloque que se está



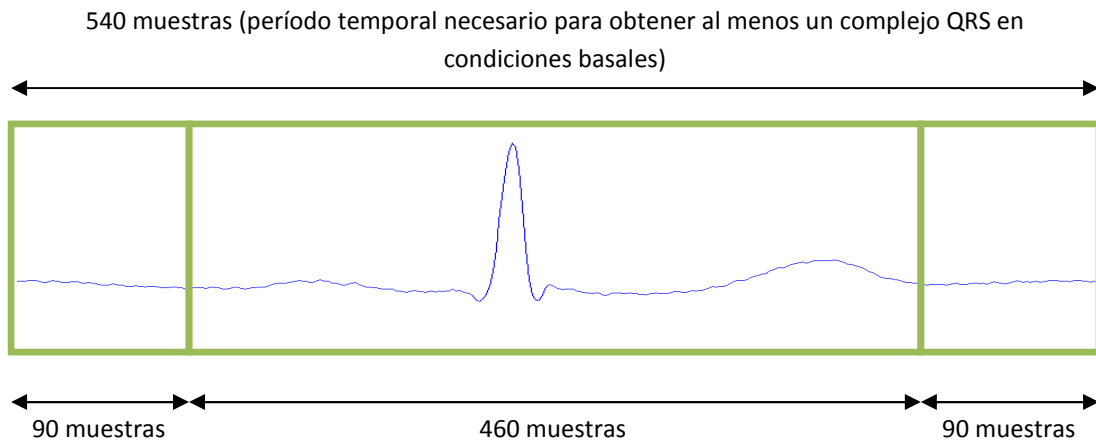
procesando, de manera de saber si el complejo se encuentra realmente en el bloque actual.

Esto obliga a procesar no sólo las M muestras del buffer que se llenó anteriormente, sino también un bloque de muestras posteriores y anteriores (que deberán ser de al menos el tamaño de un complejo QRS, para asegurar que no quedará un complejo “cortado” por la terminación de un buffer). De acuerdo a los tiempos analizados anteriormente para el intervalo QRS, podemos determinar que como máximo el bloque de análisis anterior y posterior deberá poseer una cantidad de muestras que sea análoga al menos a 110 milisegundos. Esto equivale a, aproximadamente, 40 muestras.

Se consideraron, sin embargo, ciertas cuestiones que tienden a optimizar la forma en que se realizó el código para seleccionar el valor de la cantidad de muestras anteriores y posteriores, que se enumeran a continuación:

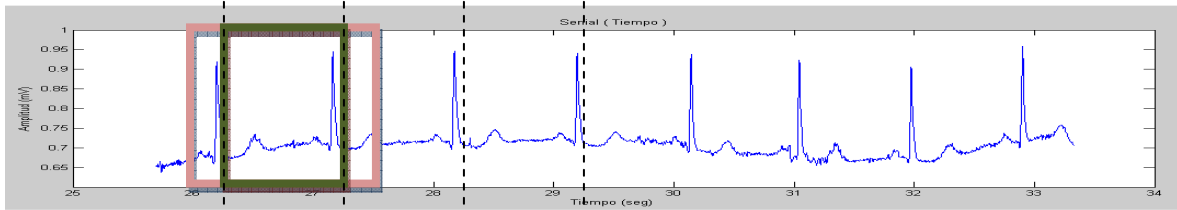
En primera instancia, resulta más sencillo (entendido como que se necesita una menor cantidad de código) que los buffers anterior y posterior sean de igual tamaño.

Por otro lado, y como una mejora adicional que se comprenderá mejor en el apartado de análisis de tiempos, es conveniente que en el bloque que se procesa haya al menos un complejo QRS, de manera de poseer una referencia con respecto al nivel de threshold que se adapta, y considerando una frecuencia cardíaca de 40ppm como un valor mínimo que puede poseer una persona en condiciones basales y una frecuencia de muestreo de la señal cardíaca de 360Hz, se traduce en que el bloque a procesar debería ser de, al menos, 540 muestras, lo que implica que, el buffer de análisis deberá ser de al menos $540 - (2 \times 90) = 460$ muestras. En el gráfico siguiente se explicita dicha limitación:

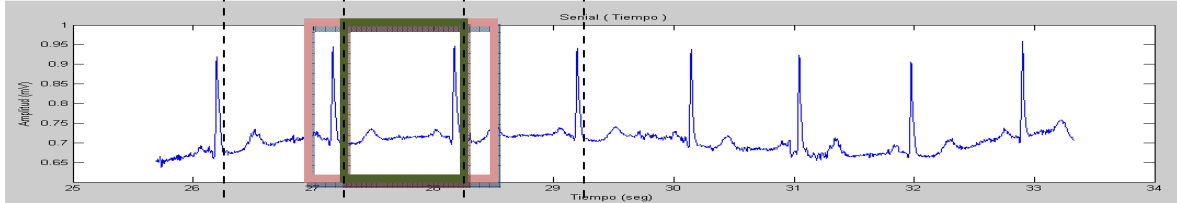


Todo lo anterior lleva a realizar un esquema de solapamiento, en el cual determinados bloques (que fueron comentados como buffers o vectores anteriores y posteriores en los párrafos anteriores) sean analizados 2 veces para llegar a una señal de salida correcta. Lo anterior se muestra en el siguiente gráfico, en el que se pretende mostrar el tamaño de los buffers con respecto al del bloque de la señal a procesar, y donde puede verse gráficamente la condición de bordes explicada anteriormente:

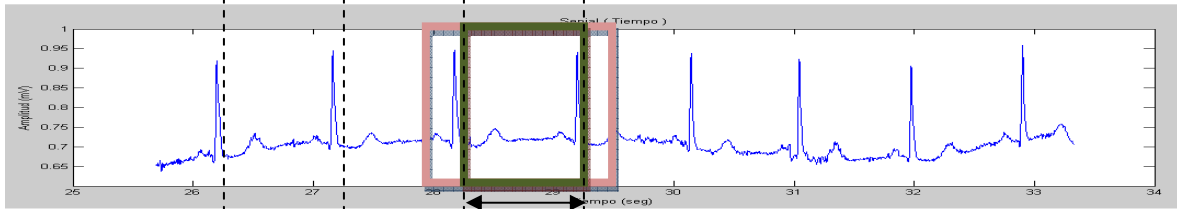
Procesamiento 1:



Procesamiento 2:

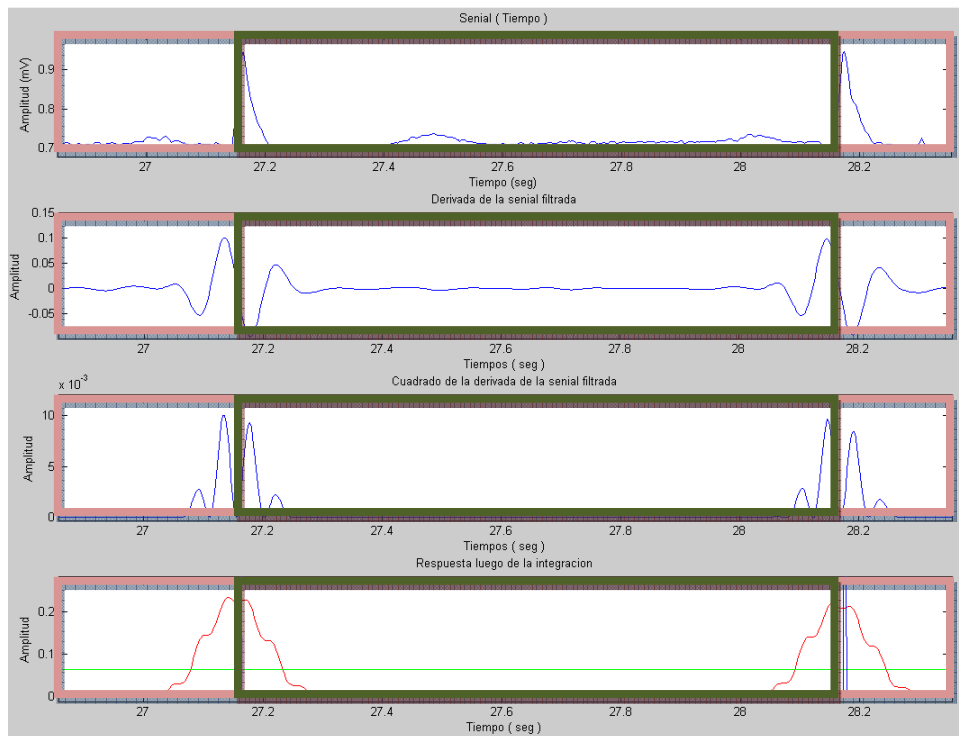


Procesamiento 3:

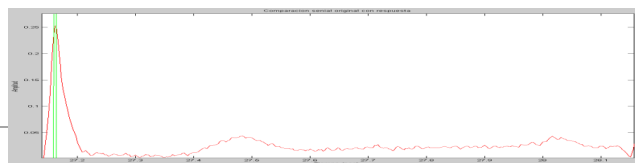


460
muestras

Ejemplo del procesamiento 2:



Salida del Algoritmo:



es en Tiempo Real |

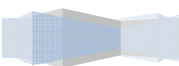
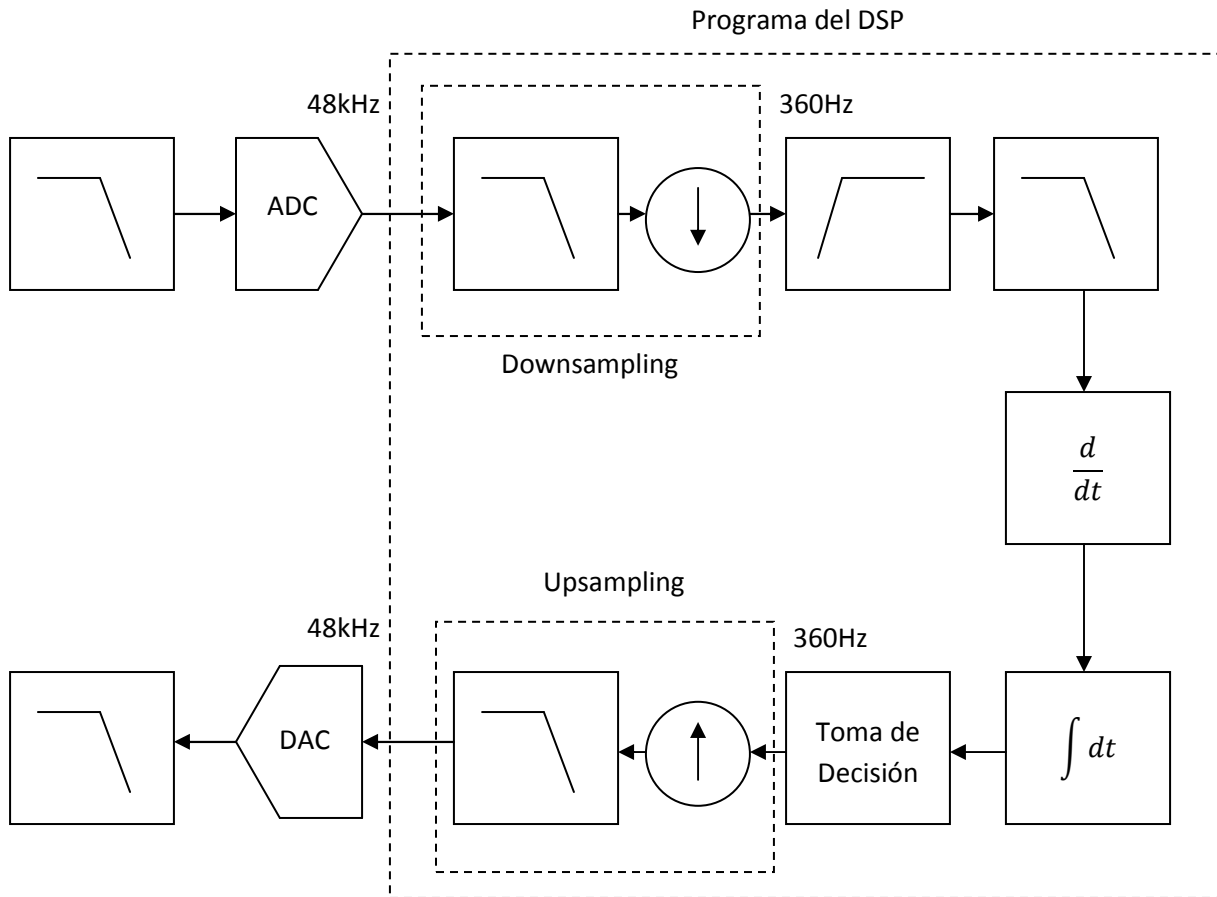


Diagrama del proceso completo:

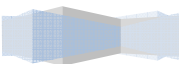
De acuerdo a lo explicado, el diagrama funcional completo del proceso que atraviesa la señal desde la salida del circuito que la genera hasta la salida del conversor digital analógico es la siguiente:



Análisis de amplitudes en formato de punto fijo:

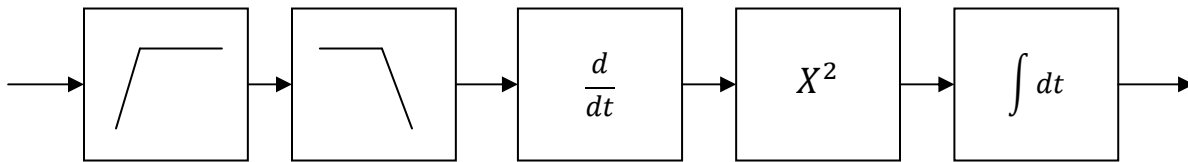
Teniendo en cuenta que el formato numérico que utilizan los registros del DSP seleccionado es el punto fijo con codificación 1.15 (1 bit para la parte entera y 15 para la parte fraccionaria), el rango de valores que los mismos pueden tener se encuentra supeditado a las limitaciones de este formato. Dada la codificación en complemento a 2, los máximos y mínimos valores representables con esta codificación son el $1 - 2^{-15}$ y el -1 , respectivamente. A los fines prácticos, esto significa que las señales no podrán tener una amplitud superior a ± 1 .

Esta conclusión supone un análisis del procesamiento de la señal, que deberá asegurar que se cumpla en todo momento la limitación enunciada.



Resulta más sencillo el análisis si se realiza del final hacia el principio, limitando la señal final a una amplitud máxima de 1, y analizando como optimizar la utilización del código sin exceder sus limitaciones.

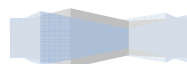
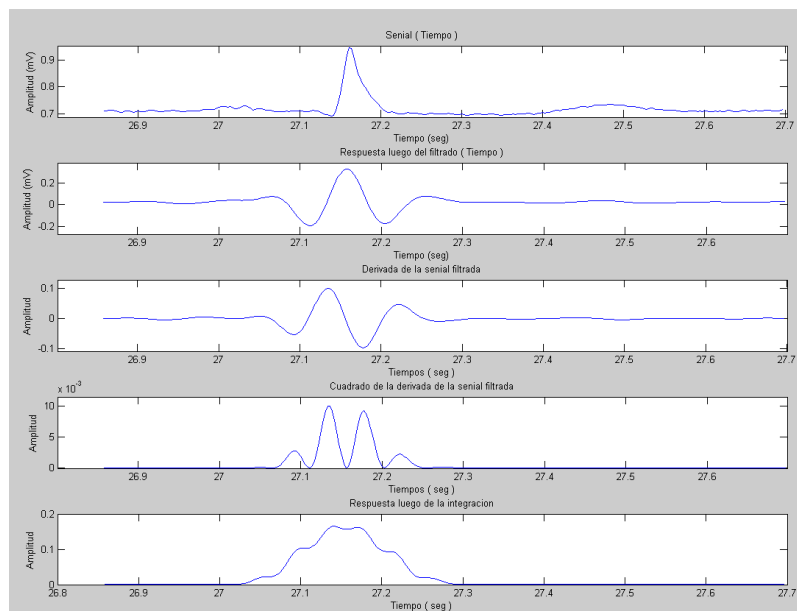
Para este análisis se partirá del diagrama en bloques del proceso, obtenido en el apartado anterior, cuya parte principal se muestra a continuación:



La condición para el mejor aprovechamiento del rango será que la salida de la etapa de procesamiento (la salida del bloque de integración) no exceda el límite del rango (el valor máximo sea unitario), y restringiremos la señal de entrada al sistema como la máxima señal que puede ser aplicada a la entrada, para luego analizar el mínimo valor de la misma.

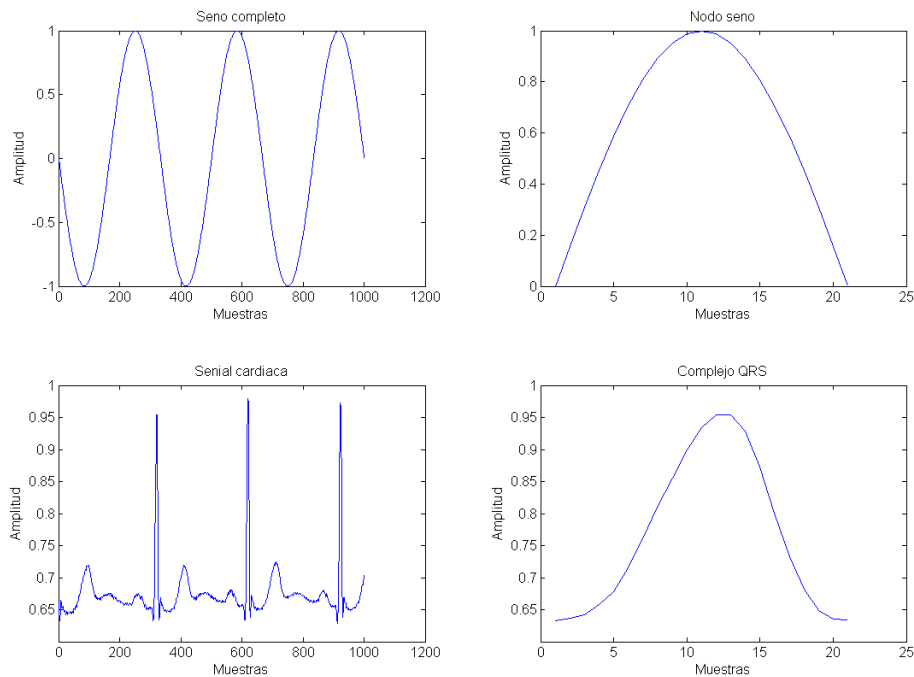
De acuerdo a lo expuesto en el párrafo anterior, supondremos que el mayor valor de la señal posterior a la integración será unitaria. A partir del conocimiento de las formas de onda de las señales en ese punto y en los puntos anteriores, intentaremos deducir por lo tanto el máximo valor admisible a la entrada del bloque de derivación (al comienzo del procesamiento de la señal), y a partir de este valor se analizará la posible inclusión de un bloque de atenuación o multiplicación de la señal, de manera de aprovechar mejor el rango dinámico del microprocesador.

La forma de las señales a la entrada y salida de los bloques de derivación, elevación al cuadrado e integración, pueden verse en el gráfico siguiente:

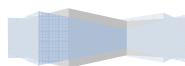


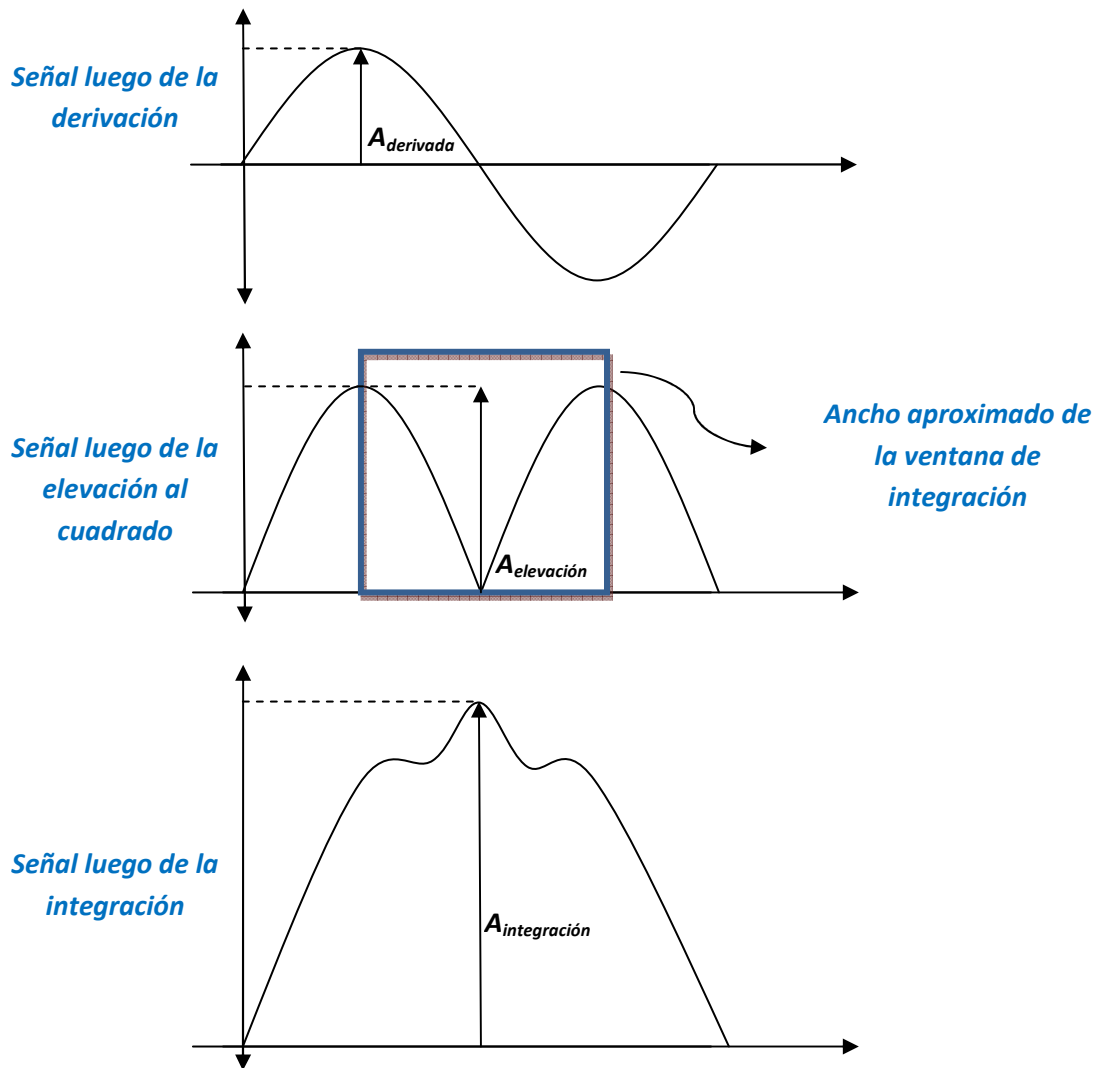
A partir de dichos gráficos intentaremos deducir una expresión matemática que las modelice correctamente. Podemos observar que la forma de onda de la derivada de la señal original posee un máximo, que se corresponde con el punto de mayor pendiente de la señal, seguido por un cero y un mínimo de un valor similar al máximo anterior, dados, como fue explicado anteriormente, por la despolarización y repolarización de las células miocárdicas. Ahora bien, si suponemos estos tiempos de órdenes similares, también podemos decir que las pendientes de subida y bajada son similares, y la forma de onda de la derivada de la señal cardíaca alrededor del entorno del punto R de la señal (que será nuestro entorno de interés), podría aproximarse como un ciclo de una señal senoidal (lo cual también se respalda en el hecho de que posterior al filtrado el espectro de frecuencias es muy angosto, por lo que la señal no poseerá cambios abruptos en su forma, tendiendo a parecerse a un seno), en donde el punto de cruce por cero será el valor temporal correspondiente al máximo del complejo QRS (punto R).

Para respaldar esta teoría, se realizó un estudio mediante la correlación cruzada entre el núcleo de un seno y un complejo QRS de igual cantidad de muestras. Para diferentes señales de análisis, el resultado de correlación se mantuvo entre 0.85 y 0.95, lo que indica una fuerte similitud entre las señales, tal como puede verse en el siguiente gráfico:



Seguindo este análisis, la elevación al cuadrado daría como resultado dos nodos del seno cuadrado de una señal, y finalmente la señal resultante posterior a la integración será la integral de esta señal entre el comienzo y fin de la misma (recordando que la integración no es de la totalidad de la señal sino simplemente de una ventana de un ancho aproximado igual al de un complejo QRS). El proceso, luego, puede esquematizarse de acuerdo al siguiente diagrama, a continuación del cual se calcularán los valores de amplitud máxima en cada bloque:





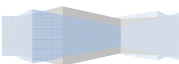
Comenzamos fijando la amplitud máxima de la señal integrada en 1, por lo que:

$$A_{integración} = 1$$

$$A_{integración} = A_{elevación} \cdot \int_{\pi/2}^{3\pi/2} \text{sen}^2(wt) \cdot dw = A_{elevación} \cdot \left(\frac{wt}{2} - \frac{\text{sen}(2wt)}{4} \right) \Big|_{\pi/2}^{3\pi/2}$$

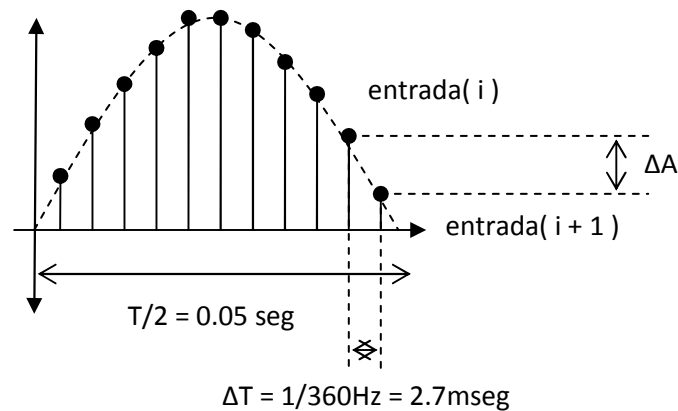
$$A_{integración} = A_{elevación} \cdot \frac{\pi}{2} \quad \therefore \quad A_{elevación} = \frac{2}{\pi} \approx 0.6366$$

$$A_{elevación} = A_{derivada}^2 \quad \therefore \quad A_{derivada} \approx 0.7979$$



Ahora bien, fijado el valor de la amplitud máxima de la señal que surge de derivar la señal original, tendremos que encontrar la relación entre ambas señales. Como fue explicado en la sección *Implementación en Matlab / Derivación*, la señal de la derivada surge de encontrar la diferencia entre 2 muestras sucesivas de la señal original. Tendremos que volver a realizar una aproximación para encontrar este valor.

Como sabemos, la derivación se realiza a partir de la señal posterior al filtrado. Por las razones expuestas anteriormente, continuaremos con la modelización de dicha señal como una señal senoidal, cuya frecuencia (dada por el valor central del filtro pasabandas que se utilizó para obtenerla) es de 10Hz. Ahora bien, si muestreamos una señal senoidal de 10Hz a 360Hz, como lo estamos haciendo, obtendremos muestras de acuerdo al siguiente esquema, de donde puede analizarse la amplitud máxima entre muestras para este caso:



De aquí, y sabiendo que la derivada del seno es máxima en cuanto el seno se anula, podremos obtener el máximo valor posible de la derivada evaluando las muestras anterior y posterior al cruce por cero, de la siguiente manera:

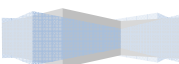
$$f(t) = A_{filtada} \left(\text{sen} \left(2\pi f \left(t + \frac{1}{f_s} \right) \right) - \text{sen} (2\pi f t) \right)$$

$$A_{derivada_{MAX}} = A_{filtada} \left(\text{sen} \left(2\pi 10\text{Hz} \left(0 + \frac{2.7\text{mseg}}{2} \right) \right) - \text{sen} \left(2\pi 10\text{Hz} \left(0 - \frac{2.7\text{mseg}}{2} \right) \right) \right)$$

$$A_{derivada_{MAX}} = A_{filtada} \cdot 0.16944$$

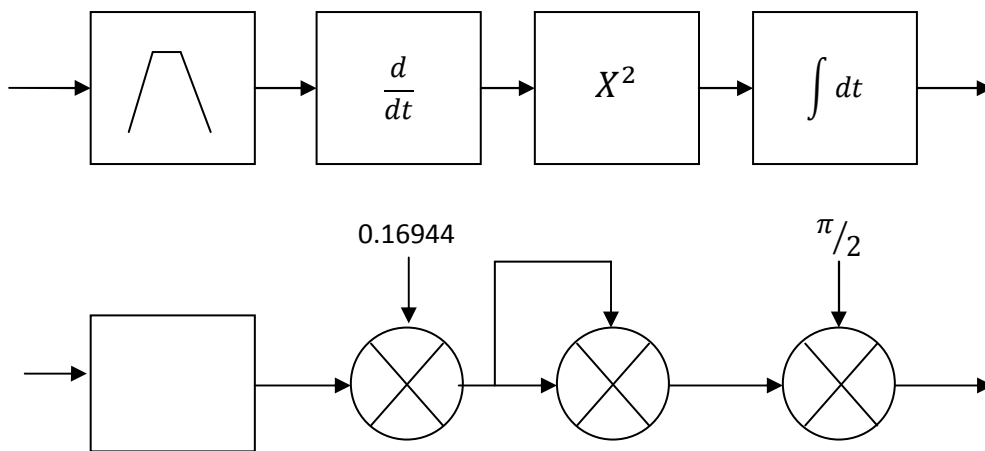
De acuerdo al desarrollo anterior, llegamos a la conclusión de que el máximo valor que podremos tener a la salida del filtrado es de:

$$A_{filtada} = 0,7979/0.16944 = 4,71$$

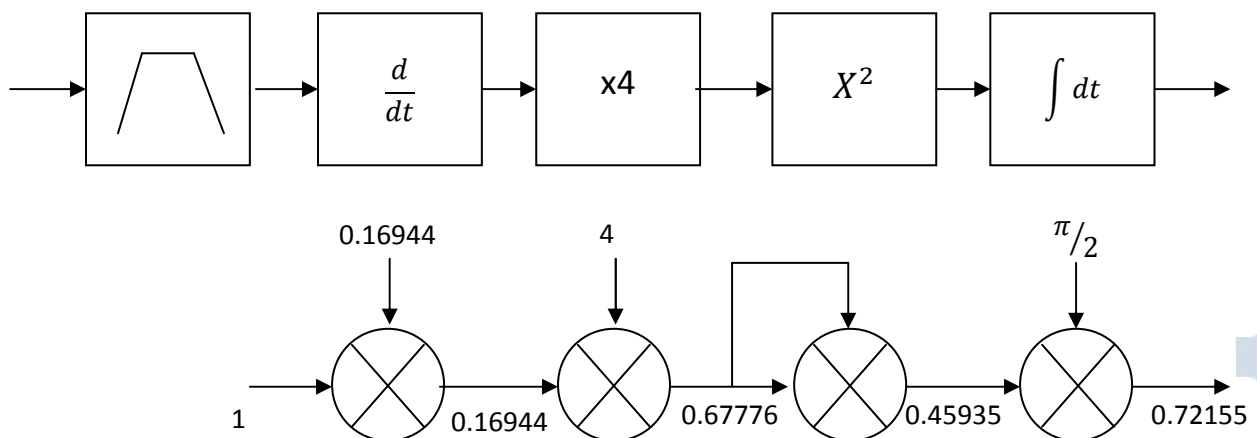


Llegamos por ende a la conclusión de que para tener una amplitud unitaria a la salida debemos tener una señal de amplitud 4.71 a la salida del filtrado. Ahora bien, como sabemos, la limitación del código no nos permite una señal de esta magnitud en ninguna sección de nuestro proceso. Para salvar este obstáculo pueden agregarse en el camino directo de la señal sucesivos multiplicadores o divisores que se encarguen de resolver este conflicto.

Esto es, podemos modelizar el sistema (solo a los propósitos del estudio de la amplitud máxima de la señal) como una serie de multiplicadores y divisores de amplitud de la siguiente manera:

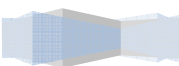
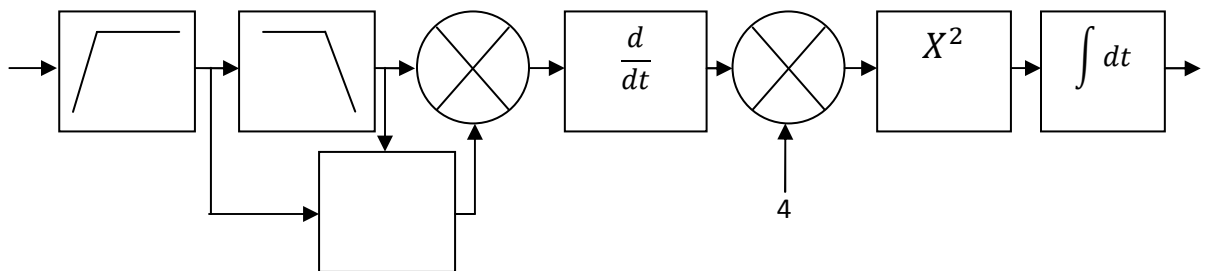


Con el sistema modelizado de esta manera, y suponiendo que a la entrada del bloque derivador habrá una señal unitaria (luego se analizará la forma de implementar esto), intentaremos que la señal posea la máxima amplitud posible (de manera de aprovechar el rango dinámico con la mayor precisión) sin superar la unidad en ninguna instancia. Partiendo de la suposición antes expuesta, y suponiendo los multiplicadores (de módulo 2) ubicados en diferentes instancias del esquema, se llega a la conclusión de que la mayor salida puede obtenerse cambiando el esquema de la manera:



Queda finalmente por considerar el bloque del filtrado de la señal. El filtrado es un bloque que pretende no modificar la señal a frecuencia central del mismo (en este caso, 10Hz), y atenuar fuertemente la amplitud de las componentes de mayor y menor frecuencia que las frecuencias de corte superior e inferior del filtro respectivamente. La amplitud de la señal a la salida de este bloque depende por lo tanto fuertemente de la fuente de la señal y sus componentes de alta y baja frecuencia. Si consideramos una señal sin una componente de continua, que aproveche completamente el rango de 3.3V a -3.3V, y que además no posea componentes de alta frecuencia de gran amplitud, el máximo valor a la salida del filtrado debería ser cercano a la unidad, mientras que si se tratase de una señal que se encontrase montada sobre un determinado nivel de continua se desaprovecharía gran parte del rango dinámico de la señal.

Para intentar aprovechar en mayor medida el rango dinámico del dsp, y aprovechando el desglose del filtro pasabanda en un filtro pasaaltos y uno pasabajos, se implementó un monitoreo del nivel medio de la señal previa y posteriormente al filtro pasaaltos. Este monitoreo tiene como objetivo el análisis del nivel de continua de la señal que se eliminó en el filtrado, y en caso de que este valor sea de una amplitud considerable, el escalado de la señal a la salida del mismo de manera de mantener la amplitud a la entrada del derivador lo más cercana a la unidad posible, para cumplir con el análisis anterior. Finalmente, el sistema completo puede esquematizarse como:



Análisis de tiempos:

Una vez obtenidos los resultados buscados en la plataforma Matlab, se hizo un análisis de los tiempos involucrados en cada parte del programa. Como conclusión se obtuvo que el procesamiento de la señal (el algoritmo descrito anteriormente) se lleva el 96% del tiempo total del algoritmo, mientras que la toma de decisión ocupa sólo el 4% restante del tiempo. Por otro lado, dentro del procesamiento, el algoritmo que más tiempo demora es el del filtrado, que ocupa casi la totalidad del tiempo de procesamiento. Esto indica que será el procesamiento la parte crucial del algoritmo, y aquella que deberá realizarse con mayor cuidado.

Por tal motivo, y por la complejidad del algoritmo de toma de decisión, se decidió implementar este último en C, mientras que el resto del procesamiento de la señal se realizó en Assembler, de manera de poder optimizar los tiempos de procesamiento.

De acuerdo a la explicación del funcionamiento del algoritmo descrito anteriormente, se puede realizar el siguiente análisis: una vez llenado el buffer con muestras provenientes del conversor A/D y de acuerdo a lo explicado en el apartado anterior, se procederá al procesamiento mientras se llena un segundo buffer con muestras. Antes que el mismo se haya llenado, el procesamiento debería de haber terminado, de manera de poder dar comienzo nuevamente al proceso.

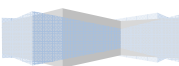
Para evitar problemas en los límites de la señal (picos en las primera o últimas muestras de un buffer, que no pueden detectarse correctamente por el transitorio de los filtrados y la imposibilidad de integrar los valores anteriores), no se procesa solamente el buffer de muestras, sino que también se procesan N muestras anteriores y posteriores a la porción de la señal en cuestión. Una vez finalizado el procesamiento del bloque, se toman los valores centrales (correspondientes con el bloque de la señal que fue extraído del buffer de entrada) y se coloca en un buffer de salida, el cual se vaciará por medio de un DMA (que enviará las muestras a uno de los canales del conversor D/A).

Esta implementación trae aparejadas limitaciones temporales, las cuales se fijaron antes de comenzar el procesamiento, de acuerdo a los siguientes valores:

1. Tiempo de demora entre un bloque de entrada y la extracción de los resultados correspondientes: $\Delta T = \frac{1}{f_s} \times TAM_{BUFFERS}$
2. Tiempo que deberá tardar el procesamiento total de la señal (antes de que se termine de llenar el buffer): $T_{procesamiento} = \frac{1}{f_s} \times (TAM_{BUFFERS} - (2 \times TAM_{FILTROS}))$
3. Tiempo máximo que podrá tardar la función de interrupción:

$$T_{interrupcion} = \frac{1}{f_{sampling_{HW}}} = 20,8\mu seg$$

Teniendo en cuenta que la señal genera alrededor de un pico por segundo, y las señales de prueba fueron obtenidas con una frecuencia de muestreo (fs) de 360Hz, se seleccionarán los siguientes valores:



$$TAM_{BUFFERS} = 360 \text{ muestras} \therefore \Delta T = 1 \text{ segundo}$$

$$TAM_{FILTROS} = 90 \text{ muestras} \therefore T_{procesamiento} = 0,5 \text{ segundos}$$

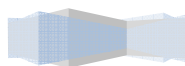
De aquí que habrá un segundo de demora entre la generación de un pico y la recolección del resultado correspondiente, y el tiempo de procesamiento deberá ser menor a 0,5 segundos para que las nuevas muestras no sobrescriban las anteriores.

Finalmente, se midieron los tiempos de procesamiento reales utilizando un osciloscopio digital y mediante el cambio de estado de un pin del puerto de salida previa y posteriormente a comenzar el bloque de código a medir. Se obtuvieron los siguientes resultados:

Etapa	Sub Etapa	Tiempo (μseg)	Máximo Teórico
Interrupción (ISR)		1,3	20,8 μseg
Procesamiento		345 a 1115	500000 μseg
	Filtrado	212	
	Derivación	6	
	Elevación	3	
	Integración	34	
	Toma de Decisión*	133 a 860	

* La toma de decisión se encuentra basada en un algoritmo que busca entre los puntos de mayor amplitud de la señal integrada posibles complejos QRS, para luego detectar si los mismos son realmente complejos reales a partir de las condiciones que se le imponen a los mismos. Dado el ancho de los buffers que se seleccionó, se parte de la suposición de que hay un complejo en el bloque, procediéndose a buscarlo bajando el nivel de comparación en la búsqueda de máximos. Esto permite encontrar complejos QRS de baja amplitud, pero también hace más lento el algoritmo en caso de ejecutar el mismo con una señal cuya frecuencia cardíaca sea menor a 40ppm, ya que el resultado del procesamiento es que no hay ningún complejo luego de buscar en todo el bloque sucesivas veces con niveles de comparación menores.

Los valores difieren, por lo tanto, bastante en caso de que haya o no una señal a la entrada del sistema, habiendo medido tiempos de 133 a 202 μseg cuando hay una señal conectada, y tiempos de 196 a 860 μseg en el caso de que no lo haya.



Referencias:

- (1) Puede encontrarse más información acerca del funcionamiento del sistema cardiovascular en las siguientes publicaciones, que fueron consultadas para la realización del presente informe:
 - a. Dr. Edmundo Cabrera Fischer , “Bases de Fisiología Para Ingenieros”.
 - b. Dr. Ricardo Armentano, Dr. Edmundo Cabrera Fischer “Biomecánica Arterial. Un enfoque desde la Ingeniería Biomédica”
- (2) Para el delineamiento de los pasos a seguir que debía tener el algoritmo, se consultaron los trabajos de:
 - a. Jiapu Pan, Willis J. Tompkins, “A Real-Time QRS Detection Algorithm”
 - b. Geoffrey Green, “Implementation of algorithms for QRS detection from ECG signals using TMS320C6713 processor platform”
 - c. Fayyaz A. Afsar, M. Arif, “QRS Detection and Delineation Techniques for ECG Based Robust Clinical Decision Support System Design”
- (3) Thakor. N.V. Webster, J. G., and Tompkins. W.J. “Optimal QRS Detector. Medical and Biological Engineering”, 1983
- (4) Rafael Gonzalez Díaz, Juan Antonio Silva “Desarrollo de Algoritmos de Búsqueda Retrospectiva y Ventana de Integración para la detección de arritmias tipo taquicardia y fibrilación ventricular, utilizando el software Matlab”
- (5) Carlos Alvarado Serranos, “Análisis de la variabilidad de intervalos de tiempo en el ECG”, Universidad Politécnica de Catalunya.
- (6) Brij N. Singh, Arvind K. Tiwari, “Optimal selectionn of wavelet basis function applied to ECG signal denoising”, Digital Signal Processing 16 (2006) 275-287
- (7) Fayyaz A. Afasar, M. Arif, “QRS Detection and Delineation Techniques for ECG Based Robust Clinical Decision Support System Design”

