UNIVERSIDAD OBERTA DE CATALUNYA

ET.Telecom. Telemàtica

TFC – Aplicacions electromagnètiques i electròniques

DISEÑO DE UN SENSOR DE ACTIVIDAD CARDÍACA

Memoria

Alumno/a: Ángel M. López Pérez

Dirigido por: Asier Ibeas Hernández

Índice

Índice de figuras

Índice de figuras de los anexos

Índice de tablas

Capítulo 1. Introducción y planificación del proyecto.

La esperanza de vida de las personas en el paleolítico superior (10.000 años a C.) era de 35 a 45 años[1], y ahora, en el año 2010 es de 68.9 años[2] y sigue creciendo, esto es debido a las mejores condiciones de vida, y este trabajo quiere colaborar en esa dirección. Para ello, en este proyecto vamos a desarrollar un sensor que mide la actividad eléctrica del corazón.

En este capítulo veremos los rudimentos del sistema circulatorio, y estudiaremos los orígenes de la actividad eléctrica del corazón, que es la base de este proyecto. Marcaremos los objetivos que puedan hacer realidad esta idea, así como la metodología empleada, sin olvidar la viabilidad técnica. Finalmente veremos el esquema general de funcionamiento y su planificación basada en diagramas de Gantt.

1.1. Idea general del proyecto

El sistema cardiovascular

El corazón tiene una función vital en el organismo humano, impulsando el flujo sanguíneo para que llegue a todos los órganos del cuerpo.

Básicamente está formado por cuatro cavidades de tejido blando: dos aurículas y dos ventrículos, dos válvulas que controlan el flujo de sangre entre cada aurícula y su ventrículo, los vasos sanguíneos que riegan los músculos cardíacos y los centros nerviosos que regulan su actividad, como podemos ver en la figura 1.

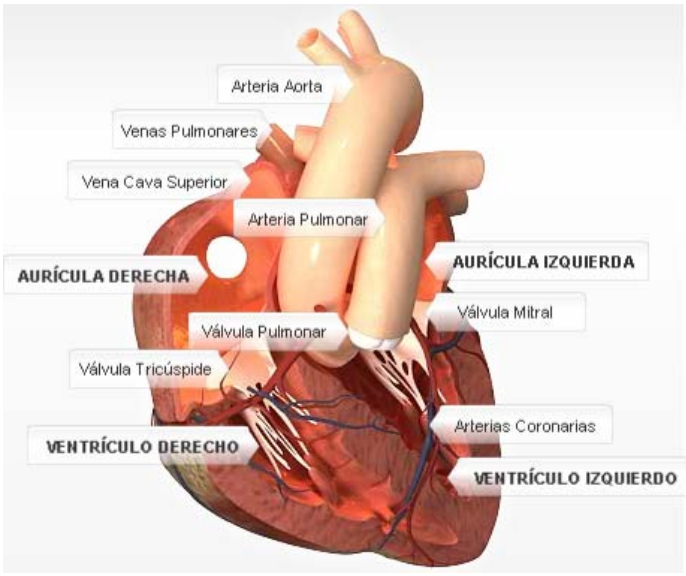


Fig. 1 Partes del corazón humano, fuente: http://www.fundaciondelcorazon.com

En el corazón se distinguen dos circuitos ó circulaciones:

- La circulación mayor comienza en la aurícula izquierda. Entonces, la válvula mitral, que es la que separa la aurícula del ventrículo izquierdo se relaja y deja pasar la sangre al ventrículo izquierdo. A continuación y una vez que se ha llenado el ventrículo izquierdo, se cierra la válvula mitral y se produce una fuerte contracción del ventrículo izquierdo, impulsando con fuerza la sangre hacia la arteria aorta[3] y a través de ella a todos los órganos que reciben el oxigeno y el alimento que transporta el torrente sanguíneo. Los órganos consumen los nutrientes y producen CO_2 y otros productos de desecho que eliminan a través de la red venosa. Esta sangre pobre en oxígeno fluye hacia las venas cava superior y cava inferior, llegando a través de ellas a la aurícula derecha.

- La circulación menor o pulmonar comienza en la aurícula derecha. El flujo venoso llega de forma continua[4] a la aurícula derecha. En esta fase la válvula tricúspide que separa la

[1] Fuente de los datos: Historia Universal Salvat, según los estudios de Vallois
[2] Fuente: Banco Mundial (Indicadores de desarrollo mundial)
[3] La aorta tiene en su base la válvula aortica que permite el paso del flujo sanguíneo pero impide el retorno.
[4] Recordemos que las venas son colectores del flujo sanguíneo de retorno, y tienen válvulas en todo su recorrido.

aurícula derecha del ventrículo derecho se relaja, y deja pasar la sangre al ventrículo derecho, a continuación se cierra la válvula tricúspide y se produce la contracción rápida del ventrículo derecho, impulsando la sangre por la arteria pulmonar[5] hacia los pulmones. La sangre en los pulmones se desprende del anhídrido carbónico CO2 y se enriquece con oxigeno O2, de aquí, regresa por las venas pulmonares hacia la aurícula izquierda completando el ciclo.

La contracción de las aurículas derecha e izquierda es simultánea, al igual que ocurre con los ventrículos derecho e izquierdo, aunque se estudia por separado para entender mejor cada recorrido de la circulación sanguínea.

1.2. Como se puede obtener información de la actividad eléctrica del músculo cardíaco.

Los especialistas pueden obtener información de los ciclos de contracción auricular y ventricular, por varios métodos, simplemente escuchando el sonido que produce el corazón al bombear la sangre, evaluando la presión arterial, midiendo la corriente que se produce en los centros nerviosos del corazón (ver figura 2.). Además existen sofisticas técnicas que permiten obtener imágenes ecográficas del corazón, o combinando varias técnicas como en la ecocardigrafía.

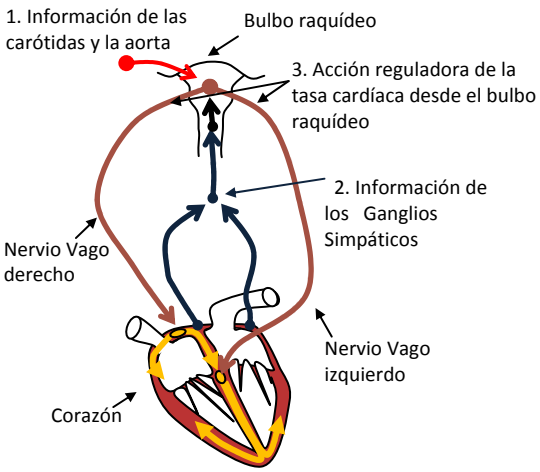


Fig. 2 Esquema sinusal, fuente: propia

El estudio del sistema cardiovascular es de suma importancia para la salud de las personas. En la mayoría de los países avanzados, la primera causa de mortalidad se debe a enfermedades del corazón, y así, en España en el año 2007 las enfermedades del sistema circulatorio[6] fueron del 32.2 % sobre todas las enfermedades diagnosticadas, y las causas de defunción debidas a estas enfermedades fue del 23.45 % (ver anexo 5).

Nosotros centraremos nuestro estudio en la actividad eléctrica del corazón, y esperamos que este trabajo pueda contribuir a mejorar la salud, en la medida de sus posibilidades.

Para que sirven las medidas de la actividad cardíaca:

Los médicos y el personal sanitario entrenado, puede interpretar adecuadamente las observaciones y medidas de la actividad cardíaca, deduciendo de las mismas si se trata de una actividad normal o de una patología.

Por tanto, de la observación de las medidas del músculo cardíaco, se puede deducir si el corazón tiene alguna patología y en ese caso y debido a la repercusión vital sobre la persona afectada, normalmente debe ser tratada inmediatamente.

[5] La arteria pulmonar tiene en su base la válvula pulmonar que permite el paso de sangre pero impide el retorno.
[6] Fuente: Instituto Nacional de Estadística. INE

La base de este trabajo es el análisis de los impulsos eléctricos como indicadores del estado de la actividad cardíaca. Los impulsos eléctricos que activan los músculos cardíacos, se originan en los nódulos del corazón. Podemos ver en la figura 3 la ubicación del nódulo SA[7] que, es donde se genera el latido de forma autónoma, aunque está controlado por el sistema nervioso vegetativo, se puede decir que el nódulo sinusal es el marcapasos del corazón.
Los impulsos eléctricos son conducidos por el nódulo AV[8] auriculoventricular, y de ahí a los músculos cardíacos.

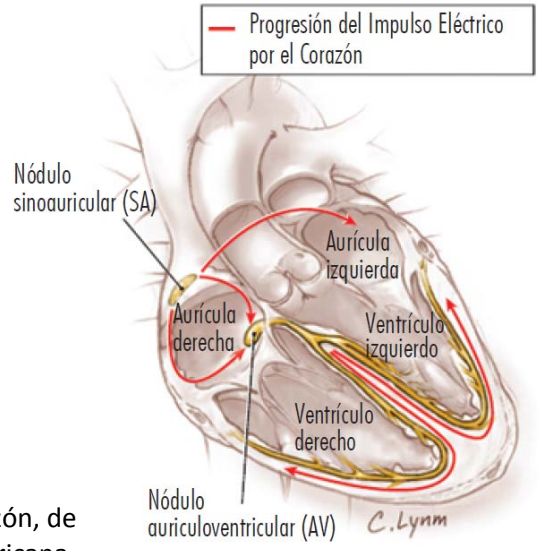


Fig. 3 Impulso eléctrico a través del corazón, de la Revista de la Asociación Médica Americana

1.3. Descripción del proyecto

El presente trabajo describe el diseño de un dispositivo para la captura y análisis de los impulsos eléctricos (señales bioeléctricas) de la actividad cardíaca, en personas de cualquier edad.

Una vez capturadas las señales eléctricas, amplificadas y procesadas para su tratamiento digital, ya tendremos la información necesaria para evaluar la actividad cardíaca del corazón y en caso de funcionamiento anómalo emitir las alarmas pertinentes, que permitan la acción médica adecuada.

La generación de alarmas es la siguiente: si el ritmo cardiaco es el adecuado el equipo no emite alarma, pero si ocurre una desviación de la frecuencia de los latidos del corazón por baja frecuencia, se produce alarma de **bradicardia**[9] y en el caso de que la frecuencia de los latidos sea alta se producirá alarma de **taquicardia**[10], además la programación de las alarmas por superación de los umbrales de las frecuencias que marcan bradicardia ó taquicardia, la podrá realizar el propio usuario del dispositivo.

El uso de este equipo electrónico es autónomo y está especialmente indicado para quienes desean controlar su actividad cardíaca[11] en actividades deportivas, por razones médicas, u otras causas.

[7] El nódulo sinoauricular, está situado en la aurícula derecha y es donde se origina el impulso eléctrico que da origen a un latido cardíaco.
[8] El nódulo auriculoventricular está localizado en el atrio derecho del corazón, y se comunica con el nódulo sinusal a través de tres fibras internodales compuestas de músculo cardíaco especializado en la conducción de impulsos
[9] Se produce bradicardia cuando el nódulo sinusal (donde se origina el impulso eléctrico que da origen a un latido cardíaco) descarga a una frecuencia menor de 60 latidos/min.
[10] Se considera taquicardia cuando la frecuencia cardíaca es superior a 100 latidos/min. en reposo.

[11] Este dispositivo es para uso privado y los pacientes que requieran tratamiento en centro médico u hospitalario, se regirán por las normas y procedimientos que se establezcan en cada centro.

El objetivo de este trabajo llega hasta aquí. La plataforma de diseño es escalable y se deja abierta para nuevas prestaciones y mejoras derivadas del proyecto, las futuras prestaciones que se pueden añadir son:

- Medida de frecuencias muy altas del puso cardíaco para reconocer la fibrilación ventricular
- Reconocimiento de formas de onda en los pulsos eléctricos cardíacos para poder prevenir infarto de miocardio
- Sensor de movimiento tipo S320150[12] para medir y grabar el estado de reposo del sujeto.
- Grabación en formato PCM ley μ[13] (*Pulse Code Modulation*) del audio de los latidos del corazón, a través de un sistema microfónico adosado a la propia banda del pecho.
- Medida y grabación a través de un sensor termopar adosado a la banda del pecho, de la temperatura del sujeto a la altura del músculo serrato mayor.
- Medida y grabación de la frecuencia respiratoria mediante un sensor de presión insertado en la banda pectoral.
- Todas las muestras de frecuencia cardíaca, sensor de reposo, sonido, temperatura, frecuencia respiratoria, etc. podrán ser presentadas en forma de listado fechado ó similar (log ó bitácora del sujeto).

1.4. Objetivos del proyecto

El objetivo principal del proyecto, que consiste en el diseño de un sistema medidor programable de los impulsos eléctricos del corazón, y que además es capaz de generar las alarmas de bradicardia y taquicardia, según la frecuencia del pulso cardíaco observado, y en los casos descritos en el apartado 1.2, se logra a través de los siguientes objetivos parciales.

Objetivo 1. Medir la señal biométrica del pulso cardíaco en una persona normal.

Dada a la dificultad para medir las tensiones que genera el corazón, que son del orden de un milivoltio pico a pico, hemos considerado un objetivo básico y previo al trabajo de diseño, el hecho de poder medir en la práctica las señales bioeléctricas del corazón.

Estas señales además se ven generalmente perturbadas por un ambiente de ruido electromagnético, lo que hace aún más difícil su medición.

Objetivo 2. Lograr una etapa de adaptación de la señal bioeléctrica.

La etapa de adaptación proporcionará una señal nítida en el intervalo de 0 a 3 segundos a partir de la señal medida con ruido y un intervalo de amplitudes de -0.25 mV a +1.25 mV

Para conseguir este objetivo, primero hay que caracterizar la señal analógica emitida por el corazón, mediante un modelo matemático que sea adecuado para las herramientas de diseño que vamos a utilizar, estas son: *Pspice* para el diseño de circuitos, *Scilab* de simulación y cálculo científico, y MPLAB de diseño y simulación de sistemas basados en microcontrolador.

[12] Sensor de vibración sin mercurio específicamente diseñado para la detección del movimiento y la vibración

[13] Ley de cuantificación logarítmica: $v(x) = \frac{\ln\left(1+\frac{\mu(x)}{x_{máx}}\right)}{\ln(1+\mu)}\operatorname{sgn}(x)$, siendo μ el número de intervalo. Fuentes: Apuntes de Transmisión Digital y WEB de Cisco [https://supportforums.cisco.com/docs/DOC-1426]

Objetivo 3. Muestrear la señal eléctrica del pulso cardíaco sin perder representatividad

En este objetivo se plantean también varias fases para lograr el tratamiento digital de las señales muestreadas, para ello primero hay que determinar la frecuencia de muestreo mínima del convertidor analógico- digital, una vez tenemos este dato podemos seleccionar el microcontrolador que cumpla este requisito fundamental, además debe cumplir otras especificaciones mínimas como son la frecuencia de reloj, la capacidad de memoria, y los periféricos necesarios, de acuerdo a las necesidades del proyecto.

Objetivo 4. Presentar la información y leer las órdenes del usuario

Otro objetivo del proyecto, es poder presentar los resultados del tratamiento digital de la información al usuario, para que éste interprete las indicaciones y alarmas que se puedan generar, de igual modo el usuario debe poder enviar las órdenes necesarias al equipo digital, para que éste se configure y trabaje correctamente.

Objetivo 5. Realizar un programa de reconocimiento de señal y disparo de alarma

En este objetivo debemos lograr que el programa de reconocimiento tomando la señal discreta muestreada, pueda identificar los pulsos principales o picos de la señal, registrando el período medido, en un intervalo de tiempo máximo de 5 segundos.

Una vez obtenido el período medio de los latidos, el programa calculará el ritmo cardíaco en pulsaciones por minuto, si el valor de las pulsaciones es menor que 60 en un intervalo de tiempo de 1 minuto activará una alarma de BRADICARDIA, si dicho valor de pulsaciones es mayor de 100 en un tiempo de 30 segundos, el software activará una alarma de TAQUICARDIA.

Estos umbrales de alarma son los normales o estándar, pero se pueden modificar de acuerdo al objetivo 4, ya que el sistema puede ser reprogramado por las órdenes que recibe del usuario.

De igual forma, la desactivación de las alarmas se puede hacer por indicación del usuario, ó porque los valores de ritmo cardíaco vuelven a su estado normal dentro del intervalo programado, ó en su defecto a los valores estándar de 60 a 100 ppm, durante un período mínimo de 2 minutos.

1.5. Metodología para el proyecto

Los objetivos definidos en el apartado anterior, deben materializarse siguiendo la metodología que se detalla a continuación:

- La relación precio-producto es la principal estrategia a seguir en el proyecto, que estará presente en todas las fases del mismo, y cuyo resultado debe poder materializarse con la tecnología disponible actualmente.
- Todas las aportaciones al proyecto deberán estar justificadas y contrastadas con las demostraciones necesarias, así como con los datos que lo justifiquen, y en las referencias a otros trabajos, se citará la fuente o fuentes de procedencia.
- Una vez definidos claramente los problemas a resolver en el proyecto, e identificados los recursos disponibles para su resolución, la metodología a seguir consiste en fragmentar dichos problemas en partes más simples, como es el caso del objetivo general, que se subdivide en objetivos parciales, y estos en fases.

- Creación de un modelo matemático que esté identificado con los valores reales observados

- Diseño de las soluciones de la parte hardware y del software, tanto por separado como en conjunto.

- Las pruebas necesarias a lo largo del proyecto, que sólo serán reales en la toma de muestras, ya que todas las demás se harán por simulación con herramientas de probada eficacia.

- Poner todas las experiencias en esta memoria y en los informes de seguimiento necesarios, y con todo ello elaborar una presentación que sea fiel al proyecto, describiendo con claridad y precisión el problema, el desarrollo y los resultados.

1.6. Viabilidad técnica del proyecto

Como sabemos por las fuentes consultadas[14 y 15] y las medidas reales del electrocardiograma (ECG), las señales capturadas son muy débiles, deben ser amplificadas y separadas galvánicamente (eléctricamente) entre la parte analógica de las señales continuas, y la parte digital que contiene toda la lógica de conversión analógico-digital[16], el procesador, la memoria y la interfaz de usuario.

La interfaz de usuario por un lado presenta la información al usuario para que éste pueda leerla o visualizarla sin dificultad y por otro lado le permite introducir las órdenes al sistema de procesamiento digital.

Mediante técnicas de amplificación y filtrado, las señales procedentes del corazón pueden ser convertidas y tratadas por el bloque digital (estas partes se describirán en detalle más adelante), por tanto podemos evaluar inicialmente los recursos necesarios para materializar el proyecto, de la siguiente forma:

- Un amplificador de alta calidad del tipo utilizado en instrumentación para multiplicar la señal sin distorsión. En este proyecto vamos a usar el modelo INA114 AP, que es un circuito integrado del fabricante *Burr-Brown Corporation*, y está disponible por 9.5 € /unidad, según factura de compra (ver anexo 6)

- Para evitar la influencia de frecuencias de señales externas no deseadas, será necesario filtrar la señal con un filtro paso banda, en la banda de las frecuencias de trabajo de 5 hasta 45 Hertzios (Hz). Este filtro se puede implementar fácilmente mediante dos filtros activos, uno de paso alto en serie con otro de paso bajo.

- Un convertidor analógico/digital A/D, para convertir las señales analógicas procedentes del filtro paso banda, en datos digitales que puedan ser tratados por el procesador ó microcontrolador que veremos en el siguiente párrafo.

  La impedancia de entrada del convertidor A/D será de 1 MΩ aproximadamente[17] y con una frecuencia de muestreo mínima de 5 KHz, ya que de acuerdo al teorema del muestreo de *Nyquist-Shannon* (ver la demostración en el anexo 8), la frecuencia de muestreo debe ser el

[14] WEB's http://es.wikipedia.org/wiki/Electrocardiograma
[15] http://www.bem.fi/book/19/19.htm
And documentation: **Certified EKG Technician Study Guide of NATIONAL HEALTHCAREER ASSOCIATION**
[16] Convierte las muestras analógicas en código binario para poder ser tratadas por el sistema de procesador digital.
[17] Este valor es normal en todos los dispositivos de conversión A/D

doble de la frecuencia muestreada, y como sabemos que existen frecuencias del espectro de la señal de hasta 2.5 KHz, la frecuencia de muestreo mínima en nuestro caso, será de: 2 * 2.5 KHz = 5 KHz.

Los cálculos necesarios, son:

V de ruido = 0.01 mV
V señal = 1 mV
Relación señal-ruido: S/N = 20log(1mV/0.01mV) = 40 dB[18] → s/n=1O^4 [01]
Máxima frecuencia del espectro de la señal: Fmáx = 2.5 KHz
Frecuencia de muestreo: Fs = 2* Fmáx = 5 KHz (por el teorema de *Nyquist-Shannon*)
Ancho de banda del filtro paso banda: B = 40 Hz
Capacidad del canal: C= B*$\log_2$(1+s/n) = 40* ln(10001)/ln2 = 40* 13.29 = 532 Hz [02]

- Es necesario un microcontrolador de 16 bits de ancho de bus de datos, 12 KB de memoria de programa y 512 Bytes de memoria de trabajo, para realizar la programación y el cálculo numérico, este dispositivo debe integrar un procesador digital de señal ó DSP, ya que el incremento de precio no afecta sustancialmente en la partida económica y posibilita un crecimiento futuro muy importante, de esta forma en la eventual ampliación del proyecto e incluyendo los sensores adecuados, se podrán medir una amplia variedad de constantes fisiológicas, todas ellas relacionadas con la actividad cardíaca del sujeto y podrán ser tratadas por la programación existente, para así presentar una información más completa y elaborada. La justificación de las capacidades de este microcontrolador se verán más adelante.

  El modelo que resuelve las necesidades actuales y futuras del proyecto ó las posibles mejoras y ampliaciones, es el circuito integrado de la casa *Microchip Technology*, el dsPIC 30F2010 que está disponible para entrega inmediata por 16 €.

- Para la interfaz con el usuario en principio, se utilizarán indicadores luminosos y/o acústicos para visualizar las alarmas emitidas por el sistema sensor, y pulsadores o teclado de membrana para las entradas al sistema, todo ello aislado eléctricamente (separación galvánica) en previsión de una interfaz más compleja basada en otros equipos que puedan estar conectados a la red de 220 V.

- Para alimentar todos los circuitos, se utiliza un sistema de batería de Níquel Metal Hidruro de 9V

- Las herramientas de diseño necesarias, son:

  1. Módulo de diseño analógico: PSPICE, para diseñar el amplificador y el filtro
  2. Módulo de modelado matemático: SCILAB, para simulación matemática.
  3. Módulo de diseño digital: MPLAB, para diseñar el software del microcontrolador
  4. Módulo de Oficina Técnica: Microsoft Office y Adobe Photoshop CS2
  5. Módulo de presentaciones virtuales.
  6. Medios técnicos de conexión ADSL y acceso a Internet, para conexión con el Tutor.

[18] Fuente consultada: Investigación y ciencia, Reducción de Ruido Digital en señales ECG con filtro por convolución

Podemos concluir que el proyecto es viable técnicamente, para ello, disponemos de los medios materiales, del conocimiento y del tiempo necesario para realizarlo, de acuerdo a los objetivos marcados en el apartado 1.3.

1.7. Diagrama de bloques de la aplicación

El diagrama de bloques representa la interconexión de los distintos elementos hardware y software, de tal manera que forman un conjunto que satisface los requerimientos del proyecto, lo vemos en la figura 4.

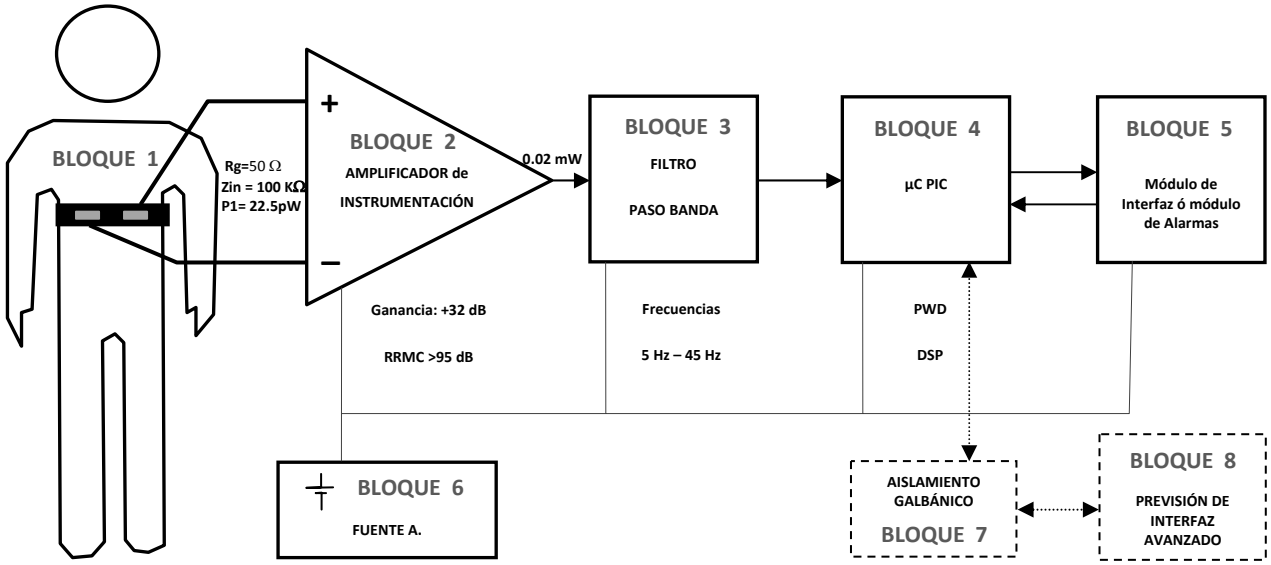


Fig. 4 Diagrama de bloques

A continuación describimos las funciones de cada uno de los bloques, según las tareas y los objetivos que le corresponden:

Los bloques 1 y 2 permiten cumplir el objetivo 1 de la siguiente manera:

- La banda del bloque 1 que es una cinta ó banda con electrodos, de las que se usan en actividades deportivas, se pone en contacto directo con la piel, para establecer el circuito a través de sus electrodos.

  En la parte central de la banda hay un par de cables soldados a los electrodos, que prolongan la conexión hasta la entrada del bloque 2, que es un circuito experimental, formado por un amplificador de instrumentación de ganancia G = 1000, y finalmente la salida del bloque 2, la podemos visualizar en la pantalla de un osciloscopio, ver figura 5.

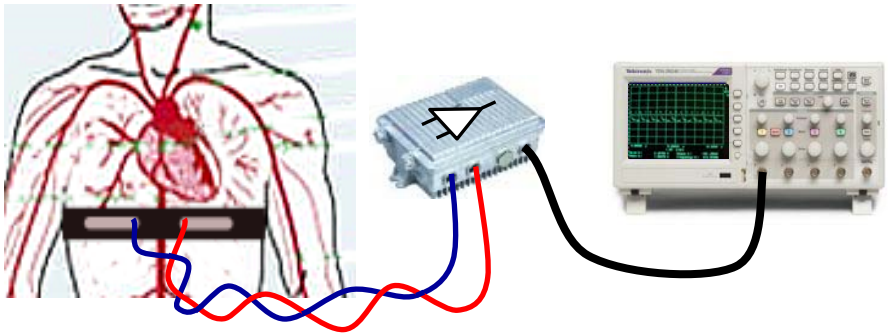


Fig. 5 Pruebas de los bloques 1 y 2

En resumen, la señal procedente del corazón es obtenida por el contacto de los electrodos con la piel (bloque 1), luego se amplifica (bloque2), y se monitoriza en la pantalla del osciloscopio.

El bloque 2, es necesario para obtener una muestra ampliada y sin distorsión de la señal original, ya que el resto de los circuitos no pueden trabajar con las tensiones tan bajas que genera el corazón. Para ello hemos seleccionado el amplificador de instrumentación INA114 AP, y lo haremos trabajar con los siguientes valores:

$G = 1+ (50\ K\Omega/R_G) = 1+ (50\ K\Omega/50\ \Omega)= 1001$, [03]
Es la ganancia del amplificador

$R_G=50\ \Omega$,
Es la resistencia de control de ganancia del amplificador de instrumentación

$G_{dB}= -20\log(V_{in}/V_{out})=-20\log(0.001)=60dB$ [04]
Es la ganancia en decibelios del amplificador.

Potencia de la señal de entrada P_1 y potencia de salida del amplificador P_2 :

$$P_1 = \frac{V_{in}^{\ 2}}{Z_{in}} = \frac{(1.5\cdot 10^{-3})^2}{10^5} = 22.5\ pW \quad [05]$$

$P_2 = P_1 \cdot G= (22.5 \cdot 10^{-12}) \cdot 1001 = 0.0225$ mW [06]

Siendo:

V_{in} = 1.5 mVpp, es la tensión pico a pico de entrada, procedente del corazón.
V_{out} = 1.5 Vpp, es la tensión pico a pico a la salida del amplificador.
Z_{in} = 100 KΩ es la impedancia de entrada del conjunto formado por la banda de electrodos y la piel

El Rechazo en modo común del amplificador INA114 AP es el CMRR[19] ó calidad del amplificador, para una ganancia G=1000 → $CMRR_{mín}$= 114 dB

$$CMRR = -20\log\left(\frac{\text{Ganancia diferencial de ruido}}{\text{Ganancia en modo común}}\right) = -20\log\left(\frac{\frac{V_{out}}{V_+ - V_-}}{\frac{V_{out}}{V_{in}}}\right) = -20\log\left(\frac{V_{in}}{V_+ - V_-}\right); \quad [07]$$

En nuestro caso la diferencia debida al ruido, es V+ - V- y vale 500 mV aprox.
Como el convertidor A/D debe discriminar hasta 1024 muestras, tenemos que:

$$Vm = \frac{1Vpp}{1024\cdot G} = 10^{-6}V \quad [08]$$

La señal Vm es la señal más pequeña que debe ser amplificada

$$\text{por tanto}: \quad CMRR = -20\log\left(\frac{10^{-6}}{0.5}\right) = 114\ dB \quad [09]$$

El diseño del bloque 3 es el último requisito para completar el objetivo 1, este bloque está constituido por un filtro activo paso banda, con frecuencias de paso entre 5 y 45 Hz, por lo que no afectará a las frecuencias centrales de la señal cardíaca, que en condiciones normales están en torno a los 22 Hz.

[19] *Common Mode Rejection Ratio*

Este filtro rechaza las frecuencias muy bajas < 5 Hz, que provienen de la propia actividad del cuerpo humano, en la contracción muscular ó por la acción metabólica, y las frecuencias muy altas >45 Hz que provienen de fuentes externas, como son:

- La RED eléctrica de 220 V (50-60 Hz), y los aparatos circundantes como interruptores, fluorescentes, etc., que producen electricidad estática e inducción electromagnética.
- Las fuentes de microondas, los teléfonos móviles, etc., que producen emisiones de radio.

Estas señales de ruido se suman a la señal original modificando, o anulando ésta.

El bloque 4 es el bloque digital, constituido por un microcontrolador (**µC)**, que satisface las necesidades del objetivo 2, ya que posee los elementos de conversión analógico/digital A/D, el procesador y la memoria, todos ellos necesarios para capturar y tratar los datos.

El µC debe responder a los requerimientos de velocidad del procesador y la resolución del convertidor A/D, será tal que permita representar la señal sin pérdida de información, como veremos en los puntos 2.1.2 y 2.1.3

El microcontrolador que utilizaremos, es el modelo dsPIC30F2010 del fabricante *Microchip Technology*. Este modelo incluye de serie un convertidor analógico digital ADC[20], mediante el cual se obtene una muestra digital de la señal analógica, la unidad central de proceso CPU (*Central Processing Unit)* que ejecutará las órdenes de la programación software, la memoria para almacenar los datos y los programas, y las unidades de entrada/ salida para interactuar con el periférico de interfaz de usuario. Además este **µC** integra un procesador digital DSP (*Digital Signal Processing*), que se usará en las fases de aplicación del proyecto.

Por último y para completar el objetivo 2, se debe realizar el trabajo de diseño del muestreador de señal, que se debe implementar en el convertidor A/D, y que veremos en detalle en el punto 3.6

El bloque 5, es el **módulo interfaz de usuario y de alarmas** está constituido por pulsadores y una pantalla LCD (*Liquid Cristal Display*) retroiluminada, para visualizar los mensajes e indicadores de la actividad y las alarmas que se generan en el procesador.

El bloque 5 se comunica con el procesador a través del bus serie I2C(*Inter-Integrated Circuit)*, La construcción de este bloque, cumple las necesidades del objetivo 4.

El bloque 6, es la fuente de alimentación que por su simplicidad y fácil implementación no se incluye en los objetivos, ésta se construye de forma aislada, con un generador de 9 V recargable.

Los bloques 7 y 8, no están incluidos en los objetivos, ya que se trata de módulos para la ampliación del proyecto, la descripción de éstos es la siguiente:

Módulo de aislamiento galvánico, este módulo está previsto que sea desarrollado en un próximo trabajo. Debe ser constituido con circuitos optoacopladores en ambos sentidos de la transmisión, es decir, que la información se transmita bidireccionalmente, mediante señales ópticas entre el procesador y el módulo Interfaz Avanzado.

[20] *Analog to Digital Converter*

Módulo de Interfaz Avanzado, al igual que el módulo de aislamiento, se desarrollará en un próximo trabajo y estará formado por los siguientes submódulos:
- Sumbmódulo de teclado ó terminal táctil OLED[21] y en este caso no sería necesario el *display* ya que el terminal OLED lo incorpora, y serviría para introducir las órdenes al procesador
- El submódulo de transmisión remota se utilizaría para la conexión con otros equipos
- La alimentación deberá estar separada por medio de una fuente de alimentación conmutada, y aislada por transformadores de alta frecuencia.

1.8. Lista de materiales

El material hardware necesario para realizar el proyecto, así como los precios de mercado de los componentes, se detallan en la tabla (1) .

Tabla 1: Lista de materiales

Item	Modelo	Denominación	Cantidad	Precio	Importe (€)
1	INA 114	Amplificador de instrumentación	1	9.5	9.5
2	OPA2604	Amplificador Operacional	1	3.8	3.8
3	IRF 150	Transistor MOSFET (De efecto de campo, canal N)	1	8.43	8.43
4	IRF 9140	Transistor MOSFET (De efecto de campo, canal P)	1	8.6	8.6
5	NE 555	Circuito Integrado Temporizador de precisión	2	1.02	2.04
6	CFA533-TMI	Display alfanumérico 2x16 con teclado	1	40.2	40.2
7	dsPIC2010	Microcontrolador, incluye procesador digital de señal DSP y convertidor Analógico/Digital ADC	1	7.0	7.0
8	Banda POLAR	Banda pectoral con electrodos	1	20.0	20.0
	Varios	Placa Circuito Impreso, bananas, cables, bobinas, condensadores, resistencias, baterías, etc.			19.8
TOTAL (€, IVA incluido)					119.37

1.9. Valoración económica del proyecto

La estimación de los costes de este proyecto, es la siguiente (ver tabla 2):

Tabla 2: Estimación de los costes del proyecto	
24 Horas del Tutor de la asignatura	450 €
112 Horas elaboración del proyecto y debate	1050 €
Recursos UOC	120 €
Material hardware (HW)	120 €
Material software (SW)	0 €
Desplazamiento Madrid – Barcelona	250 €
Material fungible y desplazamientos varios	50 €
Total	2040 €

[21] *Organic Light-Emitting Diode*

1.10. Resumen de fases

De cada uno los distintos objetivos y subobjetivos descritos en el apartado 1.3, podemos agrupar una serie de tareas, que por tener características comunes ó realizarse de forma cronológica, llamaremos fases, vemos la tabla (3) de fases a continuación.

Tabla 3 Resumen de fases

FASES	Predecesora	Objetivos a los que pertenecen					DESCRIPCIÓN
		1	2	3	4	5	
0		X					Demostración de que la señal eléctrica de actividad cardíaca se puede medir
1	0					X	Manejar y poner en práctica la herramienta de diseño MPLAB de Microchip. Observaciones: El diseño software del proyecto se basa fundamentalmente en esta herramienta, es objetivo primordial conocer el manejo, aplicación y simulación de los programas diseñados para el microcontrolador dsPIC.
2						X	Adquirir la formación sobre diseño analógico/digital y programación de microcontroladores PIC (CONTINUA)
3	0		X	X			Manejar y poner en práctica las herramienta de cálculo y diseño Scilab del Scilab *Consortium* de INRIA y la herramienta software de diseño de circuitos PSPICE
4	2			X			Calcular la velocidad de proceso y la frecuencia de muestreo
5	3		X				Diseñar el circuito del amplificador de instrumentación
6	3		X				Diseñar el circuito del filtro paso banda
7	6			X			Diseñar el bloque digital ó módulo microcontrolador
8	7				X		Diseñar el bloque interfaz de usuario y módulo de alarmas
9	7					X	Diseño del programa principal (*main*)
10	7					X	Diseño del programa de reloj
11	7					X	Diseño del programa de captura de señal
12	7					X	Diseño del programa de conversión A/D
13	1					X	Diseño del programa de introducción de umbrales de alarma
14	1					X	Diseño del programa del módulo de alarmas (disparo y salida de alarmas)
15	1					X	Módulos de programas misceláneos, (si fuese necesario)
16	1	X	X	X	X	X	Redacción de la memoria y trabajo de Oficina Técnica.

1.11. Planificación

En esta sección veremos la agenda de trabajos y tiempos empleados, para ello se ha utilizado la herramienta de planificación de proyectos PROYECT de Microsoft.

En la tabla (tabla 4) se representan las tareas del proyecto, y hay que destacar que ninguna de ellas se encarga fuera del entorno de trabajo, también podemos ver en las columnas siguientes que se representan las fechas de inicio y final, así como la duración de las mismas y las tareas predecesoras.

Tabla 4. LISTA DE TAREAS

	Nombre de tarea	Duración	Comienzo	Fin	Predeces	ESTADO
1	Captura señales bioeléctricas	5 días	mié 22/09/10	mar 28/09/10		Completada
2	Descripción y viabilidad del proyecto	4 días	jue 30/09/10	lun 04/10/10	1	Completada
3	Definición de objetivos y subobjetivos	8 días	mié 06/10/10	jue 14/10/10	2	Completada
4	Planificación	1,5 días	mar 05/10/10	mié 06/10/10	9	Completada
5	Arquitectura del sistema	3 días	jue 07/10/10	jue 28/10/10	4	Retrasada
6	Definición de los módulos Sw	8 días	jue 14/10/10	sáb 23/10/10	4	Retrasada
7	Incidencias y plan de contingencias	3 días	vie 01/10/10	mar 09/11/10		Completada
8	Calendario y fechas de entrega	2,5 días	jue 30/09/10	dom 03/10/10	1	Completada
9	Diagrama de Gannt	2 días	dom 03/10/10	lun 04/10/10		Completada
10	Material necesario	2 días	dom 03/10/10	mar 23/11/10	8	Completada
11	PAC1 Plan de Trabajo	6 días	mié 29/09/10	mar 05/10/10	1	Completada
12	Definición del modelo matemático	2 días	mié 06/10/10	jue 07/10/10	11	Completada
13	Cálculo Frec. Muestreo	3 días	vie 08/10/10	mié 27/10/10	12	Completada
14	Diseño módulo amplificador	3,5 días	vie 01/10/10	mié 06/10/10	1	Completada
15	Diseño módulo filtro y F.A.	1 día	mié 06/10/10	jue 07/10/10	14	Completada
16	Bloq. Procesador e interfaz de usuari	4 días	jue 07/10/10	mar 09/11/10	15	Completada
17	Instalación y manejo MPLAB	6,5 días	lun 18/10/10	lun 25/10/10	16	Completada
18	Programa conversor A/D	6 días	jue 28/10/10	jue 04/11/10	17	Completada
19	Programa captura de señal	3,5 días	jue 11/11/10	mar 16/11/10	18	Completada
20	Programa de alarmas	6,5 días	mié 17/11/10	lun 06/12/10	17;18;19	Completada
21	Programa principal	8 días	vie 19/11/10	mar 30/11/10	19	Completada
22	PAC2 1ª parte TRABAJO	7 días	lun 01/11/10	mar 09/11/10	13;5;3;16	Completada
23	PAC3 2ª parte TRABAJO	12 días	lun 06/12/10	mié 22/12/10	22;20;21;	Completada
24	Lliurament FINAL	16,5 días	vie 24/12/10	lun 17/01/11	22	Tarea futura
25	Instalación y manejo Sw video	4 días	mar 28/12/10	vie 31/12/10		Tarea futura
26	Depuración de errores	3 días	mié 24/11/10	vie 26/11/10	22	Completada
27	Memoria: TRABAJO FINAL	16 días	mié 22/12/10	mié 12/01/11	26	Retrasada
28	Video de presentación	15,5 días	vie 14/01/11	vie 04/02/11	23;27;25	Tarea futura
29						
30	Hitos (EN ROJO)	66,5 días	lun 01/11/10	vie 04/02/11		Retrasada

La representación gráfica de las tareas de la tabla anterior, así como sus relaciones y % de progreso la tenemos en el siguiente diagrama de Gantt fig. 6.

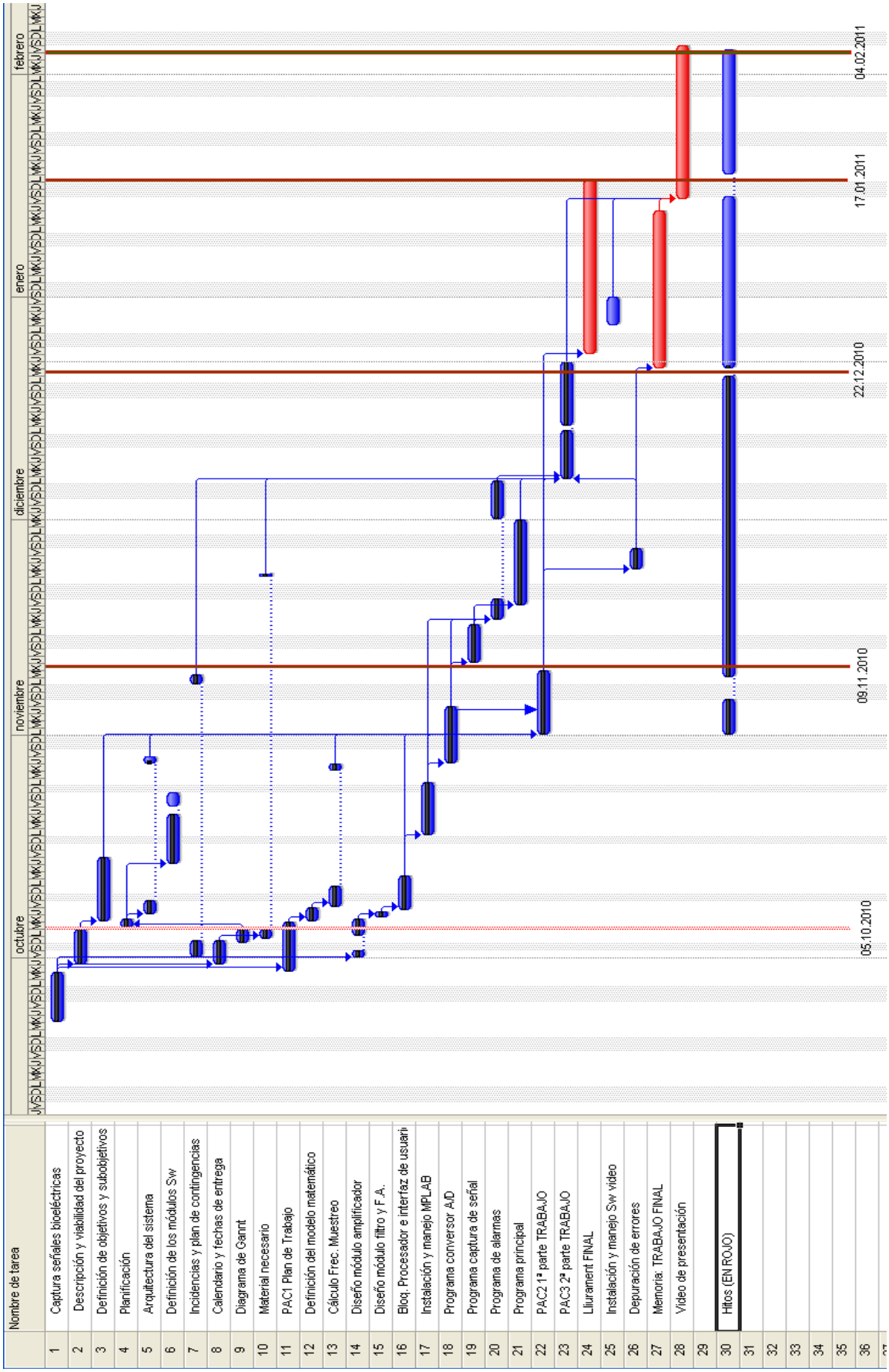


Fig. 6 Diagrama de Gantt

1.12. Cumplimiento de objetivos

En este apartado se describe el seguimiento, y el grado de cumplimiento de cada uno de los objetivos del proyecto.

Tareas objetivo 1	Estado:
- Captura señales bioeléctricas	Hecho
- Arquitectura del sistema	Retrasada, Hecho, Ok
- Material necesario	Hecho

Tareas objetivo 2	Estado:
- Diseño módulo amplificador	Hecho
- Diseño módulo filtro y F.A.	Hecho
- Bloque Procesador e interfaz de usuario	Modificado, Ok
- Instalación y manejo MPLAB	Hecho
- Programa conversor A/D	Hecho

Tareas objetivos 3 y 4	Estado:
- Definición de los módulos *Sw*	Hecho
- Definición del modelo matemático	Hecho
- Cálculo Frecuencia de Muestreo	Hecho
- Programa captura señal	Hecho
- Programa de alarmas	Hecho
- Programa principal	Hecho

Tareas comunes	Estado:
- Planificación	Modificado, Ok
- Descripción y viabilidad del proyecto	Hecho
- Definición de objetivos y subobjetivos	Modificado, Ok
- Incidencias y plan de contingencias	Hecho
- Calendario y fechas de entrega	Modificado, Ok
- Diagrama de *Gantt*	Modificado, Ok
- Instalación y manejo *Sw* de video	Hecho
- Depuración de errores	Hecho
- Repaso, mejoras y defectos, pruebas finales	Hecho
- Memoria: TRABAJO FINAL	Hecho
- Video de presentación	Hecho
- Posibles aspectos económicos y de marketing del modelo	En la última fase

1.13. Replanificación

1.13.1. Rediseño

En la simulación de los circuitos, se han observado fallos en la alimentación del amplificador de instrumentación, por lo que ha sido necesario su rediseño y vemos ahora en la gráfica de la fig. 19 que la respuesta es correcta.

1.13.2. Cambios

Se han cambiado la planificación de diseño de los módulos software de captura de señal por el de conversión A/D.

1.13.3. Retrasos

Se han retrasado las tareas de definición de la arquitectura del sistema, y el cálculo de la frecuencia de muestreo, ya que estaban supeditadas a la finalización de la tarea de

definición de los módulos software. También se ha retrasado el diseño del programa de alarmas, debido la necesidad de tener terminado el algoritmo del programa principal.

El diseño del bloque procesador e interfaz de usuario queda retrasado una semana, hasta encontrar un módulo hardware con display y teclado.

Todas las tareas se han reprogramado y su estado es de cumplimiento.

1.14. Riesgos y plan de contingencia

La fase previa de estudio ha sido crítica hasta encontrar y poder procesar las señales bioeléctricas, también es crítica la fase de tratamiento de las señales ya que hay que transformar la señal analógica original (dominio del tiempo) en información binaria que pueda ser tratada por el procesador.

Encontrar la frecuencia de muestreo de la señal analógica y el microcontrolador que mejor se adapta al proyecto, es también un aspecto de suma importancia, otro aspecto muy delicado será la identificación de patrones.

Riesgos y plan de contingencia, supondremos que si una de las fases del proyecto o la totalidad del mismo sufre retraso o imposibilidad de ejecución, habría que habilitar los medios para que esto no ocurra, pero se trata de un equipo de una persona y si contamos al director del proyecto dos personas y con fecha de entrega fija, que hacer para que todo encaje:

1.14.1. Incidencias

Se han de planificar anticipadamente y con el tiempo necesario, las posibles bajas, retrasos, coincidencia con otros asuntos o tareas ineludibles, corrigiendo éstos de inmediato con la ayuda de herramienta de proyectos PROJECT[22].

Fig. 7 Fotografía del prototipo

Para poder medir con precisión las tensiones de los pulsos eléctricos del corazón, hemos tenido que construir un amplificador de instrumentación con el circuito integrado INA114[23], esto ha retrasado unos días el proyecto, no tanto en el diseño (ver circuito anexo 1). En la figura 7 podemos ver el montaje para las pruebas.

El amplificador se alimenta con una fuente aislada continua de 9 V, (batería de 9V) se ha construido un divisor de tensión simétrico con la referencia de salida a 0V flotante, posteriormente hubo que incluir unos condensadores para estabilizar la referencia de 0V.

Con el dispositivo de la fig. 22, se ha encontrado la forma de onda de los pulsos cardíacos en el osciloscopio fig. 14, donde vemos varios períodos de la señal cardíaca que sigue la cadencia de una señal patrón capturada por un electro-cardiógrafo, semejante a la señal patrón que veíamos en la fig. 8.

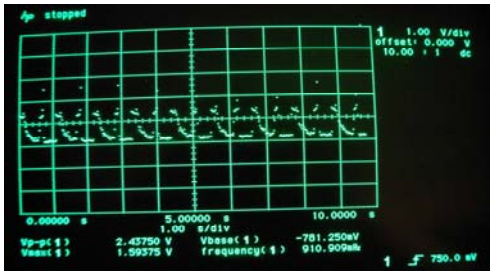


Fig. 8 Captura en el osciloscopio de la señal amplificada
EJE de ABCISAS: 1 s/div EJE de ORDENADAS 1 V /div

[22] Programa de gestión, planificación y seguimiento de proyectos

[23] PRECISION INSTRUMENTATION AMPLIFIER (amplificador de instrumentación de precisión), ver anexo 2

1.14.2. En la adquisición de conocimientos.

Adquirir los conocimientos necesarios, tanto para la resolución de los problemas que surgen en el proyecto, como para el manejo de las herramientas de diseño, es una necesidad ineludible, ya que no se trata de una empresa que pueda subcontratar los trabajos. Por tanto para esta contingencia no hay alternativa.

Cambio de µC, el componente esencial del proyecto es el µC dsPIC, el material alternativo de este chip es el modelo dsPIC30F6010A del mismo fabricante, tiene más capacidad de memoria y prestaciones de seguridad avanzadas que no vamos a usar en este proyecto.

En el caso de necesitar un convertidor A/D más rápido y externo al procesador, la alternativa es el convertidor A/D TLV1570 de Texas I. de 2.7 V a 5.5 V 8-canales de 10-BIT 1.25-MSPS (SERIAL ANALOG-TO-DIGITAL CONVERTER)

1.14.3. Software alternativo

Encontrar en caso necesario el software de diseño alternativo de los siguientes programas:

- PSPICE es un programa de diseño de circuitos, su alternativa es GNUCAP sobre Linux
- El software alternativo de SCILAB es MATLAB
- El programa MPLAB no tiene alternativa, es exclusivo para los dispositivos PIC.

1.14.4. Riesgo de pérdida de datos

Para evitar perder los ficheros de tratamiento informático del proyecto, como son: fuentes de programas, fuentes de diseño de circuitos, ficheros de simulación y prueba, fotografías, dibujos y esquemas, la documentación bibliográfica, las copias de la memoria y los vídeos de presentación, etc., se deben hacer copias de seguridad de los ficheros mencionados, al menos una vez cada semana, y siempre que el volumen de trabajo avanzado lo requiera.

1.15. Tabla 5, resumen de riesgos y contingencias

Tabla 5: Riesgos y contingencias	**Método de trabajo y alternativas**	**Solución**
Encontrar y procesar las señales bioelectricas	Trabajo de campo, pruebas y medidas	si
Identificación de patrones en la señal bioeléctrica	Identificación de picos en la señal analógica	si
Cálculo de la frecuencia de muestreo	Teorema de Shannon	si
Identificación de patrones en la señal muestreada	No forma parte del proyecto en el estado actual	
Elección del microcontrolador	dsPIC30F2010	si
Planificación con tiempo suficiente	Invertir más horas de trabajo	si
Adquirir los conocimientos necesarios	No hay alternativa	no
Búsqueda de material alternativo	µC con más memoria[24], y/o convertidor A/D externo	**probar**
Encontrar el software alternativo	Programas de libre distribución	si
	Sw de diseño MPLAB	no
Pérdida de datos	Hacer copias de seguridad semanales	si

[24] La estructura de la memoria se describe en el capítulo 2.

1.16. Planes de contingencia activados

A lo largo del proyecto, ha sido necesario retocar la planificación de unas pocas tareas, como son la definición de la arquitectura del sistema, y la definición de los módulos software, también ha sufrido retrasos el programa de alarmas, debido a la elección del interfaz de usuario.

Pero no menos importante, ha sido la aplicación del plan de contingencias, ya que se ha tenido que activar el punto 1.14.4, para prevenir la pérdida de datos, debido a que el programa de grabación de video se quedaba bloqueado, y esto hacía que se perdiesen los trabajos, sin posibilidad de poder recuperarlos.

Haciendo las copias de seguridad que se indican en el plan de contingencias, hemos conseguido salvar datos importantes, y así poder cumplir los objetivos, ya que una pérdida de datos afecta a todos los objetivos y a los plazos de entrega

1.17. Estructura y contenidos de este documento

Hasta este momento, hemos visto la descripción del proyecto, y los fundamentos necesarios que nos ayudan a entender, el origen de las señales eléctricas del corazón. También hemos definido los cinco grandes objetivos del proyecto, y las tareas necesarias para alcanzar estos objetivos.

Además hemos visto el diagrama de bloques del proyecto, y la forma de realizarlo a través de la planificación, y los planes de contingencia.

Lo que nos queda por ver, es lo siguiente:

- En el capítulo 2, vemos el diseño hardware, así, en la sección 2.5.1 diseñaremos el amplificador lineal, y en la sección 2.5.2 diseñaremos el filtro paso banda. A continuación veremos en la sección 2.6. la interfaz de usuario, que está constituida por un display de 2x16 caracterea alfanuméricos, y teclado integrado, y finalmente en la sección 2.7. veremos el diseño de las fuentes de alimentación.

- En el capítulo 3, veremos el diseño software, empezando por la selección del µC, microcontrolador (pieza clave del diseño), en la sección 3.1. Después en la sección 3.2 veremos la estructura del µC, y en las secciones posteriores veremos el diseño de los distintos módulos software, módulo principal, módulos de reloj, de exploración de señal, de conversión analógico- digital, de exploración y actuación con la interfaz de usuario, y finalmente el módulo de alarmas.

- En el capítulo 4, veremos las conclusiones de este proyecto, tanto la valoración del proyecto, como la dedicación al mismo, y las experiencias de aprendizaje que se han conseguido.

Capítulo 2. Diseño de la parte hardware

En este capítulo vamos a describir en detalle los cálculos necesarios, y cada uno de los circuitos que componen la parte hardware. Para ello, primero hay que definir la resolución con la que queremos representar las señales, tanto en amplitud como en frecuencia, y una vez obtenidos estos datos, pasar al diseño de circuitos.

2.1. Notas sobre las pruebas de detección de la actividad eléctrica del corazón

Antes de empezar el diseño hay que comprobar que efectivamente se puede detectar la señal procedente del corazón. Esto se ha hecho aplicando los electrodos del aparato medidor en el pecho de una persona, así, los resultados obtenidos confirman la existencia de una pequeña señal de aproximadamente 0.4 mV de pico a pico, y que podemos observar en las gráficas capturadas (ver figuras 9 y 10)

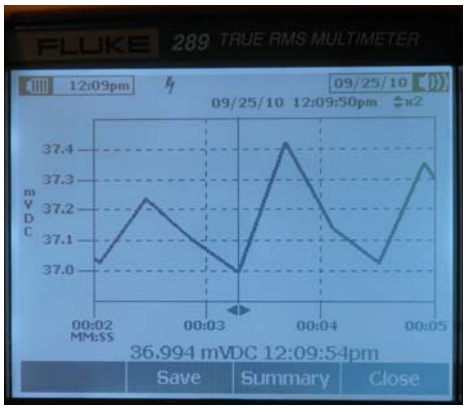


Fig. 9 captura1 FLUKE 289
EJE de ABCISAS: 1 s/div
EJE de ORDENADAS 0.1 mV/div

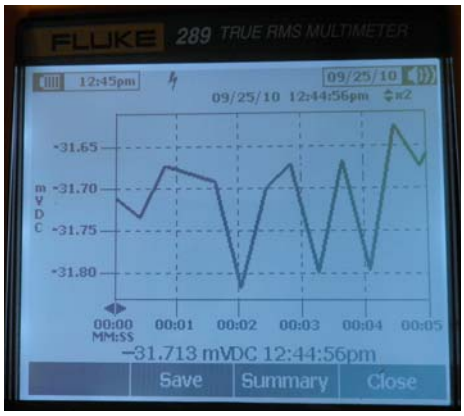


Fig. 10 captura2 FLUKE 289 EJE de ABCISAS: 1 s/div
EJE de ORDENADAS 0.5 mV/div

2.2. Material empleado para las pruebas

Las bandas que se han utilizado para esta parte del proyecto son de ciclismo, una de ellas se ha vaciado de su circuitería y batería, dejando sólo los electrodos en contacto con la piel que tienen una resistividad de 18 KΩ.cm, lo que nos da una resistencia por electrodo de 6-8 KΩ. (fig. 11), la otra banda que se utiliza, ya que viene preparada con bornes para conectar los electrodos al exterior (fig. 12) y finalmente se utiliza la banda de un pulsómetro comercial (ver figura 13)

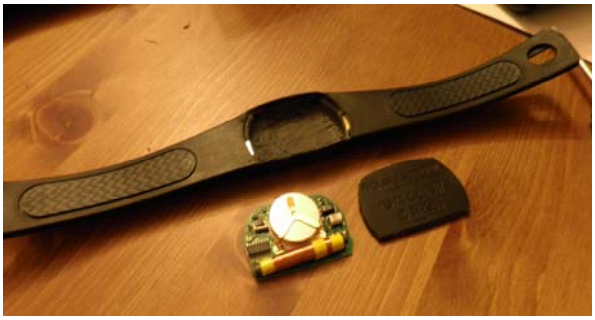


Fig. 11 banda vaciada para pruebas

Fig. 12 banda con bornes para pruebas

Fig. 13 banda comercial

Los estudios teóricos de electrocardiología nos dan una señal característica, cuyo ECG[25] básico ó función v(t) es la de la figura 14, esta gráfica es la base de un electrocardiograma real que podemos ver en el anexo 2 (se reproduce el ECG de un paciente)

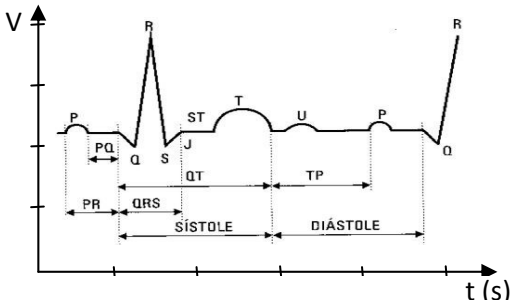


Fig. 14 señal patrón

2.3. Medidas realizadas

Fig. 15 Pulsómetro POLAR

Una vez capturadas las señales procedentes del corazón, debemos contrastar nuestras medidas con un equipo calibrado, para ello utilizamos dos bandas sensoras a la vez sobre la misma persona, una para el proyecto conectada al amplificador de instrumentación experimental construido para este objeto, y otra conectada vía radio a un pulsómetro[26] comercial de ciclismo, figura 15, que nos da las pulsaciones en tiempo real, la salida del amplificador se conecta al multímetro Fluke (aislado), en el que se observan los picos de las señales, éstos a su vez nos dan la frecuencia cardíaca (no se aprecia el oscilograma completo ya que este aparato tiene un tiempo de muestreo grande 0.5 a 0.25 segundos el más rápido), por lo que la resolución de la señal es limitada.

Para medir la frecuencia cardíaca contamos los pulsos y dividimos por el intervalo de tiempo de la ventana de muestreo y luego multiplicamos por 60, ver figura 17 de captura a 60 pulsaciones por minuto (ppm) y figura 16 de captura a 80 (ppm), coincidiendo los cálculos con las medidas observadas en el pulsómetro comercial de referencia.

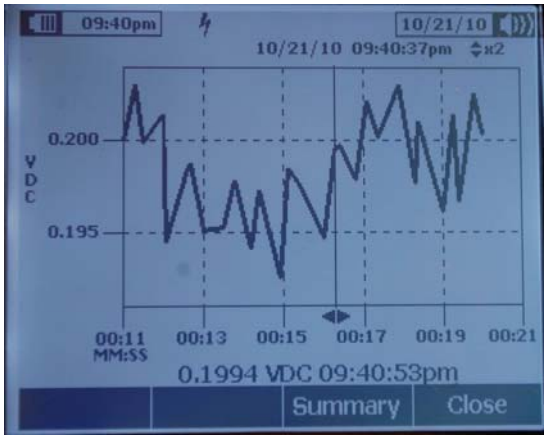


Fig. 17 captura a 80 ppm
[eje X: 2s/div | eje Y: 5 mV/div]
- caso 2: Después del ejercicio
  12 impulsos/9 seg = 1.33Hz
  60*1.33Hz = 80 ppm

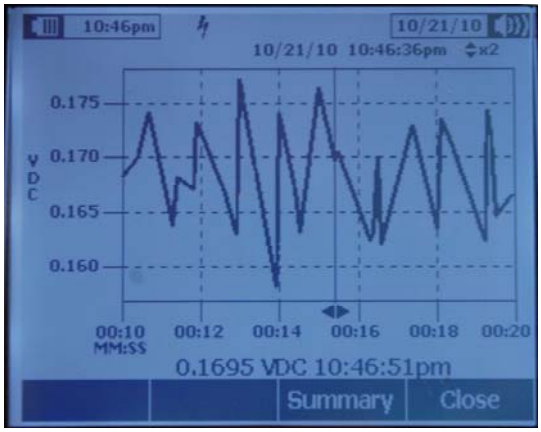


Fig. 16 captura a 60 ppm
[eje X: 2s/div | eje Y: 5 mV/div]
- caso 1: Reposo
  10 impulsos/10 seg = 1Hz
  60*1Hz = 60 ppm

[25] ECG = Electrocardiograma

[26] Pulsómetro o pulsímetro es un instrumento que mide la frecuencia cardíaca, se usa en varios deportes para controlar la capacidad del corazón en el esfuerzo.

Por tanto, el método de medida es adecuado y por ahora sólo vamos a usar el pulso del latido (los picos que aparecen en las gráficas), si se requieren nuevas prestaciones no incluidas en este proyecto, la forma de la señal se muestreará completamente de acuerdo a la ley de muestreo o Teorema de *Nyquist-Shannon* [27], esto se acometería con otras herramientas.

2.4. Cálculo de la resolución de amplitud y frecuencia de muestreo de la señal

Determinar la resolución en amplitud y la frecuencia mínima de muestreo del convertidor A/D, es uno de los factores decisivos en la elección del hardware, en concreto hay que encontrar el número de bits del Convertidor A/D y la frecuencia mínima de muestreo, para que en la conversión no se pierda información significativa.

- La resolución del convertidor debe permitir trocear la amplitud máxima de la señal en 1024 partes, tanto para las necesidades actuales del proyecto (ahora más que suficiente) como para las futuras, esto quiere decir que 2^N= 1024, por tanto N= 10 bits de resolución

- En el cálculo de la frecuencia de muestreo debemos tener en cuenta los casos más extremos de arritmia cardíaca, estos son la Fibrilación Auricular (FA) con frecuencias de 160 a 180 latidos por minuto (lpm) y la fibrilación ventricular (FV) afección muy grave que llega a frecuencias de 300 lpm.
La frecuencia de muestreo viene determinada por el intervalo de tiempo más pequeña, que necesita el convertidor para una determinada velocidad de cambio de la señal analógica, como hemos definido antes la muestra se ha troceado en 2^{10} partes en amplitud, entonces para que la conversión sea lineal necesitaremos la misma proporción en el tiempo, si como hemos hallado la frecuencia de latido más alta es de 300 lpm, es decir 5 Hz esto nos da un período por latido de 0.2 segundos. De aquí obtenemos la ranura de tiempo mínima

$t_{mín.} = 0.2/2^{10} = 195\ \mu s \sim 200\ \mu s \rightarrow$ FM (Frecuencia de Muestreo) ≥ 5 KHz [10]

De aquí podemos deducir de acuerdo al Teorema de *Nyquist-Shannon* que la máxima frecuencia que podemos reconstruir es la mitad de la frecuencia de muestro Fs (*Sampling Frecuency*) es, Fs= 2B, siendo B la frecuencia de la señal muestreada:

$$B = \frac{Fs}{2} = \frac{5KHz}{2} = 2.5KHz \qquad [11]$$

Con las pruebas descritas y efectuadas en las secciones 2.1, 2.2 y 2.3, así como los cálculos realizados en la sección anterior 2.5, cumplimos el objetivo 1.

2.5. Diseño de circuitos.

En esta sección vamos a diseñar y probar los circuitos analógicos, que son el filtro y el amplificador de instrumentación. Diseño del amplificador de instrumentación

2.5.1. Diseño del amplificador lineal

El módulo amplificador está construido por un circuito integrado, que contiene tres amplificadores de precisión de alta impedancia de entrada, y resistencias configuradas como divisores de tensión. El circuito empleado es el INA114 (ver sección 1.13.1) que ha dado buenos resultados en las pruebas realizadas.

[27] Nyquist y Claude E. Shannon establecen que la frecuencia mínima de muestreo necesaria para evitar el "*aliasing*" (imposibilidad de reconstrucción digital) debe ser. $f_m > 2.BW$ (La demostración se puede ver en el Anexo 8)

Podemos ver el esquema completo en el anexo1, y en la figura 18 vemos el esquema del circuito integrado INA114. La ganancia total en nuestro caso se obtiene poniendo la entrada V_2 a 0 V, ver figura 18 y fórmula [13], para el cálculo se pueden consultar las referencias[28].

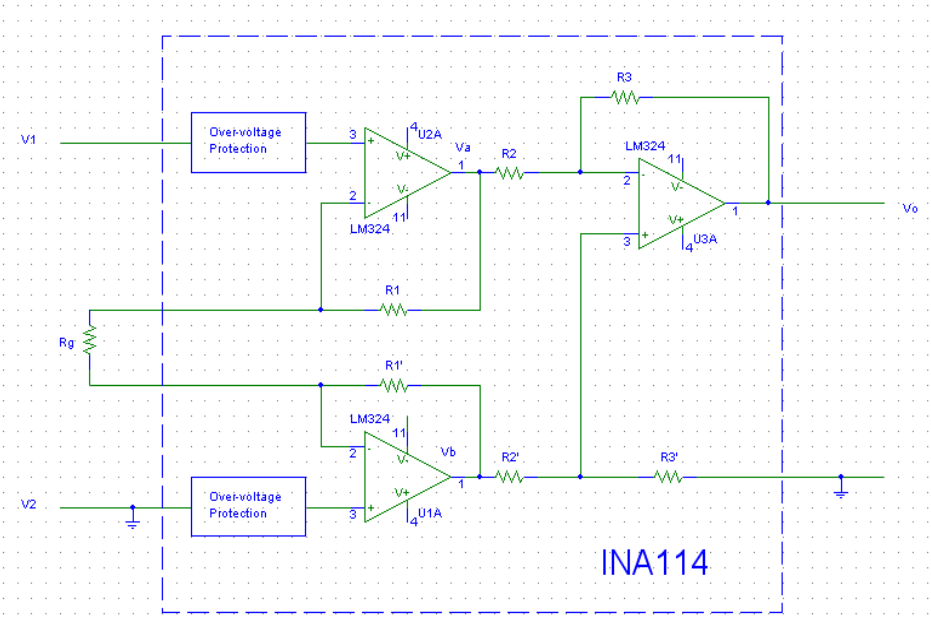


Fig. 18 Amplificador integrado INA114

Las fórmulas para hallar la ganancia, son:

$$\text{Dado que}: R_2 = R'_2 \text{ y } R_3 = R'_3$$

$$V_0 = -V_a\left(\frac{R_3}{R_2}\right) + Vb\left(\frac{R_2 + R_3}{R'_2 + R_2}\right)\frac{R'_3}{R_2} = \frac{R_3(V_b - V_a)}{R_2} \quad [10]$$

$$V_a = V_1\left(1 + \frac{R_1}{R_G}\right) - V_2\left(\frac{R_3}{R_2}\right) \quad [11]$$

$$V_b = -V_1\left(\frac{R'_1}{R_G}\right) - V_2\left(1 + \frac{R'_1}{R_G}\right) \quad [12]$$

$$\text{La ganancia total del amplificador es}: \; Gd = G_{d1} \cdot G_{d2} = \frac{R_3}{R_2}\left(1 + 2\frac{R_1}{R_G}\right) \quad [13]$$

Siendo:

Vo = tensión de entrada al amplificador

Va = tensión de entrada (-) al operacional U3A

Vb = tensión de entrada (+) al amplificador U3A

V1 = tensión de salida del amplificador U2A

V2 = tensión de salida del amplificador U1A

RG = Resistencia de control de ganancia

Gd = Ganancia total del amplificador

Gd1 = Ganancia de la primera etapa formada por U1a y U2A

Gd2 = Ganancia de la segunda etapa formada por U3A

[28] Amplificadores operacionales y circuitos integrados lineales, de *Robert F. Coughlin*, e Instrumentación Electrónica, de Miguel A Pérez (consultar bibliografía).

Las curvas de GANANCIA del amplificador en la simulación del circuito en PSPICE, se representan en la fig. 19, y en la función de transferencia se puede observar que no hay distorsión de amplitud, en la zona de trabajo (+/- 1mV de V(in)) y hasta los +/- 4 mV de V(in).

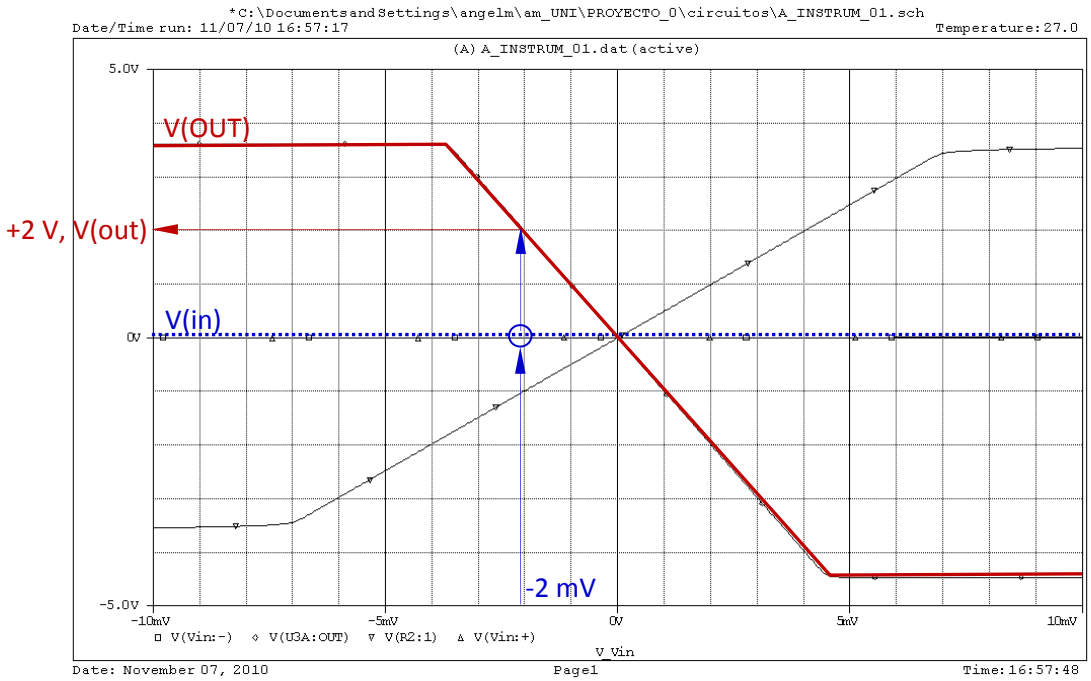


Fig. 19 Ganancia amplificador INA114; G= -V(OUT)/V(in) [Rg = 50Ω,→ G= -1000]

Esta es la respuesta que necesitamos, como se ha indicado en el objetivo 3 del punto 1.4: Buscamos una muestra sin perder representatividad, y puesto que no queremos modificar la señal, esta sólo es ampliada para luego ser capturada por el convertidor A/D.

2.5.2. Diseño del filtro paso banda

Recordemos que este diseño ha de cumplir una función primordial, que consiste en dejar pasar sólo los valores de las frecuencias centrales, que caracterizan un período del pulso cardiaco. Estos valores están de acuerdo a los datos del diagrama espectral de una señal ECG, figura 20, capturada por tres electrodos alrededor del corazón según el triángulo de *Einthoven*[29] (en el proyecto sólo se usan dos electrodos), a cada una de las fases de este triángulo se las denomina Q, R y S.

Fig. 20 Electrocardiograma real, divisiones en mm, ordenadas 0.1 mV/mm, abcisas 40 ms/mm fuente: APUNTES DE ELECTROCARDIOGRAFÍA

[29] Método del Fisiólogo holandés *Willem Einthoven*, que en 1903 desarrolló el galvanómetro que lleva su nombre, gracias al cual logró medir las diferencias de potencial eléctrico experimentadas por el corazón

En la figura 21, podemos observar el diagrama espectral del pulso cardíaco, con todas sus componentes en frecuencia, en este diagrama tenemos una frecuencia central entorno a los 10 ó 12 Hz, tal como se ha calculado en el anexo 10, al hallar la FFT de la señal definida en el modelo matemático (ver anexo 10) de la señal cardíaca.

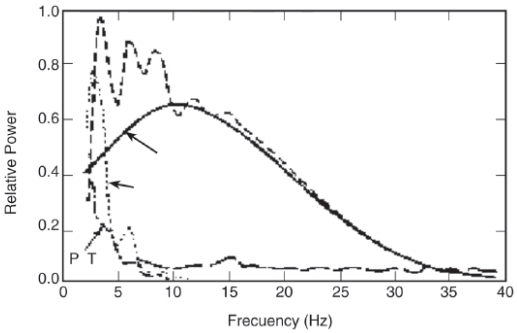


Fig. 21 Diagrama espectral de una señal ECG
De tres electrodos QRS y de algunos orígenes de ruido que afectan a la señal, fuente: DESARROLLO DE UN SISTEMA DE ADQUISICIÓN Y TRATAMIENTO DE SEÑALES ELECTROCARDIOGRÁFICAS

Las ecuaciones para el cálculo de los filtros del proyecto, son:

$$Orden\ del\ \ filtro \quad N > \left| \frac{\log\left(\frac{Af^2}{g_2^2} - 1\right)}{2\log\left(\frac{\varpi_0}{\varpi_H}\right)} \right| \quad [12]$$

siendo: Af= 1, Ganancia en la banda de paso
$g_2= 10^{-1}$, Ganancia de la respuesta a una frecuencia dada
$w_0= 2.\pi.18$ rad/s, Frecuencia central en rad./s
$w_H= 2.\pi.40$ rad/s, Frecuencia de corte a 3 dB (bandas de frecuencia) en rad./s

Para estos valores tenemos: N > |2/(-0.69)|= 2.88, por tanto N=3, y esto quiere decir que necesitamos un filtro de orden 3, y este tipo de filtro se puede implementar con el modelo Sallen-Key de tercer orden.

Vamos a realizar el diseño de un filtro paso banda activo[30], que debe estar constituido por un filtro paso alto en serie con un filtro paso bajo, ambos de 3º orden del tipo *Salley-Key Butterworth*, y de acuerdo a los cálculos efectuados con la ecuación [12], en la figura 22 tenemos su estructura funcional y las fórmulas [15] y [16] para hallar el valor de los componentes, el esquema completo está en el anexo 2.

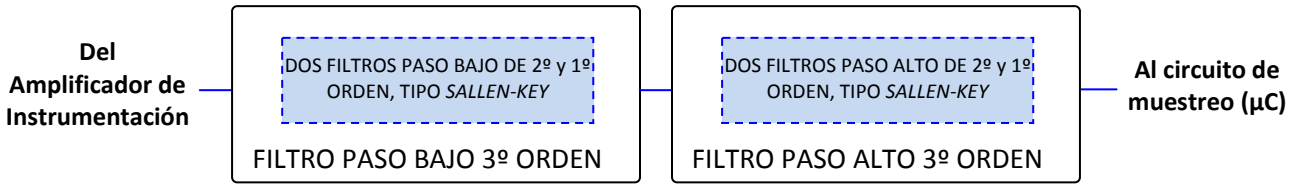


Fig. 22 Diagrama de bloques del filtro Paso Banda

$$F_0 = \frac{\varpi_0}{2\pi} = \frac{1}{2\pi RC} \quad [15]$$

siendo: F_0= Frecuencia central en Hz
R = Resistencia en Ω
C = Capacidad en Faradios

$$Q = \frac{1}{3 - Af}; \qquad Q = \frac{1}{2}\sqrt{\frac{C_1}{C_2}}, \quad [16]$$

[Q es el factor de calidad del filtro]

[30] Filtro con uno o más amplificadores operacionales, tiene mejores características en frecuencia que un filtro pasivo, es más fácil de construir y no suele utilizar inductores.

Para el cálculo de filtros *Sallen-Key Butterworth*, se puede consultar el libro de Análisis de circuitos lineales, de Francisco López Ferreras (ver bibliografía), si bien nosotros lo hemos hecho a mano con el programa PSPICE, es decir: Una vez hallados los valores teóricos con las fórmulas [15] y [16], los hemos introducido en la simulación PSPICE, retocando éstos hasta obtener las prestaciones deseadas. En el anexo 2 se pueden ver dichos valores y las respuestas en frecuencia se aprecian en las figura 23, donde se representan las curvas de atenuación/frecuencia para cada filtro.

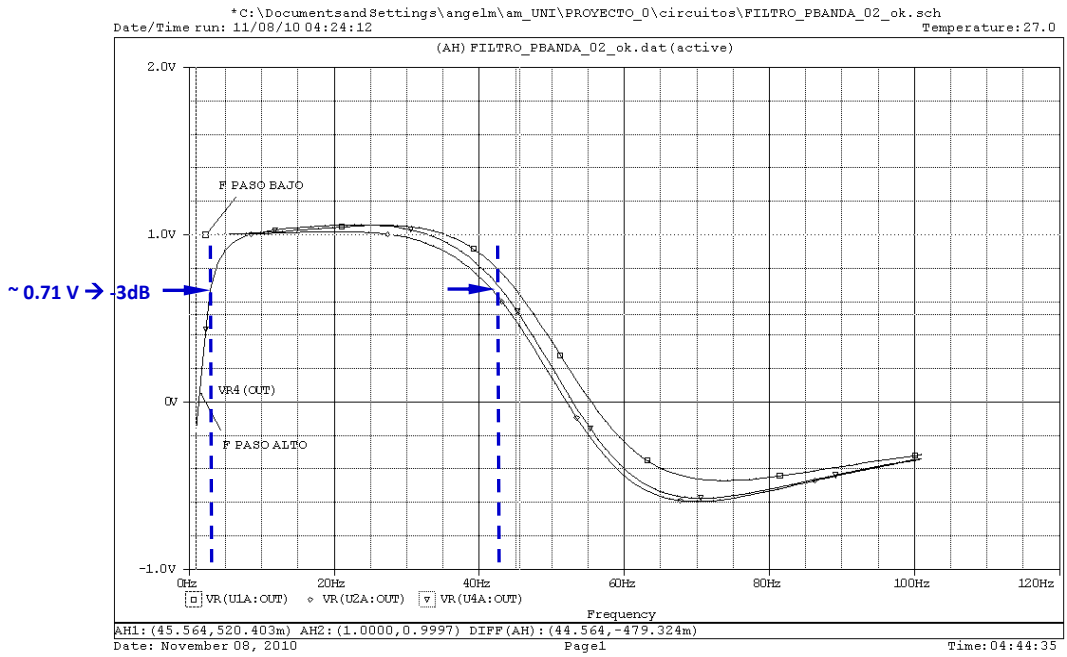


Fig. 23 Curvas de atenuación/frecuencia en la simulación del filtro PASO BANDA, escala lineal

Este es el resultado esperado, ya que necesitábamos un filtro, que en las frecuencias de trabajo de 5 Hz a 45 Hz, no tuviese atenuación (Ganancia = 1), y que en las bandas laterales tuviera una atenuación mínima de -3 dB, lo cual se cumple para el orden del filtro, en las frecuencias indicadas. Como se puede apreciar en la figura 23, ya que la ganancia en este rango de frecuencias es prácticamente 1.

2.6. Interfaz de usuario y módulo de alarmas

EL interfaz de usuario está formado por una pantalla retroiluminada ó *display* de 16 x 2 caracteres alfanuméricos, y seis pulsadores.

Tanto la pantalla de visualización, los pulsadores y la circuitería de control, están integrados en una placa de circuito impreso, con taladros de fijación (ver figura 24).

Este interfaz sirve para visualizar los siguientes datos:

- Estado de funcionamiento del procesador
- Latidos por minuto registrados por el sensor
- Alarmas de bradicardia ó taquicardia
- Valores de programación de los umbrales de alarma

Además, a través del interfaz de usuario, se puede registrar el estado de los pulsadores:

- Pulsadores de posición, arriba, abajo, derecha, e izquierda
- Pulsador central de validación ó aceptación.
- Pulsador lateral de cancelación ó borrado.

Es un dispositivo de bajo consumo que se alimenta desde 3.3 V hasta 5.5 V, y el precio por unidad es de aproximadamente 40.2 €

Fig. 24 Interfaz de usuario CFA533-TMI-KC del fabricante *Crystalfontz America, Incorporated*

La conexión de este interfaz con el procesador, se hace a través del bus *Inter-Integrated Circuit* ó más comúnmente I2C, éste es un bus serie asíncrono que dispone de reloj interno y puede funcionar como maestro si es el que inicia la conexión ó como esclavo, si es el que recibe la conexión. En la figura 25 podemos ver el esquema de conexión para la transmisión de datos.

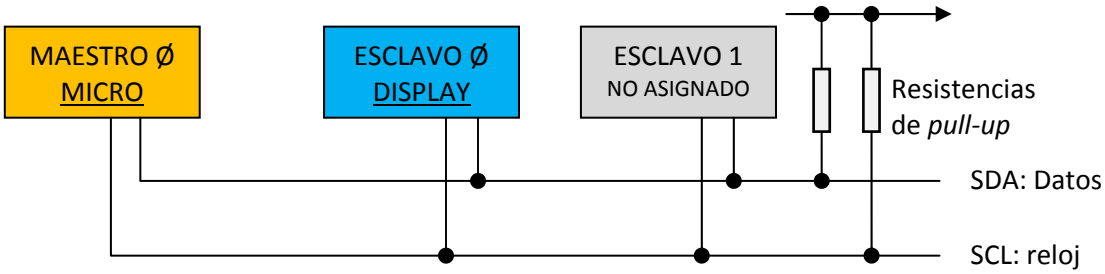


Fig. 25 Esquema de conexión del bus I2C, fuente: propia

El protocolo de transmisión se inicia cuando el maestro pone el reloj SCL a nivel alto y la señal de datos SDA pasa a nivel bajo (S, *start*), a continuación se transmiten los bits de datos ó direcciones en forma serie, empezando por el bit más significativo MSB (*Most Significant Bit*), siendo éstos bits válidos sólo cuando se produce la transición del reloj a nivel alto, una vez transmitidos los 8 bits de la palabra de datos/direcciones, en el pulso 9 del reloj se produce el reconocimiento de la información recibida por el esclavo ó *aknowledge* (ack), finalmente se pone la señal de reloj a nivel alto durante un ciclo de reloj, lo que produce la parada P, ó stop. Ver figura 26.

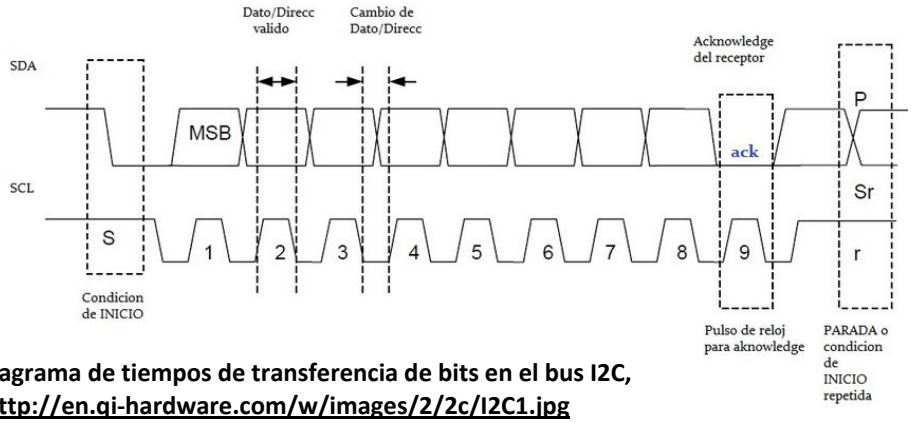


Fig. 26 Diagrama de tiempos de transferencia de bits en el bus I2C, fuente: http://en.qi-hardware.com/w/images/2/2c/I2C1.jpg

En el anexo 9, podemos ver el programa para el dsPIC30Fx, de manejo y pruebas de este interfaz de usuario.

2.7. Fuente de alimentación

El circuito de alimentación requiere el diseño de dos fuentes de alimentación conmutadas, éstas tienen un rendimiento mejor que el 90% y requieren sólo una fuente primaria de 3.6 V a 5 V, vemos el diagrama de bloques en la figura 27.

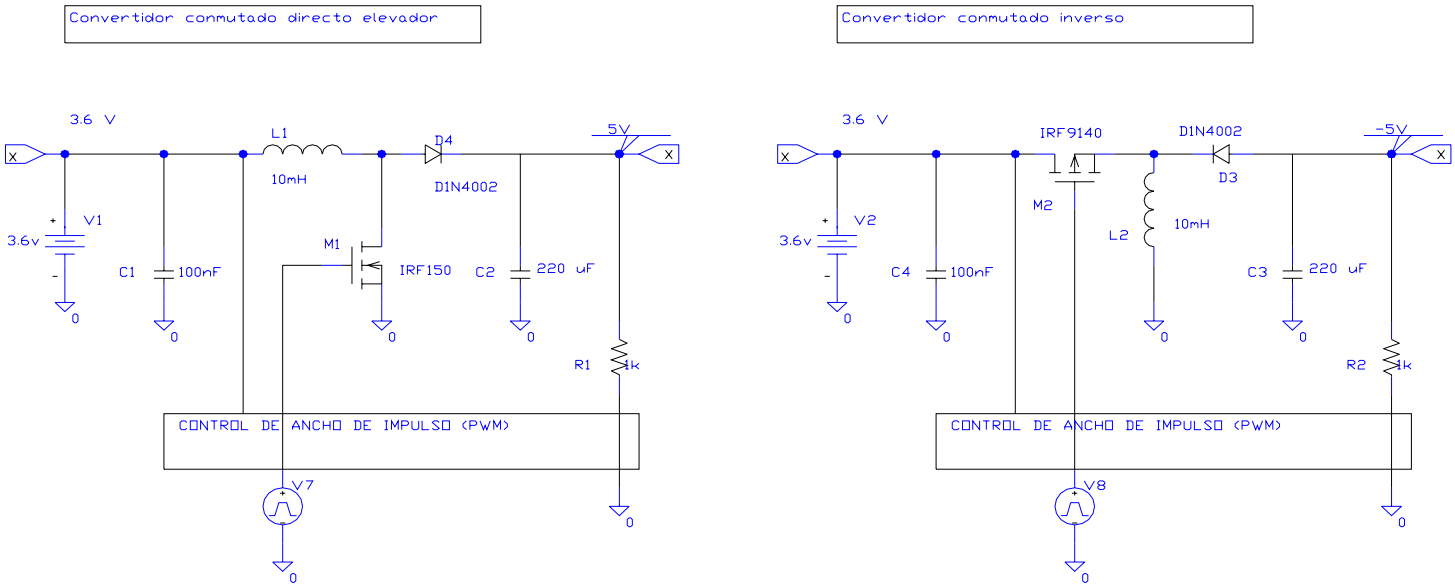


Fig. 27 Diagramas de bloques de las fuentes conmutadas.

El convertidor directo ó elevador, funciona almacenando la energía de la fuente primaria de 3.6 V en la inductancia L1, esto sucede en los semiciclos en los que el conmutador M1 está en saturación (cerrado). En el siguiente semiciclo el conmutador pasa al corte (abierto), y la tensión de la bobina L1 se suma a la fuente primaria, a través del diodo D1 alimentando la carga de +5V, el diodo impide el paso de corriente inversa.

El convertidor inverso ó *Buck-Boost,* el funcionamiento se realiza cargando la bobina L2 en el semiciclo en el que el conmutador M2 está en saturación (conduciendo), el diodo D2 impide el paso de corriente hacia la carga, pero en el semiciclo siguiente, el conmutador se corta y abre el circuito de la fuente primaria, y la inductancia L2 se descarga en sentido inverso alimentando la carga con tensión negativa.

En ambos circuitos, la regulación de la tensión de salida, se hace desde su respectivo bloque de control de ancho de impulso, siendo la frecuencia de trabajo de los conmutadores de 200 KHz. En este proyecto el control de las fuentes se puede desde el módulo PWM del µC, pero para ahorrar tiempo hemos decidido un control simple basado en los temporizadores NE555, ver bibliografía[31]

Co el trabajo realizado en las secciones precedentes, 2.5, 2.6, y 2.7 cumplimos el segundo objetivo.

[31] 555 *Timer Applications Sourcebook, with Experiments (English) ISBN: 9780672215384*

Capítulo 3. Diseño software

Hasta aquí hemos definido la parte hardware analógica, que como se indica en el capítulo 2, está formada por los módulos analógicos y los módulos digitales.

En este capítulo vamos a diseñar la parte digital y los programas que hacen funcionar correctamente cada módulo del sensor, desde el reconocimiento y muestreo de la señal analógica, pasando por el procesamiento de ésta, la interpretación de las órdenes del usuario, y la presentación de las respuestas y alarmas en el interfaz de usuario.

En primer lugar discutiremos la selección del microprocesador, que es la pieza clave del diseño e incorpora dentro del chip, el convertidos A/D, los temporizadores, y el procesador digital DSP. A continuación desarrollamos varios programas, el programa principal ó *SCHEDULER,* y los programas encargados de manejar los módulos: módulo convertidor A/D, módulo de exploración de señal, módulo de exploración-actuación y finalmente el módulo de alarmas.

3.1. Selección del µC (microcontrolador)

Como ya se ha indicado en el apartado 1.6 de viabilidad técnica, son necesarios los siguientes requisitos, ver tabla 6:

Tabla 6: Requisitos del sistema	**Observaciones:**
Capacidad de cálculo de la CPU = 16 bits	Para este proyecto no es absolutamente necesario, Pero si es necesario en las ampliaciones.
Convertidor A/D con resolución mínima de 2^{10} = 1024	Tramos lineales, y debe estar integrado en el µC
Frecuencia de muestreo máxima de 5 Ksps[32]	El modelo dsPIC 30F2010 llega hasta 100 Ksps
Hasta 12 KB de memoria de programa, 512 Bytes de RAM (Memoria de Acceso Aleatorio) y Procesador Digital de Señal ó DSP integrado	Memoria suficiente y DSP integrado en el modelo dsPIC 30F2010 ó como alternativa el dsPIC 30F6010A que ofrece características mejoradas. (Es el más alto de la gama)
Precio por pieza menor de 5 €	Una pieza 4.41 € (cambio 1 Euro = 1,3188 Dólares) (para pedidos de más de 100 piezas 1.84 €/pieza)
REF.: 2, Fuente: http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=8182&mid=14&lang=en&pageId=75	

Las características del dsPIC 30F2010, son (ver tabla 7): Tabla 7: Características dsPIC30F2010

16 bits bus de datos	Procesador digital de señal DSP de 40 bits
30 MIPS velocidad de la CPU	Convertidor A/D 6x10 bits (Velocidad de conversión mín. a 2.7 V) = 100 Ksps
12 KB memoria flash	Control de modulación de ancho de pulso (PWM)
512 Bytes RAM	Temporizadores 3 de 16 bits y 1 de 32 bits
2.5 a 5.5 V alimentación	Puertos paralelo GPIO
Oscilador interno 7.37 MHz, 512 KHz	No tiene reloj de tiempo real (RTCC) ni acceso directo a memoria (DMA)
Tecnología CMOS[33] de bajo consumo	Su precio es de 4.41 €/pieza, en fábrica, y de 1.84 € para más de 100 piezas.
Puertos: 1 UART, 1 SPI y 1 I2C	Fabricante *Microchip Technology*

[32] Kilo *samples per second* (Kilo muestras por segundo)
[33] *Complementary metal-oxide-semiconductor*

3.2. Bloque microprocesador (microcontrolador)

El microcontrolador dsPIC30F2010, tiene una arquitectura RISC (*Reduced Instruction Set)*, ó de juego de instrucciones reducido, pero esto no le resta potencia ya que cada instrucción se ejecutan en un solo ciclo de reloj[34]. En su arquitectura destacan los bloques de la CPU ó Unidad Central de Proceso, el DSP ó Procesador digital de Señal, la memoria RAM ó *Random Access Mémory* donde residen los datos y la EPROM ó *Erasable Programmable Read Only Memory* donde residen los programas, los periféricos de los cuales destacamos el convertidor A/D que usamos para obtener muestras digitales de la señal analógica y el controlador de bus I2C para comunicaciones, y además los temporizadores para el control del tiempo.

Además están los puertos de entrada salida, son cinco puertos bidireccionales agrupados por funciones, en total suman 20 líneas para comunicación con el exterior, todo esto lo vemos en el diagrama simplificado de la figura 28.

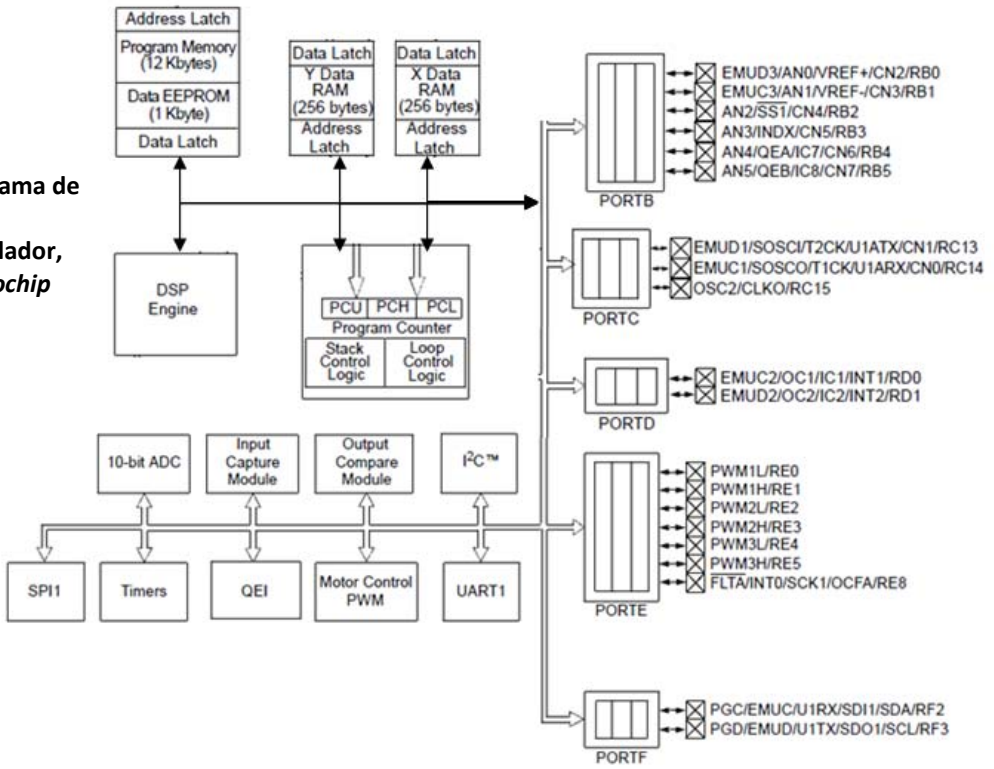


Fig. 28 Diagrama de bloques del microcontrolador, fuente *Microchip Technology*

El fabricante dispone de amplia documentación, así como un completo sistema de desarrollo, el MPLAB de *Microchip Technology*, que incorpora numerosas herramientas de diseño software tanto en ensamblador como en C, bibliotecas de programas y notas de diseño, así como simuladores de los distintos modelos. Existen numerosos kits de montaje del fabricante y de otras empresas, para desarrollo, pruebas, ó enseñanza.

[34] Excepto las instrucciones de salto y las de manejo de pila ó *stack*

El mapa de memoria de este microcontrolador, se muestra en la figura 29.

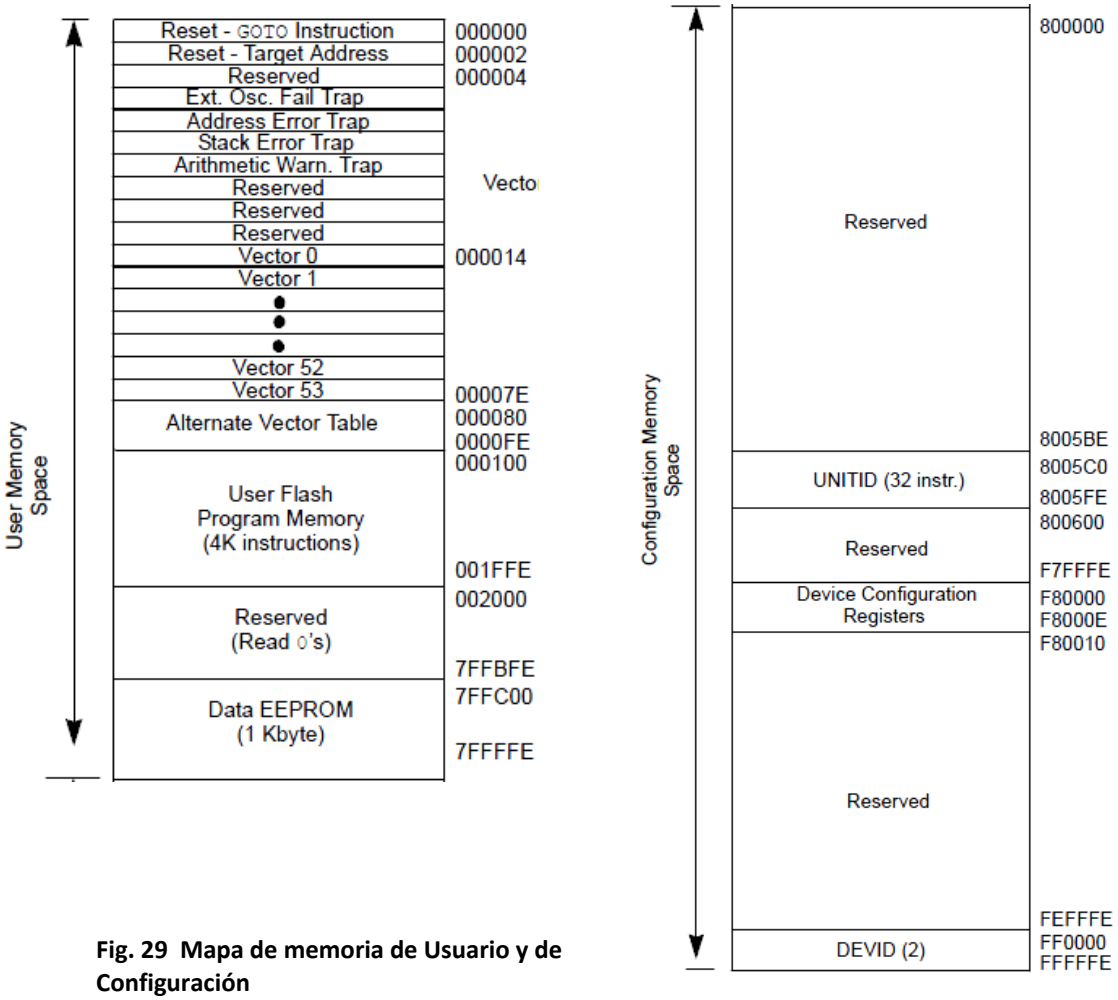


Fig. 29 Mapa de memoria de Usuario y de Configuración

Todos los puertos de entrada/salida, PORTB, PORTC, PORTD, PORTE, y PORTF, tienen direcciones en el mapa de memoria, lo vemos en la columna de *Addr*. de la tabla 8

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISB	02C6	—	—	—	—	—	—	—	—	—	—	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	0000 0000 0011 1111
PORTB	02C8	—	—	—	—	—	—	—	—	—	—	RB5	RB4	RB3	RB2	RB1	RB0	0000 0000 0000 0000
LATB	02CA	—	—	—	—	—	—	—	—	—	—	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000 0000 0000 0000
TRISC	02CC	TRISC15	TRISC14	TRISC13	—	—	—	—	—	—	—	—	—	—	—	—	—	1110 0000 0000 0000
PORTC	02CE	RC15	RC14	RC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
LATC	02D0	LATC15	LATC14	LATC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
TRISD	02D2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TRISD1	TRISD0	0000 0000 0000 0111
PORTD	02D4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	RD1	RD0	0000 0000 0000 0000
LATD	02D6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	LATD1	LATD0	0000 0000 0000 0000
TRISE	02D8	—	—	—	—	—	—	—	TRISE8	—	—	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	0000 0001 0011 1111
PORTE	02DA	—	—	—	—	—	—	—	RE8	—	—	RE5	RE4	RE3	RE2	RE1	RE0	0000 0000 0000 0000
LATE	02DC	—	—	—	—	—	—	—	LATE8	—	—	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0	0000 0000 0000 0000
TRISF	02DE	—	—	—	—	—	—	—	—	—	—	—	—	TRISF3	TRISF2	—	—	0000 0000 0000 1100
PORTF	02E0	—	—	—	—	—	—	—	—	—	—	—	—	RF3	RF2	—	—	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	—	—	—	—	—	LATF3	LATF2	—	—	0000 0000 0000 0000

Legend: — = unimplemented bit, read as '0'
Note: Refer to "*dsPIC30F Family Reference Manual*" (DS70046) for descriptions of register bit fields.

Tabla 8 Direcciones de los puertos de entrada y salida del convertidor A/D

A continuación se encuentra la descripción de cada una de las partes que componen el sistema principal. Para cada una, se presenta su diagrama de flujo así como una explicación de su funcionamiento básico.

3.3. Módulo principal.

El módulo software principal, es el programa encargado de ejecutar la actividad general de arranque del sistema, inicializando los módulos y periféricos, el módulo software que hace la planificación (*SCHEDULER*) de las rutinas y efectúa las tareas de control. El diagrama de flujo responde a esta estructura, ver figura 30.

Los programas fuente para este procesador, se pueden escribir en lenguaje ensamblador ASSEMBLER, ó en lenguaje C, ambos cumplen todas las necesidades del proyecto, pero nosotros usaremos el lenguaje C, que normalmente es facilitado por el fabricante (aunque en nuestro caso hubo que cargar el compilador C30 de los dsPIC30 aparte), en la plataforma gratuita MPLAB.

Nosotros hemos instalado la versión de MPLAB IDE 8.60.00.00, que tiene las herramientas de edición, de ejecución y pruebas. Para probar el código se debe crear un proyecto MPLAB para el µC elegido, y una vez que tenemos el programa principal (main), los módulos cargados, y si fuesen necesarias las librerías, ya podemos compilar mediante la opción “Build All”, en la ventana de Output aparece el resultado bajo la pestaña <Build>.

Para ejecutar el programa simulado, pulsamos F9, ó Debugger→Run, y ya vemos podemos ver las posiciones de memoria del procesador en Wiew → File Register y Wiew →Watch, para ver los puertos, así podemos verificar el correcto funcionamiento del código generado.

Para grabar el µC, que se supone que está insertado en el módulo externo de grabación y pruebas, como MPLAB ICD2, ó PICKit2, debemos seleccionar esta opción en la pestaña <Debugger>. El sistema de desarrollo, buscará el chip externo y transferirá el código a la memoria flash.

Vemos en la figura 30, una captura en ejecución, donde se pueden ver las posiciones de memoria en el centro, y a la derecha el contenido de los registros (se pueden observar los valores que toma el contador de programa PC, y otros registros, en hexadecimal).

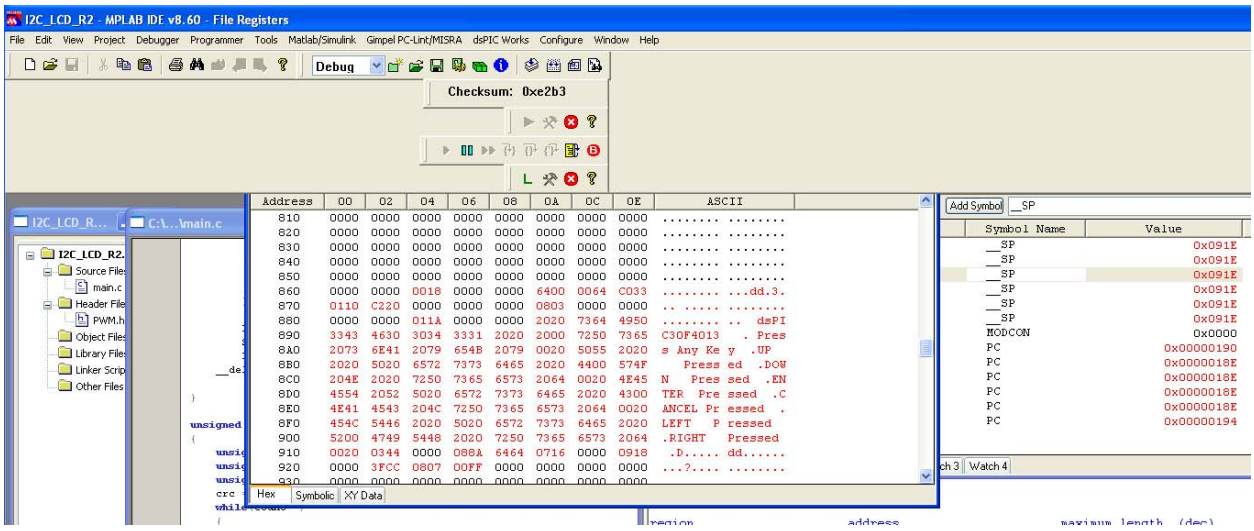
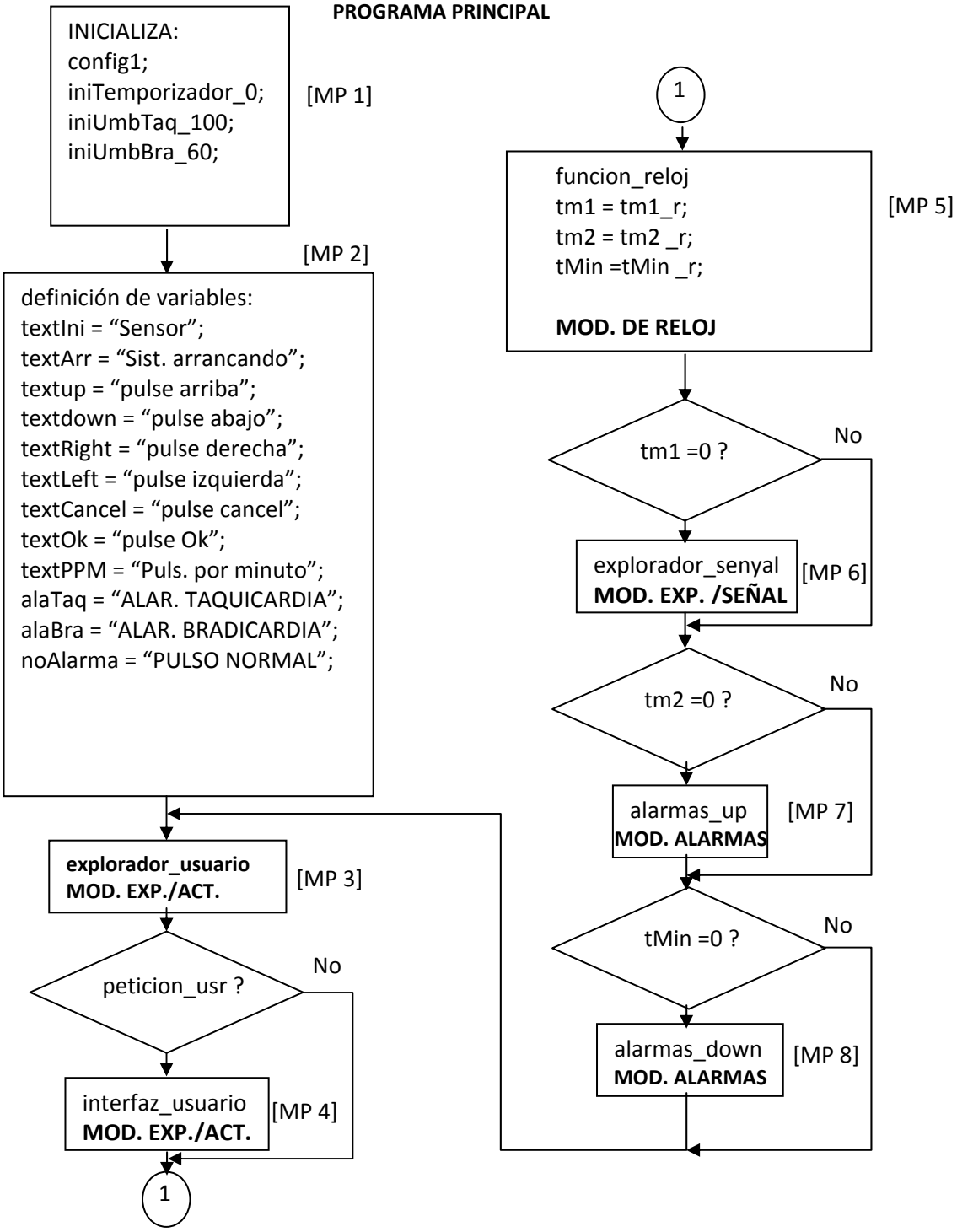


Fig. 30 Captura de la simulación del programa de prueba en MPLAB

Fig. 31 DIAGRAMA DE FLUJO PROGRAMA PRINCIPAL



Funcionamiento del programa principal:

En el bloque MP1, se ejecutan las rutinas de inicialización de periféricos:

- config1, Inicializa y configura el módulo A/D y el interfaz de usuario
- iniTemporizador_0, inicializa los temporizadores de 10 µs, 1 s, y 1 minuto
- iniUmbTaq_100, inicializa las variables de taquicardia
- iniUmbBra_60, inicializa las variables de bradicardia

El bloque MP3, llama a la rutina de exploración del teclado del usuario, si se ha pulsado una tecla da paso a la rutina de interfaz de usuario, MP4, que tratará las peticiones del usuario.

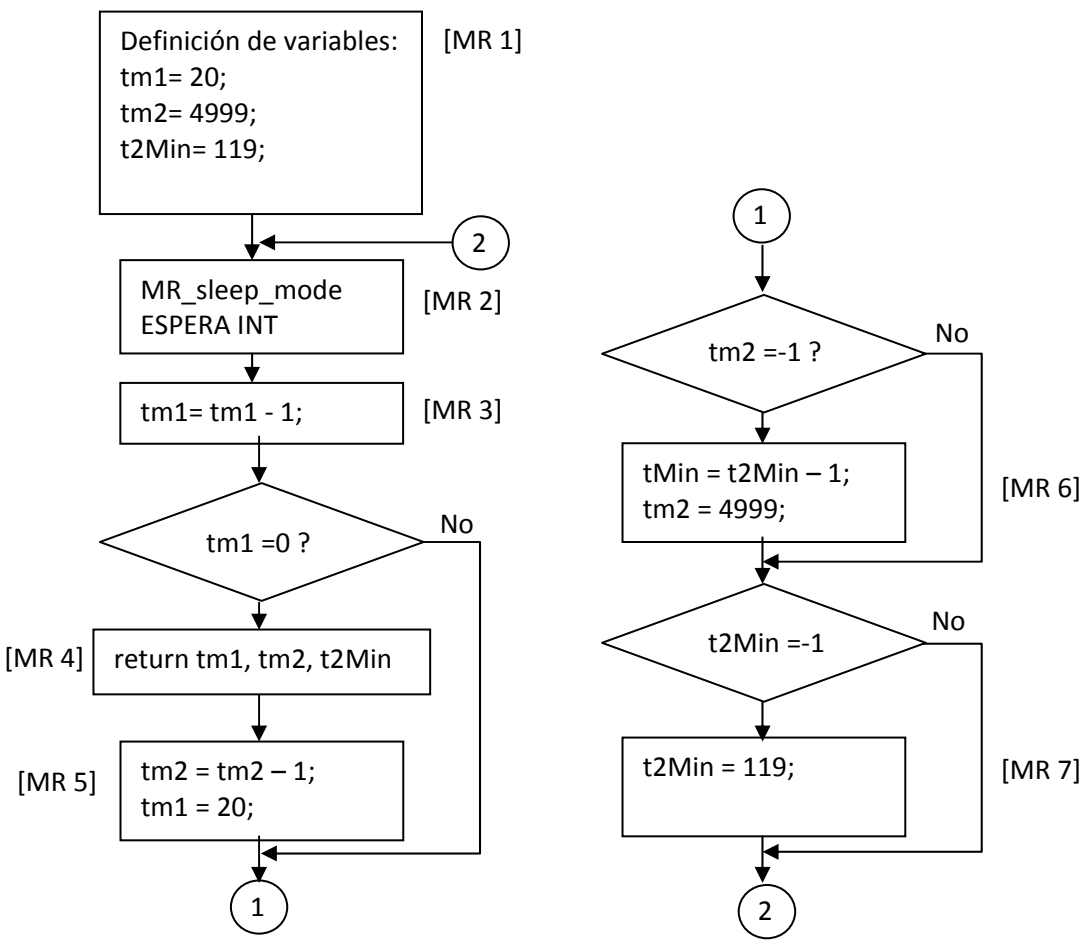
La función_reloj del bloque MP5, llama a la rutina de reloj para pedirle los relojes de tm1, tm2, y tMin, que corresponden a los tiempos de 10 µs, 1 s, y 1 minuto, respectivamente, si uno de estos telojes llega a cero, se lanza la rutina de tratamiento correspondiente:

- explorador_senyal, es la rutina de exploración de la señal procedente del convertidor A/D
- alarmas_up, es la rutina de activación de alarmas
- alarmas_down, es la rutina de cancelación de alarmas.

3.4. Módulo de reloj.

Es el encargado del subsistema de tiempos y temporizaciones, cuyo diagrama de flujo es el siguiente, ver figura 31.

Fig. 32 DIAGRAMA DE FLUJO RUTINA DE RELOJ



Funcionamiento de la rutina de reloj:

En el bloque MR_1, se inicializan las variables de la rutina, del modo siguiente:

- tm1 vale 20, como la interrupción se produce cada 10 µs, la cuenta total vale 200 µs, que es el período que necesita el sistema de muestreo del convertidor A/D.
- tm2 se inicializa a 5000 -1, para llevar la cuenta de segundos, es decir, 200 µs * 5000 = 1 s
- t2Min se inicializa a 120 -1, para llevar la cuenta de dos minutos, a partir de tm1.

En el bloque MR_2 la rutina entra en modo espera, hasta que llegue la interrupción Temporizador_0 de 10 µs

El bloque MR_3 decrementa una unidad el contador tm1, y a continuación pregunta si ha llegado a cero, si ha llegado a cero ejecuta el bloque MR_4.

MR_4 envía al programa principal los valores de los temporizadores tm1, tm2, y t2Min.

MR_5, actualiza el valor de tm1 a 20, y decrementa tm2 para obtener la cuenta de segundos.

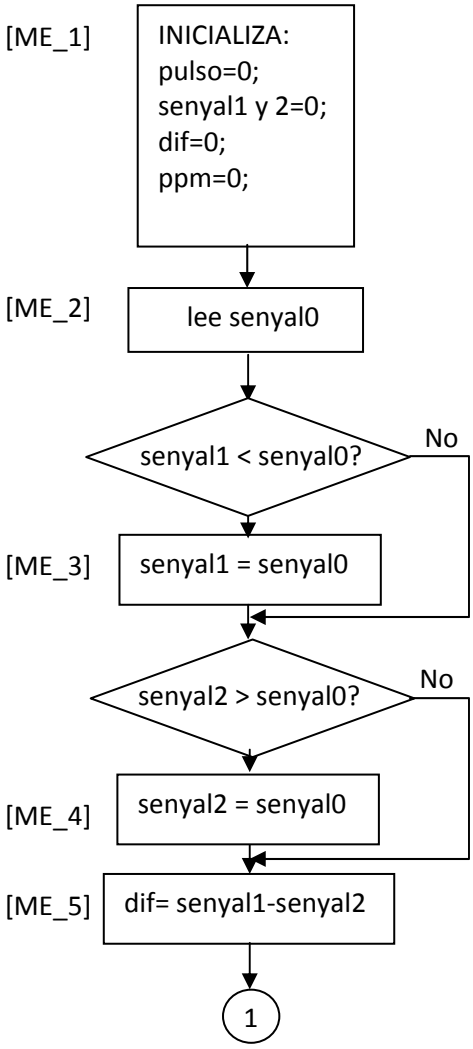
Si la cuenta de tm2 vale -1, en el bloque MR_6 se actualiza el valor de tm2, y se decrementa el contador T2Min.

Si la cuenta de t2Min vale -1, en el bloque MR_7 se actualiza el valor de t2Min, y se pasa al modo de espera en MR_2.

3.5. Módulo de exploración de señal.

El encargado de las exploraciones cíclicas de las señales provenientes de la parte analógica (figura 32), y rutina de lectura del convertidor A/D (figura 33).

Fig. 33 DIAGRAMA DE FLUJO RUTINA DE EXPLORACIÓN DE SEÑAL



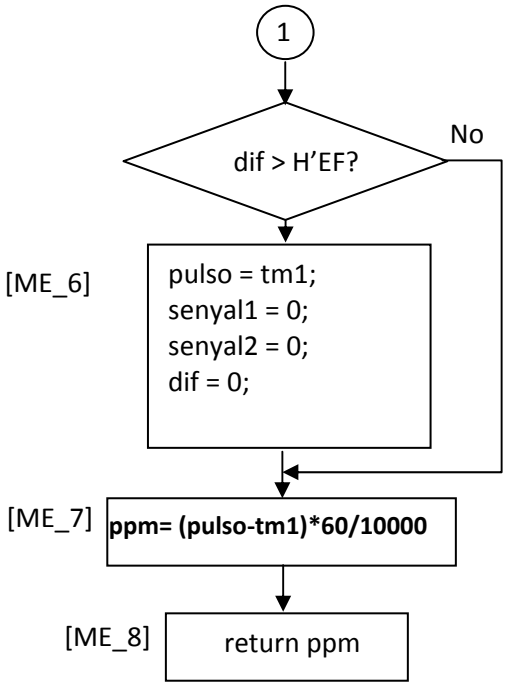
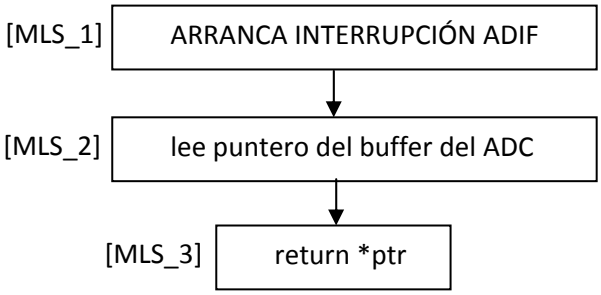


Fig. 34 DIAGRAMA DE FLUJO RUTINA DE LEER SEÑAL



Funcionamiento de la rutina de exploración de señal:

En el bloque ME_1, se inicializan las variables de la rutina, del modo siguiente:

- "pulso" es la variable que llevará la última cuenta del tiempo de un pulso, en unidades de 10 µs, se inicializa a 0.
- senyal1, almacena los valores máximo de la señal, y se inicializa a 0
- senyal2, guarda los valores mínimos de la señal, y se inicializa a 0.
- dif, es la variable que contendrá la diferencia entre los valores máximos y mínimos de la señal.
- ppm, es la variable que almacenará el cálculo de pulsaciones por minuto.

En el bloque ME_2 se lanza la rutina de lectura de la señal, el valor retornado se guarda en la variable senyal0, a continuación se pregunta si senyal1 es menor que senya0, si es cierto se pasa al bloque ME_3.

En el bloque ME_3, se guarda en la variable senyal1 el valor más grande de la serie, una vez comparadas senyal1 y senyal0, en la pregunta anterior.

A continuación se vuelve a preguntar si senyal2 es mayor que senyal0, en cuyo caso se pasa al bloque ME_4, y se guarda en la variable senyal2 el valor más pequeño de la serie, como resultado de la pregunta anterior.

En el siguiente bloque ME_%, se guarda en la variable dif, la diferencia entre senyal1 y senyal2, en la siguiente pregunta se compara la diferencia obtenida, con un valor literal, H'EF, que es el valor positivo que tiene que superar la señal para que hay un pulso, si la variable dif es mayor que H'EF, se pasa al bloque ME_6.

En el bloque ME_6, se guarda el tiempo actual tm1 en la variable pulso, y se ponen el resto de las variables a cero, para esperar otro pulso.

En el bloque ME_7, se hace el cálculo de las pulsaciones por minuto, utilizando el valor de tiempo del último pulso, que se guardó en la variable "pulso" y el tiempo actual a la llegada del nuevo pulso, se calcula la diferencia de estos valores, que nos dará en unidades de 10 µs, dividimos por 10000 para obtener la frecuencia en Hz, y multiplicamos por 60 para que el resultado esté en ciclos por minuto, ó pulsaciones por minuto.

En el bloque ME_8, devolvemos el valor calculado en la variable ppm a la rutina que llama.

Funcionamiento de la rutina de leer señal:

En el bloque MLS_1, se arranca la interrupción para que el convertidor A/D, realice la conversión de la señal analógica y guarde el resultado en el buffer apuntado por la variable puntero ptr.

A continuación, en el bloque MLS_2 lee el contenido de la dirección del puntero, que es el valor de la muestra analógica, ya convertida en señal digital.

En el bloque MLS_3, se devuelve el contenido del buffer, para pasarlo a la rutina llamante.

3.6. Módulo de conversión A/D.

El módulo de conversión es el encargado de obtener las muestras digitales de la señal analógica, en nuestro caso el módulo A/D del dsPIC30F2010, en la figura 34 tenemos el diagrama funcional de bloques.

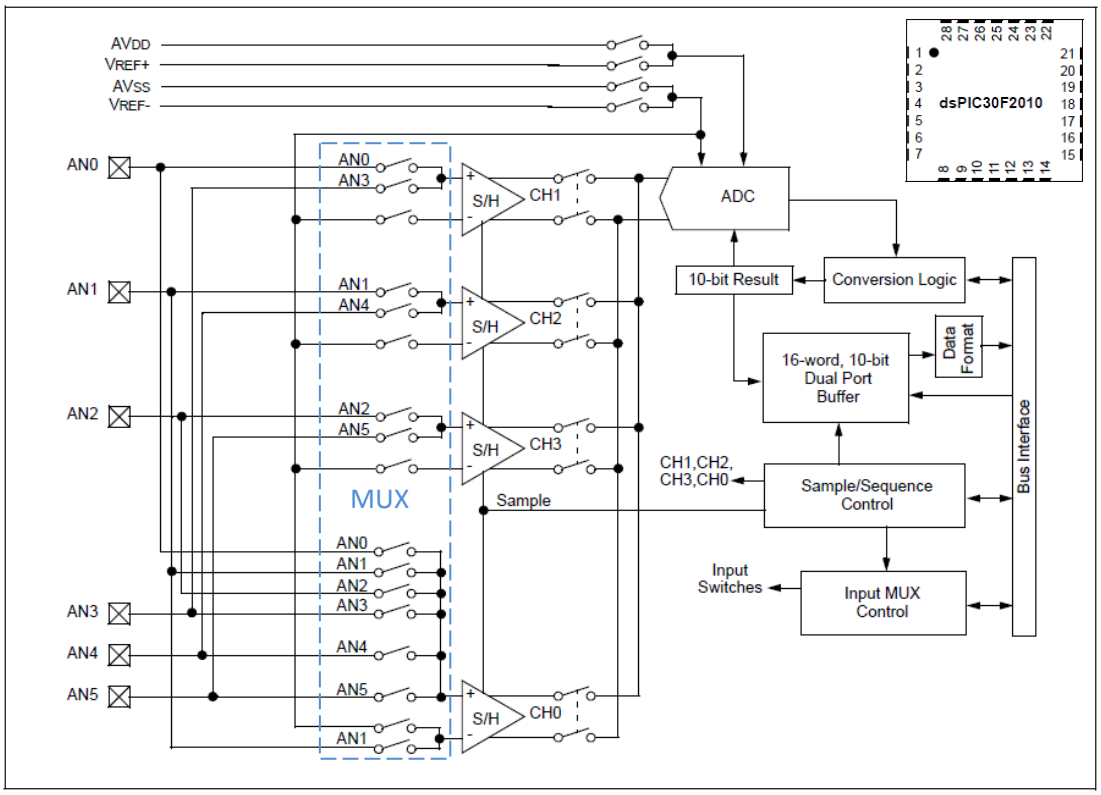


Fig. 35 DIAGRAMA FUNCIONAL DE BLOQUES del ADC dsPIC30F2010

Es un ADC[35] de 10 bits de resolución, lo que nos da una escala de 2^{10}= 1024 muestras de la amplitud de la señal, elegimos el canal 4 (AN4) para conectar la pata 3 del encapsulado de 28 patas QFN[36], y sus registros de 16 bits son (ver tabla 9):

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>					
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>			
ADPCFG	02A8	—	—	—	—	—	—	—	—	—	—	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
ADCSSL	02AA	—	—	—	—	—	—	—	—	—	—	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0

[35] ADC = Analogic Digital Converter (Convertidor Analógico Digital)

[36] QFN es el formato del microchip de 28 patas de encapsulado cuadrado plano, como se aprecia en la esquina superior derecha de la figura 18

Tabla 9 de registros del ADC

ADCON1

- ADON Módulo A/D operativo, si= 1 en funcionamiento [apagado = 0]
- ADSIDL Detiene el módulo cuando está en modo IDLE, si= 1 [no lo detiene = 0]
- FORM Formato de la conversión, si 00 = enteros, 11= ent.+signo. etc.
- SSRC Muestreo simultáneo, si =1 (sólo cuando CHPS = 01)
- ASAM Inicio automático del muestreo [=1 comienza la captura, =0 depende se SAMP]
- SAMP Permite el comienzo del muestreo, si =1 permite
- DONE Indica el final de la conversión si =1.

ADCON2

- VCFG Configuración de la tensión de referencia [=000 aplica la misma del ADC]
- CSCNA Entradas exploradas si =1
- CHPS Canales utilizados en la conversión [si =00 canal CH0]
- BUFS Indica el estado de llenado del buffer[37] [si =1 lleno] (cuando BUFM =1)
- SMPI Genera una interrupción cuando se llega a un núm. de muestreos
  si =1111 Se genera interrupción ISR en la 16ª secuencia de muestreo
  si =1110 id., en la 15ª secuencia
  … = 0000 id. en la 1ª secuencia
- BUFM Modo de configuración del buffer [si =1 dos buffers de 8 bits cada uno]
- ALTS Selección del MUX[38] de muestreo [si =1 rotativo, si =0 fija el MUX A]
- **SAMC** **Tiempo de muestreo automático** [2^{SAMC}x T_{AD}, ej.: 2^{1111}x T_{AD}= 2^{31}x T_{AD}]
- **ADRC** **Indica la fuente de reloj de conversión** [=1 reloj del ADC, =0 reloj sistema]
- **ADCS** **Selección de reloj de conversión**

Ahora vamos a calcular el tiempo total de muestreo del convertidor analógico digital ADC:

$$t = \frac{T_{CY}}{2}(ADCS + 1)$$

$$= 111111 \quad \rightarrow \quad t = \frac{T_{CY}}{2}(63+1) = 32 \cdot T_{CY}$$

...

$$= 000011 \quad \rightarrow \quad t = \frac{T_{CY}}{2}(3+1) = 2 \cdot T_{CY}$$

$$= 000000 \quad \rightarrow \quad t = \frac{T_{CY}}{2}(0+1) = T_{CY}/2$$

siendo: $\boldsymbol{T_{AD}}$ el periodo del reloj del ADC y $\boldsymbol{T_{CY}}$ el período del ciclo de instrucciones

ADCHS Registro para seleccionar las patas de la entrada del conversor

ADPCFG Registro de configuración de las entradas como entradas analógicas o salidas digitales.

[37] Buffer es una memoria intermedia usada para retener los datos mientras se realiza una operación lenta

[38] MUX es el multiplexor de las entradas a los canales de acceso al convertidor

Una vez que hemos calculado la frecuencia mínima de muestreo, en el punto 2.1.2 hallábamos que debe ser mayor de 5 KHz, ó tener un período menor de 200 µs, por tanto hemos de programar los registros para que el tiempo de muestreo T_M < 200 µs

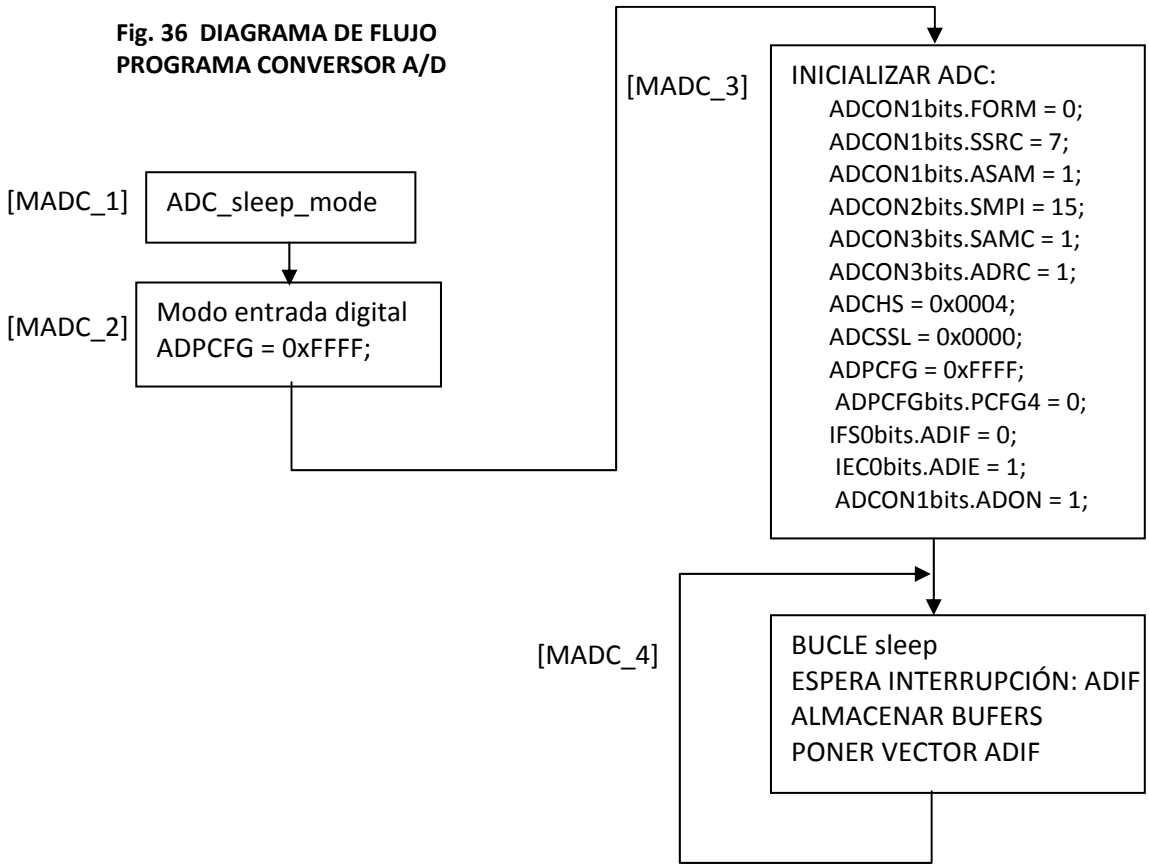
$$T_M = t(adquisición\ de\ la\ muestra) + t(conversión) + t(transferencia)$$

El tiempo de adquisición = T_{SAMP} = T_{AD} (para un solo canal) > 83.33 ns tomamos 100 ns
Tiempo de conversión = t_{CONV} = 12 T_{AD} = 1200 ns =1.2 µs
Tiempo de transferencia = 900 ns (típico), tomamos 1µs
Por tanto T_M mín. = 100 ns + 1200 ns + 900 ns = 2.2 µs
Este tiempo es inferior al máximo necesario de 200 µs, por tanto el tiempo de muestreo queda garantizado, ahora vamos a calcular el tiempo total en la rutina de conversión.
Como son 16 secuencias de muestreo las que se capturan mediante la instrucción: ADCON2bits.SMPI = 15;

El tiempo total de muestreo = 2.2 µs * 16 = 35.2 µs

La secuencia completa del ADC se muestra en el diagrama de flujo de la figura 35, que corresponde a la rutina de muestreo y conversión: *ADC_sleep_mode.c*

Fig. 36 DIAGRAMA DE FLUJO PROGRAMA CONVERSOR A/D



Cuando se genera una interrupción ADIF se prepara el puntero hacia la dirección de los buffers, y desde la rutina de medida del módulo de exploración/actuación se leen las muestras y mientras el ADC permanece en modo *sleep*[39] hasta la siguiente lectura.

[39] sleep: modo de reposo o durmiente

Funcionamiento de la rutina de conversión analógico digital:

El bloque MADC_1 prepara el módulo para ponerlo en estado de espera, hasta que le llegue la interrupción que genera la rutina de leer señal.

EL bloque MACD_2, prepara todas las entradas para que sean entradas digitales.

El bloque MADC_3, configura el convertidor A/D, inicializa todos los registros para que se produzca la conversión, y se pueda almacenar la información de la conversión en el buffer del convertidor A/D.

En el bloque MADC_4, se almacena la información de la conversión en el buffer del convertidor, se prepara el vector de interrupción ADIF para que sea llamado desde otro programa, y se establece un bucle a la espera de una nueva interrupción.

Una vez realizada la adaptación de la señal, y ésta pasa a ser muestreada en esta sección, damos cumplimiento al objetivo número 3.

3.7. Módulo de exploración/actuación.

Este módulo interactúa con la interfaz de usuario, para obtener los umbrales de alarma e informar de la actividad del sistema, ver diagramas de flujo en las figuras 36 y 37.

Fig. 38 DIAGRAMA DE FLUJO RUTINA DE EXPLORACIÓN

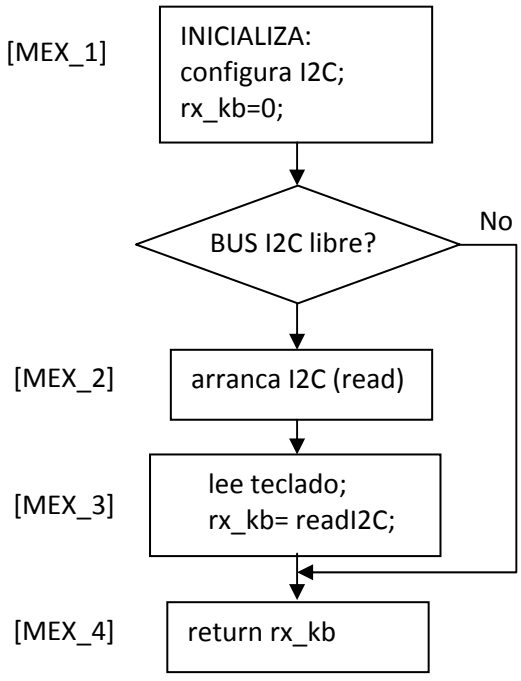
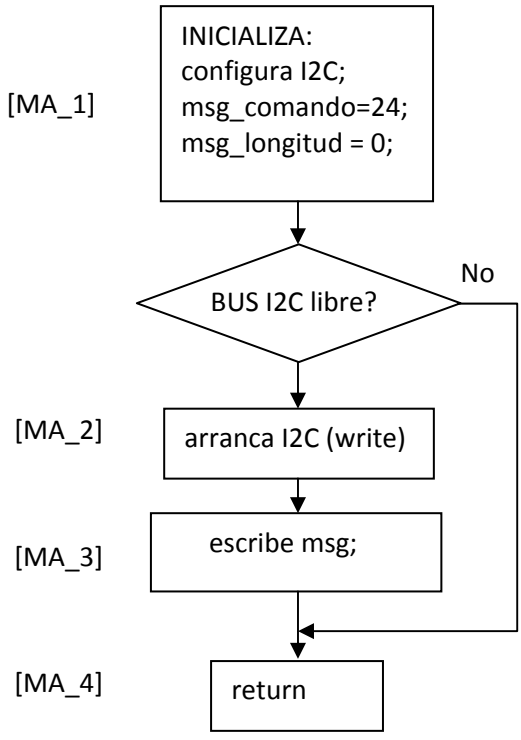


Fig. 38 DIAGRAMA DE FLUJO RUTINA DE ACTUACIÓN



Funcionamiento de la rutina de exploración:

El bloque MEX_1, configura el bus I2C del microcontrolador, e inicializa la variable de recepción rx_kb a cero, a continuación se pregunta si el bus I2C está libre, si no lo está, avanza hasta el bloque MEX_4 para terminar.

Si el BUS I2C está libre, arranca el bus I2C en modo lectura en el bloque MEX_2

A continuación en el bloque MEX_3, lee el teclado del interfaz de usuario, y lo almacena en la variable rx_kb.

Finalmente, el bloque MEX_4 devuelve el valor de rx_kb, a la rutina llamante, si no tiene datos devuelve un cero.

Funcionamiento de la rutina de actuación:

El bloque MA_1, configura el bus I2C del microcontrolador, e inicializa las variables de transmisión de mensaje de comando al valor del número máximo de caracteres del *display* alfanumérico, pone la variable msg_comando=24, y la variable msg_longitud = 0;

A continuación se pregunta si el bus I2C está libre, si no lo está se salta al bloque MA_4, para finalizar

Si el BUS I2C está libre, arranca el bus I2C en modo escritura en el bloque MA_2

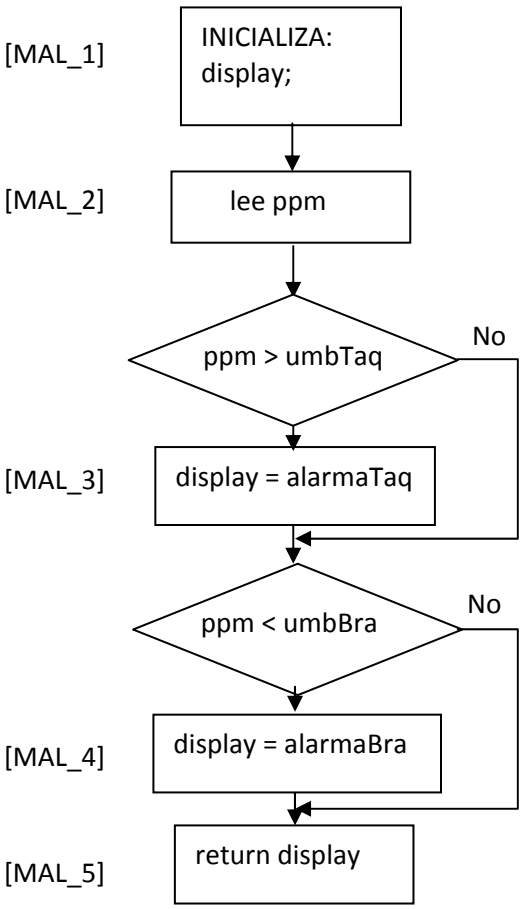
A continuación en el bloque MA_3, escribe el mensaje en la interfaz de usuario

Finalmente, en el bloque MA_4 termina la rutina.

3.8. Módulo de alarmas.

Es el encargado de la detección y generación de alarmas, ver figura 38

Fig. 39 DIAGRAMA DE FLUJO
RUTINA DE ALARMAS



Funcionamiento de la rutina de alarmas:

El bloque MAL_1, inicializa el mensaje para enviar al *display* LCD.

El bloque MAL_2, lee la variable ppm que contiene el valor de las pulsaciones por minuto, y que se ha calculado en el módulo de exploración de señal.

Si ppm es mayor que el umbral de taquicardia (umbTaq), en el bloque MAL_3 se carga el mensaje de alarma de taquicardia en la variable *display*.

A continuación, se pregunta si la variable ppm es menor que el valor del umbral de bradicardia, en cuyo caso, en el bloque MAL_4 se carga la variable *display*, con la alarma de bradicardia.

Finalmente, en el bloque MAL_5, se envía la variable *display* a la rutila llamante.

Con la realización de los módulos software de las secciones 3.3, 3.4, 3.5, 3.7, y 3.8, damos cumplimiento a los objetivos 4 y 5

Capítulo 4. Conclusiones

4.1. Grado de cumplimiento

Se han cumplido todos los objetivos, en particular se ha necesitado replanificar la arquitectura del sistema, y la definición de los módulos software, con el fin de sincronizar todas las tareas dentro de los plazos previstos. Además se ha cambiado el interfaz de usuario ó módulo de alarmas, lo que ha resultado una gran ventaja, ya que es más amigable al usuario que la opción de LED's y facilita mucho las labores de programación.

Las partes que han quedado más completas han sido, por un lado el estudio de los fundamentos de la generación de la señal biométrica, la planificación y el rigor a la hora de poner los objetivos de forma realista, y el diseño hardware.

Habría sido una buena opción la realización de un prototipo completo, pero, esto llevaría mucho más tiempo del que disponemos, y probablemente más personas. No obstante se ha realizado el circuito de adaptación de la señal, con pocos medios, es cierto, pero que ha cumplido su función.

En la implementación de código fuente de los módulos software, se han hecho los más importantes, el módulo de conversión A/D y el módulo de prueba del interfaz de usuario, no obstante, como el método de prueba de la herramienta MPLAB es el mismo, y una vez que se completen todos los módulos, la verificación de dichos módulos se hará de la misma forma.

4.2. Lecciones aprendidas

Creo que he aprendido mucho en este proyecto, por un lado sirve para consolidar muchos conocimientos anteriores, y por otro permite adquirir conocimientos nuevos, tanto en el manejo de herramientas fundamentales de los proyectos, como PROYECT, también he aprendido en la redacción de los proyectos, y por supuesto las herramientas matemáticas y de diseño.

Pero hay una característica que no se me olvidará: "El rigor" y la dedicación que requiere un proyecto serio como éste. A lo largo del mismo se ve claramente, como se va asimilando este comportamiento, y esto es fundamental para dar una buena calidad.

Por supuesto que muchas de estas lecciones, y el rendimiento obtenido, se deben al Tutor del proyecto, que ha tenido una enorme paciencia, y siempre ha estado apoyando y resolviendo cualquier duda.

Dedicando horas y horas de trabajo a este proyecto, durmiendo poco, con tesón y constancia, se puede lograr un buen trabajo, que además puede ser útil para otros.

Capítulo 5. Bibliografia

5.1. Recursos bibliográficos.

[1] Alan V. Oppenheim, Willsky y otros, Sistemas y Señales 2ed., Editorial: Prentice Hall, ISBN: 0-13-814757-4
[2] Enrique PALACIOS, Fernando REMIRO, Lucas LOPEZ, Microcontrolador PIC16F84, Desarrollo de Proyectos 2ed. (Spanish Edition, Ed.: RA-MA Editorial, ISBN: 84-7897-691-4
[3] JOSÉ MARÍA ANGULO USATEGUI, IGNACIO ANGULO MARTÍNEZ, BEGOÑA GARCÍA ZAPIRAÍN, JAVIER VICENTE SÁEZ,MICROCONTROLADORES AVANZADOS DSPIC. Ed.: Thomson Editores Spain, Paraninfo S.A., ISBN: 84-9732-385-8
[4] MPLAB^{R} IDE User's Guide, versión de libre distribución de MICROCHIP THECHNOLOGY
[5] Teresa S. Stover, El libro de Proyect 2007, Microsoft Office Proyect 2007 Inside Out (Spanish Edition), Ed.: ANAYA MULTIMEDIA, ISBN: 978-84-415-2283-1
[6] Gilberto E. Urroz, Introduction to SCILAB ed.: 2001, Distributed by InfoClearinghouse.com
[7] Calvo Rolle, Jose Luis, Scilab: Programacion y Simulacion, Editorial: Alfaomega ISBN: 9786077686675
[8] Robert F. Coughlin,Amplificadores operacionales y circuitos integrados lineales, Editorial: Prentice Hall
[9] Miguel A Pérez, Instrumentación Electrónica, Segunda Ed., Editorial: Editorial Thomson
[10] FRANCISCO LOPEZ FERRERAS, ANALISIS DE CIRCUITOS LINEALES, Editorial: RAMA, ed. 01 2010, ISBN: 978-84-7897-943-1
[11] Juan Ramón Alarcón y Antonio Blanco Solsona, Pspice Iniciación y referencia, Editorial: Mc Graw-Hill, ISBN 84-481-2820-6
[12] Sandler, Spice Circuit Handbook , Editorial: McGraw-Hill, ISBN:71468579 ISBN-13: 9780071468572
[13] 555 *Timer Applications Sourcebook, with Experiments (English) ISBN: 9780672215384*

5.2. Direcciones de Internet, recursos y enlaces WEB

- DIR. 1: http://www.eccpn.aibarra.org/temario/seccion4/capitulo56/capitulo56.htm
- DIR. 2: http://www.daycounter.com/Filters/Sallen-Key-LP-Calculator.phtml
- DIR. 3: http://es.wikipedia.org/wiki/Amplificador_de_instrumentaci%C3%B3n
- DIR. 4: http://www.daycounter.com/Filters/Sallen-Key-LP-Calculator.phtml
- DIR. 5: http://www.fundaciondelcorazon.com
- DIR. 6: http://en.qi-hardware.com/w/images/2/2c/I2C1.jpg
- DIR. 7: http://www.microchip.com

Capítulo 6. Glosario de términos

- ADSL: (Asymmetric Digital Suscriber Line) es la tecnología de los módems de alta velocidad sobre par de cobre.
- Auriculoventricular: Región entre la aurícula y el ventrículo
- Autocad: paquete de software de diseño mecánico en 3 dimensiones
- Bioeléctricas: señales eléctricas procedentes de los seres vivos
- Convolución: Operador matemático que transforma dos funciones
- Diástole: relajación de todos los músculos del corazón (permite la entrada de sangre)
- Internodales: Entre dos nodos ó nudos
- Microsoft Office: Paquete de programas de procesamiento de texto, gráficos, datos, etc.
- MPLAB: Programa de simulación y cálculo matemático
- Optoacoplador: Es un dispositivo aislado de emisión y recepción de luz
- Photoshop: paquete de software de diseño de imágenes
- PSPICE: programa de diseño de circuitos
- Retroiluminada: Cuando la luz se recibe por la parte posterior del objeto iluminado
- SCILAB: Programa de simulación y cálculo matemático
- Sinoauricular: Nódulo sinoauricular (SA), grupo de células localizadas en la pared auricular derecha del corazón.
- Sinusal: Ritmo sinusal, es un término utilizado en Medicina para describir el latido normal del corazón.
- Sístole **auricular:** contracción de las aurículas (impulsa la sangre hacia los ventrículos)
- Sístole ventricular: contracción de los ventrículos (impulsa la sangre fuera del corazón)
- Wiris: paquete de software de cálculo matemático.

Capítulo 7. Anexos

Anexo 1: Varios electrocardiogramas de prácticas clínicas

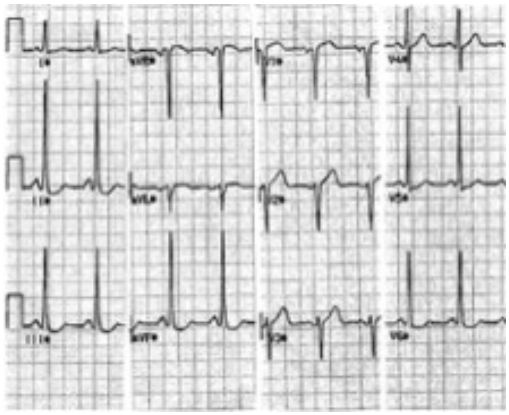


Fig. A 1, Electrocardiogramas

Anexo 2: Esquema circuito amplificador

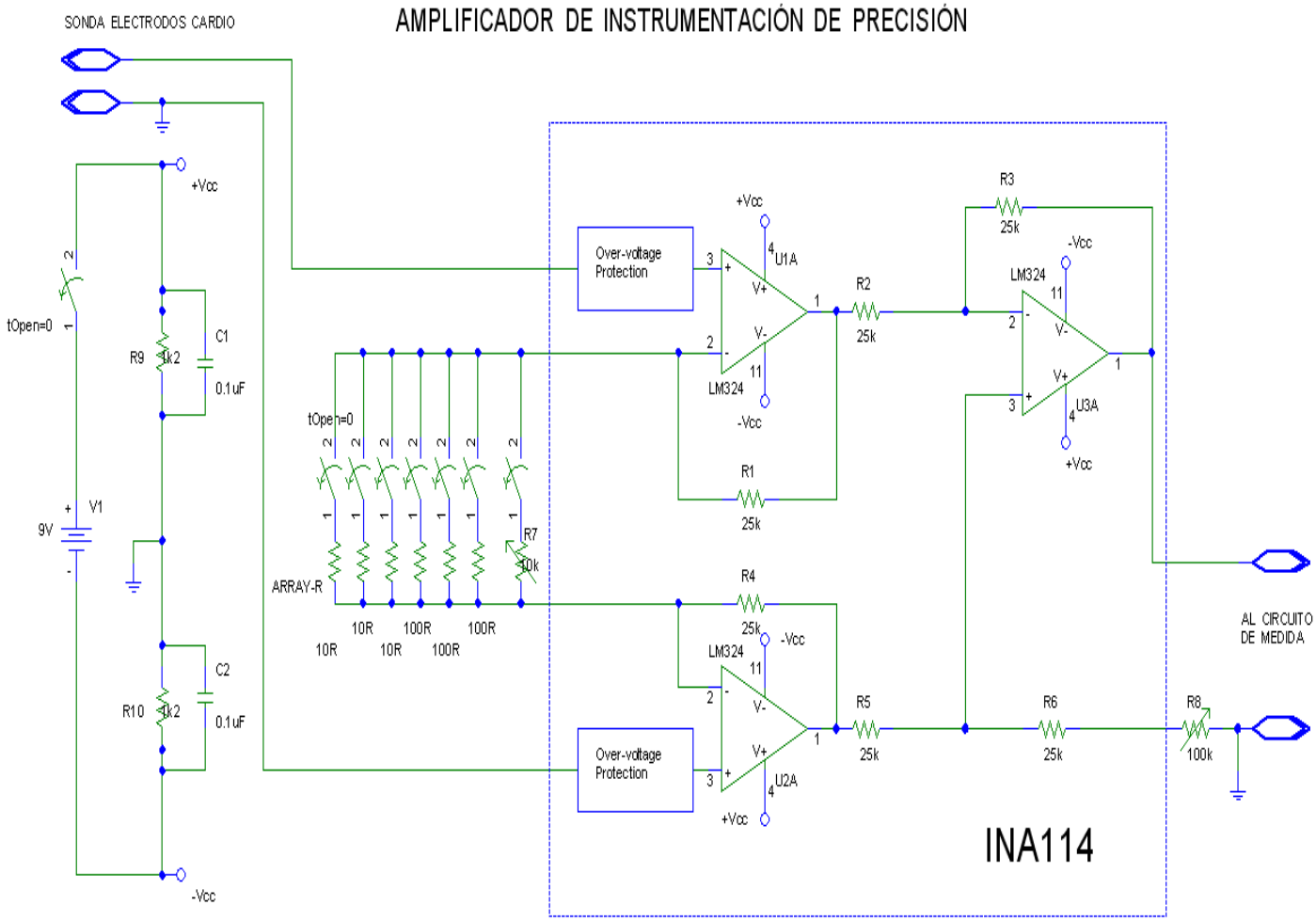


Fig. A 2, AMPLIFICADOR LINEAL

Anexo 3: Filtro paso banda

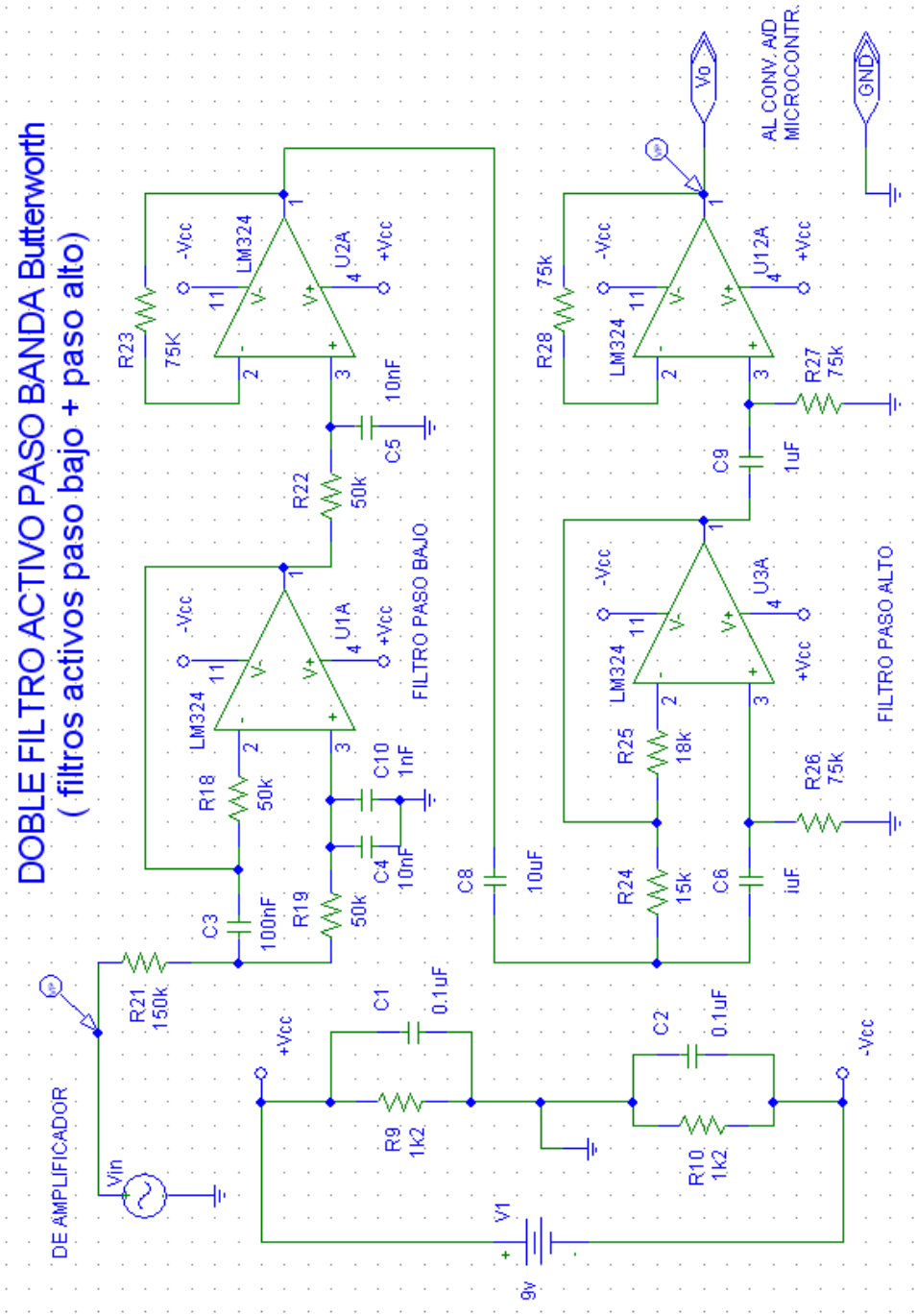


Fig. A 3, FILTRO PB diseño PSPICE

Anexo 4: Programas de MPLAB y Scilab en soporte de fichero adjunto a la memoria

```
* Procesador:             dsPIC30F
* Compilador:             MPLAB ® C30 v1.32.00 o superior
* IDE:                          MPLAB ® IDE v7.20.01 o posterior
* Dis. BOARD usado: dsPICDEM 1.1 Placa de Desarrollo
* Dependencias de hardware: Ninguna
*
* ACUERDO DE LICENCIA:
* Microchip Technology Inc. ("microchip") la licencia de este software es
* únicamente para su uso con Microchip dsPIC ® controlador de señal digital
* El software es propiedad de Microchip y está protegido por
* Leyes de copyright aplicables. Todos los derechos reservados.
*
* EL SOFTWARE SE ENTREGA "TAL CUAL". MICROCHIP EXPRESAMENTE RENUNCIA A TODA
* GARANTÍA DE NINGÚN TIPO, YA SEA EXPRESA O IMPLÍCITA, INCLUYENDO CUALQUIER
* OTRA, LAS GARANTÍAS DE COMERCIALIZACIÓN, ADECUACIÓN A UN PROPÓSITO PARTICULAR
* NO INFRACTOR. EN NINGÚN CASO MICROCHIP SE HACE RESPONSABLE POR CUALQUIER DAÑO,
* DIRECTOS, INDIRECTOS O EMERGENTES DAÑOS Y PERJUICIOS ETC., PÉRDIDA DE
* BENEFICIOS O PÉRDIDA DE DATOS, DAÑO A SU EQUIPO, LOS COSTOS DE LA ADQUISICIÓN
* DE BIENES, tecnología o servicios, CUALQUIER RECLAMACION POR TERCEROS
* (INCLUYENDO PERO NO LIMITANDO A CUALQUIER LA DEFENSA DE LOS MISMOS),
* CUALQUIER RECLAMACIÓN DE INDEMNIZACIÓN O CONTRIBUCIÓN u otros gastos similares.
*
* HISTORIA DE REVISIONES:
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Autor                    Comentarios
    Fecha de esta revisión
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Hw                              Primera versión de la fuente de archivo            09/30/05
* ANGEL                           ADAPTACIÓN PROYECTO
    08.11.2010
*
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*
* NOTAS ADICIONALES:
*
**********************************************************************/

#include "p30fxxxx.h"
// Macro para la configuración de los registros del fusible (Copiado de archivo de dispositivo
de cabecera):
_FOSC (CSW_FSCM_OFF y XT_PLL8);   // Configuración de cristal multiplicado por 8x PLL
_FWDT (WDT_OFF);                                // Desactivar el temporizador del perro
guardián.
_FBORPOR (MCLR_EN y PWRT_OFF);           // Reset para habilitar el pin MCLR y desactivar los
temporizadores
                                                                // (power-up) de tiempo de
encendido.
_FGS (CODE_PROT_OFF);                           // Deshabilitar código de protección

// Funciones y variables con ámbito global:
int main (void);
void ADC_Init(void);
void __attribute__((__interrupt__)) _ADCInterrupt(void);

unsigned int conversionResult[16];
volatile unsigned int *ptr;

int main (void)
{
        ADPCFG = 0xFFFF;                // activar pin UN para depuraciónen modo digital ICD
2
        ADC_Init ();                            // Inicializa el convertidor A/D

        while (1)                               // Inicia un ciclo contínuo - La ejecución es
por interrupciones
        {                                               // A partir de este momento.
                Sleep ();
        }

        return 0;
}

// Funciones:
// ADC_Init () se utiliza para configurar A/D para convertir 16 muestras de un canal de
entrada
// mediante interrupción. El ADC está configurado para usar un oscilador ADRC
```

```
// Se activa la conversión automática y se aplica después de 1Tad del tiempo de muestreo.
// El pin de entrada para la adquisición es AN4.
void ADC_Init(void)
{
        // Registro ADCON1
        // Configurar el  conv. A/D para el muestreo automático
        // Uso interno del contador (SAMC) para proporcionar tiempo de muestreo
        // Configurar el conv. A/D para ser leído en formato entero sin signo.
        // Configuración secuencial de muestreo para múltiples amplificadores S/H
        // Poner todos los demás bits a su estado predeterminado
        ADCON1bits.FORM = 0;
        ADCON1bits.SSRC = 7;
        ADCON1bits.ASAM = 1;

        // Registro ADCON2
        // Configurar la interrupción del ADC después de 16 muestras de llenado del búfer
        // Poner todos los demás bits a su estado predeterminado
        ADCON2bits.SMPI = 15;

        // Registro ADCON3
        // Vamos a establecer el ADRC como el reloj del convertidor A/D
        // Para que el convertidor A/D pueda funcionar cuando el dispositivo está en
        // modo de espera. Además, se asigna un período de Tad para el muestreo de tiempo.
        // El tipo de conversión para el oscilador ADRC dependerá de si el dispositivo
        // programado es un dsPIC30F ó dsPIC33F y también si el módulo A/D es de 10 ó 12 bits
        // NOTA: consultar la hoja de datos del dispositivo para la tasa de conversión "ADRC".
        ADCON3bits.SAMC = 1;
        ADCON3bits.ADRC = 1;

        // Registro ADCHS
        // Configurar el convertidor A/D para avtivar el registro de conversión de la entrada
AN4 en el MUX
        // de los amplificadores de CH0 S/H.
        ADCHS = 0x0004;

        // Registro ADCSSL
        // La exploración de canales se deshabilita. Todos los bits de la izquierda se ponen
en su estado por defecto
        ADCSSL = 0x0000;

        // Registro ADPCFG
        // Establecer canales AN4 como entrada analógica y configurar el resto como digitales
        ADPCFG = 0xFFFF;
        ADPCFGbits.PCFG4 = 0;

        // Borrar el flag de interrupción del ADC
        IFS0bits.ADIF = 0;

        // Habilitar el bit de interrupción del ADC
        IEC0bits.ADIE = 1;

        // Activar el convertidor A/D
        // ESte paso debe hacerse después de configurar los otros registros
        ADCON1bits.ADON = 1;
}

// _ADCInterrupt () Es la rutina de servicio de interrupción (ISR)el A/D.
// La rutina debe tener un alcance global con el fin de ser un ISR.
// El nombre de ISR se elige de acuerdo al linker script del dispositivo.
// Si el dispositivo está en modo SLEEP, el A/D se despierta por interrupción del dispositivo
// Los vectores de un dispositivo de la ISR del ADC estna al servicio de la llamada de INT
(servicio de despertador).
void __attribute__((__interrupt__)) _ADCInterrupt(void)
{
        int i = 0;
        ptr = &ADCBUF0;
        while (i < 16)
        {
                conversionResult[i++] = *ptr++;
        }

        //Borrar el flag de interrupción del ADC en la CPU cuando sea preciso
        //Mantener los vectores hacia la ISR
        IFS0bits.ADIF = 0;
}
/*EOF*/
```

```
//*************************************************//
// Simulación señal discreta del pulso cardíaco normal //
// UOC: Proyecto fin de carrera
// Script versión 2 de fecha 07.12.2010
//*************************************************//
N=80;
k=1:N; // Eje de tiempo discreto
u=[0,1,-.04,-.05,.02,0,.025,-.05,-.03,.01,-.01,0,0,-.01,-.025,.02,.05,.06,.05,.04,.01,0,0,-
.01,.05,.04,0,0,0,0,0,0,.01,.05,.04,-
.01,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
]; // Señal de entrada: de escalón unitario
[y,zf]=filter([1 -0.5 0 ],[1 0.2 -0.1],u); // Señal de salida obtenida mediante el F P.B.
uf=fft(u,-1); // Obtenemos la transformada rápida de fourier de la señal sin filtrar
uz=fft(y,-1); // Obtenemos la transformada rápida de fourier de la señal filtrada
scf;
subplot(2,1,1);
plot2d(k,u,-9,rect=[-5 0 N 1.5]); // Gráfica de la señal de entrada
xtitle('SCILAB: SEÑAL DISCRETA DE ENTRADA DEL PULSO CARDÍACO');
xlabel("Frecuencia en Hz");ylabel("Amplitud [mV]");
a=gca();
a.children.children.polyline_style=6;
a.children.children.foreground=2;
a.children.children.mark_foreground=2;
subplot(2,1,2);
plot2d(k,y,-9,rect=[-5 0 N 1.5]); // Gráfica de la señal de salida después del filtrado
xtitle('SCILAB: SEÑAL DISCRETA A LA SALIDA DEL FILTRO');
xlabel("Frecuencia en Hz");ylabel("Amplitud [mV]");
a=gca();
a.children.children.polyline_style=6;
a.children.children.foreground=2;
a.children.children.mark_foreground=2;
scf;
subplot(2,1,1);
plot2d(k,uf,-9,rect=[-5 0 N 1.5]); // Gráfica de la señal de entrada
xtitle('SCILAB: FFT  DE LA SEÑAL DISCRETA DE ENTRADA DEL PULSO CARDÍACO');
xlabel("Frecuencia en Hz");ylabel("Amplitud [mV]");
a=gca();
a.children.children.polyline_style=6;
a.children.children.foreground=2;
a.children.children.mark_foreground=2;
subplot(2,1,2);
plot2d(k,uz,-9,rect=[-5 0 N 1.5]); // Gráfica de la señal de entrada
xtitle('SCILAB: FFT DE LA SEÑAL DISCRETA A LA SALIDA DEL FILTRO');
xlabel("Frecuencia en Hz");ylabel("Amplitud [mV]");
a=gca();
a.children.children.polyline_style=6;
a.children.children.foreground=2;
a.children.children.mark_foreground=2;
```

Anexo 5: DISTRIBUCIÓN DE ENFERMEDADES, y NÚMERO DE DEFUNCIONES, Fuente INE.

Distribución por capítulos de la Clasificación Internacional de Enfermedades

Año 2007

Capítulos de la CIE	Nº defunciones	%
Total enfermedades	**385.361**	**100,00**
Enfermedades del sistema circulatorio	124.126	32,21
Tumores	103.329	26,81
Enfermedades del sistema respiratorio	44.029	11,43
Enfermedades del sistema digestivo	19.650	5,10
Enfermedades del sistema nervioso y de los órganos de los sentidos	16.723	4,34
Causas externas de mortalidad	15.916	4,13
Trastornos mentales y del comportamiento	12.310	3,19
Enfermedades endocrinas, nutricionales y metabólicas	12.237	3,18
Síntomas, signos y hallazgos anormales clínicos y de laboratorio	11.358	2,95
Enfermedades del sistema genitourinario	10.170	2,64
Enfermedades infecciosas y parasitarias	7.782	2,02
Enfermedades del sistema osteomuscular y del tejido conjuntivo	3.384	0,88
Enfermedades sangre y órganos hematopoyéticos y ciertos trastornos mecanismo inmunidad	1.289	0,33
Enfermedades de la piel y del tejido subcutáneo	1.186	0,31
Afecciones originadas en el periodo perinatal	969	0,25
Malformaciones congénitas, deformidades y anomalías cromosómicas	890	0,23
Embarazo, parto y puerperio	13	0,00

Número de defunciones según las causas de muerte más frecuentes[1]

Año 2007

Total enfermedades	**385.361**
Enfermedades isquémicas del corazón	37.222
Enfermedades cerebrovasculares	33.034
Insuficiencia cardiaca	20.092
Cáncer de bronquios y pulmón	19.975
Enfermedades crónicas de las vías respiratorias inferiores	15.699
Demencia	11.713
Cáncer de colon	10.409
Diabetes mellitus	10.152
Enfermedad de Alzheimer	9.941
Neumonía	8.581
Enfermedad hipertensiva	7.486
Insuficiencia renal	6.543
Cáncer de mama	6.046
Cáncer de estómago	5.747
Cáncer de próstata	5.584

1 Causas con peso relativo superior al 1,4%

Anexo 6: Factura componentes

CONECTROL, S. A. COMPONENTES ELECTRONICOS

JORGE JUAN, 57 Y 58 - 28001 MADRID
TELEFS.: 91 578 10 34 - 91 566 15 20 - 91 435 63 53
FAX: 91 577 58 40

CONECTROL, S.A.

COMPONENTES ELECTRONICOS
INFORMATICA Y COMUNICACIONES

http://www.conectrol.com
e-mail: conectrol@conectrol.com

43049660

DOCUMENTO	FECHA	NUMERO
FACTURA	20/10/2010	13282/20

	DENOMINACION	CANTIDAD	PRECIO	IMPORTE
INA114	C.INTEGRADO MOD.INA 114	2,00	9,5000	19,00
OPA2604	C.INTEGRADO MOD.OPA-2604	2,00	3,8000	7,60
REGLETA3	REGLETA DE RED 3/SALIDAS	1,00	2,5862	2,59
BULA20KN	BANANA HIRSCHMANN BULA-20 NEGRA PbF	1,00	0,7600	0,76
BULA20KR	BANANA HIRSCHMANN BULA-20 ROJA	1,00	0,7600	0,76
3390	CLIP DE PILAS DE 9V. PbF	1,00	0,1271	0,13
PC100X100	PLACA C.I.CUADROS 100X100mm.	1,00	4,1524	4,15
PM25100H	RESIST.PELI.METAL.1% 100H.	3,00	0,0080	0,02
PM2510H	RESIST.PELI.METAL.1% 10H	3,00	0,0070	0,02
CC25GR	CABLE CONEXION 0.25mm. GRIS	2,00	0,1390	0,28
CC25MA	CABLE CONEXION 0.25mm. MARRON	2,00	0,1390	0,28
CC25RO	CABLE CONEXION 0.25mm. ROJO	2,00	0,1390	0,28
CC25AM	CABLE CONEXION 0.25mm. AMARILLO	2,00	0,1390	0,28

o se admiten devoluciones posteriores a los 10 días de la recepción de la mercancía, previa presentación de este documento.

BASE IMPONIBLE	% I. V. A.	IMPORTE I. V. A.	% R. E.	IMPORTE R. E.	RECIBI	TOTAL
						(SIGUE)

ANEXO 7: Operaciones con Scilab

```
a.7.1
 -->p=poly([0 100 20 1], 'v', 'c')
 p  =          2   3
    100v + 20v + v
 -->roots(p)
 ans  =
     0
  - 10.
  - 10.
```

a.7.2

```
-->clear                        // limpiamos variables anteriores
-->x= linspace(0,.975,40);// Reservamos espacio, 40 posiciones para la x  desde 0 a 0.975
-->y=[0,0,-.04,-.05,.02,1,.025,-.05,-.03,.01,-.01,0,0,-.01,-.025,.02,.05,.06,.05,.04,.01,0,0,-
.01,.05,.04,0,0,0,0,0,0,.01,.05,.04,-.01,0,0,0,0]; //Introducimos a mano la variable y
 -->xx=[0,.025, .05,.075, .1,.125, .15,.175, .2,.225, .25,.275, .3,.325, .35,.375, .4,.425,
.45,.475, .5,.525, .55,.575, .6,.625, .65,.675, .7,.725, .75,.775, .8,.825, .85,.875, .9,.925,
.95,.975];  // introducimos a mano todos los tramos de interpolación
-->yy1=interp1(x,y,xx,'linear');      //Primera interpolación lineal
-->yy2=interp1(x,y,xx,'spline');      // Segunda interpolación adaptando la curva spline
 -->yy3=interp1(x,y,xx,'nearest');  //Tercera interpolación por proximidad de puntos
-->plot(xx,[yy1;yy2;yy3],x,y,'*')     // Dibujamos todos los cálculos
-->xtitle('SCILAB: INTERPOLACIÓN PULSO CARDÍACO');
-->xlabel("segundos");ylabel("mV");
-->legend(['aproximación ptos.'],a=5)
```

Ahora que ya tenemos la señal discreta, podemos obtener la transformada z para aplicarle el filtrado a la señal, que como vemos en la figura 6, y su frecuencia de latido es de 1 Hertzio (Hz).

Anexo 8: Demostración del teorema de Shannon

Muestreo con tren de impulsos: $x_p(t)= x(t)p(t)$, siendo $p(t)= \sum\delta(t-nT)$; t tiempo, T período, δ(..) función impulso

$$x_p(t) = \sum_{n=-\infty}^{+\infty} x(nT)\delta(t-nT) \quad \text{la transformada es}: \; x_p(jw) = \frac{1}{2\pi}\left[X(jw) * P(jw)\right] [ec.1]$$

$$P(jw) = \frac{2\pi}{T}\sum_{k=-\infty}^{+\infty}\delta(w-kw_s) \quad \text{la convolución (*) hace que}: \; x_p(jw) = \frac{1}{T}\left[X(j(w-kw_s)\right] [ec.2]$$

$$\text{de ec.1 y ec.2 tenemos, que}: \; w_{señal} < (w_{muestreo} - w_{señal}) \quad \Rightarrow \quad w_{muestreo} > 2w_{señal}$$

Anexo 9: Programas de dsPIC30Fx en soporte de fichero adjunto a la memoria

```
/*********************************************************************/
/**  I2C test for the CrystalFontz CFA533 TMI KC LCD Display/Keypad     **/
/**  Using MPLAB IDE v8.40 and C30                                     **/
/**  For the dsPIC30F4013 Microchip                                    **/
/**  Freely released for those who are having problems like I did      **/
/**  learning how to program and use I2c and need some kind of working  **/
/**  example to start with or just to work from for their projects.    **/
/**  Please don't shoot me down for lack of comments, this did not     **/
/**  happen in one day and I am not a programmer by trade. Good Luck.  **/
/*********************************************************************/
/**  INCLUDES                                                          **/
/*********************************************************************/
#include <p30f4013.h>
#include <I2C.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <libq.h>
#include <timer.h>
#include <libpic30.h>

#define FCY               29480000                    //Instruction cycle rate (Osc x PLL-16)
#define BUFFER_MAX_SIZE           24

#define USE_AND_OR
/*****************************************************************/
#define KP_UP 0x01
#define KP_ENTER 0x02
#define KP_CANCEL 0x04
#define KP_LEFT 0x08
#define KP_RIGHT 0x10
#define KP_DOWN 0x20
/*****************************************************************/

    _FOSC(CSW_FSCM_OFF & FRC_PLL16);      /* Set up for Internal Fast RC since */
    _FWDT( WDT_OFF );
    _FBORPOR( PBOR_OFF & BORV_45 & MCLR_EN & PWRT_OFF );
    _FGS( CODE_PROT_OFF );

/*-----------------------------------------------------------*/
/* Function Prototypes                                                */
/*-----------------------------------------------------------*/

unsigned short get_crc(unsigned char *, unsigned char);
void init_PIC(void);
void clear(void);
void cfont_kp(void);
void cfont_bl(const char lcd_val,const char but_val);
void cfont_clear(void);
void cfont_line1(void);
void cfont_line2(void);
void cfont_write(void);

/*-----------------------------------------------------------*/
/* Variables                                                          */
/*-----------------------------------------------------------*/

static struct DATA
{
    unsigned char  cmd;
```

```
    unsigned char  length;
    unsigned char  reply_len;
    unsigned char  tx_data[BUFFER_MAX_SIZE];
    unsigned char  rx_data[BUFFER_MAX_SIZE];
    unsigned char  line1[BUFFER_MAX_SIZE];
    unsigned char  line2[BUFFER_MAX_SIZE];
    unsigned int   count;
    int                          status;
    unsigned char  current;                          //keys currently pressed
    unsigned char  pressed;                          //keys that have been pressed since the last
poll
    unsigned char  released;                         //keys that have been released since the last
poll
    unsigned char  lcd_bl_val;                       //backlight value 0 to 100 %
    unsigned char  but_bl_val;                       //backlight value 0 to 100 %
}msg;

static union CRC {
        unsigned short  word;
        unsigned char   byte[2];
}crc;

unsigned int config2, config1, blink, Flag;
unsigned int ping_data;
unsigned char *wrptr_tx, *wrptr_rx, *L1, *L2, *wrptr;

void init_PIC(void)
{
    TRISF = 0x000c;
    TRISB = 0x0000;
    TRISD = 0x0000;

/*  Baud rate is set for 100 Khz                     */
/*  I2CBRG = (FCY/FSCK - FCY/1,111,111)-1            */
   config2 = 272; //Sorry, this is a guess and it works from microchip 70046e.pdf for FCY@30 MHZ
/* Clear the DATA Buffers                                                    */
    for(msg.count=0;msg.count<BUFFER_MAX_SIZE;msg.count++)
    {
            msg.rx_data[msg.count] = 0x00;
            msg.tx_data[msg.count] = 0x00;
            msg.line1[msg.count]   = 0x00;
            msg.line2[msg.count]   = 0x00;
    }
    msg.lcd_bl_val = 100;
    msg.but_bl_val = 100;

/* Configure I2C for 7 bit address mode                                 */
    config1 = (I2C_ON & I2C_IDLE_CON & I2C_CLK_HLD \
               & I2C_IPMI_DIS & I2C_7BIT_ADD \
               & I2C_SLW_DIS & I2C_SM_DIS & \
               I2C_GCALL_DIS & I2C_STR_DIS & \
               I2C_NACK & I2C_ACK_DIS & I2C_RCV_DIS & \
               I2C_STOP_DIS & I2C_RESTART_DIS \
               & I2C_START_DIS);
   OpenI2C(config1,config2);
}
int main()
{
            unsigned char tx_data[]      = "  dsPIC30F4013  ";
            unsigned char tx_data1[]     = " Press Any Key  ";
            unsigned char tx_UP[]        = "UP     Pressed  ";
            unsigned char tx_DOWN[]      = "DOWN   Pressed  ";
            unsigned char tx_ENTER[]     = "ENTER  Pressed  ";
            unsigned char tx_CANCEL[]    = "CANCEL Pressed  ";
            unsigned char tx_LEFT[]      = "LEFT   Pressed  ";
            unsigned char tx_RIGHT[]     = "RIGHT  Pressed  ";
            init_PIC();
            cfont_bl(msg.lcd_bl_val,msg.but_bl_val);
            cfont_clear();

            for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
            {
                   msg.line1[msg.count] = tx_data[msg.count];
                   msg.line2[msg.count] = tx_data1[msg.count];
            }
                            cfont_line1();
                            cfont_line2();
            __delay32(80000);   //may not be needed anymore
```

```
    while(1){
                        for(;;){
                                _RB8 = 1;                   //LED to let me know if something is
running
                                cfont_kp();
                                __delay32(1000000);
                                _RB8 = 0;
                                cfont_line2();
                                __delay32(1000000);
                                _RB8 = 1;
                                if(msg.current == KP_UP)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_UP[msg.count];
                                        if(msg.lcd_bl_val < 101)msg.lcd_bl_val ++;
                                        cfont_bl(msg.lcd_bl_val,msg.but_bl_val);
                                }
                                if(msg.current == KP_DOWN)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_DOWN[msg.count];
                                        if(msg.lcd_bl_val > 0)msg.lcd_bl_val --;
                                        cfont_bl(msg.lcd_bl_val,msg.but_bl_val);
                                }
                                if(msg.current == KP_ENTER)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_ENTER[msg.count];
                                }
                                if(msg.current == KP_CANCEL)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_CANCEL[msg.count];
                                }
                                if(msg.current == KP_LEFT)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_LEFT[msg.count];
                                        if(msg.but_bl_val > 0)msg.but_bl_val --;
                                        cfont_bl(msg.lcd_bl_val,msg.but_bl_val);
                                }
                                if(msg.current == KP_RIGHT)
                                {
                                        for(msg.count=0; msg.count < BUFFER_MAX_SIZE; msg.count++)
                                        msg.line2[msg.count] = tx_RIGHT[msg.count];
                                        if(msg.but_bl_val < 101)msg.but_bl_val ++;
                                        cfont_bl(msg.lcd_bl_val,msg.but_bl_val);
                                }

                                __delay32(1000000);
                                _RB8 = 0;

                                }
                    }

            return NULL;
}

/*****************************************************************************/
/**  HIGH-LEVEL FUNCTIONS (Easy Wrappers?)                                  **/
/*****************************************************************************/

/* Clear the DATA Buffers                                                    */
void clear(void)
{
    for(msg.count=0;msg.count<BUFFER_MAX_SIZE;msg.count++)
    {
            msg.rx_data[msg.count] = 0x00;
            msg.tx_data[msg.count] = 0x00;
            msg.line1[msg.count]   = 0x20;
            msg.line2[msg.count]   = 0x20;
    }
    msg.count      = 0;
    msg.status     = 0;
}

void cfont_write(void)
{
```

```
            IdleI2C();
    StartI2C();
            IdleI2C();
            msg.status = MasterWriteI2C(0x54);
            while(msg.status == -1)
            {
                    IdleI2C();
                    I2CSTATbits.BCL = 0;
                    I2CSTATbits.IWCOL = 0;
                    IdleI2C();
                    msg.status = MasterWriteI2C(0x54);
            }
            IdleI2C();

            for(msg.count = 0;msg.count < msg.length + 4;msg.count++)
            {
                    msg.status = MasterWriteI2C(msg.tx_data[msg.count]);
                    while(msg.status == -1)
                    {
                            IdleI2C();
                            I2CSTATbits.BCL = 0;
                            I2CSTATbits.IWCOL = 0;
                            IdleI2C();
                            msg.status = MasterWriteI2C(msg.tx_data[msg.count]);
                    }
                    IdleI2C();
            }

            IdleI2C();
            StopI2C();
    __delay32(50000);   //does not do back to back without this delay
            IdleI2C();
    StartI2C();
            IdleI2C();

            msg.status = MasterWriteI2C(0x55);
            while(msg.status == -1)
            {
                    IdleI2C();
                    I2CSTATbits.BCL = 0;
                    I2CSTATbits.IWCOL = 0;
                    IdleI2C();
                    msg.status = MasterWriteI2C(0x55);
            }
            IdleI2C();

            for(msg.count=0;msg.count<msg.reply_len + 4;msg.count++)
            {
                    IdleI2C();
                    msg.rx_data[msg.count] = MasterReadI2C();
                    IdleI2C();
            if(msg.count<msg.reply_len + 3)AckI2C();
            if(msg.count==msg.reply_len + 3)NotAckI2C();
                    IdleI2C();
            }

        IdleI2C();
        StopI2C();
        IdleI2C();
    __delay32(50000);   //does not do back to back without this delay

}

unsigned short get_crc(unsigned char *ptr, unsigned char count)
{
    unsigned short crc;    //Calculated CRC
    unsigned char  i;              //Loop count, bits in byte
    unsigned char  data;   //Current byte being shifted
    crc = 0xFFFF;                  // Preset to all 1's, prevent loss of leading zeros
    while(count--)
    {
            data = *ptr++;
            i = 8;
            do
            {
                    if((crc ^ data) & 0x01)
                    {
                            crc >>= 1;
```

```
                        crc ^= 0x8408;
                }
                else
                        crc >>= 1;
                        data >>= 1;
            }
            while(--i != 0);
    }
return (~crc);
}

/*****************************************************************************/
/* 6 (0x06): Clear LCD Screen                                                */
/*                                                                           */
/* Sets the contents of the LCD screen DDRAM to ' ' = 0x20 = 32 and          */
/*  moves the cursor to the left-most column of the top line.                */
/* valid data_length is 0                                                    */
/* The return packet will be:                                                */
/* type: 0x40 | 0x06 = 0x46 = 70                                             */
/* data_length: 0                                                            */
/*****************************************************************************/
void cfont_clear(void)
{
            clear();
            msg.cmd                     =      6;
            msg.length                  =      0;
            msg.reply_len         =      0;
            msg.tx_data[0]        = msg.cmd;
            msg.tx_data[1]              = msg.length;
            wrptr_tx                    = msg.tx_data;
    crc.word                    = get_crc(wrptr_tx, msg.length + 2);
            msg.tx_data[2]        = crc.byte[0];
            msg.tx_data[3]        = crc.byte[1];
    IdleI2C();
    StartI2C();
            IdleI2C();
            cfont_write();
}
/*****************************************************************************/
/* 14 (0x0E): Set LCD & Keypad Backlight                                     */
/*                                                                           */
/* valid data_length is 2                                                    */
/* data[0]: LCD backlight power setting (0-100 valid)                        */
/* 0 = off  1-100 = variable brightness                                      */
/* data[1]: keypad backlight power setting (0-100 valid)                     */
/* 0 = off  1-100 = variable brightness                                      */
/* The return packet will be:                                                */
/* type: 0x40 | 0x0E = 0x4E = 78                                             */
/* data_length: 0                                                            */
/*****************************************************************************/

void cfont_bl(const char val_d, const char val_b)
{
            msg.cmd                     = 14;
            msg.length                  = 2;
            msg.reply_len         = 0;
            msg.tx_data[0]        = msg.cmd;
            msg.tx_data[1]              = msg.length;
            msg.tx_data[2]        = val_d;
            msg.tx_data[3]        = val_b;
            wrptr_tx                    = msg.tx_data;
    crc.word                    = get_crc(wrptr_tx, msg.length + 2);
            msg.tx_data[4]        = crc.byte[0];
            msg.tx_data[5]        = crc.byte[1];
    IdleI2C();
    StartI2C();
            IdleI2C();
            cfont_write();
}

/*****************************************************************************/
/* 24 (0x18): Read Keypad, Polled Mode                                       */
/*                                                                           */
/* #define KP_UP 0x01                                                        */
/* #define KP_ENTER 0x02                                                     */
/* #define KP_CANCEL 0x04                                                    */
/* #define KP_LEFT 0x08                                                      */
/* #define KP_RIGHT 0x10                                                     */
```

```
/* #define KP_DOWN 0x20                                                                */
/*                                                                                     */
/* type: 0x18 = 24                                                                     */
/* data_length: 0                                                                      */
/* The return packet will be:                                                          */
/* type: 0x40 | 0x18 = 0x58 = 88                                                       */
/* data_length: 3                                                                      */
/* data[0] = bit mask showing the keys currently pressed                               */
/* data[1] = bit mask showing the keys that have been pressed since the last poll    */
/* data[2] = bit mask showing the keys that have been released since the last poll   */
/*************************************************************************************/

void cfont_kp(void)
{
            msg.cmd                     = 24;
            msg.length                  = 0;
            msg.reply_len         = 3;
            msg.tx_data[0]        = msg.cmd;
            msg.tx_data[1]              = msg.length;
            wrptr_tx                    = msg.tx_data;
    crc.word                    = get_crc(wrptr_tx, msg.length + 2);
            msg.tx_data[2]        = crc.byte[0];
            msg.tx_data[3]        = crc.byte[1];
    IdleI2C();
    StartI2C();
            IdleI2C();
            cfont_write();
            msg.current  = msg.rx_data[2];
            msg.pressed  = msg.rx_data[3];
            msg.released = msg.rx_data[4];
}
/*************************************************************************************/
/* 31 (0x1F): Send Data to LCD                                                         */
/* This command allows data to be placed at any position on the LCD.                 */
/* type: 0x1F = 31                                                                     */
/* data_length: 3 to 18                                                                */
/* data[0]: col = x = 0 to 15                                                          */
/* data[1]: row = y = 0 to 1                                                           */
/* data[2-21]: text to place on the LCD, variable from 1 to 16 characters            */
/* The return packet will be:                                                          */
/* type: 0x40 | 0x1F = 0x5F = 95                                                       */
/* data_length: 0                                                                      */
/*************************************************************************************/
void cfont_line1(void)
{
    unsigned char y;
    msg.cmd                 = 31;
    msg.length       = 18;
    msg.reply_len    =    0;

    msg.tx_data[0]    = msg.cmd;
    msg.tx_data[1]          = msg.length;
    msg.tx_data[2]    = 0;                      //COL
    msg.tx_data[3]    = 0;                      //ROW
    for(msg.count=4,y=0;msg.count<msg.length+2;msg.count++,y++)msg.tx_data[msg.count] = msg.line1[y];
    wrptr_tx                = msg.tx_data;
    crc.word                = get_crc(wrptr_tx, msg.length + 2);
    msg.tx_data[20]   = crc.byte[0];
    msg.tx_data[21]   = crc.byte[1];
    cfont_write();
}
void cfont_line2(void)
{
    unsigned char y;
    msg.cmd                 = 31;
    msg.length       = 18;
    msg.reply_len    =    0;
    msg.tx_data[0]    = msg.cmd;
    msg.tx_data[1]          = msg.length;
    msg.tx_data[2]    = 0;                      //COL
    msg.tx_data[3]    = 1;                      //ROW
    for(msg.count=4,y=0;msg.count<msg.length+2;msg.count++,y++)msg.tx_data[msg.count] = msg.line2[y];
    wrptr_tx                = msg.tx_data;
    crc.word                = get_crc(wrptr_tx, msg.length + 2);
    msg.tx_data[20]   = crc.byte[0];
    msg.tx_data[21]   = crc.byte[1];
    cfont_write();
}
```

Anexo 10. Modelo matemático de la señal cardíaca.

Para la simulación y obtención de los filtros, se ha elaborado un modelo matemático de la señal cardíaca. Primero como función a trozos, figura A 5, y apoyándonos en este modelo se ha construido otro para SCILAB (ver fig. A 6), con este último modelo, se han obtenido los datos del espectro de frecuencias del impulso cardíaco, y con estos datos se ha podido realizar el cálculo de los filtros. También ha sido de suma utilidad para calcular la frecuencia de muestreo.

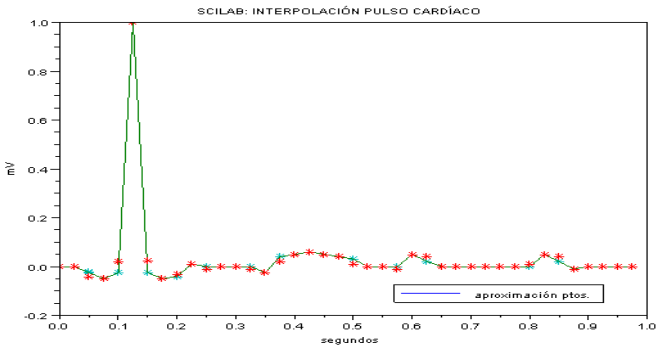


Fig. A 5 Señal resultado de la interpolación de puntos

Fig. A 4 Simulación de la ecuación matemática, con la herramienta WIRIS

Simulación mediante la herramienta scilab[40]:

Caracterizando sólo los picos de la señal de la figura A4, y normalizando a valores enteros del intervalo (fuente: Señales y Sistemas de Alan V. Oppenheim):

$$x[n] = 10^{-3}(y[n-1] + 0.2y[n-2] + 0.01y[n-3]\ [17]$$

La respuesta en frecuencia a esta señal, es:

$$H(e^{jw}) = \frac{1}{10^{-5}(100e^{-jw} + 20e^{-2jw} + e^{-3jw})} \quad [18]$$

La transformada z de esta función, será por tanto:

$$H(z) = \frac{10^5}{100z^{-1} + 20z^{-2} + z^{-3}} \quad [19]$$

Sustituyendo z^{-1} por v, tenemos:

$$G(v) = \frac{10^5}{100v + 20v^2 + v^3} \quad [20],$$

Cuyas raíces calculamos por medio de scilab (anexo 7), ó simplificando queda:

$100v + 20v^2 + v^3 = v(v+10)^2$ y tenemos 3 raíces reales $v_1 = 0$, $v_2 = -10$ y $V_3 = -10$
Ahora descomponemos en fracciones simples, con el resultado:

$$G(v) = \frac{10^5}{100}\left(\frac{1}{v} - \frac{1}{v+10} - \frac{10}{(v+10)^2}\right) \quad [21]$$ de aquí pasamos de nuevo al dominio de z

[40] Como no tenemos una función del tipo f(t)= A*sen(wt+θ)+ …, hay que poner a mano los puntos más representativos.

$$G(z)=10^3\left(\frac{1}{z^{-1}}-\frac{1}{10+z^{-1}}-\left(\frac{\sqrt{10}}{10+z^{-1}}\right)^2\right)=10^3\left(\frac{1}{z^{-1}}-\frac{1}{10}\left(\frac{1}{1+\frac{1}{10}z^{-1}}\right)-\frac{1}{10}\left(\frac{1}{1+\frac{1}{10}z^{-1}}\right)^2\right) \quad [22]$$

Finalmente obtenemos la función de respuesta al impulso unitario ð[n], siendo u[n] la función escalón:

$$h[n]=10^3\left(\delta[n-1]-\frac{1}{10}\left(\frac{1}{10}\right)^n u\{n\}-\frac{1}{10}\left(\frac{1}{10}\right)^{2n}u[n]\right) \quad [23]$$

Como cabía esperar, tenemos un pulso en [n-1] y dos funciones que convergen a cero rápidamente. Este es el comportamiento que observamos en la representación discreta de la figuras A7, y en la figura A6 tenemos el dominio de la frecuencia.

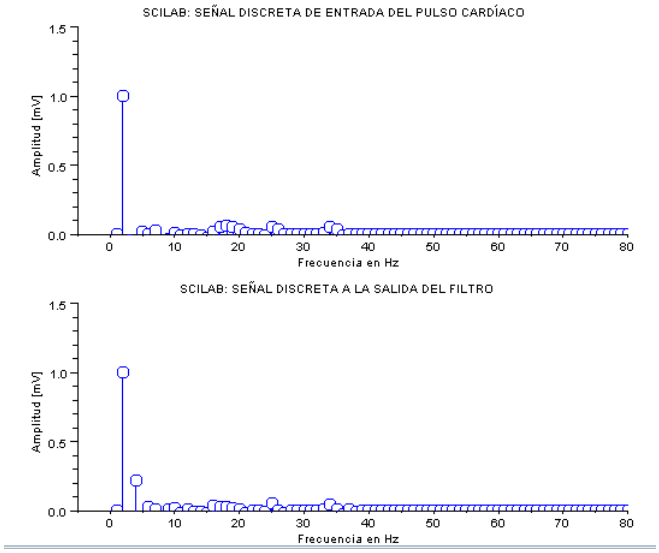


Fig. A 7 Representación gráfica en el dominio de la frecuencia, antes y después del filtro

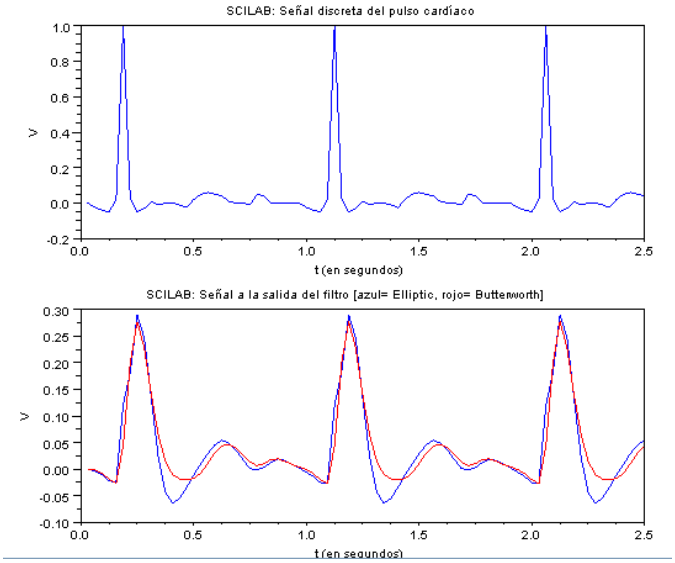


Fig. A 6 Graficas de filtrado de la señal del pulso cardíaco: simulación de filtro activo elliptic de transición optimizada en trazo azul y el filtro butterworth de amortiguamiento crítico en trazo rojo

Tenemos también la transformada rápida de Fourier FFT (Fast Fourier Transform), que nos aporta información valiosísima del espectro de frecuencias del pulso cardíaco. Ver la figura A8.

El programa se ejecuta en la herramienta scilab, que está incluido en el anexo [4].

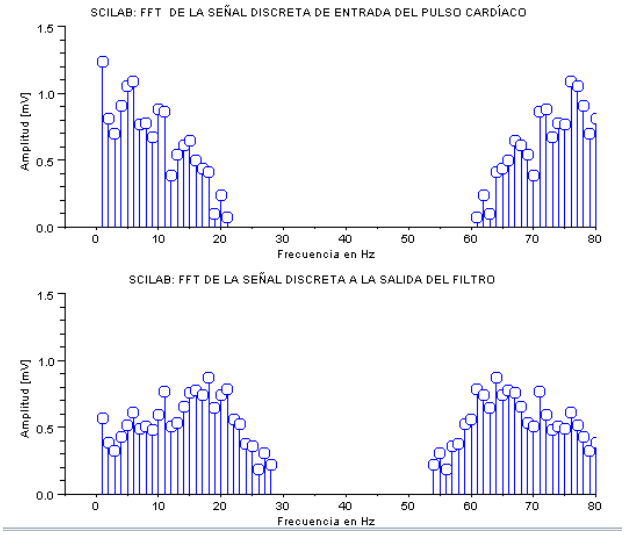


Fig. A 8 Representación gráfica de la transformada rápida de Fourier antes y después del filtro