Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Electrónica

Ergómetro con Detección de Arritmias Empleando Redes Neuronales

Autores:

Raúl Alejandro González
María Gabriela Messineo

Mar del Plata, 2004
UNMDP

Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Electrónica

Ergómetro con Detección de Arritmias Empleando Redes Neuronales

Autores:

Raúl Alejandro González
María Gabriela Messineo

Mar del Plata, 2004
UNMDP

INDICE

RESUMEN

El trabajo realizado consistió en el diseño y construcción de un ergómetro basado en PC, y en el desarrollo de un software basado en redes neuronales que permite detectar patrones anómalos en las señales registradas.

El *ergómetro* es un instrumento para tomar un registro electrocardiográfico de un paciente que está realizando algún tipo de actividad física e incrementando el esfuerzo progresivamente. La *ergometría* también se conoce como *prueba de esfuerzo*.

El instrumento desarrollado para tal fin tiene la posibilidad de adquirir cuatro derivaciones cardíacas en forma simultánea. Consta de cuatro amplificadores diferenciales, un conversor analógico digital multiplexado, y una interfase de adquisición que permite ingresar a la PC las señales adquiridas a través del puerto paralelo.

El sistema presenta la opción de visualizar en pantalla y en tiempo real, las cuatro señales en forma simultánea, o bien individualmente. A medida que las señales son adquiridas, se las va almacenando en un registro al que luego se podrá acceder para realizar su análisis.

El análisis de las señales fue efectuado mediante una red neuronal competitiva. El objetivo de tal análisis fue hallar parámetros que permitan que la red realice una clasificación de los ciclos cardíacos en *latidos de duración normal* y *latidos cortos o anticipados*, ya sea por contracción ventricular o auricular prematura. La red empleada contiene a su vez dos tipos de arquitectura: SOM y LVQ (con aprendizaje no supervisado y supervisado respectivamente), que utilizan una clasificación competitiva.

1- INTRODUCCIÓN A LA ELECTROCARDIOGRAFÍA

La electrocardiografía es el registro de los potenciales eléctricos del corazón, que preceden y dan origen a la actividad mecánica del mismo. El electrocardiograma normal debe realizarse en completo reposo, dado que el movimiento muscular también genera potenciales eléctricos que enmascararían en el registro a los del corazón. En el caso del electrocardiograma de esfuerzo el paciente está realizando esfuerzo físico, por lo que es necesario que el instrumento elimine los potenciales generados por los músculos.

El corazón es el órgano de impulsión de la sangre a todo el organismo, la cual es guiada hacia su destino por el sistema circulatorio. Se trata de una bomba doble que recibe en su sección derecha la sangre que viene de las venas, la filtra a través de los pulmones para oxigenarla, y finalmente la lanza hacia el resto del organismo mediante su sección izquierda.

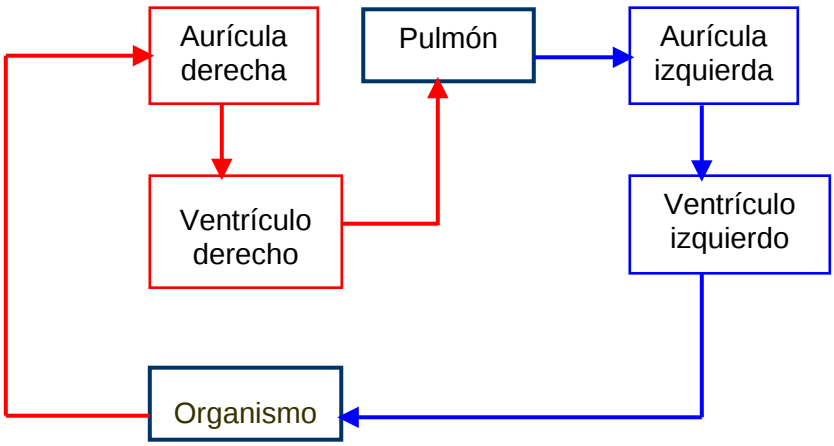


Figura 1: Sistema cardiovascular

El corazón presenta dos fases: relajamiento o *diástole*, cuando sus cavidades, llamadas aurículas y ventrículos, en estado de relajación se llenan de sangre; contracción o *sístole*, cuando ambas se contraen y expulsan su contenido en forma secuencial y sincronizada. La frecuencia cardiaca, inversa del período que abarcan ambas fases es, en el corazón normal, de 70 latidos por minuto (aproximadamente 0.86 Hz).

La actividad bioeléctrica cardiaca se origina en la actividad bioeléctrica de cada célula muscular cardiaca. A su vez, esta actividad electromecánica se produce siempre igual, latido tras latido, debido a la excitación propagada a través de vías de conducción (el haz de Hiss y las ramificaciones de Purkinje) de un impulso producido automáticamente en el nódulo seno auricular. Este nódulo es el “marcapasos” del corazón y consta de un grupo de células especializadas que se halla en la pared posterior de la aurícula derecha y tienen un automatismo de unos 70 latidos por minuto. Si fallara el nódulo mencionado existen otros nódulos de automatismo más lento que toman el control.

El impulso generado en el nódulo S-A viaja a través de las vías internodales al nódulo aurículo ventricular, donde sufre un retardo y penetra en los ventrículos mediante el haz de Hiss, donde sus ramas derecha e izquierda, que se ramifican en las fibras de Purkinje, conducen el impulso a toda la masa ventricular, que se contrae simultáneamente.

1.1 Registro de la actividad eléctrica celular

La célula cardiaca es negativa por dentro y positiva por fuera en estado de reposo. Cuando recibe un estímulo, ya sea mecánico o eléctrico, se despolariza. Es decir, luego de un tiempo se invierten las cargas, quedando cargada positivamente adentro y negativamente su exterior. Esto se debe a la apertura de canales existentes en la membrana celular que permiten el paso de determinados iones, lo que cambia su polaridad interna.

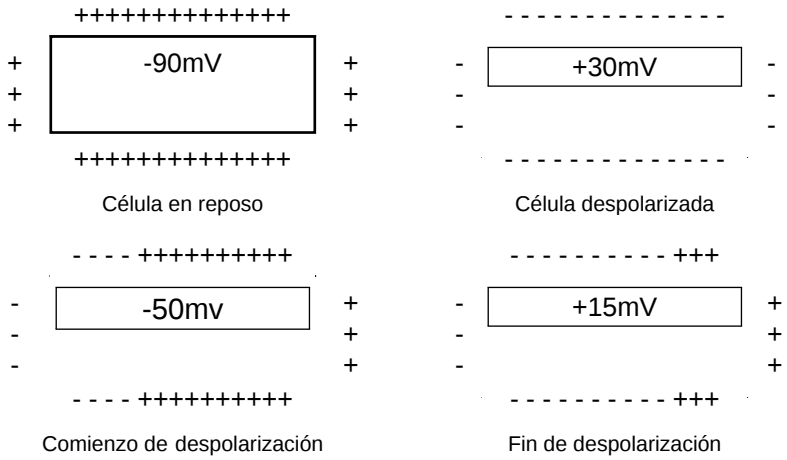


Figura 2: Despolarización y repolarización de la membrana celular

La membrana permanece despolarizada un momento y luego se repolariza cuando los iones retornan al equilibrio.

Para estudiar las variaciones temporales de la polarización y despolarización de la célula cardíaca se puede recurrir a los llamados registros de potenciales intracelulares o a los registros extracelulares. En el primero, se coloca la célula en una solución que permite su subsistencia, se emplaza un electrodo llamado indiferente en cualquier parte de la solución, y se inserta otro electrodo en el interior de la misma. Lo que se obtiene es la diferencia de potencial entre la célula y el líquido. En el registro extracelular los electrodos se colocan en extremos opuestos de la célula y se mide la diferencia de potencial entre los mismos.

Estos potenciales que se desarrollan temporalmente pueden ser interpretados como originados por la creación de un dipolo ya que, sobre la superficie de la célula, a medida que progresa la despolarización se desarrollarán cargas negativas en el sector despolarizado y positivas en el sector que aun permanezca en reposo. La máxima diferencia de potencial sobre un dipolo se obtendrá sobre el eje que une ambas cargas.

1.2 Características de un electrocardiograma normal

El electrocardiograma normal está formado por una onda P, un complejo QRS y una onda T. Las porciones del electrocardiograma entre las deflexiones se denominan segmentos; las distancias entre ondas se denominan intervalos. El ECG puede ser dividido en los siguientes intervalos y segmentos.

- La onda P está causada por corrientes eléctricas generadas cuando las aurículas se despolarizan antes de la contracción. Su duración normal es de 100 ms y la forma depende de la derivación que se tome. Un aumento del voltaje o de la

duración de esta onda indica una anomalía auricular. La ausencia de esta onda ocurre en una parada del nodo sinusal y en el bloqueo seno auricular.

- El complejo QRS representa la despolarización del ventrículo que precede a su contracción. Está formada por las ondas Q, R y S y su duración es de aproximadamente 100ms.
- La onda T representa la repolarización de los ventrículos. Normalmente es asimétrica y redondeada en su vértice. La pendiente de la rama inicial es más suave que la de su rama terminal. Las anomalías de esta onda pueden deberse a enfermedades cardíacas primarias, aunque hay casos de personas sanas con las mismas anomalías.
- Existe eventualmente también una onda U que tiene un origen fisiológico poco claro.
- Segmento PR. Corresponde a la línea isoeléctrica entre el comienzo de la onda P y la deflexión inicial del complejo QRS. Su duración normal es de 120 a 210 ms.
- Segmento ST. Es el intervalo entre el final del complejo QRS y el principio de la onda T. representa el tiempo durante el que los ventrículos permanecen despolarizados y puede iniciarse la repolarización. Normalmente este segmento es isoeléctrico, aunque puede estar ligeramente desviado. Una desviación elevada a menudo es consecuencia de un infarto de miocardio, una pericarditis aguda o una miocarditis.
- Intervalo QT. Corresponde al intervalo de tiempo entre el comienzo del complejo QRS y el final de onda T, representando la duración de la sístole ventricular. Su duración depende del ritmo cardíaco. Ver tabla 1.
- Intervalo QRS. Corresponde al intervalo de tiempo entre el comienzo de la onda Q y el final de la onda S. Es el indicador del tiempo de conducción intra ventricular.

Un registro bipolar de una célula cardiaca tiene una forma siempre difásica, con la particularidad que la fase de despolarización (complejo QRS) es rápida, mientras que la de repolarización (T) es lenta, y el área comprendidas por ambas fases es igual.

Como se ve en la figura 3, entre las dos fases hay un período isoeléctrico que se denomina segmento ST, y la polaridad de la onda T está invertida.

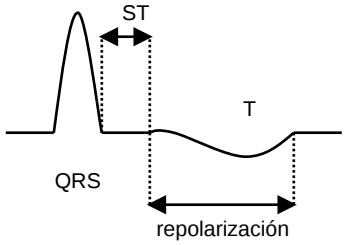


Figura 3: Polaridad de la despolarización y la repolarización de una célula ventricular

Tabla 1

Ritmo cardíaco (ppm)	Duración int. QT (seg.)
60	0,33-0,43
70	0,31-0,41
80	0,29-0,38
90	0,28-0,36
100	0,27-0,35
120	0,25-0,32

En la siguiente figura se observa un registro bipolar obtenido de un trozo de tejido que proviene del ventrículo izquierdo del corazón humano, desde el endocardio hasta el pericardio. Nótese que las ondas QRS y T tienen la misma polaridad. Ello es debido a que, por particularidades propias de las células miocárdicas de los ventrículos, el período isoeléctrico ST es mayor en el endocardio que en el pericardio, de modo que, si bien la despolarización se inicia en el endocardio, la repolarización comienza en el pericardio.

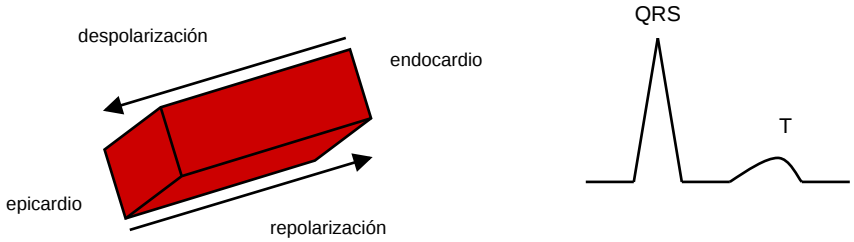


Figura 4: Despolarización y repolarización de un trozo de tejido miocárdico

Podemos considerar que el miocardio se va repolarizando en capas desde el pericardio hasta el endocardio, es por ello que en un momento habrá varias capas positivas que envuelven a las que aún se conservan negativas (despolarizadas). Con ello, el dipolo que ven los electrodos, colocados uno en el endocardio y otro en el pericardio, tiene el mismo sentido durante la polarización que durante la repolarización.

Esta situación se da también en el verdadero registro electrocardiográfico. Como se verá, existen 6 registros unipolares y 6 bipolares y aunque no es posible colocar el electrodo activo directamente sobre el endocardio o sobre el pericardio, durante la práctica electrocardiográfica unipolar normal, puede considerarse que el mismo se apoya sobre seis puntos diferentes del epicardio. En consecuencia, se registrarán los eventos eléctricos que se producen específicamente en dichos puntos del corazón.

1.3 Generación y transmisión del impulso cardíaco

Como ya se ha mencionado, el estímulo eléctrico en el corazón sano nace en el marcapaso o nódulo sino-auricular (NSA) que es una pequeña formación de tejido muscular especializado localizada en la aurícula derecha en el sitio de unión de la vena cava superior y el atrium.

La acción de este estímulo da origen a una pequeña corriente eléctrica, llamada corriente de excitación u onda de despolarización. Esta onda (P) se propaga en forma de anillos concéntricos envolviendo ambas aurículas. Cuando esta onda llega a la zona de unión entre la aurícula y el ventrículo derechos debe pasar a través de otro pequeño

centro llamado nódulo aurículo-ventricular, donde la velocidad de propagación es mucho menor que en las aurículas y que en el resto del tejido de conducción; por consecuencia la onda se retarda. Esto permite que la contracción del ventrículo ocurra luego que la de la aurícula, dejando que ésta impulse la sangre al interior del ventrículo antes de que el mismo se contraiga. Esto se revela en el ECG por la aparición de un segmento horizontal (isoeléctrico o de voltaje nulo).

En este nódulo AV tiene origen el llamado haz de Hiss, el cual es un tejido conductor cardíaco especializado, que presenta una mayor conductividad al paso de la corriente que el tejido muscular circundante. El haz de Hiss, que transcurre a través de la pared intra ventricular, desprende tres ramas, dos izquierdas y una derecha, que se internan en los ventrículos respectivos. Una vez en ellos, estas ramas emiten finas prolongaciones, llamadas fibras de Purkinje, que se entremezclan con las fibras musculares cardíacas, a las que comunican la despolarización eléctrica, a fin de lograr la contracción.

Durante la conducción del impulso desde el comienzo del haz de Hiss hasta las últimas células cardíacas obtenemos el complejo QRS, que representa la despolarización ventricular. La forma del complejo QRS indica las fuerzas eléctricas desarrolladas en los ventrículos. Los ventrículos, tras un momento de reposo (segmento ST) durante el cual permanecen despolarizados, comienzan a repolarizarse partiendo, como la onda de despolarización, de la punta del corazón hacia su base. Se habrá dibujado entonces la onda T. La repolarización de las aurículas coincide con la generación del QRS, y por ser de mucho menor voltaje es ocultada completamente por el mismo. Además de las ondas descritas, muchas veces aparece, luego de la onda T, una última onda, U, cuyo origen es desconocido pero que suele servir para realizar algunos diagnósticos.

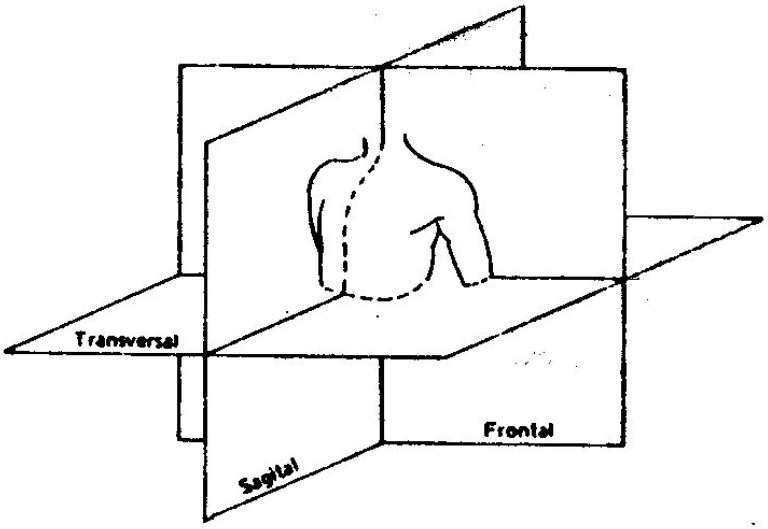
1.4 Registro de los potenciales de acción

El corazón puede ser considerado como una única masa muscular que genera una diferencia de potencial sobre la superficie del tórax con características vectoriales. Por medio de dos electrodos aplicados en puntos prefijados del mismo y conectados a las entradas de un amplificador es posible el registro de los potenciales de acción del corazón en función del tiempo, obteniéndose un gráfico característico: el Electrocardiograma (ECG).

Con el ECG se efectúa una medida de los potenciales de acción entre varios puntos de la superficie de un volumen conductor y con esa información, por el análisis de las características del registro, puede efectuarse un diagnóstico sobre la condición clínica del corazón.

Para simplificar un poco la técnica electrocardiográfica, se considera que los potenciales de acción se proyectan vectorialmente a lo largo de los ejes existentes en cada uno de los tres planos de referencia. Ellos son: el plano frontal, el plano sagital y el plano transversal, siendo posible obtener registros de cada uno de ellos.

Se ha convenido en llamar electrodo positivo a aquel que al “ver venir” un vector eléctrico cuyo extremo sea positivo dará origen a que el registro sea un trazo ascendente.



- *Registros en el plano frontal*

Existen dos sistemas de colocación de electrodos en este plano: el Sistema Estándar o Bipolar de los Miembros y el sistema Aumentado o Unipolar de los Miembros

- Sistema o Derivación Estándar

Se colocan tres electrodos, uno en el brazo derecho, uno en el brazo izquierdo y uno en la pierna izquierda. En la práctica se coloca un cuarto electrodo en la pierna derecha que hace las veces de tierra, ya que se conecta a la masa del amplificador diferencial.

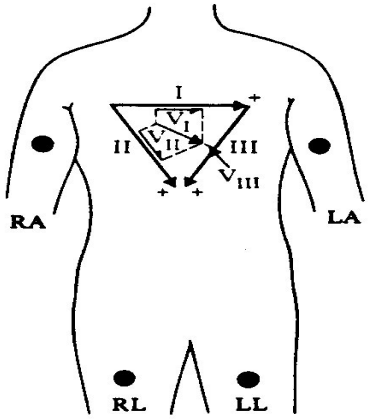


Figura 5: Derivaciones bipolares

De esta forma es posible obtener tres registros:

- DI: tomando señal entre el electrodo que está en el brazo izquierdo y el colocado en el brazo derecho.
- DII: tomando señal entre el brazo derecho y la pierna izquierda
- DIII: tomando señal entre brazo izquierdo y pierna izquierda.

Este sistema es debido a Einthoven; en él, el corazón es supuesto en el centro geométrico de un conductor en forma de triángulo equilátero. Entonces se cumple la ley formulada por él, y que es una aplicación de la 1º ley de Kirchoff, que dice: el potencial en un instante dado recogido en DII es igual a la suma algebraica de los potenciales, en el mismo instante, de las derivaciones DI y DIII.

Esta regla es útil para verificar si los electrodos han sido correctamente colocados, tanto para el sistema Unipolar como para el Estándar.

En la figura 6 puede observarse cómo un vector eléctrico cardíaco, que puede considerarse como originado en el centro de un triángulo equilátero formado por BD, BI y PI, es descompuesto en los tres lados del triángulo.

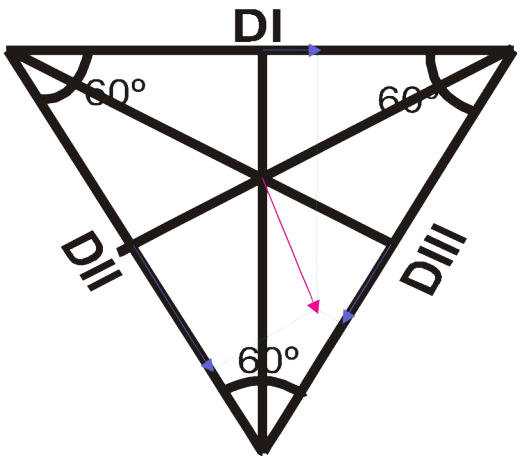


Figura 6: Triángulo de Einthoven

Se comprende que tomando dos de estas proyecciones y haciendo su suma geométrica sea posible determinar la dirección del vector cardíaco original en el plano frontal del cuerpo.

Es posible obtener así el vector eléctrico instantáneo frontal para cualquier instante del ciclo cardíaco o bien un vector eléctrico medio frontal para los distintos eventos del ciclo cardíaco tomando en cuenta para este último caso el área que yace bajo la porción del trazado considerado. Lo más común es obtener la dirección del eje eléctrico medio de la despolarización de los ventrículos, para lo cual se toma en cuenta el área yaciente bajo el complejo QRS del trazado electrocardiográfico.

- Sistema o Derivación Unipolar de los Miembros

La derivación unipolar de los miembros permite un registro unipolar del potencial eléctrico en cada una de las siguientes extremidades: brazo izquierdo (BI), brazo derecho (BD) y pierna izquierda (PI), mediante la creación de un electrodo de referencia o Terminal Central de Wilson.

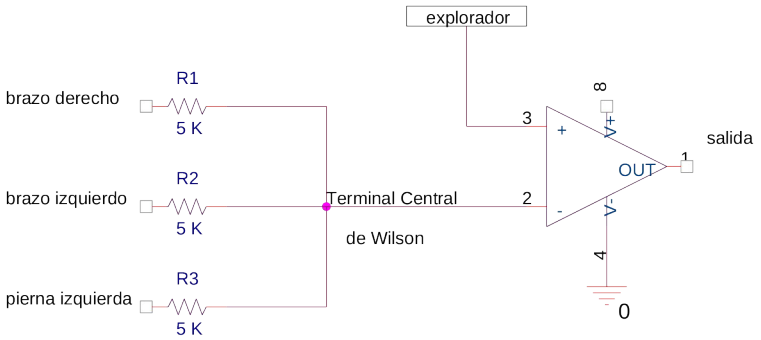


Figura 7: Terminal de Wilson

El Terminal de Wilson consiste en la unión de los tres cables provenientes de las extremidades, intercalando una resistencia de 5 KΩ. Se cumple, como es de esperar, que en dicha unión, de acuerdo con la ley de Einthoven, no circula corriente durante todo el ciclo cardíaco.

Este terminal se conecta a la entrada negativa del amplificador diferencial; de la entrada positiva del amplificador parte el electrodo activo que se colocará sucesivamente en el BD, BI y PI a fin de obtener tres registros unipolares. Esta técnica persigue obtener registros muy estables, poco afectados por las corrientes iónicas fisiológicas extracardiacas que siempre están presentes y que pueden producir una deriva de línea y deformaciones en el complejo P-QRS-T. Además permite investigar el corazón desde un plano transversal, que sumado al frontal de las bipolares, es de utilidad para identificar ciertos fenómenos normales y patológicos.

A pesar de la gran calidad del trazado obtenido, la disposición descrita ocasiona una reducción de la amplitud de los potenciales que llegan al amplificador, de manera que se utiliza una disposición debida a Goldberg y que permite obtener un muy buen registro con potenciales una vez y media más amplios que con la original de Wilson. En tal disposición persiste la Central de Wilson, pero cada vez que se requiere tomar un registro en una de las extremidades se desconecta el cable que conecta a esa extremidad con la Central. Estos registros se denominan Derivaciones Bipolares aumentadas y en el trazado se identifican aVR (BD), aVL (BI) y aVF (PI).

En realidad, estas derivaciones son simplemente otras proyecciones del vector cardíaco sobre tres ejes diferentes del plano frontal, a 60º uno del otro, pero rotados 30º con respecto a los del triángulo de Einthoven ya mencionado.

Los potenciales instantáneos de los registros tomados por los dos sistemas están relacionados de la siguiente manera:

$$aVR = -\frac{DI+DII}{2}$$

$$aVL = \frac{DI-DIII}{2}$$

$$aVF = \frac{DII+DIII}{2}$$

En la siguiente figura se muestra el sistema de ejes resultantes de unir en un punto central los tres ejes de cada uno de los dos sistemas mencionados. Resulta un sistema hexaxial del plano frontal, quedando cada eje rotado 30º con respecto al siguiente.

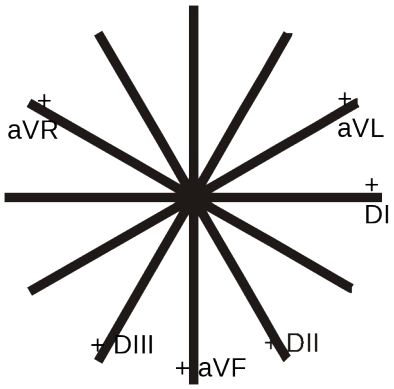


Figura 8: Derivaciones aumentadas

- *Registro en el plano transversal (horizontal)*

Para registrar la proyección del vector cardíaco en el plano transversal se usan las Derivaciones Precordiales Unipolares. La disposición de los electrodos puede verse en la siguiente figura:

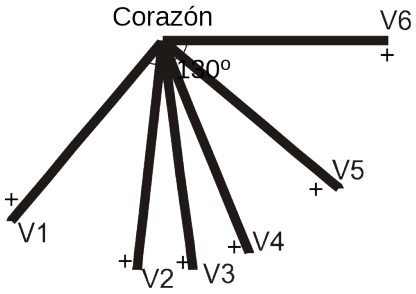


Figura 9: Derivaciones precordiales o torácicas

En esta disposición se trabaja con la central terminal de Wilson como electrodo de referencia (que oficia de electrodo negativo) y un electrodo activo o explorador (positivo) que el operador va situando en 6 o más posiciones distintas sobre la pared torácica precordial. Estas posiciones, que están estandarizadas, permiten obtener un detalle de los eventos eléctricos de las zonas cardíacas próximas al electrodo explorador.

Cada registro es designado con la letra V, seguida de un número que indica su posición sobre el tórax.

1.5 Interpretación básica del ECG

El electrocardiograma sirve no solo para informarnos acerca del estado de los nódulos productores de los estímulos (Sino-auricular y Aurículo-ventricular) y de las vías de conducción de los mismos (arritmias, fallas de conducción), sino para detectar problemas orgánicos tales como: obstrucción parcial o total de las arterias coronarias que nutren al corazón (angina de pecho, infarto de miocardio); insuficiencia de los ventrículos para expeler la sangre que les llega; patologías de las válvulas; etc. Cada uno de estos diagnósticos se realiza considerando los siguientes parámetros:

a) observación de la generación de las ondas versus el tiempo;
b) medición de los potenciales producidos (en mV);
c) estudio de la dirección de los vectores eléctricos;
d) observación de la morfología y secuencia de las ondas

2- PRUEBA DE ESFUERZO

2.1 Introducción

La prueba de esfuerzo o ergometría consiste en registrar la actividad eléctrica del corazón mediante un electrocardiógrafo y la presión sanguínea mediante un esfigmomanómetro, mientras el paciente camina en un tapiz rodante o pedalea en una bicicleta fija. La prueba de esfuerzo en tapiz rodante (ETT, por sus siglas en inglés) o en una bicicleta fija se usa para:

- Determinar si existe una enfermedad del corazón
- Determinar si el tratamiento para la enfermedad del corazón es efectivo.

La prueba de esfuerzo se realiza cuando el profesional médico cree que el paciente puede tener una enfermedad del corazón, la más común de las cuales es causada por un estrechamiento de las arterias coronarias. Las arterias coronarias son los vasos sanguíneos que llevan sangre, oxígeno y elementos nutritivos al corazón y se pueden estrechar cuando se depositan sustancias como el colesterol en sus paredes.

Muchas personas con arterias coronarias estrechas no tienen síntomas cuando están en reposo pero el ejercicio hace que el corazón trabaje más. Esto produce cambios específicos en el electrocardiograma que se pueden observar haciendo un electrocardiograma antes, durante y después de haber realizado ejercicio. La prueba de esfuerzo ayuda a saber si las arterias se han estrechado.

Se colocan electrodos consistentes en parches adhesivos o pequeñas ventosas en la espalda y el pecho para registrar el electrocardiograma. Se mide la presión sanguínea y se efectúa un registro mientras el paciente esté en reposo. Después comenzará a caminar lentamente en el tapiz rodante. La velocidad y pendiente del tapiz irán aumentando aproximadamente cada 3 minutos. Se observa el electrocardiograma de manera constante y se mide la presión sanguínea cada vez que aumente la velocidad del tapiz.

La ETT se interrumpirá si:

- El electrocardiograma o la presión sanguínea se alteran excesivamente.
- El paciente manifiesta dolor de pecho.
- El paciente se cansa demasiado como para continuar.
- Se alcanza la meta de ejercicio fijada de antemano. La meta es llegar a una frecuencia cardiaca de por lo menos 85% del número 220 menos su edad.

La ETT no es perfecta. Algunas personas con enfermedad de las arterias coronarias pueden tener un ETT normal, y otras personas saludables pueden tener un ETT anormal.

La precisión del ETT se puede mejorar de dos maneras:

- Inyectando una sustancia radioactiva como talio o tecnecio en una vena del brazo en el momento pico del ejercicio. Las sustancias radioactivas permiten ver en una cámara cómo se distribuye la sangre. Las sustancias circularán más fácilmente por

arterias sanas que por las enfermas. De esta manera el diagnóstico tiene una precisión del 90%.

- Haciendo un ecocardiograma (imágenes de ultrasonido del corazón que late) justo antes y después de haber hecho el ejercicio. Si sus arterias coronarias se han estrechado, el bombeo del corazón no se verá normal en el ecocardiograma después del ejercicio. Esto se llama ecocardiografía de esfuerzo.

La ETT es una de las pruebas más seguras y populares para diagnosticar enfermedades del corazón. Es una manera rápida de verificar si las arterias del corazón se han estrechado o bloqueado. Ayudará al profesional médico a decidir si tiene que hacer pruebas más costosas y riesgosas.

Rara vez, el corazón dejará de latir durante la prueba, pero para mayor seguridad, el profesional médico supervisará la prueba. Vigilarán los resultados del electrocardiograma y medirán la presión sanguínea constantemente. El médico está pendiente de la producción de emergencias como fibrilaciones o paros cardíacos, y debe disponer de los medios necesarios para su tratamiento.

2.2 Equipo necesario para realizar la ergometría

1- Banda sinfín o bicicleta fija
2- Equipo de registro de alta fidelidad con límites de frecuencia entre 0,05 Hz y 150 Hz. Es opcional contar con un equipo computarizado que utilice algoritmos para el análisis automatizado de las características de la señal.
3- Monitor-desfibrilador para el tratamiento de un paro cardíaco, medicamentos especiales para emergencias cardiovasculares, etc.
4- Toma de oxígeno y equipo de succión.

2.3 Comportamiento de las variables clínicas

Respuesta normal de la frecuencia cardiaca durante el ejercicio

El esfuerzo realizado por el paciente durante un ejercicio debe llegar al máximo de su capacidad; esto ocurre cuando la extracción de oxígeno de los músculos que participan en el ejercicio alcanza una meseta donde ya no es posible incrementar dicha extracción, y por consiguiente, el consumo del mismo, a pesar del aumento de la intensidad del ejercicio. Una de las formas de estimar el nivel máximo de ejercicio es por medio de la frecuencia cardiaca máxima alcanzada.

A partir de la fase de meseta del consumo de oxígeno la presión arterial sistólica depende, para su incremento, solo de las variaciones de la frecuencia cardiaca, que en condiciones normales de funcionamiento del nodo sinusal, responde al efecto adrenérgico en forma casi exponencial. El incremento en la demanda de oxígeno normalmente es paralelo a la oferta del ventrículo izquierdo. Si existen factores que disminuyen el flujo coronario se produce una disfunción, que puede ir desde la disfunción diastólica inicial, hasta la pérdida de la función sistólica y ocasionar una disminución brusca de la expulsión de sangre oxigenada. Esto es una caída de la presión arterial sistólica.

Respuesta normal y anormal de la presión arterial sistólica

La presión sistólica durante el ejercicio se incrementa en forma proporcional a la magnitud del esfuerzo y éste puede expresarse en relación con el número de METs (equivalencia metabólica de carga = 3,5 ml de oxígeno por kg de peso por minuto en estado de reposo) alcanzados durante el ejercicio. Se considera normal la elevación de la presión sistólica entre 8 a 12 mm Hg por MET alcanzado. Por ejemplo: si la presión inicial del paciente es de 120 mm Hg y el sujeto realiza un esfuerzo equivalente a 10 METs, la presión arterial máxima normal es de 240 mm Hg. Si supera el límite mencionado se cataloga como respuesta hipertensiva. En el caso contrario, cuando la presión sistólica no aumenta más de 5 mm Hg por MET alcanzado, se considera una respuesta hemodinámica anormal. Lo anterior puede llegar a observarse en pacientes con infarto de miocardio reciente, en quienes un incremento no mayor de 30 mm Hg al final del esfuerzo, por arriba del valor en reposo, pone de manifiesto la mala función ventricular. Por otra parte, hay sujetos con mala condición física y sin antecedentes de importancia cuyo comportamiento es similar al descrito anteriormente; ello no debe ser considerado como anormal.

Respuesta normal y anormal de la presión arterial diastólica

Como se dijo anteriormente, las resistencias sistémicas normalmente disminuyen en forma progresiva y es por estas circunstancias que la presión diastólica puede disminuir, en general no más de 10 mm Hg, pero puede no observarse ningún cambio en la misma y ser normal. En pacientes hipertensos, ocasionalmente se llega a observar un aumento en la presión diastólica; esto es anormal y debe considerarse como respuesta hemodinámica hipertensiva.

3- SISTEMA DE ADQUISICIÓN DE DATOS A TRAVES DEL PUERTO PARALELO

3.1 Descripción del sistema

En general el sistema se divide en cuatro bloques principales: amplificación de la señal analógica (cardiaca), sistema de conversión de datos analógicos a datos digitales, sistema de interfaz a través del puerto paralelo, y sistema de graficación en pantalla y almacenamiento de los datos en memoria. Los tres primeros puntos están relacionados con el hardware del sistema y el último lo está con el software de éste.

3.2 Diseño del sistema

Como ya se indicó, el sistema esta compuesto de cuatro partes básicas:

a) Amplificación.
b) Conversión de datos analógicos a datos digitales.
c) Interfaz a través del puerto paralelo.
d) Control, Graficación y Almacenamiento de los datos en memoria.

A continuación se explican los requerimientos de cada parte y la solución propuesta en este desarrollo.

3.3 Amplificación de la señal cardiaca.

Debido a que las señales cardíacas son del orden del mV es necesario amplificarlas para poder observarlas con una buena definición. Para eliminar interferencias del tipo de modo común como la de 50 Hz, los amplificadores deben ser diferenciales y deben presentar un gran rechazo de modo común. Esta etapa consiste de cuatro amplificadores diferenciales implementados con el integrado LM324. El mismo incluye cuatro amplificadores operacionales, de los cuales tres se utilizan para la construcción del amplificador diferencial, y el cuarto para el filtrado y amplificación posterior de la señal.

La señal amplificada debe ser del orden del Volt, ya que la entrada del conversor debe oscilar entre los 0V y los +5V. Teniendo en cuenta esto último, y que las señales a digitalizar toman valores negativos, es necesario que esta etapa agregue una tensión continua (Vref) con un valor cercano al valor medio entre 0V y 5V. Luego de la adquisición este valor medio será extraído por software para su graficación en pantalla.

Sabiendo que la máxima señal de entrada al amplificador en el caso de un ECG ronda los +4 mV, y que la referencia a la entrada del conversor es de aproximadamente 2,5V, la máxima señal a la entrada del conversor debe ser de 2,5V. Esto significa que la amplificación debe ser de unas 600 veces.

$$G = \frac{5V - Vref}{4mV}$$

Lo referente al circuito de amplificación se explica con más detalles en una sección posterior.

3.4 Conversión de datos analógicos a digitales

Como se desea lograr un producto de bajo costo y fácil implementación, se emplea un conversor analógico digital comercial, el ADC 0808. Este integrado es un componente electrónico de tecnología TTL con un conversor analógico a digital de 8 bits, un multiplexor analógico de ocho entradas y control lógico compatible con el funcionamiento de un microprocesador, característica que permite conectarlo directamente a un bus de expansión de una PC. El conversor analógico digital de 8 bits emplea la técnica de aproximaciones sucesivas para realizar la conversión. El diseño del ADC 0808 ofrece alta velocidad frente a su costo, gran desempeño, mínima dependencia de la temperatura, y mínimo consumo de potencia. La figura 10 muestra un diagrama de distribución de los pines.

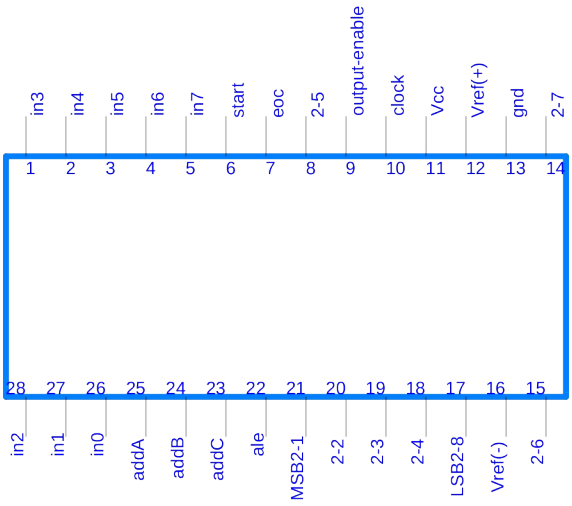


Figura 10: Diagrama de la distribución de pines del ADC 0808.

3.5 Manejo del conversor analógico digital ADC 0808

En la figura 11 se muestra el diagrama del circuito de conversión analógico a digital. La señal de clock (reloj) es generada mediante un Schmitt trigger (MC 14584), cuyo circuito se muestra en la figura 12, entregando una señal de una frecuencia aproximada a los 640 KHz. Esta es la frecuencia de clock típica del conversor, pero el ADC 0808 puede tolerar frecuencias de hasta 1.28MHz.

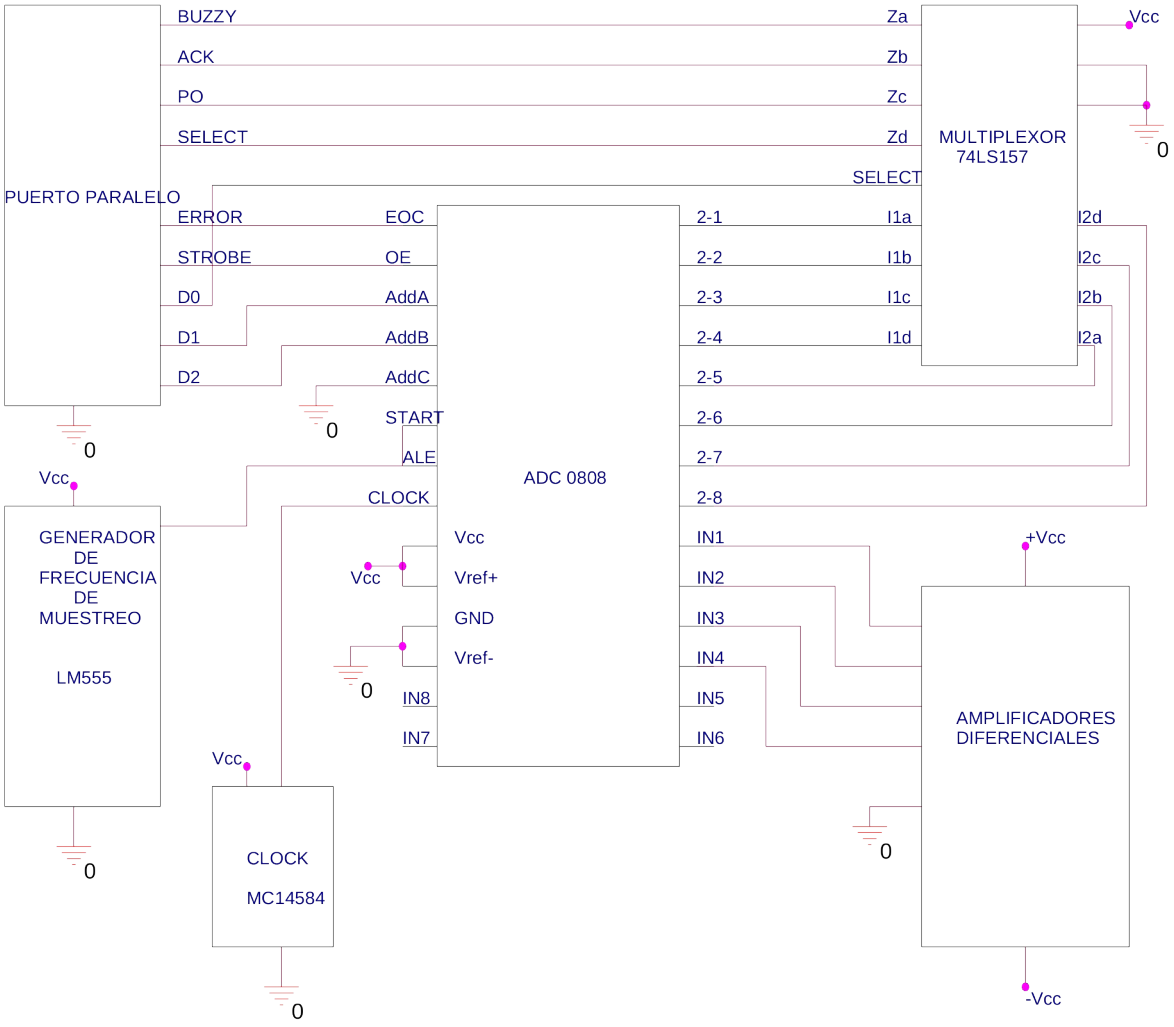


Figura 11: Diagrama de conexión del adc 0808. Este diagrama contiene únicamente las líneas que intervienen en las conexiones, y las mismas no guardan relación con su ubicación física en los integrados correspondientes.

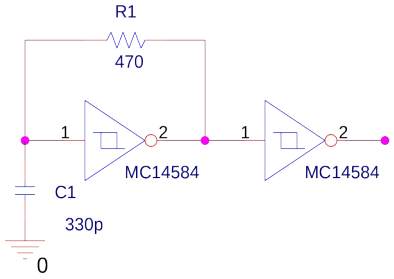


Figura 12: Circuito del oscilador con Schmitt Trigger (clock)

Las señales de START (inicio) y ALE (habilitación de canal) se emplean ligadas como un solo pin de control, las cuales, al ponerse en alto, le están indicando al conversor que tome una muestra del canal que se ha seleccionado y, al mismo tiempo, le indica al circuito integrado que inicie el proceso de conversión. Estas líneas son manejadas por otro oscilador (generador de frecuencia) implementado con un LM555 (Fig. 14)

Es importante tener en cuenta que el tiempo de conversión mínimo es de 100 μs, es decir, la frecuencia máxima de muestreo utilizando un solo canal es de 10 KHz.

El conversor posee además un multiplexor que permite manejar 8 señales analógicas, lo que reduce la frecuencia máxima de muestreo posible 8 veces. En este caso se utilizan sólo 4 entradas, lo que permite una frecuencia máxima de 2,5 KHz.

3.6 Manejo del puerto paralelo

El puerto paralelo se puede trabajar en 5 diferentes modos de operación:

- Modo de compatibilidad.
- Modo nibble.
- Modo byte.
- Modo EPP.
- Modo ECP.

La diferencia fundamental entre estos modos es la forma en que transmiten sus datos. Los 3 primeros modos trabajan con el puerto paralelo normal ó SPP, y los 2 últimos trabajan con el puerto paralelo bidireccional.

En este caso se ha decidido trabajar con el modo Nibble, para lo cual se utilizan solamente 4 bits del registro de estado (bits 10, 11, 12, 13) para la transmisión de 1 byte. Esto se logra por medio del multiplexor 74LS157 (Fig. 13), el cual se encarga de enviar primero el nibble de la parte inferior (los primeros 4 bits), y luego el nibble de la parte superior (los últimos 4 bits), conformando de esta manera el byte que se desea transmitir.

El puerto paralelo posee 1 puerto físico (1 solo conector), pero 3 puertos lógicos, esto quiere decir que se tienen a disposición 3 direcciones lógicas para la configuración de este, las cuales son:

Tabla 2

PUERTO	DIRECCION (base)
LPT1	278h
LPT2	378h
LPT3	3BCh

Para el desarrollo práctico del proyecto, se trabajó con la dirección base 378h, correspondiente a LPT2. Esta dirección posee a su vez 3 registros de 1 byte cada uno, con los cuales se puede tener acceso a todos los pines del puerto paralelo.

Los registros son los siguientes: registro de datos, registro de estado y registro de control. A continuación se presentan unas tablas que ilustran los diferentes registros, con sus respectivos pines, nombres y lógica.

Tabla 3

REGISTRO DE DATOS: (Base + 0) = 378h

NOMBRE	D7	D6	D5	D4	D3	D2	D1	D0
PINES	9	8	7	6	5	4	3	2
LOGICA INVERSA	NO	NO	NO	NO	NO	NO	NO	NO

Tabla 4

REGISTRO DE ESTADO: (Base + 1) = 379h

NOMBRE	Busy	Ack	Paper out	Select in	Error	*	*	*
PINES	11	10	12	13	15	*	*	*
LOGICA INVERSA	SI	NO	NO	NO	NO	*	*	*

Tabla 5

REGISTRO DE CONTROL: (Base + 2) = 37Ah

NOMBRE	*	*	*	*	Selec print	Init print	Auto linefeed	Strobe
PINES	*	*	*	*	17	16	14	1
LOGICA INVERSA	*	*	*	*	SI	NO	SI	SI

Como se puede apreciar en las tablas anteriores, hay algunos bits que utilizan lógica inversa dentro del puerto, por lo que es necesario negarlos. Esto se lleva a cabo mediante software, haciendo uso de máscaras y funciones matemáticas que se encuentran preestablecidas en el lenguaje de programación que se usa (C++), ya que si se hubieran utilizado inversores a la entrada del puerto, los tiempos que estas lógicas emplean para pasar de un estado a otro jugarían un papel adverso en la adquisición.

3.7 Multiplexor 74LS157

Se emplea para transmitir la información del conversor A/D 0808 (1byte = 8 bits), al puerto paralelo en forma de nibble. Esto se logra conectando la salida de 8 bits del conversor A/D a los pines 1a, 1b, 2a, 2b, 3a, 3b, 4a, 4b del multiplexor, y la salida 1Y, 2Y, 3Y, 4Y, de 4 bits del multiplexor a los pines 10, 11, 12, 13 del puerto paralelo.

El multiplexado de los datos se controla por medio del pin 2 del puerto paralelo, el cual se conecta al pin 1 del multiplexor, y funciona de la siguiente manera: cuando este bit se pone en uno por medio de software, se transmite los 4 primeros bits (nibble1) del dato del convertidor A/D, y cuando el bit se pone en cero, se transmite los 4 últimos bits (nibble2) del dato del convertidor.

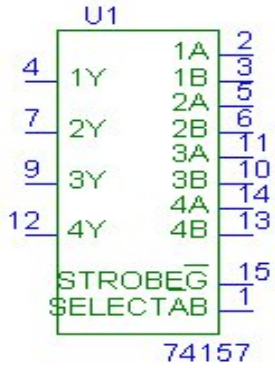


Figura 13: Multiplexor 74LS157

Los bits de salida del multiplexor (modo nibble) son introducidos al puerto paralelo por los pines 10, 11, 12, 13. Estos pines son parte del registro de estado, al cual se accede por medio de la dirección 379h.

Se debe tener en cuenta que los pines de datos 10, 11, 12, 13 del puerto paralelo se ubican dentro del registro de estado en las posiciones bit7, bit6, bit5 y bit4 respectivamente. Para el caso del nibble menos significativo se debe hacer un desplazamiento de cuatro bits a la derecha lo que permitirá que el dato tome las posiciones bit3, bit2, bit1, bit0. Luego se debe enmascarar el dato realizando una operación AND entre el registro desplazado y el dato binario "00001111", de esta manera solamente se tendrán en cuenta los 4 primeros bits, que es dónde se encuentra la información del nibble.

El nibble más significativo entra al registro de estado por las mismas líneas que el menos significativo, pero en esta caso no se realiza ningún desplazamiento, sólo se lo enmascara realizando la operación AND con el dato binario ”11110000”.

Para formar el byte transmitido originalmente se realiza la operación binario OR entre estos dos bytes.

3.8 Software

El programa se desarrolla bajo el lenguaje C++, contando con las siguientes características desde el punto de vista de la adquisición:

- Leer la línea EOC del ADC.
- Seleccionar de a uno 4 canales analógicos.
- Seleccionar el nibble manejando la línea SELECT de multiplexor.
- Leer el nibble seleccionado y armar el byte.
- Graficar las 4 señales simultáneamente.
- Posibilidad de graficar sólo una señal en pantalla.
- Detección de pico R y cálculo de frecuencia cardiaca.
- Guardar en archivos con extensión ".sgn " los datos adquiridos.
- Posibilidad de detener la gráfica de la señal en cualquier momento.
- Posibilidad de ver datos almacenados con anterioridad.

Este lenguaje posee las funciones **inport**() y **outport**(). Como su nombre lo indica, estas funciones son las encargadas de recibir y enviar datos a través de los puertos.

La función InPort posee un solo parámetros de entrada: La dirección de registro, que especifica la posición del registro empleado, por lo tanto es un dato numérico. La salida de esta función consiste en un dato de tipo numérico.

La función OutPort tiene dos parámetros de entrada: La dirección de registro, que especifica las líneas por donde se envía la información; y el dato que se va a enviar.

Estas dos funciones son las empleadas para el manejo del ADC 0808 y la posterior adquisición del dato a través del puerto paralelo. Estas funciones se organizan en forma de secuencia de la siguiente forma:

- Selección del canal.
- Lectura de la línea EOC.
- Recepción del dato a través del puerto paralelo en modo nibble.

La selección del canal se realiza con los bits 3 y 4 del puerto, correspondientes a los bits 1 y 2 del puerto de datos respectivamente (D1 y D2), que maneja las líneas AddA y AddB. La línea AddC es conectada a masa para asegurarse de que solo se seleccionen las cuatro primeras de las ocho entradas disponibles.

Tabla 6

D1	D2	D3	IN
0	0	0	1
0	1	0	2
1	0	0	3
1	1	0	4

El inicio de la conversión (líneas START y ALE) es comandado por un clock externo con una frecuencia que sea, como mínimo, 4 veces la máxima frecuencia de la señal de entrada. En este caso la frecuencia máxima de la señal es de unos 100 Hz, lo que indica que la frecuencia de muestreo mínima es de 400 Hz. Por otro lado, y como se mencionó anteriormente, la máxima frecuencia de muestreo es 2,5 KHz. La frecuencia de muestreo a utilizar es cercana a los 400 Hz, ya que a mayor frecuencia, mayor es la cantidad de puntos, y por ende, mayor la cantidad de memoria a utilizar. Con el circuito

construido se obtuvo una frecuencia del orden de 500 Hz, pero como van a comandarse cuatro canales de entrada con el mismo generador de frecuencia de muestreo, será necesario que dicha frecuencia sea cuatro veces la frecuencia de muestreo de cada canal, es decir, del orden de 2 KHz. El circuito del generador de frecuencia de muestreo se muestra en la figura 14.

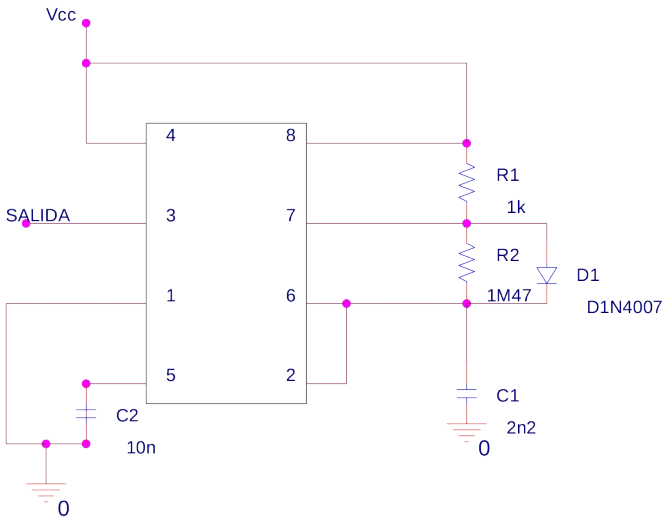


Figura 14: Generador de frecuencia de muestreo.

Cuando las líneas START y ALE pasan de estado alto a estado bajo la línea EOC pasa a estado bajo. Luego, la lectura de la línea EOC del conversor se hace leyendo el estado del bit 15 (Error) del puerto paralelo. Cuando éste pase del nivel bajo (0) al alto (1), significa que el ADC a finalizado la conversión del dato.

Para llevar a cabo la recepción del byte, se debe manejar la línea SELECT del multiplexor con el bit 2 del puerto paralelo (D0), el cual seleccionará el nibble a leer: con un cero se seleccionará el nibble menos significativo, y con un uno se seleccionará el nibble más significativo. La lectura de estos nibbles, se hace a través de los bits 10, 11, 12 y 13 del puerto paralelo, que corresponden a las líneas ACK, Buzzy, Paper Out y Select In respectivamente.

Por último se debe enmascarar el dato realizando una operación AND entre el registro desplazado en cuatro posiciones hacia la derecha y el dato binario "00001111". De esta manera solamente se tendrán en cuenta los 4 primeros bits, que es donde se encuentra la información del nibble. Para el caso del nibble más significativo al dato leído sólo se lo enmascara realizando la operación AND con el dato binario ”11110000”. Luego, para formar el byte transmitido originalmente se realiza la operación binaria OR entre estos dos bytes (0000XXXX , XXXX0000), como se menciono anteriormente.

En la figura 11 se puede ver el diagrama de conexiones entre el conversor y el puerto paralelo de la PC, incluyendo el multiplexor de entrada a éste (74LS157), y el generador de la señal de clock del ADC.

Cabe destacar que la Vref(-) y Vref(+), están conectadas a 0 Volt y 5 Volt respectivamente. A su vez la línea Vcc es conectada a 5 Volt.

La línea OE, que es la que habilita la salida de datos, es manejada por la línea strobe del puerto paralelo

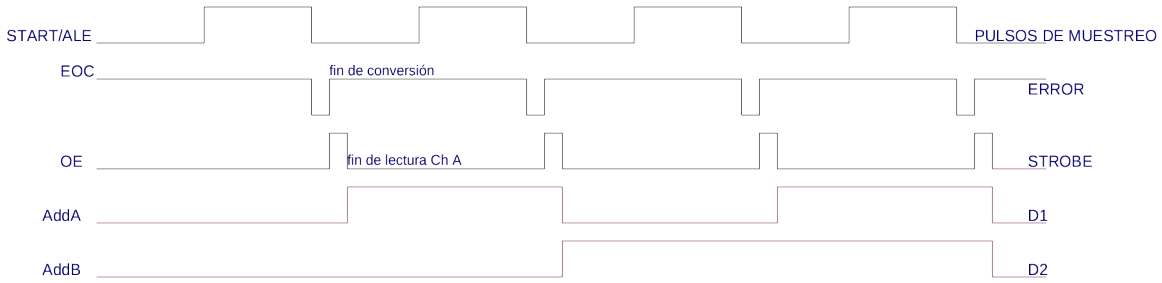


Figura 15: Diagrama de tiempo de las líneas de comando conversor-puerto

3.9 Acondicionamiento de las señales analógicas de entrada

Como se dijo anteriormente, las señales que se le suministran al conversor poseen un nivel de tensión muy bajo, del orden del milivolt, por lo que se las deben amplificar.

En lo que se refiere al proceso de toma y amplificación de estas señales, ya sea en forma unipolar o bipolar, se utiliza el principio del amplificador diferencial.

Cuando se emplea un amplificador diferencial para medir estas señales, que se captan como diferencia de potenciales entre dos electrodos, se aplican entre la entrada inversora y no inversora del amplificador. Sin embargo, para la señal de interferencia, ambas entradas aparecen como si estuvieran conectadas juntas a una fuente de entrada común. De este modo, la señal de interferencia en modo común queda amplificada solo por la ganancia en modo común, mucho menor que la ganancia diferencial.

Otro punto a tener en cuenta son las impedancias de los electrodos, que forman cada una de ellas, un divisor de tensión con la impedancia de entrada del amplificador diferencial. Si las impedancias de los electrodos no son idénticas, las señales de interferencia de la entrada inversora y la no inversora del amplificador diferencial, pueden ser diferentes, no obteniéndose el grado de anulación deseado.

Dado que las impedancias de los electrodos nunca se pueden hacer exactamente iguales, un factor de rechazo en modo común (CMRR) alto en el amplificador diferencial, solo se puede lograr si el amplificador tiene una impedancia de entrada mucho mas alta que la impedancia de los electrodos a los cuales esta conectado.

El amplificador de instrumentación es un dispositivo electrónico con características muy especiales, compuesto básicamente por tres amplificadores operacionales.

Debido a que la impedancia de entrada es relativamente baja para los amplificadores diferenciales, se pueden reforzar o aislar las entradas con seguidores de voltaje. Esto se lleva a cabo con dos amplificadores operacionales conectados como seguidores de voltaje el cual se caracteriza por una ganancia igual a uno, una impedancia de entrada alta y una impedancia de salida baja.

Además de la mejora anterior es posible mediante tres resistencias mas, dotar al circuito de ganancia ajustable, a este circuito se lo denomina amplificador de instrumentación, y es la configuración con mejores resultados en lo que a toma y amplificación de señales de este tipo se refiere.

Hay que tener en cuenta que estas señales toman valores de tensión negativa, los cuales no son aceptados por el conversor. Para solucionar este problema se le debe desviar el voltaje de salida a un nivel distinto de cero (voltaje de salida diferencial).

Los valores de las resistencias son adoptados de forma tal que la señal que se esta adquiriendo (del orden del milivolt), una vez amplificada, no supere los 5 Volt pico a pico, esto se debe al nivel de continua que se le debe adicionar, el cual es de unos 2,5

Volt para asegurar que en todo momento la señal que ingresa al conversor sea mayor que 0 Volt.

Como se mencionó anteriormente, se desea una ganancia de unas 600 veces y una buena cifra de ruido, lo que se logra haciendo que la mayor parte de la amplificación se lleve a cabo en la primera etapa (V0/Vin). Se eligieron los valores de las resistencias para que esta etapa ganara cerca de 50 veces y de acuerdo a la función de transferencia:

$$\frac{V0}{Vin} = (1+2/a)$$

Donde a=R1/R2, con R2=R3=10K y R1=470, lo que da una ganancia teórica de 43,55 veces. Las resistencias R4,R5,R6 y R7 se eligieron de forma tal que la ganancia sea unitaria.

La etapa siguiente cumple la doble función de amplificar y filtrar la señal. Para amplificar 600 veces es necesario amplificar 14 veces más en esta etapa. Esto se lleva a cabo a través de la relación R9/R8=220K/15K=14.666 veces.

Las especificaciones de ancho de banda para señales cardíacas son de 0.05Hz a 100Hz, este ancho de banda se maneja mediante la selección de los capacitores de esta etapa. La transferencia del filtro pasa banda es:

$$Vs(s)/V0(s) = sR9C1/((1+sC1R8)(1+sC2R9))$$

Como puede observarse existe un cero en el origen y dos polos, el dado por R8C1 de muy baja frecuencia, y el otro, dado por R9C2, a una frecuencia superior. Para respetar el ancho de banda especificado el polo de baja frecuencia debe encontrarse por debajo de 0.05 Hz, y el otro en 100 Hz. Es por esto que los valores elegidos fueron:

C1= 220uF, lo que da una ubicación teórica del polo en: $fl=1/(2\pi R8C1)=0.048$ Hz

C2= 6,8 nF, lo que da una ubicación teórica del polo en: $fh=1/(2\pi R9C2)=106,39$ Hz

Como se desea poder variar la amplitud de la señal que ingresa al conversor se colocó a la salida del amplificador un divisor resistivo compuesto por tres resistencias de 1K, lo que permite atenuar a la misma x1, x1,5 y x3. Estas tres resistencias están montadas sobre el nivel de continua al que se hizo referencia al principio de este apartado y que sirve para que la señal que ingresa al conversor no tenga valores negativos. Por otro lado, se colocó un capacitor de acople de alterna a la salida del filtro pasa banda. Este capacitor, junto con las tres resistencias del divisor generan un filtro pasa alto con frecuencia de corte teórica en 0,053 Hz, lo que es aceptable para los propósitos de nuestro circuito.

En la figura 16 se observa el circuito completo del amplificador diferencial que corresponde a uno de los cuatro canales.

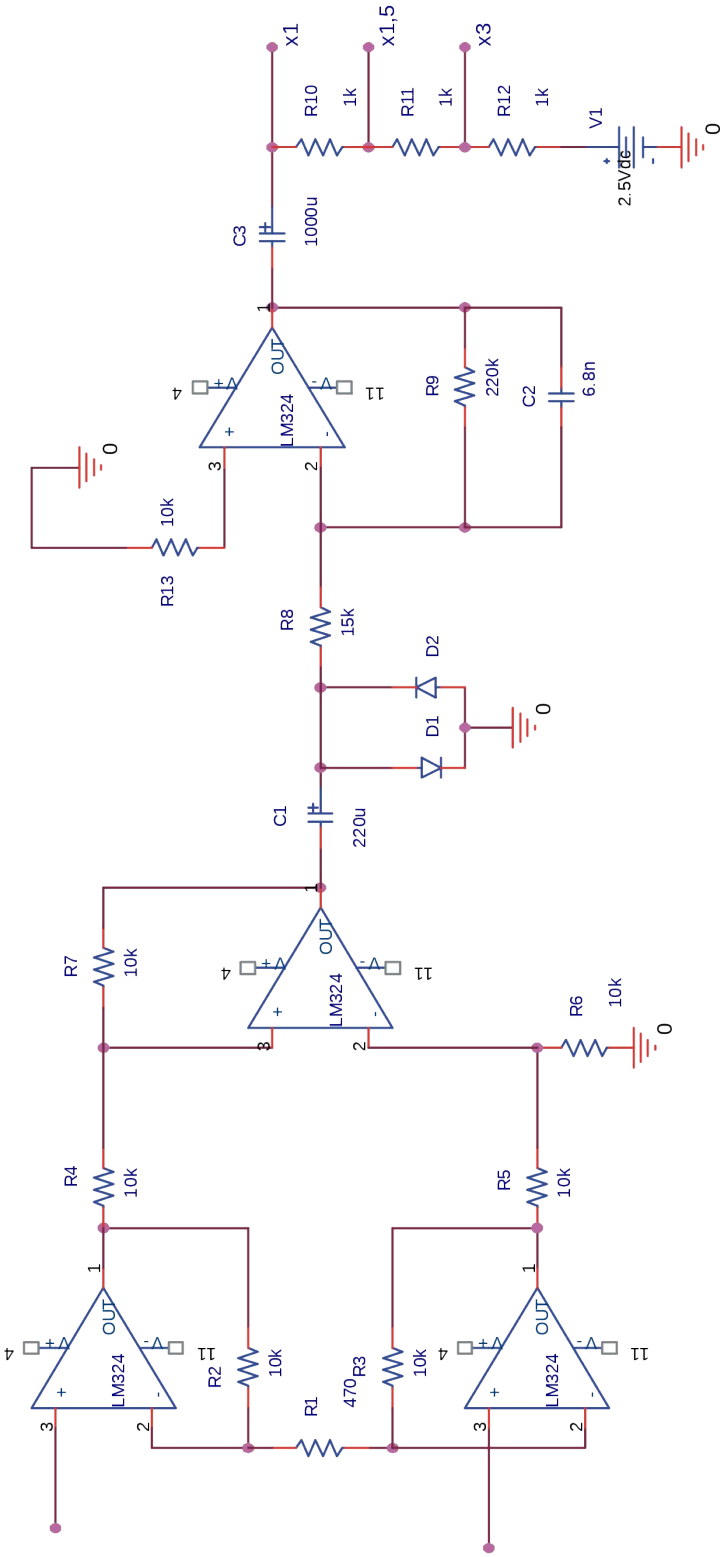


Figura 16: Amplificador diferencial

4- DETECCIÓN DEL COMPLEJO QRS

Antes de entrar de lleno en la descripción del algoritmo de detección resulta conveniente observar las características de la señal con la que se trabajará. La señal de ECG está compuesta por los complejos QRS, las ondas P y T, interferencia de la red (50 Hz), señal de (electromiografía) EMG y "artefactos de movimientos" debido al movimiento de los electrodos. Para poder lograr una detección exitosa se debe separar de la señal al complejo QRS. En la Figura 17 se muestra el espectro relativo de la señal de ECG, los complejos QRS, ondas P y T y otros ruidos, basado en un estudio desarrollado por Thakor en 1983.

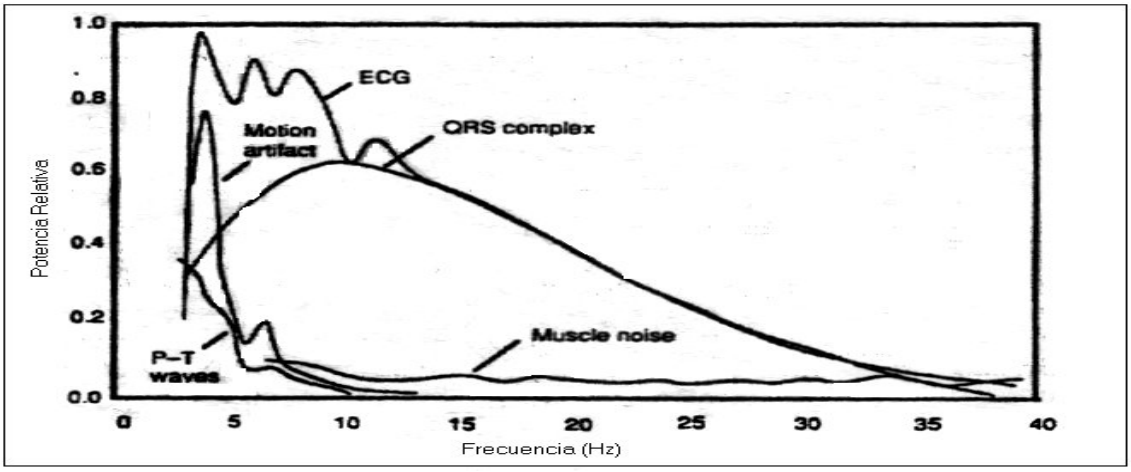


Figura 17. Espectro de la señal ECG.

De otro estudio desarrollado por Thakor (1984), se puede examinar el espectro de potencia de la SNR de los QRS con respecto a todos los demás ruidos mencionados anteriormente (Figura 18). Thakor muestra que el máximo de la SNR se obtiene utilizando un filtro pasa banda con fc=17 Hz y Q=3.

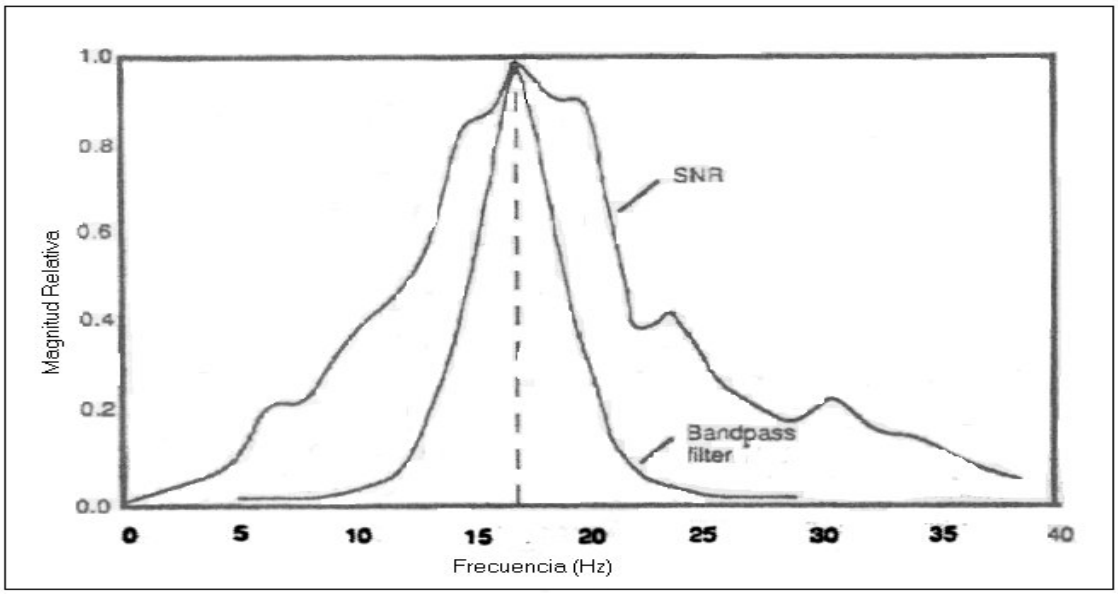


Figura 18. SNR entre el complejo QRS y los distintos tipos de ruido

4.1 Origen del Algoritmo Desarrollado

El algoritmo implementado esta basado en el desarrollado por Pan y Tompkins. En la Figura 19 se muestra un diagrama de bloques de este algoritmo, donde se observan los filtros involucrados en el análisis de la señal de ECG.

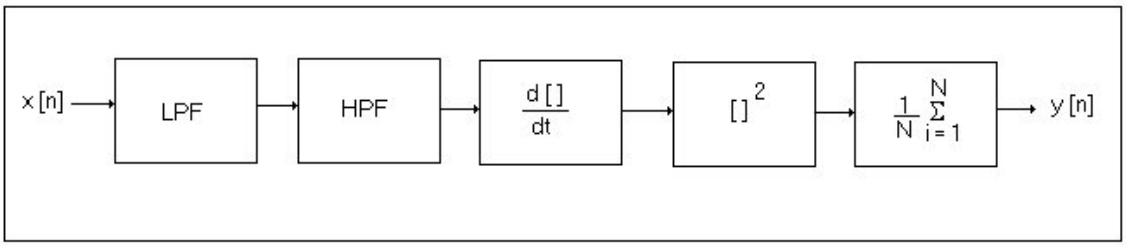


Figura 19. Diagrama de bloques de la etapa de filtrado.
LPF: Filtro Pasa Bajo
HPF: Filtro Pasa Alto

El algoritmo de Pan y Tompkins utiliza un filtro pasa banda (realizado con un LPF y un HPF), para reducir el ruido junto con las señales interferentes fuera de la banda de frecuencias en la que se encuentra el QRS. A continuación hay un derivador que enfatiza las pendientes de la onda R, un bloque que eleva al cuadrado a la señal de salida del derivador, para aumentar aún más las altas frecuencias del complejo QRS y, por último, se hace una estimación de la energía con una ventana móvil del tamaño del QRS más largo sin llegar a la onda T.

En la Figura 20 se muestra a modo de ejemplo la señal obtenida luego de pasada por los diferentes bloques mencionados anteriormente:

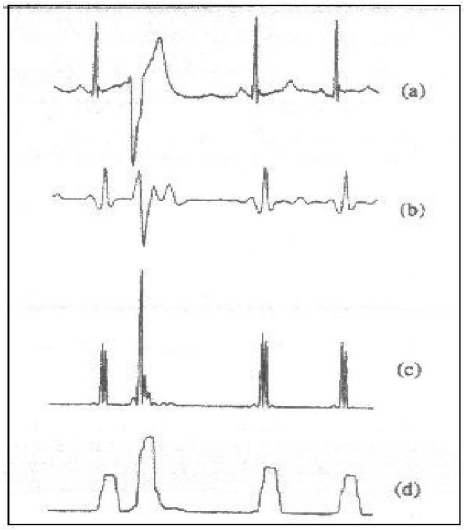


Figura 20: a) Señal original.
b) Salida de filtro pasa banda.
c) Salida después de la diferenciación y elevación al cuadrado.
d) Señal final después de ventana integradora.

La detección de los picos de la señal de salida de la ventana integradora se lleva a cabo comparando a la señal con umbrales. Cada vez que la señal supera dicho umbral, se está en presencia de un posible complejo QRS.

Se comenzó a implementar este algoritmo y se observó que se adaptaba muy bien frente a ruidos pequeños, pero en casos extremos como alto nivel de ruido o pérdida de señal, consideraba picos de ruido como de señal, detección de picos erróneos y/o a la pérdida de picos de señal. Para solucionar estos problemas se implementó un algoritmo de detección de picos de la señal filtrada, utilizando umbrales de decisión adaptativos el cual se explica detalladamente en la siguiente sección.

4.2 Algoritmo de Detección Desarrollado

El algoritmo desarrollado posee dos partes bien diferenciables: una de ellas es la de filtrado de la señal ECG que se basa en lo descrito en el punto anterior; la otra es la etapa de detección de los picos de la señal filtrada.

A continuación se realiza una explicación detallada del los filtros y el algoritmo de detección de picos utilizados.

4.2.1 Filtro Pasabajos:

Se decidió insertar un filtro pasabajos Butterworth de orden 8, con frecuencia de corte fc=20 Hz. De esta manera se atenúan aún más las "frecuencias altas" evitando así que el detector las interprete como complejos QRS.

Para la frecuencia de muestreo utilizada (fs=500 Hz), se calcularon en MATLAB los coeficientes del filtro de Butterworth de orden 8, cuya transferencia es:

$$H(z)=\frac{(0.0034+0.0272z^{-1}+0.0952z^{-2}+0.1903z^{-3}+0.2379z^{-4}+0.1903z^{-5}+0.0952z^{-6}+0.0272z^{-7}+0.0034z^{-8})\times 10^{-5}}{1-6.7121z^{-1}+19.7998z^{-2}-33.5162z^{-3}+35.5989z^{-4}-24.2887z^{-5}+10.3935z^{-6}-2.5498z^{-7}+0.2475z^{-8}}$$

En la Figura 21 se observa la gráfica de la transferencia dada en función de la frecuencia:

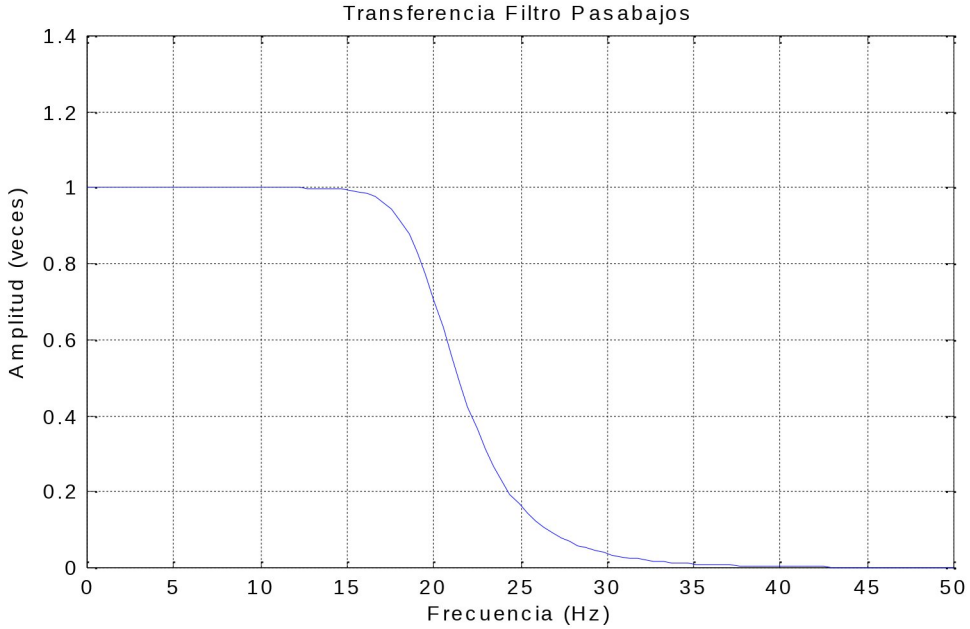


Figura 21.

4.2.2 Filtro Pasaaltos

El filtro pasaaltos finalmente implementado es un filtro de Butterworth de orden 8 con una frecuencia de corte fc= 15 Hz, con lo que junto al filtro pasabajos anterior emulan un filtro pasabanda con un ancho de anda de BW=5 Hz, centrado en aproximadamente fo=17 Hz, Para la frecuencia de muestreo utilizada (fs=500 Hz), se calcularon con MATLAB los coeficientes del filtro de Butterworth de orden 8, cuya transferencia es:

$$H(z)=\frac{0.6163-4.9304z^{-1}+17.2563z^{-2}-34.5125z^{-3}+43.1407z^{-4}-34.6125z^{-5}+17.2563z^{-6}-4.9304z^{-7}+0.6163z^{-8}}{1-7.0340z^{-1}+21.6983z^{-2}-38.3362z^{-3}+42.4248z^{-4}-30.1102z^{-5}+13.3829z^{-6}-3.4055z^{-7}+0.3798z^{-8}}$$

En la Figura 22 se observa la gráfica de la transferencia dada en función de la frecuencia:

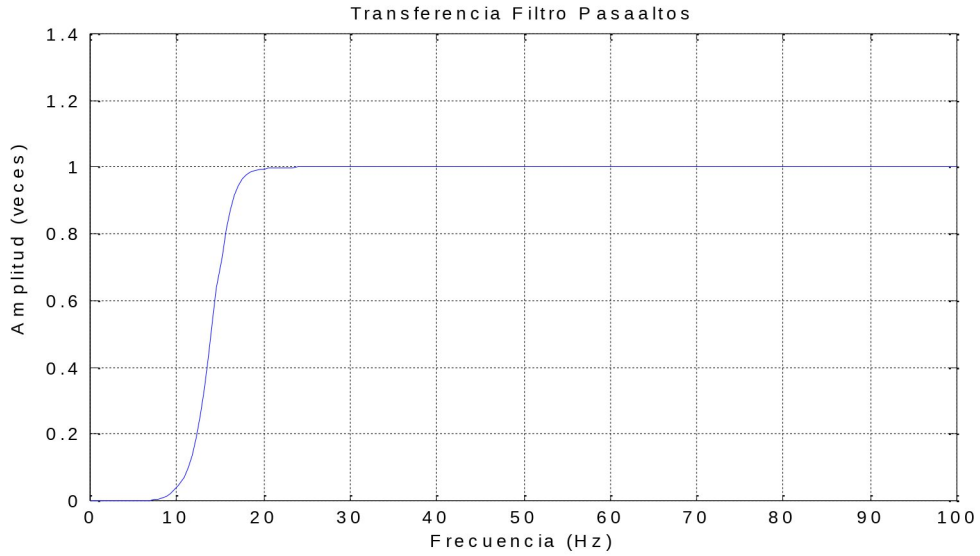


Figura 22

4.2.3 Derivador

Luego que la señal es filtrada, la mayor energía de ésta se halla en el complejo QRS. La diferenciación acentúa las altas frecuencias y atenúa las bajas, por lo tanto resalta las altas pendientes por las cuales generalmente se distinguen los QRS dentro de la señal de ECG. Se observa en la Figura 23 que el filtro utilizado funciona como derivador en el rango de frecuencias deseadas para frecuencias de muestreo mayores a 200 Hz.

La derivada fue implementada mediante la transferencia:

$$H(z)=0.1\left(2+z^{-1}-z^{-3}-2z^{-4}\right)$$

En la Figura 23 se observa la gráfica de la transferencia del derivador dada en función de la frecuencia:

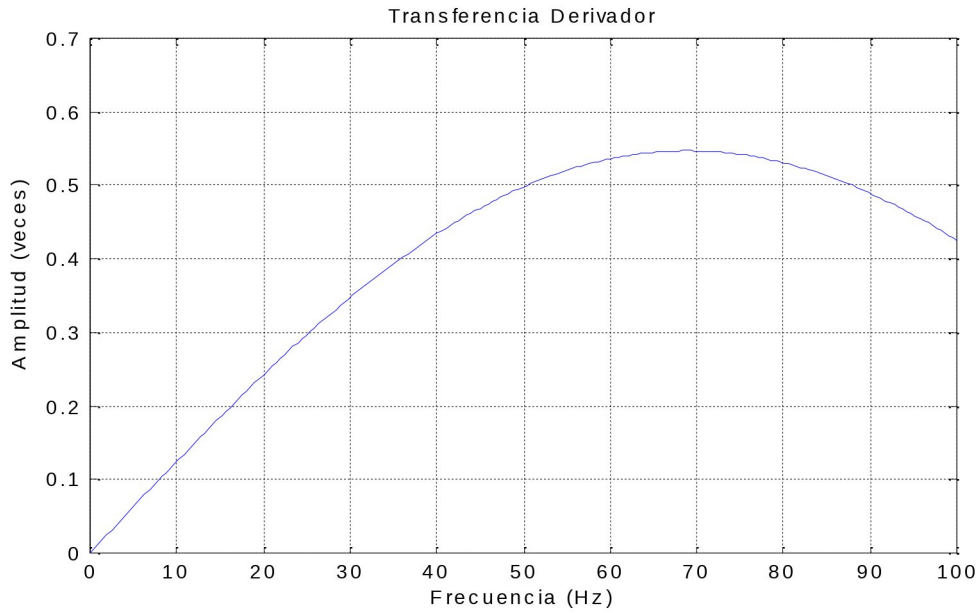


Figura 23

La salida de filtro derivador se multiplica por una constante que depende de la frecuencia de muestreo de forma de ecualizar la ganancia del filtro para las distintas frecuencias de trabajo.

4.2.4 Elevación al cuadrado

Antes de realizar el proceso de ventana integradora, la señal es elevada al cuadrado para que todos sus puntos sean positivos, y para enfatizar la señal de alta frecuencia que es principalmente el complejo QRS.

4.2.5 Ventana integradora

La utilización de la información de la pendiente de la onda R (salida del filtro derivador) no es suficiente, pues muchos complejos anormales que tengan largas amplitudes y largas duraciones podrían no ser detectados utilizando solo esta información. Por lo tanto se implementa una ventana integradora para estimar la energia de los complejos QRS.

Ésta es implementada con la siguiente ecuación en diferencias:

$$y(nT) = \frac{1}{N}\left[x(nT - (N-1)T) + x(nT - (N-2)T) + \ldots\ldots\ldots + x(nT)\right]$$

El tamaño de la ventana (N) debería ser siempre mayor o igual al QRS de mayor duración, pero si la ventana es muy larga la integración podría sumar la información de la onda T y si es muy corta no se amplificaría lo suficiente, además podría producir la detección de picos erróneos.

En la Figura 24 se muestra la relación entre el complejo QRS y el ancho de la ventana.

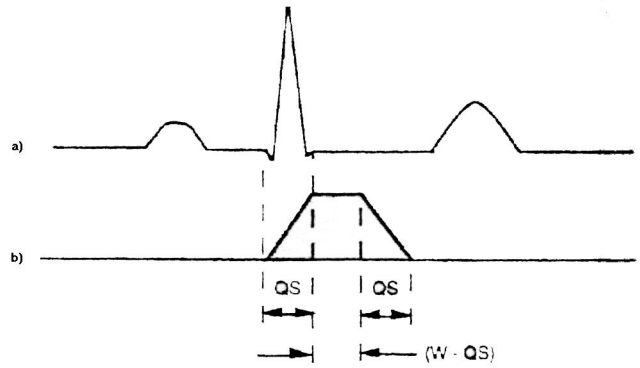


Figura 27. a) Señal ECG.
b) Salida de ventana integradora

Para este trabajo se tomó una ventana de 80 ms, es decir, que a la frecuencia de muestreo dada (500Hz) se trabaja con 40 muestras.

4.2.6 Algoritmo de detección de picos

El algoritmo de detección de picos consta de las siguientes etapas:

- Búsqueda del primer punto de la señal filtrada que supere el umbral de detección
- Búsqueda del máximo absoluto en una ventana de la señal filtrada
- Determinación del punto R en la señal ECG
- Actualización del salto
- Actualización del umbral

A continuación se describen detalladamente cada una de ellas:

a) Búsqueda del primer punto de la señal filtrada que supere al umbral de detección

Una vez detectado un pico R en la señal filtrada se actualizan las variables *salto* y *umbral*. *Salto* indica el número de muestras que deberán adicionarse al índice de la muestra R para encontrar el punto de comienzo de búsqueda del nuevo pico.

Luego de realizado el salto (Figura 25) se comienza a buscar cual es el próximo punto en que la señal filtrada supera el umbral de detección. Esta búsqueda no se realiza punto por punto ya que tendría un costo computacional muy alto, por lo que se optó por realizar la comparación cada 16ms (8 muestras para fs=500 Hz).

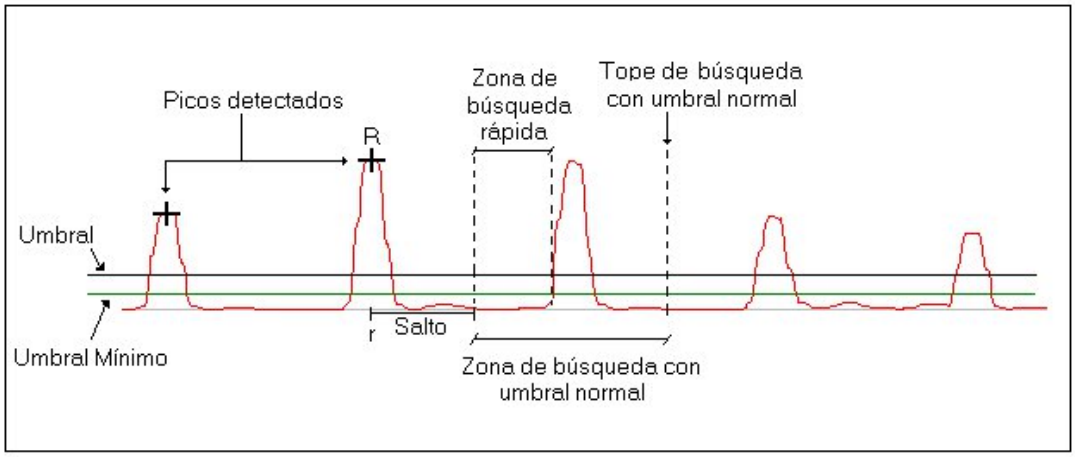


Figura 25. Descripción del salto y zona de búsqueda de pico

Cuando se detecta el punto cuya señal filtrada supera el umbral, se pasa al punto b).

En caso de que no se encuentre ese punto dentro de un intervalo de duración igual al promedio de los últimos ocho intervalos RR (RRpromedio) el algoritmo entra en una fase de rebúsqueda, vuelve al punto donde comenzó la búsqueda solo que ahora compara a la señal filtrada con el umbral mínimo, puede que la búsqueda resulte infructuosa por dos motivos: uno es que la altura del pico no sea lo suficiente como para superar el umbral, ó que realmente no hubiese un complejo QRS en el intervalo buscado (gran variabilidad de frecuencia cardíaca).

Estos cambios se aprecian claramente en la Figura 26.

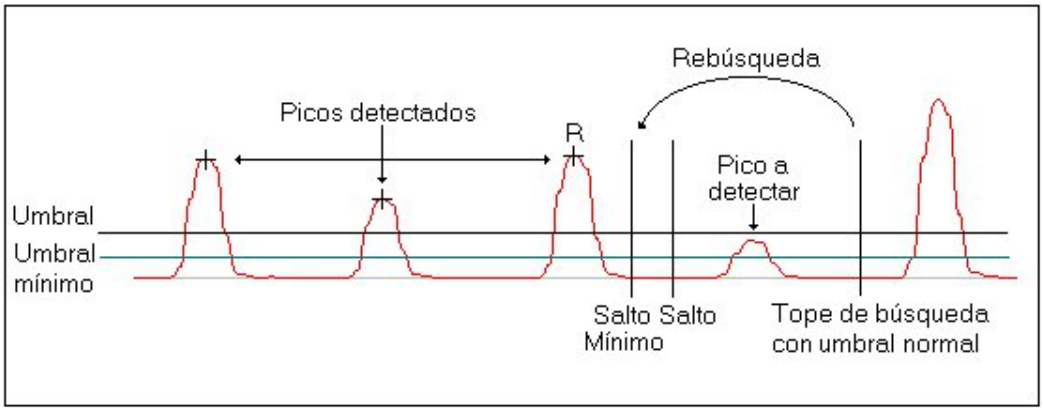


Figura 26. Algoritmo de rebúsqueda

b) Búsqueda del máximo absoluto en una ventana de la señal filtrada

Una vez superado el umbral comienza la búsqueda del máximo. Esto se hace guardando en un vector la porción de la señal que se encuentra por encima del umbral y guardando en una variable el índice del punto donde la señal superó el umbral. Luego se busca el máximo de ese vector y se lee el índice que posee, este índice sumado al que se encuentra guardado en la variable, nos da la posición exacta del pico R en la señal filtrada. Esto tiene por cometido evitar la detección de un máximo relativo que no sea realmente el máximo absoluto del pico de la señal filtrada en ese intervalo ya que la señal presenta leves ondulaciones. Una cosa que se debe tener en cuenta es la cantidad de muestras del vector donde se esta guardando la señal, ya que puede suceder que por efectos del ruido u otros fenómenos, la señal se encuentre por encima del umbral por un periodo de tiempo superior al común, tomándose dos elevaciones de dos picos diferente como uno, y provocando así la perdida de un pico. Es por esto que se tomó como un máximo de puntos para este vector el promedio de los últimos diez tamaños de vectores, y para los 10 primeros se toma un tamaño de vector correspondiente a una ventana de 160 mseg (80 muestras para fs=500). Esta ventana resulta de un compromiso entre el tiempo de cálculo y su efectividad. Por un lado cuanto más grande sea ésta, más se tarda en hallar su máximo y si es muy pequeña no necesariamente su máximo coincida con el máximo a determinar.

d) Determinación del punto R en la señal ECG

Con el máximo encontrado en la señal filtrada se está en condiciones de hallar el punto R de la señal ECG. Esto se hace estimando su ubicación a partir del máximo hallado en el ítem anterior. Esta estimación se debe hacer tomando en cuenta los retardos introducidos por los filtros. Teniendo en cuenta el retardo, en la señal original se le aplica una ventana de 200ms (2xQR estimado, 100 muestras para fs=500) de tamaño, centrada en el valor estimado de R. El primer punto de esta ventana se asume como el punto Q, el punto R se determina como más alejado en valor absoluto del punto Q como se puede ver en la Figura 37.

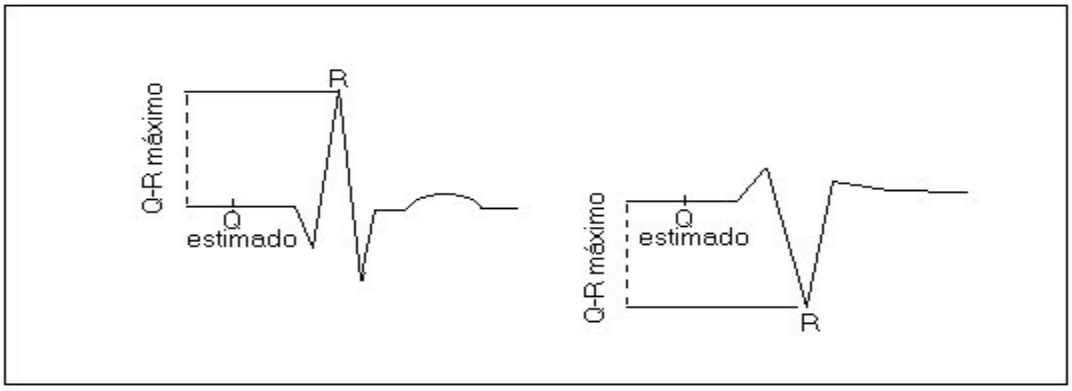


Figura 27. Ejemplos de detección del punto R.

e) Actualización del salto

Como se mencionó anteriormente la actualización del salto se realiza cada vez que se detecta un pico de la señal.

Para evitar una posible pérdida de picos, provocado por el uso de saltos demasiado largos, se utiliza un salto mínimo, que se calcula como:

$$salto\ minimo = 2\sqrt{(ultimo\ segmento\ RR\ (seg))}$$

Antes de este punto es fisiológicamente imposible que aparezca un latido, por encontrarse el ventriculo despolarizado.

f) Actualización del umbral

El umbral también se actualiza luego de la detección de cada pico con la siguiente formula:

umbral = (umbral + altura pico/4)/2))

Este umbral luego se compara con el umbral mínimo calculado para el minuto en el cual se está detectando, no pudiendo ser menor que el umbral mínimo fijado, caso en que se toma el umbral igual al umbral mínimo.

El umbral mínimo en cambio se actualiza una vez por minuto o cada 80 u 85 picos detectados, como un porcentaje de la altura promedio de estos picos.
Este primer umbral mínimo estimado se calcula como el 5% de la altura promedio.

Es conveniente recordar que todos estos valores pertenecen a la señal filtrada que es de acuerdo a la cual se actualiza el umbral.

Este umbral mínimo estimado no se utiliza directamente, ya que primero se verifica que no sea menor que el 25% del umbral mínimo del minuto anterior, esto previene que ante la pérdida de señal no disminuya el umbral mínimo.

Luego de estas verificaciones se calcula el umbral mínimo como:

umbral mínimo = **0.75 x** *umbralMinimoEstimado* **+ 0.25 x** *umbralMinimoMinutoAnterior*

5- ANÁLISIS DE LA SEÑAL CON REDES NEURONALES

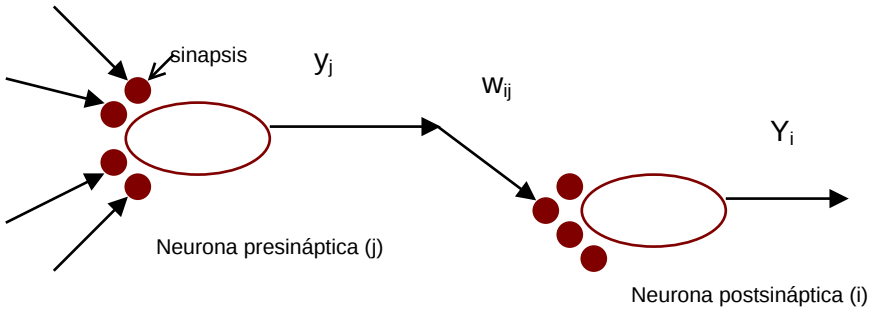
5.1 Introducción a las redes neuronales

Las señales adquiridas y guardadas utilizando el sistema descrito en las secciones anteriores serán procesadas para extraer características útiles para su posterior análisis. En cuanto a la implementación de los algoritmos para obtener dichas características, se detalla en otra sección.

Llamamos características útiles a aquellos parámetros de la señal que sirven para poder clasificarla como normal o con algún tipo de anomalía. Son ejemplos de estos parámetros: el período entre dos picos R, la duración del complejo QRS, la altura de la onda R, etc. Una vez obtenidas, estas características se utilizarán como entradas de la red neuronal.

Las redes neuronales imitan la estructura neuronal del cerebro para reproducir sus capacidades. Las neuronas son los procesadores elementales de información, la cual llega a través del axón que actúa como una línea de transmisión del impulso nervioso. Las neuronas se conectan entre sí, y esa unión entre neuronas se denomina sinapsis y está basada en la transmisión química: la neurona *presináptica* libera neurotransmisores que una neurona *postsináptica* sensible a esos neurotransmisores aceptará como estímulo.

Siguiendo estas pautas se llega a la construcción de una red neuronal artificial, cuyas conexiones entre neurona tienen una estructura como la siguiente:



En el gráfico pueden verse dos neuronas, cada una de las cuales corresponde a una capa de la red. La primera neurona (presináptica) recibe la información a través de sus sinapsis. Dentro de la neurona existen una *regla de propagación*, una *función de activación* y una *función de salida*. El resultado es la salida (y_j) de esta neurona que se transmite a la siguiente capa de neuronas. La siguiente neurona se activará en función de su peso sináptico (w_{ij}), que es la medida en que la neurona postsináptica es sensible al impulso originado en la presináptica. En esta capa el impulso transmitido vuelve a procesarse y continúa su camino de propagación.

Las redes neuronales pueden tener diversas arquitecturas: redes de una sola capa, redes multicapa, redes con realimentación, redes “feedforward”, redes con o sin capas ocultas, etc.

Para el diseño de una red neuronal debe elegirse en primer lugar una arquitectura que se adapte a los requerimientos de nuestro problema, es decir, nodos de entrada, capas intermedias si las hubiere, y capa de salida. Además debemos seleccionar el algoritmo de aprendizaje más indicado para nuestra aplicación.

El proceso de aprendizaje de la red es aquel durante el cual se adaptan los parámetros libres de la misma a partir de estímulos del entorno y puede resumirse en la siguiente secuencia:

- La red es estimulada por el ambiente
- En la red neuronal se producen cambios a partir de esa estimulación.
- La red neuronal responde de un nuevo modo al entorno debido a los cambios en su estructura.

Como mencionamos anteriormente, debe elegirse un algoritmo de aprendizaje para una determinada red. Existen diferentes tipos de algoritmos de aprendizaje, tales como: de corrección de error, hebbiano, competitivo, aprendizaje de Boltzman. Cada uno de estos algoritmos otorga un conjunto de reglas para la solución del problema de aprendizaje.

Por otro lado, también es necesario tener en cuenta el modo en que la red se relaciona con el entorno, es decir el paradigma de aprendizaje. En este sentido el aprendizaje puede ser supervisado, no supervisado (auto-organizado) o aprendizaje por refuerzo.

En el primer caso los datos están constituidos por varios pares de patrones de entrenamiento de entrada y salida, y se cuenta con la supervisión de un “maestro” que dispone como elemento de control al *objetivo*. El paradigma no supervisado tiene como conjunto de datos de entrenamiento sólo patrones de entrada y la red se adapta basándose en las experiencias recogidas de patrones de entrenamiento anteriores. Por último, el aprendizaje por refuerzo hace un mapeo entrada-salida a través de un proceso de prueba y error diseñado para maximizar un determinado índice.

5.2 El Modelo de Kohonen

Los diferentes tipos de redes neuronales obedecen a la necesidad de contar con estructuras de procesamiento adecuadas para una aplicación particular. El denominado Modelo de Kohonen resulta útil para el estudio de señales biológicas.

Teuvo Kohonen presentó en 1982 el modelo de Red Neuronal con capacidad para formar *mapas de características* de manera similar a la organización topológica de la corteza cerebral, la cual está organizada en capas bidimensionales (por ejemplo, en el sistema auditivo se detecta organización según la frecuencia a la que cada neurona alcanza la mayor respuesta). Kohonen demostró que, dada una estructura y una descripción funcional de la red, la reiterada aparición de estímulos externos es suficiente para forzar la formación de mapas.

El Modelo de Kohonen pertenece al grupo de algoritmos de codificación del vector. Dicho modelo genera un mapa topológico para ubicar de manera óptima un número fijo de vectores en el espacio de entrada de mayor dimensión y así facilitar la comprensión de datos.

Dos variantes de este modelo son el Mapa Auto-Organizativo (SOM) y el Learning Vector Quantization (LVQ).

El primero corresponde a un paradigma de aprendizaje no supervisado y un algoritmo de aprendizaje competitivo, es decir, que las neuronas de salida compiten para activarse y sólo una de ellas permanece activa ante una determinada información de entrada de la red. En función de esta única neurona activa, llamada vencedora, se ajustan los pesos de las conexiones. Por su parte el LVQ es un método supervisado que utiliza un algoritmo adaptivo que encuentra un conjunto óptimo de vectores de referencia.

5.3 SOM

Como ya se mencionó, este modelo responde a un paradigma de aprendizaje no supervisado, lo que significa que:

- En su entrenamiento no se presentan salidas objetivo que se desean asociar a cada patrón de entrada.
- A partir de un proceso de auto-organización, la red generará ciertos resultados, reflejo de las relaciones de similitud existentes entre los patrones de entrada.
- Durante el proceso de aprendizaje debe descubrir por sí misma regularidades, correlaciones o categorías en los datos de entrada, e incorporarlos a su estructura interna de conexiones (pesos).

El mapa es una grilla de unidades de procesamiento (neuronas). A cada unidad se le asocia un vector de entrada de alguna observación multidimensional. El mapa trata de representar todas las observaciones disponibles con una exactitud óptima. Al mismo tiempo las entradas se ordenan en la grilla, de modo que las entradas similares tienen salidas cercanas y entradas diferentes tienen salidas más alejadas.

La celda que corresponde a cada entrada se encuentra mediante un proceso de regresión secuencial, donde t = 1,2,... es el índice de los pasos. Para cada muestra $\mathbf{x}(t)$, primero se identifica el índice de la neurona ganadora c (best match) a través de la condición

$$\forall i, ||\mathbf{x}(t) - \mathbf{m}_c(t)|| \leq ||\mathbf{x}(t) - \mathbf{m}_i(t)||.$$

Luego, cada una de las celdas de la grilla es actualizada. Los pesos de la neurona ganadora y sus vecinas topológicas son actualizados de forma de parecerse más al vector de entrada. La actualización de los vectores se realiza de la siguiente manera:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{c(\mathbf{x}),i}(\mathbf{x}(t) - \mathbf{m}_i(t)).$$

Donde $h_{c(x)I}$ es la función vecindario, una función decreciente de la distancia entre los nodos c-ésimo e i-ésimo del mapa.

Esta regresión se reitera para todas las muestras de entrada.

Obtención del SOM

En primer lugar se determinan las características topológicas del mapa, es decir la cantidad de neuronas en la capa de entrada y en la de salida. La cantidad de neuronas en la capa de entrada se relaciona con la dimensión de los vectores de entrada. La cantidad de celdas en la grilla (salida) debe elegirse tan grande como sea posible si es que quiere lograrse un “suavizamiento” en el mapeo. Sin embargo un número excesivo de neuronas llevará a un procesamiento computacional mucho más lento.

Una vez elegida la topología de la red se inicializan sus parámetros. Existen diferentes tipos de inicialización:

- Aleatoria
- Con valores del espacio muestral.
- Lineal, donde el peso de las celdas se corresponde con las dos componentes principales del conjunto de datos de entrada.

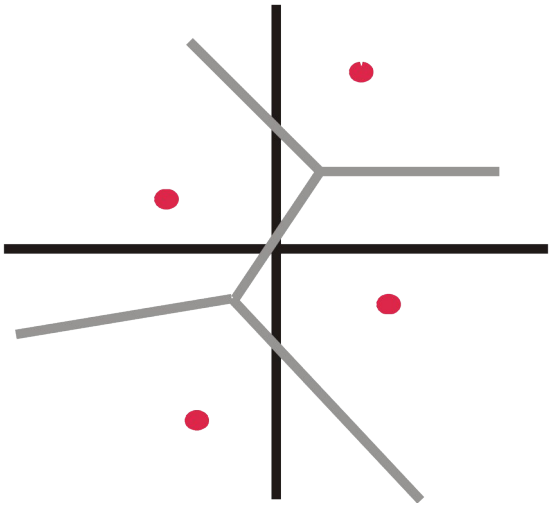
5.4 LVQ

El desempeño de un clasificador puede mejorarse si a continuación de una red SOM se utiliza una técnica de diseño supervisada, tal como el LVQ (learning vector quantization)

Cuantificación de vectores

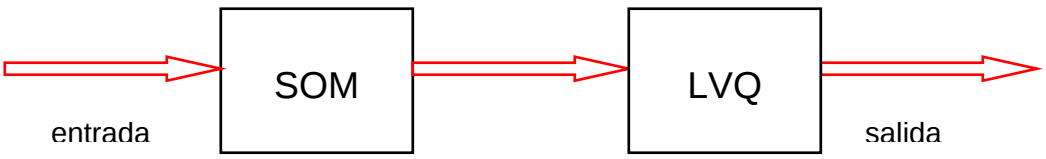
Es una técnica que explota la estructura subyacente de los vectores de entrada con el propósito de comprimir los datos. Esta técnica se utiliza en la codificación para la compresión de ancho de banda. Un espacio de entrada se divide en un número de regiones distintas y para cada región se define un *vector reproducción*. Cuando al clasificador se le presenta un nuevo vector de entrada primero se determina la región en la que cae el vector y luego éste es representado por el vector reproducción correspondiente a esa región.

Un clasificador de vectores con distorsión mínima se denomina *clasificador de Voronoi*. La siguiente figura muestra un ejemplo de un espacio de entrada dividido en cuatro celdas de Voronoi con sus respectivos vectores asociados.



Cada celda de Voronoi contiene aquellos puntos del espacio de entrada que se hallan más cerca del vector de Voronoi que el resto de los puntos.

El algoritmo SOFM (self-organizing feature map) es un método aproximado para hallar los vectores de Voronoi de manera no supervisada. El hallazgo del SOM puede ser visto como la primera etapa en la resolución del problema de clasificación de patrones; la segunda etapa es el LVQ.



LVQ es una técnica de aprendizaje supervisado que usa información de clases para mover levemente los vectores de Voronoi para mejorar la calidad de las regiones de decisión del clasificador. Un vector de entrada $\boldsymbol{x}$ se elige aleatoriamente del espacio de entrada. Si las etiquetas de clase del vector de entrada y el vector de Voronoi, $\boldsymbol{w}$, coinciden, entonces $\boldsymbol{w}$ se mueve en la dirección del vector de entrada $\boldsymbol{x}$. Si, por otro lado, las etiquetas de clase de $\boldsymbol{x}$ y $\boldsymbol{w}$ no coinciden el vector $\boldsymbol{w}$ se aleja del vector $\boldsymbol{x}$.

Algoritmo LVQ

El vector de Voronoi $\boldsymbol{w_c}$ es el más cercano al vector de entrada $\boldsymbol{x_i}$. Sea $\mathbf{C}\boldsymbol{w_c}$ la clase asociada al vector $\boldsymbol{w_c}$ y $\mathbf{C}\boldsymbol{x_i}$ la clase asociada a $\boldsymbol{x_i}$. El vector $\boldsymbol{w_c}$ se ajusta como sigue:

- $\mathbf{C}\boldsymbol{w_c} = \mathbf{C}\boldsymbol{x_i}$ entonces $\boldsymbol{w_c(n+1) = w_c(n) + \alpha_n[x_i - w_c(n)]}$ con $\boldsymbol{0 < \alpha_n < 1}$
- $\mathbf{C}\boldsymbol{w_c} <> \mathbf{C}\boldsymbol{x_i}$ entonces $\boldsymbol{w_c(n+1) = w_c(n) - \alpha_n[x_i - w_c(n)]}$

Los otros valores de Voronoi no se modifican.

Es conveniente que la constante de aprendizaje, $\boldsymbol{\alpha_n}$, sea decreciente en forma monótona con el número de iteraciones, $\boldsymbol{n}$. Luego de varias pasadas de los datos de entrada los vectores de Voronoi convergen y el entrenamiento se completa.

5.5 Diseño de la red neuronal desarrollada para el presente trabajo

En este caso se utilizó una red SOM seguida por una LVQ, lo que, como se dijo anteriormente, mejora el rendimiento del detector. El objetivo es clasificar latidos cardíacos en normales y anticipados. Para ello se realizó la detección de ciertos parámetros que son de utilidad utilizando los algoritmos de filtrado, derivación e integración descriptos en la sección 4. Todo este procesamiento de la señal cardiaca se realiza utilizando MATLAB 5.3, al igual que la implementación de las redes neuronales.

Los parámetros extraídos de la señal para usar como entradas de la red son el período RR entre dos latidos, la diferencia entre dos períodos consecutivos y la duración del complejo QRS. Con estos datos la red es capaz de determinar si los latidos son cortos, normales o largos, y posteriormente si es que hubo un latido anticipado o no.

Cuando existen latidos anticipado, ya sea por contracción prematura ventricular (PVC) o auricular (PAC), luego de un latido de período corto viene uno más largo en relación. De ahí la importancia de la diferencia de período entre dos latidos.

Red SOM

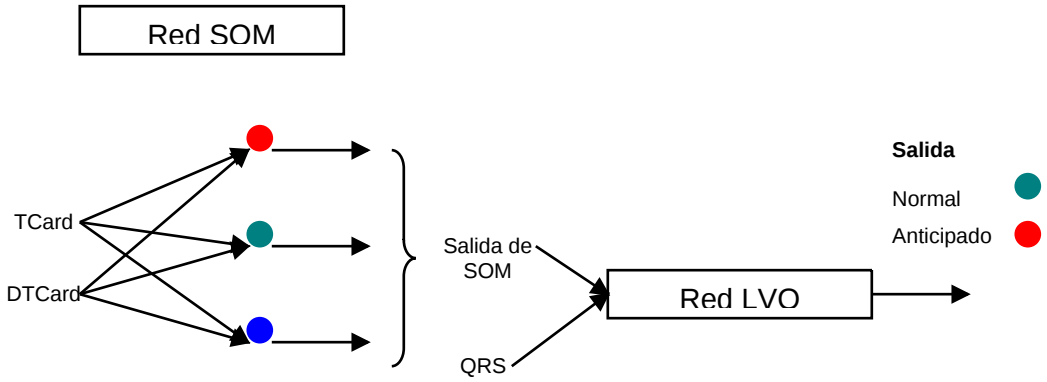
La red SOM elegida fue una con tres neuronas de salida, una para cada tipo de período: corto, normal y largo. La capa de salida actúa en forma competitiva, es decir, dado un vector de entrada sólo una de las neuronas de salida responde (la neurona ganadora) indicando de qué tipo de latido se trata. Las entradas que se le dieron a esta red fueron dos: el período cardíaco, Tcard, y la diferencia entre dos períodos consecutivos, DTCard. Se sabe que SOM utiliza un paradigma no supervisado, por lo tanto no es necesario darle una función objetivo, la red sola se entrena de acuerdo a las entradas que se le brindan con ese propósito.

Para el entrenamiento y prueba de la red se utilizaron señales de la Base de Arritmias del MIT-BIH. Específicamente se utilizó el registro 106 que contiene 2027 latidos entre los

cuales hay 520 PVC multiformes. Este registro se dividió en tres: con el segundo tramo se entrenó la red SOM, con el tercero la LVQ y la primera parte se utilizó para probar la red completa. También se utilizaron para probar la red los registros 100, 105, 200 y 203, cuyas características se detallan en el apéndice D.

Red LVQ

La arquitectura de la red LVQ consiste en una capa de 4 neuronas a las que se le ingresa la salida de la red SOM y la duración del período QRS. La salida de la red LVQ indica si el latido fue normal o anticipado.



Como la red LVQ trabaja con aprendizaje supervisado es necesario darle para su entrenamiento un vector objetivo en el cual se establecen las *clases* para la salida de la red. Una vez entrenada, la red clasificará los puntos de entrada en alguna de las clases especificadas en su entrenamiento.

En cuanto a la utilización de MATLAB para la implementación de las redes, hay que tener en cuenta ciertos parámetros de entrada para la creación y entrenamiento de las mismas.

SOM con MATLAB

NEWSOM crea un mapa auto-organizativo

Sintáxis

net = newsom(PR,[d1,d2,...],tfcn,dfcn,olr,osteps,tlr,tns)

Descripción

Las capas competitivas se usan para resolver problemas de clasificación.

Entradas:

PR - matriz de Rx2 de valores mínimos y máximos para R elementos de entrada. En este caso hay dos elementos de entrada, por lo que esta matriz es de 2x2.

Di - Tamaño de la capa i-ésima, por defecto es [5 8]. En este caso hay una capa de tres neuronas.

TFCN - Función de topología, por defecto = 'hextop'.

DFCN - Función distancia, por defecto = 'linkdist'.

OLR - Velocidad de aprendizaje de ordenamiento de fase, por defecto = 0.9.

OSTEPS – Pasos de ordenamiento de fase, por defecto = 1000.

TLR - Velocidad de aprendizaje de sintonía de fase, por defecto = 0.02;

TND - Distancia de vecindario de sintonía de fase, por defecto = 1.

Propiedades

SOM consiste de una sola capa con función de pesos NEGDIST, función de entrada NETSUM, y función de transferencia COMPET.

La capa tiene los pesos de las entradas pero no tiene umbrales.

Los pesos se inicializan con MIDPOINT.

Descripción de las funciones utilizadas por la red SOM:

- *hextop:* función de topología de capa hexagonal. Calcula las posiciones en las capas cuyas neuronas están arregladas como un patrón hexagonal.
- *Linkdist*: función distancia de conexiones. Sirve para hallar la distancia entre las neuronas de la capa dadas sus posiciones.

Algoritmo: la distancia de conexión D entre dos vectores de posición Pi y Pj de un conjunto S de vectores es:

Dij = 0, si i==j

Dij = 1, si sum((Pi-Pj).^2).^0.5 es <= 1

Dij = 2, si k existe, Dik = Dkj = 1

Dij = 3, si k1, k2 existen, Dik1 = Dk1k2 = Dk2j = 1.

Dij = N, si k1..kN existen, Dik1 = Dk1k2 = ...= DkNj = 1

Dij = S, si no se aplican ninguna de las condiciones anteriores.

- *negdist*: función de peso de distancia negativa. Las funciones de pesos aplican pesos a una entrada para obtener entradas pesadas.

  Algoritmo: negdist devuelve la distancia Euclidiana negativa:

$$z = -sqrt(sum(w-p)^2)$$

- *netsum*: función de entrada suma. Las funciones de entrada calculan las entradas de la red de la capa combinando sus entradas pesadas y umbrales.
- *compet*: función de transferencia competitiva. Las funciones transferencia calculan las salidas de la capa a partir de la entrada a la red.
- *midpoint*: función de inicialización de pesos en el punto medio. Midpoint fija los vectores peso (filas) en el centro de los rangos de entrada.

LVQ con MATLAB

NEWLVQ crea una red LVQ.

Sintáxis

```
net = newlvq(PR,S1,PC,LR,LF)
```

Descripción

LVQ se usa para resolver problemas de clasificación.

Entradas:

PR – matriz de Rx2 de valores mínimos y máximos de los R elementos de entrada. En este caso hay dos elementos de entrada, por lo que la matriz es de 2x2.

S1 – Número de neuronas ocultas. En este trabajo se usaron 4

PC - Vector con el porcentaje de cada clase.

LR – Velocidad de aprendizaje, por defecto = 0.01.

LF – Función de aprendizaje, por defecto = 'learnlv2'.

Propiedades

NEWLVQ crea una red de dos capas. La primera utiliza la función de transferencia COMPET, calcula las entradas pesadas con NEGDIST, y las entradas de la red con NETSUM. La segunda capa tiene neuronas PURELIN, calcula las entradas pesadas con DOTPROD y las entrada de la red con NETSUM.

Ninguna entrada tiene umbrales.

Los pesos de la primera capa se inicializan con MIDPOINT, pero en este caso se inicializaron en un punto centrado aproximadamente en los datos de entrada. Los pesos de la segunda capa se fijan de modo que cada neurona de salida i tiene peso unitario desde PC(i) porcentaje de neuronas ocultas.

- *learnlv2*: función de aprendizaje de los pesos. Calcula el cambio en los pesos, dW, para una neurona dada a partir de la entrada de la neurona P, la salida A, el gradiente de salida gA y la velocidad de aprendizaje LR, de acuerdo a la regla LVQ2, dado I, el índice de la neurona cuya salida es 1:

  dw(i,:) = +lr * (p-w(i,:)) si gA(i) = 0
  = -lr * (p-w(i,:)) si gA(i) = -1
- *purelin*: función de transferencia lineal.
  Algoritmo:
  Purelin (n) = n
- *dotprod*: función de pesos de producto punto a punto.

6- BASE DE DATOS DE ARRITMIAS DEL MIT-BIH

A continuación se describe cómo se obtuvieron los registros de la base de datos y se discuten las características de las señales utilizadas en el entrenamiento y prueba de la red neuronal.

6.1 Criterios de selección

La fuente de los ECG incluidos en la base de datos de arritmias del MIT-BIH es un conjunto de más de 4000 registros de Holter obtenidos por el Laboratorio de Arritmias del Beth Israel Hospital entre 1975 y 1979. La base de datos contiene 48 registros, 23 de los cuales están seleccionados al azar (numerados desde el 100 al 124, con algunos números faltantes) y los otros 25 fueron seleccionados para incluir una variedad de fenómenos clínicos raros pero importantes clínicamente. Cada registro tiene un poco más de 30 minutos de duración.

El primero de los dos grupos mencionados pretende ser una muestra representativa de la variedad de formas de onda y artefactos que un detector de arritmias puede encontrar en el uso clínico de rutina.

Los registros del segundo grupo se eligieron para incluir arritmias ventriculares, de conducción o auriculares complejas. Muchos de estos registros se eligieron por ciertas particularidades del ritmo, variación en la morfología del QRS, etc.

Los sujetos fueron 25 hombres entre 32 y 89 años, y 22 mujeres entre 23 y 89 años.

6.2 Configuración de derivaciones

En la mayoría de los registros, la señal superior es la derivación bipolar DII modificada (MLII), obtenida colocando los electrodos en el pecho. La señal inferior es usualmente V1 (ocasionalmente V2 o V5, y en una ocasión V4); como para la señal inferior, los electrodos también se colocan en el pecho. Esta configuración se usa rutinariamente en el Laboratorio de Arritmias del BIH.

Los complejos QRS normales son usualmente prominentes en la primera señal. El eje de la derivación inferior puede ser casi ortogonal al eje eléctrico cardiaco medio. Es por eso que los latidos normales son difíciles de diferenciar en esta señal, aunque los latidos ectópicos serán a menudo prominentes (como en el registro 106). En los registros 102 y 104 no fue posible usar la derivación MLII debido a un vendaje quirúrgico de los pacientes; se utilizó la derivación V5 para la señal superior de este registro.

6.3 Registro analógico

Los registros analógicos originales se realizaron utilizando nueve grabadores Del Mar Avionics modelo 445 de dos canales, designados de A a I:

Grabador	***Registro***
A	102, 107, 111, 115, 121
B	212
C	203
D	118, 124, 217
E	101, 103, 106, 108, 112, 117, 119, 122, 209, 219, 220, 223, 233
F	104, 109, 123, 205, 207, 210, 215, 221
G	100, 105, 114, 116, 213, 214, 222, 228
H	113, 201, 202, 231
I	200, 230, 232, 234

(No se sabe qué grabador se utilizó para el registro 208)

La diferencia entre las dos señales puede ser tan grande como 40 ms para algunos de los registros analógicos. Además de este desfasaje fijo que viene de pequeñas diferencias en la orientación de los cabezales de las cintas durante la grabación y su reproducción para digitalizar los registros, aparece un desfasaje variable que proviene de oscilaciones verticales microscópicas de la cinta y que puede ser comparable al fijo. Este problema puede presentar dificultades para ciertos métodos de análisis de dos canales diseñados para aplicaciones en tiempo real.

Las variaciones menores de velocidad de la cinta no deberían ser un problema para los detectores de arritmias típicos. Es difícil evitar deslizamientos o adherencias de la cinta durante una reproducción a baja velocidad, y muchos deslizamientos fueron detectados y marcados con comentarios.

Un número de artefactos en el dominio de la frecuencia ha sido identificado y relacionado con componentes mecánicos específicos de las unidades grabadoras y reproductoras:

Frecuencia (Hz)	***Fuente***
0.042	Rueda de presión del grabador
0.083	Capstan de la unidad de reproducción (para reproducción al doble de velocidad de tiempo real)
0.090	Capstan del grabador
0.167	Capstan de la unidad de reproducción (para reproducción a velocidad normal)
0.18-0.10	Bobina de extracción (la frecuencia decrece con el tiempo)
0.20-0.36	Bobina de suministro (la frecuencia se incrementa con el tiempo)

La frecuencia de los últimos dos artefactos depende de cuánta cinta hay en cada bobina; la bobina de suministro ocasiona un artefacto mucho más notable que la de

extracción. Otros artefactos provocados por la bobina de suministro aparecen en las bandas de 0.10-0.18 Hz y 0.30-0.54 Hz.

6.4 Digitalización

Las salidas analógicas de la unidad reproductora fueron filtradas para limitar la saturación del conversor A/D y para evitar el aliasing, usando un pasabanda de 0.1 a 100 Hz, más allá de las frecuencias inferior y superior recuperables en los registros. La señal filtrada se digitalizó a 360 Hz por señal utilizando hardware construido en el Centro de Ingeniería Biomédica del MIT y el Laboratorio de Ingeniería Biomédica del BIH. La frecuencia de muestreo fue elegida para facilitar las implementaciones de filtros notch de 60 Hz (frecuencia de línea en EEUU) en los detectores de arritmias. Dado que los grabadores estaban alimentados por baterías, la mayoría del ruido de línea en la base de datos aparece durante la reproducción. En aquellos registros que se digitalizaron al doble del tiempo real este ruido aparece en 30 Hz (y sus múltiplos).

Las muestra se adquirieron de cada señal casi simultáneamente (los saltos entre señal y señal son del orden de los microsegundos). Los conversores A/D utilizados son unipolares, con resolución de 11 bits sobre un rango de ±5 mV. Los valores muestreados oscilan entre 0 y 2047, correspondiendo al 0 el valor 1024.

Las muestras han sido reconstruidas y almacenadas en pares de amplitudes de 12 bits y empaquetadas de a tres bytes consecutivos.

6.5 Tabla

Símbolos usados

Símbolo	***Significado***
· *or* N	Latido normal
L	Latido con bloqueo de rama izquierda
R	Latido con bloqueo de rama derecha
A	Latido arterial prematuro
A	Latido arterial prematuro aberrante
J	Latido nodal prematuro
S	Latido supraventricular prematuro
V	Contracción ventricular prematura
F	Fusión de latido normal y ventricular
[	Comienzo de aleteo / fibrilación ventricular
!	Onda de aleteo ventricular
]	Fin de aleteo / fibrilación ventricular
E	Latido de escape arterial
J	Latido de escape nodal
E	Latido de escape ventricular
/	Latido de marcapasos
F	Fusión de latido normal y latido de marcapasos
X	Onda P no conducida
Q	Latido inclasificable
|	Artefactos aislados con “forma” de QRS

Tipos de latidos

Registro	.	L	R	A	a	J	S	V	F	O	N	E	P	F	O	Q
	.	L	R	A	a	J	S	V	F	!	e	j	E	P	f	p	Q
100	2239	-	-	33	-	-	-	1	-	-	-	-	-	-	-	-	-
101	1860	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-	2
102	99	-	-	-	-	-	-	4	-	-	-	-	2028	56	-	-	
103	2082	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	
104	163	-	-	-	-	-	-	2	-	-	-	-	1380	666	-	18	
105	2526	-	-	-	-	-	-	41	-	-	-	-	-	-	-	5	
106	1507	-	-	-	-	-	-	520	-	-	-	-	-	-	-	-	
107	-	-	-	-	-	-	-	59	-	-	-	-	2078	-	-	-	
108	1740	-	-	4	-	-	-	16	2	-	-	1	-	-	11	-	
109	-	2492	-	-	-	-	-	38	2	-	-	-	-	-	-	-	
111	-	2123	-	-	-	-	-	1	-	-	-	-	-	-	-	-	
112	2537	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	
113	1789	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	
114	1820	-	-	10	-	2	-	43	4	-	-	-	-	-	-	-	
115	1953	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
116	2302	-	-	1	-	-	-	109	-	-	-	-	-	-	-	-	
117	1534	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	
118	-	-	2166	96	-	-	-	16	-	-	-	-	-	-	10	-	
119	1543	-	-	-	-	-	-	444	-	-	-	-	-	-	-	-	
121	1861	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	
122	2476	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
123	1515	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	
124	-	-	1531	2	-	29	-	47	5	-	-	5	-	-	-	-	
200	1743	-	-	30	-	-	-	826	2	-	-	-	-	-	-	-	
201	1625	-	-	30	97	1	-	198	2	-	-	10	-	-	37	-	
202	2061	-	-	36	19	-	-	19	1	-	-	-	-	-	-	-	
203	2529	-	-	-	2	-	-	444	1	-	-	-	-	-	-	4	
205	2571	-	-	3	-	-	-	71	11	-	-	-	-	-	-	-	

207	-	1457	86	107	-	-	-	105	-	472	-	-	105	-	-	-	-
208	1586	-	-	-	-	-	2	992	373	-	-	-	-	-	-	-	2
209	2621	-	-	383	-	-	-	1	-	-	-	-	-	-	-	-	-
210	2423	-	-	-	22	-	-	194	10	-	-	-	1	-	-	-	-
212	923	-	1825	-	-	-	-	-	-	-	-	-	-	-	-	-	-
213	2641	-	-	25	3	-	-	220	362	-	-	-	-	-	-	-	-
214	-	2003	-	-	-	-	-	256	1	-	-	-	-	-	-	-	2
215	3196	-	-	2	-	-	-	164	1	-	-	-	-	-	-	-	-
217	244	-	-	-	-	-	-	162	-	-	-	-	-	1542	260	-	-
219	2082	-	-	7	-	-	-	64	1	-	-	-	-	-	-	133	-
220	1954	-	-	94	-	-	-	-	-	-	-	-	-	-	-	-	-
221	2031	-	-	-	-	-	-	396	-	-	-	-	-	-	-	-	-
222	2062	-	-	208	-	1	-	-	-	-	-	212	-	-	-	-	-
223	2029	-	-	72	1	-	-	473	14	-	16	-	-	-	-	-	-
228	1688	-	-	3	-	-	-	362	-	-	-	-	-	-	-	-	-
230	2255	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
231	314	-	1254	1	-	-	-	2	-	-	-	-	-	-	-	2	-
232	-	-	397	1382	-	-	-	-	-	-	-	1	-	-	-	-	-
233	2230	-	-	7	-	-	-	831	11	-	-	-	-	-	-	-	-
234	2700	-	-	-	-	50	-	3	-	-	-	-	-	-	-	-	-

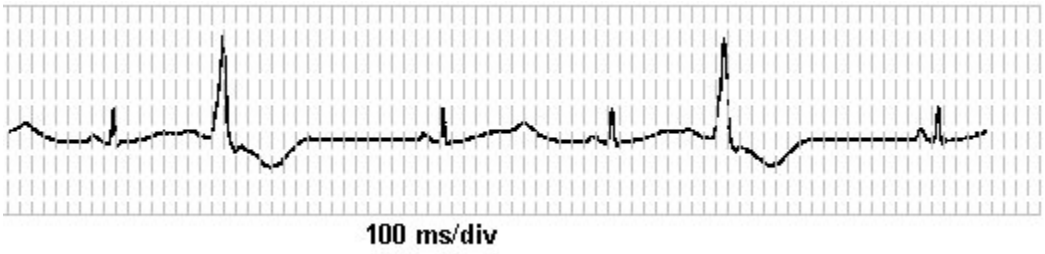
Las señales señaladas son la que se utilizaron para el entrenamiento y prueba de la red neuronal.

Las señales de la base de datos se seleccionaron de modo que tuvieran contracciónes ventriculares prematuras, que es la arritmia que pretende detectarse en este trabajo.

- CONTRACCIONES VENTRICULARES PREMATURAS (CVP):

Son despolarizaciones prematuras del ventrículo, por lo que se asocian a complejos QRS prematuros de una morfología habitualmente distinta y con una duración superior a 0,12 seg. La onda T generalmente es muy grande y de dirección contraria a la deflexión mayor del complejo QRS. Este complejo no va precedido de onda T prematura, aunque se

puede observar la onda P sinusal correspondiente a la actividad auricular. También puede aparecer una onda P retrógrada detrás del complejo QRS, oculta e invisible en el ECG.



APÉNDICE A
Software de Adquisición y Visualización

```
#include<iostream.h>
#include<graphics.h>
#include <stdlib.h>
#include<stdio.h>
#include<conio.h>
```

```
#include<bios.h>
#include<math.h>
#include<dos.h>
#include<string.h>
#include<fstream.h>
#include <time.h>
#include "qrsdet.h"

#define enter 13
#define ESC 27
#define space 32
#define P_Datos 0x378           // Dir. puerto de datos del puerto paralelo
#define P_Estado 0x379          // Dir. puerto de estados del puerto paralelo
#define P_Control 0x37A         // Dir. puerto de control puerto paralelo
#define F1 0x3B
#define F2 0x3C
#define F3 0x3D
#define F4 0x3E
#define F5 0x3F
#define F6 0x40
#define F7 0x41
#define PRE_BLANK MS200

// Prototipos locales.

int QRSFilter(int dato, int init) ;

int lpfilt( int dato ,int init) ;

int hpfilt( int dato, int init ) ;

int deriv1( int x0, int init ) ;

int deriv2( int x0, int init ) ;

int mvwint(int dato, int init) ;

int Peak( int dato, int init ) ;

int median(int *array, int datnum) ;

int thresh(int qmedian, int nmedian) ;

int BLSCheck(int *dBuf,int dbPtr,int *maxder) ;

int earlyThresh(int qmedian, int nmedian) ;


/*****************************************************************************************************/
double TH = 0.475  ;

int DDBuffer[DER_DELAY], DDPtr ;
int Dly  = 0 ;

const int MEMMOVELEN = 7*sizeof(int);
```

```
/*****************************************************************************************************/
struct datos
{
   char nombre[60];
   int edad, hc;
   float peso;
 };

/*****************************************************************************************************/
/* retardo() bucle usado como retardo
/*****************************************************************************************************/
retardo(int j)
{
 int i,k;
 for(i=0;i<j;i++)
          {
          k++;
          }
 return 0;
};

/*****************************************************************************************************/
/* TeclaPresionada() lee constantemente
 el teclado y sólo se interrumpe cuando se pulsa ENTER
/*****************************************************************************************************/
void TeclaPresionada(void)
{
 int tecla=0;
 int salir=0;
 aux:
 while(!salir)
     {
      if (!kbhit()) goto aux;
      else
            {
             tecla=bioskey(0);
             if ((tecla&255)==0) tecla=tecla>>8;
             else tecla=tecla&255;
             if (tecla==enter) salir=1;
            }
     }
};

/*****************************************************************************************************/
/* fondo() genera el marco perimetral de la pantalla
/*****************************************************************************************************/
void fondo()
{
 textattr(RED|BLACK*16);
 clrscr();
 for (int i=2;i < 80;i++)
```

```
    {
      gotoxy(i,1); cprintf("Í");
      gotoxy(i,24);cprintf("Í");
    }

 for (i=2; i < 24;i++)
    {
      gotoxy(1,i); cprintf("º");
      gotoxy(80,i);cprintf("º");
    }

 gotoxy(1,1); cprintf("É");
 gotoxy(1,24); cprintf("È");
 gotoxy(80,1); cprintf("»");
 gotoxy(80,24);cprintf("¼");
 textattr(YELLOW|BLACK*5);
}

/*****************************************************************************************************/
/* presentacion() Muestra datos referentes al proyecto
/*****************************************************************************************************/
void presentacion()
{
 fondo();
 gotoxy(31,3);cprintf(" PRUEBA DE ESFUERZO ");
 textattr(YELLOW|BLACK*16);
 gotoxy(3,7);cprintf("Realizado por: ");
 textattr(WHITE|BLACK*16);
 gotoxy(18,8);cprintf("Messineo, María Gabriela");
 gotoxy(18,9);cprintf("Gonzalez, Raúl Alejandro");
 textattr(YELLOW|BLACK*16);
 gotoxy(3,11); cprintf("Tutores: ");
 textattr(WHITE|BLACK*16);
 gotoxy(12,12); cprintf("Ing. Fernando Clara");
 gotoxy(12,13); cprintf("Ing. Isabel Passoni");
 gotoxy(3,15);cprintf("Universidad Nacional de Mar del Plata");
 gotoxy(3,17);cprintf("Facultad de Ingeniería");
 gotoxy(3,19);cprintf("Departamento de Bioingeniería");
 textattr(YELLOW|BLACK*16);
 gotoxy(3,23);cprintf("[ENTER] para continuar");
}


/*****************************************************************************************************/
/* Panel_De_Acceso() Muestra el uso del teclado
/*****************************************************************************************************/
Panel_De_Acceso()
{
 int opcion;

 fondo();
```

```
 gotoxy(30,3);cprintf(" PANEL DE ACCESO ");

 textattr(WHITE|BLACK*16);
 gotoxy(12,8);cprintf("F1 Nuevo paciente");
 gotoxy(12,10);cprintf("F2 Electro de un paciente ya ingresado");
 gotoxy(12,12);cprintf("F3 Analisis del ECG con redes neuronales");
 gotoxy(12,14);cprintf("F4 Salir");
 textattr(YELLOW|BLACK*1);
 return 0;
}

/*****************************************************************************************************/
/* IngresoDeDatos() Muestra una serie de datos referidos al paciente
/*****************************************************************************************************/
int IngresoDeDatos(void)
{
 int i, tecla=0;
 struct datos s;
 char Rta,HC[15],nombre[60],nom_archivo[60],aux[5]=".dat";
 FILE *archivo;

 fondo();
 gotoxy(30,3);cprintf(" DATOS DEL PACIENTE ");
 textattr(WHITE|BLACK*16);

 gotoxy(3,7);cprintf("Nombre y Apellido: ");
 gotoxy(3,10);cprintf("Edad: ");
 gotoxy(3,13);cprintf("Peso: ");
 gotoxy(3,16);cprintf("Historia clínica: ");

 textattr(WHITE|BLACK*16);
 gotoxy(24,7);gets(s.nombre);
 gotoxy(11,10);cin>>s.edad;
 gotoxy(11,13);cin>>s.peso;
 gotoxy(23,16);cin>>s.hc;

 textattr(YELLOW|BLACK*10);
 gotoxy(3,22);cprintf("Con [ENTER] comienza la adquisición.");
 gotoxy(3,23);cprintf("Con [ESC.] retorna al Panel de Acceso.");
 textattr(WHITE|BLACK*16);
 gotoxy(50,23);

 tecla=bioskey(0);
 if ((tecla&255)==0) tecla=tecla>>8;
 else tecla=tecla&255;
 if (tecla==ESC)
   {
     return(0);
   }

 archivo=fopen("Mis_Pacientes.dat","ab");
```

```
 fprintf(archivo, "%d \n",s.hc);
 fclose(archivo);

 itoa(s.hc,nom_archivo,10);
 strcat(nom_archivo, aux);
 archivo=fopen(nom_archivo, "wb");
 fwrite(&s, sizeof(s), 1, archivo);
 fclose(archivo);

 return(s.hc);
}

/*****************************************************************************************************/
/* Adquisicion() es la encargada de manejar las lineas y de leer los datos del puerto paralelo.
/*****************************************************************************************************/
int Adquisicion(int j)
{
 int k, i, INTR, INT, Nibble, NMS, NmS, dato;
 clock_t start, end;

 i=2*j;

 outportb(P_Datos,i);                    //Direcciona la entrada al conversor y maneja
                                         //el multiplexor de entrada al puerto paralelo

 INT=0x08;
 while(INT==0x08)                        //Lee el estado de la línea INTR del conversor,
     {                                   //cuando está en bajo sale del bucle.
      INTR=(inportb(P_Estado));          //Esto significa que comenzó la conversión.
      //retardo(1000);
      INT=INTR&0x08;
     }

 while(INT==0x00)                        //Lee el estado de la linea INTR del conversor,
     {                                   //cuando está en alto sale del bucle.
      INTR=(inportb(P_Estado));          //Esto significa que puedo leer el dato del bus.
      //retardo(1000);
      INT=INTR&0x08;
     }
 //retardo(100);
 outportb(P_Control,0);                  //Coloca en alto la línea OE del conversor

 retardo(655);

 Nibble=inportb(P_Estado);               //Nibble toma el valor que hay en el puerto

 NmS=(((Nibble&0x70) | (~Nibble&0x80))>>4)&0x0f;      //Obtiene el nibble menos
                                                     // significativo

 i=i+1;
```

```
 //retardo(200);

 outportb(P_Datos,i);
 retardo(250);

 Nibble=inportb(P_Estado);                    //Nibble toma el valor que hay en el puerto

 NMS=((Nibble&0x70) | (~Nibble&0x80))&0xf0;          //Obtiene el nibble más
                                                     //significativo
 //retardo(500);

 outportb(P_Control,1);                       //Coloca en bajo el OE del conversor.

 return (NMS|NmS);
}

/*****************************************************************************************************/
/* Visualización() Muestra la señal (con V=1 la señal que se está adquiriendo, con V=0
una señal ya guardada)
/*****************************************************************************************************/
Visualización(int HC,int V)
{
 int gdriver = DETECT, gmode, errorcode, j, i, Rdelay, cont, PPM;
 float q, x, y, z, t, t2, marca;
 int dato, dato0, dato1, dato2, dato3;
 int entrada, entrada0, entrada1, entrada2, entrada3;
 int tecla=0, salir=0, R, delay1, delay0, punto, habilitador=0;
 char pattern[8]={0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff};
 char Nom_Arch[20], Extension[5]=".sgn";
 FILE *archivo;

 clrscr();
 initgraph( &gdriver, &gmode, "c:\\borlandc\\BGI ");
 errorcode = graphresult();

 if(errorcode != grOk)
   {
    printf("graphic error: %s\n", grapherrormsg(errorcode));
    printf("Presione cualquier tecla para parar:");
    getch();
    exit(1);
   }

 clrscr();
 cleardevice();

 if(V==1)
   {
    outtextxy(250,3,"PRUEBA DE ESFUERZO");
    outtextxy(3,452,"EXPL = Exploracion                [F6-F7] Variar Tiempo de Pantalla    ");
    outtextxy(3,465,"[ESC] Finalizar    [R] Rever ECG    [F5] Retornar a Pantalla Original  ");
```

```
  }
else
   {
    outtextxy(250,3,"PRUEBA DE ESFUERZO");
    outtextxy(3,452,"[BARRA] Detener ECG   [ENTER] Reanudar
                     [F6-F7] Variar Tiempo de Pantalla");
    outtextxy(3,465,"[ESC] Finalizar   [R] Rever ECG    [F5] Retornar a Pantalla Original");
   }

setcolor(RED);
setfillpattern(pattern,0);
rectangle(30,16,629,122);
rectangle(30,122,629,228);
rectangle(30,228,629,334);
rectangle(30,334,629,440);
setcolor(WHITE);
outtextxy(0,16,"ZOOM");
outtextxy(0,60,"F1");
outtextxy(0,69,"DII");
setcolor(DARKGRAY);
line(30,69,629,69);
setcolor(WHITE);
outtextxy(0,166,"F2");
outtextxy(0,175,"V2");
setcolor(DARKGRAY);
line(30,175,629,175);
setcolor(WHITE);
outtextxy(0,272,"F3");
outtextxy(0,281,"V5");
setcolor(DARKGRAY);
line(30,281,629,281);
setcolor(WHITE);
outtextxy(0,378,"F4");
outtextxy(0,387,"EXPL");
setcolor(DARKGRAY);
line(30,387,629,387);
setcolor(WHITE);

q=x=y=z=dato0=dato1=dato2=dato3=0;

itoa(HC,Nom_Arch,10);
strcat(Nom_Arch, Extension);

if(V==1)
  {
   outportb(P_Datos,0);     //Lleva al puerto de datos a un estado conocido
   outportb(P_Control,0);   //Lleva al puerto de control a un estado conocido
   archivo = fopen(Nom_Arch, "wb");
  }
else
   {
```

```
    archivo = fopen(Nom_Arch, "rb");
   }

t=0.8;
t2=2;
aux:
while(!salir)
    {
     for(j=0;j<4;j++)
          {
           if(V==1)
             {
              entrada=Adquisicion(j);
              fprintf(archivo, "%d ",entrada);
             }
           else
              {
                  fscanf(archivo,"%d",&entrada);
                  delay(t2);
                  if (feof(archivo)!=0)
                    {
                     fclose(archivo);
                     salir=1;
                     goto aux;
                    }
              }
           dato=((entrada-127)*106/255);

           if (dato>53) dato=53;
           if (dato<-53) dato=-53;

           if(j==0)
             {
              moveto(q+30,69-dato0);
              lineto(q+31,69-dato);
              dato0=dato;
              q=q+t;
              lineto(q+30,69-dato0);
              if(q>599)
                   {
                    q=0;
                    setfillpattern(pattern,0);
                    setcolor(RED);
                    bar(30,16,629,122);
                    rectangle(30,16,629,122);
                    setcolor(DARKGRAY);
                    line(30,69,629,69);
                    setcolor(WHITE);
                    moveto(30,69);
                   }
             }
```

```
if(j==1)
  {
   moveto(x+30,175-dato1);
   lineto(x+31,175-dato);
   dato1=dato;
   x=x+t;
   lineto(x+30,175-dato1);
   if(x>599)
        {
         x=0;
         setfillpattern(pattern,0);
         setcolor(RED);
         bar(30,122,629,228);
         rectangle(30,122,629,228);
         setcolor(DARKGRAY);
         line(30,175,629,175);
         setcolor(WHITE);
         moveto(30,175);
        }
  }

if(j==2)
  {
   moveto(y+30, 281-dato2);
   lineto(y+31, 281-dato);
   dato2=dato;
   y=y+t;
   lineto(y+30,281-dato2);
   if(y>599)
        {
         y=0;
         setfillpattern(pattern,0);
         setcolor(RED);
         bar(30,228,629,334);
         rectangle(30,228,629,334);
         setcolor(DARKGRAY);
         line(30,281,629,281);
         setcolor(WHITE);
         moveto(30,281);
        }
  }

if(j==3)
  {
   moveto(z+30,387-dato3);
   lineto(z+31,387-dato);
   dato3=dato;
   z=z+t;
   lineto(z+30,387-dato3);
   if(z>599)
```

```
                {
                 z=0;
                 setfillpattern(pattern,0);
                 setcolor(RED);
                 bar(30,334,629,440);
                 rectangle(30,334,629,440);
                 setcolor(DARKGRAY);
                 line(30,387,629,387);
                 setcolor(WHITE);
                 moveto(30,387);
                }
            }
          }
if(kbhit())
     {
      tecla=bioskey(0);
      if((tecla&255)==0) tecla=tecla>>8;
      else tecla=tecla&255;

      // Chequea qué tecla se pulsó   //

      if (tecla==F1)          //Muestra sólo la entrada DII
        {
         setcolor(BLUE);
         outtextxy(0,175,"V2");
         outtextxy(0,281,"V5");
         outtextxy(0,387,"EXPL");
         setcolor(WHITE);
         outtextxy(0,69,"DII");
         auxS1:
         while(!salir)
                {
                 for(j=0;j<4;j++)
                   {
                       if(V==1)
                         {
                          entrada=Adquisicion(j);
                          fprintf(archivo, "%d ",entrada);
                         }
                       else
                          {
                           fscanf(archivo,"%d",&entrada);
                           delay(t2);
                           if (feof(archivo)!=0)
                                 {
                                  fclose(archivo);
                                  salir=1;
                                  goto aux;
                                 }
                          }
```

```
        entrada=entrada-127;
        switch(j)
                {
                 case 0:
                        moveto(q+30,228-(1.65*entrada0));
                        lineto(q+31,228-(1.65*entrada));
                        entrada0=entrada;
                        q=q+t;
                        if(q>599)
                          {
                           q=0;
                           setfillpattern(pattern,0);
                           setcolor(RED);
                           bar(30,16,629,440);
                           rectangle(30,16,629,440);
                           setcolor(WHITE);
                           line(30,228,629,228);
                           moveto(30,228);
                           outtextxy(0,69,"DII");
                           setcolor(WHITE);
                          }
                 break;

                 case 1:
                        //no hace nada
                 break;

                 case 2:
                        //no hace nada
                 break;

                 case 3:
                        //no hace nada
                 break;
                }
   }

if (!kbhit()) goto auxS1;
else
       {
        tecla=bioskey(0);
        if ((tecla&255)==0) tecla=tecla>>8;
        else tecla=tecla&255;
        if (tecla==F5)     //Vuelve a la pantalla original
          {
           salir=0;
           setcolor(WHITE);
           outtextxy(0,69,"DII");
           outtextxy(0,175,"V2");
           outtextxy(0,281,"V5");
           outtextxy(0,387,"EXPL");
```

```
    x=y=z=q;
    goto aux;
   }
 if (tecla==F2)     //Muestra sólo V2
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,69,"DII");
    outtextxy(0,281,"V5");
    outtextxy(0,387,"EXPL");
    setcolor(WHITE);
    outtextxy(0,175,"V2");
    x=y=z=q;
    goto auxS2;
   }
 if (tecla==F3)     //Muestra sólo V5
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,69,"DII");
    outtextxy(0,175,"V2");
    outtextxy(0,387,"EXPL");
    setcolor(WHITE);
    outtextxy(0,281,"V5");
    x=y=z=q ;
    goto auxS3;
   }
 if (tecla==F4)     //Muestra sólo EXPL
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,69,"DII");
    outtextxy(0,175,"V2");
    outtextxy(0,281,"V5");
    setcolor(WHITE);
    outtextxy(0,387,"EXPL");
    x=y=z=q;
    goto auxS4;
   }
 if (tecla==F6)    //Cambia la constante de tiempo a 0.2
   {
    t=0.2;
   }
 if (tecla==F7)    //Cambia la constante de tiempo a 0.8
   {
    t=0.8;
   }
 if(tecla==space)
  {
   TeclaPresionada();
  }
```

```
                    if (tecla==ESC)    //Finaliza la adquisición
                      {
                       fclose(archivo);
                       salir=1;
                       goto aux;
                      }
                    }
            }
      }

if (tecla==F2)              //Muestra sólo la entrada V2
  {
   auxS2:
   while(!salir)
          {
           for(j=0;j<4;j++)
             {
                if(V==1)
                 {
                  entrada=Adquisicion(j);
                  fprintf(archivo, "%d ",entrada);
                 }
                else
                   {
                    fscanf(archivo,"%d",&entrada);
                    delay(t2);
                    if (feof(archivo)!=0)
                          {
                           fclose(archivo);
                           salir=1;
                           goto aux;
                          }
                   }


                entrada=entrada-127;
                switch(j)
                          {
                           case 1:
                                    moveto(q+30,228-(1.65*entrada1));
                                    lineto(q+31,228-(1.65*entrada));
                                    entrada1=entrada;
                                    q=q+t;
                                    if(q>599)
                                     {
                                      q=0;
                                      setfillpattern(pattern,0);
                                      setcolor(RED);
                                      bar(30,16,629,440);
                                      rectangle(30,16,629,440);
                                      setcolor(WHITE);
```

```
                        line(30,228,629,228);
                        moveto(30,228);
                        outtextxy(0,175,"V2");
                        setcolor(WHITE);
                       }
                break;

                case 0:
                        //no hace nada
                break;

                case 2:
                        //no hace nada
                break;

                case 3:
                        //no hace nada
                break;
               }
  }
if (!kbhit()) goto auxS2;
else
      {
       tecla=bioskey(0);
       if ((tecla&255)==0) tecla=tecla>>8;
       else tecla=tecla&255;
       if (tecla==F5)    //Vuelve a la pantalla original
         {
          salir=0;
          setcolor(WHITE);
          outtextxy(0,69,"DII");
          outtextxy(0,175,"V2");
          outtextxy(0,281,"V5");
          outtextxy(0,387,"EXPL");
          x=y=z=q;
          goto aux;
         }
       if (tecla==F1)       //Muestra sólo DII
         {
          salir=0;
          setcolor(BLUE);
          outtextxy(0,175,"V2");
          outtextxy(0,281,"V5");
          outtextxy(0,387,"EXPL");
          setcolor(WHITE);
          outtextxy(0,69,"DII");
          x=y=z=q;
          goto auxS1;
         }
       if (tecla==F3)      //Muestra sólo V5
         {
```

```
                    salir=0;
                    setcolor(BLUE);
                    outtextxy(0,69,"DII");
                    outtextxy(0,175,"V2");
                    outtextxy(0,387,"EXPL");
                    setcolor(WHITE);
                    outtextxy(0,281,"V5");
                    x=y=z=q;
                    goto auxS3;
                   }
                 if (tecla==F4)     //Muestra sólo EXPL
                   {
                    salir=0;
                    setcolor(BLUE);
                    outtextxy(0,69,"DII");
                    outtextxy(0,175,"V2");
                    outtextxy(0,281,"V5");
                    setcolor(WHITE);
                    outtextxy(0,387,"EXPL");
                    x=y=z=q;
                    goto auxS4;
                   }
                 if (tecla==F6)    //Cambia la constante de tiempo a 0.2
                   {
                    t=0.2;
                   }
                 if (tecla==F7)    //Cambia la constante de tiempo a 0.8
                   {
                    t=0.8;
                   }
                 if(tecla==space)
                   {
                    TeclaPresionada();
                   }
                 if (tecla==ESC)    //Finaliza la adquisición
                   {
                    fclose(archivo);
                    salir=1;
                    goto aux;
                   }
                }
          }
   }

if (tecla==F3)              // Muestra sólo la entrada V5
   {
    auxS3:
    while(!salir)
          {
           for(j=0;j<4;j++)
              {
```

```
if(V==1)
  {
   entrada=Adquisicion(j);
   fprintf(archivo, "%d ",entrada);
  }
else
    {
     fscanf(archivo,"%d",&entrada);
     delay(t2);
     if (feof(archivo)!=0)
           {
            fclose(archivo);
            salir=1;
            goto aux;
           }
    }

entrada=entrada-127;
switch(j)
           {
            case 2:
                     moveto(q+30,228-(1.65*entrada2));
                     lineto(q+31,228-(1.65*entrada));
                     entrada2=entrada;
                     q=q+t;
                     if(q>599)
                       {
                        q=0;
                        setfillpattern(pattern,0);
                        setcolor(RED);
                        bar(30,16,629,440);
                        rectangle(30,16,629,440);
                        setcolor(WHITE);
                        line(30,228,629,228);
                        moveto(30,228);
                        outtextxy(0,281,"V5");
                        setcolor(WHITE);
                       }
            break;

            case 1:
                     //no hace nada
            break;

            case 0:
                     //no hace nada
            break;

            case 3:
                     //no hace nada
            break;
```

```
                }
  }

if (!kbhit()) goto auxS3;
else
      {
       tecla=bioskey(0);
       if ((tecla&255)==0) tecla=tecla>>8;
       else tecla=tecla&255;
       if (tecla==F5)     //Vuelve a la pantalla original
         {
          salir=0;
          setcolor(WHITE);
          outtextxy(0,69,"DII");
          outtextxy(0,175,"V2");
          outtextxy(0,281,"V5");
          outtextxy(0,387,"EXPL");
          x=y=z=q;
          goto aux;
         }
       if (tecla==F1)     //Muestra sólo DII
         {
          salir=0;
          setcolor(BLUE);
          outtextxy(0,175,"V2");
          outtextxy(0,281,"V5");
          outtextxy(0,387,"EXPL");
          setcolor(WHITE);
          outtextxy(0,69,"DII");
          x=y=z=q;
          goto auxS1;
         }
       if (tecla==F2)     //Muestra sólo V2
         {
          salir=0;
          setcolor(BLUE);
          outtextxy(0,69,"DII");
          outtextxy(0,281,"V5");
          outtextxy(0,387,"EXPL");
          setcolor(WHITE);
          outtextxy(0,175,"V2");
          x=y=z=q;
          goto auxS2;
         }
       if (tecla==F4)     //Muestra sólo EXPL
         {
          salir=0;
          setcolor(BLUE);
          outtextxy(0,69,"DII");
          outtextxy(0,175,"V2");
          outtextxy(0,281,"V5");
```

```
                    setcolor(WHITE);
                    outtextxy(0,387,"EXPL");
                    x=y=z=q;
                    goto auxS4;
                   }
                 if (tecla==F6)    //Cambia la constante de tiempo a 0.2
                   {
                    t=0.2;
                   }
                 if (tecla==F7)    //Cambia la constante de tiempo a 0.8
                   {
                    t=0.8;
                   }
                 if(tecla==space)
                  {
                   TeclaPresionada();
                  }
                 if (tecla==ESC)    //Finaliza la adquisición
                   {
                    fclose(archivo);
                    salir=1;
                    goto aux;
                   }
                }
        }
  }

if (tecla==F4)            //Muestra sólo la entrada EXPL
  {
   auxS4:
   while(!salir)
         {
          for(j=0;j<4;j++)
             {
               if(V==1)
                 {
                  entrada=Adquisicion(j);
                  fprintf(archivo, "%d ",entrada);
                 }
               else
                  {
                   fscanf(archivo,"%d",&entrada);
                   delay(t2);
                   if (feof(archivo)!=0)
                         {
                          fclose(archivo);
                          salir=1;
                          goto aux;
                         }
                  }
```

```
        entrada=entrada-127;
        switch(j)
                {
                 case 3:
                        moveto(q+30,228-(1.65*entrada3));
                        lineto(q+31,228-(1.65*entrada));
                        entrada3=entrada;
                        q=q+t;
                        if(q>599)
                          {                                          q=0;
                           setfillpattern(pattern,0);
                           setcolor(RED);
                           bar(30,16,629,440);
                           rectangle(30,16,629,440);
                           setcolor(WHITE);
                           line(30,228,629,228);
                           moveto(30,228);
                           outtextxy(0,387,"EXPL");
                           setcolor(WHITE);
                          }
                        break;

                        case 1:
                                //no hace nada
                        break;

                        case 2:
                                //no hace nada
                        break;

                        case 0:
                                //no hace nada
                        break;
                       }
  }

if (!kbhit()) goto auxS4;
else
       {
        tecla=bioskey(0);
        if ((tecla&255)==0) tecla=tecla>>8;
        else tecla=tecla&255;
        if (tecla==F5)      //Vuelve a la pantalla original
          {
           salir=0;
           setcolor(WHITE);
           outtextxy(0,69,"DII");
           outtextxy(0,175,"V2");
           outtextxy(0,281,"V5");
           outtextxy(0,387,"EXPL");
           x=y=z=q;
```

```
    goto aux;
   }
 if (tecla==F1)     //Muestra sólo DII
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,175,"V2");
    outtextxy(0,281,"V5");
    outtextxy(0,387,"EXPL");
    setcolor(WHITE);
    outtextxy(0,69,"DII");
    x=y=z=q;
    goto auxS1;
   }
 if (tecla==F2)     //Muestra sólo V2
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,69,"DII");
    outtextxy(0,281,"V5");
    outtextxy(0,387,"EXPL");
    setcolor(WHITE);
    outtextxy(0,175,"V2");
    x=y=z=q;
    goto auxS2;
   }
 if (tecla==F3)     //Muestra sólo V5
   {
    salir=0;
    setcolor(BLUE);
    outtextxy(0,69,"DII");
    outtextxy(0,175,"V2");
    outtextxy(0,387,"EXPL");
    setcolor(WHITE);
    outtextxy(0,281,"V5");
    x=y=z=q;
    goto auxS3;
   }
 if (tecla==F6)    //Cambia la constante de tiempo a 0.2
   {
    t=0.2;
   }
 if (tecla==F7)    //Cambia la constante de tiempo a 0.8
   {
    t=0.8;
   }
 if(tecla==space)
  {
   TeclaPresionada();
  }
 if (tecla==ESC)    //Finaliza la adquisición
```

```
                                {
                                 fclose(archivo);
                                 salir=1;
                                 goto aux;
                                }
                              }
                          }
                    }

                  if (tecla==F6)
                    {
                     t=0.2;
                    }
                  if (tecla==F7)              {
                     t=0.8;
                    }
                  if(tecla==space)
                    {
                     TeclaPresionada();
                    }
                  if(tecla==ESC)
                    {
                     fclose(archivo);
                     salir=1;
                    }
                 }
     }
 return 0;
}

/*****************************************************************************************************/
/* BaseDeDato() Busca por número de historia clínica en el archivo Mis_Pacientes.dat si
se encuentra el paciente.
/*****************************************************************************************************/

int BaseDeDatos()
{
 char Rta, paciente[60];
 int hc, HC, aux, aux1, i;
 FILE *archivo;

 textattr(WHITE|BLACK*16);
 gotoxy(3,16); cprintf("Número de la Historia Clínica: ");
 textattr(WHITE|BLACK*16);
 gotoxy(34,16); cin>>HC;
 archivo=fopen("Mis_Pacientes.dat","rb");

 while (feof(archivo)==0)
      {
          fscanf(archivo,"%d \n",&hc);
          if(HC==hc)
```

```
                    {
                     goto salir;
                    }
      }
 textattr(WHITE|BLACK*16);
 gotoxy(3,18);cprintf("El Paciente no se encuentra en la base de datos.");
 gotoxy(3,20);cprintf("¨Desea Ingresarlo? [S]");
 textattr(WHITE|BLACK*16);
 hc=0;
 salir:

 fclose(archivo);
 return(hc);
}

/*****************************************************************************************************/
/* IngresoDeComentario() permite escribir un texto y guardarlo
/*****************************************************************************************************/

void IngresoDeComentario(int hc)
{
 char letra, nom_archivo[64], aux[5] = ".txt";
 int i,j, tecla;

 fondo();
 gotoxy(30,3);cprintf(" ANALISIS Y COMENTARIOS ");
 textattr(WHITE|BLACK*16);
 i=j=0;
 itoa(hc,nom_archivo,10);
 strcat(nom_archivo, aux);
 ofstream fichout(nom_archivo, ios::out);
 if(!fichout)  cout<< "\n Incapaz de Crear o Abrir el fichero ";
 else
    {
     gotoxy(3,5); cprintf("Comentario:");
     gotoxy(3,7);
     aux0:
     gotoxy(3,7+j);
     i=0;
     aux1:
     letra=getche();
     tecla=letra;
     if ((tecla&255)==0) tecla=tecla>>8;
     else tecla=tecla&255;
     if (tecla==enter) goto salir;
     fichout.put(tecla);

     i++;
     if(i>72)
              {
               j++;
```

```
                if(j>7) goto salir;
                goto aux0;
               }
      goto aux1;
     }
 salir:
 fichout.close();
};

/*****************************************************************************************************/
/* LecturaDelComentario() lee un archivo de texto y lo muestra en pantalla.
/*****************************************************************************************************/

void LecturaDelComentario(int hc)
{
 int i, j;
 char nom_archivo[60], letra, aux[5] = ".txt";

 textattr(WHITE|BLACK*16);
 i=j=0;
 itoa(hc,nom_archivo,10);
 strcat(nom_archivo, aux);
 ifstream fichin(nom_archivo,ios::in );
 if(!fichin) cout<< "\n Incapaz de Abrir el Archivo ";
 else
    {
     gotoxy(3,11); cprintf("Comentario:");
     gotoxy(3,13);
     aux0:
     gotoxy(3,13+j);
     i=0;
     aux1:
     fichin.get(letra);
     printf("%c",letra);
     i++;
     if(i>72)
               {
                j++;
                if(j>7) goto salir;
                goto aux0;
               }
     goto aux1;
    }
 salir:
 fichin.close();
};


/*****************************************************************************************************/
/* Lectura() muestra en pantalla los datos de un paciente ya ingresado.
/*****************************************************************************************************/
```

```
int Lectura()
{
 struct datos s;
 char nom_archivo[15],aux1[5]=".dat";
 int H_Clin;
 FILE *archivo;

 H_Clin=BaseDeDatos();
 if(H_Clin==0)
   {
    return(H_Clin);
   }

 clrscr();
 fondo();

 gotoxy(30,3);cprintf(" DATOS DEL PACIENTE ");

 itoa(H_Clin,nom_archivo,10);
 strcat(nom_archivo, aux1);
 if ((archivo = fopen(nom_archivo, "rb"))== NULL)
    {
     gotoxy(3,7);fprintf(stderr, "No se pudo abrir el archivo.\n");
     return 1;
    }

 fread(&s, sizeof(s), 1, archivo);
 fclose(archivo);
 textattr(WHITE|BLACK*16);
 gotoxy(3,6);cprintf("Nombre y Apellido ");
 gotoxy(3,7);cprintf("Edad: ");
 gotoxy(3,8);cprintf("Peso: ");
 gotoxy(3,9);cprintf("Historia clínica: ");
 gotoxy(23,6);cout<<s.nombre;
 gotoxy(11,7);cout<<s.edad;
 gotoxy(11,8);cout<<s.peso;
 gotoxy(23,9);cout<<s.hc;
 LecturaDelComentario(s.hc);
 textattr(YELLOW|BLACK*16);
 gotoxy(3,23);cprintf("Con [ENTER] observa el electro");

 TeclaPresionada();

 return(s.hc);
}

/*************************************************************************************************/
```

Redes_neuronales() hace un análisis del electrocardiograma utilizando redes neuronales artificiales, esta función llama a un ejecutable programado en Matlab.

```
/*************************************************************************************************/
```

```
Redes_neuronales(int HC)
{
 int i=0,j, estado, entrada, opcion, latidos, normales, pc;
 float NL,PC;
 char Nom_Arch[20], Extension[5]=".sgn";
 FILE *origen, *destino, *archivo;

 fondo();
 gotoxy(20,3);cprintf(" ANALISIS DEL ECG CON REDES NEURONALES ");
 textattr(WHITE|BLACK*16);
 itoa(HC,Nom_Arch,10);
 strcat(Nom_Arch, Extension);

 origen = fopen(Nom_Arch, "rb");
 destino = fopen("ecg.sgn", "wb");

 gotoxy(4,6);cprintf("Elija una derivación para su analisis");
 gotoxy(7,8);cprintf("1. DII ");
 gotoxy(7,9);cprintf("2. V2 ");
 gotoxy(7,10);cprintf("3. V5 ");
 gotoxy(7,11);cprintf("4. EXPL. ");
 arriba:
 gotoxy(4,13+i);cprintf("Derivación: ");
 gotoxy(16,13+i);opcion=getche();
 if(opcion!=49 && opcion!=50 && opcion!=51 && opcion!=52)
   {
    gotoxy(4,13);cprintf("La opción fue errónea");
    i=1;
    goto arriba;
   }
 j=0;
 while(feof(origen)==0)
     {
      j++;
           {
            entrada=getc(origen);
            switch(j)
                    {
                     case 1:
                             if(opcion==49) fprintf(destino," %d",entrada);
                     break;
                     case 2:
                             if(opcion==50) fprintf(destino," %d",entrada);
                     break;
                     case 3:
                             if(opcion==51) fprintf(destino," %d",entrada);
                     break;
                     case 4:
                             if(opcion==52) fprintf(destino," %d",entrada);
                             j=0;
```

```
                    break;
                   }
            }
      }
 fclose(origen);
 fclose(destino);

 estado = spawnl(P_WAIT,"RNA.exe",NULL);
 if (estado == -1)
   {
    gotoxy(10,10);perror("ERROR: ");
    getch();
    return 0;
   }
 archivo = fopen("salida.red", "rb");
 fscanf(archivo,"%f %f",&PC,&NL);
 fclose(archivo);

 normales=floor((1-PC/NL)*100);
 pc=ceil((PC/NL)*100);
 latidos=ceil(NL);
 gotoxy(4,16);cprintf("Resultado del análisis");
 gotoxy(7,18);cprintf("• Número total de latidos: %i ",latidos);
 gotoxy(7,19);cprintf("  ♦ Latidos Normales: %i %",normales);
 gotoxy(7,20);cprintf("  ♦ Contracciones Prematuras: %i %",pc);

 TeclaPresionada();
 return 0;
}
/*****************************************************************************************************/
/*                                  PROGRAMA PRINCIPAL
/*****************************************************************************************************/

main()
{
 char Rta1, rever, NombreDelArchivo[60];
 int hc,tecla;

 presentacion();
 TeclaPresionada();
 inicio:
 Panel_De_Acceso();
 aux:
 while(!kbhit());
 tecla=bioskey(0);
 if ((tecla&255)==0) tecla=tecla>>8;
 else tecla=tecla&255;


 if(tecla==F1)
   {
```

```
   ingreso_dato:
   hc=IngresoDeDatos();
   if (hc==0) goto inicio;
   Visualizacion(hc,1);
   repetir:
   rever=getch();
   if(rever=='r'|rever=='R')
     {
      closegraph();
          Visualizacion(hc,0);
      goto repetir;
     }
   closegraph();
   clrscr();
   IngresoDeComentario(hc);
   goto inicio;
  }
if(tecla==F2)
  {
   hc=Lectura();
   if(hc==0)
     {
      gotoxy(3,43);cin>>Rta1;
      if(Rta1=='s'| Rta1=='S')  goto ingreso_dato;
      else goto inicio;
     }
   else
         {
         Visualizacion(hc,0);
          ver:
          rever=getch();
          if(rever=='r'|rever=='R')
            {
              Visualizacion(hc,0);
              goto ver;
            }
          closegraph();
          clrscr();
          goto inicio;
         }
  }

if(tecla==F3)
  {
   hc=Lectura();
   if(hc==0)
     {
      gotoxy(3,43);cin>>Rta1;
      if(Rta1=='s'| Rta1=='S')  goto ingreso_dato;
      else goto inicio;
     }
```

```
    else
          {
           Redes_neuronales(hc);
           goto inicio;
          }
   }

 if(tecla==F4)
   {
    gotoxy(3,18);cprintf("¨¿Está seguro que desea abandonar el programa?[S/N]");
    gotoxy(54,18);cin>>Rta1;
    if(Rta1=='s'| Rta1=='S')
      {
       exit(0);
       goto salir;
      }
    else  goto inicio;
   }
 else goto aux;
 salir:
 cleardevice();
 return 0;
}
```

APÉNDICE B
Entrenamiento, Prueba y Utilización de la Red Neuronal

Como se explicó en las secciones 4 y 5, las redes neuronales se ocupan de la clasificación de las señales, pero es necesaria una previa extracción de las características de la señal para realizar el entrenamiento de la red con el objetivo que ésta adapte sus parámetros y realice una clasificación eficiente.

A continuación se muestra la forma en que se extrajeron dichas características utilizando MATLAB.

```
clear s s1 s2 s3 s4 s5 TCard TCardProm DTCard QRS P z z1 z2 z3 FC NS
salida abs;
clear T Tc salidalvq Yc Plvq posicion;
Fs=360;%frecuencia de muestreo de las señales utilizadas para el
entrenamiento (MIT-BIH)

fid = fopen('c:\gaby\senales\106.dat','r');
senal = fread(fid,'bit12');

s=senal;
num=max(size(s));
z(1:num/2)=senal(1:2:num);  % Como en los registros de la base de datos
hay dos señales multiplexadas, se demultiplexan.
%z(1:num/2)=senal(2:2:num); %En este caso se tomaría el registro inferior

num=floor((max(size(z)))/3);
z1(1:num)=z(1:num);                %Primer tramo de la señal, se usa para
probar la red completa una vez entrenada
z2(1:num)=z((num+1):(2*num));        %Segundo tramo de la señal, se usa
para entrenar la red SOM
z3(1:num)=z((2*num+1):(3*num));    %Tercer tramo de la señal, se usa para
entrenar la red LVQ

init=1;                      % con init=1 se entrena la red SOM con s=z2
                             % con init=2 se entrena la red LVQ con s=z3
                             % con init=0 se simula la red con z=z1
if init==1,
   s=z2;
end
if init==2
   s=z3;
end
if init==3,
   s=z1;
end
% También se utilizó para probar la red el registro 100, 105, 200 y 203
completos, con lo que z=z




% Preprocesamiento de la señal para la extracción de características

% Filtro pasabajos con frecuencia de corte en 20 Hz
[B,A]=butter(8,20/(Fs/2),'low');
s1=filter(B,A,s);

% Filtro pasaaltos con frecuencia de corte en 15 Hz
[D,C]=butter(8,15/(Fs/2),'high');
```

```
s2=filter(D,C,s1);

% Derivador
F=[0.2 0.1 0 -0.1 -0.2];
E=1;
s3=filter(F,E,s2);

% Integral móvil
N=round(0.08*Fs);     %ventana de 80 mseg, número de muestras:N=0.08*Fs
delay=50;
H=ones(1,N)/N;
I=1;
s4=s3.^2;
s5=filter(H,I,s4);


% Detección de parámetros útiles
% Detector de pico R
clear A B C D E F H I N s4 s1 s2 z num senal;
num=max(size(s5));
det=zeros(1,num);
Pico_R=zeros(1,num);
incr=round(0.016*Fs);
ventana=round(0.1*Fs);
muestras=round(0.16*Fs);
Mxs5=(max(s5(500:num)));
umbral=round(0.125*Mxs5);
umbral_min=round(umbral/2);
retardo=round((8+8+4+0.152*Fs)/2);     %retardo de los filtros
j=250;                                 %indice de la señal
k=0;                                   %cuenta hasta 80 picos hallados
l=0;                                   %indice de Tcard
m=0;                                   %indice del vector  que  guarda la
parte s5 que es mayor a umbral
I=0;
sum1=0;                                %se  usa  para  sumar 8  periodos
cardiacos
sum2=0;                                 %suma las alturas  de los picos de
la señal filtrada
cont=0;                                %cuenta muestras entre pico y pico
aux=0;
habilitador=0;
rebusqueda=0;
inicio=j;
RRprom=round(60/80*Fs);
vectorum=zeros(1,num);



while j<num,
   j=j+incr;
   if j>num-1,
      j=num;
   end
   cont=cont+incr;
   if s5(j)>umbral,
      rebusqueda=0;
```

```
      k=k+1;
      l=l+1;
      indice=j;
      while ((s5(j)>umbral) & (j<num)),
         j=j+1;
         m=m+1;
         vector(m)=s5(j);
         if m>muestras,
            aux=umbral;
            umbral=100000;
            habilitador=1;
         end
      end
      if habilitador==1,
         umbral=aux;
         habilitador=0;
      end
      aux=m;
      m=0;
      [Y,I]=max(vector);
      vector=zeros(1,muestras);
      det(indice+I)=Y;
      Tcard(l)=(cont+I)/Fs;
      if ((indice+I-retardo-ventana > 0) & (indice+I-retardo+ventana <
num))
       [R,IR]=max(abs(s((indice+I-retardo-ventana:indice+I-
retardo+ventana))));
       Pico_R(indice+I-retardo-ventana+IR)=s(indice+I-retardo-
ventana+IR);
      end
      posicion(l)=indice+I-retardo-ventana+IR;
      if l<9,
         sum1=sum1+(Tcard(l)*Fs);
      else
         sum1=sum1+((Tcard(l)-Tcard(l-8))*Fs);
         RRprom=round(sum1/8);
         RR(l)=RRprom;
      end
      if k<81,
         sum2=sum2+Y;
      else
         umbral_min_est=0.1*sum2/80;
         if umbral_min_est>0.25*umbral_min,
            umbral_min=round(0.75*umbral_min_est+0.25*umbral_min);
         end
         sum2=0;
         k=0;
      end
      umbral=round((umbral+Y/4)/2);
      if umbral<umbral_min,
         umbral=umbral_min;
      end
      saltos(l)=round(2*sqrt(RRprom));
      j=j+saltos(l);
      inicio=j;
      cont=saltos(l)+aux-I;
   else
```

```
      if (cont-round(2*sqrt(RRprom))-aux+I)>RRprom,
         if rebusqueda==0,
            j=inicio;
            umbral=umbral_min;
            rebusqueda=1;
            cont=round(2*sqrt(RRprom))+aux-I;
         end
      end
   end
end

%Cálculo del segmento QRS
num=max(size(Pico_R));
s3(1:28)=[];
i=0;
j=0;
l=0;
indice=0;
piso=0;
cont=0;
indiceS=0;
while j<num,
   j=j+1;
   indice=j;
   if (Pico_R(j)>piso & j+ventana<num),
      l=l+1;
      vectorS=s(j:j+ventana);
      [Ymn,Imn]=min(vectorS);
      indiceS(l)=indice+Imn;
      posicion(l)= indice;
      clear vectorS;
      if indice>3,
         indice=indice-3;
         while (s3(indice)>=piso & indice>3),     %lee la derivada de la
señal hacia atrás, mientras sea positiva
             indice=indice-1;                      %cuando es igual a 0 o
menor que 0, significa que se encontró el punto Q
         end
         indiceQ(l)=indice;
         QRS(l)=(indiceS(l)-indiceQ(l))/Fs;       %calcula el tiempo que
dura el QRS
         piso=0;
      else
         QRS(l)=0.04;
      end
   end

end

%Cálculo del periodo promedio y la diferencia entre periodos consecutivos

SUM=0;
num=max(size(Pico_R));
i=0;
j=0;
l=0;
indice=0;
```

```
piso=0;
divisor=0;
while j<num,
   j=j+1;
   i=i+1;
   if Pico_R(j)~=0,         %lee el vector Pico_R para
      indice=indice+1;
      TCard(indice)=i/Fs;   %calcula el periodo cardiaco y
      i=0;

      if indice>=11,         %calcula el periodo cardiaco promedio de las
últimas 10 pulsaciones
         divisor=10;
         SUM=SUM+TCard(indice)-TCard(indice-10);
      else
         divisor=indice;
         SUM=SUM+TCard(indice);
      end
      TCardProm(indice)=(SUM)/divisor; %calcula periodo promedio

      if indice>=2,                      %calcula la diferencia (en seg)
entre una pulsación y la anterior
         dTC=TCard(indice)-TCard(indice-1);
         DTCard(indice)=(dTC);
      else
         DTCard(indice)=0;
      end
   end
end
```

Una vez obtenidos los parámetros que se utilizan como entradas de la red: el período cardíaco, Tcard, y la diferencia entre períodos consecutivos, DTCard, se procede al entrenamiento de la red SOM.

```
%entrenmiento de una red neuronal SOM

num=max(size(TCard));
P(1,1:num)= TCard/1.5;
P(2,1:num)= DTCard/0.85;

if init==1,
   net=newsom([0 1;-1 1],3);           %crea la red "net"
   net.trainParam.show = 50;
   net.trainParam.lr = 0,05;
   net.trainParam.epochs = 1000;
   net.trainParam.goal = 1e-5;
   net=train(net,P);                   %entrena la red "net" con la
entrada "P"
   w=net.IW{1};
   figure(1);plot(P(1,:),P(2,:),'.k'); %muestra dónde caen los pesos
   hold on;  plot(w(:,1),w(:,2),'*c'); %respecto de las entradas
   title('Clusterización de la red SOM');
end
```

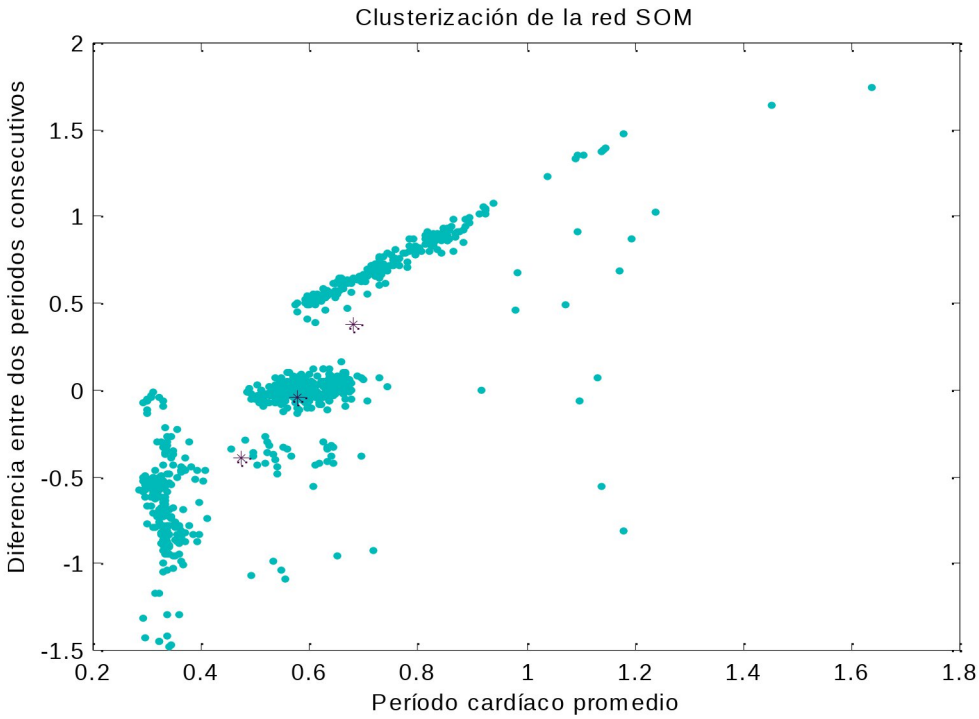


Figura 1

La figura muestra la clusterización que realiza la SOM dados los datos de entradas. Como la entrada tiene dos dimensiones puede apreciarse muy bien la ubicación de las mismas en un gráfico bidimensional, y debido a que la red tiene tres neuronas en su única capa tenemos tres puntos en el mapa, correspondientes cada uno a una neurona. Como se sabe, habrá una neurona ganadora para cada punto: la que esté topológicamente más cerca del mismo.

Luego de entrenada la red SOM, haciendo la variable *init* igual a 2, se procede a entrenar la LVQ. Para esto es necesario simular previamente la SOM ya que su salida es una de las entradas de la LVQ. La otra entrada es la duración del segmento QRS.

```
% Simulacion de la red SOM y entrenamiento de la LVQ
```

```
if init==2
   salida=sim(net,P);            %simula la red "net" con la entrada "P"
   num=max(size(P));
   for j=1:num,
      for i=1:3,
         if salida(i,j)==1 %la neurona ganadora fue la 3 y se le da el
valor 3 = LARGO
           salida(i,j)=i;   %la neurona ganadora fue la 2 y se le da el
valor 2 = NORMAL
           NS(j)= i;        %la neurona ganadora fue la 1 y se le da el
valor 1 = CORTO
         end
      end
   end

   FC=60./TCard;
   figure(2);plot(FC,'.b'); title('Clasificación de SOM');
   hold on;   plot(60*NS,'*r');
   axis([200 300 0 200]);

   figure(2),
   [X,Y]  =  GINPUT(1);          %Esta instrucción permite ver la señal
alrededor de un punto seleccionado entre los puntos clasificados
   X=round(X);
   if posicion(X)<=500
      point=501;
   else
      point=posicion(X);
   end
   if point-1000<=501,
      x1=501;
   else
      x1=point-1000;
   end
   if point+1000>max(size(s)),
      x2=max(size(s5));
   else
      x2=point+1000;
   end
   figure(3);
   t=[x1:1:x2]/Fs;
   t1=x1/Fs;
   t2=x2/Fs;
   plot(t,s(x1:x2),'k');
   hold on
   plot(t,Pico_R(x1:x2),'*r');
   title('Zoom 2seg a cada lado del latido seleccionado');
   axis([t1 t2 200 1800]);



   j=0;
   k=0;
   l=0;
   for i=1:num                % Se obtiene el vector Tc a partir de la
salida de la SOM
```

```
      if NS(i)==1           % Este es el vector clase objetivo que se le
da a LVQ
         k=k+1;
         Tc(i)=1;
      else
         l=l+1;
         Tc(i)=2;
      end
   end

   p1=k/num
   p2=l/num

   Plvq(1,1:num)= NS/3;
   Plvq(2,1:num)= QRS;
   T=ind2vec(Tc);

   net2=newlvq([0 1;0 1],4,[p1 p2]);
   net2.trainParam.epochs = 1000;
   net2.IW{1} =[0.65 0.1;0.65 0.1;0.65 0.1;0.65 0.1];

   wlvq=net2.IW{1};
   figure(4);plot(Plvq(1,:),Plvq(2,:),'.c');    %muestra  dónde  caen  los
pesos respecto de las entradas

   hold on;  plot(wlvq(:,1),wlvq(:,2),'*b');
   title('Clusterización de la red LVQ');
   xlabel('Salida de SOM');
   ylabel('QRS');
   pause;

   net2=train(net2,Plvq,T);                    %entrena la red "net2" con
la entrada "Plvq"

   wlvq=net2.IW{1};
   hold  on;  plot(Plvq(1,:),Plvq(2,:),'.c');     %muestra  dónde  caen  los
pesos respecto de las entradas
   hold on; plot(wlvq(:,1),wlvq(:,2),'*m');
   title('Clusterización de la red LVQ');
   xlabel('Salida de SOM');
   ylabel('QRS');

end
```

Las figuras correspondientes a la clasificación de la SOM, un ejemplo de lo que muestra *ginput*, y la clusterización de la LVQ se muestran a continuación. En la figura 4, que muestra la clusterización, puede observarse representada por una cruz el valor inicial de los pesos de la red y por cuatro estrellas el valor al que se trasladaron luego del entrenamiento de la misma.

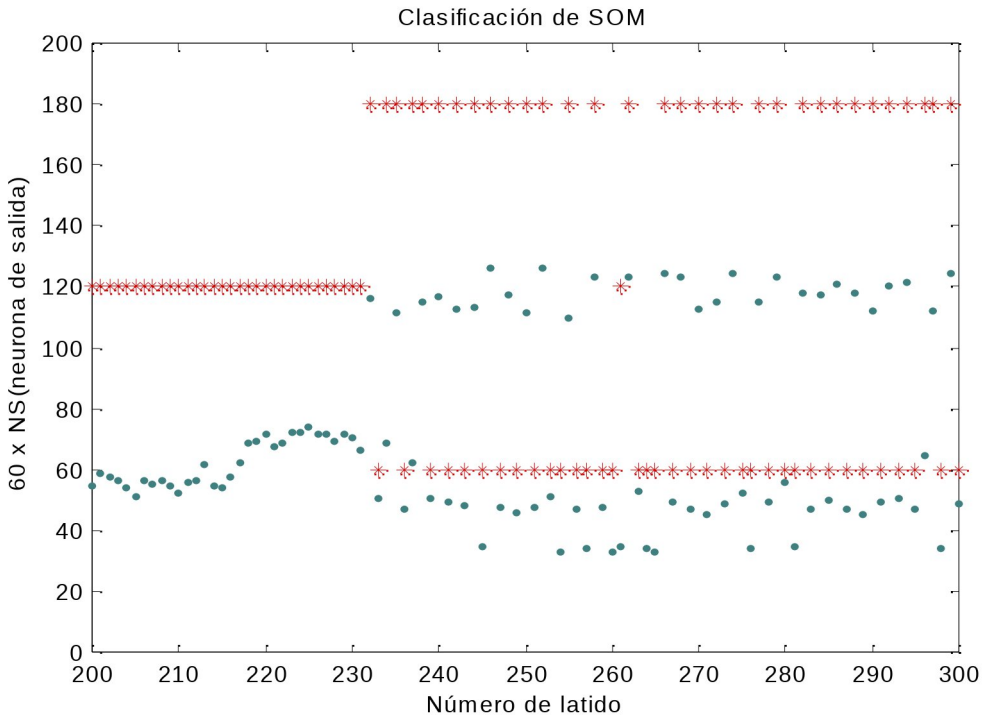


Figura 2

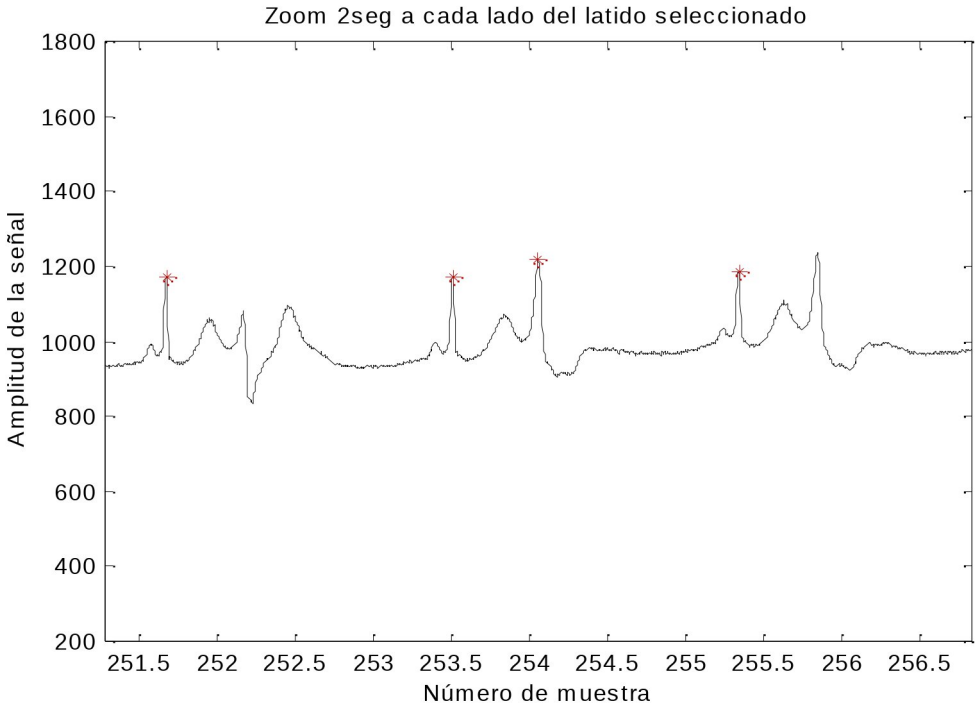


Figura 3

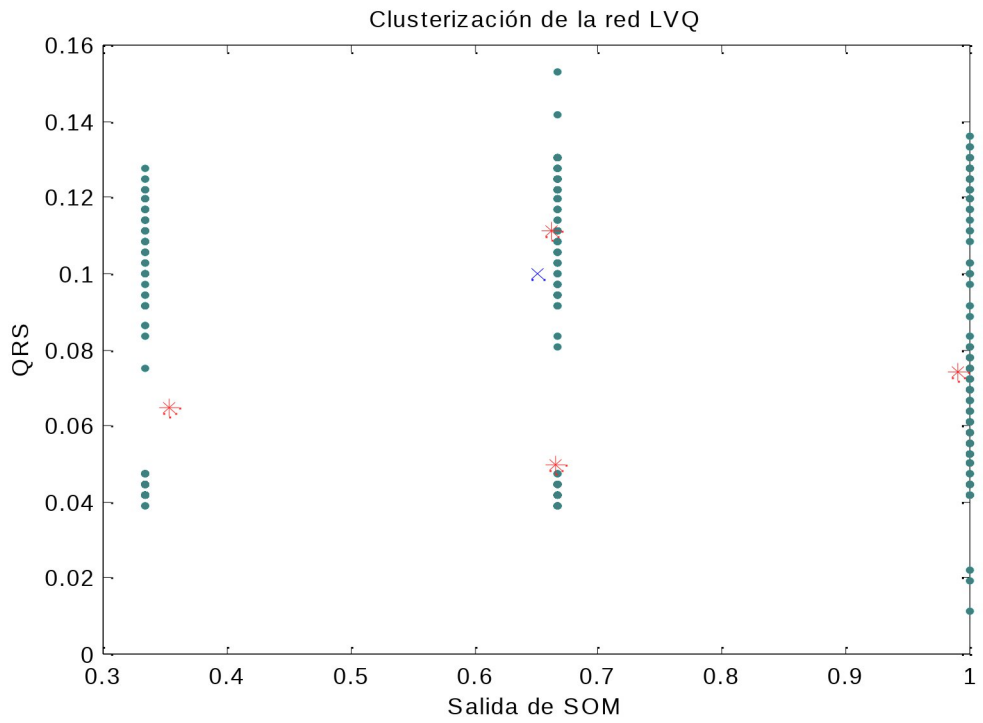


Figura 4

A continuación se realiza la prueba de la red completa, con la opción init=3, con la primera parte de la señal 106.dat. Puede apreciarse la clasificación de la red LVQ, la que simplifica la clasificación previa, distinguiendo entre latidos normales y anticipados.

```
% Prueba de la red completa
if init==3
   salida=sim(net,P);      %simula la red "net" con la entrada "P"

   num=max(size(P));
   for j=1:num,
      for i=1:3,
         if salida(i,j)==1 %la neurona ganadora fue la 1 y se le da el
valor 1 = CORTO
            salida(i,j)=i;  %la neurona ganadora fue la 2 y se le da el
valor 2 = NORMAL
            NS(j)= i;       %la neurona ganadora fue la 3 y se le da el
valor 3 = LARGO
         end
      end
   end
```

```
Plvq(1,1:num)= NS/3;
Plvq(2,1:num)= QRS;

salidalvq=sim(net2,Plvq);    %simula la red net2 con la entrada Plvq
Yc = vec2ind(salidalvq);

FC=60./TCard;
figure(5);plot(FC,'.b'); title('Clasificación de la red LVQ');
hold on;   plot(60*Yc,'*r');
```

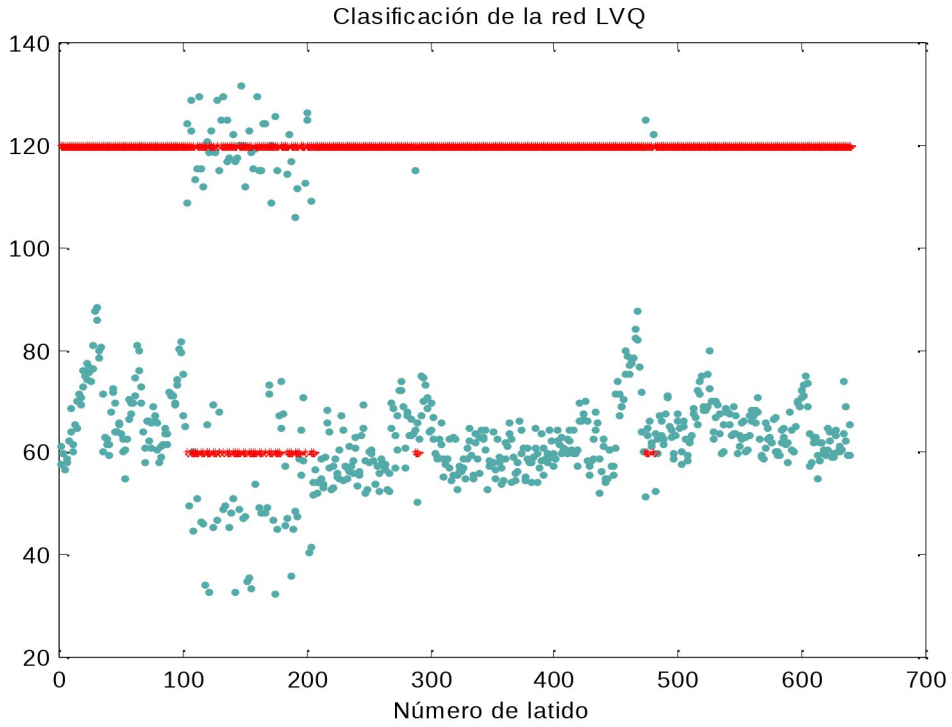


Figura 5

Aquí también se muestra el número de latidos vs. (neurona de salida) x 60. A los latidos normales les corresponde la neurona 2, es decir, son los que tienen un valor de 120. Los que tienen 60 como salida son latidos cortos.

Para terminar de probar la red vemos la figura que será la que aparezca en la pantalla del usuario del programa. Esta imagen es una especie de línea de tiempo en la

que se ve en color rojo las zonas de la señal en las que hay algún tipo de anomalía, y en color verde se ve la zona donde el registro es normal. Nuevamente, a través de la instrucción *ginput*, se puede ver la señal alrededor del punto de interés.

Las figuras 6 y 7 muestran un ejemplo habiéndose seleccionado un punto en la zona verde.

```
for i=2:num,
      if (Yc(i)-Yc(i-1))==0,
         out1(i)=1;
         out2(i)=0;
      else
         out1(i)=0;
         out2(i)=1;
      end
   end

   figure(6);
   plot(out1,'.g');
   hold on;
   plot(out2,'.r');
   title('Diagrama de Latidos (verde=NORMAL   rojo=PVC)');
   axis([1 num 0.96 1.04])
   figure(6),
   [X,Y] = GINPUT(1);
   X=round(X);
   if posicion(X)<=500
      point=501;
   else
      point=posicion(X);
   end
   if point-1000<=501,
      x1=501;
   else
      x1=point-1000;
   end
   if point+1000>max(size(s)),
      x2=max(size(s5));
   else
      x2=point+1000;
   end
   figure(7);
   t=[x1:1:x2]/Fs;
   t1=x1/Fs;
   t2=x2/Fs;
   plot(t,s(x1:x2),'k');
   hold on
   plot(t,Pico_R(x1:x2),'*r');
   title('Zoom 2seg a cada lado del latido seleccionado');
   axis([t1 t2 200 1800]);
end
```

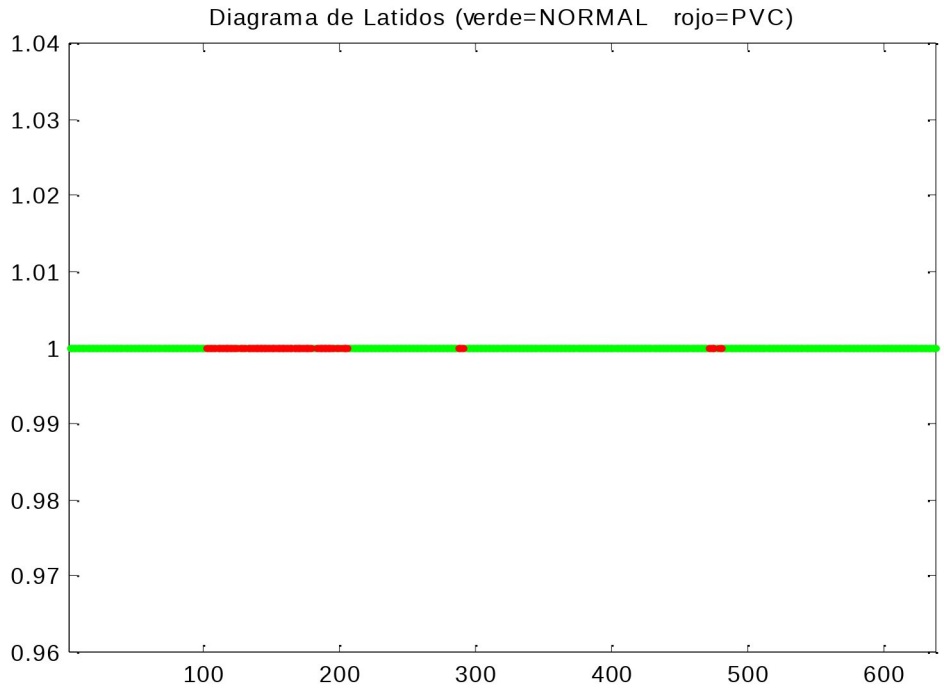


Figura 6

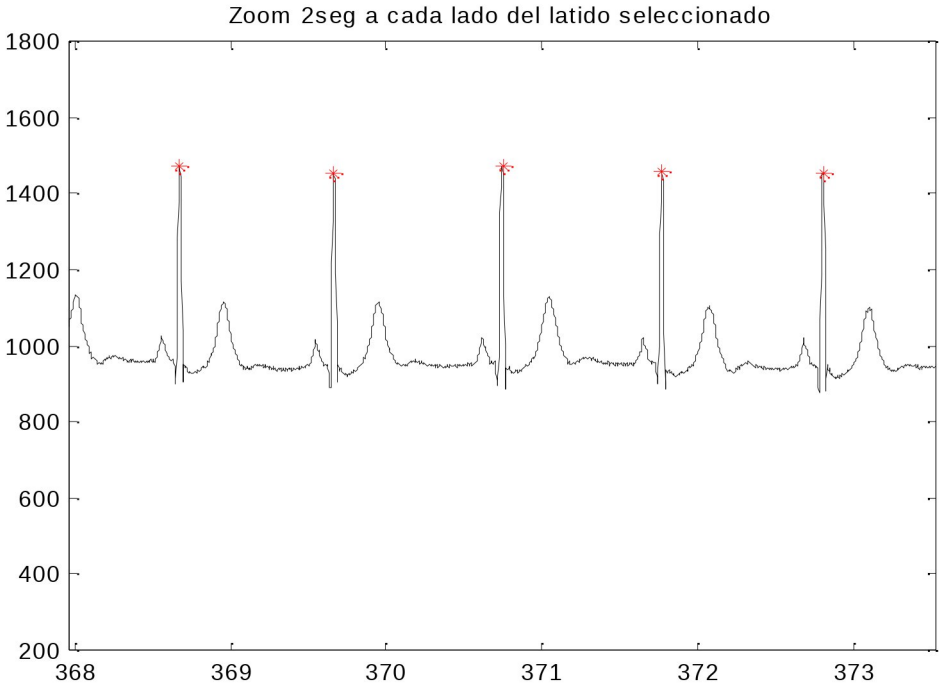


Figura 7

A continuación vemos la función que lleva a cabo el procedimiento anterior. Como puede verse la frecuencia de muestreo es la del instrumento desarrollado para el presente trabajo, 500 Hz, y las señales adquiridas se leen como archivos Ecg.sgn, que corresponde a una de las derivaciones previamente seleccionada para su análisis.

```
function RNA();

Fs=500;%frecuencia de muestreo

fid = fopen('c:\Ecgag\Ecg.sgn','r');
s = fscanf(fid,'%3d');
fclose(fid);


% Filtro pasabajos
[B,A]=butter(8,20/(Fs/2),'low');
s1=filter(B,A,s);

% Filto pasaaltos
[D,C]=butter(8,15/(Fs/2),'high');
s2=filter(D,C,s1);

% Derivada
F=[0.2 0.1 0 -0.1 -0.2];
E=1;
s3=filter(F,E,s2);

% Integral móvil
N=round(0.08*Fs);     %ventana de 80 mseg, N=0.08*Fs
delay=50;
H=ones(1,N)/N;
I=1;
s4=s3.^2;
s5=filter(H,I,s4);

% Detector de pico R

clear A B C D E F H I N s4 s1 s2 senal;
num=max(size(s5));
det=zeros(1,num);
Pico_R=zeros(1,num);
incr=round(0.016*Fs);
ventana=round(0.1*Fs);
muestras=round(0.16*Fs);
Mxs5=(max(s5(500:num)));
umbral=round(0.125*Mxs5);
umbral_min=round(umbral/2);
retardo=round((8+8+4+0.152*Fs)/2);     %retardo de los filtros
j=250;                                 %indice de la señal
k=0;                                   %cuenta hasta 80 picos que se
encontraron
l=0;                                   %indice de Tcard
m=0;                                   %indice del vector que guarda la
parte de s5 que es mayor a umbral
I=0;
```

```
sum1=0;                                 %se usa para sumar 8 periodos
cardiacos
sum2=0;                                 %suma las alturas de los picos de
la senal filtrada
cont=0;                                 %cuenta muestras entre pico y pico
aux=0;
habilitador=0;
rebusqueda=0;
inicio=j;
RRprom=round(60/80*Fs);
vectorum=zeros(1,num);

while j<num,
   j=j+incr;
   if j>num-1,
      j=num;
   end
   cont=cont+incr;
   if s5(j)>umbral,
      rebusqueda=0;
      k=k+1;
      l=l+1;
      indice=j;
      while ((s5(j)>umbral) & (j<num)),
         j=j+1;
         m=m+1;
         vector(m)=s5(j);
         if m>muestras,
            aux=umbral;
            umbral=100000;
            habilitador=1;
         end
      end
      if habilitador==1,
         umbral=aux;
         habilitador=0;
      end
      aux=m;
      m=0;
      [Y,I]=max(vector);
      vector=zeros(1,muestras);
      det(indice+I)=Y;
      Tcard(l)=(cont+I)/Fs;
      if ((indice+I-retardo-ventana > 0) & (indice+I-retardo+ventana <
num))
         [R,IR]=max(abs(s((indice+I-retardo-ventana:indice+I-
retardo+ventana))));
          Pico_R(indice+I-retardo-ventana+IR)= s(indice+I-retardo-
ventana+IR);
      end
      posicion(l)=indice+I-retardo-ventana+IR;
      if l<9,
         sum1=sum1+(Tcard(l)*Fs);
      else
         sum1=sum1+((Tcard(l)-Tcard(l-8))*Fs);
         RRprom=round(sum1/8);
         RR(l)=RRprom;
```

```
      end
      if k<81,
         sum2=sum2+Y;
      else
         umbral_min_est=0.05*sum2/80;
         if umbral_min_est>0.25*umbral_min,
            umbral_min=round(0.75*umbral_min_est+0.25*umbral_min);
         end
         sum2=0;
         k=0;
      end
      umbral=round((umbral+Y/4)/2);
      if umbral<umbral_min,
         umbral=umbral_min;
      end
      saltos(l)=round(0.39*sqrt(RRprom));
      j=j+saltos(l);
      inicio=j;
      cont=saltos(l)+aux-I;
   else
      if (cont-round(0.39*sqrt(RRprom))-aux+I)>RRprom,
         if rebusqueda==0,
            j=inicio;
            umbral=umbral_min;
            rebusqueda=1;
            cont=round(0.39*sqrt(RRprom))+aux-I;
         end
      end
   end
end

%Cálculo de los parámetros de la señal

%Cálculo del QRS
num=max(size(Pico_R));
s3(1:28)=[];
i=0;
j=0;
l=0;
indice=0;
piso=0;
cont=0;
indiceS=0;
while j<num,
   j=j+1;
   indice=j;
   if (Pico_R(j)>piso & j+ventana<num),
      l=l+1;
      vectorS=s(j:j+ventana);
      [Ymn,Imn]=min(vectorS);
      indiceS(l)=indice+Imn;
      posicion(l)= indice;
      clear vectorS;
      if indice>3,
         indice=indice-3;
         while (s3(indice)>=piso & indice>3),   %lee la derivada de la
senal hacia atras, mientras sea "+"
```

```
            indice=indice-1;       %cuando es igual a 0 o menor que 0, se
encontro el punto Q
         end
         indiceQ(l)=indice;
         QRS(l)=(indiceS(l)-indiceQ(l))/Fs;     %calcula el tiempo que
dura el QRS
         piso=0;
      else
         QRS(l)=0.04;
      end
   end

end

%Cálculo del periodo promedio y la diferencia entre periodos consecutivos
SUM=0;
num=max(size(Pico_R));
i=0;
j=0;
l=0;
indice=0;
piso=0;
divisor=0;
while j<num,
   j=j+1;
   i=i+1;
   if Pico_R(j)~=0,         %Lee el vector Pico_R para
      indice=indice+1;
      TCard(indice)=i/Fs;   %Calcula el período cardiaco y
      i=0;

      if indice>=11,        %Calcular el período cardiaco promedio de las
últimas
         divisor=10;        %10 pulsaciones
         SUM=SUM+TCard(indice)-TCard(indice-10);
      else
         divisor=indice;
         SUM=SUM+TCard(indice);
      end
      TCardProm(indice)=(SUM)/divisor; %Calcula el periodo promedio

      if indice>=2,         %Calcula la diferencia en segundos entre una
pulsación y la anterior
         dTC=TCard(indice)-TCard(indice-1);
         DTCard(indice)=(dTC);
      else
         DTCard(indice)=0;
      end
   end
end
```

```
% Redes SOM y LVQ
% Estas redes se crearon mediante la instrucción network a partir de los
parámetros obtenidos para las redes neuronales utilizadas en el
entrenamiento y simulación con las señales de prueba.

som = network(1,1,[0],[1],[0],[1],[0]);
som.layers{1}.transferFcn = 'compet';
som.inputs{1}.range = [0 1; -1 1];
som.layers{1}.dimensions=3;
som.layers{1}.distanceFcn = 'linkdist';
som.inputWeights{1}.initFcn = 'midpoint';
som.inputWeights{1}.learnFcn = 'learnsom';
som.inputWeights{1}.weightFcn = 'negdist';
som.IW{1}=[0.4434   -0.4737;
           0.5656   -0.0476;
           0.7133    0.4729];


lvq=network(1,2,[0; 0],[1; 0],[0 0; 1 0],[0 1],[0 1]);
lvq.inputs{1}.size=2;
lvq.layers{1}.dimensions = 4;
lvq.layers{1}.transferFcn = 'compet';
lvq.layers{2}.dimensions = 2;
lvq.inputWeights{1}.initFcn = 'midpoint';
lvq.inputWeights{1}.learnFcn = 'learnlv2';
lvq.inputWeights{1}.weightFcn = 'negdist';
lvq.adaptFcn= 'adaptwb';
lvq.initFcn= 'initlay';
lvq.performFcn= 'mse';
lvq.trainFcn= 'trainwb1';
lvq.trainParam.epochs= 1000;
lvq.IW{1,1}=[0.3421    0.0701;
             0.9703    0.0667;
             0.6632    0.1132;
             0.6648    0.0489];
lvq.LW{2,1}=[1     0     0     0;
             0     1     1     1];

% Simulación de una red neuronal SOM y LVQ

num=min(max(size(TCard)),max(size(QRS)));
P(1,1:num)= TCard(1:num)/1.5;
P(2,1:num)= DTCard(1:num)/0.85;

salida=sim(som,P);      %simula la red "net" con la entrada "P" (SOM)

for j=1:num,
    for i=1:3,
       if salida(i,j)==1 %la neurona ganadora fue la 1 y se le da el
valor 1 = LARGO
         salida(i,j)=i;  %la neurona ganadora fue la 2 y se le da el
valor 2 = NORMAL
         NS(j)= i;       %la neurona ganadora fue la 3 y se le da el
valor 3 = CORTO
       end
    end
end
```

```
Plvq(1,1:num)= NS(1:num)/3;
Plvq(2,1:num)= QRS(1:num);
salidalvq=sim(lvq,Plvq);    %simula la red net2 con la entrada Plvq
Yc = vec2ind(salidalvq);
for i=2:num,
   if (Yc(i)-Yc(i-1))==0,
      out1(i)=1;
      out2(i)=0;
   else
      out1(i)=0;
      out2(i)=1;
   end
end

figure(1);
plot(out1,'.g');
hold on;
plot(out2,'.r');
title('Diagrama de Latidos (verde=NORMAL   rojo=PVC)');
axis([1 num 0.96 1.04])
xlabel('Número de Latido');
Y=0;
while Y<1.02,
   figure(1),
   [X,Y] = GINPUT(1);
   X=round(X);
   if Y<1.02,
      if posicion(X)<=500
         point=501;
      else
         point=posicion(X);
      end
      if point-1000<=501,
         x1=501;
      else
         x1=point-1000;
      end
      if point+1000>max(size(s)),
         x2=max(size(s5));
      else
         x2=point+1000;
      end
      figure(2);
      t=[x1:1:x2]/Fs;
      t1=x1/Fs;
      t2=x2/Fs;
      plot(t,(s(x1:x2)-127)*5000/153600,'k');
      hold on
      plot(t,(Pico_R(x1:x2)-127)*5000/153600,'*r');
      title('Porción de la señal seleccionada');
      axis([t1 t2 -3.835 4.1666]);
      xlabel('Tiempo (seg)');
      ylabel('Amplitud (mV)');
      pause;
   end
   close;
end
```

```
% Guarda en Ecgag\salida.red la cantidad de latidos normales y
anticipados
h=hist(Yc,2);
fid=fopen('c:\Ecgag\salida.red','w');
fprintf(fid,'%i  %i\n',h);
fclose(fid);
```

APENDICE C
Medición de Amplificadores y Filtros

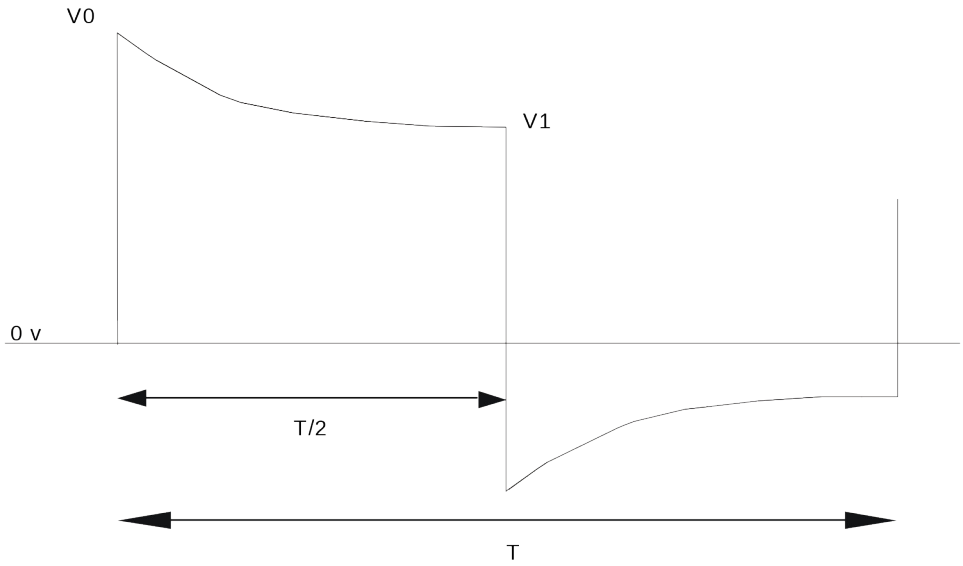
Aquí se detallan las mediciones que se realizaron en los amplificadores diferenciales y filtros pasabanda que se usaron en el equipo. Las mediciones se efectuaron mediante un osciloscopio digital aplicando dos tipos de señales: senoidales y cuadradas.

Aplicando señales senoidales y variando su frecuencia se obtuvieron los siguientes valores de la frecuencia de corte en baja, en alta y de ganancia:

Canal	Fc en baja	Fc en alta	Ganancia
DII	0.045 Hz	118 Hz	700 veces
V2	0.050 Hz	120 Hz	700 veces
V5	0.047 Hz	122 Hz	700 veces
Exp..	0.048 Hz	120 Hz	700 veces

Para la medición de la frecuencia de corte con la señal cuadrada se usaron dos frecuencias: una cercana a la frecuencia de corte en baja, y otra cercana al frecuencia de corte en alta.

Para el caso de la medición de la frecuencia de corte en baja se usó una frecuencia Fs=1 Hz y se obtuvo una salida muy similar a la de la figura, de la que se obtuvieron los valores Vo y V1.



Salida de los amplificadores para una señal de entrada de 1Hz

Luego, haciendo uso de la ecuación que representa una descarga de capacitor, la cual es:

$$V=Vo.\ e^{-t/\tau}$$

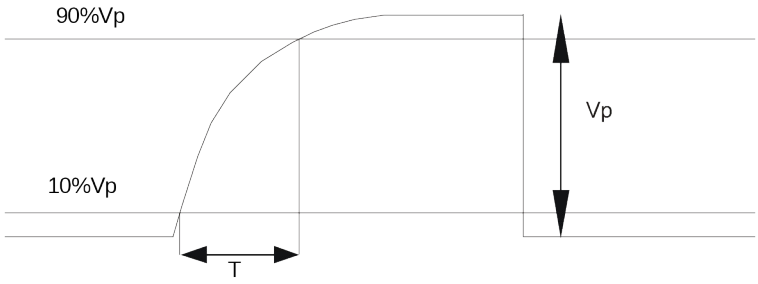
se llega a que la frecuencia de corte en baja (Fc_1), que está dada por:

$$Fc_1=2.\ln(Vo/V1)\ /(\ 2\pi\ T)$$

donde T es el periodo de la señal de entrada T=1/Fs.
En el siguiente cuadro se detallan las mediciones para cada canal:

Canal	Vo	V1	Fc1
DII	2.51 v	2.18 v	0.045 Hz
V2	2.48 v	2.16 v	0.044 HZ
V5	2.49 v	2.16 v	0.045 Hz
Expl.	2.5 v	2.16 v	0.046 Hz

Para la medición de la frecuencia de corte en alta se uso una frecuencia Fs=80 Hz, obteniéndose como salida una señal similar a la de la siguiente figura, a la que se le midió Vp y T.



Salida de los amplificadores para una señal de entrada de 80 Hz

Luego, haciendo uso de la ecuación de carga de un capacitor, la cual esta dada por:

$$V=Vp.(1-e^{-t/\tau})$$

se llega a que la frecuencia de corte en alta (Fc_2) esta dada por:

$$Fc_2=\ln(9)\ /(\ 2\pi\ T)$$

En el siguiente cuadro se detallan las mediciones para cada canal:

Canal	T	Fc2
DII	3.0 ms	116.56 Hz
V2	2.9 ms	120.58 Hz
V5	3.0 ms	116.56 Hz
Expl.	3.1 ms	112.80 Hz

APÉNDICE D
Resultados de la Prueba de la Red Neuronal

Los registros utilizados para la prueba y entrenamiento de la red fueron extraídos de la base de datos de arritmias del MIT-BIH. A continuación se brinda un resumen de las características de dichos registros así como información del tipo de paciente de quienes se obtuvieron.

Registro **100** (MLII, V5; hombre, edad: 69)

Medicaciones: Aldomet, Inderal

Latidos	**Antes de 5:00**	**Después de 5:00**	**Total**
Normales	367	1872	2239
APC	4	29	33
PVC	-	1	1
Total	371	1902	**2273**

Ectopía supraventricular

- 33 latidos aislados

Ritmo	**Frecuencia**	**Episodios**	**Duración**
Normal sinus rhythm	70-89	1	30:06

Calidad de la señal	**Episodios**	**Duración**
Ambas	1	30:06

Puntos de interés:
11:03 Ritmo sinusal normal
25:13 PVC
26:09 APC
27:55 Ritmo sinusal normal

Registro **105** (MLII, V1; mujer, edad:73)

Medicaciones: Digoxin, Nitropaste, Pronestyl

Latidos	**Antes de 5:00**	**Después de 5:00**	**Total**
Normales	405	2121	2526
PVC	12	29	41
Inclasificables	-	5	5
Total	417	2155	**2572**

Ectopía ventricular

- 41 latidos aislados

Ritmo	Frecuencia	Episodios	Duración
Ritmo sinusal normal	78-102	1	30:06

Calidad de la señal	Episodios	Duración
Ambas limpias	31	22:18
Ruido en la superior	3	0:10
Ruido en la inferior	28	3:27
Ambas ruidosas	23	4:06
Ilegible	4	0:04

Notas:

Los PVC son uniformes. Lo más resaltable de esta registro es el alto grado de ruido y artefactos

Puntos de interés:
05:27 Artefacto
07:57 PVC
15:16 Ritmo sinusal normal
17:52 Artefactos
22:02 Ruido
26:45 PVC
27:27 Ruido
28:08 Ruido
29:07 Ruido

Registro **106** (MLII, V1; mujer, edad:24)

Medicaciones: Inderal

Latidos	Antes de 5:00	Después de 5:00	Total
Normales	271	1236	1507
PVC	60	460	520
Total	331	1696	**2027**

Ectopía ventricular

- 327 latidos aislados
- 95 acoplamientos
- 1 seguidilla de 3 latidos

Ritmo	Frecuencia	Episodios	Duración
Ritmo sinusal normal	49-87	21	22:36

Bigeminia ventricular	55-103	18	7:15
Trigeminia ventricular	57-90	1	0:13
Taquicardia ventricular	121	1	0:02

Calidad de la señal	**Episodios**	**Duración**
Ambas limpias	15	16:25
Ruido en la inferior	15	13:41

Notas:

Los PVC son multiformes.

Puntos de interés:

00:19 Ritmo sinusal normal, ruido en la señal inferior
01:37 Acoplamientos ventriculares
02:53 Taquicardia ventricular, 3 latidos
04:23 PVC
07:57 Ruido en la señal inferior
10:52 Ruido en la señal inferior
12:27 Bigeminia ventricular (2 tipos)
16:17 PVC multiforme, acoplamiento ventricular
25:13 Acoplamiento ventricular
25:52 Acoplamientos ventriculares

Registro **200** (MLII, V1; hombre, edad:64)

Medicaciones: Digoxin, Quinidine

Latidos	**Antes de 5:00**	**Después de 5:00**	**Total**
Normales	305	1438	1743
APC	2	28	30
PVC	126	700	826
PVC de fusión	-	2	2
Total	433	2168	**2601**

Ectopía supraventricular

- 28 latidos aislados
- 1 acoplamiento

Ectopía ventricular

- 721 latidos aislados
- 42 acoplamientos
- 5 seguidillas de 3 latidos
- 2 seguidillas de 4 latidos

Ritmo	**Frecuencia**	**Episodios**	**Duración**

Normal sinus rhythm	69-111	70	15:58
Ventricular bigeminy	60-108	71	13:52
Ventricular tachycardia	90-141	7	0:15

Calidad de la señal	**Episodios**	**Duración**
Ambas limpias	14	21:44
Ruido en la superior	6	0:44
Ruido en la inferior	16	6:36
Ambas ruidosas	8	1:02

Notas:

Los PVC son multiformes. Hay explosiones ocasionales de ruido de alta frecuencia en al canal superior, y ruido severo y artefactos en el canal inferior.

Puntos de interés:
01:42 Taquicardia ventricular, 3 latidos
05:38 Ruido
18:14 Taquicardia ventricular, 4 latidos
20:52 Ruido
24:49 Taquicardia ventricular, 3 latidos
26:12 Acoplamientos ventriculares
28:31 Acoplamiento ventricular
29:01 APC, bigeminia ventricular
29:18 APC, PVC
29:51 PVC

Registro **203** (MLII, V1; hombre, edad:43)

Medicaciones: Coumadin, Digoxin, Heparin, Hygroton, Lasix

Latidos	**Antes de 5:00**	**Después de 5:00**	**Total**
Normales	426	2103	2529
APC aberrante	2	-	2
PVC	71	373	444
PVC de fusión	-	1	1
Inclasificable	-	4	4
Total	499	2481	**2980**

Ectopía supraventricular

- 2 latidos aislados

Ectopía ventricular

- 238 latidos aislados
- 64 acoplamientos
- 13 seguidillas de 3 latidos
- 6 seguidillas de 4 latidos
- 1 seguidilla de 7 latidos
- 1 seguidilla de 9 latidos

Ritmo	Frecuencia	Episodios	Duración
Ritmo sinusal normal	63-173	1	2:43
Aleteo arterial	61-180	2	5:14
Fibrilación arterial	54-180	20	21:32
Trigeminia ventricular	100-116	1	0:04
Taquicardia ventricular	124-189	21	0:33

Calidad de la señal	Episodios	Duración
Ambas limpias	21	24:28
Ruido en la superior	20	3:17
Ruido en la inferior	7	1:49
Ambas ruidosas	8	0:30
Ilegible	1	0:02

Notas:

Los PVC son multiformes. Hay cambios en la morfología del QRS en el canal superior debido a desplazamientos de los ejes. Hay considerable cantidad de ruido en los dos canales, incluyendo artefactos musculares y desplazamientos de la línea de base. Este es un registro muy difícil, aún para el ojo humano.

Puntos de interés:

05:00 Taquicardia ventricular, 4 latidos y 9 latidos
13:14 Fibrilación arterial, acoplamiento ventricular
15:02 Ruido
22:02 Acoplamiento ventricular, PVC
23:25 Ruido
24:04 PVC
24:46 Ruido
26:39 Taquicardia ventricular, 7 latidos
26:51 Acoplamiento ventricular, PVC
27:15 Taquicardia ventricular, 3 latidos

Notas:

- La duración de los episodios está dada en minutos.
- PVC: contracción ventricular prematura.
- APC: contracción arterial prematura.

Resultados de las pruebas del detector implementado:

Señal	Latido Corto	Latido Normal	Total de Latidos	Latidos Detectados – Total de Latidos
100	18 / 34	2254 / 2239	2272 / 2273	-1
105	116 / 41	2484 / 2526	2600 / 2572	28
106	603 / 520	1491 / 1507	2094 / 2072	22
200	399 / 856	2319 / 1743	2718 / 2601	117
203	859 / 446	2416 / 2529	3275 / 2980	295

Tasa de falsos positivos (latidos cortos detectados/latidos cortos reales)

- **100:** TFP = 0.52
- **105:** TFP = 2.82
- **106:** TFP = 1.15
- **200:** TFP = 0.46
- **203:** TFP = 1.92

- En la señal 100 se detectaron prácticamente todos los latidos, pero la tasa de latidos cortos detectados es baja a pesar de que es la señal más “limpia”. Esto puede ser consecuencia de que nuestro clasificador tiene como parámetro de entrada la diferencia entre dos períodos consecutivos y no hay diferencia de períodos entre los latidos con APC, por lo que están siendo clasificados como normales.

- En las señal 105 se observa una alta tasa de falsos positivos, esto puede deberse al alto nivel de ruido que presenta la señal y a la presencia de artefactos que pueden confundirse con falsos complejos QRS. Como puede verse en la tabla, el total de latidos detectados fue de 2600, cuando en el registro hay sólo 2572 latidos.

- La señal 106 es la que tiene mejor TFP dado que esta fue la señal utilizada para entrenar la red.

- En el caso de la señal 200 la TFP es sumamente baja, detecta menos de la mitad de los latidos cortos.

- Por último, la señal 203, como ya se mencionó, es una señal sumamente difícil de analizar, aún para el ojo del médico. En este caso la TFP es mayor a la unidad, por lo que se están detectando más cantidad de latidos ectópicos de los que en realidad hay.

BIBLIOGRAFÍA

- ***Interactive Electrocadiography***. Programa basado en el *Basic Electrocardiography* de Stephen Scheidt y desarrollado por Hurley Myers, Ahmad F. Moukaddem y Nong Tongsak.

- ***El diagnóstico electrocardiográfico de las arritmias***. ***Metodología de análisis y definiciones.*** Dra. M. Susana Halpern

- ***Amplificación de potenciales bioeléctricos.*** Carlos A. Lopez y Luis Llamosa. Revista *Electrónica y Computadores.*

- ***Hojas de datos de los circuitos integrados utilizados.***

- ***The principles of QRS detection.*** Bert-Uwe Köhler, Carsten Hemmig, Reinhold Orglmeister. Universidad de Tecnología de Berlín. IEEE Enero/Febrero 2002.

- ***Optimal QRS detection algorithms.*** Aman cohen y Mladen Poluta. Universidad Ben Gurion, Israel y Universidad de Cape Town, Sudáfrica.

- ***Design of a VLSI neural network arrhythmia classifier.*** H. Shawkey, H. Elsimary, H. Haddara, H. F. Ragaie. Universidad de El Cairo, Egipto. Feb. 1999.

- ***QRS detection using a fuzzy neural network.*** Kevin P. Cohen, Willis J. Tompkins, Adrianus Djohan, John G. Webster y Yu H. Hu. Universidad de Wisconsin. IEEE 1995.

- ***Utilización de Modelo de Kohonen y del perceptrón multicapas para detectar arritmias cardíacas.*** L.B. Barbosa, G.H. Kleisinger, A.D. Valdez, J.E. Monzón. Universidad Nacional de Nordeste, Corrientes, Argentina. Congreso Latinoamericano de Ingeniería Biomédica, La Habana 2001, Cuba.

- ***Detection of the QRS complex, P wave and T wave in electrocardiogram.*** K.F. Tan, K.L. Chan y K. Choi. City University, Hong Kong.

- ***Procedimiento digital para la detección del complejo QRS y el establecimiento de la marca fiducial.*** Rubén Orozco Morales. Dic. 1998, Cuba.

- ***Is LVQ really good for classification? An interisting alternative.*** W. Poechmueller, M. Glessner, H. Juergs. Universidad de Tecnología de Darmstadt. Darmstadt, Alemania.

- ***Detección de parámetros electrocardiográficos con modelo SOM y LVQ de redes neuronales.*** G.H. Kleisinger, Eduardo Del Valle, Jorge Monzón. Universidad Nacional de Nordeste. Corrientes, Argentina.

- ***Tesis doctoral: "Estudio de Métodos para el procesamiento y agrupación de señales electrocardiográficas".*** David Cuesta Frau. Universidad Politécnica de Valencia.