A D Q C A R

*Equipo de **ADQ**uisición de señales biológicas en memoria de estado sólido, con aplicación en **CAR**diología como Holter digital y miniaturizado.*

Proyecto y construcción de un prototipo de uso clínico para satisfacer los requerimientos de la materia "Proyecto" del Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República Oriental del Uruguay

Estudiantes:
Javier Rodríguez
Rodrigo Duarte
Javier Borca

Docentes:
Prof. Agr. Ing. Franco Simini
Prof. Agr. Dr. Fernando Nieto(†)

Núcleo de Ing. Biomédica
Hospital de Clínicas, Piso 15, Sala 2
Tel: 487 1515 int. 2438

Montevideo, setiembre 2000 - diciembre 2002

PREFACIO:

En determinadas investigaciones de medicina se necesita el registro de señales biológicas durante un tiempo prolongado, interfiriendo lo menos posible con el sujeto en estudio. Nos planteamos el desarrollo de una tecnología que permita la adquisición de señales biológicas con un dispositivo portátil (y eventualmente implantable), de dimensiones reducidas, con tiempo de captura muy largo, y capacidad para almacenamiento de distinto tipo de datos. El equipo tendría que incluir también una inteligencia suficiente para ser capaz de realizar cálculos complejos en tiempo real, como ser algoritmos de compresión, filtrados y clasificación. En definitiva, permitir realizar un pre-procesamiento de los datos a medida que se van adquiriendo, para optimizar la memoria disponible y también acortar el tiempo de descarga y posterior análisis de los datos registrados.

Las características del equipo aquí propuesto, permitirían múltiples aplicaciones para ser utilizado en diferentes áreas, por ejemplo: seguimiento de ganado, marcapasos, medidas de temperaturas, EEG para capturar crisis de epilepsia y registros de 24 horas de ECG (Holter).

ADQCAR es la aplicación de este planteo al diseño de un Holter, que junto con el proyecto CLASICAR permite adquirir, analizar e informar registros de 24 horas. ADQCAR es un prototipo de 3 canales que almacena los datos en una memoria de estado sólido extraíble. ADQCAR también incluyen un programa de computador que permite descargar los datos adquiridos, luego de las 24 horas de captura.

ADQCAR puede utilizar la inteligencia mencionada, por ejemplo para la clasificación en tiempo real de señales ECG, realizado en el proyecto CARDIDENT.

AGRADECIMIENTOS:

Son varias las personas a las que queremos agradecer, entre ellas a médicos, ingenieros, amigos y por supuesto a nuestros familiares, que tanto nos han apoyado y soportado durante todo este tiempo. En particular solo nombraremos a los más allegados al tema del proyecto.

Especialmente nos gustaría poder agradecerle al Dr. Fernando Nieto, quien lamentablemente ya no está entre nosotros. Él nos guió en el comienzo del diseño del proyecto, brindándonos un enorme apoyo y compartiendo con nosotros su gran conocimiento.

A nuestro Director de Proyecto, Ing. Franco Simini, quien nos dió la posibilidad de trabajar en el Núcleo de Ingeniería Biomédica (NIB) desarrollando el proyecto.

Al Dr. Ing. Fernando Silveira por hacerse un tiempo para evacuar nuestras dudas.

A nuestros compañeros del NIB: Rafael Alonso, Juan Carlos Cigarán, Leonardo Díaz, Adriana Ferreira, Santiago González, Fiorella Haim, Raúl Hartman, Rogelio Hernández, Jorge Lobo, Alfredo Rodríguez, Mateo Ruétalo y Rodolfo Suárez, con quienes compartimos conocimientos y experiencias.

INDICE

CAPITULO 1
INTRODUCCCION

CAPITULO 2
ORIGEN Y CAPTACION DE SEÑALES ECG

CAPITULO 3
RESUMEN

CAPITULO 4
ETAPA DE ENTRADA

CAPITULO 5
MULTIPLEXADO Y DIGITALIZACIÓN

CAPITULO 6
MEMORIA EXTRAÍBLE DE ESTADO SÓLIDO

CAPITULO 7
MICROCONTROLADOR

CAPITULO 8
SOFTWARE DEL SISTEMA

CAPITULO 9
ESTUDIO DEL CONSUMO

CAPITULO 10
MANUAL DEL EQUIPO

CAPITULO 11
TIEMPOS, COSTOS Y PRODUCCIÓN

ANEXOS

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DEL CD

1. Documentación del Proyecto
2. Programa ADQCAR.exe para PC
3. Código fuente ADQCAR
4. Desarrollo
5. Material Bibliográfico
6. Presentaciones

BIBLIOGRAFIA

[1] John W. Clark, Jr., Michael R. Neuman, "Medical Instrumentation", J.G. Webster (Ed), pp 121-285, 1998.

[2] Adel S. Sedra, Kenneth C. Smith, “Circuitos Microelectrónicos” Cuarta Edición, Oxford University Press, Mexico. ISBN 970-613-379-8.

[3] Javier Rodríguez Sánchez, “Captación, Amplificación y Acondicionamiento de Señales Bioeléctricas”, Universidad de Alcalá, Práctica 1, 1998.

[4] SanDisk Corporation, “MultiMediaCard Product Manual”, SanDisk Corporation, Rev. 3, 7 / 2001.

[5] Brian W. Kernighan, Dennis M. Ritchie, “El Lenguaje de Programación C” Segunda Edición, AT&T Bell Laboratories, Murray Hill, New Jersey. (Ed) Prentice Hall. ISBN 968-880-205-0.

[6] C. J. Savant Jr, Martín S. Roden, Gordon L. Carpenter, “Diseño Electrónico: circuitos y sistemas”, 1ª edición español de la 2ª en inglés, Wilmington Del, Addison-Wesley Iberoamericana, ISBN 0-201-62925-9, 1992.

[7] Robert F. Coughlin, Frederick F. Driscoll, “Amplificadores Operacionales y Circuitos Integrados Lineales”, traducido de la 4ª edición en inglés. México: Prentice-Hall Hispanoamericana, ISBN 968-880-284-0, 1993.

[8] Motorola Inc., “MMC2001 Reference Manual”, MMC2001RM/D, Motorola Inc., 1998.

[9] Motorola Inc., “M·CORE microRISC Engine Programmer’s Reference Manual”, MCORERM/AD, Motorola Inc., 1997.

[10] Motorola Inc., “MMCCMB1200 Controller and Memory Board (CMB1200) User’s Manual”, Rev. 2, MMCCMB1200UM/D, Motorola Inc., 1998, 1999.

[11] Motorola Inc., “MMCPFB1200 Platform Board User’s Manual”, MMCPFB1200UM/D, Motorola Inc., 1999.

[12] SanDisk Corporation, “SanDisk MultiMedia Card Product Manual”, Rev. 3, SanDisk Corporation, 2001.

[13] José Alfaro, Tabaré Forcellati, Fabián Sarutte, "CLASICAR: Sistema de ayuda a la clasificación automática de complejos QRS para análisis de registros ECG de 24 horas", Núcleo de Ingeniería Biomédica, 2001.

[14] Bernabé Ocampo, Juan Carlos Sturzenegger, Martín Vallarino, "Sistema Automático de Monitorización y Grabación de Señales Biológicas para Computador PC-Compatible", Núcleo de Ingeniería Biomédica, Mayo 1995.

[15] Dr. Elías Rovira Gil, "Urgencias en enfermería", 2ª edición, Difusión Avances de Enfermería, (DAE S.L.), ISBN 84-931330-2-7, 2001.

[16] Roy Hoffman, “Data Compression in Digital Systems”, Chapman and Hall, 1st edition, ISBN 0-412-08551-8, April 1997.

[17] Antti Ruha, Sami Sallinen, Seppo Nissilä, “A Real-Time Microprocessor QRS Detector System with a 1-ms Timing Accuracy for the Measurement of Ambulatory HRV”, *IEEE Transactions on Biomedical Engineering*, Vol. 44, pp: 159-167, March 1997.

[18] Paul Lander, Edward J. Berbari, “Time-Frequency Plane Wienner Filtering of the High-Resolution ECG: Development and Application”, *IEEE Transactions on Biomedical Engineering*, Vol. 44, pp: 256-265, March 1997.

[19] Michael L. Hilton, “Wavelet and Wavelet Packet Compression of Electrocardiograms”, *IEEE Transactions on Biomedical Engineering*, Vol. 44, pp: 394-402, May 1997.

[20] A. G. Ramakrishnan, Supratim Saha, “ECG Coding by Wavelet-Based Linear Prediction”, *IEEE Transactions on Biomedical Engineering*, Vol. 44, pp: 1253-1261, May 1997.

[21] Hanwoo Lee, Kevin M. Buckley, “ECG Data Compression Using Cut and Align Beats Aproach and 2-D Transforms”, *IEEE Transactions on Biomedical Engineering*, Vol. 46, pp: 556-564, May 1999.

[22] Hanwoo Lee, Kevin M. Buckley, “ECG Data Compression Using Cut and Align Beats Aproach and 2-D Transforms”, *IEEE Transactions on Biomedical Engineering*, Vol. 46, pp: 556-564, May 1999.

[23] Valtino X. Alfonso, Willis J. Tompkins, Truong Q. Nguyen, Shen Luo, “ECG Beat Detection Using Filter Banks”, *IEEE Transactions on Biomedical Engineering*, Vol. 46, pp: 192-201, February 1999.

[24] Jie Chien, Schuichi Itoh, “A Wavelet Transform-Based ECG Compression Method Guaranteeing Desired Signal Quality”, *IEEE Transactions on Biomedical Engineering*, Vol. 45, pp: 1414-1419, December 1998.

[25] Shubha Kadambe, Robin Murray, G. Faye Bourdeaux-Bartels, “Wavelet Transform-Based QRS Complex Detector”, *IEEE Transactions on Biomedical Engineering*, Vol. 46, pp: 838-848, July 1999.

CAPITULO 1 – INTRODUCCIÓN

1.1 Propuesta

El análisis continuo del electrocardiograma (ECG) es una técnica usual en cardiología, para el cual normalmente se utiliza un equipo portátil de adquisición y grabado en cinta magnética o memoria de estado sólido, que luego es analizada en una estación de visualización.

Este proyecto propuso el diseño y construcción de un equipo Holter operante de 3 canales para el registro de la señal ECG completa de 24 horas, es decir que no se registran sólo eventos aislados. Se utiliza una memoria de estado sólido para el almacenamiento de la señal, acorde con la tecnología existente en el mercado. También se implementó la descarga de los datos hacia un PC, creando para esto el software necesario y utilizando un dispositivo lector de tarjeta de memoria extraíble, con comunicación USB al PC.

Todo el proyecto se encaró como una base para futuros desarrollos relacionados con la adquisición de señales biológicas. También se tuvo muy en cuenta en el estudio, que el consumo de energía sea mínimo.

1.2 Motivación

Nuestra motivación fue la de armar un equipo funcional completo que realmente tenga utilidad práctica en el mercado, así como el trabajar con tecnología de punta.

Utilizamos un microcontrolador de gran capacidad y de última generación, usado actualmente en equipos como por ejemplo teléfonos celulares. Para el almacenamiento trabajamos con tarjetas MultiMedia Card, que cada vez se están utilizando más en los dispositivos portátiles modernos, en sustitución de otras tecnologías que ocupan mayor tamaño.

También tuvimos que encarar diferentes desafíos a lo largo del proyecto, que nos permitieron aprender mas de lo esperado, lo cual era parte implícita de nuestra motivación inicial.

CAPITULO 2 – CARACTERÍSTICAS DE LAS SEÑALES ECG

Para poder desarrollar este sistema, primero debimos comprender mejor las señales de origen bioeléctrico generadas en el corazón, que será registrada por nuestro equipo. A continuación explicaremos de forma general el origen de estas señales en el cuerpo del paciente, así como sus características eléctricas.

2.1 Conceptos básicos de electrocardiografía

La electrocardiografía registra los potenciales eléctricos generados por el corazón. La actividad bioeléctrica cardiaca tiene su origen en la actividad bioeléctrica de cada una de las células musculares cardíacas. Esta actividad eléctrica produce la contracción rítmica del corazón y se desarrolla según un orden estricto, latido tras latido. Las células miocárdicas son excitadas por un estímulo eléctrico propagado por el haz de His y ramificaciones de Purkinje, que distribuyen el impulso inicial según una secuencia que se explicará más adelante.

Actividad eléctrica celular:

La estimulación de una célula muscular aumenta la permeabilidad de su membrana, produciendo una serie de cambios iónicos de ambos lados de dicha membrana. El registro de este fenómeno es una curva que se denomina **potencial de acción transmembrana** (PAT), que consta de las siguientes partes y fases (ver Figura 2.1):

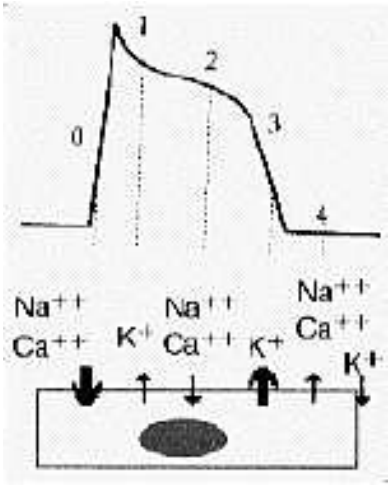


Figura 2.1 – *Potencial de acción trasmebrana de una célula (tomado de* "Medical Instrumentation", J.G. Webster *[1]).*

- **Despolarización** ("activación")

    - Fase 0: Entrada súbita de Ca++ y Na++ al interior de la célula.

- **Repolarización** ("recuperación")

    - Fase 1 e inicio de la fase 2: Persiste la entrada de Ca++ y Na++ y se inicia la salida de K+ al exterior de la célula.
    - Final de la fase 2 y fase 3: La salida de K+ es máxima. Se inicia el restablecimiento del equilibrio iónico inicial.
    - Fase 4: Se restablece el equilibrio iónico inicial mediante un mecanismo de transporte activo.

Desde un punto de vista eléctrico se pueden definir dos zonas: el subepicardio y el subendocardio. Ambas están separadas por lo que se denomina endocardio eléctrico. La zona subendocárdica es la primera que se despolariza y la última que se repolariza, y de esta manera el PAT del subendocardio se inicia antes y finaliza más tarde que el PAT del subpericardio. El ECG de superficie es la resultante de las dos curvas como se observa en la Figura 2.2

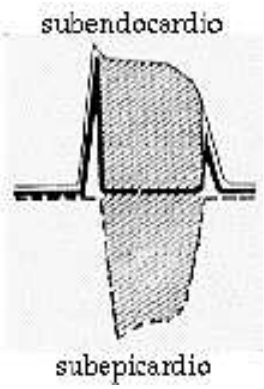


Figura 2.2 – *La señal ECG de superficie es la resultante de las componentes subepicardio y subendocardio [1].*

Generación y Registro de la Señal ECG:

A continuación se realiza una breve descripción de los fenómenos básicos que aparecen en un electrocardiograma real. Como se muestra en la Figura 2.3, el sistema de conducción eléctrica del corazón está formado por haces de células especializadas en la conducción eléctrica, como si se tratara de conductores metálicos que distribuyen la energía a las células miocárdicas contráctiles.

El estímulo eléctrico en el corazón sano nace en el marcapaso o nódulo sinoauricular (NSA), que es una pequeña formación de tejido muscular especializado, localizado en la aurícula derecha en el sitio de unión de la vena cava superior y la propia aurícula.

La acción de este estímulo da origen a una pequeña corriente eléctrica, llamada corriente de excitación u onda de despolarización (onda P). Esta onda se propaga en forma de anillos concéntricos envolviendo ambas aurículas.

Cuando esta onda de excitación llega a la zona de unión entre la aurícula y el ventrículo derecho, debe pasar a través de otro nódulo llamado aurículo-ventricular (AV), donde la velocidad de propagación es mucho menor que en las aurículas y que el resto del tejido de conducción, como consecuencia la onda se retarda. Esto se revela en el ECG por la aparición de un segmento horizontal ya que la energía generada en este nódulo es demasiado pequeña como para registrarla con los electrodos externos.

En este nódulo AV comienza el llamado haz de His, que es un tejido cardíaco especializado, que presenta una mayor conductibilidad al paso de la corriente que el tejido muscular circundante. Del haz de His se desprenden tres ramas, dos izquierdas (para el Ventrículo Izquierdo) y una derecha (para el Ventrículo Derecho). Una vez en ellos, estas ramas emiten finas prolongaciones, llamadas fibras de Purkinje, que se entremezclan finalmente con las fibras musculares cardíacas, a las que comunican la energía eléctrica a fin de lograr su contracción.

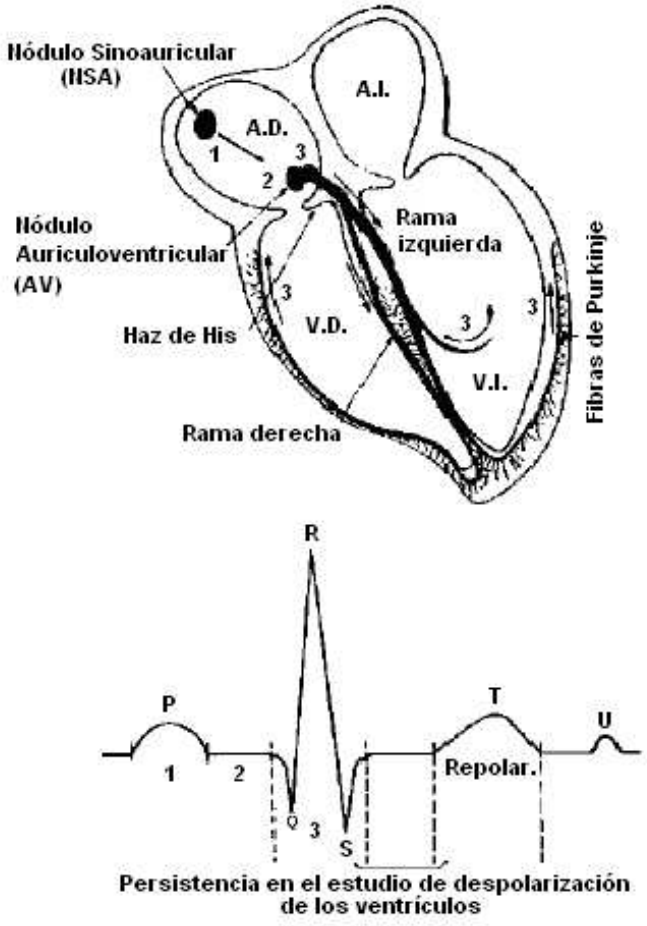


Figura 2.3 – *Generación del pulso eléctrico y vías de conducción hasta las células miocárdicas [1].*

Durante la conducción del impulso, desde el comienzo del haz de His hasta las últimas células cardíacas, se obtiene una onda llamada QRS, con las características mostradas en la Figura 2.3, que representa la llamada despolarización ventricular. La forma del complejo indica las fuerzas eléctricas desarrolladas en los ventrículos. Los ventrículos, tras un momento de reposo durante el cual permanecen en un estado de desporalización (segmento ST), comienzan a repolarizarse, partiendo como la onda de despolarización, desde la punta hacia la base del corazón, siguiendo prácticamente el mismo camino que esa onda. Es en ese momento en el cual se genera la onda llamada T.

La repolarización de las aurículas coincide con la generación del QRS, y por ser de mucho menor voltaje es ocultada totalmente. Además de las ondas mencionadas, luego de la onda T aparece una última onda U, cuyo origen es desconocido.

En el electrocardiograma se miden los potenciales de acción entre varios puntos de la superficie de un volumen conductor. Para simplificar su medida se ha desarrollado un modelo simple para representar la actividad eléctrica del corazón. En este modelo, el corazón consiste en un dipolo eléctrico localizado en el tórax, como se muestra en la Figura 2.4.

Este campo particular, y el dipolo que produce, representan la actividad eléctrica del corazón en un instante específico. En un instante posterior el dipolo puede cambiar su orientación y magnitud, por lo tanto puede causar cambios en el campo eléctrico. Una vez aceptado este modelo, se puede representar este campo eléctrico por el momento dipolar M, conocido como vector cardiaco. En el progreso del ciclo cardíaco la magnitud y dirección de M varían, porque el dipolo eléctrico varía.

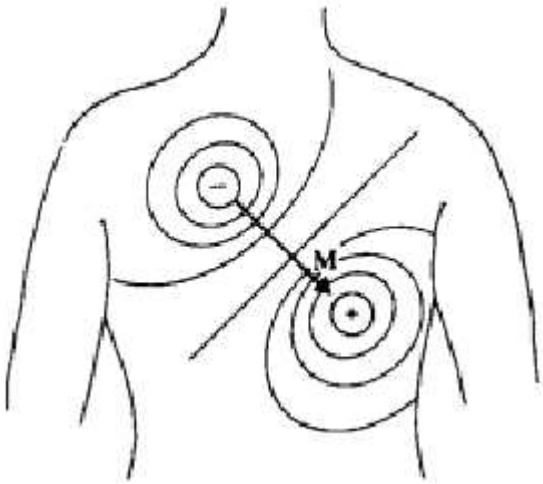


Figura 2.4 - *Representación de la actividad eléctrica del corazón mediante un dipolo eléctrico [1].*

Los potenciales eléctricos generados por el corazón atraviesan el cuerpo y aparecen en su superficie. Se determinan las diferencias de potencial ubicando electrodos en la superficie del cuerpo y midiendo el voltaje entre ellos, así se obtienen proyecciones del vector M. Si dos electrodos son ubicados en diferentes líneas equipotenciales del campo eléctrico del corazón, se medirá una diferencia de

potencial distinta de cero. Pares de electrodos diferentes ubicados en distintos sitios generalmente producen diferentes resultados por la dependencia espacial del campo eléctrico del corazón. Para esto es importante mantener cierto estándar de posiciones para la evaluación clínica de la señal ECG.

Para obtener la actividad cardiaca completa, se considera que los potenciales cardíacos se proyectan a lo largo de los ejes existentes en cada uno de los tres planos de referencia, el plano frontal, el plano sagital y el plano transversal (según Figura 2.5). Se realizan varios registros o derivaciones tomadas en el plano frontal y en el plano transversal.

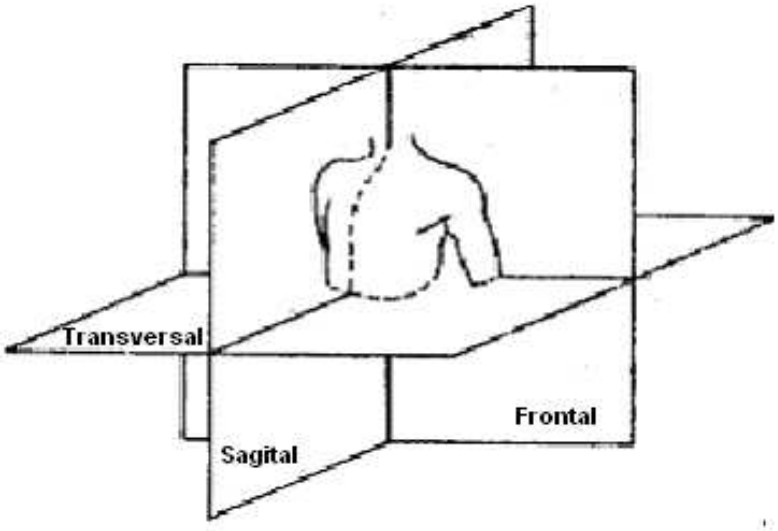


Figura 2.5. *Planos Sagital, Frontal y Transversal [1]*

Se obtienen tres derivaciones básicas en el plano frontal: cuando los electrodos se ubican en el brazo derecho (right arm RA ver Figura 2.6), brazo izquierdo (LA) y la pierna izquierda (LL). Como resultado de estas derivaciones se obtienen los vectores I, registro entre LA y RA, II como registro entre LL y RA y el III entre LL y LA. Estos vectores pueden aproximarse a un triangulo equilátero, conocido como triángulo de Eindhoven, en el plano frontal del cuerpo, como se muestra en la Figura 2.6.

Es posible así obtener el vector eléctrico frontal para cualquier instante del ciclo cardíaco. Una derivación adicional en el plano frontal, usada comúnmente, se llamada derivación unipolar y es el promedio de señales que provienen de dos o más electrodos. Esta derivación es la llamada del electrodo de referencia o “Terminal Central de Wilson”, el voltaje en este nodo es el promedio de los voltajes en tres electrodos ubicados en los tres miembros mencionados anteriormente, conectados a través de una resistencia de 5 MΩ, tomando como referencia de tierra un electrodo puesto en la pierna derecha (RL).

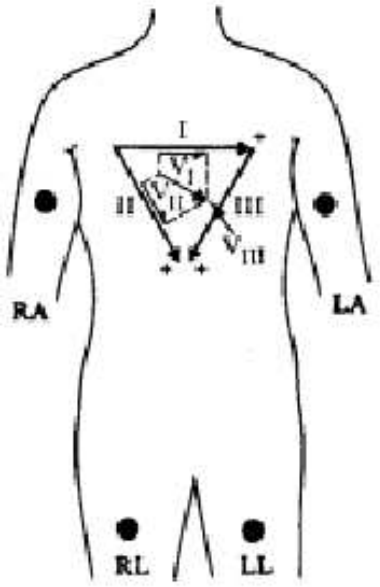


Figura 2.6. *Conexión de los electrodos en el sistema estándar [1].*

En el plano transversal se registran las llamadas derivaciones precordiales, ubicando los electrodos en varias posiciones definidas, como se muestra en la Figura 2.7 (a), mediante la obtención del voltaje entre estos electrodos y el "Terminal Central de Wilson" se obtienen estas derivaciones particulares. En la Figura 2.7 (b) también se muestran las direcciones de los vectores obtenidos en cada una de las posiciones.

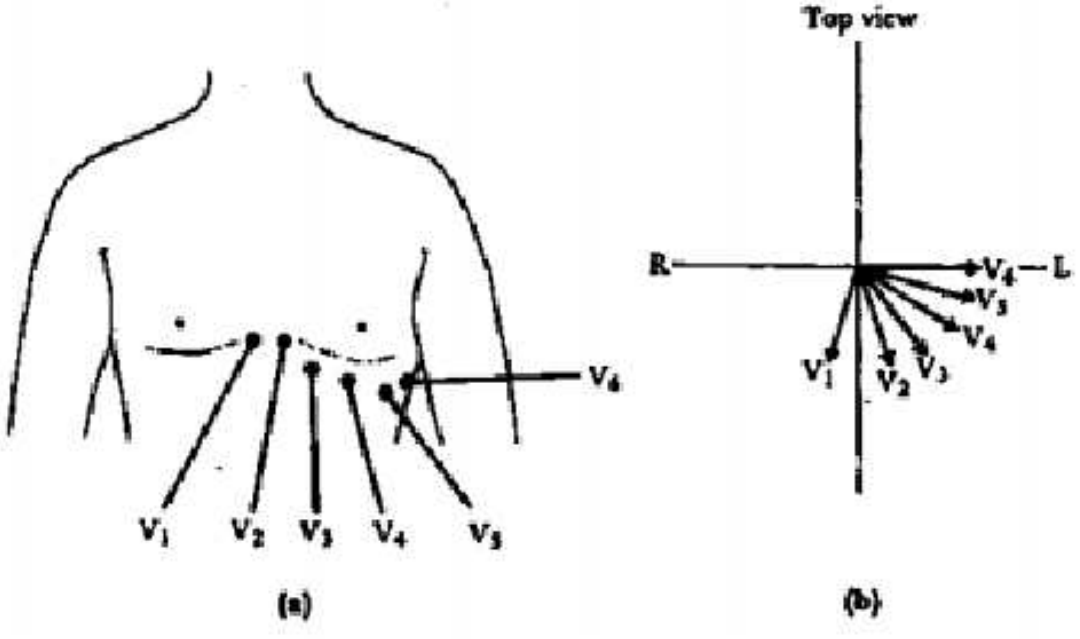


Figura 2.7 *a) Posición de las derivaciones precordiales.*
b) Dirección de los vectores obtenidos. [1].

2.2 Características generales de las señales bioeléctricas

Las señales que tratamos son potenciales bioeléctricos. Los del cuerpo humano o de cualquier animal raramente son determinísticos. Sus magnitudes varían con el tiempo, incluso cuando todos los factores que los originan están controlados. Las medidas pueden variar enormemente entre diferentes individuos aunque estos estén sanos y las condiciones de medición sean las mismas. Esto quiere decir que los valores pueden ser muy diferentes para diferentes personas aunque sean valores normales.

Las señales que son estudiadas en un cuerpo humano son las que detallamos a continuación. También exponemos sus rangos de valores típicos:

Señal	Magnitud	Ancho de banda
ECG (electrocardiograma)	0,5 a 4 mV	0,01 a 250 Hz
EEG (electroencefalograma)	5 a 300 μV	0 a 150 Hz
EGG (electrogastrograma)	10 a 1000 μV	0 a 1 Hz
EMG (electromiograma)	0,1 a 5 mV	0 a 10000 Hz
EOG (electroculograma)	50 a 3500 μV	0 a 50 Hz
ERG (electroretinograma)	0 a 900 μV	0 a 50 Hz

Tabla 2.1 – *Características de diferentes señales bioeléctricas*

Estos valores muestran la ganancia y la banda de frecuencia que debe manejar cada tipo de equipo de medida. Organizaciones como la American Heart Association recomiendan que el ancho de banda de los equipos destinados a captar señales ECG sea de al menos de 0,1 a 100 Hz, con lo cual se garantiza que las deformaciones serán menores al 10%.

En el caso del diseño de un Holter hay otros parámetros que nos obligan a utilizar un ancho de banda menor. La información que pretende capturar un Holter de 24 horas es diferente a la capturada por un electrocardiógrafo, en un estudio Holter no se intenta estudiar en detalle la forma de la onda, sino que se buscan eventuales anomalías que ocurren a lo largo del día. Además, los electrocardiógrafos realizan el registro sobre un paciente en quietud, mientras que el estudio Holter se realiza en un paciente totalmente activo, mientras realiza sus tareas habituales. Para reducir el efecto de esos artefactos de movimiento, se acepta reducir el ancho de banda de la señal adquirida, en baja frecuencia se acepta hasta 0,6 Hz y en alta frecuencia entre 40 y 60 Hz.

A continuación veremos los diferentes factores que dificultan la captación de las señales, y orientándonos a las de ECG iremos viendo las implicaciones a imponer en el Holter para que el registro de éstas sea el correcto y no tenga deformaciones.

2.3 Interferencias en la amplificación de la señal electrocardiográfica

El bajo nivel de la señal a registrar podría hacernos pensar que el único problema relacionado con su captación es el de desarrollar un amplificador con suficiente ganancia para acondicionar la señal para su posterior digitalización. Sin embargo la dificultad fundamental consiste en reducir al mínimo las interferencias que se encuentran mezcladas con la señal deseada. Estas señales interferentes en nuestras medidas muchas veces tienen un nivel de tensión miles de veces superior a la que queremos medir.

Estas interferencias se pueden clasificar de la siguiente forma:

Interferencias externas al equipo de medida:

1) Interferencias capacitivas:
   Acoplamiento capacitivo con el paciente.
   Acoplamiento capacitivo con los conductores y el equipo.
2) Interferencias Inductivas.
3) Originadas por la interfaz electrodo-electrolito-piel.
4) Debidas a otros potenciales bioeléctricos.
5) Debidas a otros sistemas fisiológicos
6) Debidas a cargas electrostáticas.

Interferencias internas al equipo de medida:

7) Provocadas por el transformador de la fuente de alimentación (en el caso que haya).
8) Debidas al rizado de la fuente de alimentación.
9) Ruido generado por los componentes electrónicos.

Las interferencias externas son todas aquellas que tienen su origen fuera del equipo amplificador. La principal fuente de interferencia de este tipo es, sin duda, la red de distribución de energía eléctrica, donde sus 50 Hz se introducen por acoplamiento inductivo y capacitivo en nuestro sistema.

Otras interferencias importantes, externas a nuestro equipo, son las generadas por biopotenciales de diferente origen al que queremos medir. En el caso de medir los potenciales eléctricos del corazón, encontramos que el movimiento muscular de brazos, piernas, o el respiratorio interfieren en nuestras lecturas. Al registrar un ECG durante 24 horas (Holter), el paciente realiza sus actividades normales diarias, por lo que este tipo de interferencia es muy común.

Con estos comentarios quedan más claros los problemas que se deben enfrentar en el diseño de un Holter. Muchos se atienden en la parte analógica y otros luego de la digitalización de la señal.

Ahora veamos, de forma general, cuáles son las acciones a tomar en el diseño y uso de los equipos, para evitar al máximo este tipo de interferencias:

1) Interferencias capacitivas:

El acople capacitivo que existe entre los elementos del sistema y la red eléctrica provoca la aparición sistemática de 50Hz en los registros bioeléctricos.

Hay que diferenciar entre el acople capacitivo con el paciente y el acople capacitivo con los conductores y el equipo.

- Acoplamiento capacitivo con el paciente:

Este es el origen más importante de todos los ruidos y condiciona en gran medida las primeras etapas de los equipos de medidas.

El cuerpo humano y los conductores de la red eléctrica forman armaduras, que con el aire entre ellas a modo de dieléctrico, conectan el paciente se a tierra y a la tensión de la red mediante dos condensadores.

Para el caso de un paciente bien aislado y acostado sobre una cama conectada a tierra, se pueden aproximar el valor de estos condensadores como:

Cr = 2 pF (capacidad entre el paciente y la red)
Cm = 520 pF (capacidad entre el paciente y tierra)

Analizando un simple circuito con forma de divisor resistivo, se estima que el paciente queda a una tensión de 2.5 voltios respecto a tierra.

Queda claro que esta tensión es muy alta respecto a los pocos milivoltios de un ECG, por lo que se puede entender las dificultades que plantea en la medida. Cuando se trata de equipos portátiles, como el caso del Holter, que no se encuentran conectados a la red eléctrica porque se alimentan con baterías, estos efectos no se manifiestan con tanta intensidad, dado que el cuerpo humano y el Holter son afectados de igual forma por la interferencia, y no afecta tanto a las medidas.

- Acoplamiento capacitivo con el equipo de medida:

El acople capacitivo de la red eléctrica con los cables que conducen la señal al equipo hará que aparezcan corrientes de desplazamiento. Estas corrientes fluirán a tierra a través de las interfaces electrodo-piel, el cuerpo del paciente y el tercer electrodo del amplificador, produciendo tensiones en modo común y diferencial.

Para reducir estas corrientes, el caso más general es utilizar conductores apantallados con la pantalla conectada a masa. Esta conexión hace que el acople capacitivo se dé con las pantallas y éstas derivan las corrientes interferentes de 50 Hz a tierra (o a masa).

Este apantallamiento también trae consecuencias en la disminución de la impedancia de entrada de modo común del amplificador, por lo que modificaciones como el colocar lo que se denomina una “guarda activa” (ver 2.4) logran dar una mejor solución.

2) Interferencias Inductivas:

Los campos magnéticos que produce la red eléctrica atraviesan los bucles de cables conectados a ADQCAR, lo cual induce tensiones de 50 Hz (ley de Lenz). Para disminuir estos efectos, como la inducción depende del área del bucle, entonces se recomienda trenzar los cables.

3) Interferencias originadas por la interfaz electrodo-electrolito-piel:

Al colocar un electrodo en contacto con la piel a través de un electrolito se produce una distribución de cargas entre la interfaz electrodo-electrolito que da lugar a la aparición de un potencial. Si el electrodo se mueve respecto del electrolito, lo cual es comprensible para un Holter, se producirá una alteración en la distribución de la carga que provocará una variación transitoria en el potencial mencionado. De la misma forma, en la interfaz electrolito-piel sucederá un caso similar. Estos problemas generan interferencias a señales muy bajas (menores a 1 Hz).

Este problema se puede disminuir utilizando electrodos que permanezcan más fijos a la piel, o que presenten polarizaciones menores como los de Ag/AgCl.

Para los Holter, como el paciente está en movimiento continuo este problema es más grave y se suele filtrar un poco más las frecuencias bajas, pero de modo que no distorsione la señal medida. En un equipo digital como ADQCAR, que incluye filtrado digital, este parámetro podrá ser estudiado mejor en la práctica y luego cambiar la frecuencia de corte, modificando el programa y no el equipo, asegurándose previamente que no sature la etapa de entrada analógica.

4) Interferencias debidas a otros potenciales bioeléctricos:

La actividad de otros potenciales bioeléctricos presentes en el mismo paciente es una fuente de interferencia difícilmente evitable. Por ejemplo las interferencias del ECG materno en el registro del ECG fetal, o las del EMG (electromiograma) sobre el ECG. Estas últimas afectan en gran medida cuando un electrocardiograma es tomado en una prueba de esfuerzo físico, pero de forma similar afecta cuando el registrador del ECG es un Holter que lleva por 24 horas.

No es posible evitar estas interferencias, pero se puede al menos disminuirla. Para esto, en el caso del Holter lo que se hace es colocar los electrodos en puntos donde la señal ECG es más fuerte que la provocada por otros movimientos musculares, es así que se colocan cerca del corazón pero en zonas donde el cuerpo tiene pocos músculos.

5) Interferencias debidas a otros sistemas fisiológicos:

La interacción entre los diferentes sistemas fisiológicos de los seres vivos se traduce en la aparición de interferencias sobre la señal que se desea medir. Una de las interacciones más estudiadas es la variación de las señales del sistema cardiovascular por la acción del sistema respiratorio. Esta provoca cambios de amplitud y de forma en los registros del ECG, así como una variación del ritmo

cardíaco. En el espectro de frecuencia del ritmo cardíaco se destaca una componente de 0,2 Hz aproximadamente.

6) Interferencias debidas a cargas electrostáticas:

Son las interferencias provocadas por la circulación a través de los electrodos a tierra, de las cargas electrostáticas almacenadas en el cuerpo del paciente. Esto provoca, normalmente, fluctuaciones de la línea base y en ocasiones la saturación de los amplificadores.

La generación de la carga electrostática puede tener diversos orígenes, y es almacenada en la capacidad creada entre la superficie corporal y tierra. Esto provoca la aparición de una diferencia de potencial entre el cuerpo y tierra que dependerá de la carga generada.

En equipos portátiles esta interferencia es mucho menor ya que su efecto se manifiesta por la diferencia de potencial creada en el cuerpo al existir una corriente por descarga electrostática hacia tierra.

Veamos ahora las interferencias debidas a fuentes internas al equipo:

7) Interferencias provocadas por el transformador de la fuente de alimentación:

Si la fuente de alimentación del amplificador incorpora un transformador, el rizado de 100 Hz correspondiente a la rectificación de los 50 Hz de la red, provocará interferencias que pueden ser importantes. Tampoco es problema cuando se usan baterías.

8) Interferencias debidas al rizado de la fuente de alimentación:

Esta interferencia, es debida a que el regulador de alimentación puede no tener un rechazo al rizado suficiente.

Se recomienda, para mejorar las respuestas de estos dos últimos casos, utilizar amplificadores operacionales con un alto PSRR (Power Supply Rejection Ratio), lo que indica la relación de rechazo a variaciones de la tensión de alimentación. Este dato es suministrado por el fabricante del componente.

Otra solución aceptable para estos problemas es usar baterías para la alimentación del amplificador, que además de reducir el ruido mencionado, agrega seguridad al paciente. Estas características son las que se encuentran en un Holter, ya que al ser portátil, basa su funcionamiento en una fuente alimentada por baterías.

En caso de utilizar conversores DC-DC, no existen mayores inconvenientes, siempre que la frecuencia de switcheado utilizada esté suficientemente lejos del rango de frecuencias de interés.

9) Ruido generado por los componentes electrónicos:

Los componentes electrónicos, tanto activos como pasivos, generan señales de ruido aleatorio que contaminan las medidas. En los sistemas que requieren gran amplificación este problema puede ser crítico y exige una adecuada selección de los componentes de las primeras etapas.

El diseño del circuito tiene mucha importancia. Además de intentar depender lo menos posible de la sensibilidad de esos componentes, es importante agregar filtros en las etapas apropiadas para quitar las frecuencias no deseadas.

Para redondear el tema de las interferencias en las medidas, podemos decir que hemos visto varios factores que afectan el diseño final del equipo. Algunos aportan métodos o elementos a considerar en el diseño del circuito eléctrico, en los materiales, en los cables, filtros, etc. pero otros nos obligan a restringir las características finales que tendrá nuestro ECG u otra señal que vaya a ser medida.

2.4 Seguridad en el paciente

Mientras se capta información del paciente se debe tener en cuenta su protección, para lo cual hay normas de seguridad establecidas.
Estas no dan garantía absoluta de protección al paciente, pero ayudan a minimizar los riesgos.

Los efectos que la corriente eléctrica produce sobre el cuerpo humano depende fundamentalmente de los siguientes factores:

- Magnitud de la corriente que circula
- Frecuencia
- Tiempo de duración del paso de corriente
- Zona por la que circula (superficial o profunda)

La intensidad mínima de corriente que el ser humano es capaz de detectar se define como nivel de percepción. Este nivel es variable y depende de cada persona y de las condiciones dadas. En el caso de una persona que toca dos cables con sus manos secas, y circula entre ellos una corriente, el rango en que se percibe la sensación es de entre 2 a 10 mA. En cambio, si la persona tiene las manos mojadas, una corriente de 50 Hz ya sería percibida a los 0,5 mA.

Aumentando la corriente o circulandola por contactos más cercanos a los nervios y músculos, la percepción es mayor y las alteraciones que estas corrientes pueden ocasionar también. En el rango de 9,5mA a 16mA muchos investigadores consideran que se generan contracciones musculares que pueden llegar a ser dolorosas, y aumentando más se ocasiona una pérdida del control motor del músculo afectado.

Entre los 18mA y 22mA, contracciones involuntarias de los músculos respiratorios pueden proveer asfixia si la corriente es constante. Aumentando aún más, se crea una pérdida de sincronismo en las fibras del músculo cardíaco.

Si el paciente está sometido a un potencial suficientemente diferente a tierra, una conexión a tierra del equipo podría proveer un camino de baja impedancia que permita la circulación de estas altas corrientes. También diferentes tensiones de diferentes equipos de medición conectados al mismo paciente pueden utilizar a éste para cerrar circuitos y someterlo a una corriente alta.

El Holter solo trabaja con tensiones diferenciales, por lo que rechaza las tensiones netas del cuerpo del paciente filtrándolas como de modo común.

2.5 Electrodos y cables

Los electrodos, primer elemento en todo sistema de captación de biopotenciales, tienen la misión de transformar las corrientes iónicas del cuerpo humano en corrientes eléctricas, susceptibles de ser amplificadas y tratadas.

Al poner en contacto el electrodo con la piel se forma una distribución de cargas en la interfase, originando un potencial llamado de contacto. Este potencial varía cuando se mueve el electrodo, originando fluctuaciones en la línea base de la señal bajo análisis.

Un electrodo debe presentar las siguientes características:

1. Poco invasivo para el paciente.
2. Capaz de realizar la transformación de corrientes con pocas pérdidas.
3. Biocompatible, que no produzca alteraciones en el paciente.
4. Rigidez mecánica.
5. Impedancia baja.
6. El potencial de contacto debe ser estable, sin variaciones y el menor posible.
7. Estable en el tiempo, para que no pierda sus propiedades.

Hoy en día, fabricantes como Hewlett Packard o 3M, ofrecen una gran variedad de electrodos que cumplen las características anteriores, tanto desechables (que ya incorporan el gel y un adhesivo para sujetarlo al paciente), como reutilizables (de ventosa o cazoleta). Algunos ejemplos de electrodos se muestran en la Figura 2.8.

Otro aspecto importante radica en disminuir al mínimo posible la impedancia de unión con el paciente, formada por el electrodo y la piel. Se debe limpiar con alcohol la superficie de la piel y eliminar en lo posible la capa de células muertas, que es la que tiene una mayor impedancia, mediante un suave frote. Además se aplica un gel en los electrodos. Este gel puede venir incluido en los electrodos, o bien se puede adquirir en tubos para los electrodos reutilizables. Será necesario dejar que se absorba parcialmente el gel para que la impedancia de la piel (dermis) disminuya.

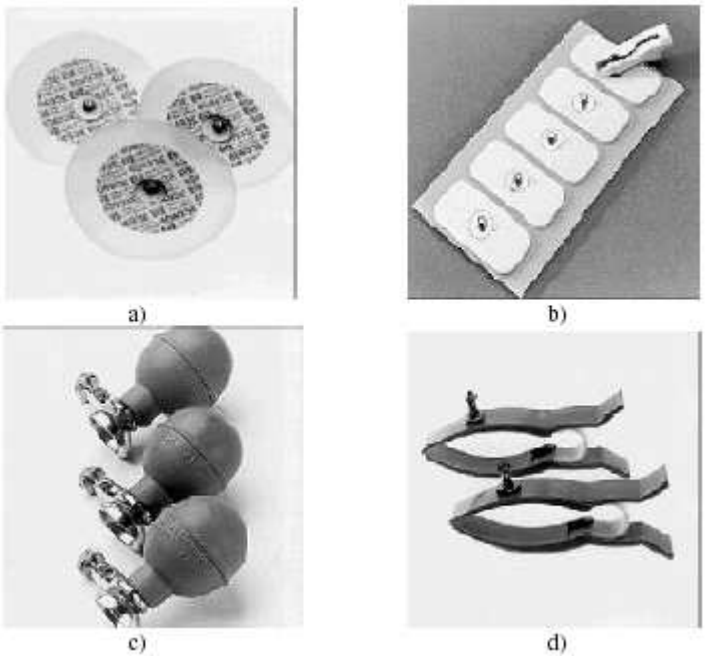


Figura 2.8 - *Diferentes tipos de electrodos: a) y b) desechables, c) reutilizable, d) pinza para las extremidades. [1].*

Los cables se conectan a los electrodos mediante un corchete o pinza de cocodrilo en los desechables o mediante una banana en los reutilizables. Deben ser flexibles, de muy baja impedancia y trenzados, para evitar los lazos de corriente de superficie grande. Otra posibilidad es utilizar cables apantallados, conectando la malla a un voltaje medio, con el fin de disminuir el acoplo capacitivo en los cables (esta técnica es la llamada “guarda activa”).

Para el caso de los Holter, como en nuestro proyecto, los utilizados son del tipo desechable, y se colocan en el pecho del paciente de forma estándar.

2.6 Nomenclatura de las ondas ECG

Cuando se registra un ECG, se inscribe una serie de ondas por cada ciclo cardíaco. Einthoven denominó a estas ondas P, Q, R, S y T, de acuerdo con su orden de inscripción, correspondiendo la onda P a la despolarización auricular, el complejo QRS a la despolarización ventricular y la onda T a la repolarización ventricular (Figura 2.9). En ocasiones, a continuación de la onda T se graba una pequeña onda llamada U.

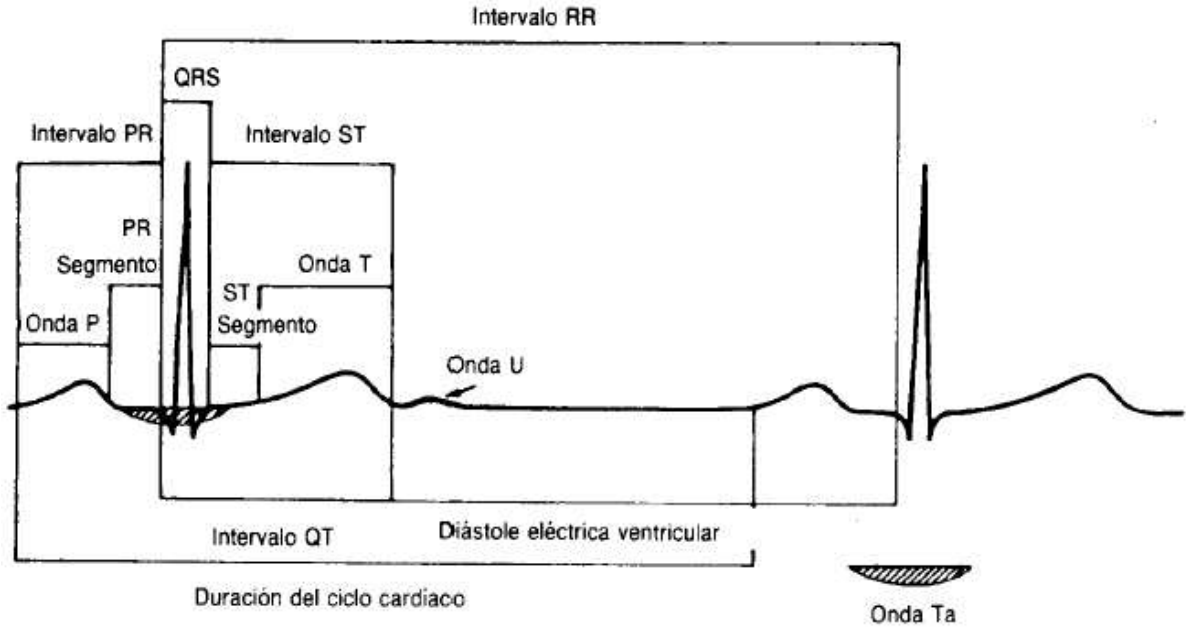


Figura 2.9 - *Relaciones temporales entre las diferentes ondas del ECG y nomenclatura de los diferentes intervalos y segmentos [1].*

CAPITULO 3 – RESUMEN

3.1 Características de ADQCAR

El estudio de las características deseables para un equipo de adquisición de señales biológicas prolongadas, y en particular de un Holter, nos llevó a definir las siguientes especificaciones de ADQCAR.

- **Etapa de entrada:** Amplificación, cancelación de ruido y filtrado de los 3 canales. Señal filtrada en una banda de *0.5 Hz* a 60 *Hz*.

- **MUX/ADC:** Multiplexado y muestreo a 12 bits y 200 muestras/seg. Conversor Analógico-Digital y multiplexor, integrado en un mismo chip programable.

- **Microcontrolador:** Control de los procesos, filtrado digital y compresión de las señales. El microcontrolador elegido es el MMC2001 de Motorola (32bits, RISC, incluye interfaz SPI).

- **Memoria:** Almacenamiento de las señales en una tarjeta de memoria flash extraíble, formato MultiMedia Card (MMC), con capacidad entre 8 y 256 MB.

- **Volcado de datos a PC:** Lectura de la tarjeta de memoria, y adaptación de los datos a un formato de archivo estándar. La tarjeta MMC se lee desde el PC con el dispositivo USB ImageMate.

- **Alimentación:** Baterías y conversor DC-DC, dependiendo del tiempo de registro y los algoritmos a implementar. Se utilizaron componentes de bajo consumo de 3V, lo que permite una operación continua de 24 horas con dos baterías AA.

3.2 Diagrama de bloques y funcionamiento general

A continuación mostramos los bloques de ADQCAR y explicamos a grandes rasgos su funcionamiento:

En la Figura 3.1, se pueden ver los bloques comentados en la sección 3.1. A la etapa de entrada analógica se le conectan los cables que provienen de los 7 electrodos del paciente. Cada par de electrodos corresponde a un canal diferencial de ECG, y el séptimo electrodo se utiliza para fijar una referencia.

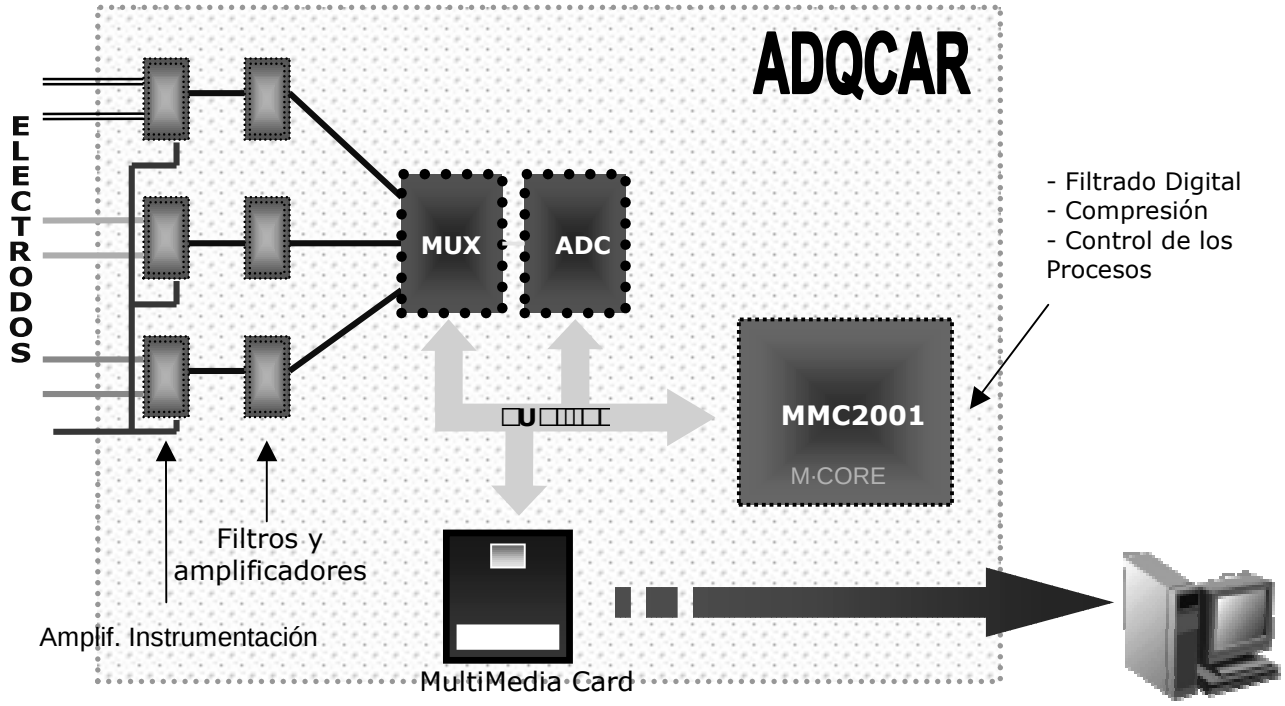


Figura 3.1 – *Diagrama de Bloques de ADQCAR*

Los amplificadores de instrumentación toman las pequeñas tensiones diferenciales de cada par de electrodos (canales). La amplitud de la señal de entrada es del orden de ± 5 mV. Los amplificadores de instrumentación también adaptan la impedancia del circuito para posteriormente filtrar y amplificar cada canal. Las señales, luego de amplificadas, están acotadas entre 0 y 3.3V.

El multiplexor transmite secuencialmente cada canal hacia el conversor A/D. Esta operación es controlada por el microcontrolador a través del BUS SPI. Por este BUS también se envían las muestras digitalizadas hacia el microcontrolador.

En el microcontrolador se realizan un filtrados digitales pasabajo, que determina el ancho de banda final de la señal que será almacenada.

Además de controlar procesos, el microcontrolador puede realizar otras tareas, como comprimir la señal adquirida, para optimizar el uso de memoria Flash y reducir el tiempo necesario para descargarlos en el PC. Se estudiaron varios algoritmos de compresión y se llegó a simular uno de ellos en Matlab, que sería adecuado para adaptar a futuras versiones de ADQCAR.

Manejando un buffer de muestras, el microcontrolador las graba en la tarjeta MMC usando un formato propio, diseñado para almacenar datos secuenciales. Luego, usando el bus SPI, se graban los datos en lotes de 512 bytes, aproximadamente 340 muestras si no se comprimen.

Al terminar la toma de datos del paciente durante 24 horas, se retira la MMC del equipo y se coloca en el lector ImageMate, que por un enlace USB se conecta al PC.

En este último, el programa ADQCAR permite leer el formato grabado en la tarjeta descargándolo al disco duro y guardándolo en un formato estándar compatible en el entorno del sistema operativo Windows (NT, 2000, XP).

3.3 Integración física de los diferentes bloques

Antes de profundizar en los siguientes capítulos, donde se explica con más detalle el desarrollo de ADQCAR, veremos genéricamente su estructura física, mostrando como se integran los diferentes bloques mencionados.

ADQCAR consiste en dos circuitos impresos superpuestos, donde se encuentran montados la mayoría de los componentes. Baterías, conectores, zócalos para la tarjeta removible de memoria, interruptores de control y un led indicador completan el equipo. Una caja metálica permite asegurar un soporte firme para todo el conjunto del sistema.

La Figura 3.2 muestra una perspectiva de las partes que constituyen ADQCAR:

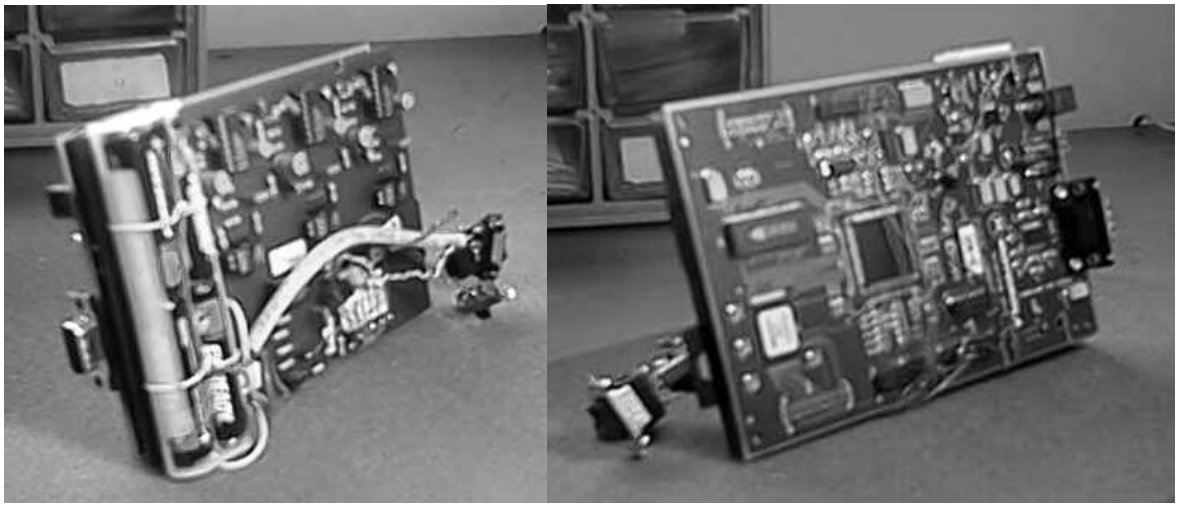


Figura 3.2 – *Las dos placas de ADQCAR aún sin colocar en su caja: (Izq.: Vista de la placa que incluye los bloques Etapa de Entrada, MUX/ADC y memoria extraíble. Der.: Vista de la placa digital con el microcontrolador)*

CAPITULO 4 – ETAPA DE ENTRADA

4.1 Introducción

Esta etapa cumple funciones de amplificación, filtrado y adaptación de las señales de cada uno de los tres canales obtenidos a partir de los electrodos, para poder luego continuar con la digitalización, procesamiento y almacenamiento de las mismas.

El diagrama de bloques para cada etapa de entrada se representa en la Figura 4.1. Este diagrama comienza por los electrodos que son transductores que transforman las corrientes iónicas del cuerpo en una señal eléctrica, seguido del de protección contra desfibrilación (no implementado). A continuación se encuentra un preamplificador diferencial de gran impedancia de entrada. Luego el filtrado analógico con ganancia para obtener las frecuencias de interés y por último un ajuste de ganancia para obtener una amplitud adecuada en la señal de salida previo a la digitalización.

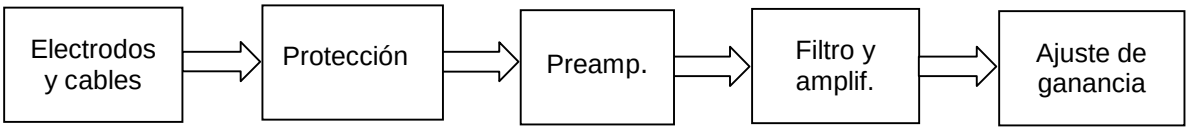


Figura 4.1 - *Diagrama de bloques de la etapa de entrada*

4.2 Electrodos y cables

Los electrodos son el primer elemento en todo sistema de captación de biopotenciales y tienen la finalidad de transformar las corrientes iónicas del cuerpo en señales eléctricas capaces de ser amplificadas. Como se detalló en 2.3 existe un potencial de contacto electrodo-piel, el cual puede modelarse eléctricamente según se muestra en la Figura 4.2.

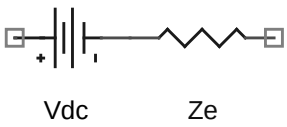


Figura 4.2 - *Potencial de contacto electrodo piel.*

Como veremos más adelante, el preamplificador a utilizar es un amplificador de entrada diferencial, por lo que tendremos un electrodo por cada entrada. Si los potenciales de contacto fueran iguales, no existiría ningún problema, pero hemos

observado en la práctica que esto no es así, existiendo una tensión de continua que también se amplifica. Esto va a limitar la ganancia del preamplificador para evitar que sature.

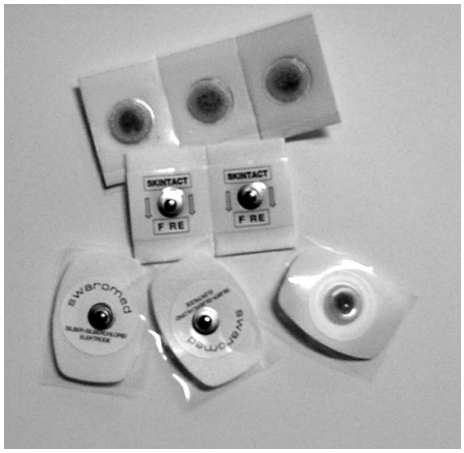


Figura 4.3 – *Electrodos usados en los ensayos de ADQCAR*

En las pruebas realizadas pudimos constatar que los mejores electrodos son los desechables que incorporan gel y adhesivo de sujeción (Figura 4.3). Además de ser los más cómodos para el paciente.

Un problema no menor a tener en cuenta son los cables a utilizar. Los mejores resultados se observaron con cables apantallados. Estas pantallas están conectadas a la masa del equipo.

Por tratarse de un prototipo, se utilizaron cables apantallados de audio, de gran flexibilidad, aunque de diámetro mayor que los originales para ECG. Con estos no se observaron diferencias en los ensayos respecto a los originales.

Para conectar los cables a los electrodos utilizamos broches de presión metálicos (los utilizados en prendas de vestir) soldados a pinzas cocodrilo (Figura 4.4).

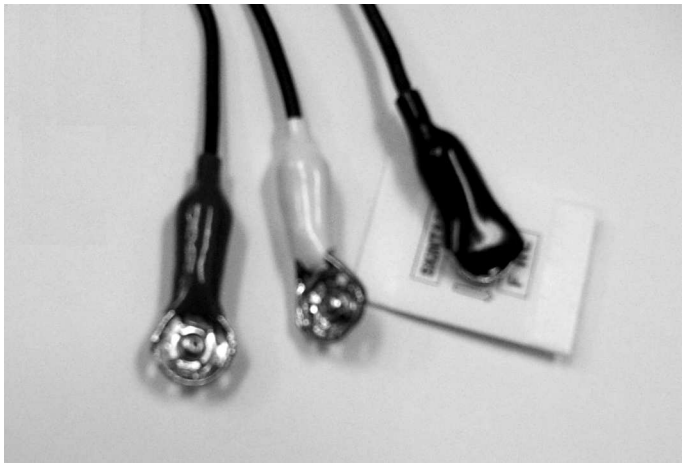


Figura 4.4 – *Terminales de los cables con broches de presión metálicos*

4.3 Protección

En caso de que un paciente que se encuentre en estudio requiera una desfibrilación, esta debe realizarse lo antes posible, y por consiguiente, no se puede perder tiempo en retirar el Holter. Por otro lado, si pensamos en futuros desarrollos, en los cuales el tamaño sería muy reducido o incluso implantable, se ve la necesidad de proteger al equipo contra una desfibrilación.

Una desfibrilador descarga una energía máxima de 360 Joules, con una tensión máxima de 9kV [1]. Dependiendo de las condiciones eléctricas del paciente esta descarga demora unos pocos milisegundos.

Pensamos primeramente en utilizar un circuito como el de la Figura 4.5, el cual se colocaría en los electrodos 2 a 2 para proteger al Holter de forma independiente del camino que tomen las corrientes.

En este circuito R1 y R2 representan los cables y electrodos y R3 y R4 son pequeñas. Algunos diodos Zener podrían evitarse debido a que los preamplificadores utilizados están provistos de protección (±40V).

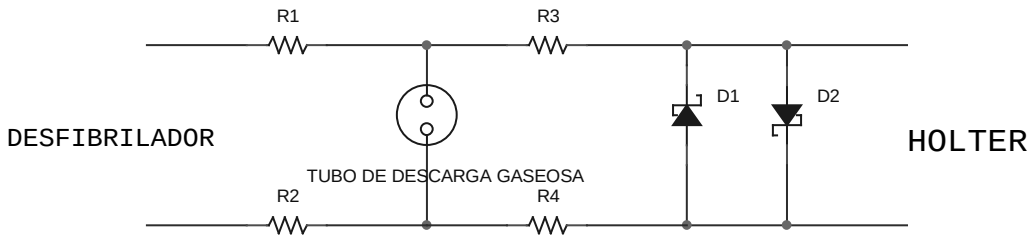


Figura 4.5 – *Circuito inicial propuesto para protección contra desfibrilador*

De este modo el circuito de protección se podría realizar de forma totalmente independiente al resto del circuito, incluso se pensó en incorporar este al conector de los 7 cables que van hacia los electrodos.

Comenzando por la búsqueda de componentes, encontramos tubos de descarga gaseosa en el mercado local, los cuales se alejaban un poco de las especificaciones requeridas y tenían un precio muy elevado, por lo que se decidió buscar estos componentes en Internet.

Se resolvió no llevar a cabo la implementación de esta protección para no extender los tiempos del proyecto, ya que por un lado el estudio y ensayos de esta etapa lo iban a requerir y además perfectamente se podía incorporar al Holter en una segunda versión de prototipo, sin modificar el resto del circuito.

Además no se disponía en el NIB de herramientas para realizar los ensayos, y en caso contrario, hubiera representado un riesgo para nosotros, los estudiantes y también nuestros compañeros.

4.4 Preamplificador

La etapa de preamplificación es la más delicada del circuito ya que hay que tener en cuenta las interferencias en la señal.

Primeramente, por tratarse de un equipo portátil (sin conexión a tierra), pensamos en una amplificación no diferencial. Recordemos que una amplificación de este tipo amplifica una señal respecto a una referencia, siendo, en general, esta referencia la utilizada a lo largo de todas las etapas. Esto hace que los canales no puedan ser independientes y por este motivo desechamos esta configuración.

Elegimos entonces una configuración con amplificación diferencial utilizando un amplificador de instrumentación, la que además nos permite reducir el ruido modo común no deseado.

Elección del Amplificador de Instrumentación:

Al seleccionar el modelo de amplificador de instrumentación a utilizar (uno en cada uno de los 3 canales de entrada), buscamos aquel que se acercara lo más posible a los siguientes requerimientos:

1) Bajo consumo.
2) Ganancia seleccionable
3) Tensión de alimentación de +/-1.5 V o menor.
4) Gran impedancia de entrada
5) Alto rechazo al modo común (CMRR)
6) Excursión completa a la salida (rail to rail).

1) El bajo consumo es un requisito general de todo el equipo, por que tanto éste como todos sus componentes se eligieron de bajo consumo.

2) Si bien no dejaría de funcionar si la ganancia es fija, nos deja la libertad de elegirla, para tomar la más adecuada frente al compromiso de menor relación señal a ruido y evitar la saturación como se verá en unas líneas más adelante.

3) Cuanto menor sea la tensión de alimentación, menor será el consumo del equipo. La elección de esta tensión de alimentación no implica que se utilice en el prototipo final, pero deja las puertas abiertas para usarla.

4) La impedancia de entrada debe ser grande frente a las variaciones de la impedancia de los electrodos, pero además para tener la posibilidad de introducir circuitería previo a esta etapa si fuera necesario (esto se detalla más adelante).

5) Si bien podríamos pensar que no va a haber modo común por tratarse de un equipo portátil, queremos asegurar su rechazo y así evitar posibles fuentes de ruido.

6) La excursión completa nos mejora el rango de no saturación debido a artefactos de movimiento, posibilitando una ganancia mayor en esta etapa.

Surgieron 3 propuestas que se detallan en la tabal 4.1.

Componente	**AD627**	**INA118**	**LT1101**	**Unidad**
Fabricante	*Analog Devices*	*Burr Brown*	*Linear Technology*	
Corriente de alimentación máxima	85	385	145	μA
Selección de ganancia	variable	variable	10 ó 100	-
Tensión mínima de alimentación	±1.1 ó +2.2	±1.35 ó +2.7	±0.9 ó +1.8	V
Impedancia de entrada	20	10	5	GΩ
CMRR	95	90	100	dB
Excursión de salida Vcc = 5V	4.9	3.65	4.3	V

Tabla 4.1- *Comparación ente amplificadores de instrumentación de bajo consumo.*

Además destacamos la diferencia en cuanto a circuito interno que tiene cada uno de los componente:

- AD627: compuesto de 2 operacionales de tal forma que se puede obtener la tensión de modo común del punto medio de la resistencia de ganancia.
- INA118: circuito convencional de 3 operacionales. Esto da la ventaja de manejar un circuito bien conocido. También al igual que el primero permite, en caso de ser necesario utilizar la resistencia de ganancia para obtener la tensión de modo común.
- LT1101: compuesto de 2 operacionales en cascada. Si bien se puede variar la ganancia entre 10 y 100, esto requiere de 2 resistencias lo que trae una pérdida en el CMRR con la dispersión entre éstas.

De esta comparación se desprende que el más conveniente es el AD627 de Analog Device que es usado en instrumentación biomédica de bajo consumo y en adquisidores de datos de baja potencia.

Dado que este tipo de componentes no se encuentran en el mercado local, realizamos su búsqueda por Internet, pero no encontramos distribuidores de Analog Device a América del Sur por menos de 100 unidades ni se pudo obtener muestras del propio fabricante.

En la importación realizada, igualmente se trajeron el INA 118 y el LT1101.

Realizamos pruebas con ambos, obteniendo un circuito mucho más estable con el INA118 que con el LT1101. Por esta razón nos inclinamos por el primero. Si bien es el que más consume, está específicamente diseñado para la obtención de ECG y estamos tratando con un amplificador de instrumentación clásico. Además presenta la ventaja de tener la misma topología de encapsulado que el AD627, lo que

permitiría migrar a este en caso de ser necesario con solo cambiar el componente y el valor de la resistencia de ganancia.

Elección de la ganancia:

En la Figura 4.6 se puede ver el diagrama interno del integrado INA118 con los tres amplificadores operacionales, resistencias y protecciones contra sobre voltajes. De este diagrama se desprende que la ganancia del amplificador se puede expresar:

$$G = 1 + 20k / Rg$$

por lo que vemos que Rg es quien fija la ganancia.

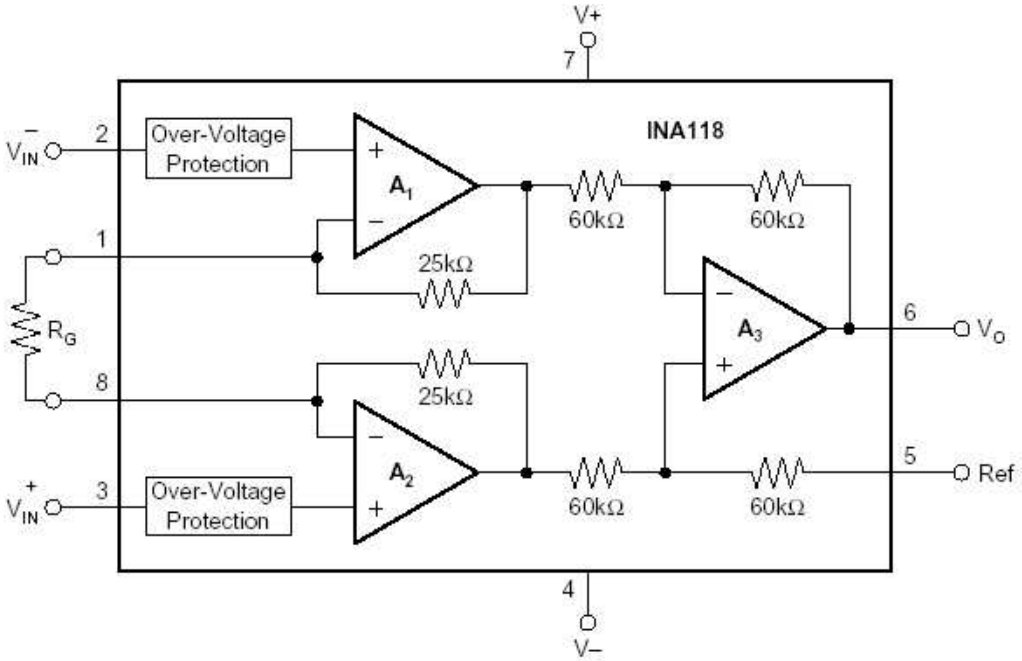


Figura 4.6 – *Diagrama interno del amplificador de instrumentación INA118.*

La señal ECG tiene una excursión acotada por ±4 mV, si planteamos una excursión a la salida acotada por ±3V, la ganancia requerida es del orden de 750. Igualmente para una excursión de ±1.5V la ganancia será 375.

Es deseable la mayor ganancia posible en esta etapa para poder así minimizar la relación señal-ruido. Pero debemos ser prudentes pues, como ya se comentó, el potencial de contacto de los electrodos limita esta ganancia.

Ensayamos entonces el comportamiento del amplificador, alimentado con ±1.5V, para varias ganancias, observándose que con valores del orden de 200 ~ 250 la salida quedaba muy inestable, saturando al menor movimiento del paciente. Por esto fijamos para el preamplificador una ganancia de 26, eligiendo también las correspondientes a cada etapa (Tabla 4.2).

Preamplificador	Filtro	Ajuste
26	8	1.5 a 6

Tabla 4.2 - *Ganancia elegida para cada etapa.*

Con esta elección obtenemos una ganancia aproximada ajustable entre 300 y 1200.

Configuración elegida:

Con el fin de evitar el movimiento de la línea base y por lo tanto la saturación del INA118, estudiamos varias configuraciones según se muestra en la Figura 4.7.

La configuración indicada en la Figura 4.7.a, es la más sencilla, pero tiene 2 desventajas, la primera que en continua tiene ganancia 1 (y no 0) y la segunda, el tamaño del condensador Cg, el cual debe ser electrolítico bipolar de más de 400 µF si buscamos una ganancia por lo menos mayor a 25 en el INA118. Un condensador de éstas características tiene dimensiones considerables y alto costo, además de ser difícil de conseguir, motivo que llevó a desechar esta opción.

En la Figura 4.7.b se presenta otra configuración que filtra las oscilaciones previo a la entrada del INA118. El operacional se encarga de darle un camino a las corrientes de polarización de este amplificador. Es claro que esto disminuye la impedancia de entrada a Rin1 + Rin2, por lo que estas resistencias deberán ser lo más grande posible, lo que trae un compromiso frente a las corrientes de polarización del amplificador limitando su ganancia. Dado que esta configuración debe ser estudiada con mucho detenimiento para garantizar su funcionamiento, lo que implicaría mucho tiempo, preferimos pasar a estudiar primeramente la de la Figura 4.7.c que es más sencilla y reanudar su estudio más adelante si es necesario.

Por último se implementó la configuración de la Figura 4.7.c. Si se estudia el circuito interno del amplificador de instrumentación, puede observarse que la etapa amplificadora de éste debe limitarse para no tener saturaciones frente al movimiento de la línea base. En la práctica, si bien la ganancia se vio algo limitada, se obtuvieron resultados muy satisfactorios, tanto en estabilidad como en simplicidad del circuito, motivo por el cual se eligió. Cabe destacar que es sugerida en la hoja de datos del INA118.

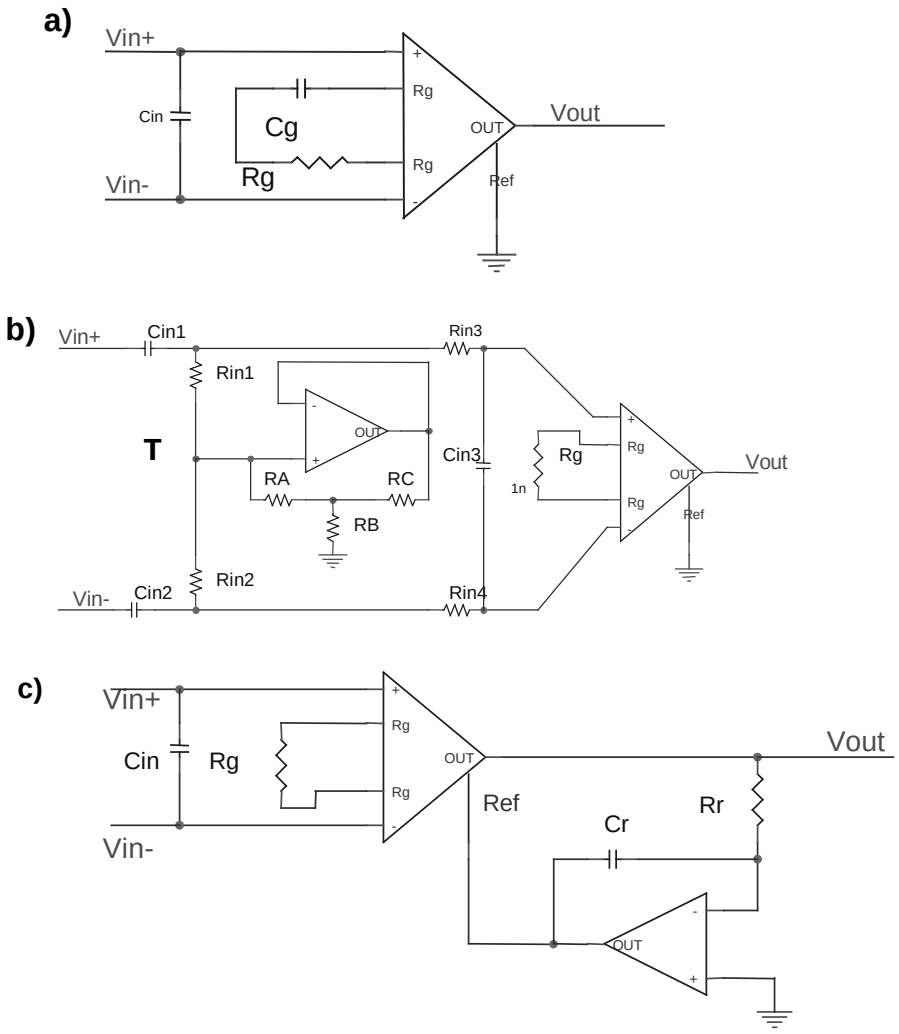


Figura 4.7 – *Configuraciones estudiadas para la etapa preamplificador.*

La transferencia de la configuración elegida es:

$$G = (1 + 50K / Rg) * Rr.Cr. S / (1 + Rr.Cr.S)$$

Se puede observar de la transferencia que se trata de un filtro pasa altos de primer orden. Más adelante discutimos sobre los detalles de este filtrado.

Es fácil ver aquí que las tensiones de offset no se amplifican a la salida y que no es necesario incluir una resistencia en paralelo con el condensador (como se suele hacer en un integrador).

Elección del amplificador de realimentación:

Detallamos a continuación los requerimientos para este operacional:

1) Bajo consumo
2) Alta impedancia de entrada

1) El bajo consumo es un requisito general de todo el equipo, por que tanto éste como todos sus componentes se eligieron de bajo consumo.
2) Una gran impedancia de entrada nos permitiría utilizar resistencias del orden de los 10 MΩ, reduciendo así el consumo.

Componente	LT1079	OPA4241	TLV2404	Unidad
Fabricante	*Linear Technology*	*Burr Brown*	*Texas Instruments*	
Corriente de alimentación máxima (por operacional)	50	25	<1	μA
Tensión mínima de alimentación	±1.1 ó +2.2	±1.35 ó +2.7	±1.25 ó 2.5	V
Impedancia de entrada	400	10	300	MΩ
Corriente de polarización	10	-25	0.3	nA
Corriente de offset	50	2000	250	pA
Tensión de offset	0.3	0.4	1.2	mV
Excursión de salida Vcc = 3V	2.2	2.8	2.85	V
Slew Rate	70	10	2.5	V/ms

Tabla 4.3 - *Comparación de operacionales (se consideran valores extremos)*

De los tres componentes de la Tabla 4.3, se eligió el TLV2404 de Texas Instruments fundamentalmente por su micro consumo.

Séptimo electrodo:

La configuración de preamplificador elegida (Figura 4.7.c) exige para su funcionamiento un camino para las corrientes de polarización de entrada al amplificador de instrumentación. Este camino se le da por medio de un electrodo adicional (el séptimo electrodo).

Este electrodo puede conectarse:

- por medio de una realimentación activa
- directamente a masa (punto medio de la alimentación).

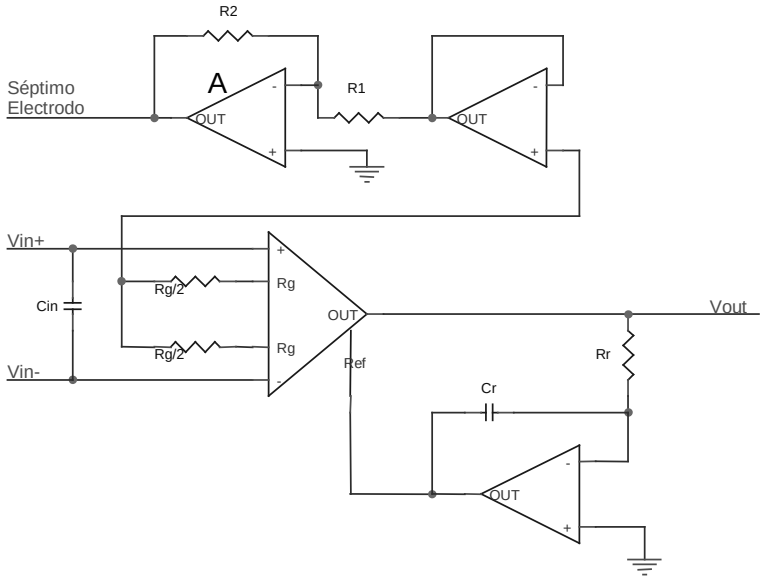


Figura 4.8 - *Realimentación activa.*

La realimentación activa se utiliza para minimizar el modo común en la entrada. Si bien el acople capacitivo con tierra no se da en el circuito por tratarse de un equipo portátil, para descartar que el ruido de red observado a la salida de esta etapa (50 Hz y como consecuencia 100 Hz) es de modo común, realizamos pruebas utilizando la realimentación de la Figura 4.8, sin observar mayores diferencias en la señal de salida respecto a la conexión directa.

De estas pruebas se pudo observar que existe gran inestabilidad debido a las bajas tensiones de alimentación cuando se incluye la realimentación activa alimentando con ±1.5V. Esto se debe a la saturación del operacional **A** en la Figura 4.8.

Por lo tanto se decidió no incluir esta realimentación en el circuito, conectando el "séptimo electrodo" directamente a masa.

Este electrodo tiene además, la función de referir los restantes electrodos a masa y favorecer así el rango de voltaje de entrada, evitando saturaciones frente a artefactos de movimiento.

En la Figura 4.9 se observa un diagrama para un canal indicando el camino de estas corrientes. El electrodo inferior es el denominado "séptimo electrodo".

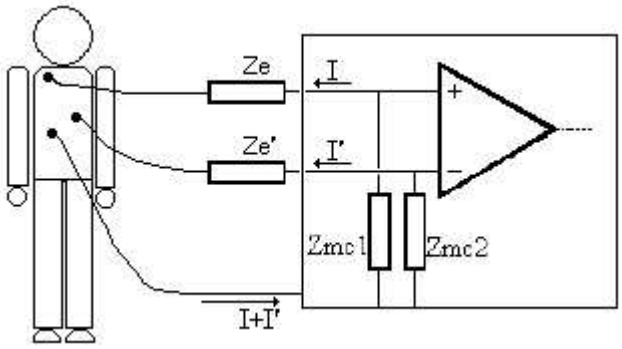


Figura 4.9 – *Camino de las corrientes de polarización.*

Diseño alternativo futuro:

Se puede ver que la configuración de la Figura 4.7.b no requiere del séptimo electrodo ya que el operacional provee de un camino para las corrientes de polarización, eliminando además tanto la tensión de continua dada por los electrodos como parte de los artefactos de movimiento, permitiendo una mayor amplificación en el preamplificador.

Sabemos que esta sería una mejora importante para un diseño futuro, pero preferimos no implementarla pues nos hubiera requerido un estudio más profundo para establecer los ajustes (CMRR, resistencia de entrada, entre otros) y eliminar ruidos no deseados, lo que por lógica hubiera demandado mucho tiempo.

Por su lado, la de 7 electrodos cumple con los requisitos primarios y ya se encontraba funcionando.

4.5 Filtro y amplificación.

Previo a la digitalización, es necesario filtrar para eliminar las bajas frecuencias (artefactos de movimiento) que modifican la línea base y las altas frecuencias para eliminar el solapamiento en frecuencia (“aliassing”). Aprovechamos la propia etapa de filtrado para amplificar la señal y llevarla al valor deseado, ya que en la etapa de preamplificación se vio limitada la ganancia. Como vimos en la Tabla 4.2 la ganancia elegida para esta etapa es 8.

Recordemos que la señal de ECG tiene un espectro entre 0.01Hz y 250Hz, pero los tipos de patología a estudiar no requieren de gran precisión en la señal. Aprovechamos este hecho para hacer el filtrado.

Para esto evaluamos distintos filtros según se detalla en la Tabla 4.4.

	Bessel	**Butterworth**	**Legendre**	**Tchebyscheff**
Pendiente de corte	Muy mediocre	Mediocre	Media	Buena
Regularidad de la curva	Excelente	Excelente	Muy buena	Ondulaciones
Orden para una determinada selectividad	Muy elevado	Elevado	Medio	Bajo
Dificultad de ajuste	Pequeño	Pequeño	Pequeño	Medio
Disparidad de los valores	Muy pequeña	Pequeña	Media	Considerable

Tabla 4.4 – *Características de los filtros polinomiales*

Filtro pasa bajos:

Estudiando las características de las señales y del filtro deseado pudimos observar que el que presenta menos rizado en la banda pasante es el filtro de Butterworth, si bien la pendiente no es de las mejores, tiene buena linealidad. Además, si observamos un espectro de frecuencias de un ECG real, a partir de los 35 a 40 Hz sus componentes no son muy significativos (Figura 4.10).

De las reuniones iniciales con el Dr. Fernando Nieto(†) y las pruebas realizadas posteriormente con diferentes frecuencias de corte, es que decidimos utilizar una frecuencia de corte de 60Hz. De esta forma evitamos filtrar la banda que tiene más información en el espectro y también el solapamiento en frecuencia.

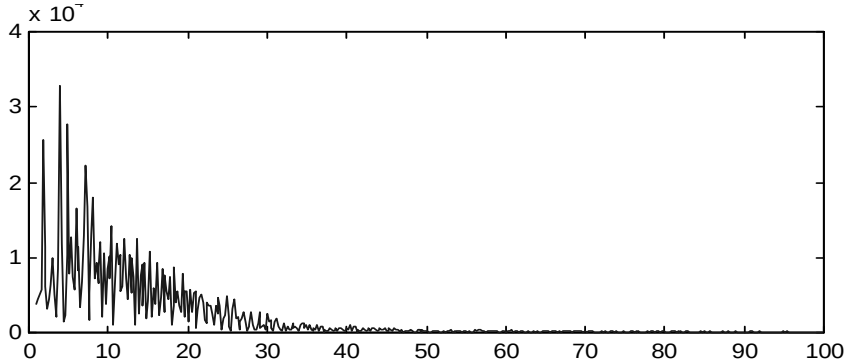


Figura 4.10 - *Espectro de frecuencias de un ECG obtenido con ADQCAR con frecuencia de corte 90 Hz.*

En un principio, se utilizó un filtro de Butterworth de orden 2 y topología “Salen-Kelly” (Figura 4.11.a), la que luego modificamos a MFB (Figura 4.11.b) por razones íntimamente relacionadas al filtro pasa altos, que lo explicaremos más adelante.

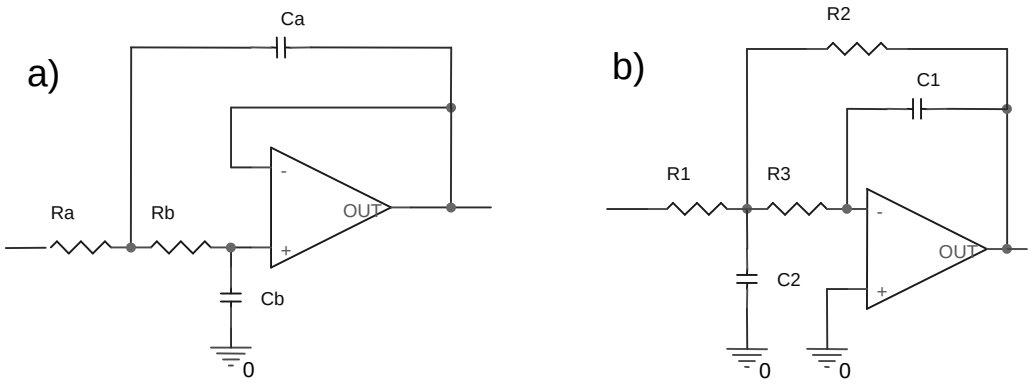


Figura 4.11- *Filtro de Butterworth de orden 2 topología: a) Salen-Kelly b) Multiple Feedback (MFB)*

El gran problema encontrado fue que los 100 Hz provenientes de la red eléctrica tenían una componente muy notoria a la salida de la etapa de entrada. Luego de armar cuidadosamente la versión final de “protoboard” y apantallarla para minimizar interferencias, observamos que este ruido continuaba presente, aunque en menor magnitud.

Decidimos entonces aumentar el orden del filtro a 4. Tomando los 60Hz de corte, con este nuevo filtro tenemos una atenuación de -18dB a 100 Hz, frente a -9.4dB del primero. Es decir que atenuamos casi 3 veces más.

La topología utilizada finalmente es una mezcla entre “Salen Kelly” y “Múltiple Feedback” (MFB) (Figura 4.12), es decir que en la primera etapa de este filtro pasa bajo utilizamos una topología “Múltiple Feedback” con amplificación y en la segunda, una “Salen Kelly”.

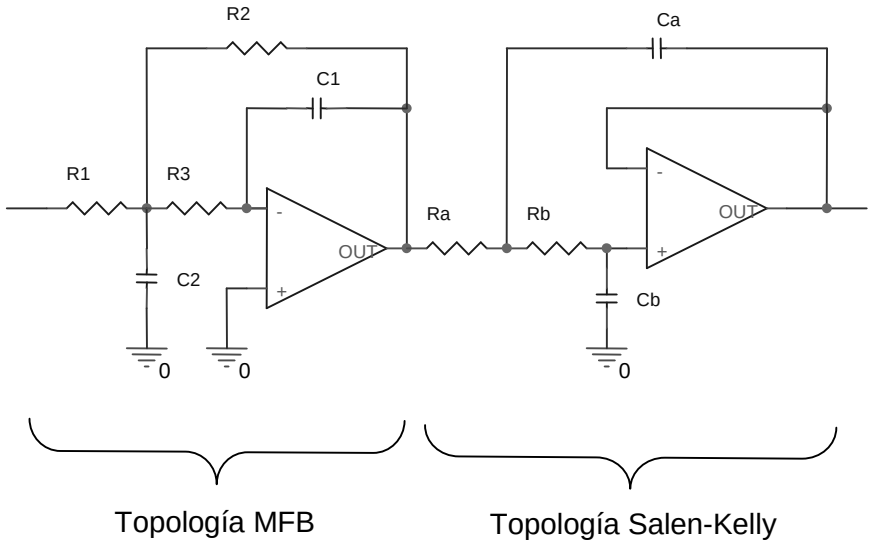


Figura 4.12 - *Filtro de butterworth de orden 4*

Filtro pasa altos:

El ancho de banda del ECG comienza a partir de los 0.01Hz, si bien podríamos pensar en un filtro pasa altos con este frecuencia de corte, debemos tener en cuenta los artefactos de movimiento que se encuentran en frecuencias por debajo de 1 Hz . La presencia de este tipo de interferencia provoca oscilaciones en la línea base del ECG, lo que hace saturar los amplificadores y más aún en nuestro caso que trabajamos con tensiones bajas. Para minimizar esta interferencia es conveniente entonces un corte en baja frecuencia superior a 0.01Hz.

Si bien se podría utilizar un filtro de butterworth para este filtrado, recordemos que la configuración elegida para la etapa preamplificadora es ya un filtro pasa bajo de primer orden.

Para minimizar el tamaño del circuito y por lo tanto el consumo, buscamos la forma de aumentar el orden del filtro pasa altos, sin aumentar necesariamente la cantidad de componentes del circuito.

Si colocamos un condensador en serie con “Ra” en el circuito de la Figura 4.11.a, podemos observar que le estamos quitando el camino a la corriente de polarización. En cambio, si lo colocamos en serie con “R1” en el circuito de la Figura 4.11.b, la corriente de polarización continúa teniendo un camino por “R3 y R2”, por lo que no se afecta el funcionamiento del operacional. A su vez, si observamos la configuración del circuito en la banda pasante (sin C1 ni C2), no es más que un inversor de ganancia R2/R1, por lo que al agregar el condensador en serie con R1, se convierte en un filtro pasa bajos de primer orden.

Es así que el filtro pasa bajos exige que una de las etapas del filtro pasa alto debe ser en configuración MFB.

Realizamos múltiples ensayos con distintas frecuencias de corte, obteniendo una línea base muy estable en frecuencias próximas a 0.1Hz e incluso menores.

Por todo esto, el filtro pasa altos lo realizamos con 2 polos simples, el de la etapa preamplificadora a 0.07Hz de frecuencia de corte y el de la etapa de filtrado a 0.03Hz aproximadamente.

Si bien podríamos haber implementado un polo doble en 0.07Hz (o menor), preferimos desplazar el segundo hacia frecuencias menores para que la caída en frecuencias en 0.1Hz sea menor y así obtener mayor información de ECG.

Finalmente el circuito para esta etapa de filtrado y amplificación que se muestra en la Figura 4.13, no difiere mucho del de la Figura 4.12.

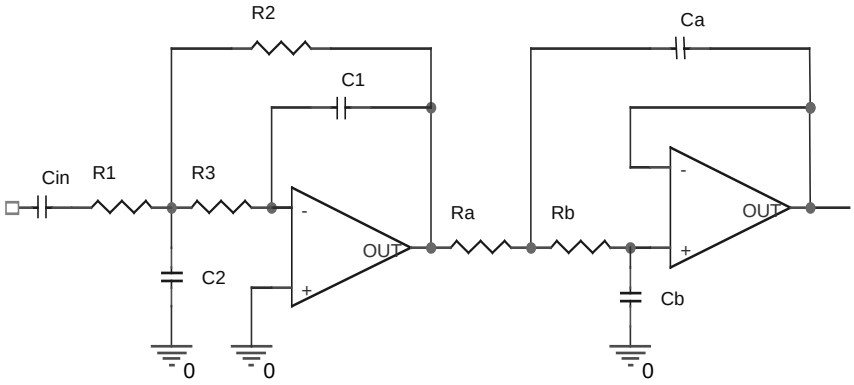


Figura 4.13 - *Etapa de filtrado y amplificación de ADQCAR*

En la Figura 4.14 se puede observa la respuesta en frecuencias de la etapa preamplificador y la de filtrado juntas.

Figura 4.14 – *Respuesta en frecuencia del preamplificador y filtrado.*

Diseño alternativo futuro:

Si bien se cumplieron los requisitos una vez implementado el prototipo, nos percatamos que una mejor solución hubiera sido la de la Figura 15, donde la primera etapa es con ganancia y la segunda no. Esto trae las siguientes ventajas:

- Se mejoran las dependencias con las tensiones de offset ya que estas no se verían amplificadas.
- Se puede bajar el valor del condensador Ci, pasando de un electrolítico bipolar (4.7uF fue el utilizado) a uno común del orden de los nF, lo cual reduce espacio y es más económico.
- No es necesario proveer a la etapa amplificadora de resistencias de compensación de corriente de offset.

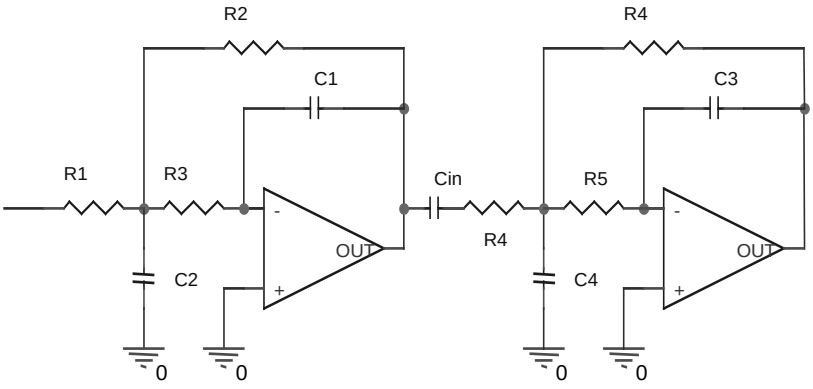


Figura 4.15 - *Etapa de filtrado y amplificación mejorada (no implementada).*

Elección de los operacionales para los filtros:

Los requerimientos para los operacionales de los filtros son los mismos que para el utilizado en la realimentación del preamplificador. Por este motivo se elige el mismo operacional, el TLV2404.

4.6 Ajuste de ganancia

Esta etapa tiene la finalidad de ajustar la ganancia para que el ECG a la entrada del conversor A/D tenga una excursión próxima a 1V, permitiendo así un buen aprovechamiento de la digitalización, dejando también un margen para oscilaciones.

Para mantener la relación entre los 3 canales, este ajuste de ganancia se realiza luego de la multiplexación. O sea que hay una sola etapa de ajuste de ganancia para los 3 canales (Figura 4.16).

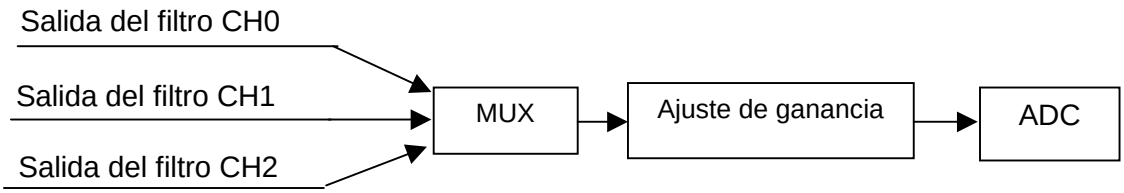


Figura 4.16 - *Ubicación en el circuito de la etapa de ajuste de ganancia.*

Este ajuste consiste en un amplificador no inversor de ganancia variable mediante una resistencia variable, seguido de un filtro RC de alta frecuencia. Este filtro tiene la función de eliminar ruido de alta frecuencia que se introduzca en la señal. El circuito se muestra en la Figura 4.17.

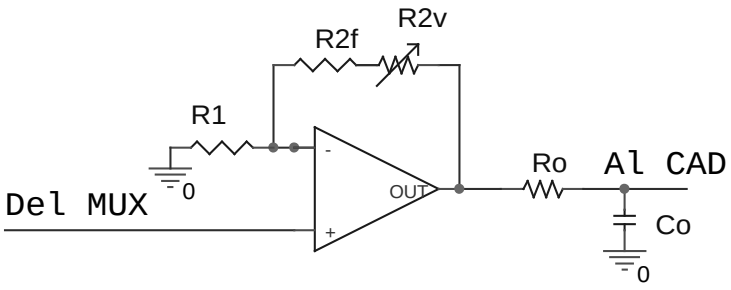


Figura 4.17 - *Etapa ajuste de ganancia*

Elección del operacional para ajuste de ganancia:

Los requerimientos para este operacional tienen alguna exigencia más que para los casos anteriores. A saber:

1) Bajo consumo
2) Alta impedancia de entrada
3) Bajo Slew Rate
4) Excursión completa a la salida

1) El bajo consumo es un requisito general de todo el equipo, por que tanto éste como todos sus componentes se eligieron de bajo consumo.
2) Una gran impedancia de entrada nos permitiría utilizar resistencias del orden de los 10 MΩ, reduciendo así el consumo.
3) Dado que se trata de operacionales de bajo consumo, las fuentes de corriente internas son pequeñas, lo que hace un bajo desempeño del Slew Rate cuanto menor sea esta corriente (o el consumo). Si bien los problemas que puede traer este parámetro son solucionables por software (hasta cierto punto) esperando la estabilización del operacional antes de tomar la muestra, es deseable que este sea lo menor posible.
4) La excursión completa nos ajusta la ganancia a un valor mayor logrando así una mayor resolución en la digitalización.

El parámetro más crítico es el Slew Rate, por lo que lo analizaremos en detalle.

El tiempo que demora un operacional en responder a un escalón de 3V es 3V/SR, que deberá ser el tiempo mínimo que debe esperar el microcontrolador entre la selección de un canal y la adquisición.

Estamos trabajando con una frecuencia de muestreo de 200 muestras/s, lo que equivale a 1 muestra / 5ms.

Si consideramos que tenemos 3 canales y que la espera se realiza en modo de bajo consumo del microcontrolador (3mA), el menor consumo se da cuando :

{(3V * 3 ch. * 3mA) / (SR * 5 ms) + consumo del operacional} es mínimo

	LT1079	OPA4241	TLV24204	Unidad
Slew Rate	70	10	2.5	V/ms
Corriente de alimentación	50	25	<1	μA
Consumo de espera del micro + operacional	0.13	0.57	2.2	mA

Tabla 4.5 - *Comparación de operacionales para ajuste de ganancia.*

De la Tabla 4.5 se obtiene que la mejor elección es el LT1079, lo que nos alejaría de tener excursión completa a la salida. Por otro lado el OPA4241, además de generar un mayor consumo, no tiene impedancia de entrada grande lo que nos obliga a utilizar resistencias del orden de los cientos de kΩ.

En los ensayos encontramos que el funcionamiento es satisfactorio con cualquiera de los dos, pero preferimos tener mayor excursión, sacrificando consumo, por lo que utilizamos el OPA4241.

4.7 Circuito completo

En la Figura 4.19 mostramos el circuito completo de la etapa de entrada con los valores de resistencias y condensadores utilizados en el prototipo.

En este circuito se incluyen además las resistencias para minimizar la corriente de polarización

Utilizando este circuito, se obtuvo el ECG que se muestra en la Figura 4.18 en uno de sus canales.

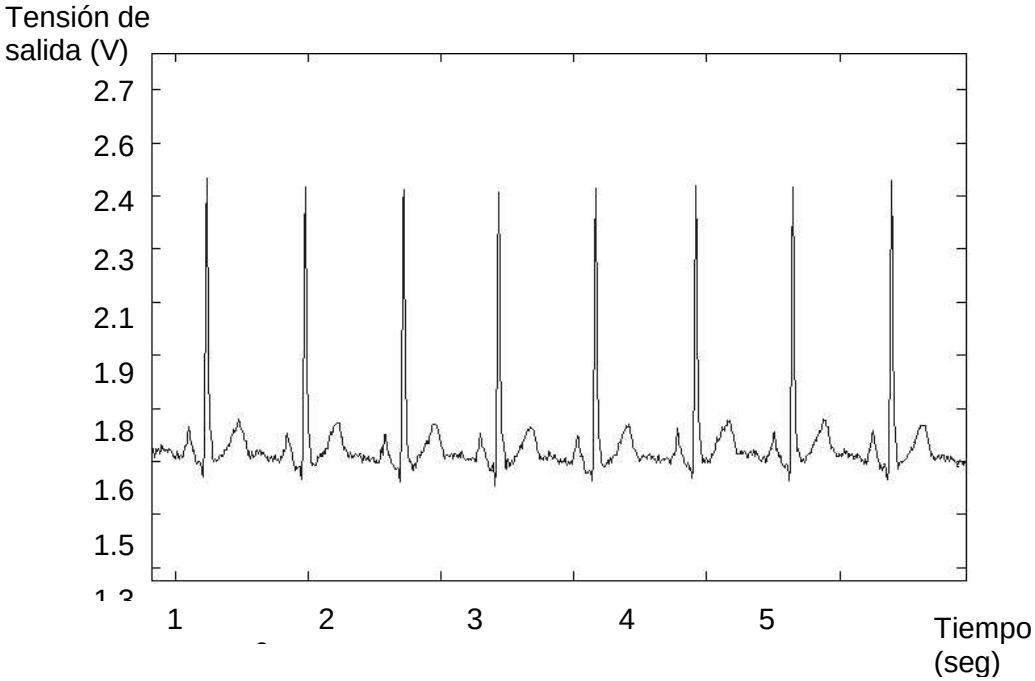


Figura 4.18 – *ECG obtenido por ADQCAR*

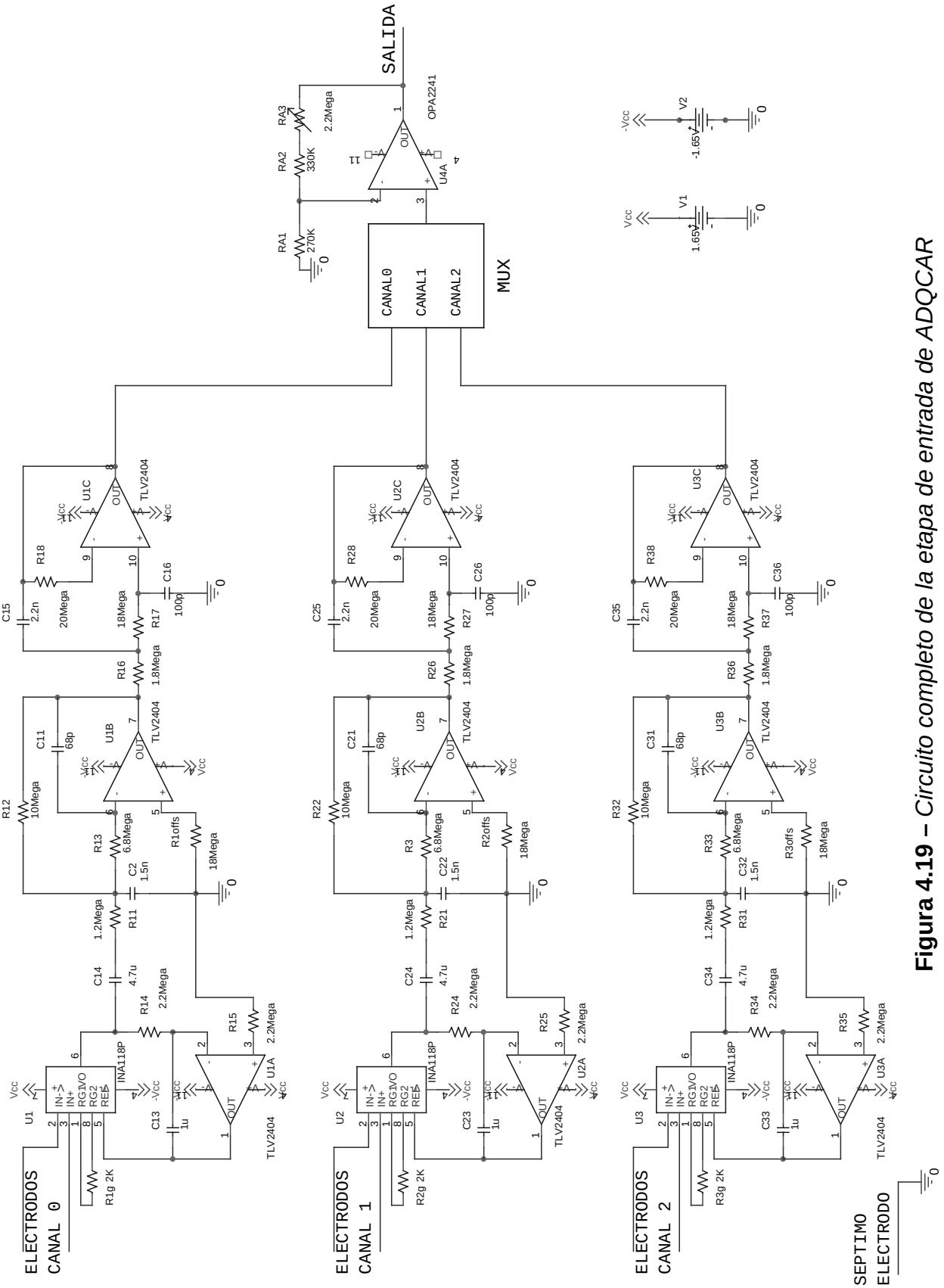


Figura 4.19 – *Circuito completo de la etapa de entrada de ADQCAR*

CAPITULO 5 – MULTIPLEXADO Y DIGITALIZACIÓN

5.1 Elección del sistema de multiplexado y digitalización

Dado que el equipo debe adquirir tres señales analógicas que luego serán almacenadas en una memoria de estado sólido, es necesaria una etapa de multiplexado de las señales y otra de digitalización. En principio se tuvieron en cuenta dos configuraciones posibles (Figuras 5.1 y 5.2):

A) Un Multiplexor analógico de 3 canales y un conversor A/D.

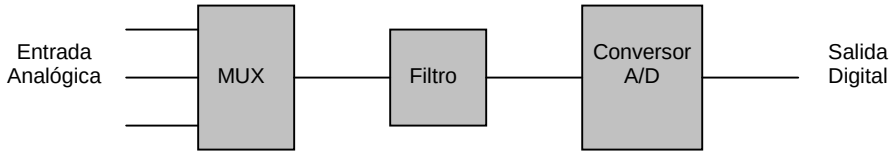


Figura 5.1 – *MUX analógico*

B) Tres conversores A/D y un Multiplexor digital.

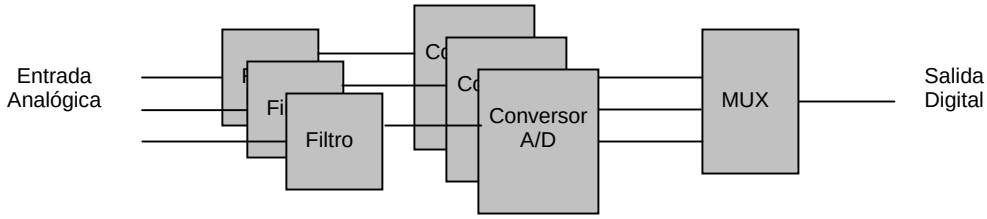


Figura 5.2 – *MUX digital*

Tomando como criterio el número de componentes, la opción A) es la más conveniente.

Para tener una idea más clara de las opciones del mercado empezamos buscando en Internet los componentes adecuados. Al igual que con el amplificador de instrumentación, buscamos integrados que funcionen con alimentación de 3V, para ser compatible con el resto del sistema, y de bajo consumo, ya que será alimentado con baterías.

Además de multiplexores y conversores A/D por separado, existe una gama de sistemas de adquisición de señales que incluyen ambas funciones en un solo integrado.

Multiplexor:

Como ya se mencionó, optamos por la configuración A), por lo tanto, buscamos un multiplexor analógico de 3 canales o más.

Algunos de los multiplexores estudiados son:

- LTC1390CN: Multiplexor analógico de 8 canales, bajo consumo y alimentación con 3V. (*Linear Technology*).

- CD74AC153: Multiplexor de 4 canales, el canal se selecciona mediante dos entradas. Funciona con 5 V y 3.3 V (*Texas Instrument*).

Conversor Analógico/Digital (ADC):

El primer parámetro que tuvimos en cuenta en la elección del conversor A/D fue el número de bits de la conversión. Este es un factor fundamental para todo el diseño, porque es el que determinará, junto con la frecuencia de muestreo, la cantidad de información que será almacenada en la memoria flash a lo largo de la adquisición de 24 horas. Considerando este factor, conviene una conversión de pocos bits, por ejemplo de 8 bits.

Por otro lado hay que considerar la forma de la señal que se muestrea para determinar de cuántos bits se desea la reconstrucción. Para determinar este factor recurrimos a la opinión del Dr. Fernando Nieto, quien nos indicó que una reconstrucción de 8 bits, o algo más de ser posible, es adecuada para analizar un electrocardiograma fijo, pero en el caso de un Holter es deseable una reconstrucción de 10 o 12 bits, de modo de lograr absorber, con los bits extra los movimientos de la línea base.

En un electrocardiograma fijo, la señal que se toma del paciente se puede amplificar al máximo del rango disponible, y con una conversión de 8 bits se obtendrá una reconstrucción de 256 niveles. Pero para un Holter, la señal no se puede amplificar al máximo porque pasaría la mayor parte del tiempo saturada debido a los movimientos. En este caso se puede utilizar aproximadamente 1/3 de la excursión disponible, es decir que si se utilizan 10 bits para la reconstrucción, la señal sería reconstruida en $2^{10}/ 3 = 1024 / 3 = 341$ niveles, es decir que es aceptable. Realizando el mismo cálculo para 12 bits se obtiene $2^{12}/ 3 = 4096 / 3 = 1365$ niveles, lo cual es mucho mejor.

Optamos entonces por usar **12 bits** para el conversor A/D, ya que, en caso de no necesitar tanta resolución se puede reducir, pero si optamos por 10 bits no se puede aumentar.

Otra de las características que se debe determinar al elegir el conversor A/D, es si se comunica de forma serie o paralela. En general los ADC paralelos son más rápidos que los serie, pero para esta aplicación, la velocidad no es el factor predominante, ya que se adquieren señales lentas, en comparación con otras aplicaciones, por ejemplo de telecomunicaciones.

Otra ventaja de la comunicación paralelo, es la interface con el microcontrolador, ya que se lee la muestra directamente desde un puerto.

Como desventaja, la comunicación paralelo requiere un bus de datos, lo que dificulta su manejo en la etapa de prototipo y el trazado de pistas sobre una plaqueta, considerando que otro de los objetivos buscados es el tamaño reducido.

En cuanto al consumo, observamos que los ADC con comunicación serie consumen entre 50 y 80 veces menos que los ADC con comunicación paralelo, lo que es muy importante a la hora de evaluar el consumo de esta etapa.

Por otro lado, otros componentes del sistema propuesto manejan interfaz serie SPI para la comunicación con el exterior, por lo tanto, es una característica a tener en cuenta.

Algunos de los ADC estudiados son:

- AD7858L, AD7853L: Las principales características de estos integrados son: alta velocidad, bajo consumo, conversión de 12 bits y operan con alimentación de 3V o 5V. (*Analog Devices*).

- TLV2544: Conversor de 12 bits. Es compatible con la interfaz SPI. Se alimenta con una fuente desde 2,7V a 5,5V y es de bajo consumo. (*Texas Instruments*).

- TLV5618AM: Conversion de 12 bits. También es compatible con la interfaz SPI. Offset y ganancia ajustables digitalmente. No es de bajo consumo. (*Texas Instruments*)

Sistemas de adquisición:

Son integrados que incluyen ambas funcionalidades, multiplexado de varios canales analógicos y conversor A/D.

Esto elimina los posibles problemas que puedan surgir al comunicar dispositivos además de la notoria reducción del espacio necesario en la plaqueta.

Algunos de los sistemas estudiados son:

- AD7890: Sistema de adquisición de 12 bits y multiplexa 8 canales de entrada. (*Analog Devices*).

- ADC12L030: Sistema de adquisición de 12 bits con interfaces de entrada/salida serie y multiplexores configurables. Funciona con alimentación de 3.3V. (*Analog Devices*).

- LTC1594L: Este integrado incluye un multiplexor analógico de 4 canales y un conversor A/D de 12 bits. La comunicación de los datos es serie. (*Linear Technology*).

Con las ideas más claras respecto a los componentes necesarios, recorrimos el mercado local en busca de alguno de los componentes estudiados o alguno de características similares. El resultado fue nulo, ya que en plaza no se consiguieron componentes de bajo consumo de las características que buscamos.

La siguiente opción fue la compra a través de Internet. Visitamos las páginas de varios distribuidores de componentes, pero encontramos otras dificultades. Por un lado, no son muchos los distribuidores que venden directamente a Uruguay, y dentro de los que venden a Uruguay, no todos venden componentes al por menor.

Existen fabricantes que venden directamente desde sus sitios Web, pero aún en ese caso, no siempre hay stock disponible de los componentes buscados.

	Componente		
Características	**AD7890**	**ADC12L030**	**LTC1594L**
Alimentación (V)	0 - 4	0 – 3.3	0 – 2.7
Nº de canales MUX	8	2, 4 o 8	4 o 8
Interfaz	Serie, flexible	Compatible con MICROWIRE y SPI	Compatible con QSPI, SPI y MICROWIRE
Consumo máximo	15 mA	4,5 mA	0,32 mA

Tabla 5.1 – *Comparación de conversores Analógico/Digital (ADC) (datos obtenidos de sus respectivas hojas de datos en Internet)*

En la Tabla 5.1 se pueden ver las características principales de los conversores estudiados, y en particular se puede observar que el LTC1594L tiene un muy bajo consumo.

Debido a este menor consumo, y a poderse importar directamente al Uruguay componentes de Linear Technology, es que decidimos traer este último conversor.

5.2 Características del integrado LTC1594L

En la Figura 5.3 se muestra el integrado LTC1598L, que es la versión de 8 canales.

Si bien hemos comentado antes las ventajas de este componente, debemos indicar que tiene la desventaja de que sólo existe en encapsulado SO, es decir, que las características del empaquetado es para montaje superficial. Este pequeño tamaño dificulta el manejo, sobre todo en la etapa de pruebas por no poder montarse directamente en un protoboard, y también al soldarlo finalmente al circuito impreso.

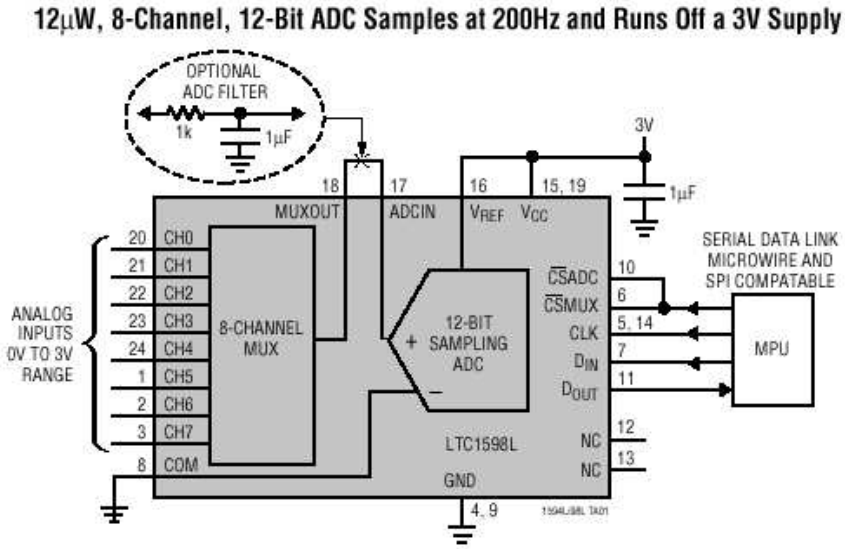


Figura 5.3 – *Esquema del LTC1598L (MUX-AD)*

5.3 Utilización del MUX/ADC

En esta parte veremos como utilizamos el chip LTC1594L en nuestro diseño. La función primera que cumple es la de seleccionar alternadamente cada uno de los tres canales de entrada. Luego, toma muestras de 12 bits de cada uno de ellos con una frecuencia de 200 veces por segundo para cada canal.

El chip contiene dos bloques separados. Por un lado el multiplexor de 4 canales y por otro el conversor AD.

En la Figura 5.4 podemos ver esta estructura y también el nombre de los pines a los cuales haremos referencia más adelante.

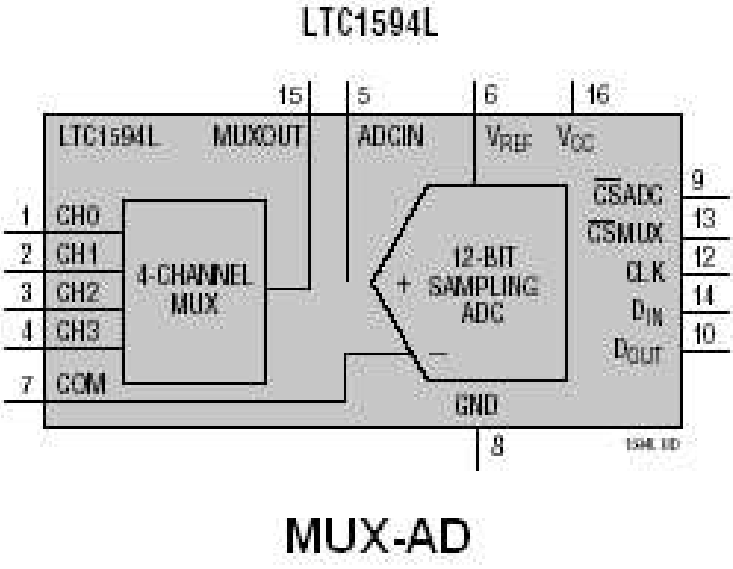


FIGURA 5.4 - *Estructura interna y pinout del chip LTC1594L*

Los tres canales amplificados y filtrados adecuadamente en las primeras etapas analógicas, son conectados a los pines 1, 2 y 3 para ser multiplexados. La salida de esta etapa (pin 15 llamado MUXOUT) enviará la señal multiplexada hacia un filtrado y amplificado exterior al chip, para luego ingresar por el pin 5 (ADCIN) al mencionado conversor AD.

El chip es alimentado por los pines 8 (GND) y 16 (Vcc). En el prototipo utilizamos la tensión regulada obtenida de la tarjeta SLK y ésta es de 0 y 3.3V, respectivamente.

Los pines 7 (COM) y 6 (Vref) darán la amplitud máxima tomada como referencia para digitalizar la señal en la entrada del conversor AD. Estos pines los colocamos a 0 y 3.3V respectivamente.

Lo que nos falta ver ahora, es como se transmiten los datos desde el conversor AD hasta el microprocesador. Para esto se utiliza la interface compatible con la SPI que soporta tanto el chip como el microcontrolador.

Antes de ver con más detalle cómo funciona esto, comentamos que el bloque multiplexor y el bloque conversor AD tienen pines de selección independientes (chip select). En nuestro caso ambos pines los tenemos conectados entre si, y esto se debe a la forma de utilizar la mencionada interface.

En la Figura 5.5 se presenta un diagrama de tiempo, donde se pueden ver como varían las tensiones de la interface entre el microcontrolador y el chip MUX-AD al momento de intercambiar datos:

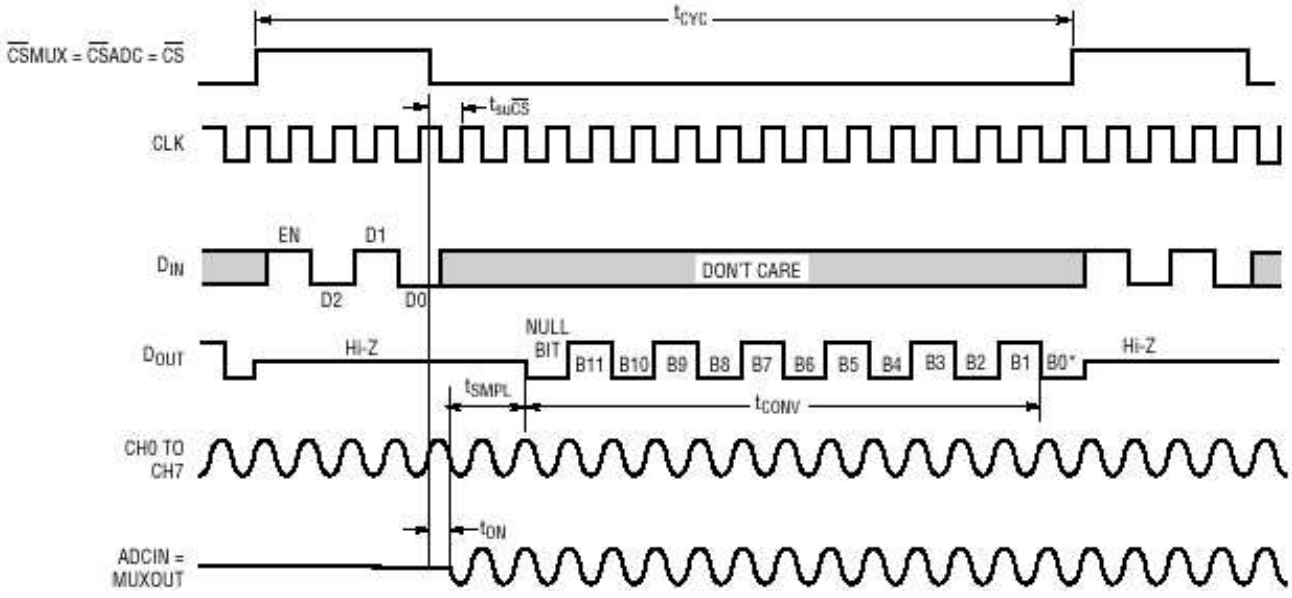


FIGURA 5.5 - *Comunicación SPI del MUX-AD con el equipo maestro (el microcontrolador)*

Como se detalló en la Tabla 5.1, este conversor es compatible con la interface serie SPI (Serial Peripheral Interface), la cual consiste en el manejo de 4 líneas digitales

que permiten la intercomunicación entre un dispositivo maestro y otro esclavo. Para el caso del MUX-AD, en el diagrama anterior podemos identificar estas líneas como:

CLK: Reloj de sincronismo comandado por el equipo maestro (en nuestro caso el microcontrolador)

D-IN: Transmisión de bits de datos, desde el equipo maestro hacia el esclavo.

D-OUT: Transmisión de bits de datos, desde el equipo esclavo hacia el maestro.

CS=**CSMUX**=**CSADC**: Selector del chip esclavo con el cual intercambiará información el maestro. Para el prototipo implementado, se utiliza un segundo CS que controla el uso de la tarjeta MMC.

La velocidad de transferencia de datos para este conversor, está fijada por la máxima frecuencia de reloj admitida, pin CLK, que no puede superar los 200 KHz.

En el microcontrolador, una rutina se encargará de dar la tensión adecuada a los pines CLK, D-IN, D-OUT y CS en cada instante de tiempo. Pero en este capítulo orientado al uso del chip MUX-AD nos remitiremos a explicar cómo funciona el protocolo de comunicación.

En primer lugar el micro le indica al multiplexor qué canal debe seleccionar. Para esto, el CS se pone en 1 (tensión lógica equivalente a 3.3V), se envía señal de reloj y se envían una serie de bits por la línea D-IN que indicarán qué canal se debe multiplexar.

En la Figura 5.5 se pueden observar unos “bits” en la línea D-IN señalados como EN, D2, D1 y DO. Al bajar el CS, estos últimos 4 bits que llegan por la SPI son validados por el chip programando a éste el número de canal a multiplexar.

En siempre debe ser 1 si es que se quiere habilitar un canal. Si es 0 entonces el multiplexor no multiplexa ningún canal hacia el pin MUXAUT.
D2, D1 y D0 forman un número binario de 3 bits que indican 1 de hasta 8 canales posibles a multiplexar. Nuestro chip es de solo 4 canales, pero el mismo formato también es utilizado en los multiplexores de 8.

Luego de este momento en que se baja el CS, el bloque conversor AD muestrea la señal que le llega por el pin ADCIN y luego de un tiempo “tsmpl” emite serialmente por la salida D-OUT los 12 bits correspondientes a la muestra. Esta ráfaga de datos comienza por un bit nulo. El microcontrolador, que es quien bajó el CS, lee estos bits cargando la muestra del canal solicitado en un registro interno. Finalmente se levanta el CS para volver al estado inicial.

En la 2 líneas inferiores graficadas en la Figura 5.5, se puede observar como el multiplexor continúa enrutando hacia el conversor AD el canal elegido, peso a que cambiaron los demás tensiones de control. Aquí se entiende porque existe la opción de programar un canal nulo (EN=0) para que el multiplexor no envíe señal hacia la salida MUXOUT. De aquí en adelante, la implementación de este protocolo es parte del software realizado para el microcontrolador.

CAPITULO 6 – MEMORIA EXTRAÍBLE DE ESTADO SÓLIDO

Una de las características importantes en el proyecto ADQCAR fue el de elegir un medio apropiado para almacenar la información, así como también la forma en que se descargarían estos datos en un PC.

Luego de estudiar las diferentes posibilidades decidimos utilizar una memoria extraíble, que luego de almacenar las 24 horas de los 3 canales ECG en el Holter, ésta se saca manualmente y se coloca en un lector con conexión USB a un PC. Un aplicativo realizado por ADQCAR, lee el contenido de la tarjeta y genera un grupo de archivos de datos que se guardan en el disco duro del computador.

6.1 Especificaciones buscadas al seleccionar la memoria

Digitalizando a 12 bits y a 200 muestras por segundo cada uno de los tres canales, 24 horas de ECG ocupan casi 75MB, por lo que con una compresión sencilla de factor entre 2 o 3 bien podemos pensar en 32MB para nuestra memoria.

Otra característica que se tuvo en cuenta fue la robustez en el almacenamiento de los datos, de forma que una falla en el sistema, como puede ser la pérdida de energía no provoque la pérdida de los datos almacenados.

Tratándose de un equipo portátil el consumo es un parámetro muy importante, ya que no se dispone de una fuente ilimitada de energía, lo que nos obligó a pensar en una memoria que sea de muy bajo consumo.

Otro factor buscado es que la descarga de la información desde la misma fuera rápida, para que la lectura completa de todos sus datos al pasarlos a un PC no sea muy lenta. Sin embargo la escritura en la misma no tendría tantos requerimientos de velocidad ya que la captura de datos se realiza a una frecuencia lenta.

Otros factores tenidos en cuenta fueron la facilidad buscada por los médicos al momento de colocarle el equipo al paciente y también la practicidad para descargar los datos en un PC.

6.2 Diferentes tipos de memorias

A partir de estas consideraciones, empezamos a evaluar desde lo mas general los distintos tipos de memoria posibles, para elegir la más conveniente.

Es así que la comparación comenzó por optar entre las siguientes posibilidades:

- SRAM: Static Random Access Memory

> Estas memorias son utilizadas generalmente como "caches". Su nombre de "estáticas" proviene de que se estructuran con circuitos "flip-flop", por lo que mientras se les provea de energía (en forma ininterrumpida), mantendrán la

información almacenada. En general son más rápidas que las DRAM, y no necesitan un circuito adicional para su correcto funcionamiento. En cuanto a precios tienden a ser varias veces más caras que las DRAM.

- **DRAM**: Dynamic Random Access Memory

  Son dinámicas porque deben refrescarse cientos de veces por segundo (lo que se realiza releyéndolas) para no perder la información que almacenan. Esto requiere la implementación de un sistema de refresco, aumentando no solo el consumo sino el software y/o hardware del equipo. Al igual que las SRAM, necesitan una fuente interrumpida de energía para no perder la información.

- **FLASH**: tipo EEPROM, Electrically Erasable Programmable Read Only Memory.

  Las memorias FLASH, combinan las ventajas de los tipos anteriores. Son de alta densidad, no volátiles y tienen un alto desempeño en lo que respecta a lectura y escritura, además de ser de bajo consumo. Una gran ventaja es que a medida que pasa el tiempo, sus precios tienden a bajar en relación a su capacidad de almacenamiento.

  En particular este tipo de memorias difieren un poco de las EEPROM, ya que a diferencias de aquellas estas se graban de a bloque por lo que se optimiza un poco más la velocidad.

  Otra característica importante es que existen en el mercado memorias flash extraíbles, lo que hace independiente la descarga de datos al PC del Holter.

Para fijar ideas, se adjunta una tabla comparativa (Tabla 6.1) de los diferentes tipos, para poder comparar valores típicos en cuanto a capacidades, consumos, precio relativo, entre otros parámetros.

Se puede observar una menor velocidad del tiempo de acceso para el tipo de memoria FLASH. El tiempo de lectura para una memoria flash de 32MB es del orden de unas pocas decenas de segundos. Además el tiempo de escritura no es crítico dado que la baja frecuencia de muestreo es varios órdenes menor. Esto implica que este parámetro es ampliamente satisfactorio para ADQCAR.

Hay dos razones fundamentales que descartan la elección de las dos primeras, por un lado estaría limitada la confiabilidad del equipo ante una interrupción de energía por agotamiento de baterías, y por otro se consume energía por el solo hecho de almacenar la información.

Otra desventaja, no menos importante, es que descargar la información a un PC exigiría la implementación de una interfase que pertenezca al Holter, dejando todo el aparato inutilizado hasta que no se descarguen los datos; en cambio para la FLASH existen tarjetas extraíbles con sus correspondientes lectores que simplifican esta tarea.

TIPO	CONSUMO (mA)	TENSION (3V)	TIEMPO DE ACCESO POR BYTE (n Seg)	CAPACIDAD > 100MB	VOLATIL	RELACIÓN DE PRECIO
SRAM	200	SI	4 a 20	SI	SI	4
DRAM	200	SI	10 a 70	SI	SI, REQUIERE CIRCUITO DE REFRESCO	1
FLASH	100	SI	> 250	SI	NO	2

Tabla 6.1 – *Comparación de memorias.*
(Datos actualizados al 22/11/02 de: http://www.footefamily.org/train/memory.htm y http://www.pcguide.com/ref/ram/timing.htm. Además hojas de datos de Compact Flash, MMC, KM416V4004C, V54C33316G2V, MCM6949, MCM6343)

Estas características hicieron que optemos en primer lugar por este tipo de memorias.

6.3 Tipos de memoria Flash

Hablar de memorias Flash también es muy genérico ya que en el mercado estas se pueden encontrar de muchas formas y capacidades. Es así que analizaremos los tipos posibles con los cuales trabajar.

En el momento de nuestro estudio, en el mercado se encontraban memorias Flash básicamente de dos tipos:

- *Chips de memoria Flash*
- *Tarjetas de memoria Flash*

Los *Chips de memoria Flash* son circuitos integrados que se utilizan por lo general para almacenar el BIOS en las computadoras, o para almacenar programas en dispositivos como teléfonos celulares. Se manejan en forma similar a los integrados de memoria ROM con la diferencia de que esporádicamente puedan requerir ser regrabados. Las dimensiones de estos integrados varían desde los integrados comunes hasta los de montaje superficial con dimensiones mucho menores. En cuanto a la capacidad de almacenamiento, los integrados al momento de comenzado el proyecto solo se encontraban con una capacidad máxima de hasta 64 Mbits, es decir 8 MBytes.

Las *tarjetas de memoria Flash* son dispositivos de almacenamiento portátiles y extraíbles similares a un disquete. Estas tarjetas se utilizan principalmente como medio de almacenamiento en dispositivos portátiles como las cámaras fotográficas digitales, los Asistentes Personales Digitales (PDA), los reproductores MP3, etc. La capacidad de estos dispositivos es mucho mayor que la de los chips individuales.

Por razones de capacidad de datos nos inclinamos por estas últimas, además el hecho de que la memoria completa pueda ser extraíble le da más practicidad al equipo y fue muy bien aceptada por los médicos que nos asesoraban.

Pensando en este tipo de tarjetas de memoria, investigamos las diferentes opciones que nos daba el mercado.

6.4 Tipos de tarjetas de memoria extraíbles

Existen varios tipos de tarjetas, algunos son fabricadas para dispositivos específicos de ciertos fabricantes, sin embargo el mercado abierto nos da la posibilidad de encontrar algunas que han sido aceptadas por varios fabricantes y de allí que sean más difundidas en el mercado.

En la Tabla 6.2 se indican las opciones corrientes.

Modelo	Tamaño (mm x mm)	Cantidad de conectores	Capacidades (MB)	Costo (U$S)
Smart Media	37 x 45	20	8 a 64	18 a 90
Compact Flash	36 x 43	40	8 a 300	31,5 a 735
MultiMedia Card	24 x 32	7	8 a 64	31,5 a 117

Tabla 6.2 – *Comparación de precios al 14 de enero de 2001.*

Algunos de los criterios tomados para decidirnos por una de ellas fueron: el tamaño de la tarjeta, tamaño del conector para adaptarla a ADQCAR, robustez mecánica, cantidad de contactos, tipo de interfase para escritura/grabación y su uso en otros dispositivos como para que en un futuro siga siendo una buena opción.

Analizando estas características finalmente elegimos la Multimedia Card (MMC).
En la Figura 6.1 podemos ver una MMC del fabricante SanDisk.

Figura 6.1 – *Tarjeta MultiMedia Card*

6.5 Características de la MultiMediaCard

Por razones de tamaño externo era en enero de 2001 y sigue siendo casi dos años después la más pequeña del mercado. Su estructura y en especial la disposición de sus 7 pines (únicos medios de contacto) le brindan una gran robustez frente a los otros modelos. Por ejemplo la Compact Flash utiliza 40 contactos comunicándose de forma paralela y la Smart Media utiliza 20.

Otra característica importante es que puede manejar 2 posibles protocolos de comunicación, uno es propio de la MMC y presenta varias facilidades para la conexión de múltiples tarjetas en el mismo bus. El otro protocolo que resultó el elegido, es el estándar SPI (Serial Peripheral Inteface) el cual es también utilizado por algunos microcontroladores, por ejemplo la línea de microcontroladores de *Motorola* y *Texas Instrument* , así como también conversores A/D y multiplexores analógicos.

Esto último es otro de los puntos que nos impulsaron a optar por la MMC, ya que al trabajar con un micro de Motorola se simplificaría la comunicación con la memoria. Además, otra de los componentes del sistema, el multiplexor-conversor A/D, también se pudo conseguir con interfase SPI.

La capacidad de datos de la memoria así como su precio también fueron tenidos en cuenta, y si bien pensábamos en 32 MB, al momento de estudiar las memorias se podía ver claramente que los precios iban a bajar por lo que solo compramos una de 8 MB para las pruebas iniciales.

Otro factor visto, es que los fabricantes de Holter han optado por utilizar memorias extraíbles, aunque utilizan otros modelos de tarjetas de mayor tamaño, como por ejemplo las Flash Card, lo cual muestra que ha sido bien aceptado el hecho de manipular este tipo de tarjetas en el ambiente médico respectivo.

6.6 Escritura y lectura en la MMC

Esta sección contiene una introducción al funcionamiento de las tarjetas de memoria del tipo MultiMediCard (MMC). También incluye la modalidad de comunicación desde el microcontrolador para grabar y leer datos.

El formato de tarjetas MultiMediaCard es un estándar y es producido por varios fabricantes. Es actualmente el formato de tarjeta más pequeño del mercado y también el que utiliza menos pines en la interfase con el exterior. Pero al igual que otro tipo de dispositivos similares, el empaquetado tiene en su interior un controlador propio que se encarga de gestionar todo el funcionamiento de la tarjeta.

En la Figura 6.2 se reproduce la arquitectura general de una tarjeta MMC, donde se puede apreciar que un controlador intermedia con el exterior en todas las funciones de acceso a la memoria.

Internamente la memoria se divide en grupos numerados llamados “Write Protected Groups”, los que a su vez se dividen en “Erase Groups”, los cuales contienen sectores (16 o 32, según el tamaño de la memoria) de 512 bytes cada uno.

Existen dos posibles modos y protocolos de comunicación con el controlador de la tarjeta: El propio modo MultiMedia Card y el SPI:

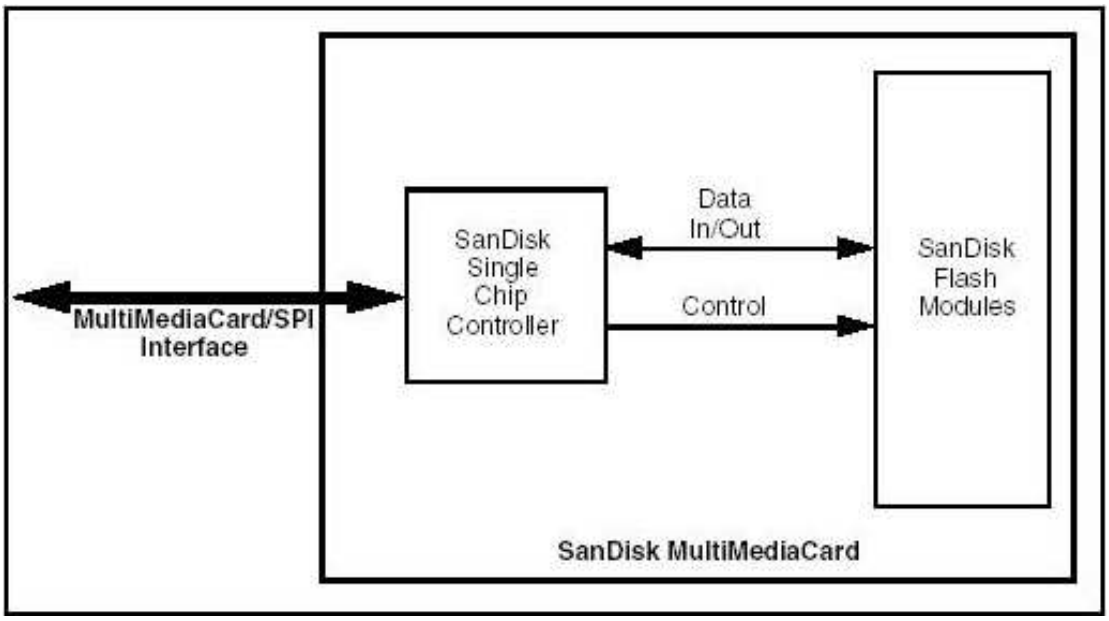


Figura 6.2 - *Esquema de la arquitectura interna de una tarjeta MMC (Tomado de "MultiMediaCard Product Manual", SanDisk Corporation, Rev 3, 7/2001 [4])*

Modos de comunicación con el módulo MMC

1) Modo MultiMedia Card

Este modo es el predeterminado y propio de este tipo de tarjetas, lo cual es una desventaja al intentar compartir el bus de datos con otro tipo de dispositivos diferentes.

Como ventajas incluye la posibilidad de direccionar en el mismo bus más de 65000 tarjetas o dispositivos. También permite leer de a bytes cosa que en el otro modo, el SPI, no es posible.

La MultiMediaCard soporta tres modos de lectura/escritura. Ellos son:

1) *Stream Mode*: Este modo permite escribir o leer datos de forma continua. Se especifica la dirección inicial de memoria donde grabar o leer, y luego se transfieren los datos secuencialmente. Para terminar, un comando de stop finaliza la transmisión. Para el caso de la escritura, el bloque total debe estar alineado por sector.

2) *Single Block Mode*: En este modo, solo se puede leer o escribir un bloque de largo predeterminado. Este bloque se protege en la transmisión con la incorporación de un CRC de 16 bits.

   El largo máximo del bloque a leer se limita a un sector (512 bytes), pero se puede llegar a direccionar hasta un mínimo de un byte. En el caso de la

escritura, para todos los modos, lo menor a direccionar es un sector. Por lo que este modo solo nos permite grabar de a sector.

3) *Multiple Block Mode*: Este es similar al Single Block Mode en lo referente a lectura y escritura, solo que se pueden direccionar múltiples bloques de datos (también en la escritura se debe alinear a sector).

2) Modo SPI:

Este segundo modo de comunicación de la MultiMedia Card se puede encontrar en varios otros componentes del mercado. Tal es así que fue el que elegimos ya que tanto el microcontrolador como el multiplexor-conversorA/D que decidimos utilizar lo implementan.

A diferencia del anterior, este modo no nos permite seleccionar por el bus el dispositivo a conectar, si no que se deben utilizar señales separadas para indicar por hardware qué dispositivo se quiere controlar.

Las posibilidades de acceso a memoria también están más restringidas, en este caso solo se puede leer o escribir en modo *Single Block* (el modo *Multiple Block* se incorporará en las especificaciones futuras).

MultiMedia Card Mode	SPI Mode
Bus serial de datos con 3 hilos (reloj, comandos, datos)	Bus serial de datos con 3 hilos (reloj, data IN, data Out) + señal CS para seleccionar tarjeta
Hasta 64000 tarjetas direccionables por el bus	Cada tarjeta se selecciona por hardware, con una señal CS.
Transferencia de datos con protección de error	Transferencia de datos con protección de error opcional.
Transferencia orientada a modo secuencial y para simple o múltiples bloques.	Lectura y escritura por simple o múltiples bloques. No soporta transferencias secuenciales.
Utilizado solo en este tipo de tarjetas. Este bus es incompatible con otros dispositivos como por ejemplo los SPI.	Utilizado en varios dispositivos de diferentes tipos (ejemplo: MUX/ADC, micro MMC2001). Se puede compartir el bus entre todos los dispositivos.
Set de instrucciones muy completo, orientado a aplicaciones muy complejas.	Subconjunto del set de instrucciones MultiMedia Card

Tabla 6.3 – *Características de los modos de comunicación de la MMC*

El modo SPI se ajusta a nuestras exigencias de comunicación. La compatibilidad del bus entre el MUX/ADC, el microcontrolador y la MMC, junto a la mayor sencillez del mismo, hicieron que nos inclináramos por este modo.

Funcionamiento de la comunicación con el modo SPI de la MMC:

La MMC tiene 7 pines, que para el modo SPI se definen así:

1_ CS *Chip select*
2_ DataIN *Host to Card Commands and Data*
3_ VSS1 *Supply Voltage Ground*
4_ VDD *Supply Voltage*
5_ CLK *Clock*
6_ VSS2 *Supply Voltage Ground*
7_ DataOUT *Card to Host Data and Status*

Aquí se menciona el Host, que en nuestro caso sería el microcontrolador que como Maestro controla todas las comunicaciones.

La señal CS en nivel bajo es para indicar que la tarjeta fue seleccionada.

DataIN es una señal unidireccional que recibe tanto comandos como datos en forma serial de parte del Host.

DataOUT es una señal también unidireccional que devuelve en forma serial datos o respuestas a los comandos recibidos.

La señal CLK es para indicar el momento que se intercambian los bits seriales de las señales DataIN y Data OUT.

Dos pines de tierra y uno de mayor tensión (3.3V) proveen la alimentación a la MMC.

Antes de intercambiar datos se deben enviar comandos para indicarle a la MMC lo que queremos hacer. Este modo de la MMC especifica una serie de comandos, de los cuales explicamos los más importantes a continuación:

Comandos utilizados:

CMD0 *Resetea la MMC y debe ser utilizado para inicializar la misma.*
CMD1 *Activa la MMC en el proceso de inicialización.*
CMD9 *Lee el registro CSD de configuración de la tarjeta.*
CMD10 *Lee el registro CID de identificación de la tarjeta (16 bytes)*
CMD17 *Lee un bloque de largo especificado*
CMD24 *Escribe un bloque de largo especificado (512 bytes por defecto)*
CMD35 *Selecciona Erase Group inicial*
CMD36 *Selecciona Erase Group final*
CMD38 *Borra todos los sectores previamente seleccionados*

El registro CSD contiene toda la información requerida para el acceso a los datos, tiempos, consumo, capacidad, parámetros de formato, entre otros. El registro CID contiene la identificación de la tarjeta, como ser su número de serie, fechas de fabricación, modelo, capacidad de memoria, etc.

Otros comandos como el CMD16, permiten definir el largo del bloque a escribir. Predeterminadamente está seleccionado el de 512 bytes, que al ser el utilizado en ADQCAR no requerimos modificarlo.

Existen otros varios comandos, 27 en total para el modo SPI, que permiten otras funcionalidades. Entre ellas, algunas a destacar son la de leer y/o modificar registros internos de la tarjeta que permiten varias configuraciones de la misma, por ejemplo proteger zonas, provisoria o definitivamente, para que no se puedan borrar.

Formato de los comandos:

La Figura 6.3 muestra las señales de cada línea del bus, mientras se intercambia un comando entre la MMC y el procesador:

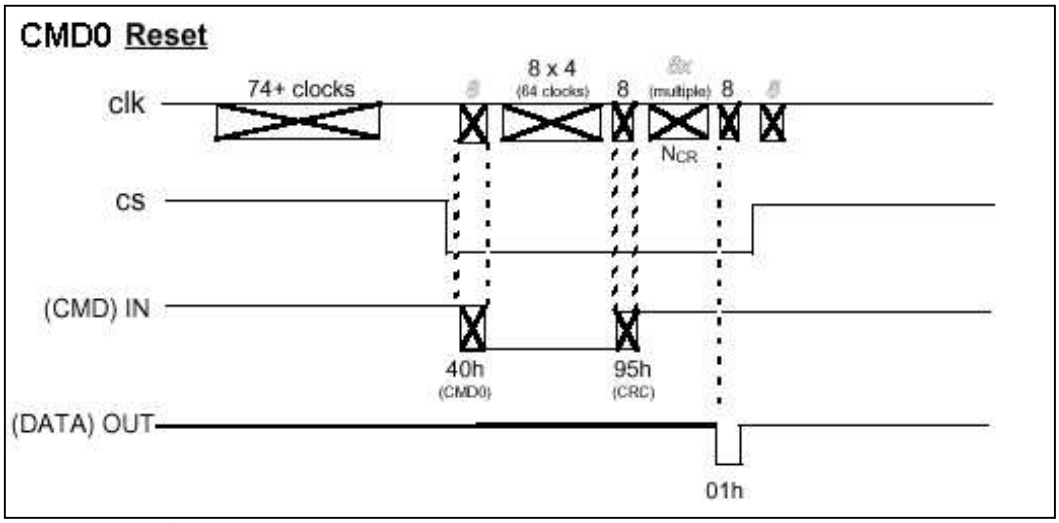


Figura 6.3 - Comando CMD0 que resetea la tarjeta.
(Tomado de correcciones a la documentación publicadas por SanDisk Corporation en su web: www.sandisk.com, 2002)

Para implementar este comando, debemos generar con el microcontrolador las líneas identificadas en el diagrama como CLK, CS y (CMD)IN y debemos confirmar que la tarjeta MMC responde según el diagrama en la línea (DATA)OUT. En caso de no responder la MMC a tiempo o que lo haga inadecuadamente con otro valor, el software esperará, desistirá o reintentará nuevamente el comando.

Para establecer una comunicación de datos con la MMC, la secuencia de comando a enviar será la siguiente:

1º: **CMD0** Resetea la tarjeta MMC y selecciona el modo SPI.

2º: **CMD1** Inicializa la tarjeta, dejándola lista para recibir comando que soliciten escritura, lectura, etc.

3º: **CMD17**, **CMD24**, **CMD9**, **CMD10** u otro, solicitando leer, escribir, etc.

Comando CMD1 de inicialización:

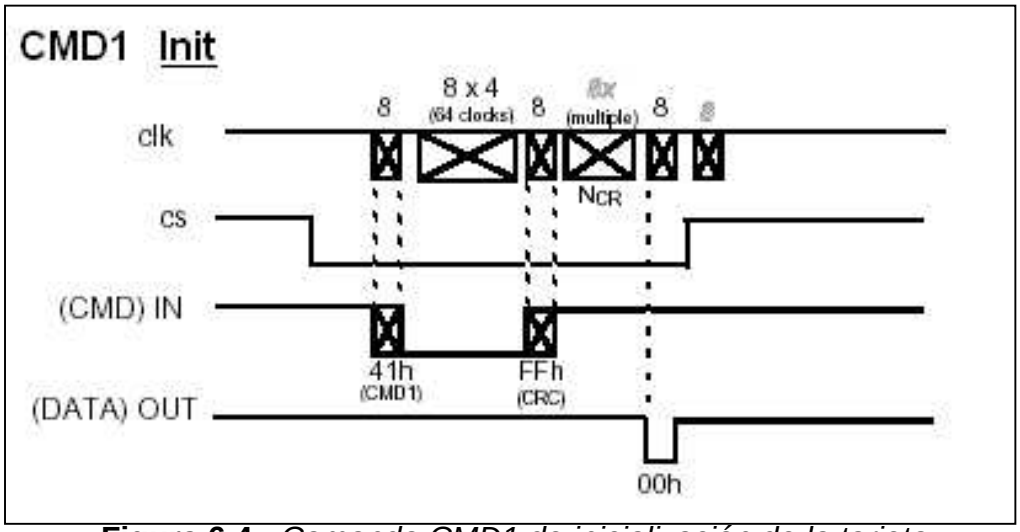


Figura 6.4 - *Comando CMD1 de inicialización de la tarjeta (Tomado de correcciones a la documentación publicadas por SanDisk Corporation en su web: www.sandisk.com, 2002)*

Comando CMD17 de lectura:

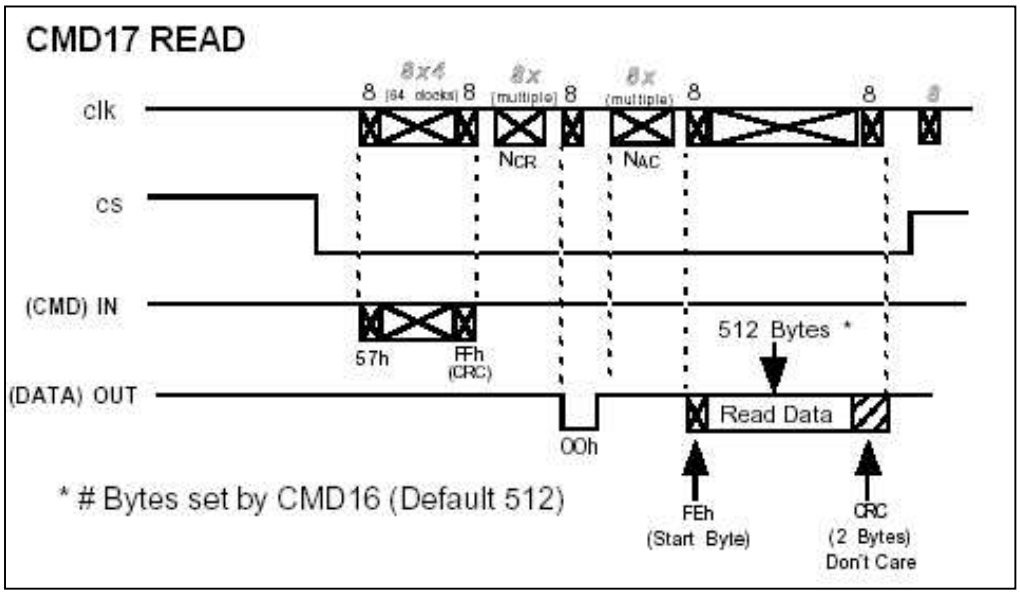


Figura 6.5 - *Comando CMD17 para leer un bloque de la tarjeta.* (Tomado de correcciones a la documentación publicadas por SanDisk Corporation en su web: www.sandisk.com, 2002)

Comando CMD24 de escritura:

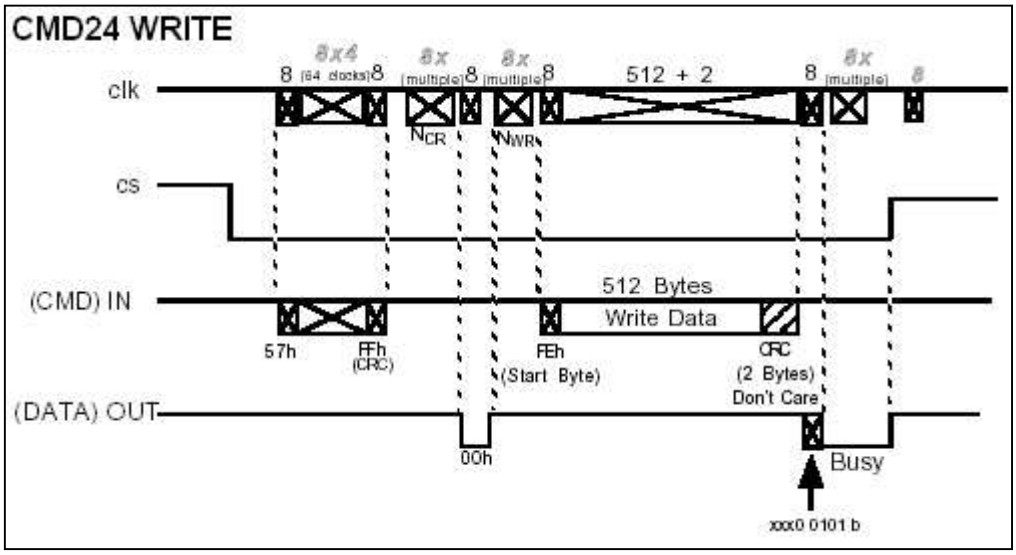


Figura 6.6- *Comando CMD24 para escribir un bloque de memoria (Tomado de correcciones a la documentación publicadas por SanDisk Corporation en su web: www.sandisk.com, 2002)*

Estos comando, entre otros utilizados, los implementamos en software de bajo nivel (assembler) hasta formar una librería, la que es utilizada por rutinas de mayor nivel.

La realización de la mencionada librería en el microcontrolador, implica usar el módulo ISPI y algunas de los módulos PWM también. En los capítulos dedicados al microcontrolador se explican con más detalle las características de estas interfaces. Para profundizar en las características de su funcionamiento, referirse a “MultiMedia Card Product Manual” [4].

6.7 Elección del lector para descargar datos en un PC

Luego de que un paciente vuelve al control médico para que se le retire el Holter, se debe descargar la información en un PC, para que el médico cardiólogo pueda analizar los datos.

Uno de los factores resaltados por los médicos y técnicos, es que la descarga de los datos al PC debe ser práctica y en particular, muy rápida. Aunque se trate de casi 52 millones de muestras de 12 bits cada una, para un médico es imposible aceptar que la operación de descarga demore más de 5 minutos, por lo que buscamos una interfaz rápida para tal operación.

Pero aunque busquemos un medio rápido, debemos pensar también en una interfaz fácilmente integrable a un PC estándar, que incluso ya podría tenerlo el médico (o técnico) utilizándolo para otras aplicaciones similares. Tanto el hardware buscado como el software (desarrollado por ADQCAR) no debe entonces limitar el tipo de computador a utilizar.

Basándonos en estas consideraciones mencionadas, vimos que el mercado nos proveía del hardware necesario para nuestros requerimientos. Sin embargo tuvimos que diseñar el programa de descarga, ya que para optimizar la grabación en la MMC y lograr una rápida descarga posterior, creamos un formato de grabación y un programa para descargar la información. Éstos nos permitieron obtener un óptimo rendimiento de la operación.

Buscando las opciones del mercado, podemos resumir los principales dispositivos útiles en la Tabla 6.4.

Modelo	Costo (U$S)	Interface
Flash Path MMC	80	Disquetera
Adaptador PC Card	65	PCMCIA
ImageMate MMC (SDDR-12)	30	USB
Combos y lectores MMC/SD	70	USB

Tabla 6.4 – *Diferentes dispositivos de lectura/escritura de tarjetas MMC (precios de enero 2002)*

Los dispositivos **Flash Path** son adaptadores de forma similar a los disquetes de 3½ con una ranura donde se coloca la MMC. La ventaja es que no requieren ningún otro dispositivo especial en el PC, el sistema operativo leerá a la MMC como si fuese simplemente un disquete. La desventaja es que la velocidad de transferencia es la de una disquetera, que al ser muy baja no cumple un requisito fundamental que planteábamos antes.

Los **Adaptadores para PC Card** permiten leer las tarjetas MMC por una interfase PCMCIA, que es muy usual en los notebook (donde se le acoplan módems o tarjeta de red). Si bien el adaptador es pequeño y se integra totalmente al zócalo PCMCIA, casi ningún PC de escritorio incluye esta interfase, por lo que nuevamente se aleja de una de las características buscadas.

El **ImageMate MMC** (ver Figura 6.7) es un dispositivo que exteriormente se parece a un ratón (“mouse”) USB con una ranura por donde se inserta la MMC para leerla.
Tiene la ventaja de la velocidad del USB y de contar con esta interfase estándar en cualquier PC de hoy en día, es de muy fácil instalación y existen drivers para que funcione en casi cualquier sistema operativo. Además el precio es muy accesible. En resumen muy apto para lo buscado.

Figura 6.7 – *Lector ImageMate SDDR-12*

También se pueden encontrar dispositivos llamados **Combos**, lectores **MMC/SD** o incluso los llamados **6 en 1,** los cuales permiten leer/escribir más de un tipo de tarjeta Flash. Si bien también cubren nuestros requisitos y se están difundiendo, en el momento del estudio eran más costosos y de una complejidad innecesaria.

En la Figura 6.8 podemos apreciar el Imageate SDDR-73 que permite trabajar con tarjetas MMC, SD y FC (Flash Card).

Figura 6.8 - *Lector ImageMate SDDR-73*

Por lo expuesto, el lector elegido para ADQCAR es el **ImageMateMMC**.

Luego de varios meses de haber utilizado el ImageMate SDDR-12, nos sorprendimos al descubrir que empezó a tener fallas en su funcionamiento. Desconcertados de si el problema era en la única tarjeta MMC que teníamos o en el único lector de tarjetas que estábamos utilizando, decidimos traer más tarjetas y más tarde un nuevo lector.

Con el nuevo material, comprobamos que definitivamente el lector era el que estaba fallando, y que la primer tarjeta MMC no tenía ningún problema. Debido a que ese modelo de lector ya no se encontraba en el mercado, optamos por traer su sucesor que es el ImageMate SDDR-73, que aun siendo un “combo” a tenía un precio muy económico. Éste dispositivo comprado, es justamente el que presentamos en la Figura 6.8.

6.8 Elección del formato para almacenar los datos en la MMC

Cuando pensamos en almacenar información en una memoria flash (o disco duro) para ser luego descargada a un computador, surge rápidamente la necesidad de un formato, la forma en que ordenaremos dicha información para que pueda ser legible.

Esto hace que tengamos dos opciones para encarar este problema:

- utilizar un formato conocido (por ejemplo los utilizados por Windows)
- utilizar un formato propio

Los lectores de MMC normalmente permiten ver el contenido de las tarjetas como si se tratase de un disco duro más del PC, ya que el formato por defecto es el FAT-12. Sin embargo, si quisiéramos usar este formato en la tarjeta de memoria, al grabar, ADQCAR debería hacer muchas operaciones para conservar ese formato, lo que implicaría mayor tiempo de proceso, y como consecuencia mayor consumo de energía. Además se debería comprar un kit de desarrollo para generar el código que realice dicha operación, lo que incrementaría los costos del proyecto, o bien estudiar nosotros todos los detalles para implementarlo, comprometiendo los tiempos de desarrollo.

Es claro que un formato conocidos (FAT-12) provee una gran ventaja y es el hecho de ser estándar, por lo que lo podemos encontrar en todos los computadores con sistema operativo Windows (e incluso otros sistemas operativos). Pero también tendríamos el problema de que el Holter sólo funcionaría para este formato y en caso de utilizar otro sistema operativo en el computador habría que modificar el código de formato del microcontrolador.

Por otro lado, si utilizamos un formato propio, éste podría ser diseñado a la medida de los requerimientos de las aplicaciones en el equipo portátil, pero se tendría que realizar un programa que pueda levantar la información almacenada en la tarjeta de memoria desde el computador.

Si bien es claro que ambas opciones tienen ventajas y desventajas, se observó que definir un formato no era una tarea demasiado compleja, por lo que basándonos fundamentalmente en:

- no incrementar los costos de ADQCAR,
- minimizar los tiempos de desarrollo,
- simplificar el trabajo del microcontrolador (minimizar el consumo),

es que decidimos utilizar un formato propio.

Teniendo la meta de desarrollar un formato propio, planteamos cuáles eran las características que debía cumplir el formato a utilizar para acercarnos al óptimo.

Estas características son:

- **No hay necesidad de acceso aleatorio.** Toda la información es grabada y leída, en su totalidad, en forma secuencial.

- **Permitir diferentes tasas de datos**. A los efectos de ofrecer generalidad, se debe poder almacenar por ejemplo ECG (200 muestras / segundo), presión arterial (1 muestra / 10 minutos), temperatura (1 muestra / minuto), entre otras señales biológicas, sin que disminuya el rendimiento en cuento a sobrecarga por información de formato.

- **Cada sector en la memoria debe ser grabado la menor cantidad de veces posibles** (esto incluye la propia información de formato). Este punto está directamente relacionado con el **consumo** del equipo portátil. Tengamos en cuenta que el pico de consumo se da cuando se escribe en memoria. El hecho de actualizar constantemente tablas de formato genera un consumo extra que no es deseable.

- **La escritura de datos es simple**. El cálculo en el microcontrolador debe ser el menor posible. Nuevamente se involucra el **consumo**. Es preferible que el cálculo se realice en el computador que se encuentra conectado a la red eléctrica y no en el equipo portátil alimentado a baterías.

- **Cantidad de archivos conocida a priori**. El número de canales a registrar (tanto sea ECG, presión, etc) es conocido previo a su grabación.

- **Robusto**. Ante una extracción accidental de la memoria, rotura del equipo portátil o agotamiento repentino de las baterías, la información almacenada no debe perderse. Esto implica que se debe disponer de suficiente información de formato en todo momento.

- **Sobrecarga pequeña**. Obtener una sobrecarga de información de formato lo más pequeña posible. Esto minimiza tanto el consumo como la sobrecarga por información de formato.

Trabajando con estos requerimientos es que definimos el formato que se describe en el siguiente punto.

6.9 Definición del formato para adquisición de Datos Secuenciales

El formato para ADQCAR implica un encabezado en los primeros sectores de la MMC (se utiliza uno, pero se puede generalizar a más si es necesario), seguido de los archivos dispuestos en bloques, obedeciendo un sencillo algoritmo de direccionamiento. En este encabezado además de grabarse información general, ADQCAR agrega los datos de cuantos canales se graban y de que tipo es cada uno (por ejemplo velocidad de muestreo). Con estos datos el programa de escritura del microcontrolador y luego el de lectura residente en el PC, logran optimizar la velocidad de acceso al disco así como también una mínima ocupación de espacio dedicada al formato.

El nombre elegido para el formato es **FADS** (Formato para Adquisición de Datos Secuenciales). Para explicar en forma detallada el **FADS**, lo dividimos en 5 partes:

1) Encabezado
2) Bloques de datos inicial.
3) Algoritmo de redefinición de bloques.
4) Fin de archivo
5) Robustez

1) Encabezado

El encabezado consiste en 46 bytes a partir de la dirección 0 de la MMC, seguidos de los bytes necesario para completar 1, 2 o más sectores (512 bytes) según el largo de este encabezado (el cual se encuentra definido en el byte 45). En la Tabla 6.5 se describe el comienzo del encabezado.

Posición (nº de byte)	Largo (bytes)	Descripción	Valor por defecto
0	34	Firma (1)	"ADQCAR_2002_RODRIGUEZ_ BORCA_DUARTE"
34	1	Versión de FADS	0
35	1	Revisión de FADS	0
36	1	Versión de Hardware	0
37	1	Revisión de Hardware	0
38	1	Versión de Software	0
39	1	Revisión de Software	0
40	4	Sectores de la memoria a utilizar (2)	Cantidad total de sectores en la memoria
44	1	Número de canales que se van a guardar	3
45	1	Sectores que ocupa el encabezado	1
46	466 + (n*512)	Se dejan libres para información particular de la aplicación (frec. muestreo, nombre paciente, fecha, hora, entre otras opciones).	

(1) Esta firma es utilizada por el programa de descarga para identificar la tarjeta de memoria.
(2) No necesariamente debe ser el total de sectores de la memoria

Tabla 6.5 - *Descripción del encabezado del FADS*

Si bien el número de sectores del encabezado es variable, con un máximo de 256, en general es suficiente solo con el sector 0.

2) Bloques de datos al inicio

A continuación del encabezado se reserva espacio para los bloques de datos (Figura 6.9). Estos son de igual tamaño (salvo el último que puede diferir en 1 o 2 sectores según el caso), ocuparan el resto de la memoria (la indicada en el byte 40 del encabezado) y son tantos como canales definidos en el encabezado (byte 44). Cada bloque se encuentra asociado a un canal (bloque j → canal j, etc).

ENCABEZADO	BLOQUE 0	BLOQUE 1	BLOQUE 2

Figura 6.9 - *Distribución de los bloques al comienzo de la grabación de los datos*

Cada bloque inicial posee un pie de bloque (Tabla 6.6) en sus últimos bytes. Es importante aclarar que no necesariamente este corresponde a su canal asociado (esto se explica en el algoritmo).

Posición (nº de byte)	Largo (bytes)	Descripción	Valor por defecto
.	.	***DATOS*** [4]	0
.	.		0
.	.		0
.	.		.
.	.		.
.	.		.
.	.		0
.	.		0
Ultimo – 6	2	Puntero a byte [3]	0
Ultimo – 4	4	Puntero a sector [2]	0
Ultimo	1	Numero de redefinición [1]	0

(1) Tiene sentido para el algoritmo de redefinición de bloques. Si es igual a 0 indica fin de archivo, en caso contrario indica el número de redefinición.

(2) Si "numero de redefinición" = 0 indica el último sector del bloque que posee datos.
Si "numero de redefinición" ≠ 0 indica en que sector continúa el bloque.

(3) Si "numero de redefinición" = 0 indica el último byte de datos del sector apuntado por "puntero a sector".
Si numero de redefinición ≠ 0 es un byte de datos más.

(4) Los datos se guardan en forma secuencial.

Tabla 6.6 - *Representación de un bloque*

3) Algoritmo de redefinición de bloques

Llamado así pues se encarga de redefinir los punteros que manejan los bloques.

Para lograr llevar un orden en la información, lo que implica saber en que orden se van completando los bloques, es preciso llevar la cuenta de como se van realizando las redefiniciones. Para esto se utiliza un contador llamado "número de redefinición". Este contador se inicializa en cero y se incrementa previo a cada redefinición, indicándose este valor en el pie del bloque a redefinir.

La distribución mostrada en la Figura 6.9, representa solo el inicio de la grabación previo al llenado de cualquiera de los bloques, es decir que solo permanece así hasta que el primer bloque se completa (Figura 6.10.a). Si este bloque necesita más espacio en memoria se elige por algún método (no perteneciente al formato) uno de los otros bloques y un sector dentro de éste. De esta forma al bloque elegido se le quita espacio desde este sector hasta el final, el cual pasa a ser la continuación del bloque completo inicial. Además se debe indicar en el pie del bloque completo

(Tabla 6.6) el número de redefinición y el puntero a sector (el sector elegido) donde continúa dicho bloque.

En la Figura 6.10, se representan 2 redefiniciones, primero del bloque 1 en el bloque 0 y luego del bloque 2 en el bloque 1.

Es interesante notar que si bien al final de cada bloque no completo hay un pie de bloque, éste no necesariamente pertenece al bloque que lo contiene.

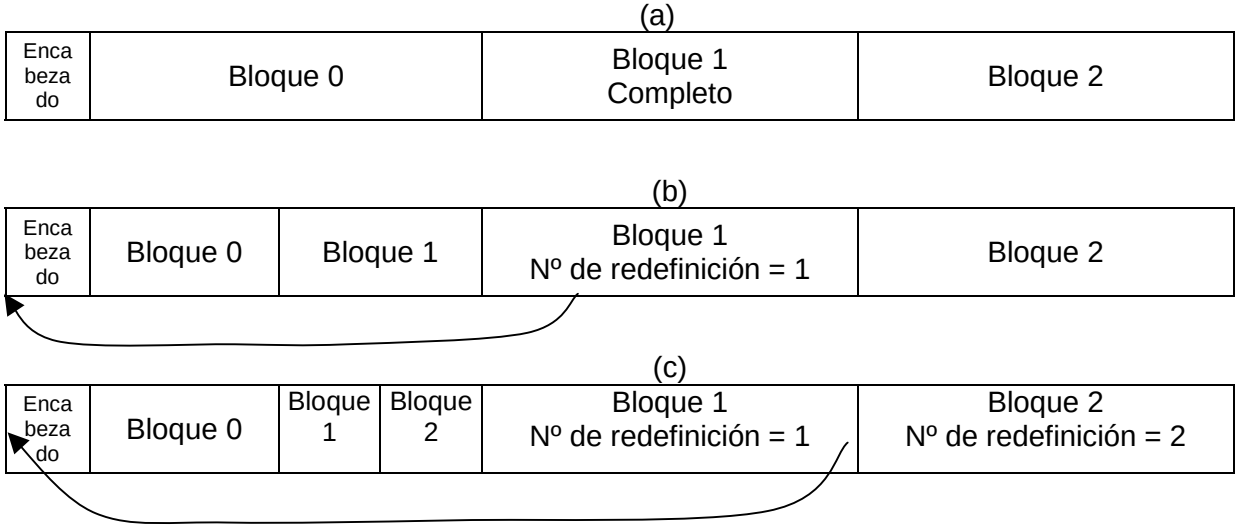


Figura 6.10 - (a) Bloques iniciales con bloque 1 completo.
(b) Redefinición 1, bloque 1 en el bloque 0, .
(c) Redefinición 2, bloque 2 en bloque 1.

Luego de cada redefinición, los bloques quedan definidos de forma idéntica al caso inicial, por lo que esta redefinición se realiza en forma recursiva.

El número máximo de redefiniciones es 255 por lo que nos preguntamos si es suficiente para un caso crítico. Suponiendo el caso de 2 canales, uno que ocupa 1 sector y el otro el resto de la memoria, y un algoritmo que divide en 2 a cada bloque que cede sectores, el tamaño de memoria que soporta es del orden de 2^{255} bytes (muchos ordenes mayor al que soporta “puntero a sector”).

Esto también nos da una idea de la sobrecarga por formato que podemos llegar a tener. Suponiendo una memoria del 32MB, en las mismas condiciones anteriores, tendremos aproximadamente 35 redefiniciones, lo que ocupa 245 bytes (menos que un sector).

4) Fin de archivo

Para indicar el final de una adquisición, lo que implica cerrar el archivo, se completa en el pie de bloque con los valores de la Tabla 6.7.

Nº de redefinición	0
Puntero a sector	Último sector del bloque con información.
Puntero a byte	Ultimo byte de información del sector apuntado por "puntero a sector"

Tabla 6.7 - *Pie de bloque indicando fin de archivo*

5) Robustez

Para garantizar que la información no se pierda en caso de baterías bajas o intento de extracción de la memoria, podría pensarse en proveer al equipo de señales indicadoras de forma que la inteligencia actúe frente a estos acontecimientos. Otra forma de lograrlo es guardando la información en forma robusta.

Si bien el formato queda definido por completo con 1), 2), 3) y 4), no está garantizada su robustez. Esto se logra simplemente borrando (con "ceros") el total de la memoria previo a la grabación de la información, lo que implica que todo pie de bloque tendrá por defecto un fin de archivo (número de redefinición = 0) y puntero a sector también cero. Cuando se detecta este hecho, se puede rastrear hasta donde hay ceros y así recuperar el 100% de la información.

Es responsabilidad del equipo de grabación que el último byte grabado sea distinto de cero y que la información tenga sentido por si sola (en caso de tener compresión y/o análisis). En muchos casos, como es el nuestro, puede no importar las condiciones de borde que introduce esto.

Es importante destacar que borrar el contenido de la MMC implica tan solo ejecutar 3 comandos, luego de los cuales el borrado es realizado por el controlador interno (en 850 ms para el caso de 32MB).

En ADQCAR se implementó este borrado de forma automática antes del comienzo de cada adquisición, con lo que logramos darle mayor robustez al sistema.

6.10 Programa para descargar la MMC al PC

Decidimos realizar el programa ADQCAR.exe de descarga de datos a PC bajo el sistema operativo Windows debido a que es el más conocido dentro del ambiente médico.

Para utilizar un formato propio es necesario poder leer byte por byte de la MMC, por lo que la primera tarea fue estudiar la forma de acceso a un disco físico. Esto no fue tan sencillo, dado que no había experiencia en los docentes consultados en el IIE, lo que se tradujo en muchas horas de búsqueda en Internet.

La herramienta utilizada para acceder a disco es el API WIN32, el cual funciona completamente diferente si se trata de Windows 95/98 o de Windows NT/2000/XP. Esto nos obligó a separarnos de la idea de que el programa funcionara para todos los sistemas operativos Windows, por lo que tomamos la decisión de realizarlo para Windows NT/2000/XP. Por otro lado, como la mayoría de los ejemplos encontrados

en Internet están escritos en lenguaje C, se optó por éste, utilizando el ambiente Visual Studio (del que disponíamos) para desarrollar el programa.

Una vez que la MMC se encuentra en el lector es vista desde el PC como un disco duro más. Una de las tareas más costosas fue lograr detectar su ubicación exacta ya que por tratarse de un lector “plug and play”, la asignación del número de disco es dinámica, pudiendo ésta cambiar cada vez que se conecta dicho lector. Además, es posible que en el futuro los PC’s dispongan de varios lectores de memorias flash lo que complicaría aún más el problema.

En términos generales, el ADQCAR.exe se encarga de detectar la ubicación del lector, la presencia de la tarjeta MMC con firma válida y de leer los archivos en FADS, para así dejarlos en el disco duro local en formato FAT. Los archivos descargados se llaman CANAL_i.adq (donde i es el número de canal comenzando por 0) y son tantos como canales se hayan adquirido.

Es importante destacar que, en caso accidental de que uno o más archivos no posean fin de archivo, ADQCAR.exe se encarga de generarlos y luego continúa con la descarga de los datos como en el caso normal. Este proceso de recuperación introduce una demora en el tiempo de descarga de los datos hacia el PC, pero ese tiempo es inversamente proporcional al tiempo que el equipo ha estado adquiriendo. Es decir que si la falla ocurre a las 23 horas de adquisición, la recuperación es más rápida que si se han adquirido unos minutos.

ADQCAR.exe fue encarado para ser utilizado como “driver” por parte de otro programa, como por ejemplo CLASICAR, devolviendo además de los archivos, códigos de error para indicar los problemas que pudieron haber ocurrido. Por esta razón no existe interfaz de usuario.

Si bien la realización del programa tomó menos tiempo del previsto, las largas búsquedas en Internet y el aprendizaje de las nuevas herramientas y ambientes requirieron bastante más, lo que se tradujo en una demora significativa de esta parte del proyecto.

6.11 Conclusiones

Sobre el formato FADS:

La sobrecarga debido al formato es inferior al ***0.0023%*** de la MMC para 32 MB (en un registro de 24 horas equivale a menos de 2 segundos).

La programación en el microcontrolador es directa y simple.

La programación en PC es compleja.

Los sectores en la MMC se escriben una única vez.

Frente a una extracción accidental de la memoria o agotamiento de baterías, el PC es capaz de reconstruir cada canal en un 100%.

Sobre la MultiMedia Card:

Este formato de tarjetas se puede encontrar hoy en día en muchos de los pequeños dispositivos portátiles del mercado, y están desplazando, como medio de almacenamiento, a otros soportes tradicionales como las Compac Flash, Flash PCMCIA, entre otros, lo que muestra que son muy bien aceptados por parte de los usuarios finales. Algunas empresas hoy en día ya ofrecen equipos Holter con almacenamiento en memoria MMC, por lo que creemos que nuestra decisión inicial de utilizarlas ha sido muy acertada para este tipo de equipos.

Empresas fabricantes de memorias tipo MMC, han estudiado el agregado de nuevas características a estas tarjetas, como ser encriptado de datos y aumento de la velocidad de comunicación, desarrollando un nuevo modelo llamado Secure Digital (SD). Exteriormente estas tarjetas son similares a la MMC, diferenciándose por incorporar 2 pines más a los 7 presentes en la MMC. Según el material consultado, las SD manejan un nuevo formato, pero igualmente siguen soportando los formatos de las MMC, en particular el SPI que es el utilizado por ADQCAR. Esto implica, aunque no lo hemos probado en la práctica, que con ADQCAR también se pueden usar tarjetas SD. Recomendamos un artículo que anexamos a la documentación, publicado en PC WORLD PERU llamado “LOS ESTANDARES DE ALMACENAMIENTO DEL TERCER MILENIO”, donde comparan las tarjetas de almacenamiento más populares del mercado actual.

Una característica importante para el personal médico que opera con dispositivos tipo Holter, es que la descarga de datos hacia un PC sea práctica y rápida. Utilizando un lector de tarjetas MMC con interface USB y el programa ADQCAR.exe, logramos una velocidad de lectura de los datos almacenado por el Holter en la MMC de 820 KB/seg. Comprobando que ésta es la misma velocidad lograda al leer tarjetas en formato FAT, vemos que es el dispositivo de lectura quien limita la velocidad de transferencia, por lo que logramos que nuestro formato FADS y el programa ADQCAR.exe permitan aprovechar al máximo el rendimiento del lector.

Sobre los objetivos de lograr tiempos de descarga bajos, podemos decir que descargar un ECG de un Holter de casette demora unos 20 minutos, uno de tarjeta Compact Flash con lector por puerto paralelo requiere 2 minutos, y ADQCAR solo requiere 1 minuto y medio. Un algoritmo de compresión estadístico de 3:1, sin pérdida de información, reduciría el tiempo de descarga de ADQCAR a solo 30 segundos. Con lo anterior, confirmamos el haber logrado un buen tiempo de descarga de datos.

Resumiendo, el hardware elegido y software desarrollado lograron cumplir los requerimientos establecidos, aunque el software para el PC requirió una dedicación mayor a la esperada.

CAPITULO 7 – MICROCONTROLADOR

7.1 Elección del tipo de componente

Al momento de la elección de los principales componentes, tuvimos que definir cómo controlar los diferentes procesos del Holter. Debíamos manejar un multiplexor de al menos 3 canales, un conversor A/D, la grabación en la tarjeta MMC y prever la implementación de un filtrado digital, un algoritmo de compresión y/o análisis de las muestras.

La utilización de lógica programable (PLD, FPGA) la descartamos, pues, si bien permite cierta flexibilidad, los algoritmos a utilizar requieren chips de costo considerable. Además, modificar código de hardware es más costoso en horas hombre que modificar código de software. Por esto, pensamos en elegir un microcontrolador o un microprocesador genérico simple.

Si bien en primer lugar pensamos en utilizar componentes ya conocidos en especial por gente del Instituto de Ingeniería Eléctrica (IIE), la limitación de consumo nos condujo más tarde a buscar en otros lados y no contar con ese apoyo.

Pensando en la capacidad de operación requerida y las interfases de comunicación hacia los otros componentes hizo que nos decidiéramos finalmente por buscar un microcontrolador.

7.2 Elección del microcontrolador

Además de las características comentadas mas arriba, otros factores participaron en nuestra opción final.

Como veremos en el capítulo 6, la memoria MMC permite dos modos de comunicación, y uno de ellos es el bus SPI*.* También el MUX-AD elegido maneja esta interfase, por lo que buscamos un microcontrolador que la incluyera.

Previo a la elección del microcontrolador, concluimos que la utilización de un algoritmo de compresión simple, sin pérdida de información, tenía ventajas importantes como parte del sistema total, tanto en el consumo, como fundamentalmente en el tiempo de descarga de los datos al PC. Por lo tanto estudiamos varios métodos de compresión, realizando pruebas en Matlab para establecer una primera aproximación de los requerimientos del microcontrolador. Esto nos perfiló a uno que posea registros de 16 bits o más y que además nos permita multiplicar y dividir directamente dichos registros. La exigencia de estas características evita la necesidad de agregar software adicional o hardware externo. También advertimos que para el uso de técnicas más complejas, tanto en compresión como en análisis, los requerimientos serían mayores aún.

Basándonos en lo anterior, comenzamos nuestra búsqueda en la línea de microcontroladores de Motorola ya que disponen de interfase SPI, lo que soluciona uno de los problemas.

El siguiente criterio que tomamos en cuenta fue la cantidad de bits necesarios. Encontramos la familia HC12 y HC16, que son microcontroladores de 16 bits a diferencia de su antecesor, el HC11 de 8 bits.

Enfocados a conseguir este tipo de controladores, buscamos en el IIE para ver si existía alguno de similares características y que además incluyera el kit de desarrollo respectivo. Este último elemento es crítico ya que implica el mayor costo en la adquisición del componente. En esta búsqueda, encontramos dos kits de desarrollo, uno para el controlador HC11 y otro para el MMC2001, este último es un controlador de última generación también de la línea de Motorola.

Optamos por elegir uno de estos para no incrementar los costos de ADQCAR y por el tiempo ganado por tenerlos ya a disposición. A continuación se especifican las características de cada uno:

	HC11	**MMC2001**
Nº de bits	8	32
Velocidad (MHz)	8	32
Alimentación (volts)	5	de 1.8 a 3 V
Consumo (mA)	35	40
Especificado para bajo consumo	NO	Sí, Ultra Low Power
Set de instrucciones	8 y 16 bits, incluye división de 16 bits	32 bits, incluye división de 32 bits
Memoria interna suficiente para correr nuestro programa	NO	SI
Maneja interface SPI	SI	SI
Disponibilidad de software de desarrollo	SI	SI, muy completo
Experiencia en el mismo por personas del IIE	Poca. Pero mucha información en Internet	Ninguna. En Internet hay poca ya que es un producto relativamente nuevo. El IIE tenía mucho interés en que desarrolláramos experiencia con él.
Precio aproximado (abril 2001)	U$S 10	U$S 22

Tabla 7.1 – *Comparaciones entre el microcontrolador HC11 y el MMC2001 disponibles*

Analizando las características anteriores, tomamos la decisión de utilizar el microcontrolador y el kit de desarrollo del MMC2001. Si bien puede parecer sobredimensionado para esta aplicación en particular, tenemos en cuenta que todo el proyecto está encarado como una base para futuros desarrollos.

Por otro lado el HC11 nos dificultaría las tareas al trabajar con 8 bits, y además se alimenta con 5 V, lo que implica mayor consumo.

7.3 Características del MMC2001 y su Kit de desarrollo

El MMC2001:

Además de lo comentado antes y siendo más gráficos, podemos agregar que el chip tiene 144 pines aunque es de muy reducidas dimensiones (montaje superficial).

En su interior integra un procesador M·Core, que maneja registros internos de 32 bits, con arquitectura RISC que funciona a 32 MHZ.

El chip incluye además:

- Memoria: 256 KB de memoria ROM y 32 KB de RAM
- 2 UART (Universal Asynchronous Receiver/Transmitter Module)
- 6 PWM (Pulse-Width Modulation Module)
- 16 bits de propósito general como I/O (especialmente utilizadas para un pequeño teclado).
- Una ISPI (Interval Mode Serial Peripheral Interface)
- OnCE (módulo para depuración de código)
- Timer/Reset Module (PIT, Watchdog, Time-of-day)

Antes de explicar con más detalles las interfaces que utilizamos, mencionaremos los demás elementos que posibilitaron la programación del microcontrolador como lo es el Kit de desarrollo.

KIT DE DESARROLLO:

Podemos resumir las herramientas utilizadas en 5 elementos bien diferenciados:

1) MMCCMB1200 (tarjeta)
2) MMCPFB1200 (tarjeta)
3) EBDI - Enhanced Background Debug Interface
4) SLK - Student Learning Kit (tarjeta)
5) CodeWarrior, Embedded Systems Development Tools (software)

1) La **MMCCMB1200** es una tarjeta multicapa, que incorpora un microcontrolador MMC2001 y otros componentes auxiliares. Extiende la memoria con 1 MB de ROM (flash) y 1 MB de SRAM externas. Tiene incorporadas dos interfaces RS232 conectadas a los UART del microcontrolador, las que permiten utilizar el depurador de código llamado Picobug (que viene grabado en la ROM), y permite interactuar con un computador. Incluye el conector para EBDI. Presenta también 4 LEDS que conectados a pines de entrada/salida de propósito general de 4 de los módulos PWM, ayudan visualmente en las operaciones de depuración.

2) La **MMCPFB1200**, que es la de mayores dimensiones, se fija por debajo de la tarjeta anterior utilizando conectores específicos para esto, quedando superpuesta una a la otra. En su perímetro varios contactos permiten acceder fácilmente a los 144 pines del microcontrolador para poder monitorear las señales con un osciloscopio o un analizador lógico. Además agrega 6 MB de

RAM y 6 MB de ROM a los recursos de la placa anterior. Incluye también un LCD de 16 X 2 caracteres, IrDA (infrarrojo), entre otras, que no utilizamos.

3) La **EBDI** es una interface de depuración que se comunica con el microcontrolador a través del módulo ONCE de éste. Internamente está constituido por otro microcontrolador el cual se programa desde el PC por medio de una interfase RS232 utilizando el software CodeWarrior. Esta interfase nos permite depurar las aplicaciones de una forma muy dinámica y sin interferir en las tareas del microcontrolador, en especial las interrupciones.

4) Luego de que recorriéramos un largo camino en el desarrollo de nuestro proyecto, el IIE obtuvo una nueva donación de un **kit estudiantil** llamado **SLK (Student Learning Kit),** el cual rápidamente llegó a nuestras manos. Consiste en una tarjeta pequeña con funcionalidades como la MMCCMB1200 pero mucho más restringidas. Entre ellas, incorpora 256 KB de ROM Flash, 256 KB de SRAM externas y un solo RS232 para cargar y depurar las aplicaciones.

   Para este primer prototipo de ADQCAR se utiliza esta tarjeta SLK como parte del circuito final, que aunque solo utilizamos el microcontrolador, osciladores y circuito de reset, nos evitó el tener que mandar a realizar una tarjeta específica, con todos los contratiempos que esto implica. Esto hace que el consumo de energía no sea el óptimo debido tanto al regulador lineal que incluye la tarjeta, que no es eficiente, como a los componentes extras no utilizados. Todo esto se describe en más detalle en el capítulo 9.

5) El kit inicial 1), 2) y 3) incluye varios CD con diferentes herramientas, entre ellas herramientas GNU y el software de desarrollo **CodeWarrior**. Nosotros optamos por este último debido a la complejidad de nuestro proyecto y a las facilidades en cuanto a depuración de código que permite.

En cuanto a la documentación de lo anterior 1), 2) y 3) y del propio microcontrolador, debemos decir que es bastante completa. Sin embargo al ser la primera versión de todo este material, nos encontramos con varios errores e incongruencias. Los problemas más comunes fueron más que nada en la descripción de características del microcontrolador, relojes en los kits a diferentes frecuencias de las estipuladas, tamaños de memoria diferentes, entre otros.

Nos iniciamos con la versión 1.0 del CodeWarrior, que aunque tenía muchos errores, tanto en el propio software como en su documentación, nos sirvió para comprender el funcionamiento del mismo y el microcontrolador.

Los problemas encontrados en esta primera instancia fueron más que nada códigos assembler con sintaxis diferentes en documentación y software, funciones documentadas incorrectamente o bien no documentadas, funcionamiento erróneo del depurador, falta de documentación referente a interrupciones y de directivas para cargar programas en memoria, sin dejar de mencionar que el programa para grabar directamente en la ROM no funcionaba, lo que nos imposibilitaba ejecutar aplicaciones desde esta memoria.

Más adelante, junto con el Kit SLK obtuvimos la versión 1.3, la cual corrige algunos errores, fundamentalmente en la parte de depuración, pero aún continuábamos con los otros ya mencionados.

Finalmente, comunicándonos por correo electrónico con personal de Metrowerks (empresa que desarrolla el software), en marzo de 2002 obtuvimos la versión 2.0 del CodeWarrior y su respectiva documentación. Esta nueva versión soluciona muchos de los problemas, pero incorpora varios cambios contextuales incompatibles con versiones anteriores por lo que tuvimos que adecuar nuestros programas; por ejemplo, una nueva forma de expresar el mapeo del programa en la memoria ("linker").

Luego de entender y adaptarnos a los cambios, con todas las herramientas funcionando correctamente, pudimos cargar nuestro programa en ROM (pero externa) para ejecutarlo completamente en el kit, sin tener que recurrir al computador para cargarlo en RAM luego de cada reset.

Todo estos contratiempos terminaron dificultando en gran medida las pruebas realizadas, así como el desarrollo del software, lo que trajo como consecuencia dedicarle mucho más tiempo del previsible.

7.4 Módulos utilizados del MMC2001

En la Figura 7.1 se muestra un diagrama representativo del interior del chip del microcontrolador. En él se pueden observar las diferentes interfaces, relojes, memorias y al microprocesador M·Core.

A continuación describiremos las características de los diferentes módulos utilizados:

Módulo ISPI (Interval Mode Serial Peripheral Interface):

Este módulo permite realizar transferencias seriales de 1 a 16 bits manejando la interfaz SPI. Tiene 3 modos de funcionamiento, el modo *Maestro Manual*, *Maestro Intervalo* y *Esclavo*.

En el modo *Maestro Manual*, el microcontrolador es el que inicia y comanda la transferencia. La misma se inicia inmediatamente después de que el programa lo indica cuando escribe el dato a enviar en un registro del módulo.

Éste modo fue el utilizado para las transferencias SPI con la MMC y el MUX-ADC.

El modo *Maestro Intervalo* es igual que el anterior, pero provee además al usuario de la habilidad de intercambiar datos a intervalos periódicos programables. Este modo comienza su operación cuando se lo indica explícitamente en otro registro del módulo.

Si bien este último modo hubiera sido útil, no se utilizó pues el indicador de tiempos no funciona correctamente.

En el *Modo Esclavo*, el intercambio de datos es controlado por un dispositivo externo.

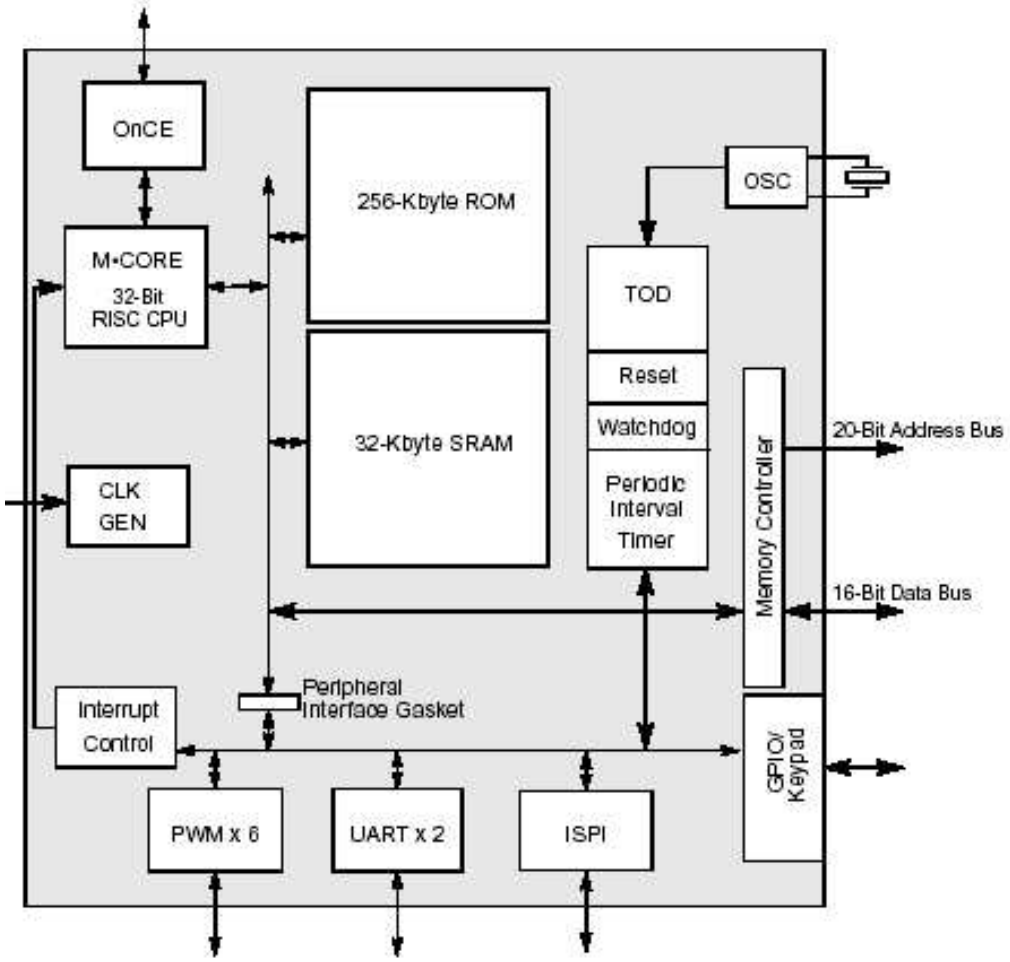


Figura 7.1 - *Diagrama de bloques del microcontrolador MMC2001*

Este módulo contiene varias señales hacia el exterior, que cuando el microcontrolador trabaja en modo Maestro, se interpretan de esta forma:

SPI_CLK	*(reloj de sincronismo, suministrado por el microcontrolador)*
SPI_MOSI	*(para enviar datos)*
SPI_MISO	*(para recibir datos)*
SPI_EN	*(puede ser utilizada para habilitar un dispositivo externo)*
SPI_GP	*(propósito general, puede ser usada par habilitar otro dispositivo externo)*

Las tres primeras líneas llegan por igual a todos los dispositivos en forma de bus. La cuarta, necesaria para habilitar el dispositivo, puede ser una de las dos mencionadas o cualquier otra que cumpla con los requerimientos de la interface SPI.

Para nuestro caso resultó más practico utilizar SPI_GP para habilitar el MUX-ADC y una salida de propósito general de un módulo PWM para seleccionar la tarjeta de memoria MMC (dado que SPI_EN no funciona correctamente).

Módulo PWM (Pulse Width Modulator):

El módulo PWM contiene 6 canales idénticos e independientes PWM5 a PWM0. Tienen dos modos de funcionamiento: PWM, que permiten generar una salida digital modulada por ancho de pulso y GPIO, entrada o salida de propósito general.

Este módulo, fue muy usado durante el desarrollo, configurado como salida de propósito general, para controlar leds y así monitorear el correcto funcionamiento del equipo. Se debió en gran parte a que el depurador no funcionaba correctamente en las primeras versiones del CodeWarrior.

En el prototipo se utilizaron los estos módulos según se indica en la Tabla 7.2.

	MODO	PROPOSITO
PWM0	-	-
PWM1	GPIO, salida	Chip Select de la MMC
PWM2	PWM	Salida analógica en modo TEST
PWM3	GPIO, salida	LED indicador externo
PWM4	-	-
PWM5	GPIO, entrada	Pulsador del panel frontal

Tabla 7.2 – Uso de los módulos PWM en ADQCAR

Programmable Interval Timer (PIT):

El módulo Timer/Reset Module incluye tres submódulos: Time-Of-Day (TOD), WatchDog (WD) y el Programmable Interval Timer (PIT). Este último fue el utilizado.

El PIT se sincroniza a partir de un oscilador independiente, diferente del utilizado por el microcontrolador. La gran ventaja de esta arquitectura es que se puede detener al microprocesador sin perder una fuente sincronizada de interrupciones.

Básicamente el PIT es un contador que se decrementa usando el reloj del módulo Timer/Reset. En un registro se setea el valor de comienzo del contador decremental, y en otro registro se maneja la información de control del mismo.

Decidimos utilizar el submódulo **PIT** para generar una cadencia de 200 interrupciones por segundo, sincronizando así el muestreo de los tres canales de entrada.

Módulo Interrupt Controller

Este módulo se encarga de controlar todas las fuentes de interrupciones y provee una interface para que el microprocesador interno las atienda. En la Figura 7.1 se puede ver cómo los diferentes módulos se comunican con éste, el que a su vez lo hace con el procesador.

Este controlador puede manejar interrupciones generadas por software, por hardware externo o hardware interno. Explicaremos de forma general cómo se manejan.

El procesador M·Core dispone de 2 líneas de interrupción:

- **INT**: Para generar interrupciones Normales.
- **FINT**: Para generar interrupciones Fast, que son de mayor prioridad que las anteriores.

Este controlador soporta un máximo de 32 fuentes de interrupción (aunque no todas están definidas) y lleva a cabo las siguientes funciones:

- Indica las fuentes de interrupciones pendientes por medio de un registro.
- Independientemente habilita y deshabilita cualquier fuente de interrupción.
- Selecciona la prioridad “Normal” o “Fast” de una fuente de interrupción.
- Provee un mecanismo para generar interrupciones por software.

El controlador se puede representar con el esquema de la Figura 7.2, donde los 5 registros indicados son de 32 bits.

El registro de control INTSRC relaciona cada uno de sus 32 bits con una fuente de interrupción, de tal forma que un “1” o un “0” indicará si la fuente correspondiente a ese bit solicitó una interrupción o no, respectivamente. Por ejemplo, el bit 20 hace referencia a la ISPI, entonces, cuando se genera una interrupción, podremos consultar este registro para conocer la fuente de la misma.

Como se observa en la Figura 7.2, el registro INTSRC es enmascarado, por un lado por el registro NIER y por otro por el FIER, generando los “registros” NIPND y FIPND respectivamente, los que a su vez generan las señales INT y FINT.

Se pueden leer cada uno de los 5 registros, pero solo se puede escribir el NIER y el FIER para enmascarar y desenmascarar interrupciones. De esta forma se habilitan y deshabilitan las fuentes de interrupción. En caso de estar habilitada una misma fuente de interrupción para ambas prioridades, el M·Core atiende la interrupción como “Fast”.

Para determinar cuál fue la fuente de interrupción, basta con leer en la rutina de atención a la interrupción el “registro” NIPND o FIPND, según corresponda, o bien leer directamente el registro INTSRC.

En ADQCAR las fuentes de interrupción que utilizamos fueron el PIT y el ISPI.

Aquí también encontramos inconvenientes en el software para desarrollo, por lo que tuvimos que trabajar todo en assembler para el desarrollo de las interrupciones. El problema encontrado fue que al compilar programas en código C, éste no realizaba bien los respaldos de registros, generando errores en tiempo de ejecución.

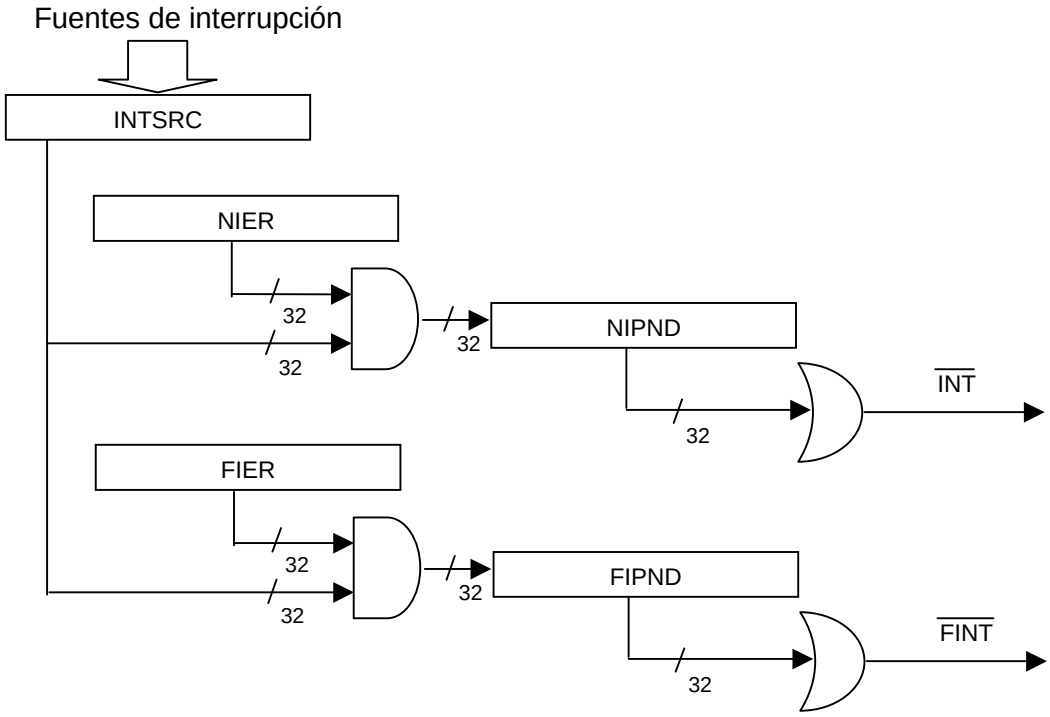


Figura 7.2 - *Controlador de Interrupciones del MMC2001*

7.5 Estados de bajo consumo

El microcontrolador MMC2001 tiene 4 modos de operación. Ellos son **Run**, **Wait**, **Doze** y **Stop**, donde los tres últimos están relacionados con el bajo consumo.

A modo general, podríamos decir que la diferencia entre los modos es que para cada caso se van desactivando más grupos de módulos, de forma de ir disminuyendo el consumo de energía (Tabla 7.3).

	MODO			
MODULO	**RUN**	**WAIT**	**DOZE**	**STOP**
CPU M·CORE	Corriendo	Detenido	Detenido	Detenido
ISPI	Corriendo	Corriendo	Según Programa	Detenido
PWM's	Corriendo	Corriendo	Según Programa	Detenido
PIT	Corriendo	Corriendo	Según Programa	Según Programa
Control. Int.	Corriendo	Detenido	Detenido	Detenido

Tabla 7.3 – *Estados de los módulos y la CPU según el modo de funcionamiento*

En ADQCAR programamos el PIT para que siga corriendo en modo STOP y así genere una interrupción cada 5 ms (200 Hz) y la ISPI para que continúe corriendo en modo DOZE.

En el capitulo 9, detallamos el ahorro de energía en los distintos modos y cómo esto aumenta la autonomía del equipo.

CAPITULO 8 – PROGRAMACIÓN DEL MICROCONTROLADOR

El programa que corre en el microcontrolador se encarga del control de los procesos de adquisición de las señales, del procesamiento de las muestras adquiridas y del grabado en la tarjeta de memoria. Cuando no está procesando, el microcontrolador pasa a un estado de bajo consumo.

Parte del programa fue escrito en lenguaje C y parte en assembler. El motivo de esto es que ciertas tareas relacionadas con cargar un valor en determinado registro, o manejar las muestras a nivel de bit, son más sencillas de implementar en assembler directamente. Otras tareas, como el filtrado o el manejo de las estructuras de datos, corresponden a un lenguaje de alto nivel.

8.1 Modos de funcionamiento

El sistema de adquisición tiene dos modos de funcionamiento: el modo **NORMAL** y el modo **TEST**. Mediante un botón ubicado en el panel frontal de ADQCAR, se selecciona el modo de funcionamiento al encender el interruptor de energía. El sistema permanecerá en el modo seleccionado hasta que se corte la energía, como se explica en el Manual del Equipo (Capítulo 10).

- Modo **NORMAL*:*** En este modo se realiza la adquisición de las muestras y el grabado en memoria. Es el modo en el que se encuentra ADQCAR durante las 24 horas de adquisición, cuando se usa como Holter.

- Modo **TEST*:*** En este modo el equipo adquiere las muestras de los canales de la misma forma que en el modo NORMAL, pero en lugar de almacenar las muestras en la tarjeta de memoria, las envía a la salida analógica disponible en el panel frontal de ADQCAR. La señal adquirida, ya filtrada digitalmente, se puede visualizar en tiempo real utilizando por ejemplo un osciloscopio. Pulsando sucesivamente el botón ubicado en el panel frontal, se puede alternar entre los 3 canales adquiridos.

Desde el punto de vista del programa el funcionamiento es similar. La principal diferencia se da en la rutina que atiende la interrupción. En el modo NORMAL, luego de adquirir las nuevas muestras desde el ADC, se verifica si existen datos para almacenar en la memoria. En el modo TEST no se realiza la grabación en Flash y, en su lugar, se llama a la rutina que envía las muestras a las salidas PWM.

Los criterios en los que nos basamos para implementar estos modos de funcionamiento son los siguientes:

- El modo TEST sería utilizado por el especialista antes de que el paciente se retire del consultorio, por ejemplo para verificar la correcta colocación de los electrodos. En ese caso no se almacenan las señales de ECG porque esto obligaría a iniciar el equipo con la tarjeta de memoria colocada. Usando el modo

TEST se le pueden colocar electrodos a otro paciente mientras se descargan los datos de la memoria hacia el PC.

- En el modo NORMAL, el paciente se encuentra desarrollando su actividad diaria, por lo que no necesita ver la señal que se está adquiriendo. Además, en el modo NORMAL se puede reducir el consumo de energía de forma más eficiente que en el modo TEST, esto se explica con más detalle en la sección 8.2.

8.2 Esquema de funcionamiento

En términos generales se pueden dividir las tareas del microcontrolador en dependientes del tiempo o independientes del tiempo.

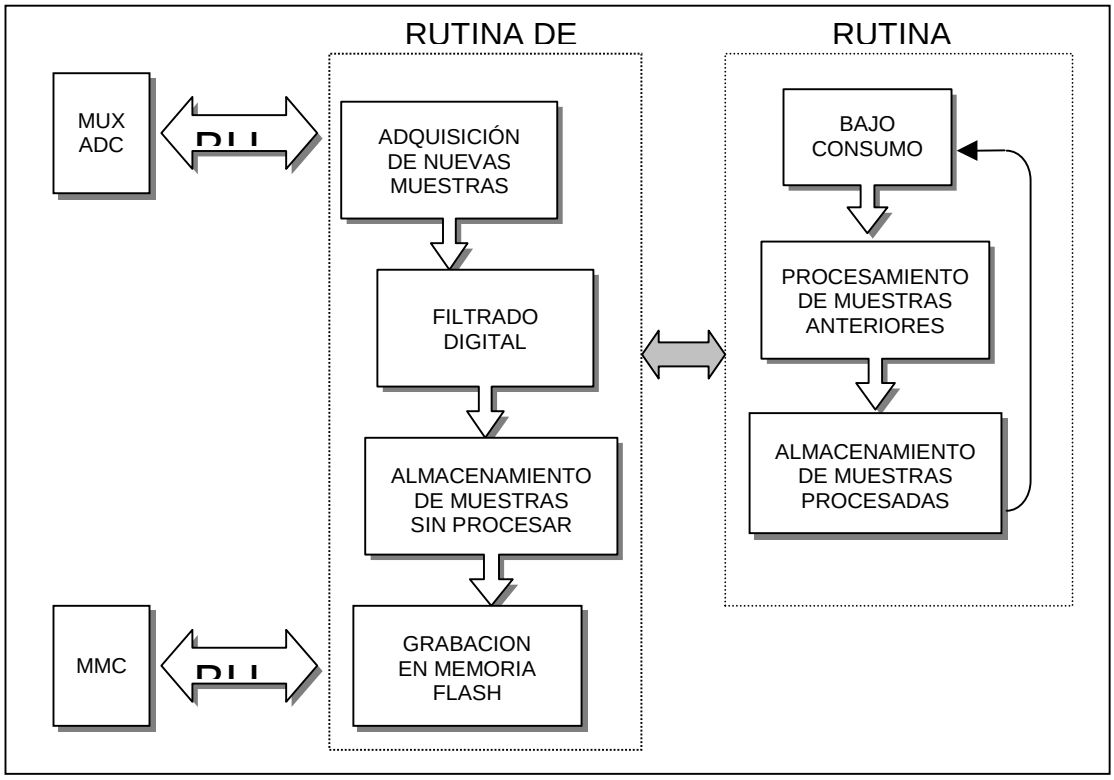


Figura 8.1 – *Diagrama de bloques del programa ADQCAR y su comunicación con los periféricos.*

- Tareas dependientes del tiempo: La principal es el muestreo de la señal, que se debe realizar periódicamente y no admite grandes demoras, por lo tanto no es conveniente interrumpirla.

  Otra tarea indirectamente dependiente del tiempo es el grabado de las muestras en la memoria Flash. Como se observa en la Figura 8.1, el bus SPI es compartido por la memoria y el conversor A/D, y como la adquisición de las muestras no se puede retrasar, hay que asegurar que la memoria no esté grabando en el momento de requerir nuevas muestras.

Debemos aclarar que el bloque de Filtrado Digital de las muestras aparece en la Figura 8.1 como dependiente del tiempo, sin embargo no es una tarea con exigencias de tiempo, pero es previa al almacenamiento de las muestras, como se explica en la sección 8.5.

- Tareas no dependientes del tiempo: Son las tareas que pueden ser interrumpidas en cualquier momento, por ejemplo una eventual compresión de las muestras o cualquier otro procesamiento que se quiera realizar.

Cuando el microcontrolador no está procesando muestras, adquiriendo o grabando la memoria Flash, pasa a un estado de bajo consumo, donde permanece hasta la próxima adquisición.

Estados de bajo consumo:

El bajo consumo fue un objetivo perseguido en todas las etapas de desarrollo de ADQCAR, el programa que corre en el equipo debe lograr ese objetivo también.

Lo que se busca es que si no hay muestras para procesar ni otra tarea que requiera del procesador, entonces éste debe pasar a un estado de bajo consumo.

Existen dos estados de bajo consumo en los que puede estar el microcontrolador, DOZE y STOP. La diferencia, además del consumo, es que en el estado DOZE se le puede indicar a cada uno de los periféricos del microprocesador si debe detenerse o no al pasar al estado de bajo consumo. En el estado STOP, solo el PIT continúa funcionando, los demás periféricos del procesador pasan al modo de bajo consumo. En particular las señales PWM se desactivan, lo que explica por qué en el modo TEST el microcontrolador consume más que en el modo NORMAL.

En cuanto al consumo, es siempre preferible pasar al estado STOP que al DOZE si se observan los consumos en la Tabla 8.1

Modo	Icc
STOP	60 μA
DOZE	3 mA
RUN	40 mA

Tabla 8.1 - *Consumo del procesador en los estados STOP, DOZE y RUN*

En los estados de bajo consumo el reloj del microprocesador se detiene, y vuelve a funcionar cuando surge una interrupción de alguno de sus periféricos. En particular en el modo STOP, solo el PIT puede despertar al microprocesador, porque los demás periféricos están inactivos.

8.3 Adquisición de las muestras

Para controlar el período de adquisición de las muestras utilizamos el temporizador del microcontrolador PIT (Programmable Interrupt Controller). Como su nombre lo indica, es un temporizador programable que periódicamente genera interrupciones al microcontrolador. En el capítulo 7 se comenta su funcionamiento, así como también otras interfaces utilizadas del microcontrolador.

La rutina que atiende a la interrupción se encarga de programar el MUX y luego recibir los datos del ADC, para la comunicación se usa el bus SPI. Este proceso de programar el MUX y leer los datos del ADC se repite tres veces, una por cada canal.

Después de obtener las muestras, deben ser almacenadas en la memoria RAM del microcontrolador, para ser luego procesadas. El almacenamiento se lleva a cabo mediante el uso de estructuras de pilas de datos, que se detalla en esta sección.

Cuando se han obtenido las tres muestras, el bus SPI queda libre, por lo que es el momento de verificar si existen datos para transferir a la memoria, que es otra tarea que hace uso del bus SPI. En caso de existir datos para almacenar, se llama a la rutina encargada de comunicarse con la tarjeta de memoria y de transferirle los datos almacenados en la RAM del microcontrolador. Cuando se terminan de transferir los datos, se libera el bus SPI con tiempo suficiente para la próxima adquisición.

ESTADOS DE LA INTERRUPCIÓN:

La adquisición de un canal mediante la SPI, se realiza en dos etapas (ver Figura 8.2):

- Primero se programa el canal deseado en el mutliplexor, para eso el microcontrolador, que trabaja como maestro en la transferencia SPI, envía el número de canal deseado y activa el reloj de la SPI. Cuando se envió el último bit que indica el canal, se activa el Chip Select del ADC (CSADC). En este momento el multiplexor selecciona el canal indicado, que queda disponible a la entrada del ADC.

- Como el MUX/ADC funciona en modo esclavo, necesita pulsos de reloj del microcontrolador para enviar las muestras, por lo tanto, en la segunda etapa de la adquisición el microcontrolador envía pulsos de reloj al ADC. Cuando el ADC detecta estos pulsos, realiza la conversión A/D de la señal que tiene en su entrada, y luego, en cada clock recibido, envía los bits de la conversión hacia el microcontrolador.

  Después de recibida la muestra, se repite el proceso con los demás canales.

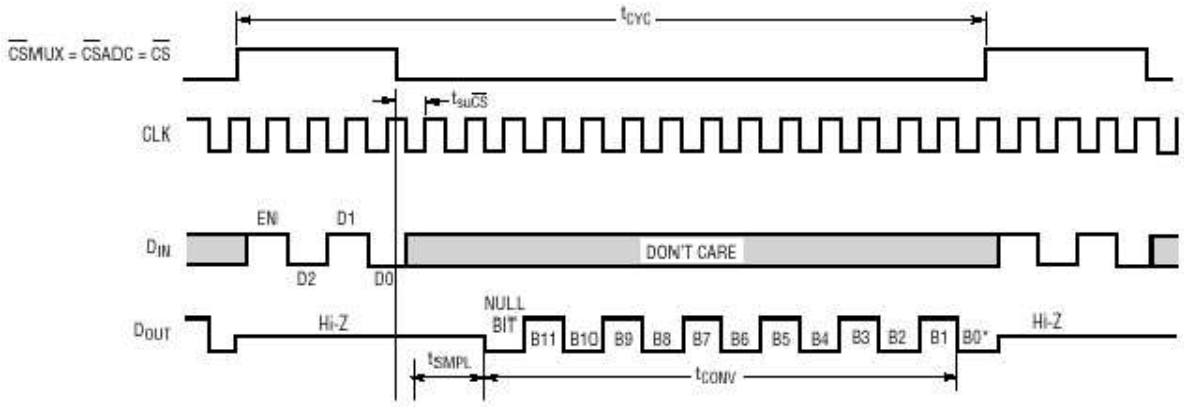


Figura 8.2 – *Adquisición de un canal usando SPI*

En la comunicación del microcontrolador con el MUX/ADC, la velocidad de transferencia de los datos está limitada a la que soporta el MUX/ADC, que para este conversor es de 200 KHz.

La interfaz SPI del microcontrolador se puede programar para varias velocidades, que son divisiones del reloj principal (HI_REFCLK) respecto a un coeficiente (BAUD_RATE). Se buscó entonces la velocidad más cercana a los 200 KHz del MUX/ADC. El coeficiente BAUD_RATE utilizado es 256, y el reloj principal del microcontrolador, HI_REFCLK es de 32 MHz. El resultado de la división es 32 MHz/256 = 125 KHz. Es decir que como el microcontrolador trabaja a 32 MHz, se cuentan 256 pulsos de su reloj por cada pulso de reloj de la SPI (pin CLK).

Considerando lo anterior, y que para cada muestra de un canal dado se necesitan 4 bits para su selección, más 12 bits de datos más 2 adicionales, que son necesarios para efectuar la conversión, la comunicación completa requiere de 4+12+2 = 18 bits. Teniendo en cuenta que se adquieren 3 canales, el total de bits comunicados es 18*3 = 54 bits. Traduciendo este número a pulsos de reloj del microcontrolador, se obtienen 54*256 = **13824** pulsos de reloj del microcontrolador para cada adquisición de 3 canales.

Con el objetivo de reducir el consumo y el tiempo de procesamiento requerido para la adquisición, descartamos la idea de un loop de espera en cada comunicación con la SPI. Otra alternativa es dejar el procesador en un estado de bajo consumo, mientras el módulo SPI del controlador se encarga de la comunicación. Esto soluciona el tema del consumo, pero no se recupera el tiempo de procesamiento.

Para poder utilizar el tiempo de comunicación en procesar muestras anteriores, se utilizan interrupciones entre cada transferencia SPI. De esta forma, la adquisición de las señales se realiza en varias etapas, según la Figura 8.3. En esta figura se representa, en el primer eje de tiempo, el uso del bus SPI, en el segundo eje el procesamiento del microcontrolador cuando atiende a las interrupciones, y en el tercer eje el procesamiento normal del microcontrolador.

Lo que se observa es que, mientras se está utilizando la SPI, el microcontrolador está procesando muestras anteriores (P) o se encuentra en estado de bajo consumo, si no hay mas muestras para procesar.

Cuando no hay tareas para realizar, pero el bus SPI está activo (transmitiendo), entonces se pasa al estado DOZE, del que se saldrá con la próxima interrupción. Si en cambio el bus SPI no está activo, se puede pasar al modo STOP. No se puede pasar al modo STOP mientras la SPI esté activa, porque se detendría la transferencia.

El proceso de adquisición comienza cuando el temporizador PIT solicita una interrupción al microcontrolador. En ese momento se pasa al **estado 0** de la interrupción, donde se escriben los registros de la SPI para indicarle al MUX que seleccione el canal 0 y se activa la transferencia. Al mismo tiempo se vuelve a la rutina principal, mientras la comunicación se lleva a cabo.

Cuando se han transferido todos los bits, la SPI solicita una interrupción al microcontrolador, pasando al **estado 1**. En este estado, se activa el ADC (bajando la pata CS) y es entonces cuando el multiplexor selecciona el canal 0. Los datos están disponibles a la entrada del conversor A/D, pero para que se realice efectivamente la conversión, es necesario enviarle pulsos de reloj. Se activa nuevamente la SPI para recibir la muestra del canal 0.

Nuevamente, cuando la SPI termina la transferencia, solicita una interrupción al microcontrolador, pasando al **estado 2**. En este momento, la muestra del canal 0 se lee desde el registro de datos de SPI (SPDR) y se guarda en una variable en RAM del microcontrolador. Luego se prepara la programación del canal 1 y se activa la transferencia SPI.

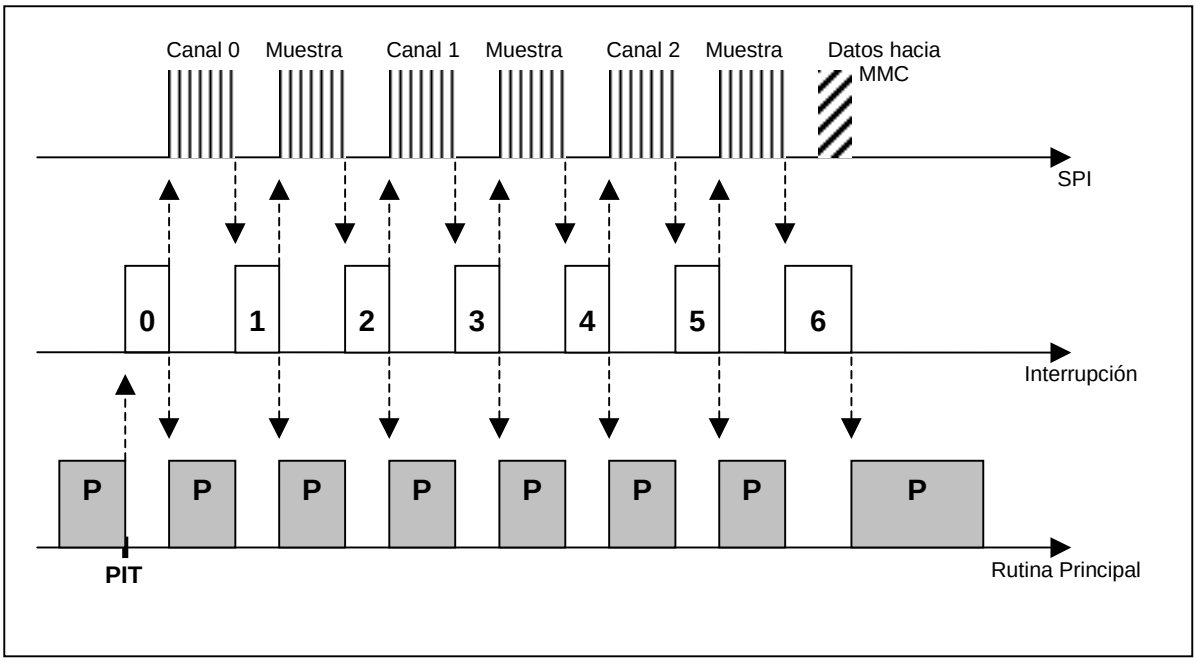


Figura 8.3 – *Estados de la interrupción de ADQCAR:* ***0****,****1****,****2****...****6*** *procesos que dependen del tiempo,* ***P*** *porciones del programa principal, sin requisitos de tiempo*

El mismo proceso se repite para los demás canales. En el **estado 3** se envían los pulsos de reloj para recibir la muestra del canal 1, y en el **estado 4** se lee la muestra

del registro SPDR. También en el estado 4 se programa el último canal. En el **estado 5** se envían los pulsos de reloj necesarios, y en el **estado 6** se lee la última muestra, correspondiente al canal 2.

Al final del estado 6, existe otra transferencia SPI, representada en la Figura 8.3 con barras en diagonal. Esto corresponde a la transferencia de datos desde el microcontrolador hacia la tarjeta de memoria. Esta transferencia ocurre después de la adquisición de las muestras, pero no en todas las interrupciones del PIT, sino sólo cuando hay datos suficientes para almacenar.

Pilas de datos:

Luego de obtenidas las muestras desde el MUX/ADC, éstas quedan en variables globales del programa.

En principio se puede pensar en procesar las muestras a medida que se van adquiriendo, pero, teniendo en cuenta que muchos de los algoritmos de compresión estudiados necesitan varias muestras para empezar a comprimir, ya sea para calcular la correlación, hacer algún promedio o detectar latidos, es necesario contar con una estructura que almacene las muestras a la espera de ser procesadas. De esta forma el programa se puede adaptar a varios algoritmos de compresión.

Por otra parte, luego de procesadas, las muestras serán almacenadas en la tarjeta de memoria Flash, pero la memoria MMC, funcionando en modo SPI, solo puede ser grabada en bloques de 512 bytes (un sector). Entonces, luego de procesadas, las muestras deben permanecer en memoria RAM del controlador hasta completar el sector.

La solución para almacenar las muestras es el uso de pilas de datos. Por cada canal que se adquiere se necesitan dos pilas, una para los datos de entrada y otra para los datos procesados.

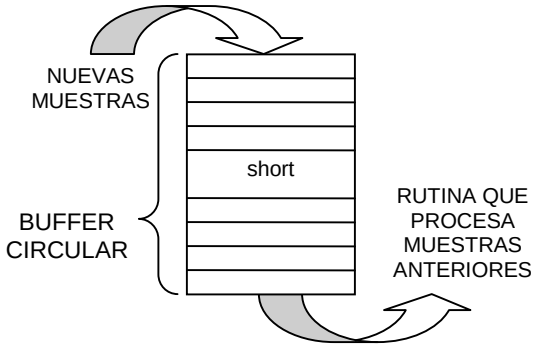


Figura 8.4 - *Pila de datos de entrada de ADQCAR*

Como se observa en la Figura 8.4, la pila de datos de entrada es compartida por la rutina que almacena las nuevas muestras y por la rutina que las procesa. Las nuevas muestras se ingresan a la pila de datos de entrada luego de ser filtradas digitalmente. Se ingresa una muestra por cada canal en cada interrupción del PIT.

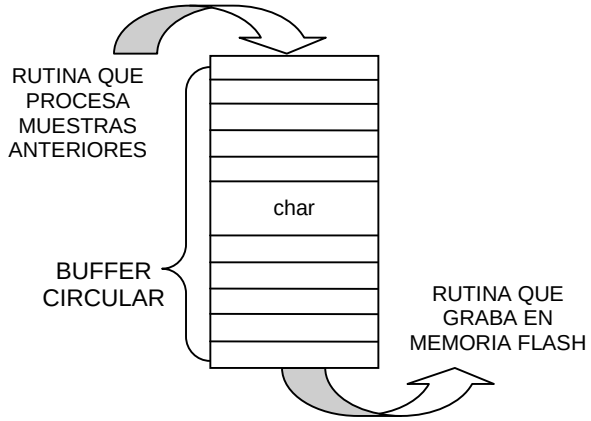


Figura 8.5 - *Pila de datos procesados de ADQCAR*

La rutina que procesa las muestras (compresión por ejemplo), toma los datos de la pila de entrada y, luego de procesarlos, los almacena en la pila de datos procesados, que se muestra en la Figura 8.5. Esta pila es compartida con la rutina que se encarga de almacenar los sectores en la tarjeta de memoria.

Se implementó en lenguaje C, una estructura de datos que contiene un array para las muestras, punteros que apuntan al primer y último elemento válido del array y una variable que lleva la cuenta del número de elementos válidos. El número de elementos válidos no es indispensable, ya que se puede deducir mediante los punteros, pero resulta más práctico para las demás rutinas el disponer de ese número y no calcularlo en cada oportunidad.

También se implementaron las funciones que agregan y extraen los datos de las estructuras. Estas funciones se encargan de mantener actualizados los punteros del array y el contador de elementos.

Las pilas de datos son estructuras FIFO (First In First Out), de forma que, mientras una rutina ingresa nuevas muestras, otra quita las más viejas.

Para mantener acotada la memoria RAM utilizada, es necesario que las pilas se comporten como un buffer circular. Esto se logra actualizando adecuadamente los punteros a los elementos del array. Es decir que si se llega al final del array, el nuevo elemento se lee o se guarda en el primer lugar.

8.4 Grabación en memoria Flash MMC

El programa ADQCAR se encarga de controlar la grabación de los datos procesados en la tarjeta de memoria Flash y de mantener el formato de datos diseñado (FADS). Para lograr esta tarea, se implementaron las rutinas de comunicación con la memoria mediante el bus SPI.

La memoria recibe por el bus SPI una serie de comandos, con los que se puede controlar su estado, realizar una escritura, una lectura, o consultar alguno de sus registros internos. Se implementaron rutinas en assembler para iniciar la memoria, pasarla al modo SPI, leer un sector, escribir un sector, borrar todo su contenido, consultar los registros CID y CSD, de donde se puede deducir la capacidad de la memoria.

Antes de utilizar la memoria en el modo SPI es necesario inicializarla. Esto se realiza en la rutina de inicialización, enviando a la memoria los comandos CMD0 y CMD1. Luego la memoria queda lista para recibir los otros comandos.

Para grabar un sector en la memoria, primero es necesario reunir 512 bytes (un sector), en la pila de datos procesados se guardan los datos de la rutina de procesamiento hasta llegar a 512 bytes, sin embargo, la pila de datos procesados puede contener más de 512 bytes, para soportar un retraso en el vaciado de la pila debido a que la memoria está ocupada con un sector anterior.

Si se considera que en cada interrupción del PIT se adquiere una muestra de 12 bits de cada canal, entonces las tres pilas de datos llegan simultáneamente a contener 512 bytes, y por lo tanto requerirán almacenar sus datos en la memoria al mismo tiempo. Sin embargo, si se considera que las señales pasarán por una etapa de compresión, entonces las velocidades de llenado de las tres pilas de datos son distintas, porque la tasa de datos que genera la compresión depende de la señal. Pensando en el caso general, consideramos que la velocidad de llenado de datos es distinta para cada canal.

La escritura de la memoria se realiza mediante el comando WRITE_BLOCK (CMD24), que luego de recibir los datos, debe grabarlos efectivamente en la memoria Flash. El proceso de grabado de la Flash depende de la tecnología de la memoria utilizada, para las primeras memorias, que ya no se fabrican, el tiempo de grabado luego de recibir el sector, puede llegar a 5 ms en teoría, y algo más en la práctica. Las nuevas memorias reducen el tiempo de grabación a 1 ms aproximadamente. Luego de recibir los 512 bytes, la memoria pasa a un estado BUSY, mientras su controlador interno grabar efectivamente el sector en memoria. En este estado no recibirá más comandos (a menos que se reinicie la memoria).

Para independizarnos de ese tiempo, la rutina de escritura en memoria verifica que la misma no esté en el estado BUSY antes de enviar los datos, es decir que está disponible para recibir datos nuevos. Si encuentra que la memoria está ocupada entonces no graba y espera a la siguiente interrupción del PIT para hacer un nuevo intento. Esto explica por qué las pilas de datos deben contener más de 512 bytes.

En el prototipo, las pilas de datos pueden contener hasta 550 bytes, es decir un sector y 38 bytes adicionales. Los 38 bytes adicionales corresponden a 25 muestras de 12 bits. Considerando que se adquiere una muestra cada 5 ms, los 38 bytes adicionales se traducen en 125 ms. En otras palabras, los 512 bytes de datos procesados se pueden mantener en la pila de datos hasta 125 ms, antes de ser sobrescritos. El peor caso que se puede presentar es que la pila del canal 0, por ejemplo, tenga que esperar a que se graben los datos del canal 1 y canal 2 antes de disponer del bus SPI. Teniendo en cuenta el peor caso (para las MMC de la primera generación), el tiempo de grabación de un sector puede superar los 5 ms en la práctica y tomando como cota superior 10 ms, aún así la grabación de dos sectores (canal 1 y canal 2) requeriría unos 20 ms. Es decir que a lo sumo en 20 a 25 ms se estará vaciando la pila de datos del canal 0, restando 100 ms antes de que se sobrescriba una muestra.

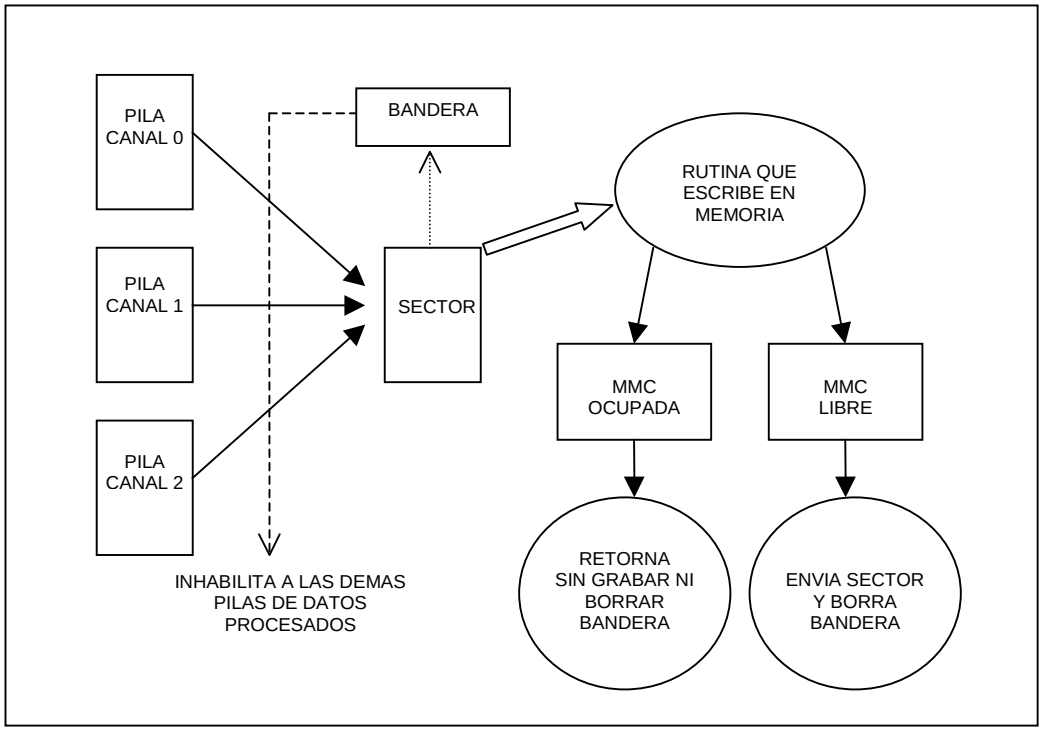


Figura 8.6 – *Esquema de grabación de sectores en MMC desde el punto de vista del programa*

En la Figura 8.7, se observa un diagrama que explica la forma en la que se controla la grabación de los datos que provienen de los tres canales. La rutina de procesamiento va llenando la pila de datos de cada canal; en caso de usar compresión, la tasa de llenado de cada canal puede ser diferente. Se observan además, una variable llamada *bandera*, y otra variable llamada *sector*, que es un array de 512 bytes.

Periódicamente, con cada interrupción del PIT, se verifica si alguna de las pilas de datos contiene 512 bytes, en ese caso se pasan los 512 bytes a la variable *sector* y se activa la *bandera* que indica que hay datos para guardar en la memoria Flash. La próxima interrupción, como la bandera se encuentra activa, se inhabilita a los demás canales a llenar la variable *sector*, y se llama a la rutina que se comunica con la

memoria MMC, pasándole como parámetro la variable *sector*, y el número de sector de la memoria que se desea escribir.

Si la memoria está libre, la rutina que la graba transferirá los 512 bytes al microcontrolador interno de la MMC utilizando el comando WRITE_BLOCK (CMD24). Cuando terminó la transferencia, desactiva la variable *bandera* y queda pronto para recibir un nuevo sector. En caso de que la rutina que escribe en la MMC encuentre que la memoria está ocupada (escribiendo un sector anterior), entonces no intenta escribir y vuelve a la rutina que la invocó, pero sin limpiar la bandera, de modo que en la siguiente oportunidad se hará otro intento por escribir el sector. Como se mencionó anteriormente, el sector se escribirá antes de que peligren los datos de las demás pilas.

Escritura en MMC usando el bus SPI:

En el caso de la escritura de MMC, y en general cualquier comando enviado a la MMC, la comunicación se realiza usando el bus SPI. La velocidad de comunicación en este caso no está limitada por el dispositivo externo, como ocurre con el MUX/ADC, sino que el límite está dado por la máxima velocidad del bus SPI que incluye el microcontrolador.

La máxima velocidad que permite el bus SPI del microcontrolador es de 4 MHz, mientras que la máxima velocidad que soporta la MMC es de 20 MHz, por lo tanto, utilizamos la máxima velocidad de 4 MHz para la comunicación con la MMC.

La cantidad de bytes que se transmiten por el bus SPI para grabar un sector en la memoria se puede acotar en 530 bytes, por lo que el tiempo de transferencia de un sector hacia la memoria es de 1,06 ms. Este es el tiempo que estará ocupado el bus luego de la adquisición de las muestras desde el ADC, sin embargo, el microcontrolador interno de la MMC continuará trabajando, sin usar la SPI, en la grabación de los bytes en la Flash interna. Como se ha mencionado, el tiempo de grabado de un sector depende de la tecnología de la memoria, si existe una futura mejora en la tecnología de grabación, no hay necesidad de cambiar el algoritmo con el que se comunica el programa ADQCAR, ya que los comandos se mantienen y la velocidad de transferencia de los datos en el bus está limitada por el microcontrolador y no por la memoria.

Formato de datos:

El programa debe encargarse de mantener el formato de los datos que se guardan en la memoria. Para eso se implementaron funciones donde, por cada canal, se mantienen actualizadas las variables que indican el próximo sector a grabar, eso incluye el cálculo del nuevo sector cuando se ha terminado el espacio en el bloque asignado.

Al encender el equipo y luego de iniciar la memoria, se inicializa la estructura que hace posible recuperar desde el PC los datos grabados en la MMC. Antes que nada se realiza el borrado completo de la memoria para poder recuperar los datos en caso

de que accidentalmente no se complete la adquisición. Luego se graba el primer sector con la información de formato y características de la memoria, a continuación se inicializan en el microcontrolador las variables que mantendrán el formato FADS.

Cuando se llega al final de la adquisición, se indica el fin de los archivos de datos, colocando los punteros adecuados, es lo que llamamos *cerrar los archivos*.

Mantener actualizado el formato es una tarea que no requiere demasiados recursos desde el microcontrolador, la mayoría de las veces sólo se necesita incrementar una variable. Desde un principio se definió el formato FADS para que no consumiera recursos en el microcontrolador, pasando el trabajo hacia el PC en el momento de recuperar los datos.

8.5 Filtrado Digital

El filtrado digital de las muestras es un módulo del programa que se encarga de filtrar las señales de entrada para reducir el ruido y fijar el ancho de banda de interés.

Este módulo, a modo de bloque independiente, puede formar parte de la etapa de procesamiento o se puede colocar previo al almacenamiento de las muestras en las respectivas pilas de entrada. En ADQCAR optamos por colocarlo en la etapa previa al almacenamiento de las muestras. La razón es que, para cualquier algoritmo de compresión que se utilice, el factor de compresión aumenta si la señal está libre de ruido.

Una de las ventajas de contar con una etapa de filtrado digital es que se reduce la complejidad del circuito de entrada, que solo se encarga del filtrado necesario, previo al muestreo. Se pasa parte del trabajo al microcontrolador, aprovechando su capacidad de procesamiento.

Como desventaja, al implementar un filtro digital aparecen otros factores que se deben tener en cuenta, como por ejemplo la cuantificación de los coeficientes. Esta es una limitante en el momento de elegir el filtro, ya que, a medida que aumenta el orden del filtro, el número de coeficientes crece y decrece la precisión con la que se pueden representar.

Una opción para aumentar la precisión de los filtros digitales es usar el tipo de datos *double*, que es un entero de 64 bits, o el tipo *float* para trabajar con punto flotante. El problema es que los tipos *float* y *double* no son tipos básicos de C, y por lo tanto requieren el uso de una librería. El software CodeWarrior cuenta con una librería de punto flotante, que incluye el tipo *double*, pero el tamaño ocupado por esa librería adicional es superior al tamaño de nuestra aplicación. Si bien la memoria del microcontrolador es suficiente para soportarla, optamos por no incluirla y limitarnos a trabajar con el tipo *long*, es decir una palabra de 32 bits.

En caso de necesitar mayor precisión para los coeficientes, se puede reducir el número de bits de las muestras y aumentar el número de bits para los coeficientes.

O, si se requiere aun más precisión, se pueden implementar la suma y el producto *double* en assembler.

Filtrado digital implementado en el prototipo:

Para complementar el filtrado analógico de las señales, se implementó en el prototipo un filtro digital de Butterworth, de orden 5, diseñado con Matlab.

Filtro:

$$a_1y_6 = b_1x_6 + b_2x_5 + b_3x_4 + b_4x_3 + b_5x_2 + b_6x_1 - a_2y_5 - a_2y_4 - a_4y_3 - a_5y_2 - a_6y_1$$

donde los x_i son los valores de la entrada y los y_i los de la salida en el tiempo i. Los a_i y los b_i son los coeficientes del filtro. Estos coeficientes no son números enteros, así que para realizar las operaciones multiplicamos la ecuación anterior por una constante *K* resultando:

$$\begin{aligned} Ka_1y_6 = {} & Kb_1x_6 + Kb_2x_5 + Kb_3x_4 + Kb_4x_3 + Kb_5x_2 + Kb_6x_1 \\ & - Ka_2y_5 - Ka_2y_4 - Ka_4y_3 - Ka_5y_2 - Ka_6y_1 \end{aligned}$$

Para calcular el valor de y_6, se divide la ecuación entre Ka_1.

Para aprovechar al máximo los 32 bits disponibles para las operaciones, y considerando que los x_i y los y_i son enteros de 12 bits, hay que asegurar que los valores Ka_i y Kb_i se puedan representar con 20 bits (32 - 12), para que el producto Ka_iy_j y Kb_ix_j sea representable con 32 bits.

Para implementar la suma de la ecuación anterior, se realiza cada operación por separado; primero cada producto Kb_ix_j y Ka_iy_j que es representable con 32 bits. Luego se divide el resultado entre Ka_1, que es lo mismo que dividir entre K, porque el coeficiente $a_1 = 1$. Al dividir el resultado entre K, el nuevo resultado es del orden de las muestras, por lo que se puede realizar la suma sin problema de desbordamiento y obtener así el nuevo valor de la salida y_6.

El valor de K se calcula para que $K*\max(|a_i|, |b_j|) * 2^{12} < 2^{32}$, de esta forma el producto de K con las muestras y los coeficientes nunca desborde.

En la Tabla 8.2 se muestran los coeficientes utilizados en el prototipo, ya multiplicados por el factor K.

b1 = 23347	a1 = 1064187
b2 = 116739	a2 = -1048570
b3 = 233478	a3 = 1036357
b4 = 233478	a4 = -411155
b5 = 116739	a5 = 118299
b6 = 23347	a6 = -11986

Tabla 8.2 – *Coeficientes del filtro de Butterworth digital pasabajo*

Como se están adquiriendo y filtrando 3 canales, se necesita conservar los valores de las muestras para cada uno de los canales por separado, aunque se utilicen los

mismos coeficientes. Para eso, por cada canal que se filtra se mantiene un array con los valores anteriores de entrada y salida.

8.6 Finalización de la adquisición

La adquisición y grabado de las muestras puede finalizar por las siguientes razones:

a) La memoria se llenó
b) El temporizador llegó al final
c) Se solicita finalizar manualmente
d) Se agotó la batería o se cortó la alimentación accidentalmente
e) La memoria se extrajo mientras se realizaba la adquisición

El caso a) ocurre cuando no hay mas sectores disponibles en la tarjeta de memoria, por lo que las rutinas que actualizan el formato indican al programa, mediante una bandera, que la MMC se llenó.

En b), la adquisición termina cuando han pasado los minutos de adquisición previamente seleccionados desde el programa. Esto sucede a las 24 horas en caso de usar ADQCAR como Holter.

Para proveer una forma que termine correctamente la adquisición antes de las 24 hs, se puede finalizar manualmente la adquisición, caso c), manteniendo presionado el botón del panel frontal por unos 5 segundos en el modo NORMAL.

En los casos a), b) y c), se ejecutan los siguientes procesos para detener la adquisición:

- Se detienen las interrupciones del PIT, para no adquirir más muestras.

- Se cierran los archivos de datos en la MMC. En caso de que finalice la adquisición debido al temporizador está previsto vaciar las pilas de datos, pero en caso de que la adquisición termine porque se acabó la memoria, es imposible almacenar esos datos porque no hay lugar. De todos modos la cantidad de datos en las pilas no es importante frente al resto de la información almacenada. Si quedaran 512 bytes (340 muestras de 12 bits), faltarían 1,7 segundos de adquisición en 24 horas.

- Por último, se deja al microcontrolador en un estado de bajo consumo, donde permanece, ya que no hay ningún generador de interrupciones activado.

El caso d) no debería suceder si se utiliza correctamente el equipo, ya que antes de comenzar una adquisición de 24 horas se deben remplazar las baterías por unas completamente cargadas, sin embargo podría suceder que la energía se corte debido a un golpe fuerte por ejemplo. En cualquier caso, si el equipo se reinicia, los datos guardados no se pierden. En una futura versión de este equipo, se utilizaría un conversor DC-DC como fuente de alimentación. Muchos de estos integrados cuentan con un sensor de batería baja, el que puede ser revisado periódicamente por el programa para tomar acciones en caso de acabarse la batería.

El caso e) puede ser accidental, no debería ocurrir durante el funcionamiento normal. Si durante la adquisición el programa detecta que la memoria no responde por un tiempo determinado, se detendrá la adquisición y se pasará a un modo de bajo consumo. Previamente se indica el error mediante el LED ubicado en el panel frontal del equipo ADQCAR.

En los casos c) y d), los archivos no se han cerrado adecuadamente, pero esto no implica que la información se ha perdido. El programa que descarga los datos hacia el PC, detectará el error y recuperará la información almacenada. De todos modos, es más eficiente el funcionamiento del programa cuando los archivos han sido cerrados.

Temporizador de adquisición:

En el momento de compilar el programa que se carga en el sistema, existe la posibilidad de establecer la cantidad de minutos de adquisición deseados.

Para lograr esto se lleva la cuenta de las interrupciones ocurridas desde el inicio del sistema en una variable *temporizador*. Como las interrupciones tienen un período fijo (200 muestras/seg), se puede calcular fácilmente el tiempo de adquisición transcurrido y finalizar el sistema cuando *temporizador* llega a un valor establecido.

El número de minutos de adquisición se debe determinar antes de compilar el programa que se cargará en el sistema, ya que ADQCAR no cuenta con un control externo de esta variable. Para cumplir las especificaciones, el prototipo adquiere durante 24 horas.

El temporizador solo funciona en el modo NORMAL, en el modo TEST no existe un control de tiempo.

Finalización manual:

Como se indicó antes, se puede detener la adquisición de forma manual sin necesidad de quitar la memoria o apagar el equipo antes de tiempo. El usuario puede solicitar esta finalización solo en el modo NORMAL, manteniendo presionado durante un tiempo determinado (5 segundos para el prototipo) el botón del panel frontal. Haciendo esto se logra descargar los datos desde el PC en un tiempo menor, ya que el fin de archivo es fundamental para recuperar los datos si se quiere optimizar el tiempo de descarga.

8.7 Uso de las PWM

Las interfaces PWM del microcontrolador tienen dos modos de uso: como salida digital de una señal modulada por ancho de pulso o como entrada/salida digital de propósito general (GPIO).

Como el bus SPI es compartido entre el MUX/ADC y la tarjeta de memoria MMC, es necesario controlar la habilitación de cada uno de estos dispositivos para que no exista un conflicto en el bus, en particular en DOUT, es decir la línea de salida de datos de los dispositivos (MISO visto desde el microcontrolador).

El bit EN de la interfaz SPI presenta fallas por defectos de fabricación del microcontrolador, por lo tanto usamos el bit GP como Chip Select (CS) para el MUX y también para el ADC, ya que en el circuito ambas señales se manejan juntas.

Para manejar el CS de la memoria, se utiliza la salida PWM1, funcionando en modo GPIO (General Purpose Input/Output). El controlar el CS de la memoria de forma independiente tiene sus ventajas al momento de implementar las rutinas que leen y escriben de la memoria.

La salida digital PWM3, también en modo GPIO, se usa para manejar el LED indicador externo.

Utilizando PWM5 como entrada de propósito general, se determina el modo de funcionamiento en el momento de encender el equipo. Si el botón ubicado en el panel frontal está presionado en el momento del encendido, se pasa al modo TEST. Si el botón no está presionado, se pasa al modo NORMAL y la adquisición comienza cuando el botón se presiona por segunda vez. En el Capítulo 10 se explica con más detalle la forma de encender el equipo.

Se usaron las restantes PWM (PWM2 y PWM4), para implementar salidas digitales moduladas por ancho de pulso. En particular, la señal de PWM2 se filtra para reconstruir una señal analógica, que se usa en el modo TEST.

LED indicador:

Como forma de controlar externamente el correcto funcionamiento de la adquisición, se agregó un LED, visible desde el panel frontal.

Este LED parpadea cuando se enciende el equipo, esto indica que se han iniciado correctamente las variables y todo el sistema en general. En el modo NORMAL indica también que se ha iniciado la memoria.

En el modo TEST sirve para controlar el número de canal que se está reconstruyendo. Cuando se inicia en modo TEST se comienza la reconstrucción del canal 0, presionando el botón del panel frontal se puede cambiar el canal que se reconstruye. La secuencia que se sigue es: canal 0, canal 1, canal 2, canal 0... Cada vez que se cambie de canal, el LED parpadeará respectivamente 1, 2 o 3 veces indicando el número de canal.

En modo NORMAL el LED parpadea periódicamente para indicar que se han grabado *N* sectores en la tarjeta de memoria. El número *N* es una variable que se elige en el momento de compilar el programa, es decir que se puede variar. Se toma *N = 6* en el prototipo para no consumir mucha energía durante una adquisición de 24 horas.

En el prototipo no se implementó un algoritmo de compresión, por lo que la frecuencia de parpadeo es casi constante. En caso de implementar un algoritmo de compresión, la tasa de datos varía al variar la forma de la señal, por lo tanto también cambiará la frecuencia con la que el LED indica la correcta grabación de los *N* sectores.

También se utiliza el LED para indicar el fin de la adquisición, encendiendo cada vez que se cierra un archivo de datos en la MMC.

Reconstrucción en modo TEST:

Para obtener una señal analógica a partir de la modulada PWM, se necesita un filtro pasabajo. Optamos por reconstruir la señal con un filtro sencillo RC, ya que lo que se busca es verificar el funcionamiento y ajustar la ganancia del equipo, no realizar un análisis sobre la señal en pantalla. Por lo tanto, para simplificar la electrónica necesaria, optamos por el filtro de la Figura 8.6.

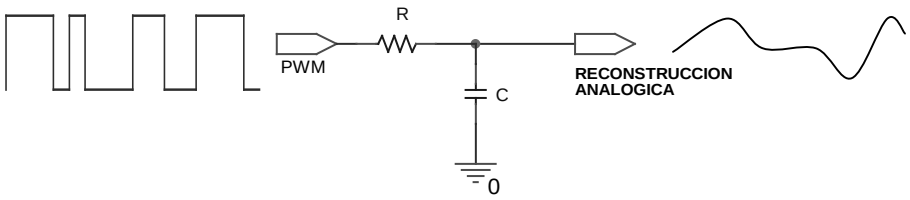


Figura 8.7 - *Circuito para reconstruir la señal analógica a partir de la modulada PWM*

Detalles de la reconstrucción PWM:

Los valores utilizados para la reconstrucción mediante PWM son los siguientes:

R = 120 KΩ y *C = 22 nF* → Esto da como resultado un polo en *60 Hz*

Para la reconstrucción PWM se utilizó una división del reloj principal HI_REFCLK de 4 y se reconstruye la señal con 1024 niveles (10 bits) de resolución.

8.8 Inicialización del sistema (RESET)

Para que el sistema quede en condiciones de pasar a uno de los modos de funcionamiento descritos antes, se deben ejecutar una serie de inicializaciones.

En primer lugar, luego de que el sistema recibe energía, lo primero que sucede es que el microcontrolador lee una palabra (32 bits) desde la dirección más baja de la memoria ROM externa del kit de desarrollo SLK. Esta palabra indica la dirección de la rutina de reset del sistema.

La rutina de reset se encarga de inicializar en primer lugar al microprocesador, asignando ciertos valores a sus registros, en particular el registro PSR, donde se indica el modo de funcionamiento y se habilitan las interrupciones. Luego inicializa el kit de desarrollo, asignando los chip select para que el procesador tenga acceso a la memoria externa.

A continuación se inicia el puntero del stack y se ejecuta la rutina encargada de copiar las variables y las funciones del programa desde la ROM a la RAM, donde serán utilizadas. Finalmente se llama a la rutina principal y comienza efectivamente el programa.

Memoria RAM y memoria ROM:

El programa está desarrollado para correr en la memoria RAM interna del microcontrolador por varias razones:

1) La memoria RAM interna del microcontrolador es varias veces más rápida que la memoria RAM externa o que la memoria ROM. Ver Tabla 8.3.

2) El microcontrolador MMC2001 cuenta con 32 KB de memoria RAM interna, que resulta suficiente para el programa de adquisición.

3) Si se utiliza una memoria lenta, se comprometen los tiempos de ejecución de algunas de las rutinas, en particular las dependientes del tiempo.

4) En una futura etapa de desarrollo de ADQCAR, el sistema se simplifica al no requerir chips adicionales.

	RAM interna	**ROM externa**	**RAM / ROM**
Rutina A	7,4 μs	27,6 μs	26,8 %
Rutina B	1,82 μs	8,32 μs	21,9 %
Rutina C	216 μs	960 μs	22,5 %

Tabla 8.3 – *Tiempos de proceso de tres rutinas en RAM interna, en ROM externa y relación de velocidades*

Todas las variables del programa se deben copiar a RAM antes de ser usadas, pero en principio no es necesario que todas las rutinas se ejecuten en RAM, solo aquellas rutinas encargadas de la adquisición de las muestras y el grabado en MMC. Las rutinas de procesamiento pueden correr desde RAM externa o desde ROM en caso de ser necesario.

En el prototipo, todo el programa y las variables están cargados en ROM externa, y desde allí se copian a RAM interna en el momento de la inicialización. Esto se debe a que la memoria ROM interna del microcontrolador es utilizada por el debugger propio del kit SLK. En una futura etapa de desarrollo, no será necesario el Kit de Desarrollo sino solamente el microcontrolador, ya que todo el programa puede residir en ROM interna y luego, al momento de iniciar el sistema, se copia a RAM interna, del microcontrolador.

8.9 Resultados: uso de memoria y rendimiento del programa

A continuación se detallarán los resultados obtenidos con el programa desarrollado en cuanto al uso de memoria y al rendimiento desde el punto de vista del tiempo de proceso utilizado.

Uso de memoria:

La Tabla 8.4 muestra el uso de memoria ROM y RAM. Estos datos se obtienen del mapa de memoria generado por el software de desarrollo CodeWarrior.

	MMC2001	**ADQCAR**	**% uso memoria**
Memoria RAM	32 KB	11,3 KB	35,3 %
Memoria ROM	256 KB	11,5 KB	4,5 %

Tabla 8.4 – *Uso de memoria ROM y RAM del programa ADQCAR y porcentaje de recursos utilizados del microcontrolador MMC2001*

La gran diferencia que se observa en el porcentaje de uso de la memora RAM y la memoria ROM es que, con excepción de las rutinas de reset, todas las demás rutinas, y por supuesto las variables, se copian de ROM a RAM al iniciar el sistema. En caso de necesitar más memoria RAM, se pueden dejar las rutinas no dependientes del tiempo (por ejemplo compresión) en memoria ROM y dejar en RAM solo las rutinas más exigentes en cuanto a tiempo.

Rendimiento del programa:

En la Tabla 8.5 se muestra la distribución de tiempos del programa ADQCAR entre ejecución y estados de bajo consumo.

	% tiempo en cada estado	
RUN	1 %	
DOZE	10 %	99 %
STOP	89 %	

Tabla 8.5 – *Distribución de tiempos entre procesamiento y estados de bajo consumo (valores aproximados, puede ser algo menor)*

Se observa que el **1 %** del tiempo el procesador está trabajando y el resto del tiempo, el **99 %** está en alguno de los dos modos de bajo consumo. Este estado de inactividad se puede interpretar como el tiempo del que disponen las futuras rutinas de procesamiento que se quieran implementar en ADQCAR. En caso de implementar una rutina se deberá tener en cuenta el incremento en el consumo del microcontrolador, si la rutina implementada es de compresión hay que considerar la relación entre el consumo adicional de procesamiento y la reducción en el consumo de la memoria, ya que el factor de compresión afecta directamente la frecuencia de su uso.

8.10 Herramientas de desarrollo – CodeWarrior

Una de las primeras decisiones que tuvimos que tomar al momento de comenzar a programar, fue la elección de las herramientas de programación que utilizaríamos.

El Kit de desarrollo MMCMB1200, prestado por el IIE, incluyó un CD con diversas herramientas de desarrollo para la nueva familia de microcontroladores de Motorola con procesador M·CORE. Todos estos programas de desarrollo trabajan sobre el sistema operativo Windows y cuentan con un ambiente gráfico.

Cuando comenzamos a estudiar las herramientas, encontramos que no todas eran funcionales, algunas estaban muy limitadas en cuanto a sus posibilidades, por ser copias de evaluación. Otras requerían una licencia para ser activadas, en algunos casos se solicitó la licencia de evaluación por 30 días pero sin éxito.

Dentro de las opciones contamos con el software “CodeWarrior for M·CORE Embedded Systems”, de la empresa Metrowerks (perteneciente a Motorola). La versión de evaluación es completamente funcional. Se requiere una licencia de evaluación, y a diferencia de otras opciones, obtuvimos la licencia sin problema, vía correo electrónico.

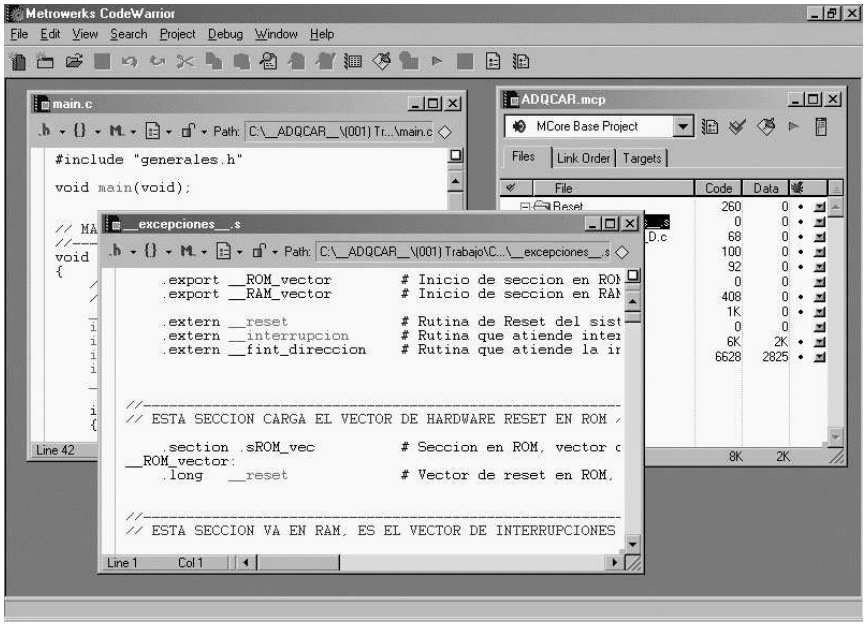


Figura 8.8 – *Ambiente de desarrollo CodeWarrior*

Una alternativa a estos sistemas de desarrollo son las herramientas GNU para M·CORE. La ventaja es que son gratuitos, pero el ambiente de desarrollo no es gráfico. Tomando en cuenta que nuestra experiencia en estos sistemas era nula, optamos por comenzar las pruebas usando el CodeWarrior, que por otra parte es un software de desarrollo muy conocido para varias plataformas, además de los procesadores M·CORE.

Una de las principales dificultades encontradas al manejar esta herramienta fue la gran cantidad de errores (bugs). Esto se debió a que la versión con la que comenzamos era la 1.0. Ese fue un problema con todos los sistemas de desarrollo para M·CORE, ya que es una tecnología nueva y por lo tanto las herramientas para manejarla están en desarrollo.

Luego de trabajar con este programa por un tiempo, y sin considerar los errores del mismo, notamos que el ambiente gráfico resultó de gran ayuda para comenzar. En la Figura 8.7 se muestra el ambiente de desarrollo CodeWarrior.

De la versión 1.0 pasamos con el tiempo a la 1.3, que corrigió algunos de los bugs de la versión 1.0. Finalmente accedimos a la versión 2.0, que es la última que utilizamos y sigue siendo la más reciente hasta la fecha. Esta versión solucionó la mayoría de los errores graves. El pasaje de la versión 1.0 a la 1.3 fue casi inmediato desde el punto de vista de la compatibilidad con proyectos creados con la versión 1.0, sin embargo, al pasar a la versión 2.0 tuvimos que estudiar de nuevo la forma de trabajar del linker, ya que la versión 2.0 no es compatible con la 1.3.

Librerías del programa:

En la Figura 8.8 se muestra el árbol de archivos que componen el programa y que se explican en la Tabla 8.6:

Archivo	Descripción
__excepciones__.s	Contiene el vector de excepciones que se carga en RAM en el inicio del sistema.
init_board_rev_D.c	Esta función es necesaria para iniciar el Kit de desarrollo SLK, se encarga básicamente de iniciar los Chip Select de la memoria externa al microcontrolador.
reset.c	Rutina de reset que se ejecuta al inicio del sistema, desde aquí se inicia el procesador, llama a la inicialización del kit y luego salta a la rutina main.
rom_copy.c	Se encarga de mover las variables y las funciones desde ROM hasta RAM.
linker.lcf	Es el linker del programa, con él se ubican los archivos en memoria.
otras_int.c	Es la rutina que se ejecuta en caso de una excepción no prevista.
__int_normal__.s	Rutina de atención a interrupciones normales, es la que implementa la adquisición de las muestras desde el ADC.
__int_fast__.s	Rutina de atención a interrupciones fast.
__iniciaMMC.s	Rutina que inicializa a la memoria MMC y la deja en modo SPI.
__leeCID.s	Rutina para leer el registro interno CID de la MMC.

__leeCSD.s	Rutina para leer el registro interno CSD de la MMC.
__borraMMC.s	Rutina para borrar la memoria MMC.
__leeSector.s	Rutina que lee un sector de la MMC.
__escribeSector.s	Rutina que escribe un sector en la MMC.
definiciones.h	Contiene todos los #define del programa, desde este archivo se pueden ajustar los parámetros de funcionamiento de todo el programa.
biblioteca.h	Solo incluye a las demás librerías.
generales.h	Es la rutina que implementa la mayoría de las funciones del programa, se sirve de las demás librerías para eso. Incluye al inicio a biblioteca.h
PilasDatos.h	Contiene funciones para manejar la estructura de pilas, en particular para agregar y quitar elementos de las pilas.
filtros.h	Implementa el filtrado digital.
MMC.h	Funciones para manejo de la MMC, en general son funciones en C que llaman a las rutinas de assembler.
formato.h	Funciones para mantener el formato de datos en la memoria MMC.
PWM.h	Funciones para manejar las PWM
main.c	Rutina principal

Tabla 8.6 – *Descripción de los archivos que componen el programa ADQCAR en el equipo*

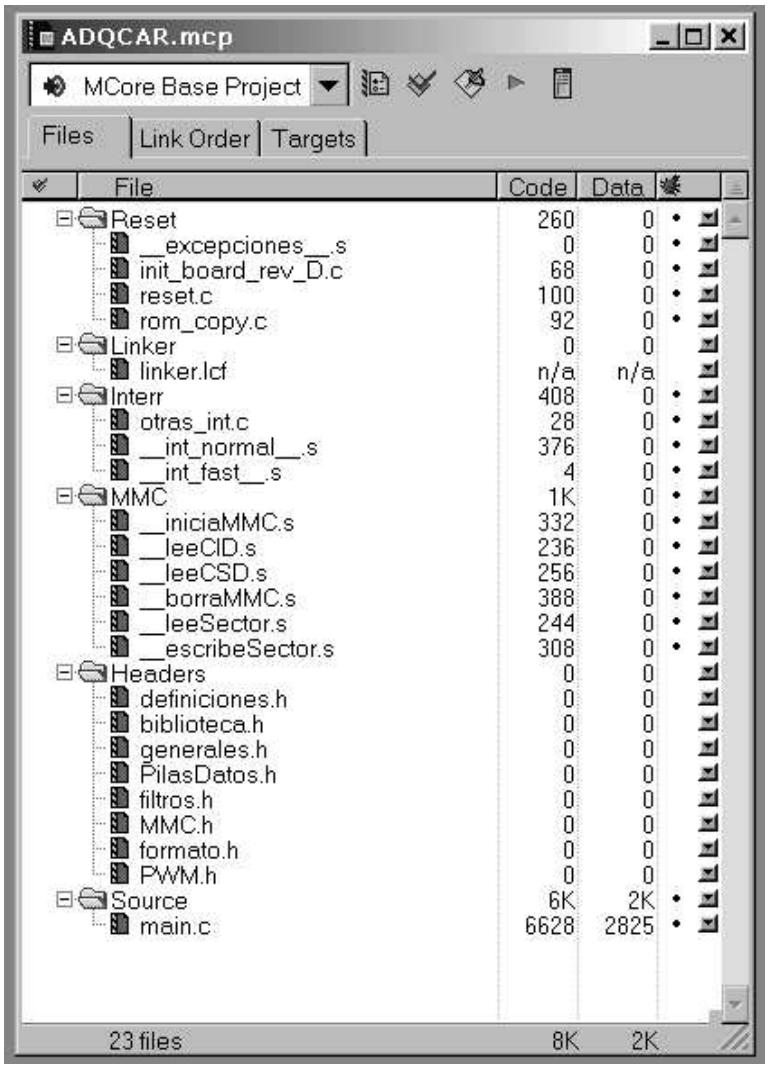


Figura 8.9 – *Árbol de archivos que componen ADQCAR*

CAPITULO 9 – ALIMENTACIÓN Y CONSUMO

9.1 Generalidades

Las características de portabilidad de ADQCAR y el hecho de que deba funcionar por más de 24 horas sin requerir cambio de pilas, exige reducir al mínimo el consumo. Esto lleva a elegir componentes pequeños y de bajo consumo en todo el prototipo, como se describió en los capítulos anteriores.

Cuando realizamos, para cada una de las etapas, la búsqueda de componentes de bajo consumo, observamos que la gran mayoría se alimentaba con 3 V o incluso menos.

La memoria MMC, que fue el primer componente seleccionado, trabaja con alimentación de 3 V. Esto nos condujo a la idea de buscar los demás componentes con la misma tensión de alimentación. Los componentes analógicos se buscaron con las mismas restricciones, aunque en las primeras versiones, la etapa de entrada se alimentó con 6 V debido a su inestabilidad. Una vez solucionado este problema, también la alimentamos con 3 V, razón por la cual todo el sistema se alimenta con esa tensión.

Para implementar el prototipo se intentó conseguir, dentro de lo posible, integrados con encapsulado DIP para facilitar su manipulación. Sin embargo, tanto el MUX-ADC como el microcontrolador son de montaje superficial.

Si bien al primero logramos soldarlo con mucha dificultad, no pudimos hacer lo mismo con el segundo ya que la separación de sus 144 pines es aún menor. Para no extender los tiempos del proyecto se decidió utilizar la placa SLK. Esta placa incluye elementos adicionales y es alimentada con 5 V.

Por simplicidad utilizamos la fuente reguladora de 3,3 V que incorpora esta placa para todo el prototipo. De todos modos, previo a la decisión de utilizar la SLK, ya habíamos estudiado una fuente de alimentación para la idea inicial.

9.2 Implementación de la fuente

Para lograr adaptar los niveles de tensión a los 3 V requeridos por todo el sistema, los componentes de mayor rendimiento son los conversores DC-DC. Como punto de partida, nos planteamos algunos requisitos que estos conversores debían cumplir. Entre ellos:

1) Corriente de salida
2) Tensión de entrada
3) Tensión de salida
4) Rendimiento

1) Debemos saber el pico máximo de corriente que debe ser capaz de entregar la fuente. Este pico se da cuando se está grabando en la memoria flash, en que la etapa de entrada, el microcontrolador y la memoria se encuentran funcionando a pleno, por lo que el valor es aproximadamente 100 mA. Estos picos de consumo ocurren cada 1,7 seg durante 3 mseg.

2) La tensión de entrada es un punto importante, porque cuanto más baja pueda ser la tensión de entrada, más energía se podrá extraer de las baterías, y como consecuencia, mayor será el tiempo de uso. Además, si bien no es un requerimiento inicial, se pretende que ADQCAR funcione con una sola batería AA.

3) La tensión de salida debe ser 3V. Aunque también planteamos la posibilidad de utilizar 2 conversores, uno para la parte digital a 3V y otro para la analógica a 6V en caso de ser necesario.

4) El rendimiento debe ser máximo (próximo al 100%) en los rangos de corrientes a utilizar para evitar pérdidas de energía en la conversión.

Existen 2 tipos de conversores, los que utilizan un inductor y los que utilizan condensadores para realizar la conversión. Dentro de estos grandes grupos, observamos que los de mayor rendimiento son los primeros, por lo que nos inclinamos por ellos.

	LTC3401	**LT1110**	**LT1073**	**MC33680**	**Unidad**
Corriente de conmutación (máx)	1000	1000	1000	>200	mA
Tensión de entrada	0,5 a 5,5	1 a 30	1 a 30	1 a 6	V
Tensión de salida seleccionable	2,5 a 5,5	>0,22	> 0,20	2,7 a 5	V
Rendimiento	> 85	70	70	75 a 85	%

Tabla 9.1 - *Comparación entre conversores DC-DC*

En el Tabla 9.1 se puede observar una comparación entre algunos de los muchos conversores estudiados. En esta se puede ver claramente que el componente más adecuado es el LTC3401.

Una desventaja que presenta este conversor es que solo se fabrica con encapsulado de montaje superficial tipo MSOP, que si bien es ideal para el producto final, dificulta en gran medida los ensayos e incluso el armado del prototipo. Por esa razón, junto con el LTC3401 importamos el LT1110, el que sí se fabrica en encapsulado DIP.

Lamentablemente, por error importamos la versión de salida fija 5 V del LT1110 (que es el LT1110-5), por lo que no podíamos obtener los 3 V deseados. De todas formas armamos un circuito para comprender su funcionamiento.

Seguidamente se intentó utilizar el conversor elegido (LTC3401), el cual planteó las dificultades de ser de montaje superficial (MSOP de 10 pines), por lo que no podíamos probarlo directamente en un protoboard. Construimos entonces un pequeño impreso donde soldarlo y así adecuarlo al protoboard, como muestra la Figura 9.1.

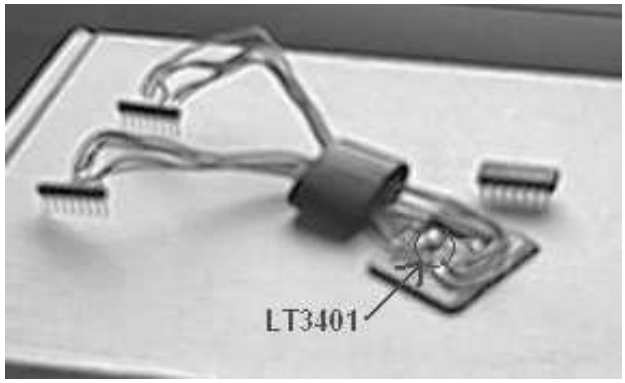


Figura 9.1 – *Adaptación realizada para el chip LT3401 para poder ser conectado a un protoboard. Se puede observar el pequeño tamaño del chip en comparación con un encapsulado DIP de 14 pines*

Luego de varias pruebas observamos en un osciloscopio que el chip no funcionaba, y aunque no logramos confirmarlo, es probable que debido a su pequeño tamaño no haya soportado la temperatura de la soldadura “manual” realizada.

Para ese entonces ya habíamos decidido el uso de la SLK como parte del prototipo, por lo que no tenía mucho sentido seguir dedicando tiempo a optimizar la alimentación. Pasamos a utilizar los clásicos reguladores 7805 y 7905, los cuales son usados en muchos equipos, y aunque no tienen buen rendimiento, se pueden conseguir en plaza y son muy fáciles de utilizar.

Para obtener 5 V a la salida, se necesitan al menos 7 V en la entrada, lo que implica usar 5 pilas de 1,5 V o una batería de 9 V. Elegimos la batería por su practicidad.

Al realizar pruebas se observó que la batería se agotaba rápidamente, por lo que se buscó una solución más simple; utilizar el propio regulador de la tarjeta SLK alimentando ésta con 4 pilas AA directamente. Previamente se debió verificar que la tensión máxima de 4 pilas no quemaran componentes en la tarjeta.

Estudiando el circuito de alimentación de la SLK, una gran variedad de hojas de datos de baterías AA (tensión máxima y resistencia interna) y realizando ensayos con una fuente de tensión variable, se llegó a la conclusión de que esto es posible, sin ningún tipo de perjuicio para la tarjeta, aún en el peor caso en el que las baterías nuevas estuvieran sobrecargadas.

Finalmente la alimentación del prototipo se simplificó al uso del regulador lineal de la tarjeta SLK. Para lograr el punto medio necesario para la etapa analógica se utilizó un operacional, en configuración seguidor, alimentado por un divisor resistivo calibrado por un preset. El banco de baterías se compuso de 4 pilas AA comunes (no necesariamente alcalinas).

9.3 Cálculos teóricos del consumo

Veremos el cálculo teórico del consumo para el caso de utilizar una tarjeta específica para el microcontrolador (no la SLK) y que la fuente de alimentación se realice con un conversor DC-DC de alto rendimiento, analizando luego el desempeño de una pila AA.

No tiene sentido realizar este análisis para el prototipo que incluye a la SLK, ya que el consumo adicional de la misma supera en unas 10 veces al consumo del resto del circuito.

Cálculos teóricos de consumo para las diferentes etapas:

a) <u>Etapa analógica</u>:

Comenzaremos analizando el consumo de este bloque, el que contiene amplificadores de instrumentación, amplificadores operacionales y resistencias. En esta etapa existen algunos operacionales que no fueron utilizados, debido al encapsulado de 4 operacionales por chip, pero también fueron considerados.

A partir de las hojas de datos podemos acotar estos consumos con los siguientes valores:

3 amplificador de instrumentación	*1,155 mA (máx)*
4 integrados con 4 operacionales	*0,156 mA (máx)*
Resistencias en general	*0,500 mA*

La máxima corriente promedio consumida es: ***1,811mA***

Este consumo es constante durante las 24 hs.

Nota: Para acotar el consumo en las resistencias, consideramos por un lado los operacionales de instrumentación próximos a la saturación en continua (por tensión electrodo piel) y el resto de las resistencias con una tensión fija de 1V, lo cual es una cota superior muy elevada. Esto también incluye el consumo de la etapa de potencia (clase AB) de los operacionales y el amplificador de instrumentación.

b) <u>Multiplexor y conversor analógico digital</u>:

En este bloque, que sigue a la entrada de datos, tenemos un chip que se encarga de multiplexar los tres canales de entrada, así como también de digitalizar las respectivas señales.

A partir de las hojas de datos del LTC1594L acotamos:

Corriente máxima	*320 μA (máx)*
Corriente mínima	*1nA (shutdown)*

Este chip es habilitado solo en el momento de multiplexar y digitalizar los tres canales y luego, al mantenerlo inhabilitado, el consumo baja a 1 nA.

Las 3 muestras se toman cada 200 Hz y el chip como cota superior se mantiene activo unos 1,21 ms. Es decir que cada 5 ms se enciende 1,21 ms, que es el 24,2% del tiempo.

La corriente máxima promedio consumida es: 0,320 * 0,242 = ***0,077 mA.***

c) <u>MultiMedia Card</u>:

Veamos primero los consumos según hoja de datos de la tarjeta.
En modo SPI el consumo se indica según su utilización:

Lectura	*< 58mA*
Escritura	*< 60mA*
Inactivo	*<150uA*

Para almacenar un ECG de 24 hs en muestras de 12 bits a 200 muestras por segundo y sin compresión, se necesitan 151.875 sectores de memoria flash (cada sector consta de 512 bytes). En total serían 74,15 MB de datos.

Debido al protocolo de comunicación utilizado para guardar los datos en la MMC, éstos se envían de a grupos de 1 sector (512 Bytes). El tiempo de grabación de cada sector no se puede definir exactamente, ya que parte del proceso es controlado por el microcontrolador interno a la tarjeta MMC y varia sensiblemente según el sector grabado y si tenía información grabada previamente. De las pruebas realizadas pudimos observar que este tiempo es levemente superior a 1ms, por lo que tomamos como cota 1,5 ms para tener un margen de seguridad. Esto implica que para grabar los 151.875 sectores se requieren 228 segundos de funcionamiento. A esto debemos sumar el tiempo de borrado inicial, que según las pruebas prácticas, es de 850 ms, lo que podemos acotar a 1 segundo. Con esto, tenemos 229 segundos de funcionamiento en 24 hs., lo que equivale a 0,265% del tiempo.

La corriente máxima promedio consumida es: (60 * 0,00265 + 0,15) = ***0,309mA***

d) <u>Microcontrolador</u>:

Originalmente consideramos el microcontrolador montado en una tarjeta expresamente diseñada con solo lo específicamente necesario para hacer funcionar nuestro equipo; ese es el caso que vamos a analizar a continuación.

Según hoja de datos del MMC2001:

Procesando	*(RUN)*	*40,00 mA*
Bajo consumo	*(DOZE)*	*3,00 mA*
Detenido	*(STOP)*	*0,06 mA*

Hemos trabajado mucho en este punto para que el microcontrolador esté la mayor parte del tiempo en modo Stop o Doze. En modo Doze la mayor parte del microcontrolador se apaga pero la SPI y el PIT continúan funcionando, sin embargo en modo Stop solo funciona el PIT que es el temporizador que nos interrumpe para comenzar el muestreo de cada canal. Así, la optimización de la elección del modo nos permitió mantener la mayor parte del microcontrolador apagado cuando no se utiliza.

Para estimar el consumo teórico monitoreamos el bus SPI utilizando un osciloscopio para identificar el tiempo en el que el microcontrolador se encuentra en cada estado.

Resultados obtenidos.

Procesando	*(RUN)*	*1%*
Bajo consumo	*(DOZE)*	*10%*
Detenido	*(STOP)*	*89%*

La corriente máxima promedio consumida es:

(40 * 0,01 + 3 * 0,10 + 0,06 * 0,89) = ***0,753mA***

e) <u>Led indicador</u>:

Tenemos que tener también en cuenta aquellos elementos como el led indicador, que enciende por un tiempo inferior a los 4 ms, una vez de cada seis sectores grabados en memoria.

El led se encuentra en serie con una resistencia de 270 Ω, lo que implica que si su caída de tensión es 1V, la corriente que consume es 8,5 mA. Tomamos 10 mA como cota superior.

Si en 24 hs. se graban 151.875 sectores, el led prende 25.313 veces, lo que representa 101 segundos, o sea el 0,12 % del tiempo total.

La corriente máxima promedio consumida es: (10 * 0,0012) = ***0,012mA***

f) <u>Fuente de alimentación</u>:

Estudiaremos el caso de utilizar un conversor DC-DC en la implementación de la fuente.

Sumando las corrientes promedio máximas consideradas para cada bloque, según se especificó mas arriba, tendremos una corriente total promedio máxima de 2,96 mA, que tendrá que ser entregada por el conversor DC-DC manteniendo los 3 V a la salida, lo que hace una potencia de 8.89 mW.

Observando las curvas de rendimiento de los conversores LT1110 y LT3401, podemos acotar este valor al 70% manteniendo un buen margen de seguridad, por lo que la batería deberá entregar 12.69 mW en las 24 hs, lo que equivale a una carga de 203 mAh.

Alimentando con una sola pila y observando diferentes hojas de datos de éstas, podemos asegurar el funcionamiento mínimo en horas según se indica en la Tabla 9.2.

Batería	Carga (mAh)	Funcionamiento (mínimo en hs)
AA alcalina alto rendimiento	3135	300
AA alcalina	2565	250
AAA alcalina	1050	100
AA recargable alto rendimiento	1200	120
AA recargable (Ni-Cd)	750	80

Tabla 9.2 – *Autonomía teórica con diferentes pila.*

Por otro lado, utilizando el programa de simulación “Micropower SwitcherCAD” de Linear Technology, para el conversor DC-DC LT1110 (el LT3401 no está disponible) con el circuito de la Figura 9.2, alimentado con una pila AA alcalina común, obtuvimos la curva de descarga de la pila en la Figura 9.3, lo que nos asegura un funcionamiento próximo a las 250 hs.

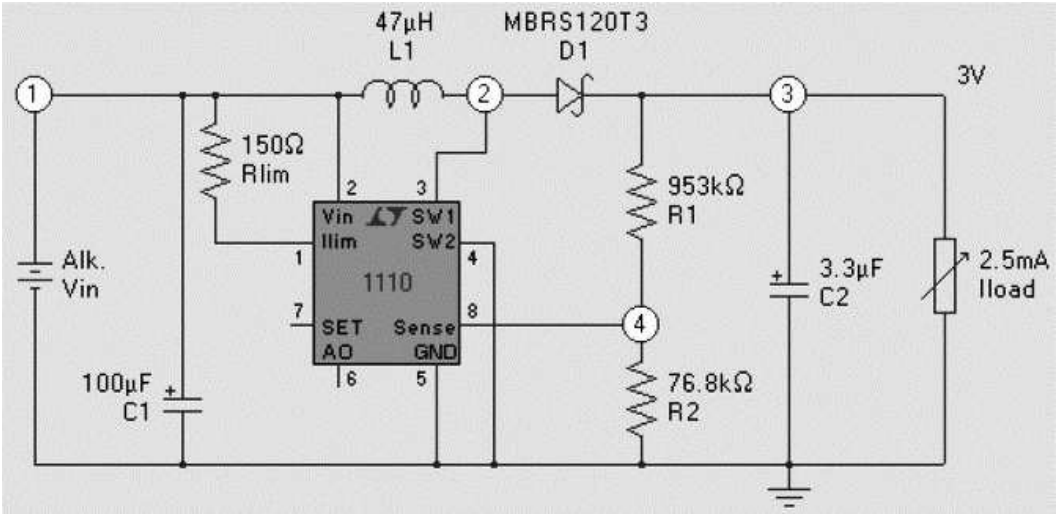


Figura 9.2 – Circuito para fuente de alimentación utilizando el *conversor DC-DC LT1110*

Para la simulación utilizamos una carga periódica de período 5 ms (período de muestreo) con el 99% a 2,5 mA y el 1% a 100 mA.

Realizamos simulaciones similares para pilas AAA y recargables AA de Niquel-Cadmio, obteniendo nuevamente resultados similares a los de la Tabla 9.2.

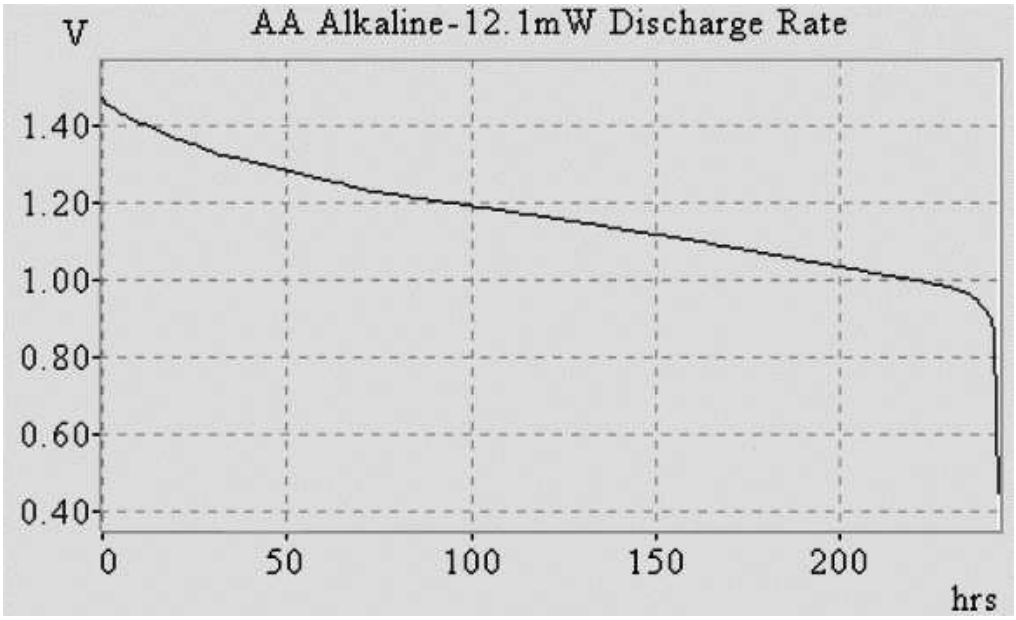


Figura 9.3 – *Curva de descarga de un pila AA alcalina.*

9.4 Medidas reales del consumo

Para el prototipo de ADQCAR como mencionamos antes, utilizamos 4 pilas tipo AA. Estas alimentan el regulador lineal de la tarjeta SLK, del cual se extrae la energía para todo el equipo.

La idea inicial fue realizar una placa específica para montar el microcontrolador, su oscilador y demás elementos mínimos necesarios para su funcionamiento. Pero por razones comparativas entre la idea inicial y lo realizado, expondremos aquí cuánto consume el equipo al utilizar el kit de desarrollo SLK.

Detallamos a continuación las medidas reales de la tarjeta SLK y la tarjeta realizada por ADQCAR, y luego el consumo del equipo total.

a) <u>Tarjeta SLK</u>:

La SLK incluye componentes como memoria RAM, así como controladores de un puerto serie, entre otros. La plaqueta está pensada para ser usada en un laboratorio y alimentada por medio de un transformador (5 V).

Para calcular el consumo de la SLK realizamos medidas prácticas, obteniendo los siguientes datos:

Procesando	*(RUN)*	*59,0 mA*
Bajo consumo	*(DOZE)*	*22,5 mA*
Detenido	*(STOP)*	*16,3 mA*

Encontramos problemas de mal funcionamiento del microcontrolador a último momento al implementar un tiempo de espera en bajo consumo. Para solucionarlo rápidamente, implementamos un tiempo de espera por programa, lo que aumenta considerablemente el porcentaje en modo RUN respecto al mencionado anteriormente. Los porcentajes reales de cada modo pasaron a ser:

Procesando	*(RUN)*	*8.5 %*
Bajo consumo	*(DOZE)*	*2.5 %*
Detenido	*(STOP)*	*89.0 %*

La corriente máxima promedio consumida para la SLK es:

(59 * 0,08 + 22.5 * 0,015 + 16.3 * 0,895) = ***20,1 mA***

Destacamos que este punto es solucionable fácilmente, basta realizar pruebas para determinar exactamente el fallo en el microcontrolador y si corresponde con la documentación encontrada al respecto. Luego podemos implementarlo aún con fallas, basta tener cuidado que éstas no influyan en el funcionamiento buscado.

b) Tarjeta realizada por ADQCAR:

Esta tarjeta incluye la etapa de entrada, la memoria MMC, el MUX/AD y el led indicador. El consumo promedio observado fue de 2.05 mA con picos no apreciables en el instrumento que se corresponden el grabado en la MMC (una vez cada 1,7 segundos).

c) Equipo ADQCAR:

El consumo promedio observado de ADQCAR en modo NORMAL es 22,4 mA, con picos no apreciables en el instrumento que superan 25 mA que también se corresponden con el grabado en la MMC.

Observando el consumo promedio máximo de la MMC calculado anteriormente, podemos acotar el consumo total del prototipo a una **corriente máxima promedio de 25 mA**. Casi diez veces mayor al calculado para el caso de no utilizar la tarjeta SLK.

En las pruebas realizadas observamos que si la alimentación es inferior a 4,6 V actúa el circuito de reset, lo que provocaría el final de la adquisición. Esto indica que cada pila puede llegar a tener 1,15 V como mínimo (suponiendo que se descargan todas igual). Además si esta tensión es superior a 6.14 V actúa un diodo Zener que

protege a la SLK. Partiendo de esto y de hojas de datos de pilas, se puede observar en la Tabla 9.3 las horas mínimas de funcionamiento de ADQCAR.

Batería	Carga (mAh)	Funcionamiento (mínimo en hs)
AA alcalina alto rendimiento	3135	73
AA alcalina	2565	60
AAA alcalina	1050	26
AA recargable alto rendimiento	1200	26

Tabla 9.3 – *Autonomía estimada para ADQCAR con diferentes pila.*

Finalmente, probamos el equipo alimentado con pilas comunes (no alcalinas) de las más económicas de plaza, observando que la autonomía fue superior a 26 hs.

Aprovechando esta adquisición, registramos la curva de descarga de una de las pilas durante las 26 horas de adquisición. Para esto conectamos directamente a la entrada de uno de los canales del MUX/AD la tensión de una de ellas, obteniendo la curva de descarga de la Figura 9.4. Se observa una descarga más abrupta al inicio, lo que se debe a que la tensión inicial de entrada a la SLK era superior a 6,14 V por lo que actuó el diodo Zener de protección. La tensión inicial medida en las 4 pilas fue 1,52 V y la final 1,17 V.

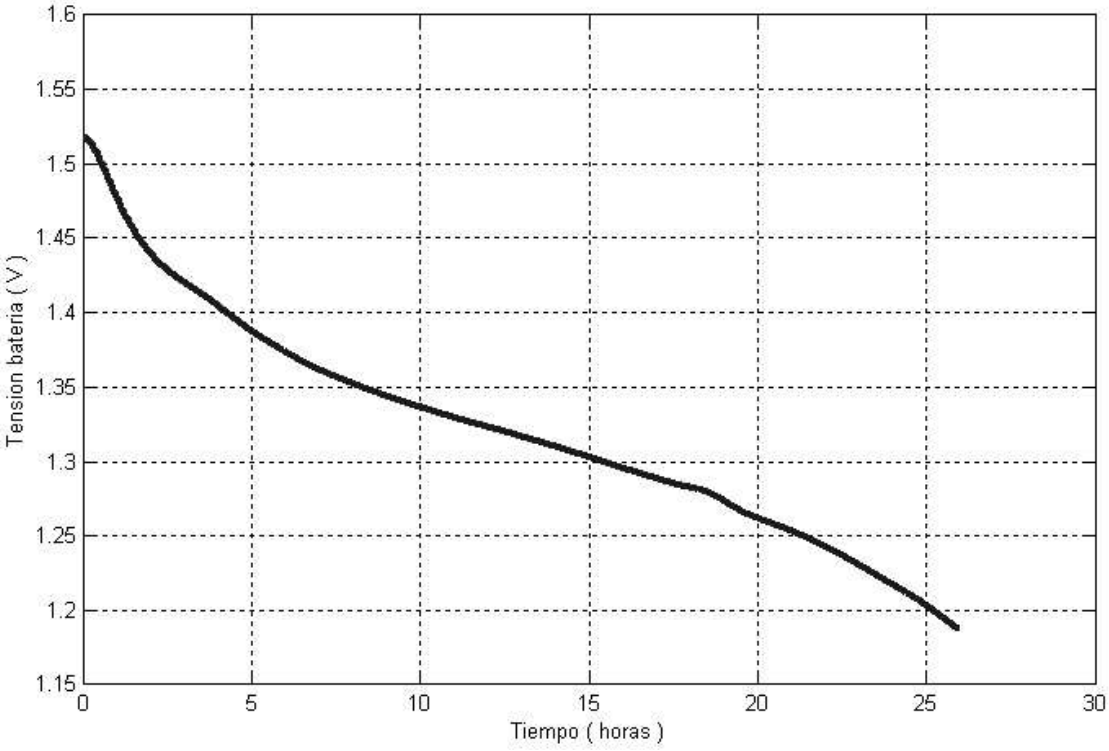


Figura 9.4 – *Curva de descarga de una pila AA común en la prueba de larga duración utilizando ADQCAR*

9.5 Conclusiones

Hemos obtenido un equipo que no solo es capaz de funcionar por más de 24 hs en su fase de primer prototipo, sino que estimamos una autonomía muy superior para una etapa final de desarrollo que incorpore un conversor DC-DC de alto rendimiento, dejando de utilizar la placa SLK.

Además, ADQCAR puede ser de interés en la investigación veterinaria, pero para esto se debe disponer de una autonomía aún mayor. Considerando el caso en que el equipo se alimente con 3 pilas AA alcalinas y un conversor DC-DC LT1173 (configuración step-down), podemos asegurar su **funcionamiento por un tiempo superior a los 30 días (720 horas).**

Otra aplicación de interés es que el propio equipo realice un análisis de los datos, lo que va a requerir un porcentaje mayor de utilización del microcontrolador en estado RUN. Si suponemos este funcionando en un 100%, el consumo teórico del equipo sería de 42 mA, lo que permite asegurar con tres pilas y el LT1173 una **autonomía de 40 hs**.

Estos valores se obtuvieron de simulaciones con el programa Micropower SwitcherCAD de Linear Technology.

CAPITULO 10 – MANUAL DE USUARIO

Holter ADQCAR

Contenido:

1. Introducción al Holter ADQCAR
2. Funcionamiento MODO TEST
3. Funcionamiento MODO NORMAL 24 horas
4. Cambio de baterías
5. Descarga de datos en computador
6. Especificaciones técnicas

10.1 Introducción al Holter ADQCAR:

En esta sección podrán conocer los conectores, llaves y todo lo relacionado al hardware de ADQCAR:

Figura 10.1 – *Imagen general del equipo ADQCAR y sus cables*

En la Figura 10.1 se puede ver el equipo ADQCAR, el que se sujeta a la cintura del paciente. Los cables de conexión terminan en broches que son colocados a presión sobre los electrodos.

A continuación mostramos una vista frontal del equipo, indicando sus conectores e interruptores:

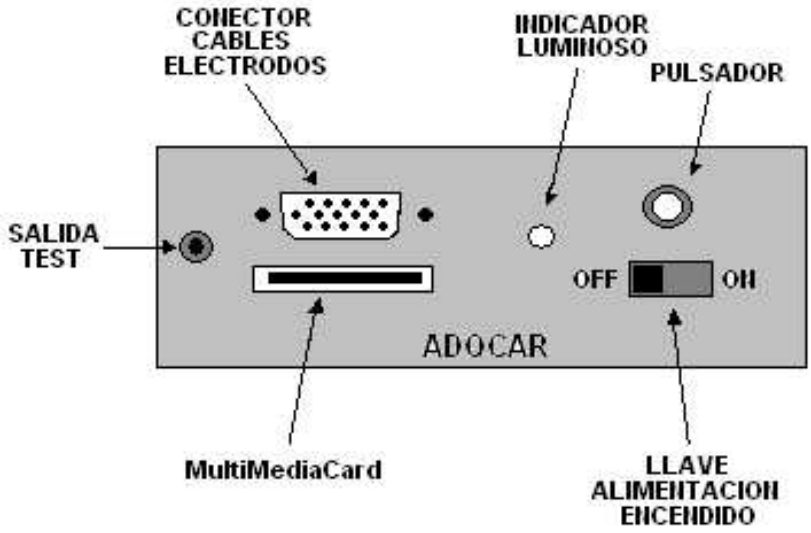


Figura 10.2 - *Vista frontal de ADQCAR*

CONECTOR CABLES ELECTRODOS	En este conector, tipo DB15 macho, se conecta el grupo de cables que va hacia los 7 electrodos del paciente.
INDICADOR LUMINOSO	Informa sobre el estado de funcionamiento del equipo mediante destellos luminosos
PULSADOR	Selecciona el modo de operación (TEST / NORMAL)
LLAVE ALIMENTACION ENCENDIDO	Conecta la alimentación del equipo
MultiMedia Card	Zócalo donde se inserta la tarjeta MultiMedia Card para la grabación de datos en el modo NORMAL
SALIDA TEST	Conector, tipo spika, para monitoreo de la señal adquiridos. Funciona en el modo TEST.

10.2 Funcionamiento MODO TEST

Antes de comenzar la adquisición de 24 horas (**MODO NORMAL**), se puede comprobar la correcta colocación de los electrodos. Para esto, al encender el equipo se selecciona el MODO TEST, con lo cual se emitirá por la **SALIDA TEST** una señal analógica, correspondiente alternativamente a cada uno de los 3 canales.

Para obtener esta señal, el equipo adquiere de igual forma que en el modo de adquisición de 24 horas. Pero en lugar de que el microcontrolador grabe las muestras en la tarjeta de memoria MMC, convierte los canales nuevamente en señales analógicas por la **SALIDA TEST**. Esta salida puede ser controlada con un osciloscopio u otro dispositivo externo, como un PC.

Activación de ADQCAR en MODO TEST:

1) Primero se debe conectar la salida TEST a un dispositivo, como ser un osciloscopio o una tarjeta adquisidora de datos, para poder visualizarla.

   Este dispositivo se deberá regular para recibir una señal de entre 0 y 3.3 voltios y con un ancho de banda acotada entre 0 y 80 Hz.

2) Encender el interruptor de alimentación, manteniendo simultáneamente el **PULSADOR** presionado hasta que el **INDICADOR LUMINOSO** parpadee dos veces.

3) Luego de 2 parpadeos del **INDICADOR LUMINOSO** el equipo estará activo en **MODO TEST.** En el dispositivo visor se podrá monitorear el ECG correspondiente al primer canal.

4) Presionando sucesivamente el **PULSADOR**, ADQCAR irá alternando la señal de la **SALIDA TEST** entre los 3 canales. En cada selección, el **INDICADOR LUMINOSO** parpadeará un numero igual de veces al número de canal seleccionado.

10.3 Funcionamiento MODO NORMAL 24 horas

Si bien el equipo permite una gran flexibilidad de programación con respecto al procesamiento y almacenamiento de datos, a los efectos del prototipo ADQCAR se programó con el llamado MODO NORMAL de adquisición.

Este modo, luego de activado grabará tres canales de ECG por un período de 24 horas, sin compresión, a 12 bits por muestra y a 200 muestras por segundo.

Activación de ADQCAR en MODO NORMAL:

1) Colocar una tarjeta del tipo MultiMedia Card (MMC) en la ranura correspondiente del equipo. Para esto se deberá prestar atención de que la etiqueta quede en la parte superior, y que el corte diagonal que tiene la MMC quede a la derecha. En

otras palabras, la etiqueta de la MMC se deberá poder leer correctamente al tenerla en la mano, y en esta posición se coloca en la ranura del equipo.

2) Luego encender el interruptor de alimentación. El **INDICADOR LUMINOSO** parpadeará 3 veces. Este parpadeo confirma el resultado correcto de un test general realizado que asegura la correcta inicialización de la MMC así como de toda la parte digital.

   Si no se coloca una tarjeta en el zócalo correspondiente, el **INDICADOR LUMINOSO** no se encenderá.

3) Presionar el **PULSADOR**. ADQCAR responderá con dos parpadeos lentos del **INDICADOR LUMINOSO**, confirmando que comenzó la adquisición y grabación de las muestras por un período de 24 horas.

4) Aproximadamente cada 3 segundos el **INDICADOR LUMINOSO** dará un rápido parpadeo indicando la grabación correcta de muestras en la MMC.

   Si se retirara la MMC, entonces el **INDICADOR LUMINOSO** dejará de parpadear.

5) Llegadas las 24 horas de adquisición, el **INDICADOR LUMINOSO** parpadeará 4 veces indicando la finalización de la adquisición, permaneciendo luego apagado.

   También se puede detener la adquisición manualmente antes de las 24 horas. Para eso, mientras el equipo está en modo NORMAL, se debe mantener presionado el **PULSADOR** durante 5 segundos aproximadamente. El **INDICADOR LUMINOSO** parpadeará 4 veces indicando la finalización.

Debido a la existencia de memorias MMC de diferentes capacidades, si ésta llegara a grabarse completamente antes de terminar la adquisición, entonces el equipo lo detectará automáticamente culminando la adquisición correctamente.

En caso de retirar la tarjeta MMC antes de terminada la grabación, o si sucede algún otro tipo de imprevisto, ADQCAR detectará el problema y luego de hacer parpadear el **INDICADOR LUMINOSO** 30 veces como aviso, pasará a la inactividad.

10.4 Cambio de baterías

La duración y por lo tanto el recambio de las 4 pilas está estipulado para ser realizado con cada adquisición de 24 horas.

Antes de comenzar un nuevo estudio de adquisición, se deben remplazar.

Forma de recambiar las pilas:

1) Apagar el equipo moviendo el interruptor de alimentación a la posición OFF.

2) Quitar los 4 tornillos que fijan la cubierta.

3) Desplazar hacia el frente la misma, dejando visibles las 4 pilas ubicadas en la parte interna posterior.

En la Figura 10.3 se puede apreciar el Holter con la cubierta desplazada.

Figura 10.3 – *Cubierta desplazada para el recambio de pilas.*

4) Retirar las pilas y sustituirlas por nuevas.

En caso de no utilizar el equipo por un tiempo prolongado, se recomienda retirar las pilas de su interior.

10.5 Descarga de datos en computador

Elementos necesarios:

- Lector de tarjetas MultiMedia Card (ImageMate SDDR-73 por ejemplo)
- Programa de descarga de datos ADQCAR
- Computador con sistema operativo Windows NT, 2000 o posterior, y con un puerto USB libre.
- Tarjeta de memoria MultiMedia Card con datos adquiridos por ADQCAR.

Instalación de ADQCAR

Instalación del programa ADQCAR:

Simplemente se copia el ejecutable suministrado, de nombre “ADQCAR.exe” en una carpeta del disco duro del computador.

Instalación del lector MultiMedia Card:

Seguir los pasos indicados en el manual del lector.

Utilización del software:

ADQCAR.exe se puede utilizar directamente para descargar los datos en el disco duro del computador, generando 3 archivos en formato binario correspondientes a cada canal adquirido, nombrados como “CANAL_0.adq”, “CANAL_1.adq” y “CANAL_2.adq”.

Este programa puede ser incorporado a modo de “driver” en aplicaciones avanzadas de análisis de señales cardiológicas. Consulte para estar informado sobre la disponibilidad de aplicaciones complementarias de ADQCAR, como por ejemplo CLASICAR.

Pasos a seguir para descargar los datos en el computador:

Colocar la tarjeta MMC con los datos adquiridos en el lector, según especifique el manual del mismo.

Hacer doble click sobre el icono del programa, el que se encargará de detectar, leer y descargar los datos en la misma carpeta donde se ubica el programa. Se crearán 3 archivos en formato binario correspondientes a cada canal ECG adquiridos.

10.6 Especificaciones técnicas

CARACTERISTICAS	OBSERVACIONES	VALORES TIPO	UNIDADES
Número de canales registrables	-	3	canales
Frecuencia Muestreo	-	200	muestras/seg
Ancho de banda de señal grabada	-	0.07 - 60	Hz
Medio de almacenaje (registro normal de 24 hs)	MultiMedia Card	32	MB
Máxima capacidad de memoria	MultiMedia Card	256	MB
Software actualizable (firmware)	-	SI	-
Alimentación (prototipo)	-	4 pilas AA	-
Compresión por software	Método estadístico sin pérdida de información	3:1 (Prevista pero no implementada)	-
Ganancia	Selector continuo	300 - 1200	-
Salida de test	-	Analógica	-
Tiempo de borrado de tarjeta de memoria MMC de 32 MB	-	850	mseg
Velocidad de volcado de datos al PC	75 MB	90	seg

Tabla 10.1- *Especificaciones técnicas*

CAPITULO 11 – TIEMPOS, COSTOS Y PRODUCCIÓN

11.1 Tiempos

Veremos los tiempos que se dedicaron al desarrollo de las diferentes actividades que permitieron la realización de ADQCAR. La Figura 11.1, muestra el porcentaje de horas dedicado a las diferentes actividades del proyecto.

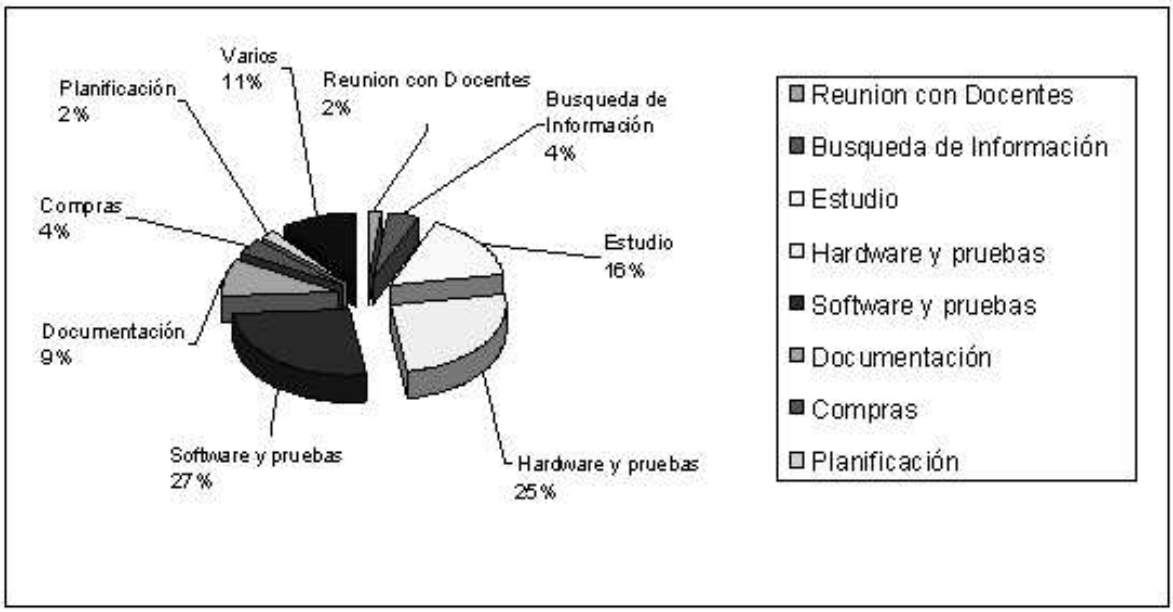


Figura 11.1 - *Porcentaje de horas dedicado a las diferentes actividades a lo largo del proyecto. (Datos obtenidos de la bitácora de trabajo)*

Agrupando las 9 actividades anteriores en 3 grupos que identifican el tiempo de dedicación a circuitería, programación y diseño de ADQCAR, resulta el gráfico representado en la Figura 11.2.

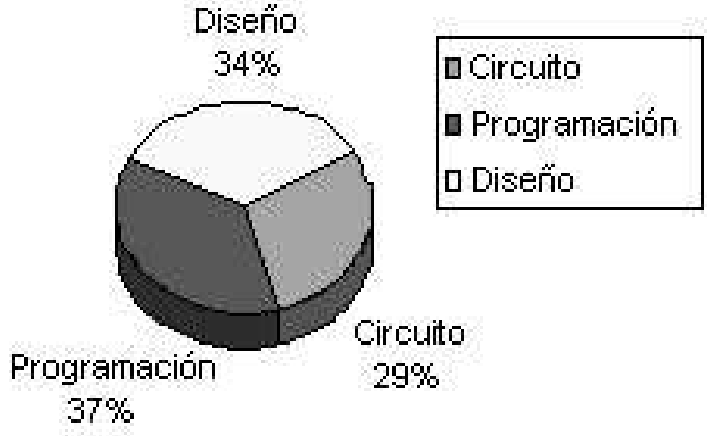


Figura 11.2 – *Porcentajes de tiempo dedicado a las actividades de Circuitería, Programación y Diseño. (Datos obtenidos de la bitácora de trabajo)*

Del análisis de estas gráficas, se puede observar un reparto de tiempos similar para estas tres actividades. Debemos agregar que a la tarea diseño se le incluyó el tiempo dedicado a tres presentaciones del proyecto ADQCAR en: Seminario de Ingeniería Biomédica (Facultad de Ingeniería, IIE), Jornadas ISTEC (Facultad de Ciencias y ORT) y Eureka (en el Palacio Legislativo).

Analizando en el tiempo la dedicación horaria a cada una de éstas tres áreas de actividades, tomamos de la bitácora de trabajo la información para componer el gráfico presentado en la Figura 11.3. En él se observa la carga horaria dedicada por mes a cada actividad.

En los primeros meses además del tiempo dedicado al diseño, aparecen varias horas de carácter programación. Éstas no se relacionan con la programación del microcontrolador, que aún no lo teníamos, sino que fueron dedicadas a la simulación de algoritmos de compresión de datos, trabajando en Matlab. Esto permitió el estudio de algoritmos de compresión sin pérdida de información que confirmaron la viabilidad del hardware a usar para cumplir los requerimientos iniciales del proyecto.

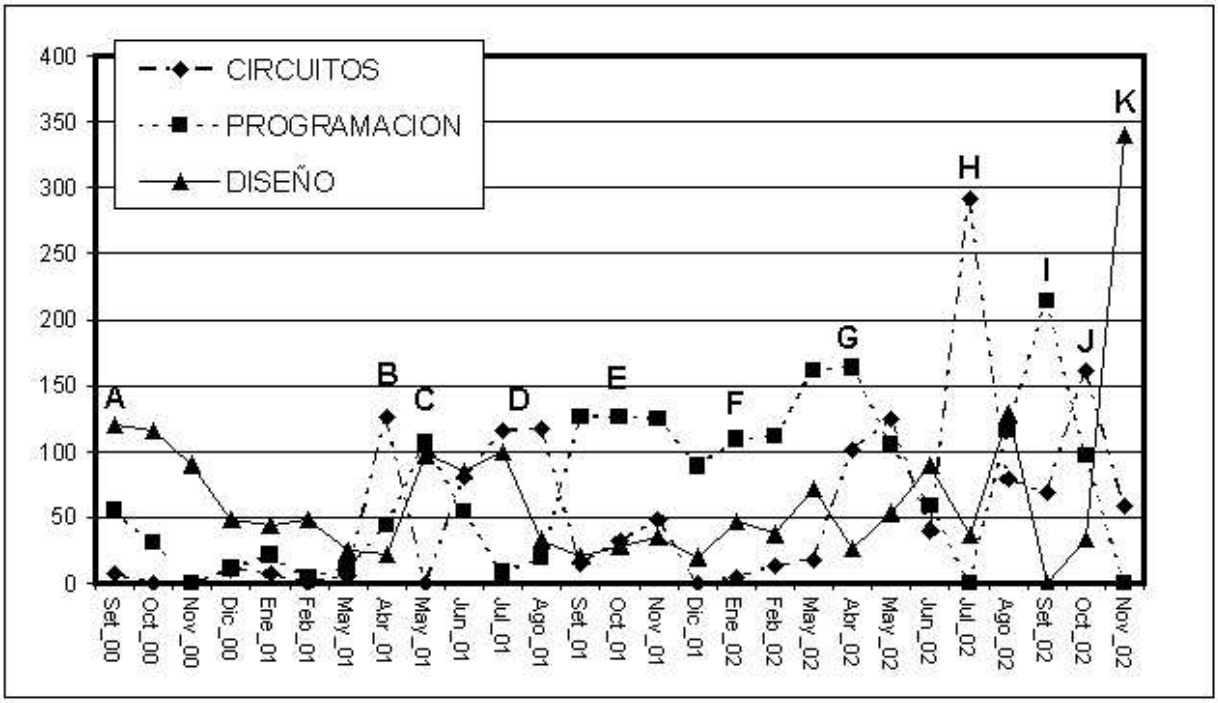


Figura 11.3 – *Carga horaria por actividad, por mes y a lo largo de todo el proyecto.*

Hitos referenciados en la Figura 11.3:

A. Reunión con F. Simini y F. Nieto, dando comienzo formal al proyecto ADQCAR, luego de tener la aprobación del IIE.
B. Nos instalamos en el NIB con el primer kit de desarrollo del microcontrolador.
C. Estudio del kit e inicio de pruebas.
D. Implementación en protoboard de circuitos de test.
E. Realización de las primeras librerías para el manejo de la MMC y el MUX_AD.
F. Logramos manejar las interrupciones del microcontrolador adecuadamente.
G. Logramos comunicarnos con la MMC exitosamente.

H. Problemas con el programa que descarga datos al PC, se comprueba que, inesperadamente, es el lector comprado el que funciona mal.
I. Se prepara presentación para Eureka
J. Realización de la tarjeta analógica definitiva.
K. Finalización de la documentación.

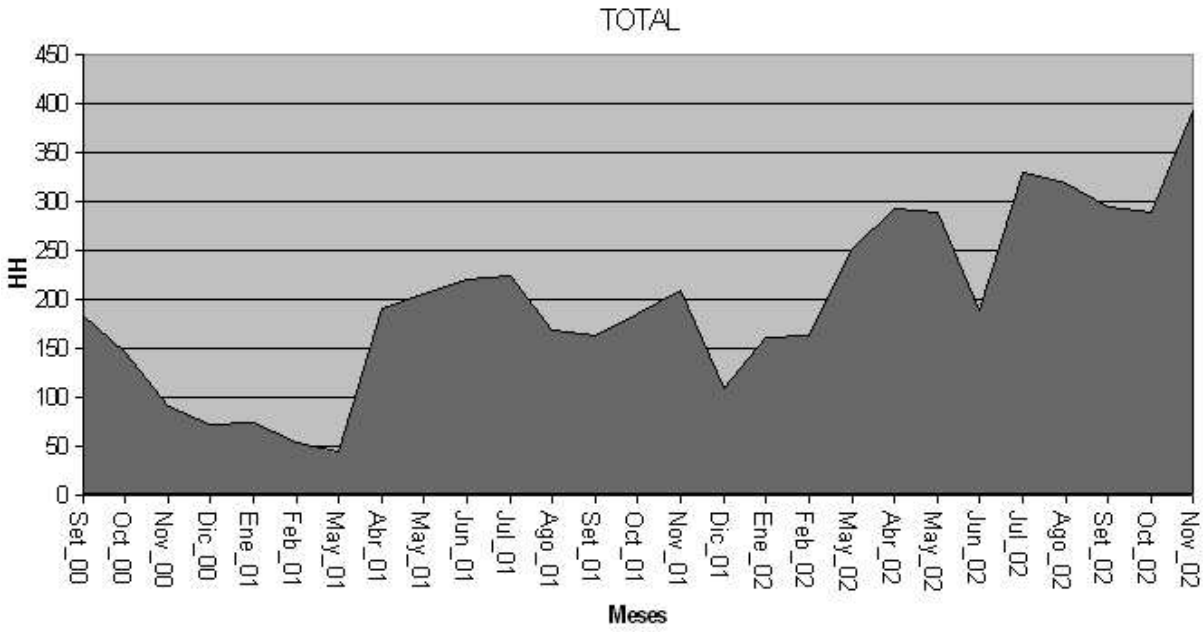


Figura 11.4 – *Horas Hombre totales, dedicadas por cada mes de proyecto*

En la Figura 11.4 se puede observar el desarrollo del proyecto, y podemos agregar algunos comentarios. Por ejemplo, al llegar a febrero y marzo del 2001, estábamos rindiendo exámenes, por lo que nuestra dedicación al proyecto se redujo.

El aumento de horas dedicadas en abril y mayo de 2001, se debieron a que instalamos nuestro lugar de trabajo en el NIB con el primer kit de desarrollo del microcontrolador, y disponiendo de un lugar fijo donde trabajar pudimos mejorar la dedicación.

A fines de diciembre del 2001 uno de nosotros se ausentó por un viaje de 20 días, y sumado al fin de año bajaron un pocos las horas dedicadas.

Un intento por terminar el proyecto para fines de mayo del 2002, implicó una carga horaria que se puede visualizar en el gráfico. Luego de esto las horas bajaron, sumado a que uno de nosotros se ausentó por su luna de miel.

En los últimos meses, con miras a terminar en diciembre, utilizamos toda la licencia posible incrementando el tiempo de dedicación para poder finalizar el proyecto.

Como complemento, en la Tabla 11.1 presentamos, con mas detalles, la dedicación horaria a las diferentes actividades del proyecto.

Mes/Año	Reunión con Docentes	Búsqueda de Información	Estudio	Hardware y pruebas de hard.	Software y pruebas de soft.	Documentación	Compras	Planificación	Varios	TOTAL
Set_00	21	25	52	0	55	0	7	22	0	183
Oct_00	0	31	66	0	30	0	0	18	0	146
Nov_00	7	3	65	0	0	0	0	15	0	90
Dic_00	0	3	43	0	11	2	11	0	0	71
Ene_01	1	30	14	0	21	0	7	0	0	73
Feb_01	5	3	35	0	0	5	0	0	5	53
May_01	0	17	7	0	0	0	7	0	11	43
Abr_01	0	0	22	0	31	0	125	0	12	191
May_01	0	0	98	0	80	0	0	0	26	204
Jun_01	5	2	76	80	39	0	0	2	15	220
Jul_01	12	21	61	116	8	6	0	0	0	224
Ago_01	0	16	15	93	17	0	24	2	2	169
Set_01	0	2	19	16	88	0	0	0	39	162
Oct_01	0	5	24	32	89	0	0	0	37	185
Nov_01	0	7	29	48	89	0	0	0	35	208
Dic_01	0	16	1	0	66	2	0	0	22	108
Ene_02	0	17	3	4	107	0	0	27	2	161
Feb_02	19	8	1	13	100	5	1	5	11	162
May_02	5	0	51	16	159	2	2	15	2	251
Abr_02	10	0	5	98	159	12	4	0	5	292
May_02	4	2	30	119	79	18	6	0	26	289
Jun_02	0	0	0	40	55	89	0	0	4	188
Jul_02	0	0	25	292	0	12	0	0	0	328
Ago_02	4	25	87	64	0	0	15	12	116	319
Set_02	0	0	0	69	41	0	0	0	173	294
Oct_02	5	0	30	161	71	0	0	0	26	288
Nov_02	8	0	0	59	0	332	0	0	0	391
Total	106	232	858	1319	1398	485	210	119	569	5294

Tabla 11.1 – *Horas hombres dedicadas a las diversas actividades, por cada mes de proyecto. (datos obtenidos de la bitácora de trabajo del proyecto)*

1.2 Costos:

Durante el desarrollo de ADQCAR, se utilizaron diferentes elementos como ser componentes electrónicos y software. En la Tabla 11.2 indicamos los mismos y el valor correspondiente.

Descripción		Proveedor	Cant.	Valor unit. (U$S)	Valor total (U$S)	Valor pagado(U$S)
Compras realizadas en el exterior						
FILTRO	LTC1068CN	Linear Technology	2	6,85	14	14
MUX-ADC 4ch	LTC1594LCS	Linear Technology	2	6,95	14	14
AMP INST	LT1101AIN8	Linear Technology	1	10,17	10	10
MUX 8ch	LTC1390CN	Digi-Key	1	4,38	4	4
MUX-ADC 2ch	LTC1288CN	Digi-Key	1	11,88	12	12
AMP INST	INA118P	Digi-Key	3	5,67	17	17
Memoria MMC SDMB-8		SanDisk	1	34,99	35	35
IMAGE MATE SDDR-12		SanDisk	1	29,99	30	30
AMP. OP.	LT1079CN	Linear Technology	10	4,17	42	42
AMP. OP.	LT1179	Linear Technology	7	4,58	32	32
CONV. DC/DC	LT1110CN8-5	Linear Technology	2	3,67	7	7
CONV. DC/DC	LTC3401	Linear Technology	2	3,1	6	6
Conector MMC	C70910A0071001		5	1,95	10	10
MUX - ADC	LTC1594LCS		2	11,25	23	23
Amp. Instrum.	INA118P		3	7,18	22	22
MEMORIA MMC 32MB	SDMB-32	SanDisk	2	34,99	70	70
LECTOR SD/MMC	SDDR-73-07	SanDisk	1	29,99	30	30
Compras en plaza						
Inductancias, capacitores, etc.		Mundo Electrónico	varios		15	15
Conectores varios		Eneka	varios		4	4
Zócalos, Conectores, Leds, etc.		Eneka	varios		15	15
Electrodos para ECG		Medi Import Ltda.	varios		15	15
Condesadores y resistencias		Eneka	varios		6	6
Pilas			varios		18	18
Insumos para crear PCB	Press'NPeel, acetona, placa, mecha, etc		varios		18	18
Kit de desarrollo de Motorola, prestado por el IIE						(*)
MMCCMB1200					625	0
MPFB1200	No está disponible a la venta.				0	0
MMC14EBDI02					569	0
Software CodeWarrior for M-CORE Embedded System 2,0					4995	0
Kit SLK	Valor: 10 unidades por 650 dólares				65	0
Microcontrolador MMC2001 (incluido en el kit SLK y en la MMCCMB1200, cuesta 11,5 dólares)					0	0
TOTALES (dólares)					**6722**	**468**

(*)Precios actualizados al mes de diciembre de 2002.

Tabla 11.2 – *Valor de los componentes utilizados en el desarrollo de ADQCAR. (precios en U$S)*

Los kits de desarrollo utilizados, fueron prestados por el IIE para poder desarrollar ADQCAR, pero para poder evaluar el costo de los mismo en caso de tener que comprarlos, agregamos en la tabla el precio correspondiente de cada uno.

Para la creación de ADQCAR, se utilizó el software CodeWarrior, pero debemos aclarar que solo se usó a modo de debugger no incluyendo ninguna de las librería que éste trae. Esto fue tenido en cuanta para bajar los costos del equipo, ya que para ADQCAR creamos el 100% de la librería utilizada, sin necesitar pagar a terceros por las mismas.

Para el formato de grabación en la tarjeta MMC, también desarrollamos un formato propio de grabación, lo que evitó comprar el kit de desarrollo del fabricante, que permite manejar el formato FAT en la memoria desde el microcontrolador, y que cuesta U$S 2545 (Precio de marzo de 2001).

Luego de estas aclaraciones, y viendo la Tabla 11.2, deducimos que el costo en equipamiento para el desarrollo de ADQCAR fue de 6722 dólares. Aunque solo se requirió pagar 468 dólares durante el desarrollo, obteniendo el resto del equipamiento a modo de préstamo.

Las Horas Hombre (HH) dedicadas fueron 5294, por lo que considerando un costo de 20 dólares por HH, da un total de 105880 U$S en costo de desarrollo.

Sumando las HH y gasto en materiales, el desarrollo costó:

6722 + 105880 = 112602 U$S

y se realizó en un plazo de 28 meses.

Evaluación de la fabricación de más equipos similares al prototipo:

Evaluamos ahora el costo de fabricar 1, 10, 100 o 1000 equipos como este, teniendo en cuenta que, por razones de mercado, es recomendable invertir en una nueva etapa de desarrollo para adaptar mejor el equipamiento al usuario final.

Para esto incluiremos los costos de desarrollo y estimaremos el costo involucrado en multiplicar el mismo prototipo ADQCAR.

Considerando solo los componentes que forman el prototipo, los elementos auxiliares para armarlo, y la mano de obra involucrada, el costo de ADQCAR lo estimamos en:

Componentes del equipo portátil (sin pilas ni tarjeta MMC): *230 U$S (*)*
HH par el armado del equipo: *30 HH de mano de obra simple (240 dólares aprox.)*

Equipamiento auxiliar básico:

Lector para tarjetas MMC:	*30 U$S*
2 Tarjetas MMC de 32 MB:	*35x2 = 70 U$S*

Estos datos nos permite construir la Tabla 11.3.

Consideraciones	1 unidad	10 unidades	100 unidades	1000 unidades
Considerando el valor de los equipos prestados por el IIE	**112602**	**11683**	**1591.3**	**582.1**
Sin considerar el valor de los equipos prestado por el IIE	**106348**	**11058**	**1529**	**575.9**

Tabla 11.3 – *Costos estimados de fabricar mas prototipos similares a ADQCAR (precios en U$S)*

11.3 Evaluación de un modelo comercializable

Pensando en la futura producción de ADQCAR, habría que invertir en desarrollar el mismo para poder adaptarlo mejor a los intereses del mercado.

Las mejoras que habría que realizarle al equipo serían las siguientes:

1. Se dejaría de utilizar la SLK, diseñando una nueva tarjeta que evite componentes innecesarios.

2. Sustitución de componentes electrónicos por elementos de montaje superficial.

3. Implementación de un nuevo bloque de alimentación, que permita utilizar una pila tipo AAA.

4. Incluir el software de compresión estadística estudiado, para: aumento de capacidad de almacenamiento y aumento de velocidad de descarga de datos.

5. A partir de las características anteriores, el tamaño y peso del equipo se podrá reducir para llegar a algo similar a un teléfono celular de los más pequeños.

Evaluación de costos para la fabricación de un nuevo modelo:

Se estima que esta nueva etapa de desarrollo, que permitiría contar con un equipo mucho más competitivo e interesente para el mercado, requeriría un 30% de HH que las ya dedicadas para lograr ADQCAR. El costo de materiales por pasar a utilizar componentes de montaje superficial no superaría el orden de lo ya invertido, y no se requeriría invertir en nuevos kits de desarrollo.

Evaluando costos, entendemos que lo mejor es enviar a construir los PCB al exterior, y debido a las dificultades de soldar el microcontrolador (de 144 pines) en la nueva tarjeta, también lo más conveniente es mandarlo realizar. Aunque esta producción se podría realizar en Uruguay, ya que empresas como CCC disponen de

la tecnología necesaria, evaluaremos para este desarrollo el contratar en Estados Unidos la realización de las placas.

Consideraciones:

HH para el nuevo desarrollo:

1588 HH (a 20 U$S/h da un valor de 31760 U$S, 30% de lo ya invertido)

Materiales pruebas para nuevo desarrollo:

500 U$S (redondeo de 468 dólares)

Envío a realizar un PCB a EEUU y componentes:

PCB: 100 U$S. Existen ofertas de 33 dólares por placa, pero utilizamos datos más acotados. Al solicitar 10, 100 o más placas, los precios descenderán de forma importante, pero nos manejaremos con el de 100 dólares por no conocer con precisión esos valores.

Componentes: 200 U$S. Igualmente, a mayor cantidad este valor descendería pero no lo tendremos en cuenta.

Costo para el armado final del equipo:

Estimamos 25 HH. Este equipo requeriría menos horas para su montaje, debido a que contará con menos conexiones y soldaduras que las requeridas en el prototipo. El costo de mano de obra sería de 200 dólares, tomando en cuenta sueldos en pesos uruguayos y en la actualidad. Se incluyen leyes sociales.

Otras consideraciones a resaltar, es que costos como la caja del equipo, tiempo para realizar compras y demás, descenderán al considerar un numero cada vez mayor de equipos Holter a construir. Aclaramos que estos costos los consideramos fijos e independientes de la cantidad de aparatos a fabricar, al realizar las evaluaciones de la Tabla 11.4.

Considerando	1 unidad	10 unidades	100 unidades	1000 unidades
Nueva inversión a partir del prototipo	**32360**	**3686**	**818.6**	**531.9**
Inversión desde cero, sin considerar kits de desarrollo	**138708**	**14321**	**1882**	**638.2**
Inversión completa considerando los kits de desarrollo utilizados	**144962**	**14946**	**1944.6**	**644.5**

Tabla 11.4 – *Costos de 1, 10, 100 y 1000 equipos Holter, considerando nueva etapa de desarrollo que los adecua para su comercializables (precios en U$S)*

Estos datos son muy generales, y optimizando tanto la producción como la compra de componentes, pensamos que se pueden llevar a valores inferiores.

Complementando nuestros cálculos de costos, averiguamos el precio final de productos similares y para evaluarlos presentamos la Tabla 11.5 comparativa.

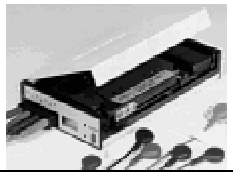
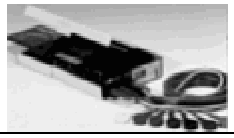
Marca/Modelo	**ADQCAR**	**Advanced Bio** *Cassette Recorder* **DM-400-7**	**Advanced Bio** *Flash Card Recorder* **DL-700**	**IQmark** *Digital Patient Recorder*
Precio en origen	644.5 U$S	945 U$S	1095 U$S	1500 U$S
País	URU	USA	USA	USA
Canales	3	3	3	3
Tiempo grabación	> 24hs	24hs	24hs	24hs
Tipo almacenaje	MMC	Cassette	Flash PCMCIA	Compac Flash
Capacidad memoria	Hasta 256 MB	(analógico)	Hasta 40 MB	48MB y 96MB
Banda	0.05 – 60 Hz	-	0.05 – 50 Hz	0.05 – 100 Hz
Bits/muestra	12	(analógico)	8	8
Muestreo	200 muestras/seg	-	128 muestras/seg	128 muestras/seg
Tamaño	*(prototipo) 160x115x45*	-	150x89x23mm	113x70x26mm
Peso	*No se tomó en cuanta en esta etapa del diseño*	-	285 g	113 g
Baterías	1 pila AA	-	2 pilas AA	2 pilas AA
Imagen	*(Prototipo)* 			

Tabla 11.5 – *Comparación de ADQCAR con modelos comerciales*

Los precios presentados en la Tabla 11.5 no se pueden comparar directamente, debido a costos de importación e impuestos, pero la mostramos a modo de comparación. Se pueden observar las buenas características técnicas de ADQCAR frente a otros equipos del mercado, y también que su costo previsto muestra la viabilidad de su producción.

Otros costos ocultos como los de mantenimiento hacen mucho mas interesante la propuesta comercial de ADQCAR.

Conclusiones:

Se han logrado los objetivos planteados inicialmente. El prototipo desarrollado, en su estado actual, puede ser utilizado para adquirir 3 canales de ECG ambulatorio durante un período de 24 horas, solo es necesario insertar una memoria de capacidad adecuada.

Otras posibles aplicaciones de ADQCAR es el estudio de algoritmos de compresión o de detección de QRS, ya que el software es fácilmente actualizable y existen recursos disponibles, tanto de tiempo como de memoria. También se puede

implementar un detector de eventos, solo se requieren algunas modificaciones en el programa ADQCAR.

Se pueden adquirir otras señales de origen biológico agregando la circuitería adecuada, puede resultar de interés para el médico la adquisición simultánea de otras señales además del ECG.

Se destaca el desarrollo de dos nuevas tecnologías que hicieron posible a ADQCAR, y que aún no se habían desarrollado en un ambiente universitario local. Ellas son el procesador M·CORE y la memoria MultiMedia Card. El primero es la nueva generación de microcontroladores de Motorola, y está siendo utilizado para dispositivos portátiles como teléfonos celulares. La memoria MultiMedia Card tuvo un desarrollo constante desde que fue seleccionada hasta el día de hoy, bajando la relación precio/capacidad.

Queda planteada además la posibilidad de llevar este desarrollo a una etapa comercial, demostrando con la experiencia lograda, que es posible diseñar y fabricar en Uruguay este tipo de equipamiento. Considerando también el precio de los mismos en el exterior, vemos que puede ser rentable su fabricación en nuestro país.

La tecnología utilizada, como se planteó desde un primer momento, abre las puertas a diversos tipos de equipamientos médicos que no difieren mucho para su construcción de la plataforma ADQCAR. Además, al reutilizar la tecnología estudiada en equipamientos similares, los costos de desarrollo serán menores y aumentarán la viabilidad comercial para la fabricación.

ANEXO 1 – COMPRESIÓN

Para almacenar en un Holter de tres canales, un ECG de 24 hs con muestras de 12 bits, tomadas a 200 muestras por segundo y sin compresión, se necesitan 74,15 MB de memoria. La descarga de estos datos en un PC desde una memoria flash demora 1,5 minutos aproximadamente.

Este tiempo puede considerarse prolongado cuando se realizan estudios frecuentemente. O, si por ejemplo deseamos usar este equipo para investigación (por ejemplo veterinaria), los datos pueden aumentar significativamente y con ello el tiempo de descarga.

También podríamos pensar en aumentar la frecuencia de muestreo para estudios particulares, lo que también traería un aumento en la cantidad de bytes a almacenar.

Todos estos planteos hicieron que estudiáramos algunos algoritmos de compresión para analizar los requerimientos mínimos de la inteligencia del equipo.

Primeramente estudiamos los algoritmos con pérdida de información como AZTEC, FAN y SAPAN, los que han sido muy utilizados. Luego realizamos pruebas en Matlab con aproximaciones polinómicas obteniendo resultados en compresión entre 3 y 8, con “muy buena definición”. Pero la desventaja de estos algoritmos es que tienen pérdida de información, y la forma que utiliza un matemático para expresar la proximidad de la señal comprimida a la señal real es muy distinta a la del médico. Puede ocurrir que un ECG, que matemáticamente se aparta poco del original, sea una peor aproximación desde el punto de vista médico que otro que se aparta más.

Es por esto que decidimos buscar algoritmos, pero sin pérdida de información, y además que fueran sencillos de implementar para tener una idea del cálculo necesario para una compresión mínima.

Los métodos de compresión de uso común se pueden separar en 2 ramas: los basados en un esquema de diccionario y los basados en un método estadístico. En un principio los métodos basados en diccionario fueron muy populares, pero a medida que el poder de cálculo aumenta, es posible combinar una codificación aritmética con un modelo estadístico poderoso.

Los métodos de compresión basados en diccionarios reemplazan un grupo de símbolos por otro de largo fíjo, como por ejemplo la compresión LZW.

Los métodos estadísticos, en cambio, codifican símbolos uno a uno. Los símbolos son codificados en códigos de largo variable. El largo de este código varía según la probabilidad o frecuencia del símbolo. Probabilidades bajas se codifican con más bits y las altas con menor cantidad.

Es claro que en cierta forma ambas ramas utilizan la estadística, pero los métodos estadísticos lo hacen de una forma más “pura”.

Estudiamos entonces codificación de Huffman y codificación aritmética, siendo esta última la que nos interesó más.

Código de Huffman

Este código es muy conocido. Describe un método para crear una tabla de código para un conjunto de símbolos y sus probabilidades.

El código de Huffman asigna un código de salida para cada símbolo de entrada, por lo que la salida más corta que existe es 1 bit y la más larga puede ser considerablemente más larga que la propia entrada. Esto depende de sus probabilidades.

El problema con éste método es que los códigos de salida tienen un número de bits entero.

En general, el número óptimo de bits que puede ser usado por cada símbolo es:

$$\text{Número óptimo de bits} = \log_2(1/p)$$

donde p es la probabilidad del símbolo en cuestión

O sea que, si la probabilidad de un símbolo es 1/3, el número óptimo de bits sería 1,6, en cambio este código tiene que asignarle el 1 o el 2, no hay otra opción.

Aún peor sería el caso en que la probabilidad de un símbolo sea el 90%, lo que implica un largo de código de salida de 0,15 bits. En cambio Huffman le asignaría como mínimo 1 bit, lo que es 6 veces mayor al caso óptimo.

Otro problema es cuando se intenta utilizar código Huffman en forma adaptiva, ya que en el caso anterior la codificación era única para transmisor y receptor, pero para esto debe existir forma de transmitir la tabla de códigos lo que generaría una sobrecarga en la propia codificación. Además la reconstrucción de esta tabla es costosa en términos de memoria y cálculo.

Esto hace que el código no sea óptimo, motivo que nos hizo buscar una mejor opción.

Código Aritmético

El código aritmético se caracteriza por tomar un gran número de símbolos de la entrada y dar a la salida un número comprendido entre 0 y 1. Este número puede ser decodificado en forma única por el receptor y así obtener todos los símbolos iniciales.

La idea es asignarle a cada símbolo un rango dentro de [0,1), de forma que el tamaño debe ser igual (a menos del borde superior) a la probabilidad del símbolo y la intersección de todos sea el conjunto vacío. Con esto tendremos que la unión de

todos los rangos debe ser el propio rango [0,1]. El orden de esta asignación no importa, siempre y cuando el receptor y transmisor tomen el mismo.

Ejemplo:

Tomemos los símbolos de "BILL GATES"

Símbolo	Probabilidad	Rango
ESPACIO	1/10	0.00 - 0.10
A	1/10	[0.10 - 0.20)
B	1/10	[0.20 - 0.30)
E	1/10	[0.30 - 0.40)
G	1/10	[0.40 - 0.50)
I	1/10	[0.50 - 0.60)
L	2/10	[0.60 - 0.80)
S	1/10	[0.80 - 0.90)
T	1/10	[0.90 - 1.00)

Entonces para indicar la letra “B” basta con un número comprendido en [0.20, 0.30).

Lo que debemos hacer, una vez codificada la primer letra, es codificar la segunda dentro del rango de la primera en forma proporcional. O sea que para la “I” el rango es [0,50; 0,60), que al codificarlo dentro de [0.20; 0.30) en forma proporcional obtenemos [0.25 ; 0.26).

Así, el algoritmo a seguir sería:

```
low = 0;
high = 1;
while “hay símbolos de entrada” do
{
    leo símbolo “S”;
    S_low = mínimo del rango de S;
    S_high = máximo del rango de S;
    rango = high – low;
    high = low + rango * S_high;
    low = low + rango * S_low;
}
rango salida = [low ; high)
```

Con este algoritmo obtenemos lo siguiente:

Símbolo	Low	High
-	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
ESPACIO	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Luego, cualquier palabra en el rango [0.2572167752 ; 0.2572167756) codifica a la palabra "BILL GATES". Tomemos 0.2572167752.

Para decodificar se procede de forma similar; 0.2572167752 pertenece al rango de la letra “B”, por lo que el primer símbolo es esta letra. Para obtener la segunda, debemos transformar el rango, eliminando la letra “B”. Esto se realiza según el siguiente algoritmo:

```
codigo = entrada
Hacer
    Encontrar el símbolo cuyo rango contiene al código de entrada.
    Saco el símbolo
    S_low =  low del simbolo encontrado;
    S_high = high del símbolo encontrado
    rango = S_high – S_low
    codigo = codigo - S_low
    codigo = codigo / rango
hasta que no haya más símbolos
```

Así al decodificar obtendremos:

Código	Símbolo de salida	Low	High	Rango
0.2572167752	B	0.2	0.3	0.1
0.572167752	I	0.5	0.6	0.1
0.72167752	L	0.6	0.8	0.2
0.6083876	L	0.6	0.8	0.2
0.041938	ESPACIO	0.0	0.1	0.1
0.41938	G	0.4	0.5	0.1
0.1938	A	0.2	0.3	0.1
0.938	T	0.9	1.0	0.1
0.38	E	0.3	0.4	0.1
0.8	S	0.8	0.9	0.1
0.0	-	-	-	-

Si bien estos algoritmos son muy sencillos, a primera vista parecen ser impracticables, porque se tiene que trabajar con punto flotante.

Esto se puede transformar muy sencillamente a trabajar con enteros, solo hay que tener presente que puede ocurrir un desborde, pero se prevé una solución.

De esta forma, para codificar en binario se comienza con el valor:

```
HIGH:   99999
LOW:    00000
```

o mejor, como trabajamos con bits.

```
HIGH:   11111111111.....1111
LOW:    00000000000.....0000
```

No explicaremos aquí todos los detalles al respecto, pues con lo expuesto alcanza para darnos cuenta de lo sencillo que resulta el algoritmo para codificar.

Pero, por qué puede ser útil este algoritmo para ADQCAR?

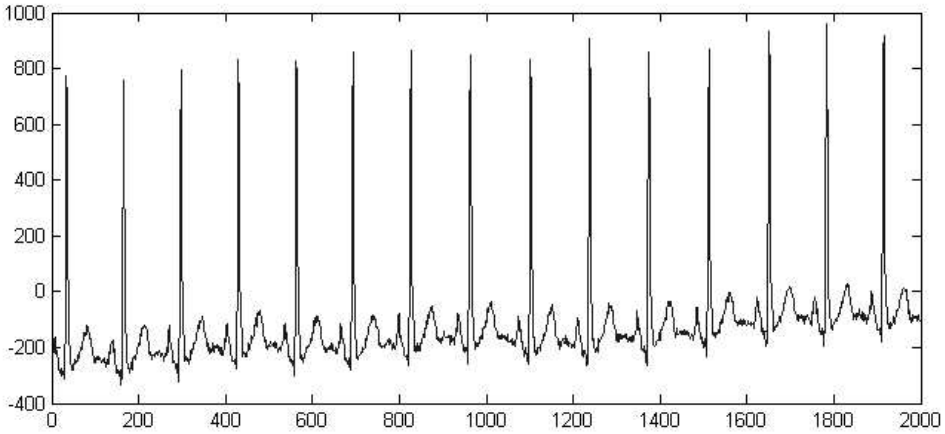


Figura A.1 - *ECG*

En el caso de no hacer compresión necesitamos guardar 12 bits por muestra, pero si observamos la Figura A.1, podemos observar que la gran parte de los valores se encuentra en un entorno dado. Si además consideramos la diferencia entre muestras consecutivas Figura A.2 y Figura A.3, este entorno se reduce aún más, tomando la mayor cantidad de muestras un valor entre ±30, y algunas pocas un valor mayor.

Esto permitiría codificar la diferencia en la mayoría de los casos con 6 bits, por lo que prácticamente estaríamos hablando de compresión de factor 2. Hemos realizado alguna pruebas en Matlab obteniendo resultados de factor 2.8, simplemente tomando una estadística fija. Por esto consideramos que un leve manejo de la estadística, seguramente adaptatiiva, no bajaría de un factor 3.

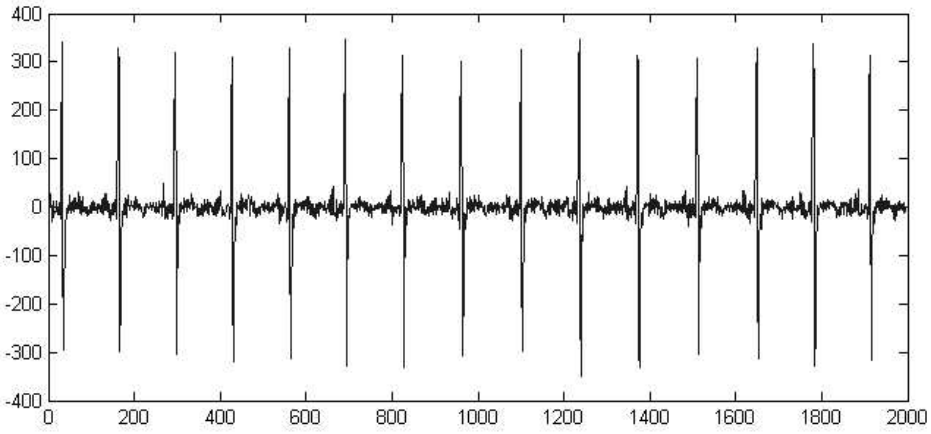


Figura A.2 – *ECG de la Figura A.1 tomando diferencias*

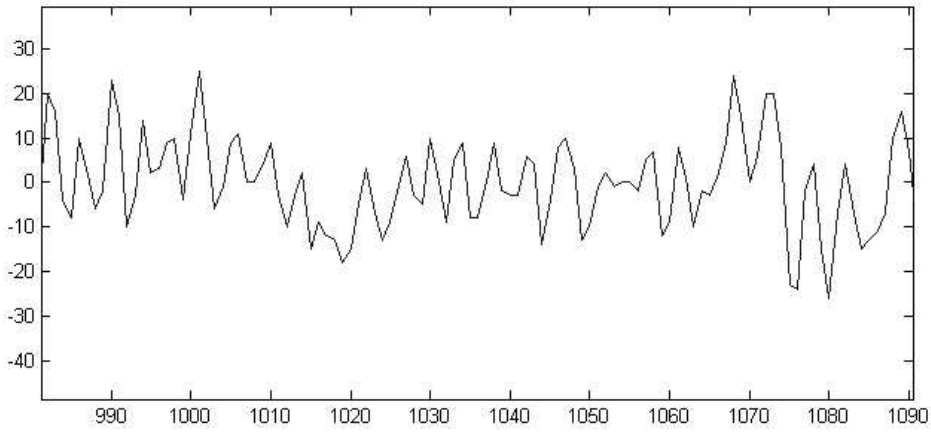


Figura A.3 – *Detalle ampliado de la Figura A.2 mostrando el intervalo entre picos (QRS)*

Dado que la compresión no es el objetivo del proyecto se dio por concluido el estudio con esta aproximación. (Referencias [16] a [25]).