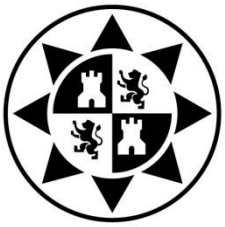


Universidad
Politécnica
de Cartagena

industriales
etsii UPCT

Simulador de biopotenciales reprogramable basado en microcontrolador 18F14K50 con comunicaciones USB y convertidor D/A por SPI

Titulación: I.T.I. Electrónica Industrial
Intensificación: Tecnología Electrónica
Alumno/a: Alberto García Salinas
Director/a/s: Joaquín Roca González

Cartagena, 20 de junio de 2014

ÍNDICE

Capítulo 1

Introducción y objetivos

El objetivo principal de este proyecto, es el diseño de una tarjeta que actúe como simulador de biopotenciales del cuerpo humano, en este caso concreto se trabaja con las señales de electrocardiograma (ECG), señal proveniente del corazón, y de electroencefalograma (EEG), señal producida en el cerebro.

Dicho simulador puede ser usado en entornos clínicos como objeto de ayuda para la calibración de los equipos de medición de parámetros vitales, así como, en docencia con el fin de ofrecer una visualización realista de algunas señales generadas por el cuerpo humano, sin necesidad de realizar mediciones "*in situ*".

El proyecto se ha estructurado en torno a diferentes bloques con el fin de facilitar el correcto y pleno entendimiento del mismo; en ellos se describen las características funcionales del sistema de simulación, además de los trabajos software y hardware efectuados para la elaboración del proyecto.

Es de esperar, que la realización de este proyecto contribuya y pueda ser exportado para posteriores estudios en el campo de la docencia en la Universidad Politécnica de Cartagena; para su futura visualización en las asignaturas del campo de la Ingeniería Biomédica.

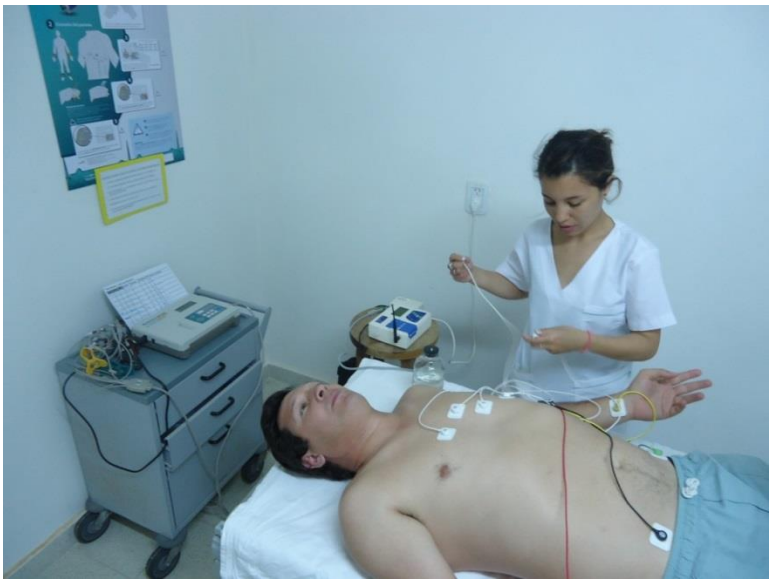


Fig. 1.1 - Paciente sometido a un electrocardiograma

Fig. 1.2 - Paciente sometido a un electroencefalograma

Capítulo 2

Revisión del estado del arte

2.1 - Origen de los biopotenciales

Los biopotenciales, es decir, las señales eléctricas de naturaleza biológica que se pueden registrar en la superficie de la piel (como son el electrocardiograma ECG, o el electroencefalograma EEG) tienen su origen en la actividad eléctrica celular.

La membrana celular es la envoltura de la célula, cuyo espesor es de unos 80 Amstrongs. A través de ella se efectúan los intercambios necesarios con el medio que la rodea; regula el paso de las sustancias, ya sea en un sentido como en el otro; es selectiva, pues permite el paso de los elementos necesarios provenientes del exterior y a través de ella se eliminan los desechos.

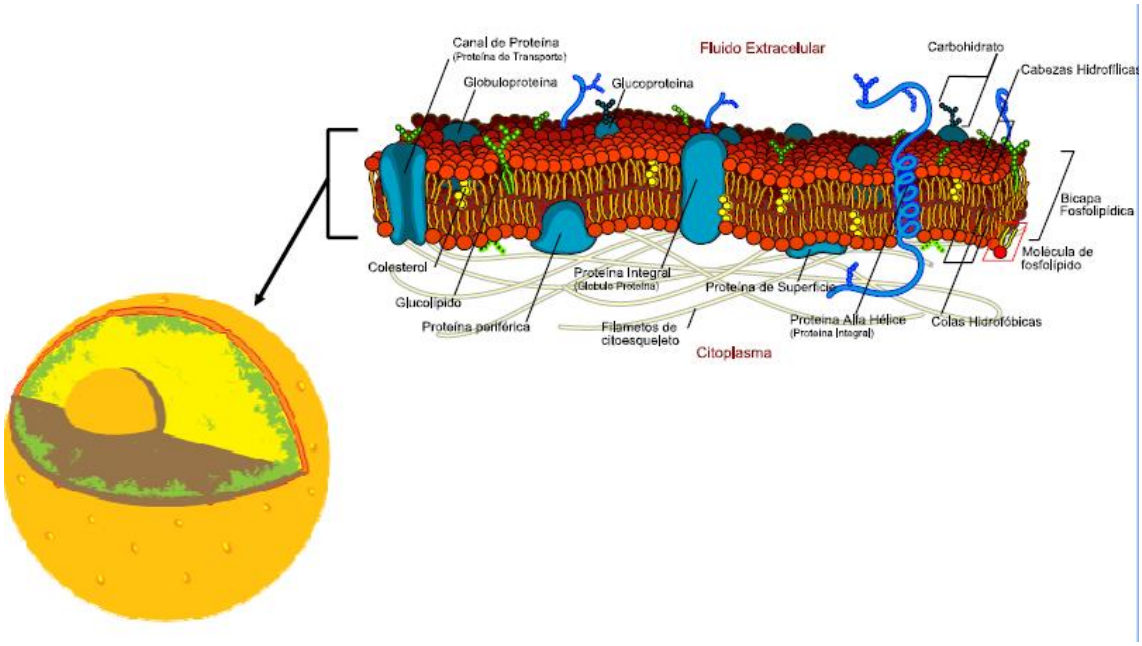


Fig. 2.1 - Célula

El establecimiento de la diferencia de potencial sobre la membrana celular se debe a una permeabilidad diferencial en relación con ciertos iones, por ejemplo, en estado de reposo, la membrana de la mayoría de las células no permite que los iones sodio (Na+) pasen a través de ellas, pero es hasta cierto punto permeable a los iones potasio (K+).

Estas propiedades, únicas de la membrana celular, hacen posible que haya diferencia en la composición entre el líquido intracelular y extracelular, y que se establezca sobre ella una diferencia de potencial, siendo la parte interna negativa con respecto a la externa. A este potencial se le llama potencial de reposo y se expresa con signo negativo. Su magnitud varía de un tejido a otro y es causado por un flujo de corriente iónica a través de la membrana. Se dice que una membrana está polarizada cuando se establece sobre ella una diferencia de potencial y seguirá así mientras permanezca impermeable a los iones que la polarizan. Cuando se estimula, pierde momentáneamente su impermeabilidad y los iones, que antes no podían franquearla, fluyen a través de ella, por lo tanto se despolariza. Para que la membrana se polarice de nuevo, la célula debe "bombear" iones positivos hacia el exterior, lo que requiere un gasto de energía.

Los intercambios entre la célula y el medio que la rodea son posibles porque la membrana es selectivamente permeable. El agua, con peso molecular de 18 y radio de 01 nm pasa casi libremente a través de ella y se difunde por ósmosis, pero la membrana es prácticamente impermeable a las proteínas intracelulares y otros aniones.

La membrana también contiene compuertas o válvulas que se abren o se cierran en respuesta a estímulos eléctricos o estímulos bioquímicos.

- **POTENCIAL DE REPOSO**

El potencial de reposo es la diferencia de potencial que existe entre el interior y el exterior de una célula. Se debe a que la membrana celular se comporta como una barrera semipermeable selectiva, es decir permite el tránsito a través de ella de determinadas moléculas e impide el de otras. En las células eléctricamente excitables, el potencial de reposo es aquel que se registra por la distribución asimétrica de los iones (principalmente sodio y potasio) cuando la célula está en reposo fisiológico, es decir, no está excitada. Este potencial es generalmente negativo.

- **POTENCIAL DE ACCIÓN**

Un potencial de acción o también llamado impulso eléctrico, es una onda de descarga eléctrica que viaja a lo largo de la membrana celular. Los potenciales de acción se utilizan en el cuerpo para llevar información entre unos tejidos y otros, lo que hace que sean una característica microscópica esencial para la vida. Pueden generarse por diversos tipos de células corporales, pero las más activas en su uso son las células del sistema nervioso, para enviar mensajes entre células nerviosas o desde células nerviosas a otros tejidos corporales, por ello los potenciales de acción son la vía fundamental de transmisión de códigos neurales. Sus propiedades pueden frenar el tamaño de cuerpos en desarrollo y permitir el control y coordinación centralizados de órganos y tejidos.

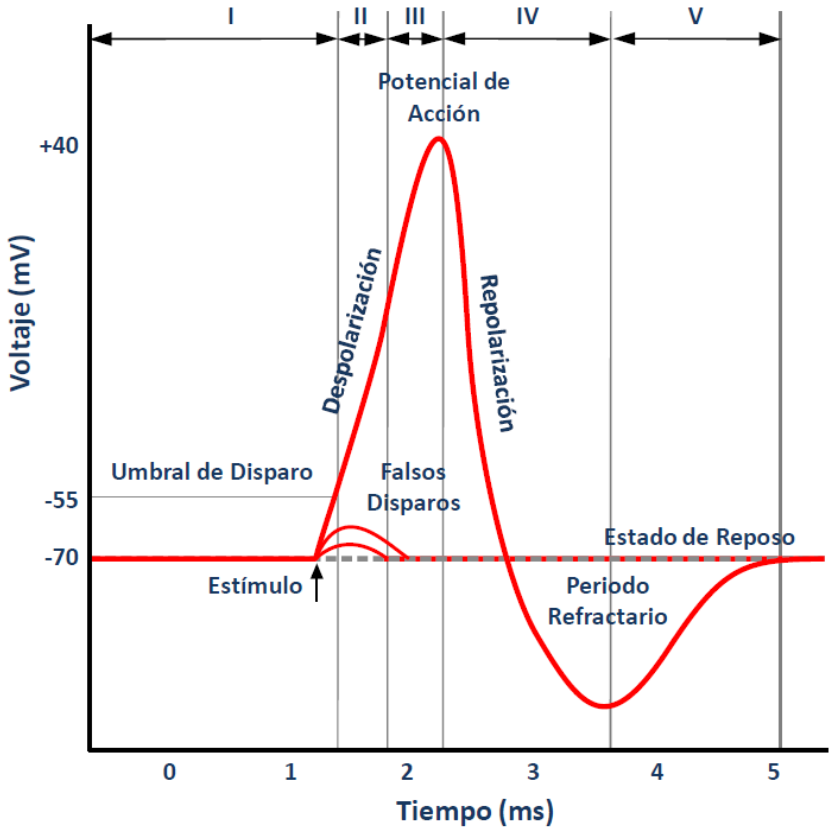


Fig. 2.2 - Potencial de acción

Partes de la estimulación de la menbrana celular:

I. Periodo de latencia (tiempo en que tarda en llegar el potencial al registro).
II. Fase de despolarización (se hace positivo el potencial).
III. Potencial de acción (cuando el potencial está invertido).
IV. Fase de repolarización (se hace negativo el potencial de nuevo).
V. Fase de hiperpolarización (se hace más negativo que el potencial en reposo).

El ciclo anterior puede describirse de la siguiente forma:

1. Reposo
2. Activo
3. Inactivo
4. Reposo.

2.2 - Amplificación de Biopotenciales

En general, las señales producidas por el cuerpo humano poseen una amplitud y una potencia de señal muy pequeñas, lo que hace necesario que se deba amplificar la señal deseada para poder ser visualizada de forma correcta.

Para la mayoría de las aplicaciones médicas, los parámetros más importantes de la amplificación son los siguientes:

1) **Ganancia.** Como ya se ha dicho anteriormente, las señales resultantes de la actividad electrofisiológica tienen unas amplitudes muy bajas; del orden de los mV o μV. Por tanto el voltaje de tales señales debe ser amplificada adecuadamente para el equipo de adquisición de datos, tal ganancia suele ser de, por lo menos, 1000. La forma de calcular esta ganancia es:

$$Gain(dB) = 20\log_{10}(ganancia\_lineal)$$

2) **Respuesta en frecuencia.** El ancho de banda de un amplificador de biopotenciales debe amplificar, sin atenuación, todo el espectro de frecuencia presente en la señal fisiológica de interés. El ancho de banda de cualquier amplificador es la diferencia entre la frecuencia de corte superior y la frecuencia de corte inferior.
3) **Rechazo en modo común.** El cuerpo humano es un buen conductor, por lo tanto, actúa como antena que recoge la radiación electromagnética presente en

el ambiente. La interferencia resultante en cada uno de los electrodos es tan potente que a menudo perturba la información de la señal electrofisiológica.

4) **Ruido y deriva.** Son señales adicionales no deseadas que contaminan la señal que se desea medir, que son producidas intrínsecamente por el propio circuito de medida. El ruido posee una frecuencia superior a 0.1 Hz, mientras que la deriva es inferior a 0,1Hz, por ello la deriva es descrita como cambios en la línea base de la señal. Ambas señales son análogas a introducir una señal de tensión de entrada diferencial.
5) **Recuperación de línea base.** Las altas tensiones de los electrodos producidas por corrientes de movimiento, estimulación, pulsos de desfibrilación, etc. producen interrupciones transitorias en el amplificador, debido a que el amplificador entra en saturación y necesita un tiempo para retornar a la línea base original. El tiempo transcurrido en esta acción es el tiempo de recuperación.
6) **Impedancia de entrada.** La impedancia de entrada de un amplificador debe ser lo suficientemente elevada como para no distorsionar la señal que se encuentra bajo medición. Puesto que la impedancia de salida de conjunto cuerpo-electrodo es muy elevada, se hace preciso contar con amplificadores de muy alta impedancia de entrada.
7) **Polarización de los electrodos.** Los electrodos se fabrican generalmente de metal, en contacto con un electrolito. El intercambio de iones entre ellos da lugar a la aparición de potenciales de electrodo que debe ser considerado en la ganancia del amplificador, para que éste no se sature y se produzcan errores en la medición.

2.2.1 - Adquisición del ECG

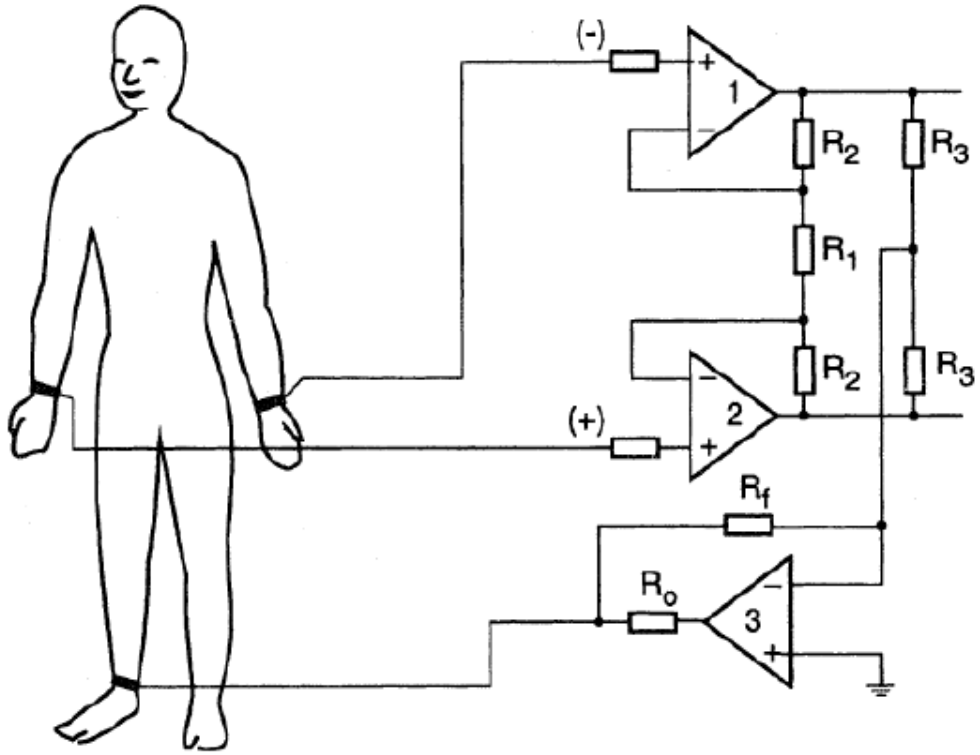


Fig. 2.3 - Montaje clásico para la adquisición de biopotenciales

El propósito de un Amplificador de Biopotenciales es amplificar las débiles señales biológicas hasta obtener un nivel apropiado para que puedan ser registradas.

Un amplificador de biopotenciales debe procesar altas tensiones de DC (corriente continua) de modo diferencial e importantes desbalances en las impedancias del generador de señal (cuerpo humano). Todo esto debe realizarse manteniendo buenas características; entre ellas, un alto rechazo de modo común (CMRR) y un bajo nivel de ruido. Estas condiciones son simples de cumplir asignando una ganancia importante a la etapa de entrada, pero esto conlleva la amplificación del ruido y la deriva propia de la señal. El problema fundamental de diseño de un amplificador de biopotenciales es disponer de una ganancia elevada en la primera etapa en presencia de componentes de DC.

La principal causa de interferencia electromagnética es la tensión de la red de distribución eléctrica. Los mecanismos por los cuales la misma ingresa al sistema de medida son conocidos, y día a día se desarrollan modelos cada vez más completos y precisos. Esto permite estimar los niveles de interferencia electromagnética para distintas situaciones, para así desarrollar técnicas para reducir sus efectos.

La tensión de la red interfiere según diversas formas, una de ellas es a través de la tensión de modo común. Esta tensión puede contaminar el registro debido a dos causas: el Rechazo de Modo Común CMRR finito del amplificador y a la transformación de modo común

a modo diferencial debida al desbalance entre las impedancias de los electrodos. El primer inconveniente no es muy restrictivo, pero el segundo exige, para el caso de mediciones con tres electrodos, elevadas impedancias de entrada de modo común. Una forma de reducir en parte esta perturbación es utilizar circuitos activos, que a partir de una realimentación negativa de la tensión de modo común, permiten reducirla sustancialmente.

La topología históricamente más utilizada como primer etapa es el bien conocido amplificador de instrumentación de tres amplificadores operacionales. En general, esta etapa está acoplada en forma directa y las componentes de DC son eliminadas en etapas posteriores. Esta primera etapa usualmente no genera ganancia suficiente, por lo tanto se requieren etapas adicionales para completar la amplificación requerida, con esta ganancia el CMRR es importante, siendo además posible conseguir bajos niveles de ruido. Este último dependerá casi exclusivamente de los amplificadores operacionales que componen la etapa de entrada.

En general, los amplificadores para biopotenciales hacen uso de tres electrodos: dos proveen la señal biológica en forma diferencial y el tercero, masa.

La señal atraviesa finalmente por un proceso de procesamiento y transmisión de la información, formado por: pre-amplificación, filtro pasa-alto, amplificación, filtro pasa-bajo, ajuste de voltaje DC, multiplexación, digitalización y transmisión.

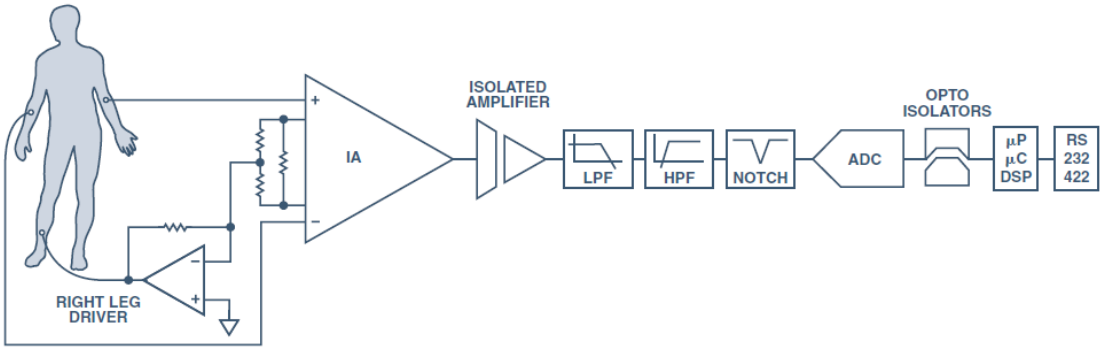


Fig. 2.4 - Electrocardiógrafo típico

- **DERIVACIONES**

Las disposiciones específicas de los electrodos, se conocen como derivaciones y en la práctica clínica se utilizan un número de 12 estándares, pudiéndose ampliar, clasificadas de la siguiente forma:

- Derivaciones Bipolares Estándar

Estas derivaciones (DI, DII, DIII) son las que originalmente eligió Einthoven para registrar los potenciales eléctricos en el plano frontal.

Los electrodos son aplicados en los brazos derecho e izquierdo y en la pierna izquierda, y se coloca un electrodo en la pierna derecha que sirve como polo a tierra, quedándose registradas las diferencias de potencial eléctrico entre estos electrodos.

- DI: Brazo izquierdo (+) Brazo derecho (-)
- DII: Pierna izquierda (+) Brazo derecho (-)
- DIII: Pierna izquierda (+) Brazo izquierdo (-)

- Derivaciones Amplificadas del Plano Frontal.

En los inicios de la electrocardiografía eran unipolares (VR, VL y VF), pero éstas fueron modificadas para amplificarlas en el registro, convirtiéndose en bipolares amplificadas (aVR, aVL yaVF).

En estas derivaciones se coloca el positivo en uno de los miembros y se compara con la sumatoria de los otros miembros conectados al polo negativo.

- aVR: Brazo derecho (+) y Brazo izquierdo + Pierna Izquierda (-)
- aVL: Brazo izquierdo (+) y Brazo derecho + Pierna Izquierda (-)
- aVF: Pierna izquierda (+) y Brazo derecho + Brazo Izquierdo (-)

- Derivaciones del plano horizontal

Son derivaciones mono o unipolares, pues comparan la actividad del punto en que se coloca el electrodo a nivel precordial con la suma de los tres miembros activos.

- V1: 4º espacio intercostal con linea paraesternal derecha.
- V2: 4º espacio intercostal con línea paraesternal izquierda.
- V3: Equidistante entre V2 y V4.
- V4: 5º espacio intercostal con linea medioclavicular izquierda.
- V5: 5º espacio intercostal con línea axilar anterior izquierda.
- V6: 5º espacio intercostal con línea axilar media izquierda.

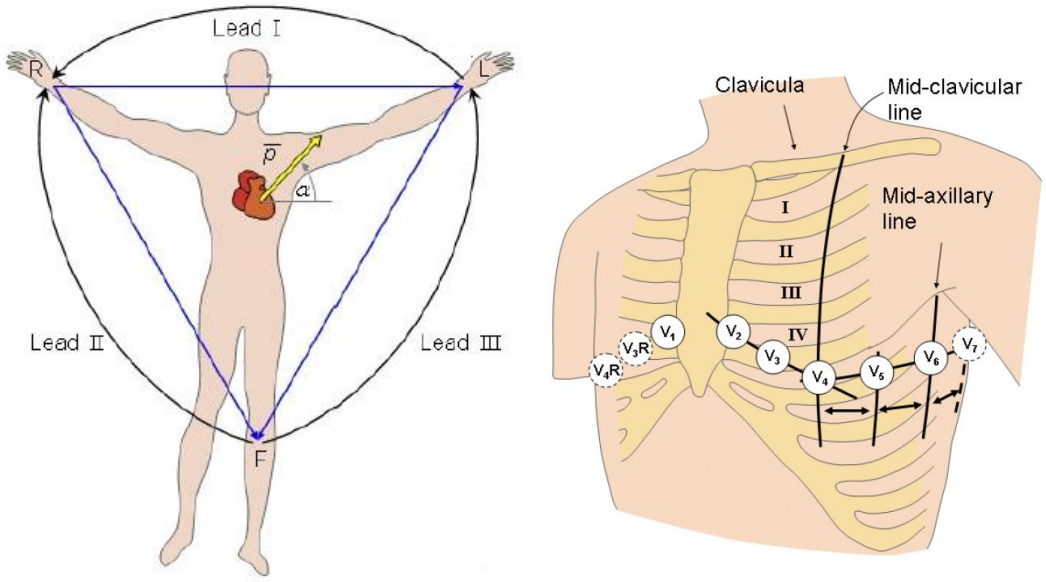


Fig. 2.5 - Posición de los electrodos

2.2.2 - Adquisición del EEG

Para proceder a registrar el EEG se parte de una serie de electrodos situados sobre la superficie del cuero cabelludo en situaciones precisas, determinadas según el sistema internacional 10-20 (leído diez-veinte). Cada electrodo es un punto de registro, sin embargo, para poder realizar este registro es preciso disponer de dos terminales.

Por esto habrá que seleccionar cuáles de los electrodos deben ser la fuente de señal registrada en el electroencefalógrafo, dependiendo del número de canales disponibles y del propósito específico del registro a realizar. En este aspecto, la primera decisión que se deberá tomar será el seleccionar entre registros monopolares o bipolares.

En los registros monopolares (fig. 2.6) se obtiene la señal de cada uno de los electrodos de forma independiente. Esto se ejecuta mediante el uso de dos electrodos, el electrodo de registro (activo) y el electrodo de referencia.

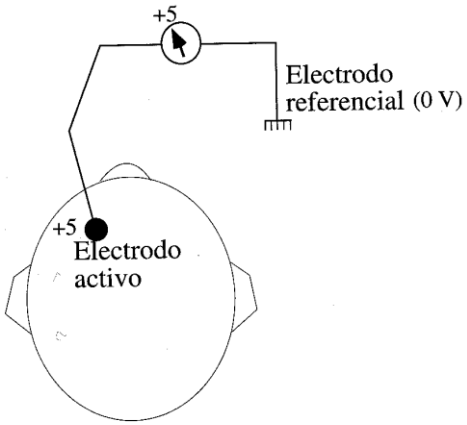


Fig. 2.6 - Esquema del montaje de registros unipolares

En los registros bipolares se utilizan parejas de electrodos, y se registran las diferencias de tensión entre cada par de puntos (fig. 2.7). Los dos electrodos de cada pareja son activos.

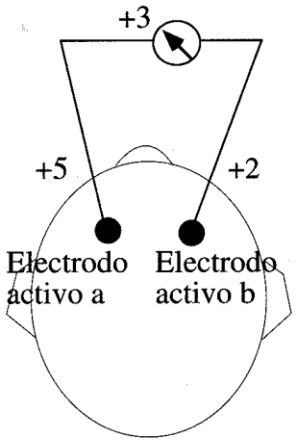


Fig. 2.7 - Esquema del montaje de registros bipolares

De acuerdo con lo anterior es posible realizar un número enorme de registros bipolares diferentes, tantos como parejas diferentes de electrodos puedan formarse, pero no todas generan información útil.

A causa de esto, los montajes han sido clasificados por la Federación Internacional de EEG y Neurofisiologia en Longitudinales y Transversales.

En los Montajes Longitudinales se registra la actividad de pares de electrodos dispuestos en sentido anteroposterior de cada mitad del cráneo. En los Montajes Transversales se realizan registros de pares de electrodos dispuestos transversalmente según los planos sagitales anterior, medio o posterior (fig. 2.8).

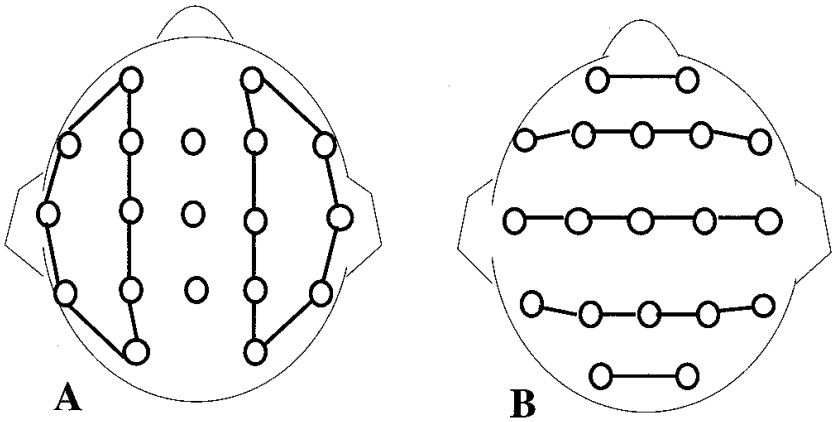


Fig. 2.8 - A. Longitudinal bipolar - B. Transversal bipolar

2.3 - La señal de ECG

Un electrocardiograma (ECG) es una prueba que registra la actividad eléctrica del corazón, que se utiliza para medir el ritmo y la regularidad de los latidos, así como el tamaño y posición de las cámaras cardíacas, cualquier daño al corazón y los efectos de drogas o instrumentos utilizados para regularlo (como un marcapasos). Es un gráfico que se obtiene a partir del electrocardiógrafo en forma de cinta gráfica continua.

El corazón tiene un sistema de conducción compuesto por fibras de músculo cardiaco especializadas en la transmisión de impulsos eléctricos. Aunque el corazón tiene inervación por parte del sistema simpático, late aun sin estímulo de este, ya que el sistema de conducción es autoexcitable, es por esto que no tenemos control sobre los latidos de nuestro corazón.

El sistema de conducción debe transmitir el impulso eléctrico de las aurículas a los ventrículos, y se compone de los siguientes elementos, el nodo sinoauricular (sinusal), el nodo auriculoventricular y haz de His, con su rama derecha y su rama izquierda.

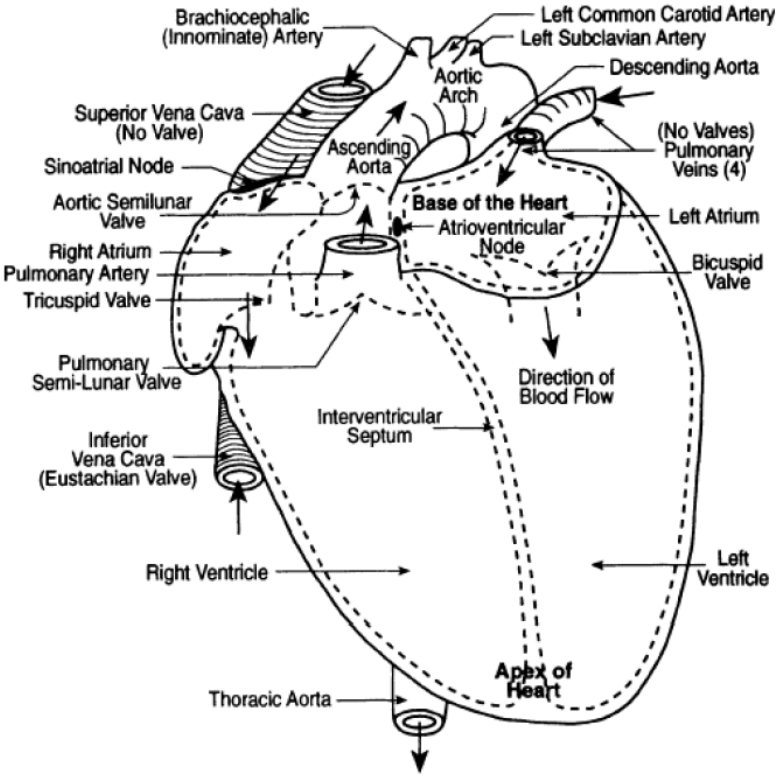


Fig. 2.9 - Corazón humano

El estímulo eléctrico en el corazón nace en el nodo sino-auricular (NSA) o marca-paso, que es una pequeña formación de tejido muscular especializado, localizado en la aurícula derecha en el sitio de unión entre la vena cava superior y el atrium.

La acción de este estímulo da origen a una corriente eléctrica, llamada corriente de despolarización (onda P). Cuando esta onda de excitación llega a la zona de unión de la aurícula y el ventrículo derecho debe pasar a través de otro nodo, llamado áurico-ventricular (AV), donde la velocidad de propagación es significativamente menor, dando lugar a un retraso en la propagación de dicho estímulo.

En este nódulo AV comienza el llamado haz de His, que presenta una mayor conductividad al paso de la corriente que el tejido muscular circundante.

Durante la conducción hasta las últimas células cardíacas se obtiene una onda llamada QRS, que representa la despolarización ventricular. Los ventrículos, tras un momento de reposo, durante el cual permanecen en un estado de despolarización (ST), comienzan a repolarizarse, siguiendo el mismo camino que la onda de despolarización, dando lugar a la onda llamada T. Acto seguido a la onda T, se produce una última onda U, cuyo origen es desconocido.

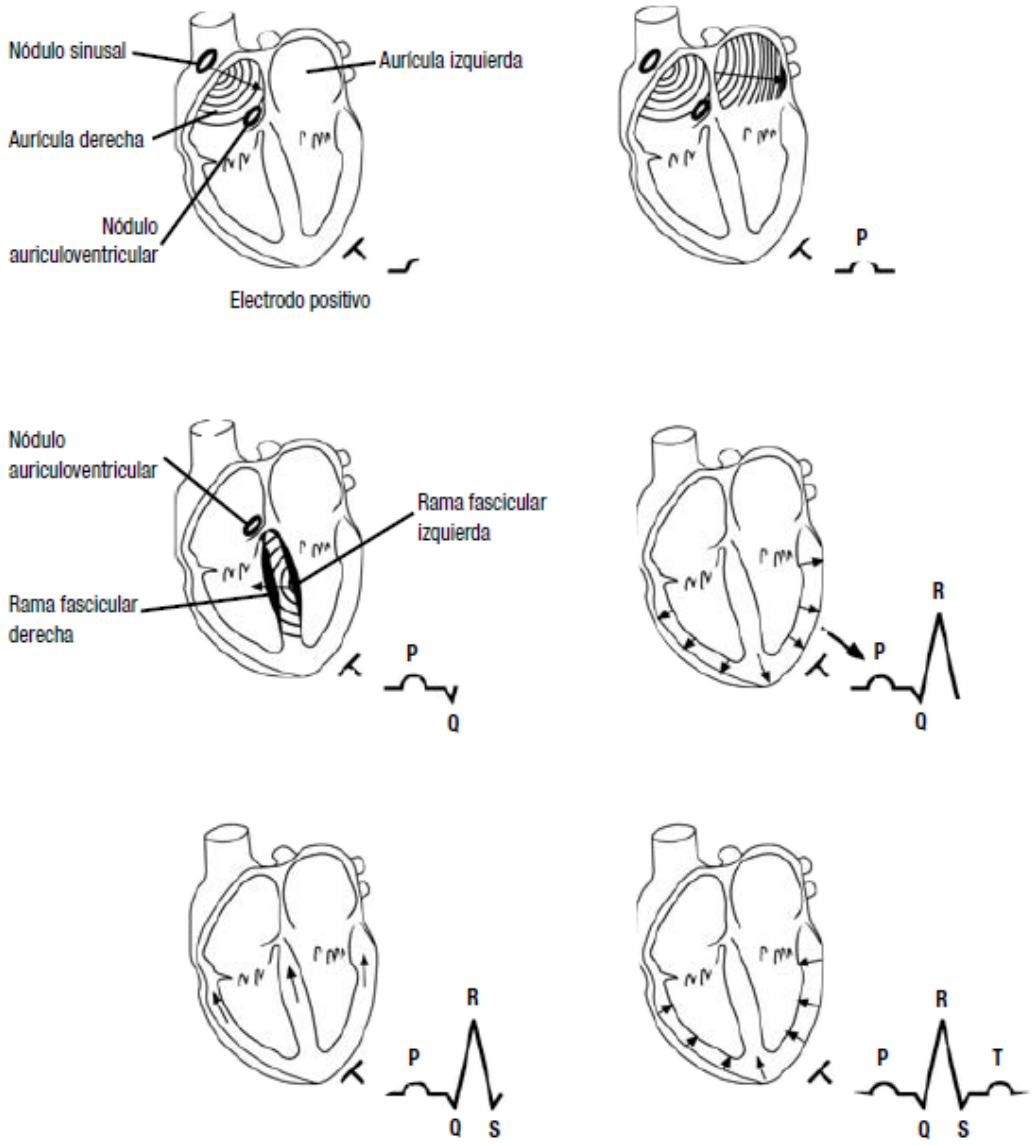


Fig. 2.10 - Actividad eléctrica del corazón

2.3.1 - Reseña histórica

En 1872, Alexander Muirhead, durante sus estudios de postgrado en el Hospital de San Bartolome de Londres, conectó alambres a la muñeca de un paciente febril con el fin de obtener un registro de los latidos del corazón. Esta actividad se registró directamente para ser visualizado por un electrómetro de Lippmann ideado por el fisiólogo británico John Burdon Sanderson.

En el siglo XIX se hizo evidente que el corazón generaba electricidad. La actividad bioeléctrica correspondiente al latido cardiaco fue descubierta por Kolliker y Mueller en 1856.

El primero en aproximarse sistemáticamente a este órgano bajo el punto de vista eléctrico fue Augustus Waller, que trabajaba en el hospital St. Mary, en Paddington (Londres).

Aunque en 1911 aún veía pocas aplicaciones clínicas a su trabajo, el logro llegó cuando Willem Einthoven, que trabajaba en Leiden (Países Bajos), descubrió el galvanómetro de cuerda, mucho más exacto que el galvanómetro capilar que usaba Waller. Einthoven asignó las letras P, Q, R, S y T a las diferentes deflexiones y describió las características electrocardiográficas de gran número de enfermedades cardiovasculares. Le fue otorgado el Premio Nobel de Fisiología o Medicina en 1924 por su descubrimiento.

Por otro lado la compañía Cambridge Scientific Instruments, ubicada en Londres, fabricó por primera vez la máquina de Einthoven en 1911, y en 1922 se unió con una compañía en Nueva York para formar Cambridge Instruments Company, Inc. Desde entonces, ambas compañías se han beneficiado con el intercambio mutuo de tecnología. Poco tiempo después el electrocardiógrafo demostró su valor en el diagnóstico médico y hoy se mantiene como uno de los instrumentos electrónicos más empleados en la medicina moderna.

El electrocardiógrafo ha evolucionado desde el enorme aparato original hasta el sistema electrónico compacto actual, que a menudo incluye una interpretación computarizada de electrocardiograma.

2.4 - La señal de EEG

El EEG es el encargado de leer el sistema nervioso y a partir de él, poder realizar el análisis pertinente. El sistema nervioso es un conjunto de tejidos dentro de nuestro cuerpo, encargados de captar y procesar rápidamente las señales internas y externas, tomando el control y coordinación sobre los demás órganos, para así, lograr una oportuna y eficaz interacción con el medio ambiente cambiante. La unidad básica del sistema nervioso es la neurona, la cual tiene la capacidad de comunicarse eléctricamente con otras células, sean éstas nerviosas o no. La información viaja entre neuronas por medio de impulsos eléctricos que se transmiten de unas neuronas a otras. Estos impulsos, se reciben de otras neuronas en las dendritas y pasan a través de la neurona hasta ser conducidas por el axón a los terminales de salida, los cuales pueden conectarse con otra neurona, fibras musculares o glándulas.

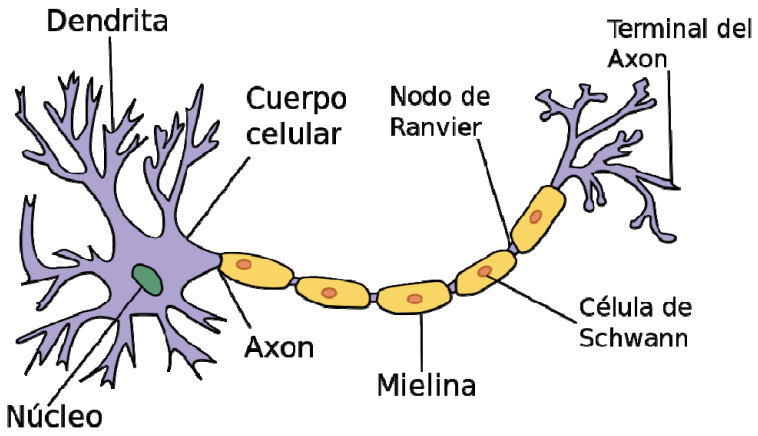


Fig. 2.11 - Estructura de una neurona

Cuando se produce un estímulo externo, el sistema nervioso actúa de la siguiente manera. El estímulo es recibido en alguna región sensorial capturando alguna información, la cual es transportada por el sistema nervioso (a través de las neuronas) hasta una componente integradora en donde se analiza. Esta componente elabora la respuesta, que es conducida a través de las neuronas hacia fibras musculares (respuesta motora) o hacia glándulas (secreción glandular). Hay que tener en cuenta que la actividad cerebral es producida por un número muy elevado de neuronas (aproximadamente cien mil millones en un cuerpo humano medio) y cada una de las tareas que nuestro cuerpo puede realizar provoca una actividad cerebral con forma e intensidad diferentes, además de localizarse en distintas zonas del sistema nervioso.

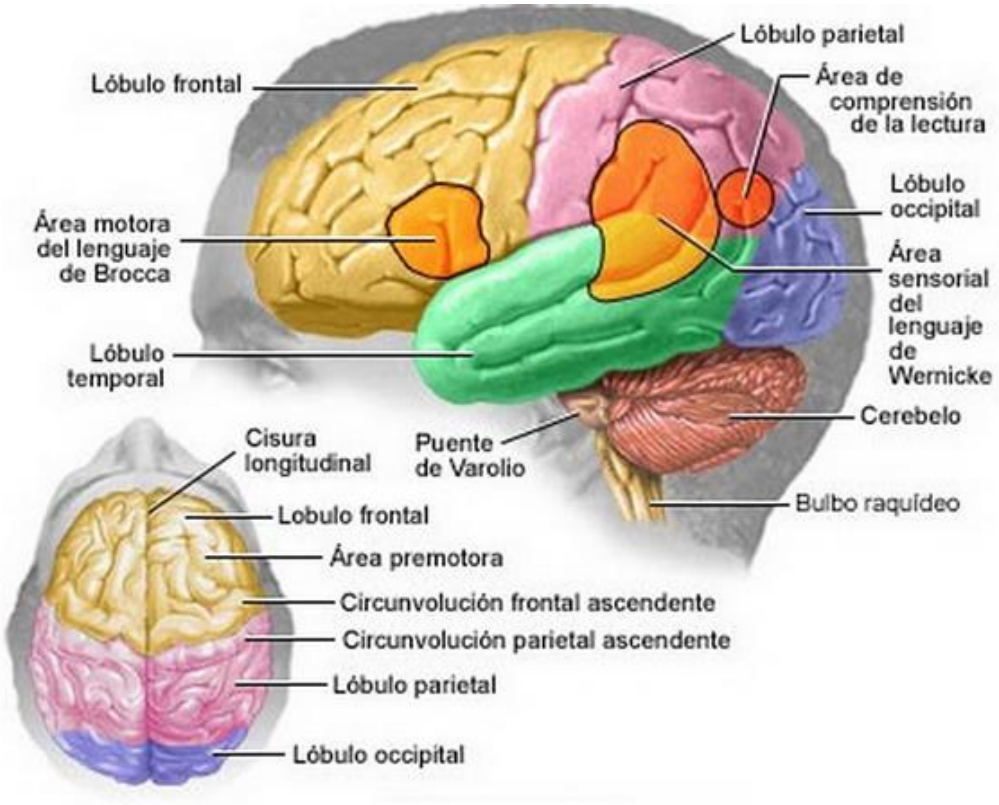


Fig. 2.12 - Cerebro humano

Aunque siempre predominen unas ondas frente a otras, todas están presentes durante el desarrollo de una actividad con diferente intensidad, según sea el tipo de tarea que se desarrolla.

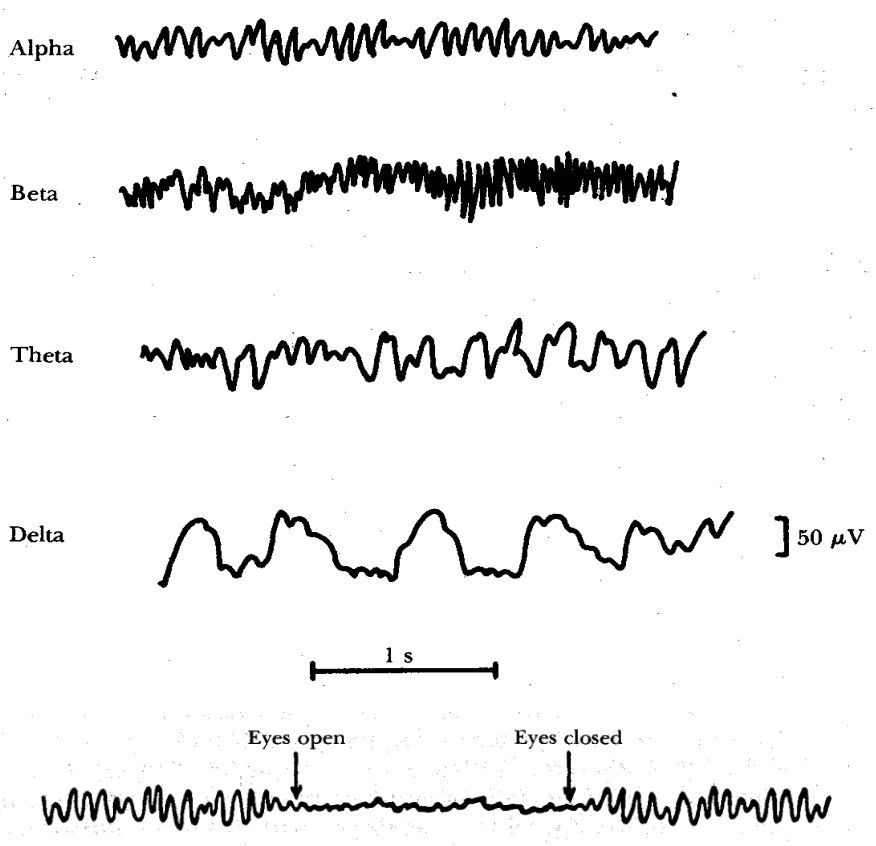


Fig. 2.13 - Ritmos normales en electroencefalografía

Ritmos	Situación mental
Alpha	Se registran especialmente momentos antes de dormirse. En adultos se observa al cerrar los ojos o en estados de relajación y poca actividad mental. La amplitud disminuye durante la atención, especialmente visual, y durante el esfuerzo mental.
Beta	Se registran cuando los sentidos se hallan volcados hacia el exterior y en plena actividad mental
Theta	Se producen durante el sueño o en meditación profunda
Delta	Surgen principalmente en el sueño profundo y muy raras veces se manifiesta estándo despierto

Mu	De frecuencia y amplitud similar al ritmo alfa. Está presente en estados de reposo o concentración y su amplitud disminuye durante la realización de movimientos o su imaginación motora.

2.4.1 - Reseña histórica

En 1870, Fritsch y Hitzig, médicos militares del ejército prusiano, observaron que al estimular, mediante corriente galvánica, determinadas áreas laterales del cerebro se producían movimientos en el lado opuesto del cuerpo. Cinco años más tarde R. Caton confirmó que el cerebro es capaz de producir corrientes.

Ferrier, siguiendo en la misma línea, experimentó con la «corriente farádica». Como resultado de todo ello, hacia finales de siglo, se tenían suficientes pruebas de que el cerebro de los animales poseía propiedades eléctricas, comparables a las encontradas en el nervio y en el músculo.

En 1913, Prawdwicz- Neminski registró lo que llamó «electrocerebrograma», de un perro, siendo el primero en intentar clasificar semejantes observaciones. Hay que puntualizar, sin embargo, que todos los experimentos se hacían sobre cerebros al descubierto. Al ser los cambios eléctricos tan pequeños y sin procedimientos de amplificación, era imposible registrar los impulsos que alcanzaran el exterior del cráneo aún de haberse sospechado de su existencia.

En 1928, Hans Berger ideó un método que prometía una investigación de la actividad eléctrica cerebral, descubriendo lo que se conoció como «ritmo de Berger». Sin embargo debido a su falta de conocimientos técnicos, no fue hasta algunos años después cuando se reconoció su importancia. Mientras tanto, las posibilidades de la electroencefalografía clínica se discutían, por primera vez, en una reunión en el Laboratorio central de Patología del Hospital Maudsley de Londres, en 1929. A pesar de que el grupo de investigadores intentara obtener registros del «ritmo de Berger» usando amplificadores y un galvanómetro vetusto, no se tomaba en serio el estudio del cerebro ni los descubrimientos de Berger.

En 1934, en una demostración pública ante un auditorio británico en una reunión de la Sociedad de Fisiología, en Cambridge, Adrian y Matthews verificaron por primera vez el «ritmo de Berger». Berger, utilizando las mejoras introducidas por Adrian, observó, por ejemplo, que cuando el sujeto abría los ojos o resolvía algún problema mentalmente, se alteraba el ritmo amplio y regular. Esto fue verificado posteriormente por Adrian y Matthews, quien al tener mejores conocimientos científicos y mejores técnicas, desarrollaron de forma notable lo

demostrado por Berger, demostrando que el ritmo regular y amplio de diez ciclos por segundo urgía de las áreas visuales de asociación y no de todo el cerebro. Años más tarde se apreció la magnitud de tal descubrimiento.

Posteriormente la electropatología del cerebro creció en importancia, confirmándose las predicciones de Golla sobre las alteraciones de las oscilaciones rítmicas de las enfermedades. Se avanzó mucho en este campo, comenzando a interesar entre los investigadores del EEG, el estudio de la epilepsia y otras enfermedades mentales, remarcándose la complejidad del objeto de estudio, que imposibilita el aislamiento de funciones simples, siendo necesario estudiar al cerebro como un órgano total.

A partir de estos comienzos, con el paso de los años y mediante evoluciones sucesivas, se han llegado a conocer otros aspectos del EEG, tal como lo conocemos hoy en día.

2.5 - Simuladores de señales fisiológicas

El empleo de generadores o simuladores de biopotenciales se lleva realizando desde que el uso de la electrónica se ha introducido en el ámbito de la biomedicina. El uso de una señal artificial, simulando una señal real de ECG, o cualquier otro tipo de bioseñales, es necesario para el desarrollo y servicio de los equipos encargados de ellos. Esto hace innecesario realizar mediciones directamente en el ser humano, y particularmente con las actividades de investigación y reparación, elimina un potencial riesgo para el sujeto de prueba.

2.5.1 - Simulador M.A.S.H. (1976)

El uso de estos 'calibradores' no es reciente, puesto que, durante la década de los años 70, ya se llevaban a cabo este tipo de procedimientos, como se puede apreciar en las figuras 2.14 y 2.15. Como se puede observar en las mismas, este simulador hace uso de componentes analógicos; transistores, amplificadores operacionales y circuitos temporizadores de propósito general (555).

Este circuito es capaz de simular las señales propias de un electrocardiograma, a partir de un oscilador de referencia (configurado para trabajar en torno a los 65 b.p.m). Los pulsos generados son entonces conformados (con recortadores e integradores) y sumados con distintas configuraciones operacionales.

A efectos de este proyecto, se hace especialmente interesante la configuración de la etapa de salida en divisor de tensión, diseñada para acoplarse de forma directa a amplificadores de biopotenciales de tres electrodos.

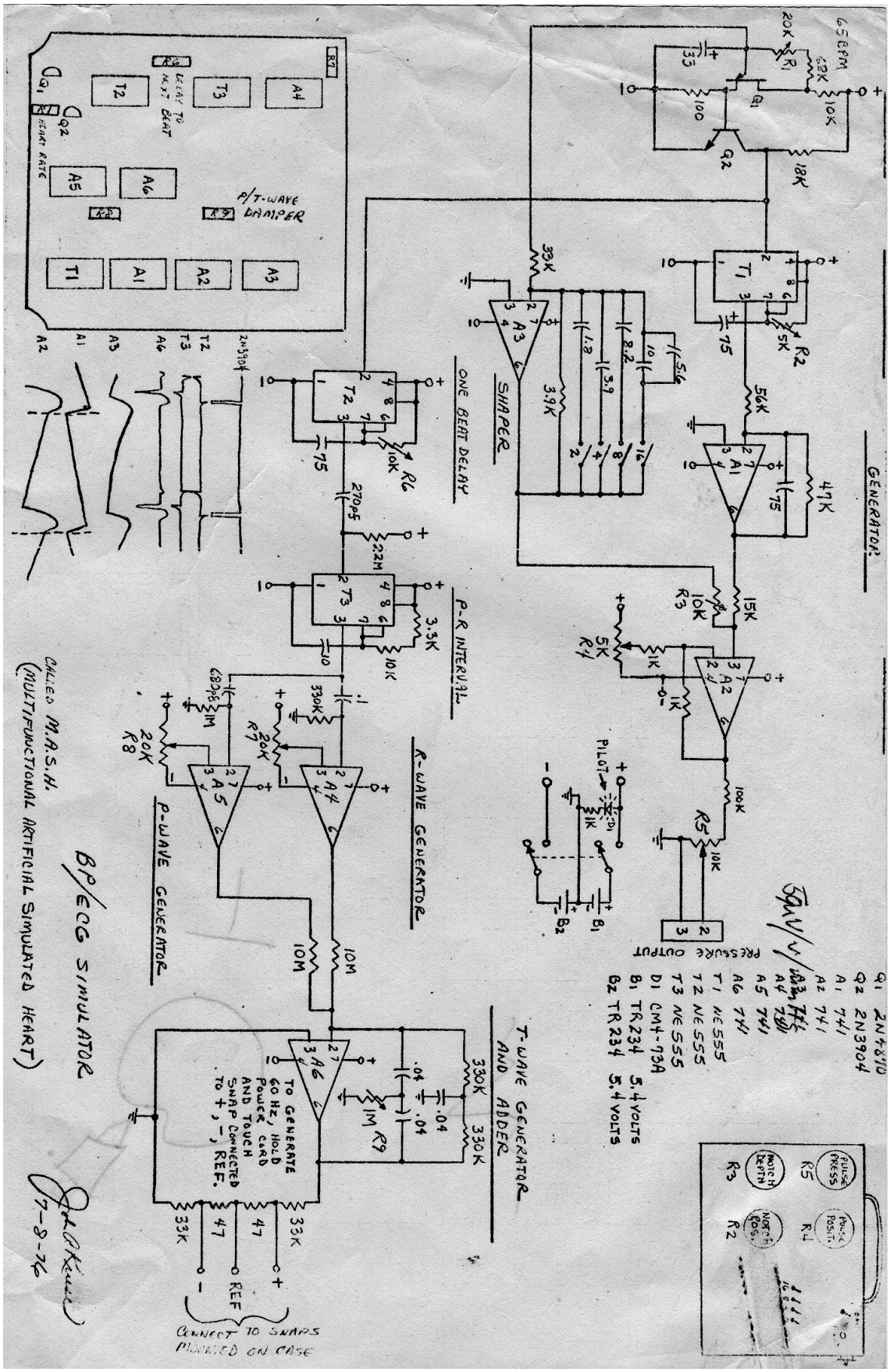


Fig. 2.14 - Esquemático del simulador M.A.S.H

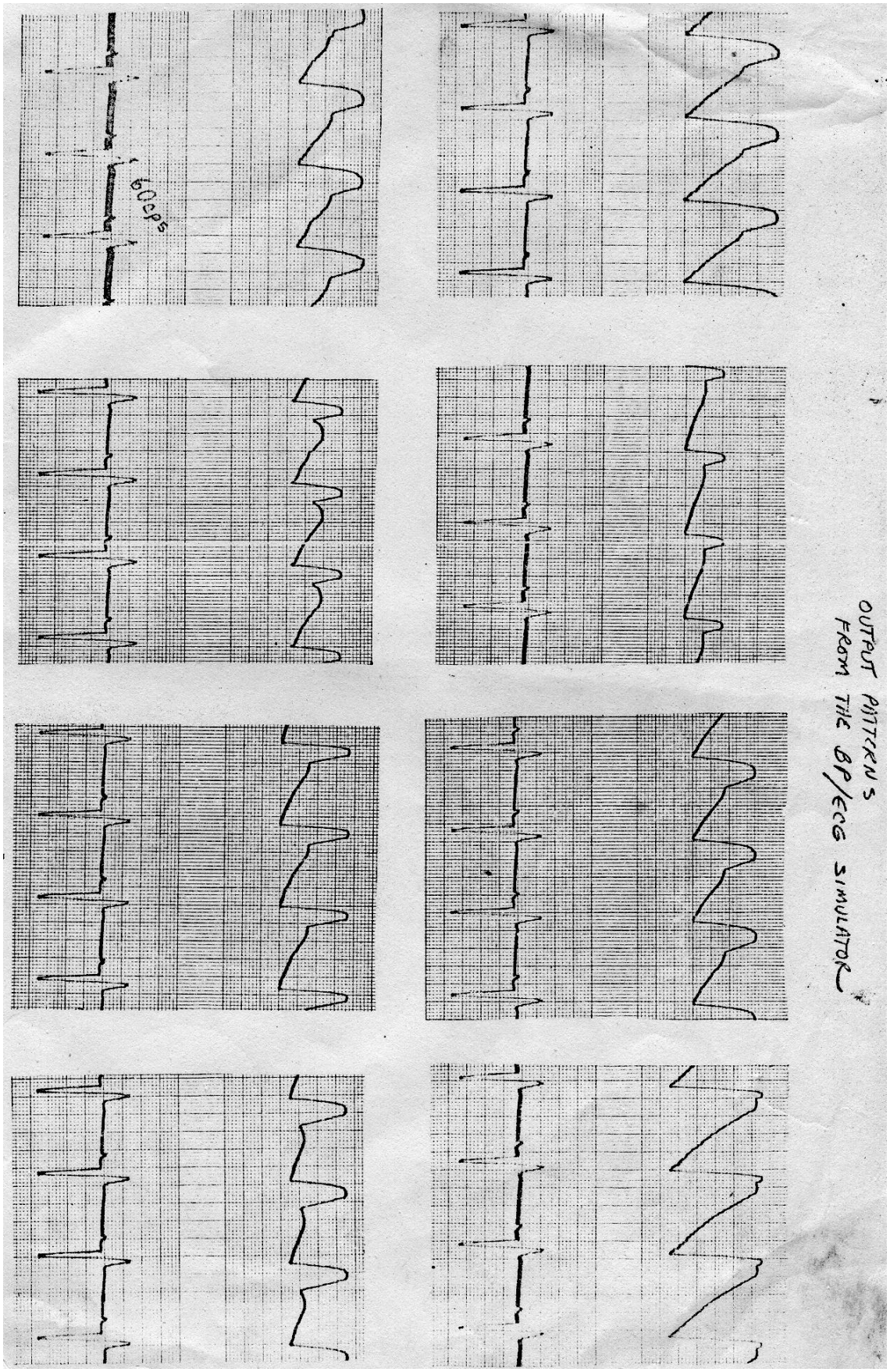


Fig. 2.15 - Patrones de salida

2.5.2 - Simulador y amplificador ECG (2002)

Este circuito simulador (ver figura 2.16) se estructura en torno a un circuito temporizador de uso general (tipo 555) en configuración astable, responsable de la generación de pulsos para la simulación de los complejos QRS. De forma adicional, incorpora una fuente acoplada en alterna para la generación de interferencias de red (en este caso, a 60Hz).

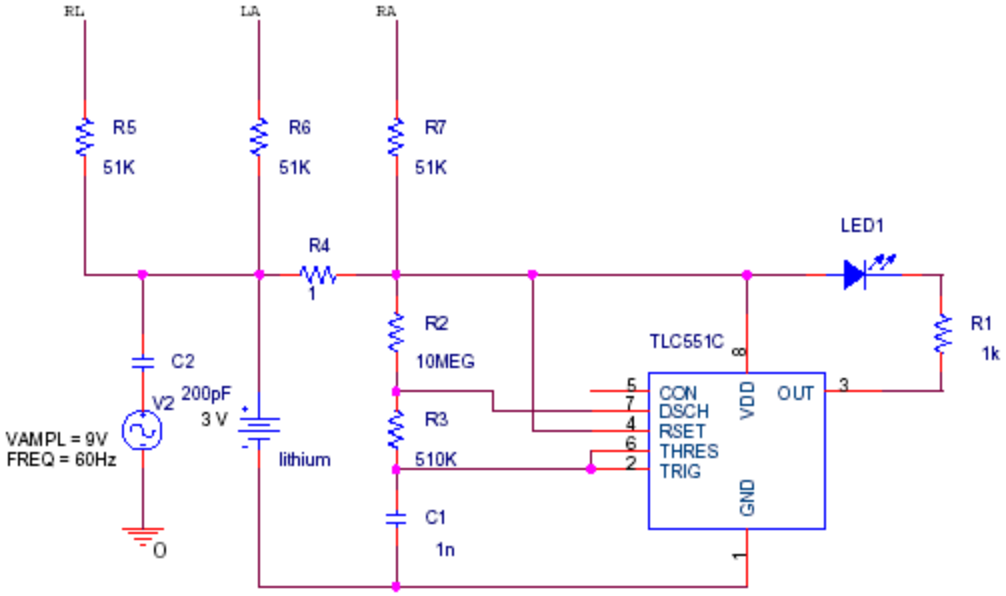


Fig. 2.16 - Esquemático de simulador analógico de ECG

En la siguiente figura (2.17) se muestra un circuito amplificador capaz de tratar (pre-amplificación, amplificación, filtro paso alto, filtro paso bajo y circuito atenuador) las señales generadas por este simulador.

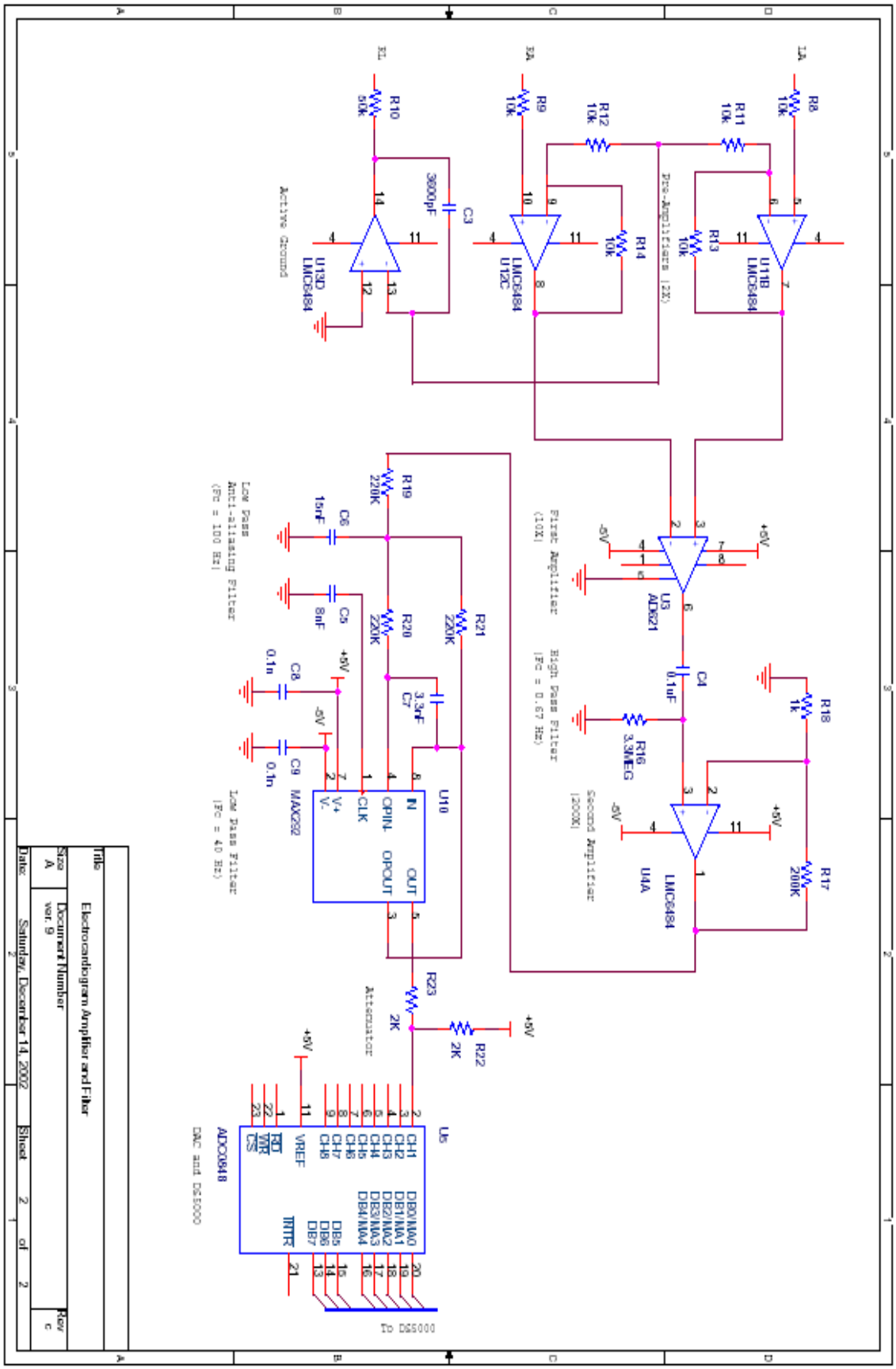


Fig. 2.17 - Filtro y amplificador del simulador de ECG

2.5.3 - Simulador EEG Modina Bioengineering, por S. Jung (2008)

Este circuito simulador es capaz de generar ondas de EEGs, basado en el uso de una señal cuadrada acoplada a una señal sinusoidal de frecuencia superior. Para la configuración de las diferentes señales que se manejan en este simulador se dispone de 3 modos de funcionamiento, en las cuales se modifican individualmente cada una de estas señales:

- **SPOT**. Generación de una onda sinusoidal de frecuencia permanente (imagen 2.19).
- **SWEEP**. Barrido de la señal sinusoidal mediante el uso de diferentes métodos (imagen 2.20).
- **PULSE**. Generación de un pulso cuadrado (imagen 2.21).

Para seleccionar el funcionamiento requerido, este dispositivo dispone de 8 interruptores (imagen 2.18) para modificar, por ejemplo, la frecuencia, duración del pulso, etc.

Nuevamente, incorpora una etapa de salida en divisor de tensión para la adaptación a bioamplificadores de 3 electrodos.

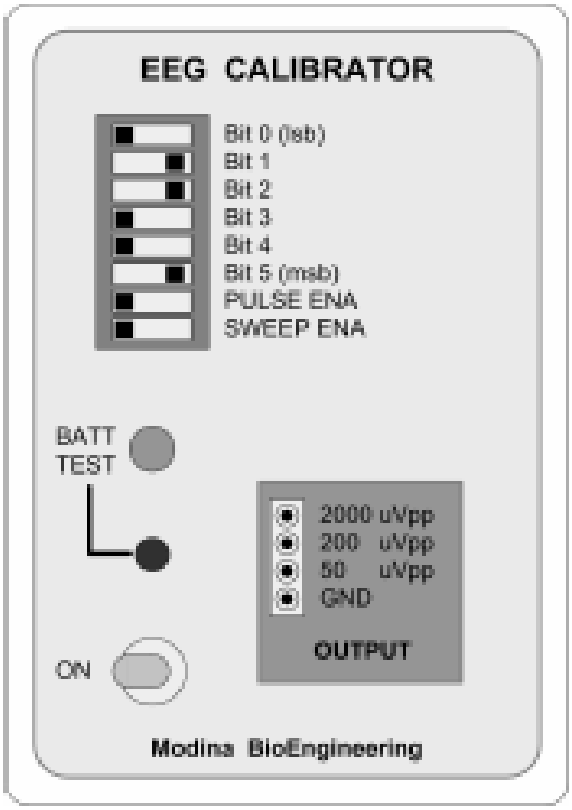


Fig. 2.18 - Panel frontal

	Bit 0(lsb)	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5(msb)	PULSE ENA	SWEEP ENA
SW	1	2	3	4	5	6	7	8
SPOT SINEWAVE: ON	-	F	R	E	Q	-	■	■
SPOT SINEWAVE: OFF								
1 Hz	-	0	0	0	0	0	0	0
2 Hz	-	1	0	0	0	0	0	0
3 Hz	-	0	1	0	0	0	0	0
4 Hz	-	1	1	0	0	0	0	0
▼	-	▼	▼	▼	▼	▼	▼	▼
64 Hz	-	1	1	1	1	1	0	0

0 = DIP switch OFF
1 = DIP switch ON

Fig. 2.19 - Configuración de la generación de la señal sinusoidal

SWEEP SINEWAVE:	SW	1 Bit 0 (lsb)	2 Bit 1	3 Bit 2	4 Bit 3	5 Bit 4	6 Bit 5 (msb)	7 PULSE ENA	8 SWEEP ENA
	ON / OFF	S1	S2	R1	R2	R3	R4		
reserved	-	0	0	X	X	X	X	0	1
Decreasing freq. saw-tooth	-	0	1	X	X	X	X	0	1
Increasing freq. saw-tooth	-	1	0	X	X	X	X	0	1
Rise & fall, triangular	-	1	1	X	X	X	X	0	1
Sweep rate 1 (slow)	-	X	X	0	0	0	0	0	1
Sweep rate 2	-	X	X	1	0	0	0	0	1
Sweep rate 3	-	X	X	0	1	0	0	0	1
Sweep rate 4	-	X	X	1	1	0	0	0	1
Sweep rate 5	-	X	X	0	0	1	0	0	1
Sweep rate 6	-	X	X	1	0	1	0	0	1
Sweep rate 7	-	X	X	0	1	1	0	0	1
Sweep rate 8	-	X	X	1	1	1	0	0	1
Sweep rate 9	-	X	X	0	0	0	1	0	1
Sweep rate 10	-	X	X	1	0	0	1	0	1
Sweep rate 11	-	X	X	0	1	0	1	0	1
Sweep rate 12	-	X	X	1	1	0	1	0	1
Sweep rate 13	-	X	X	0	0	1	1	0	1
Sweep rate 14	-	X	X	1	0	1	1	0	1
Sweep rate 15	-	X	X	0	1	1	1	0	1
Sweep rate 16 (fast)	-	X	X	1	1	1	1	0	1

X = Don't care
0 = DIP switch OFF
1 = DIP switch ON

Fig. 2.20 - Configuración del barrido de la señal

PULSE GENERATION:	SW	1 Bit 0 (lsb)	2 Bit 1	3 Bit 2	4 Bit 3	5 Bit 4	6 Bit 5 (msb)	7 PULSE ENA	8 SWEEP ENA
	ON / OFF	W1	W2	W3	R1	R2	R3		
Pulse width: 16 mS	-	0	0	0	X	X	X	1	0
Pulse width: 32 mS	-	1	0	0	X	X	X	1	0
Pulse width: 48 mS	-	0	1	0	X	X	X	1	0
Pulse width: 64 mS	-	1	1	0	X	X	X	1	0
Pulse width: 80 mS	-	0	0	1	X	X	X	1	0
Pulse width: 96 mS	-	1	0	1	X	X	X	1	0
Pulse width: 112 mS	-	0	1	1	X	X	X	1	0
Pulse width: 128 mS	-	1	1	1	X	X	X	1	0
Pulse repetition: 0.5 sec	-	X	X	X	0	0	0	1	0
Pulse repetition: 1.0 sec	-	X	X	X	1	0	0	1	0
Pulse repetition: 1.5 sec	-	X	X	X	0	1	0	1	0
Pulse repetition: 2.0 sec	-	X	X	X	1	1	0	1	0
Pulse repetition: 2.5 sec	-	X	X	X	0	0	1	1	0
Pulse repetition: 3.0 sec	-	X	X	X	1	0	1	1	0
Pulse repetition: 3.5 sec	-	X	X	X	0	1	1	1	0
Pulse repetition: 4.0 sec	-	X	X	X	1	1	1	1	0

X = Don't care
0 = DIP switch OFF
1 = DIP switch ON

Fig. 2.21 - Configuración de la generación de pulsos

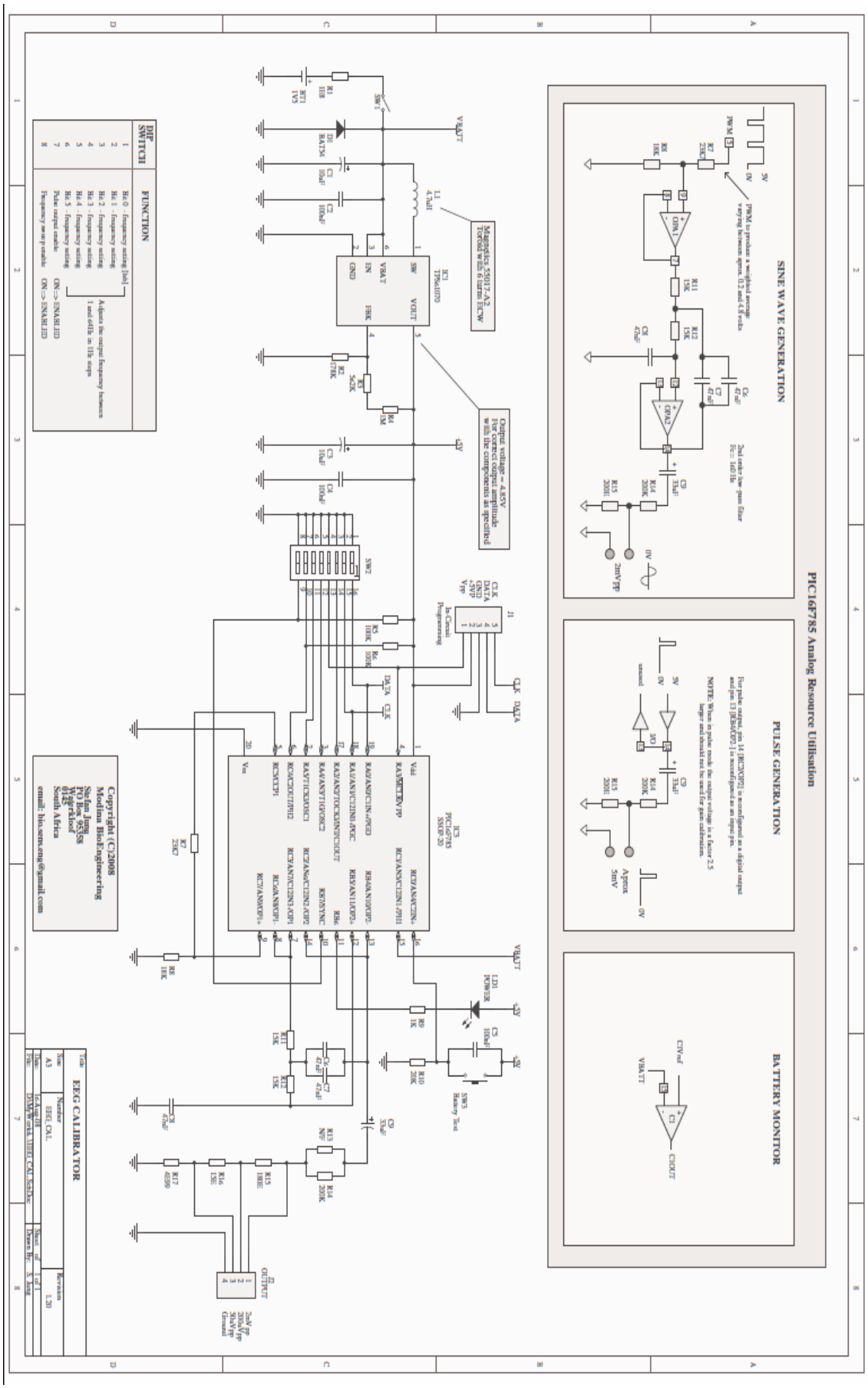


Fig. 2.22 - Esquemático de un calibrador de EEGs de Modina Bioengineering

2.5.4 - Simulador ECG Adafruit Menta de James Lynch (2013)

Este simulador, el más reciente de los contemplados en esta revisión, se configura en torno a una placa de desarrollo Arduino. Diseñado para simular señales de ECG entre 30 y 150 b.p.m.; hace uso de una señal almacenada en una tabla de memoria y un convertidor DAC externo, encargado de la transformación de las magnitudes digitales guardadas en memoria a magnitudes analógicas. De nuevo, incorpora una etapa de salida en divisor de tensión para la adaptación a bioamplificadores de 3 electrodos.

Hace uso de un potenciómetro regulador de la frecuencia cardiaca, así como una pantalla LCD donde se muestra dicha frecuencia, como puede observarse en la imagen 2.23.

Fig. 2.23 - Parte frontal del simulador ECG

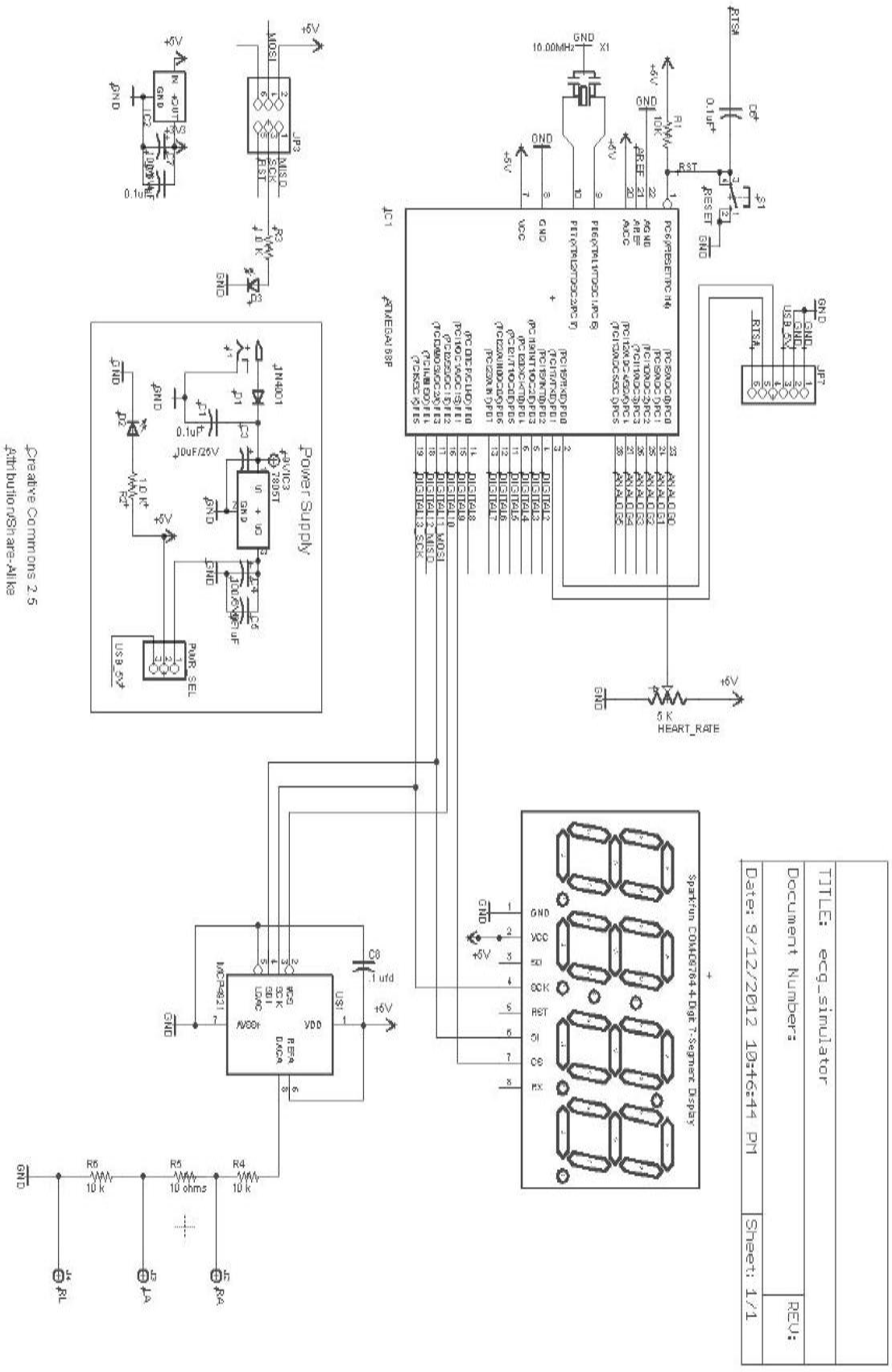


Fig. 2.24 - Esquemático del simulador digital de ECG de Adafruit Menta

Capítulo 3

Desarrollo software

3.1 - Descripción general

En el siguiente diagrama de bloques se muestra la organización de los componentes que conforman el proyecto, mostrando las relaciones de cada uno de los diferentes dispositivos que se emplean en este documento.

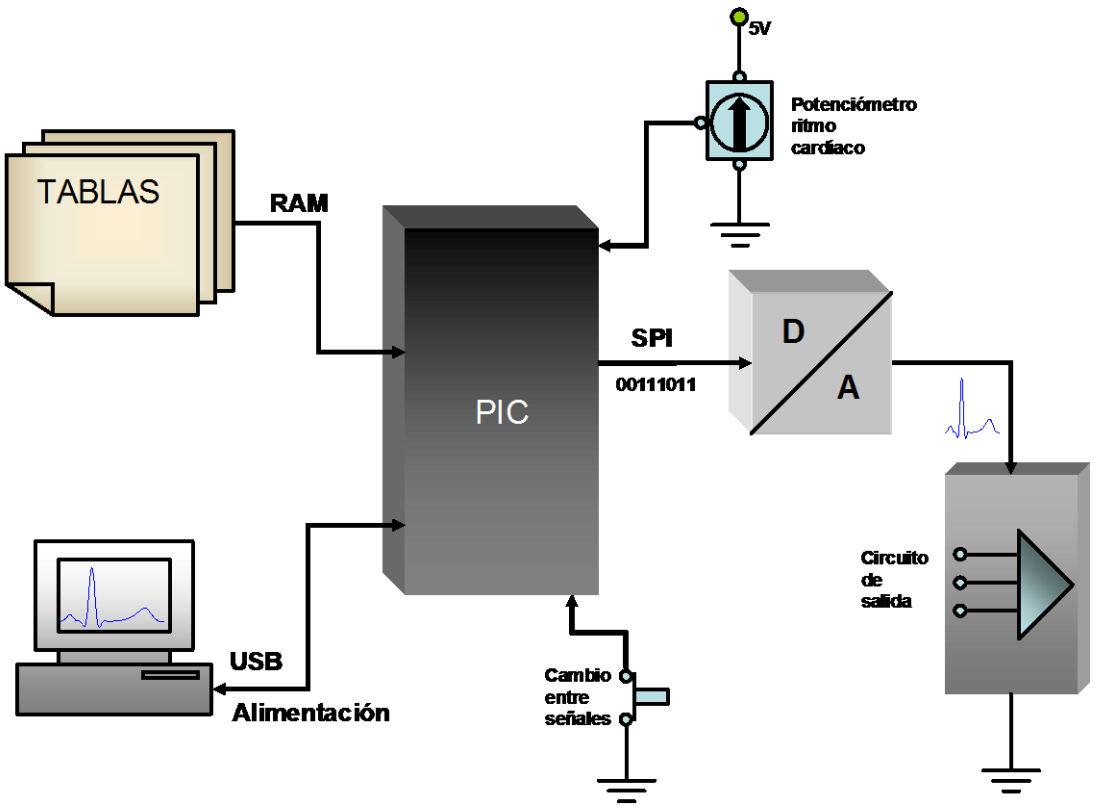


Figura 3.1 - Diagrama Hardware

El sistema comienza con la inicialización del PIC, el cual lee las tablas de valores almacenadas en la memoria RAM del mismo. Estas tablas son enviadas al PC, por puerto USB, y al convertidor A/D, vía SPI, que transforma los valores digitales a valores analógicos, para seguidamente ser enviados al circuito de salida, para su medición.

Para la programación del PIC y la comunicación de datos entre el PIC y el PC, se hace uso de la comunicación USB, además, el puerto USB es utilizado como alimentación del PCB.

Para ajustar la frecuencia de latidos por minuto en la señal de ECG se dispone de un potenciómetro, el cual regula el ritmo cardiaco desde los 38 bpm hasta los 110 bpm.

Para realizar la permuta de la elección de señal se dispone de un pulsador, que al ser accionado se alterna la señal elegida.

Para realizar este proyecto se ha hecho necesario una programación modular a causa de su dificultad, a continuación se expone el entorno de desarrollo con el que se ha elaborado el proyecto, así como del desarrollo software realizado para tal fin.

3.2 - Entorno de desarrollo

Para la generación del firmware a incluir en el microcontrolador, se utilizó una configuración similar a la que puede verse en la figura 3.2.

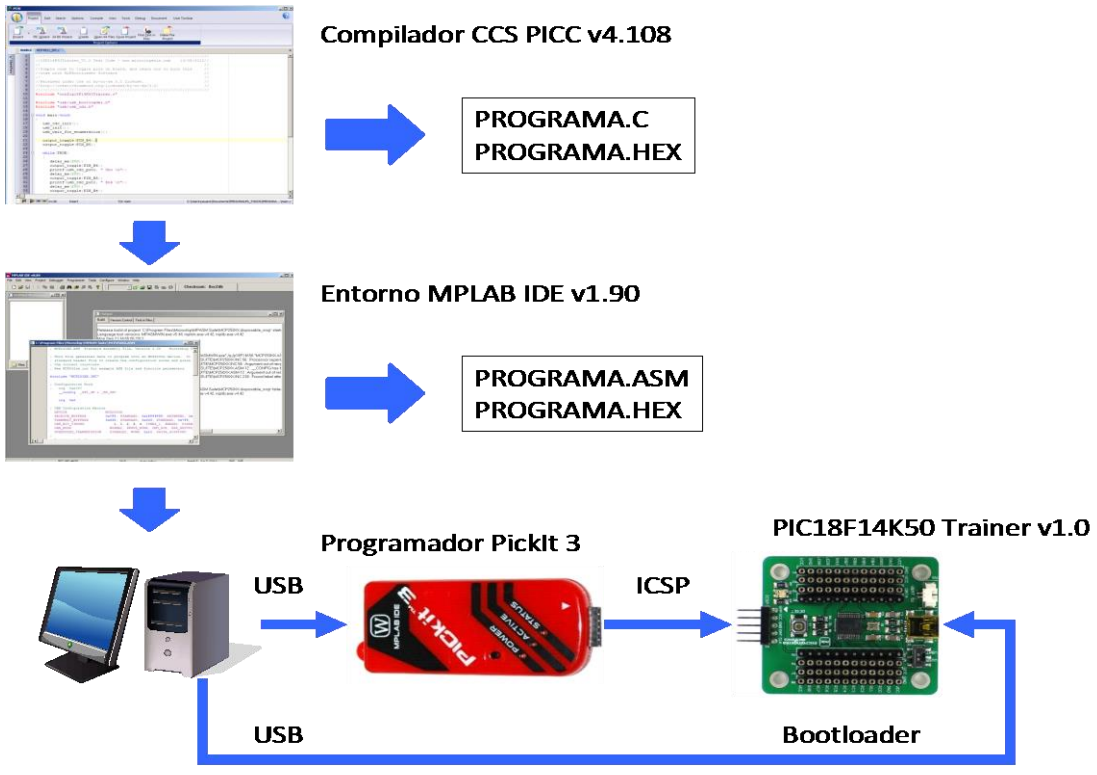


Figura 3.2 - Entorno de desarrollo

Para la programación del PIC se puede proceder de diferentes modos.

- Primer procedimiento:

1º. Se genera el archivo *.ASM* con el programa MPLAB.

2º. El Pickitt 3 se conecta con el PC via USB y se carga en el dispositivo el archivo *.ASM*.

3º. El Pickitt se conecta a la placa de desarrollo mediante el conector ICSP y se vuelca el programa.

Una vez programado por primera vez en el microcontrolador el firmware del bootloader, se puede proceder como sigue:

- Segundo procedimiento

1º. Se genera el archivo *.HEX* con el compilador CCS de C ó con MPLAB.

2º. Se conecta el PC con la placa de desarrollo mediante el conector USB.

3º. Se vuelca el archivo *.HEX* empleando el programa Bootloader.

3.1.1 - Compilador CCS C

Para la programación del microcontrolador se ha utilizado el compilador C de CCS, ya que éste incluye bibliotecas que incorporan determinados comandos que no son estándar, sino específicos para la familia de microcontroladores PIC, puesto que el lenguaje C estándar es demasiado genérico.

Fig. 3.3 - Logotipo CCS

Este compilador posee varias directivas de preprocesado, que no dispone el lenguaje genérico C:

- Directivas relacionadas con la especificación del dispositivo.
- Directivas de cualificación de funciones.
- Directivas de control del compilador.
- Directivas de control de la memoria del microcontrolador.

Este compilador proporciona, además, una gran cantidad de funciones incorporadas para acceder y manipular los periféricos del micro, esto hace mucho más sencillo su configuración, sin verse obligado el usuario a trabajar con los registros asociados a cada funcionalidad.

Además de todo lo señalado anteriormente, este entorno de desarrollo incluye una selección de dispositivos (Device Selection Tool), cuya base de datos incluye todos los dispositivos que puede programar el compilador, incluyendo todas sus características hardware.

Por estas razones se ha decidido utilizar este entorno de desarrollo para realizar todo el software programado en el PIC.

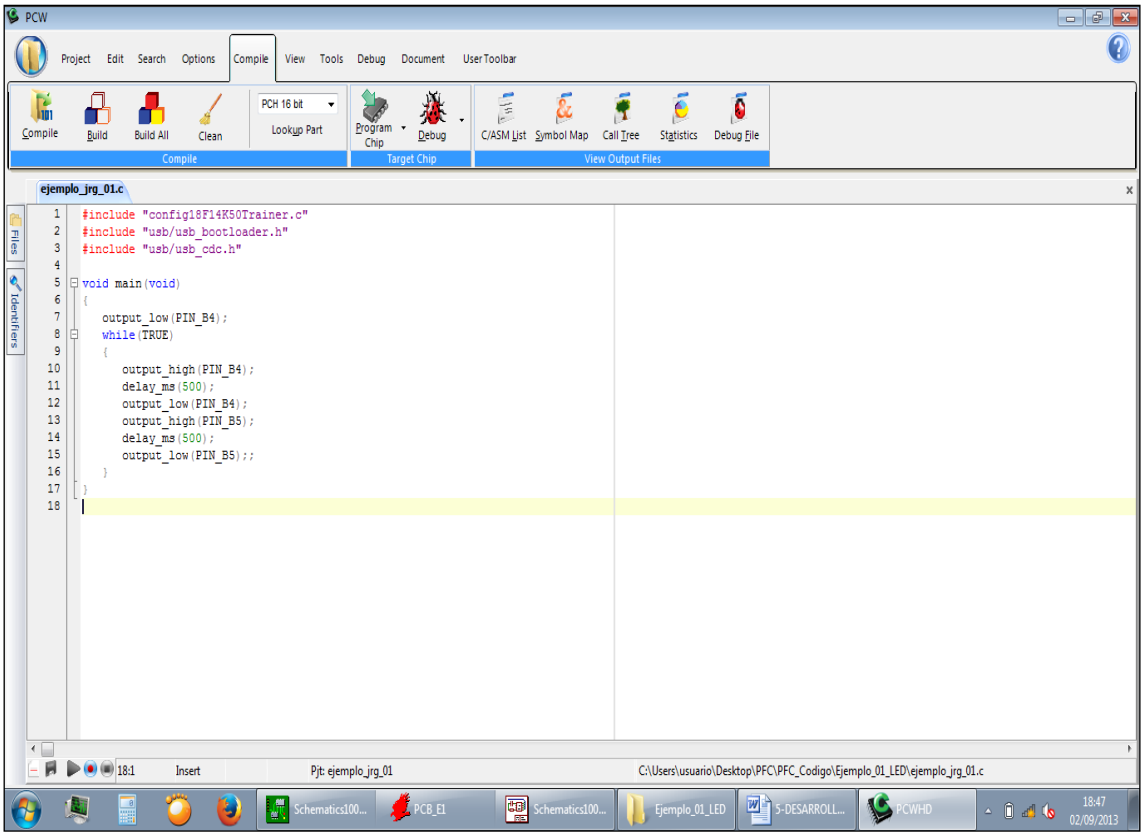


Fig. 3.4 - Entorno de desarrollo CCS

3.2.2 - Placa de desarrollo 18F14K50

Para la toma de contacto con el microcontrolador y el diseño del programa se ha utilizado la placa de desarrollo 18F14K50 Trainer1.0, realizada por Microingenia S.L., cuyo fin es el la implementación rápida de diferentes sistemas en el que se requiera el uso del microcontrolador 18F14K50.

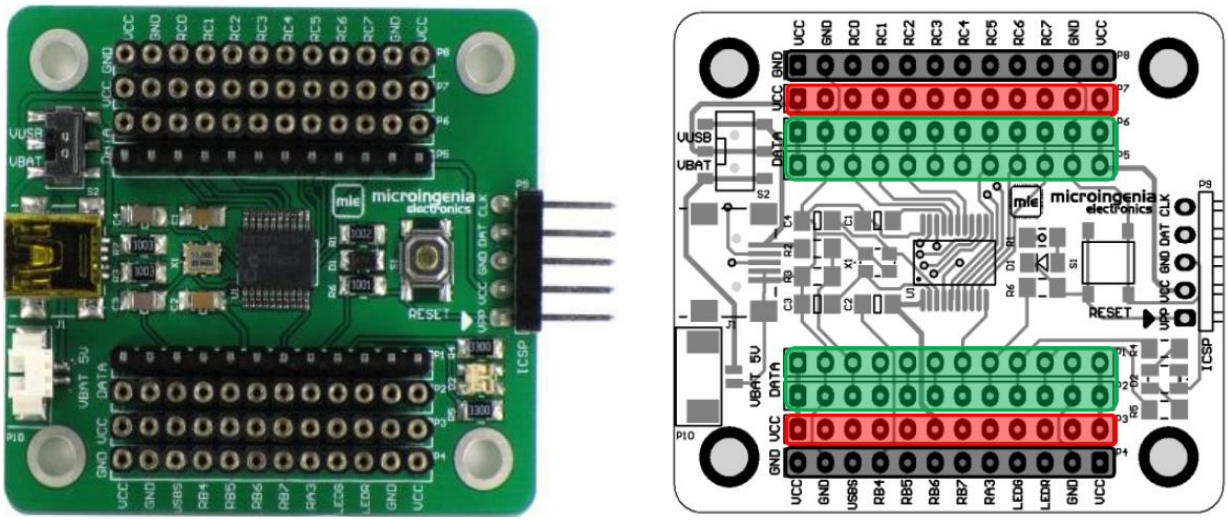


Fig. 3.5 - Placa de desarrollo 18F14K50 Trainerv1.0

- **Características generales**
  a) PIC 18F14K50
  b) 48 MHz CPU
  c) 16 KB de FlashROM
  d) 256 Bytes de EEPROM
  e) 768 Bytes de RAM
  f) 17 Entradas/Salidas Digitales
  g) 9 Entradas Analógicas
  h) LED bicolor
  i) Conector ICSP para conexión con el grabador/depurador Pickit 2/3
  j) Conector USB, para alimentación y programación
  k) Interruptor de Reset
- **Especificaciones**
  a) Alimentación: 5 VCC (USB o VBat)
  b) Dimensiones: 50,8 x 43,18mm (2'' x 1.7'')
  c) Peso: 15,9g

3.3 - Pruebas iniciales

Con el fin de poder compilar programas para la plataforma elegida (PIC18F14K50), se hace preciso incorporar un fichero de configuración especialmente programado a los fines de este proyecto. Para ello se incluyeron los ficheros de cabecera propios del microcontrolador, la

configuración del convertidor AD y las conexiones de E/S utilizadas para la comunicación SPI con el MCP4921. El fichero de configuración resultante puede verse en el siguiente código.

```
#include <18F14K50.h>
#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,
MCLR
#device ADC=10
#use delay(clock=48000000)

#use SPI (BITS=16)
#define MCP4921_SELECT PIN_B5
#define MCP4921_DI     PIN_C7
#define MCP4921_DO     PIN_B4
#define MCP4921_CLK    PIN_B6
```

Configuración del PIC

La estructura de este fichero es:

- ✓ La primera línea del código, hace referencia al microcontrolador que va a ser utilizado en este proyecto (18F14K50).
- ✓ La segunda línea, se encarga de las directivas de preprocesado del microcontrolador.

Comando	Explicación
HS	Habilita el uso de reloj externo de alta velocidad
NOWDT	Deshabilita el uso del perro guardián
NOPROTECT	Deshabilita la protección del código
NOLVP	Deshabilita Low Voltage ICSP Programming
NODEBUG	Deshabilita la depuración en línea
NOBROWNOUT	Deshabilita reset del PIC por caída de voltaje
USBDIV1	Divisor de frecuencia USB igual a 1
PLLEN	Habilita el uso de PLL
CPUDIV1	Divisor de frecuencia CPU igual a 1
PUT	Power Up Timer
MCLR	Habilita pin de reset

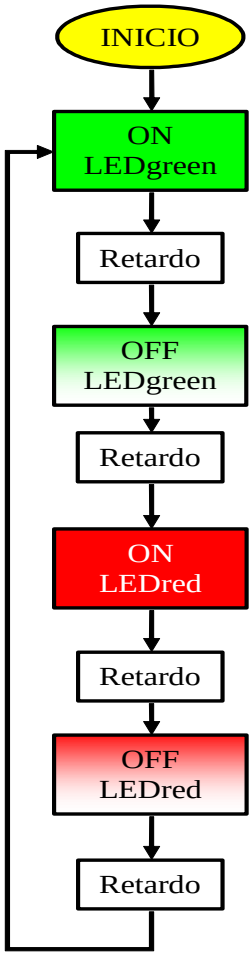
- ✓ La tercera línea, se encarga de configurar el número de bits que se utilizaran para la conversión A/D, dicho convertidor está incluido en el microcontrolador. Esta línea de código debe estar escrita en esta posición, y solo en esta, ya que situándola en otra posición se produciría un error de compilación, y el convertidor no trabajaría con una conversión de 10 bits, si con una conversión de 1 bit.
- ✓ La cuarta línea, configura la frecuencia a la que trabaja el microcontrolador, en este caso trabaja a 48MHz, velocidad necesaria para lo operación de las comunicaciones USB.

- ✓ Las últimas cinco líneas identifican la configuración y los pines necesarios para la comunicación SPI con el dispositivo MCP4921, esto será explicado en un apartado posterior.

3.3.1 - Encendido/apagado de LED

Como primer paso para la puesta a punto del entorno de desarrollo CCS, se plantea un problema clásico en la programación de microcontroladores, el encendido y apagado temporizado de un LED. Para ello, se hace uso de los diodos LEDs incluidos en la placa de desarrollo del microcontrolador 18F14K50.

Con este ejercicio se pretende comprobar la correcta configuración del microcontrolador, así como, de las características básicas de la implementación del lenguaje C bajo el compilador CCS.



El funcionamiento del programa consiste en el encendido y apagado de cada uno de los citados LEDs de forma sucesiva. El sistema seguirá ejecutándose hasta que se presione el botón de reset de la placa de desarrollo, o bien se desconecte de su fuente de energía.

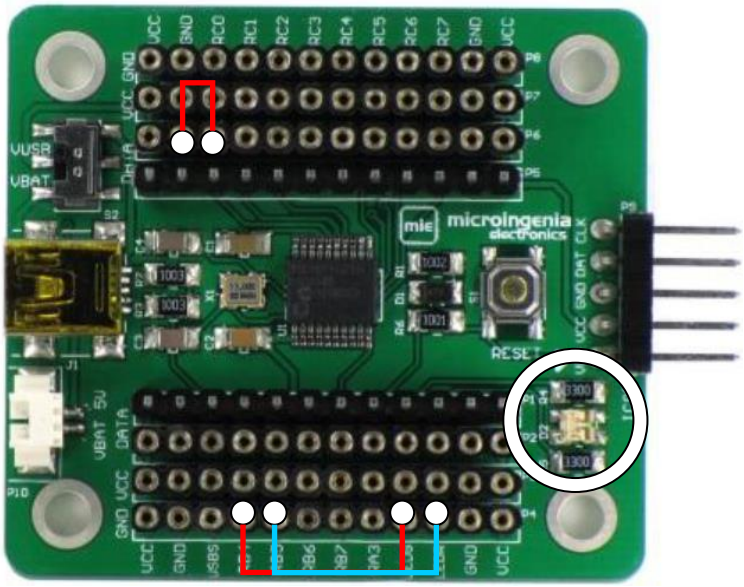


Fig. 3.6 - Conexiones en placa 3.3.1

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"

void main(void)
{
   output_low(PIN_B4);
   while(TRUE)
   {
      output_high(PIN_B4);
      delay_ms(500);
      output_low(PIN_B4);
      output_high(PIN_B5);
      delay_ms(500);
      output_low(PIN_B5);;
   }
}
```

Primer programa encendido/apagado LED

Esta primera forma de programación, la más sencilla, temporiza el retardo mediante el uso de la función 'delay', en este caso, el argumento es de 500ms.

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"

void main(void)
{
    disable_interrupts(GLOBAL);
    disable_interrupts(INT_TIMER0);
    setup_timer_0(RTCC_DIV_256);
    set_timer0(18661);//Valor de precarga para que el contador timer0
cuente 0.5 segundos
    enable_interrupts(GLOBAL);
    enable_interrupts(INT_TIMER0);
```

```
    output_low(PIN_B4);
    output_high(PIN_B5);

    while(TRUE){}
}
#int_timer0
void isr_timer0(void)
{
output_toggle(PIN_B4);
output_toggle(PIN_B5);
}
```

Segundo programa encendido/apagado LED

Este segundo programa, algo más complejo, hace uso de contadores e interrupciones.

El programa comienza con un LED apagado y otro encendido; cuando el contador llega a contar 18632 Bits, alrededor de 0,5 segundos, se produce la interrupción por desbordamiento del contador timer0, pasando a ejecutarse su rutina de interrupción. En esta interrupción se lleva a cabo un 'toggle' (pasar de nivel alto a nivel bajo y viceversa) de los dos pines a los que están conectados los LEDs.

El cálculo del valor de precarga es el siguiente:

Tiempo = 0,5 s

Frecuencia del µC = 48 MHZ

Divisor frecuencia = 128*4

Frecuencia resultante = 93750 Hz

Periodo resultante = 10,67 µs

Conteo necesario = 500000 / 10,67µ = 46.875

Temporización efectiva = 46875*10,67 µs = 500156,25µs = 500,156ms

Valor de precarga = 65536 – 46875 = 18661

3.2 - Comunicación USB

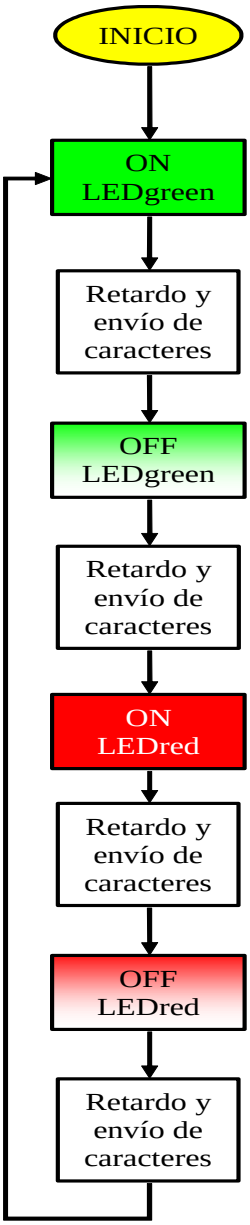
El objetivo de este programa, es la comprensión de la comunicación USB así como de los comandos necesarios para su uso de forma correcta y eficiente.

Los comandos propios de la comunicación USB son:

usb_cdc_init ()	Configura los baudios, bit paridad etc, de la comunicación USB
usb_init ()	Inicializa el hardware del USB
usb_wait_for_enumeration ()	Espera hasta que el PicUSB sea configurado por el host

Se utiliza como fichero base, el algoritmo de programación del punto 4.3.1, de encendido de diodos LEDs (primer ejemplo).

Tomando como base dicho programa, se lleva acabo el envío, vía USB, de diferentes cadenas de caracteres dependiendo de la acción que se lleva a término en cada momento.



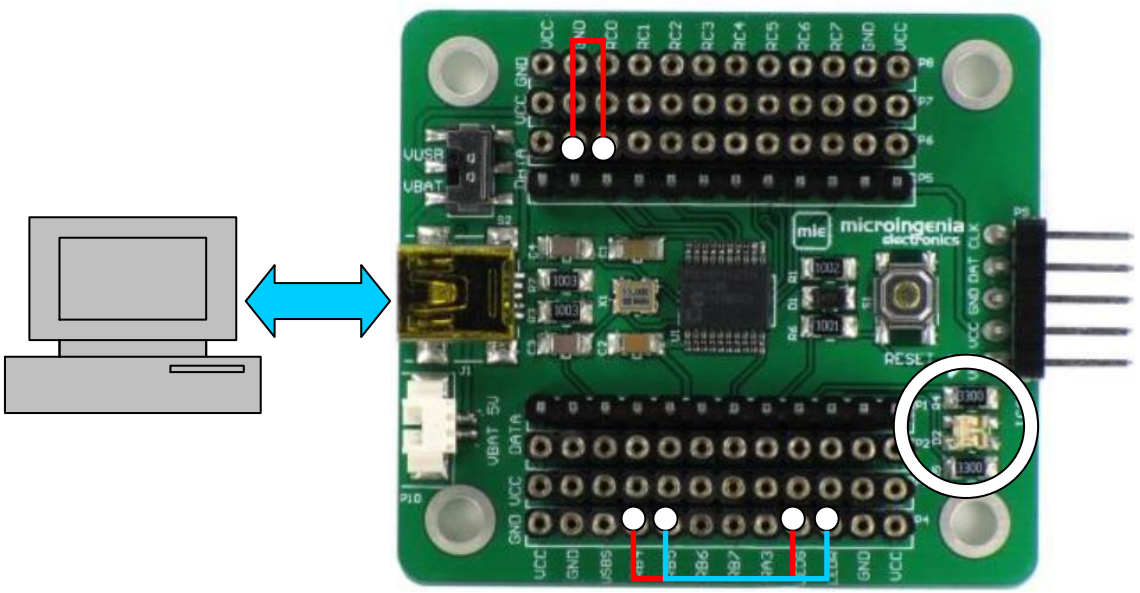


Fig. 3.7 - Conexiones placa de desarrollo 3.3.2

```c
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

void main()
{
   usb_cdc_init();
   usb_init();
   usb_wait_for_enumeration();

   output_low(PIN_B4);
   while(TRUE)
   {
      output_high(PIN_B4);
      printf(usb_cdc_putc, "Encendido LED verde\n\r");
      delay_ms(500);
      output_low(PIN_B4);
      printf(usb_cdc_putc, "Apagado LED verde\n\r");
      output_high(PIN_B5);
      printf(usb_cdc_putc, "Encendido LED rojo\n\r");
      delay_ms(500);
      output_low(PIN_B5);
      printf(usb_cdc_putc, "Apagado LED verde\n\r");
   }
}
```

Programa de comunicación USB

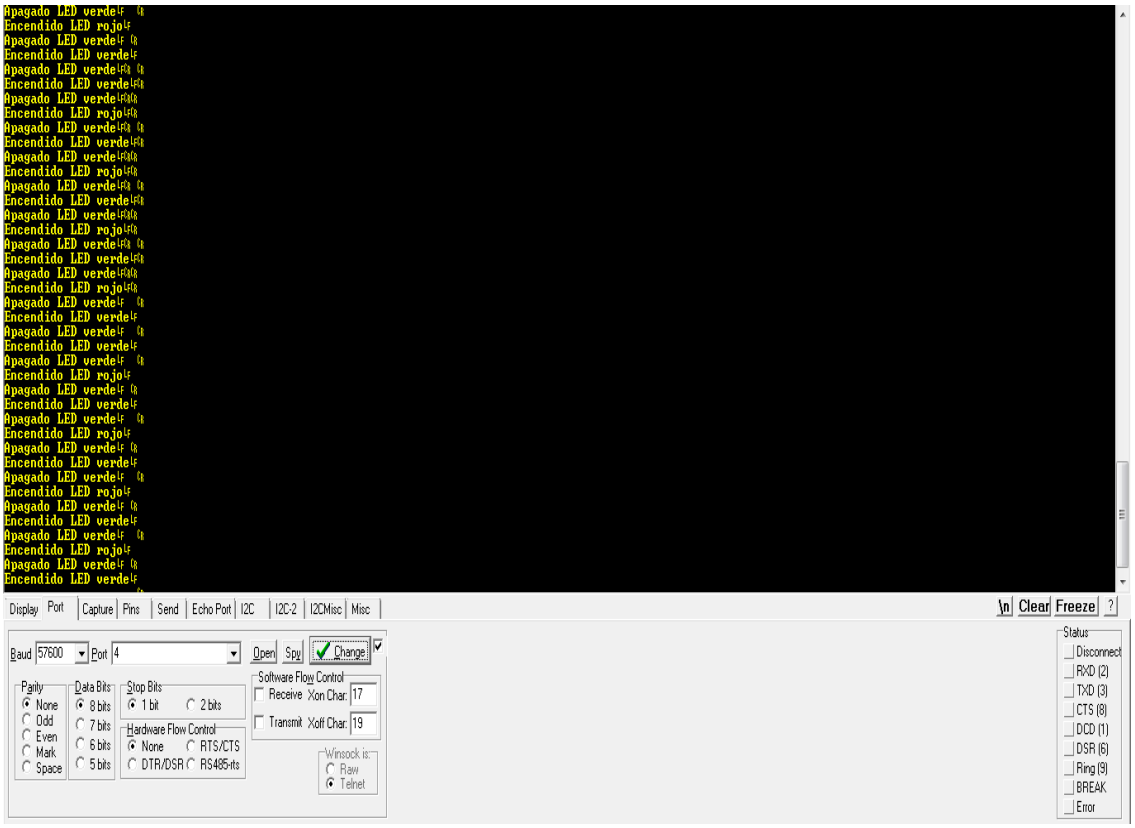


Fig. 3.8 - Envío de cadena de caracteres por USB

En esta imagen (fig. 3.8), se puede apreciar que el programa envía la cadena de caracteres, encendido 'x' o apagado 'y', según la acción que esté realizando el microcontrolador en ese instante.

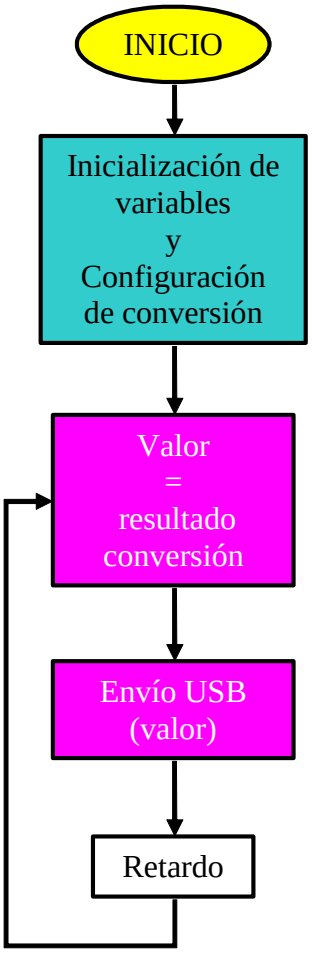
Fig. 3.9 - Logotipo de Realterm®

La imagen 4.8 es una captura de pantalla hecha en un momento cualquiera de la ejecución del software, que está siendo recibida al PC mediante el programa Realterm (www.realterm.sourceforge.net), el cual refleja los datos en formato ASCII.

3.3.3 - Conversión del potenciómetro

Con este programa se pretende configurar la correcta conversión Analógico-Digital del potenciómetro usado en este proyecto.

Su funcionamiento consiste en el envío al PC, vía USB, del dato generado al producirse dicha conversión.



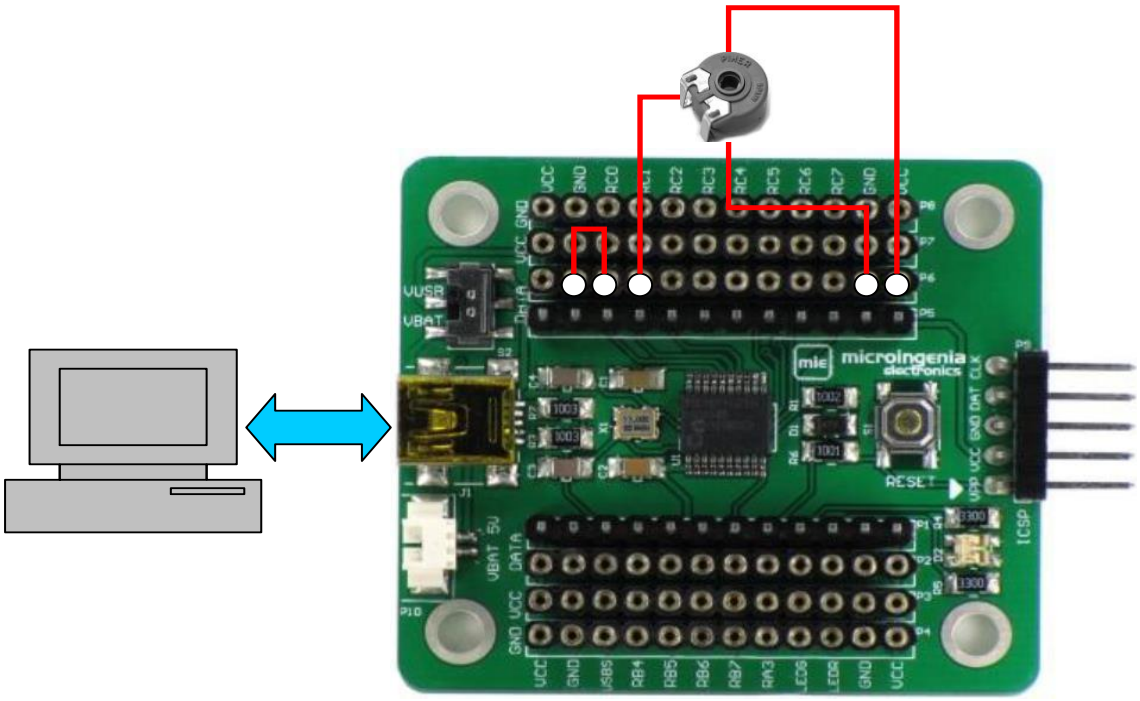


Fig. 3.10 - Conexiones placa de desarrollo 3.3.3

Los parámetros para la conversión A/D del potenciómetro son:

`setup _adc(ADC_CLOCK_INTERNAL)`	Configura la conversión A/D, reloj igual que el reloj interno de la conversión
`setup_adc_ports(sAN5|VSS_VDD)`	Configura que pines son digitales o analógicos, el valor de referencia de la conversión es VSS o VDD
`set_adc_channel(5)`	Se utiliza el canal 5 para la conversión

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"
void main()
{
 signed int16 Res=0;

 usb_cdc_init();
 usb_init();
 usb_wait_for_enumeration();
 setup_adc(ADC_CLOCK_INTERNAL);
 setup_adc_ports(sAN5|VSS_VDD);
 set_adc_channel(5);

  while(true)
   {
         Res = read_adc();
         printf (usb_cdc_putc,"%li \r\n", Res);
         delay_ms(110);
   }
}
```

Programa de conversión del potenciómetro

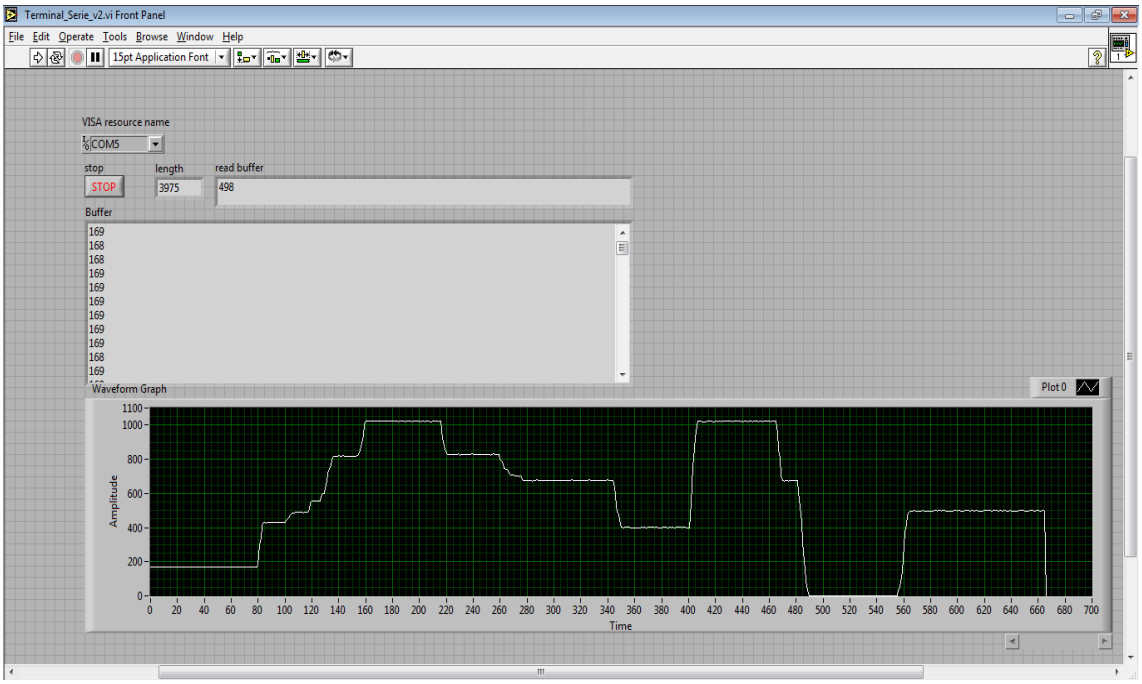


Fig. 3.11 - Captura de los valores de la conversión A/D

Como se puede apreciar en la imagen 3.11, el valor de salida de la variable (Res) fluctúa en igual medida en la que fluctúa la posición resistiva en el potenciómetro.

Fig. 3.12 - Logotipo de Labview®

Para este y todos los programas que se han efectuado posteriormente, se ha realizado un programa en Labview, cuyo cometido es la recepción de los datos envíados al PC para su posterior visualización, tanto en una tabla de valores, como en una gráfica, para que el usuario pueda comprobar los datos de una manera más descriptiva.

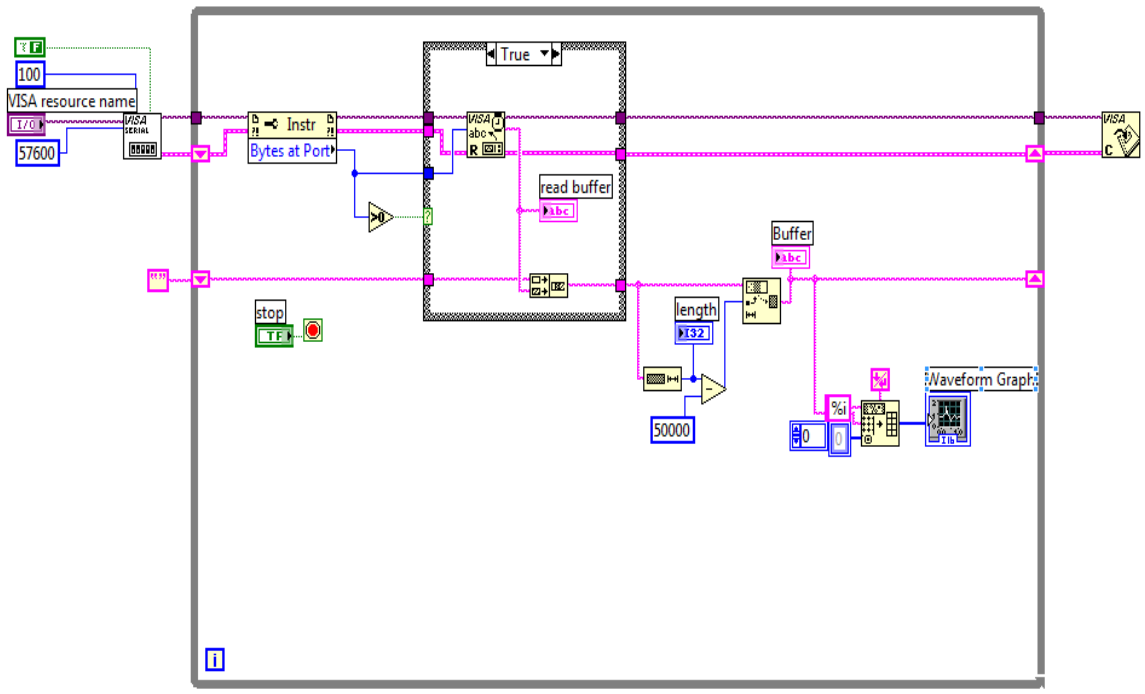


Fig. 3.13 - Programa de obtención de datos en Labview.

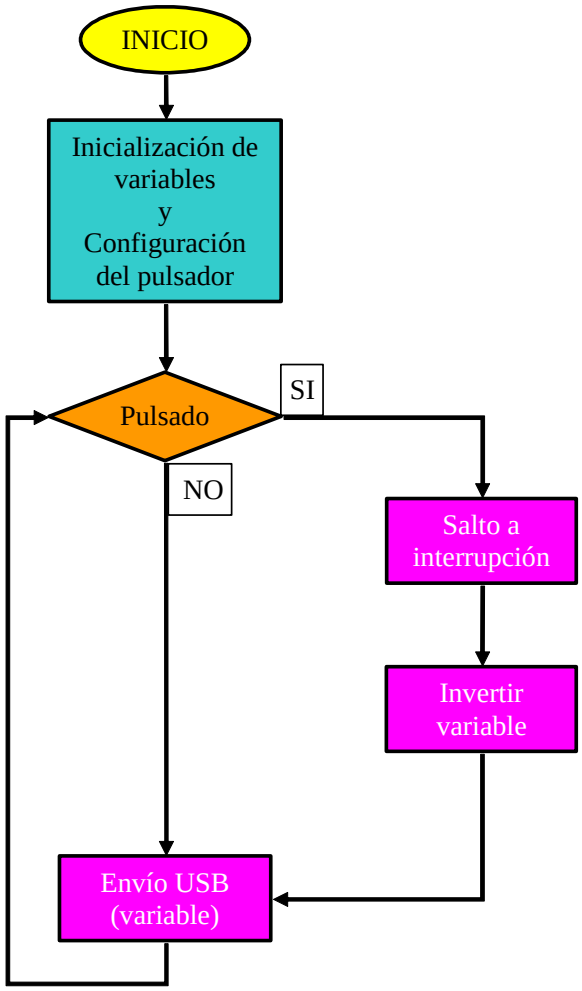
La única modificación que sufre el programa es en su panel frontal, en el cual sólo se modifica la escala de la gráfica, para la óptima visualización de los datos.

3.3.4 - Leer Pulsador

Este programa tiene como función la configuración y el correcto funcionamiento del pulsador al ser accionado.

El funcionamiento del programa consiste en saltar a la interrupción externa 2 cuando se presiona el pulsador. En el interior de esta interrupción se modifica el valor de la variable 'puls', de [1] a [0] o de [0] a [1], según el caso. Este valor es enviado al PC por USB.

`ext_int_edge(2,L_TO_H)`	La interrupción externa 2 se produce del paso 'low' a 'high'
`enable_interrupts(INT_EXT2)`	Habilita la interrupción externa 2
`enable_interrupts(GLOBAL)`	Activa las interrupciones ya habilitadas



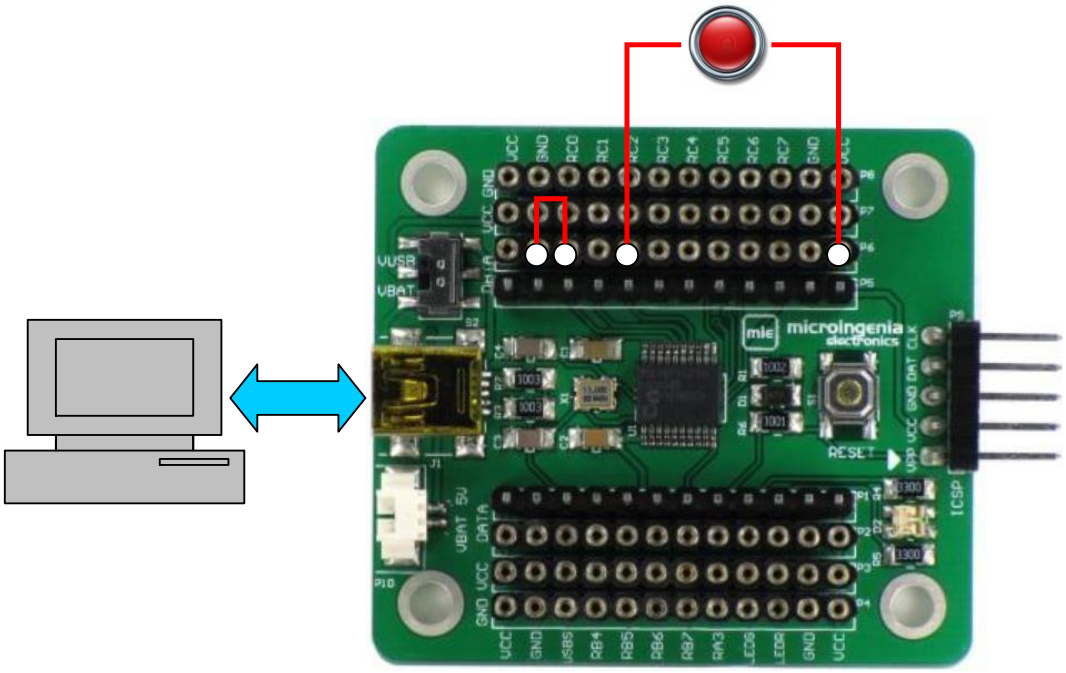


Fig. 3.14 - Conexiones placa de desarrollo 3.3.4

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

int1 puls=0;

void main(void)
{
   usb_cdc_init();
   usb_init();
   usb_wait_for_enumeration();

   ext_int_edge(2,L_TO_H);
   enable_interrupts(INT_EXT2);
   enable_interrupts(GLOBAL);

   while(TRUE)
   {
      printf(usb_cdc_putc,"%i \r\n",puls);
   }
}

#INT_EXT2
void interrupt_RC2()//accionar pulsador conectado a RC2 para que se
imprima por pantalla una señal u otra
{
   puls=!puls;
   delay_ms(175);
}
```

Programa de lectura de pulsador

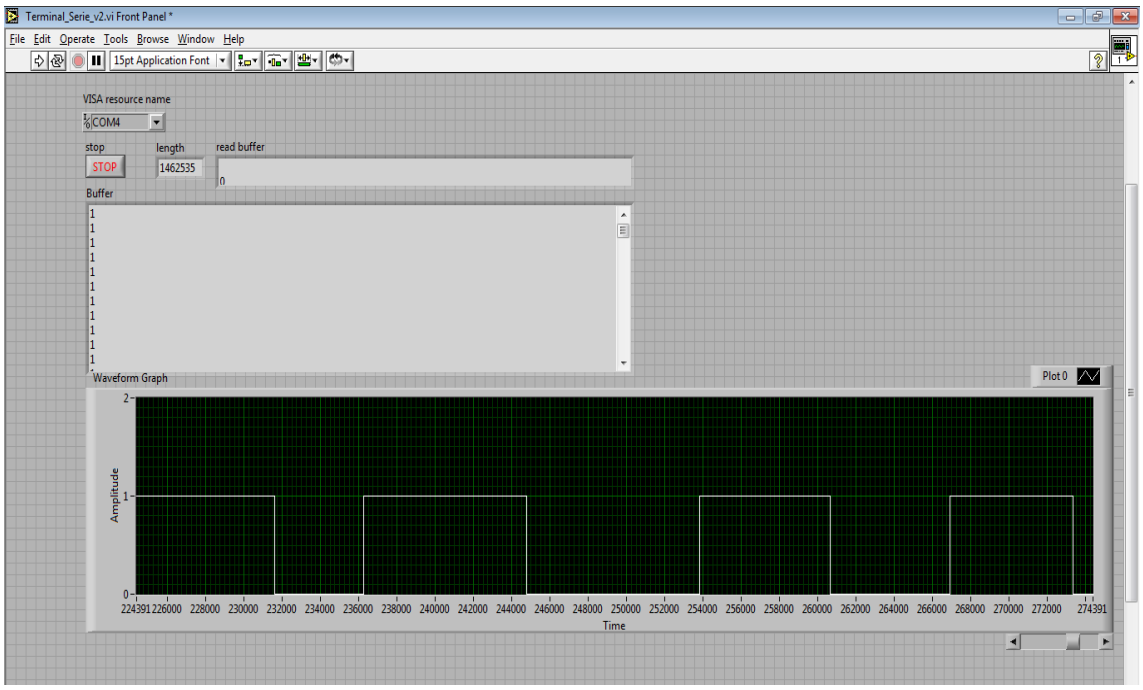


Fig. 3.15 - Captura de valores de la acción del pulsador

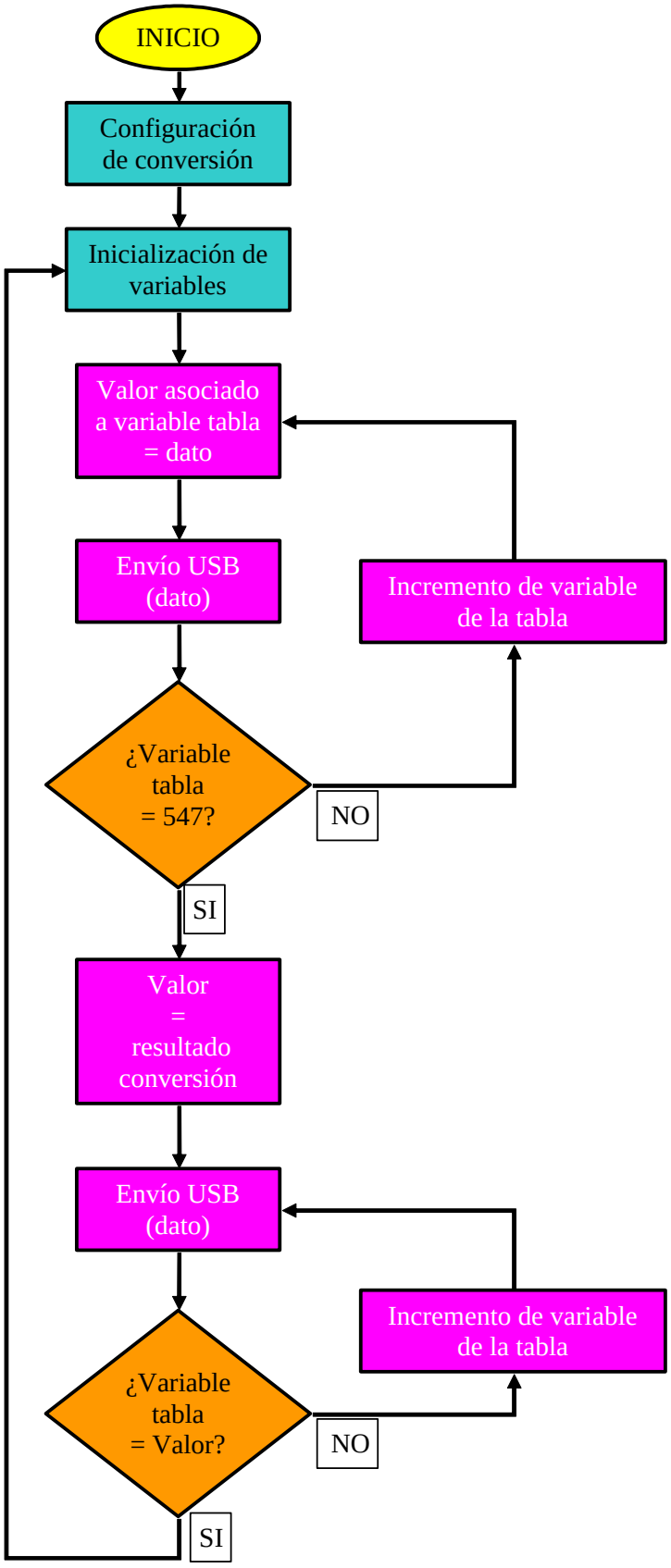
La variable enviada al ordenador, cambia de valor [0] a [1], o viceversa, cuando el pulsador es accionado, como se puede apreciar en la imagen 3.15.

3.3.5 - Lectura y envío de Tabla

El objetivo de este programa, es el envío al PC, por USB, de una tabla de valores perteneciente a la señal de calibración, con el fin de comprobar la correcta lectura de la misma.

La forma en la que se recorre la tabla de valores es mediante el uso del comando 'for', que recorre la tabla desde su primer valor hasta el último, con un espaciado temporal de 1ms entre valor y valor.

El tiempo transcurrido entre el último valor de un ciclo de la señal y el primer valor del siguiente ciclo, es dependiente del valor de conversión del potenciómetro. Durante este tiempo se envía por USB el último valor del primer ciclo, que además coincide con el primer valor del segundo ciclo.


INICIO
Configuración de conversión
Inicialización de variables
Valor asociado a variable tabla = dato
Envío USB (dato)
Incremento de variable de la tabla
¿Variable tabla = 547?
NO
SI
Valor = resultado conversión
Envío USB (dato)
Incremento de variable de la tabla
¿Variable tabla = Valor?
NO
SI

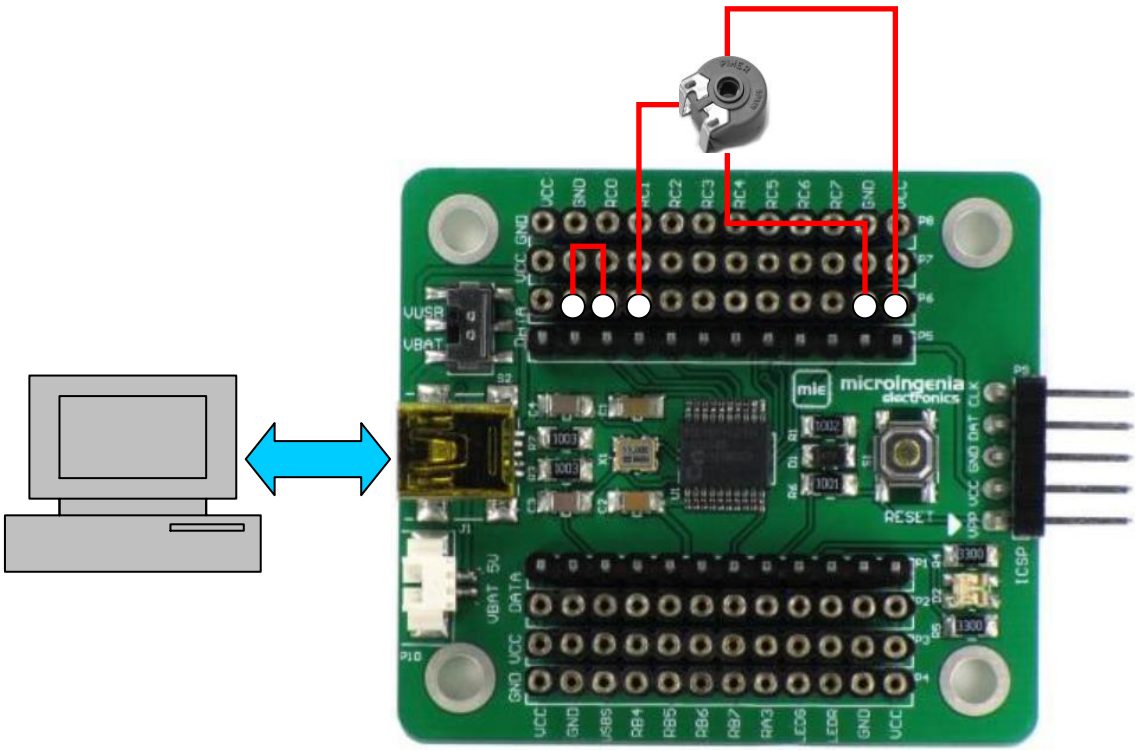


Fig. 3.16 - Conexiones placa de desarrollo 3.3.5

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

signed int32 i=0, j=0, xpot=0, reposo=540;
const int16 y_data[543] = {
939, 940, 941, 942, 944, 945, 946, 947, 951, 956,
962, 967, 973, 978, 983, 989, 994, 1000, 1005, 1015,
1024, 1034, 1043, 1053, 1062, 1075, 1087, 1100, 1112, 1121,
1126, 1131, 1136, 1141, 1146, 1151, 1156, 1164, 1172, 1179,
1187, 1194, 1202, 1209, 1216, 1222, 1229, 1235, 1241, 1248,
1254, 1260, 1264, 1268, 1271, 1275, 1279, 1283, 1287, 1286,
1284, 1281, 1279, 1276, 1274, 1271, 1268, 1266, 1263, 1261,
1258, 1256, 1253, 1251, 1246, 1242, 1237, 1232, 1227, 1222,
1218, 1215, 1211, 1207, 1203, 1199, 1195, 1191, 1184, 1178,
1171, 1165, 1159, 1152, 1146, 1141, 1136, 1130, 1125, 1120,
1115, 1110, 1103, 1096, 1088, 1080, 1073, 1065, 1057, 1049,
1040, 1030, 1021, 1012, 1004, 995, 987, 982, 978, 974,
970, 966, 963, 959, 955, 952, 949, 945, 942, 939,
938, 939, 940, 941, 943, 944, 945, 946, 946, 946,
946, 946, 946, 946, 946, 947, 950, 952, 954, 956,
958, 960, 962, 964, 965, 965, 965, 965, 965, 965,
963, 960, 957, 954, 951, 947, 944, 941, 938, 932,
926, 920, 913, 907, 901, 894, 885, 865, 820, 733,
606, 555, 507, 632, 697, 752, 807, 896, 977, 1023,
1069, 1127, 1237, 1347, 1457, 2085, 2246, 2474, 2549, 2595,
2641, 2695, 3083, 3135, 3187, 3217, 3315, 3403, 3492, 3581,
3804, 3847, 3890, 3798, 3443, 3453, 3297, 3053, 2819, 2810,
2225, 2258, 1892, 1734, 1625, 998, 903, 355, 376, 203,
30, 33, 61, 90, 119, 160, 238, 275, 292, 309,
325, 343, 371, 399, 429, 484, 542, 602, 652, 703,
758, 802, 838, 856, 875, 895, 917, 938, 967, 1016,
1035, 1041, 1047, 1054, 1060, 1066, 1066, 1064, 1061, 1058,
```

```
1056, 1053, 1051, 1048, 1046, 1043, 1041, 1038, 1035, 1033,
1030, 1028, 1025, 1022, 1019, 1017, 1014, 1011, 1008, 1006,
1003, 1001, 999, 998, 996, 994, 993, 991, 990, 988,
986, 985, 983, 981, 978, 976, 973, 971, 968, 966,
963, 963, 963, 963, 963, 963, 963, 963, 963, 963,
963, 963, 963, 963, 963, 963, 963, 963, 963, 963,
964, 965, 966, 967, 968, 969, 970, 971, 972, 974,
976, 978, 980, 983, 985, 987, 989, 991, 993, 995,
997, 999, 1002, 1006, 1011, 1015, 1019, 1023, 1028, 1032,
1036, 1040, 1045, 1050, 1055, 1059, 1064, 1069, 1076, 1082,
1088, 1095, 1101, 1107, 1114, 1120, 1126, 1132, 1141, 1149,
1158, 1166, 1173, 1178, 1183, 1188, 1193, 1198, 1203, 1208,
1214, 1221, 1227, 1233, 1240, 1246, 1250, 1254, 1259, 1263,
1269, 1278, 1286, 1294, 1303, 1309, 1315, 1322, 1328, 1334,
1341, 1343, 1345, 1347, 1349, 1351, 1353, 1355, 1357, 1359,
1359, 1359, 1359, 1359, 1358, 1356, 1354, 1352, 1350, 1347,
1345, 1343, 1341, 1339, 1336, 1334, 1332, 1329, 1327, 1324,
1322, 1320, 1317, 1315, 1312, 1307, 1301, 1294, 1288, 1281,
1275, 1270, 1265, 1260, 1256, 1251, 1246, 1240, 1233, 1227,
1221, 1214, 1208, 1201, 1194, 1186, 1178, 1170, 1162, 1154,
1148, 1144, 1140, 1136, 1131, 1127, 1123, 1118, 1114, 1107,
1099, 1090, 1082, 1074, 1069, 1064, 1058, 1053, 1048, 1043,
1038, 1034, 1029, 1025, 1021, 1017, 1013, 1009, 1005, 1001,
997, 994, 990, 991, 992, 994, 996, 997, 999, 998,
997, 996, 995, 994, 993, 991, 990, 989, 989, 989,
989, 989, 989, 989, 988, 986, 984, 983, 981, 980,
982, 984, 986, 988, 990, 993, 995, 997, 999, 1002,
1005, 1008, 1012};

void main(void)
{
    usb_cdc_init();
    usb_init();
    usb_wait_for_enumeration();

    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(sAN5|VSS_VDD );
    set_adc_channel(5);

   while(TRUE)
   {
      xpot=read_adc();
      reposo=1024-xpot;

      if (reposo <0) reposo =0;
      if (reposo >1024) reposo = 1024;

      for(i=0;i<543;i++)
      {
       j = y_data[i];
       printf(usb_cdc_putc,"%li \r\n",j);
       delay_ms(1);
      }
      for(i=0;i<reposo;i++)
      {
       printf(usb_cdc_putc,"%li \r\n",j);
       delay_ms(1);
      }
   }
}
```

Programa de envío de señal de referencia

Fig. 3.17 - Captura de envío de señal de referencia

Como se puede deducir de la imagen 3.17, el tiempo entre un ciclo y otro de la señal no es constante, si no que fluctúa.

3.4 - Generación de las tablas de la señal

Fig. 3.18 - Logotipo Matlab®

Para este proyecto se ha utilizado Matlab para el filtrado y procesado de las señales empleadas, ya que este programa es capaz de realizar las transformaciones matemáticas requeridas para dicho filtrado.

Matlab es un paquete de software orientado hacia el cálculo numérico científico e ingenieril. Integra cálculo numérico, computación de matrices y gráficos en un entorno de trabajo cómodo para el usuario. Fue escrito inicialmente en base a los ya existentes paquetes de cálculo matricial LINPACK y EISPACK, aunque posteriormente se han ido añadiendo librerías, denominadas toolboxes, especializadas en distintas áreas.

En entornos universitarios se ha convertido, junto con MATEMÁTICA y MAPLE, en una herramienta básica para cursos de matemáticas aplicadas así como para cursos avanzados en otras áreas. En entornos industriales se utiliza para investigar y resolver problemas prácticos y cálculos de la ingeniería. Es de destacar la aplicación en el estudio, simulación y diseño de los sistemas dinámicos de control.

A continuación se exponen los algoritmos realizados en Matlab para la conversión de las señales proporcionadas por physiobank, ya que éstas no son válidas para incluirlas directamente en la programación del microcontrolador.

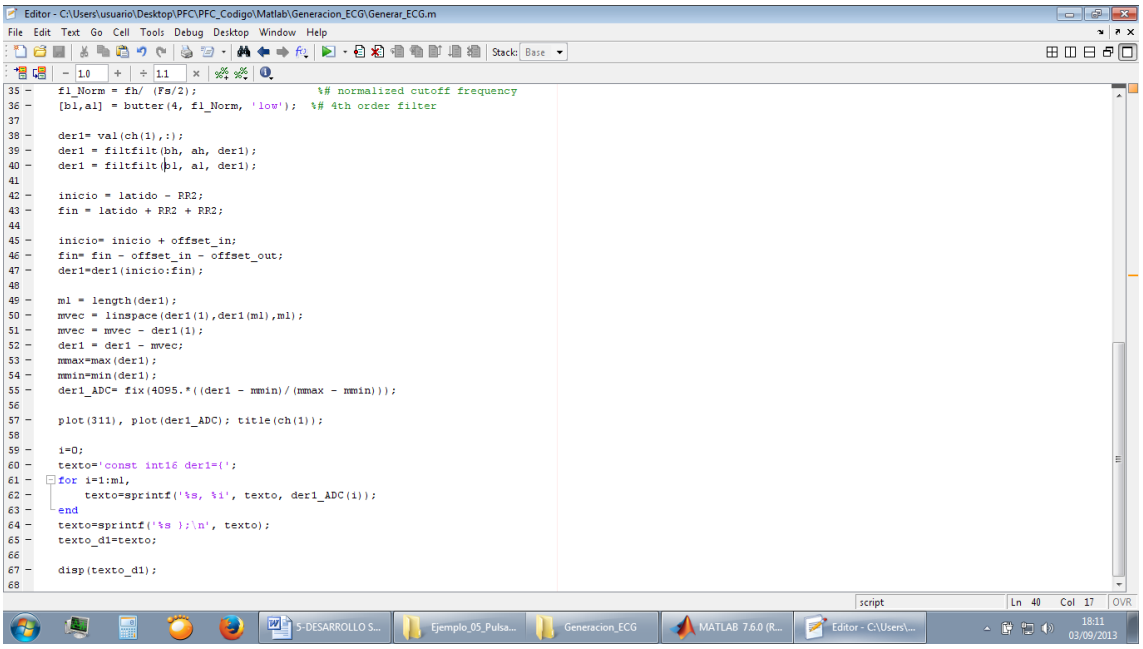


Fig. 3.19 - Entorno de desarrollo Matlab

3.4.1 - Physiobank

PhysioBank es un gran y creciente archivo de grabaciones digitales de señales fisiológicas y datos similares, pensadas para su uso por la comunidad de investigación biomédica. Actualmente incluye bases de datos de múltiples parámetros, como pueden ser cardiopulmonares, neuronales, etc. además de otras señales biomédicas de sujetos sanos y pacientes con una variedad de condiciones con importantes implicaciones de salud pública, incluyendo la muerte súbita cardíaca, insuficiencia cardíaca, epilepsia, apnea del sueño, y envejecimiento. En la actualidad contiene más de 50 bases de datos que se pueden descargar libremente.

Para este proyecto se han seleccionado las bases de datos PTB y CAP Sleep, dos bases de datos incluidas en el almacenamiento de physiobank,

Las señales de ECG consideradas fueron extraídas de la base de datos PTB, cuya dirección web es: http://physionet.org/physiobank/database/ptbdb/.

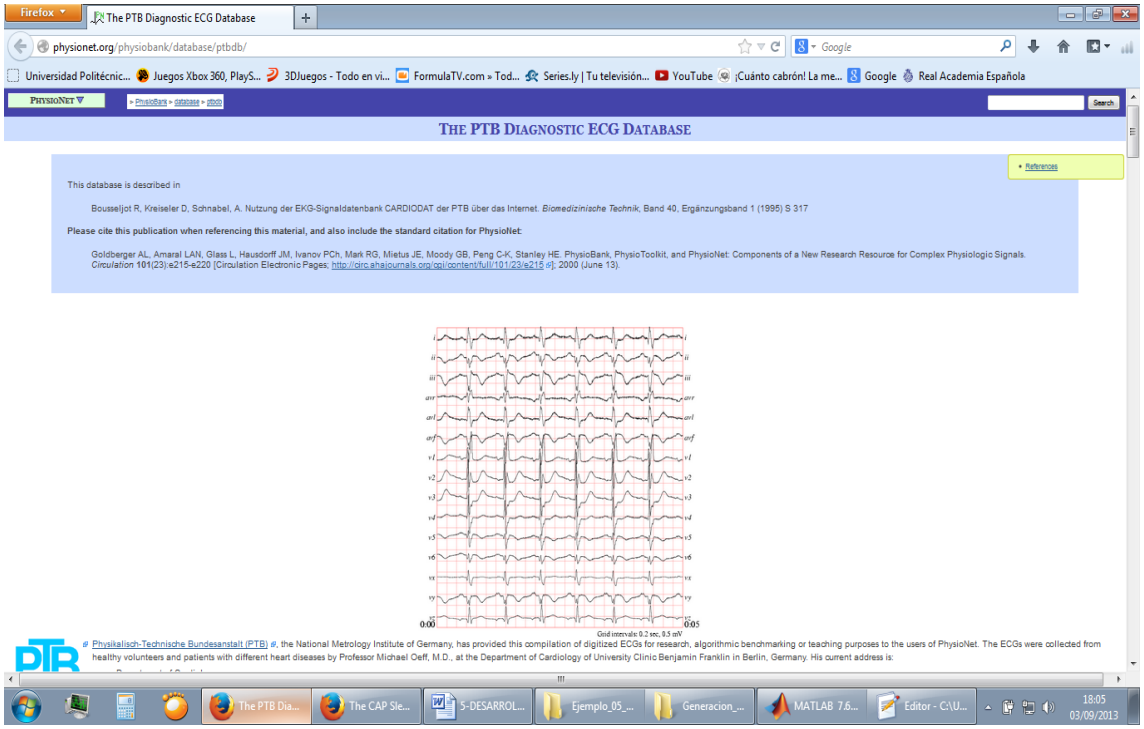


Fig. 3.20 - Base de datos PTB en physiobank

La base de datos contiene 549 registros de 290 pacientes. Los pacientes son sujetos de 17 a 87 años con una media de edad de 57,2. Se dispone de 209 hombres cuya edad media es 55,5 años y 81 mujeres con una edad media de 61,6 años. Cada sujeto es representado mediante el uso de uno a cinco registros, que incluyen 15 señales medidas simultáneamente: las 12 derivaciones convencionales (I, II, III, AVR, AVL, AVF, v1, v2, v3, v4, v5, v6), junto con las tres derivaciones de Frank ECG (vx, vy, vz). Estas señales se encuentran muestreadas con una frecuencia de 1000 muestras por segundo, con una resolución de 16 bits en un rango de ± 16,384 mV.

Las señales de EEG consideradas fueron extraídas de la base de datos CAP Sleep Research, cuya dirección web es: (http://physionet.org/physiobank/database/capslpdb/)

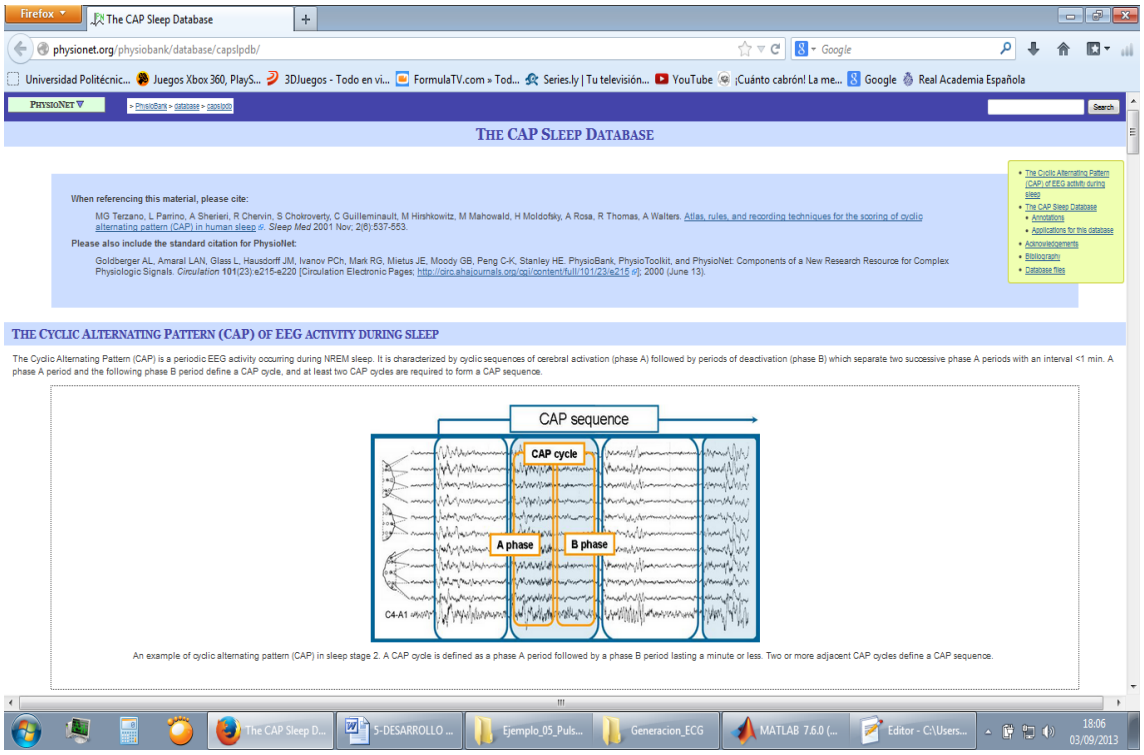


Fig. 3.21 - Base de datos CAP en physiobank

El Cyclic Alternating Pattern (PAC) es una actividad electroencefalográfica periódica que ocurre durante el sueño en fase NREM (No Rapid Eye Movement). Se caracteriza por secuencias cíclicas de la activación cerebral (fase A), seguidas por periodos de desactivación (fase B), que separan dos fases sucesivas A, cuyos períodos de duración son de menos de 1 minuto. Un período de la fase A y el siguiente período de la fase B definen un ciclo de CAP, y se requieren al menos dos ciclos CAP para formar una secuencia PAC.

Los 16 sujetos sanos del estudio no presentan ningún trastorno psicológico y están libres de fármacos que puedan alterar el sistema nervioso central. Las 92 grabaciones patológicas incluyen 40 grabaciones de sujetos diagnosticados de insomnio, narcolepsia, etc. Los pacientes son sujetos de 14 a 82 años con una media de edad de 45,2 años.

La base de datos del sueño CAP es una colección de 108 registros polisomnográficos registrados en el Centro de Trastornos del Sueño del Ospedale Maggiore de Parma, Italia. Las formas incluyen al menos 3 canales de EEG (F3 o F4, C3 o C4 y O1 y O2, se hace referencia A1 o A2), señales de respiración (aire, esfuerzo abdominal y torácica, y Saturación de O2) y ECG.

3.4.2 - Generación de señales de ECG con Matlab

La explicación de funcionamiento del software final usado será explicada paso a paso para su adecuado entendimiento. Se parte del registro ‘s03061rem’ de la base de datos PTB,

señal de ECG con una amplitud máxima de 1487 µV muestreada a 1000 Hz y 16 bits, de 10s de duración (85200 muestras).

```
load('s0306lrem.mat');
ch=[1];
der1= val(ch(1),:);
plot(der1)
```

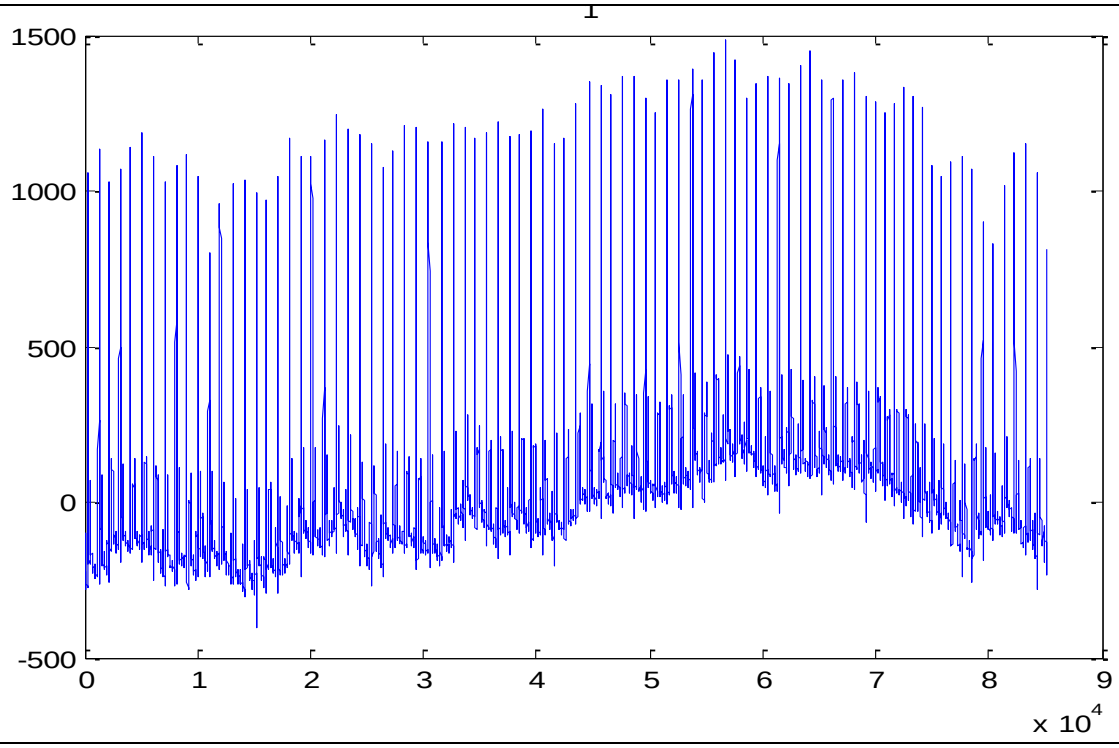


Fig. 4.22 - Señal de partida sin procesar

Como se puede comprobar en la imagen, la señal, de gran duración, presenta un contenido espectral muy elevado, por lo que debe ser filtrada.

```
load('s0306lrem.mat');

fl=0.5;
fh=20;
Fs=1000;
ch=[1];
der1= val(ch(1),:);

fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency
[bh,ah] = butter(4, fh_Norm, 'high');    %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');    %# 4th order filter

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);
plot(der1); title(ch(1));
```

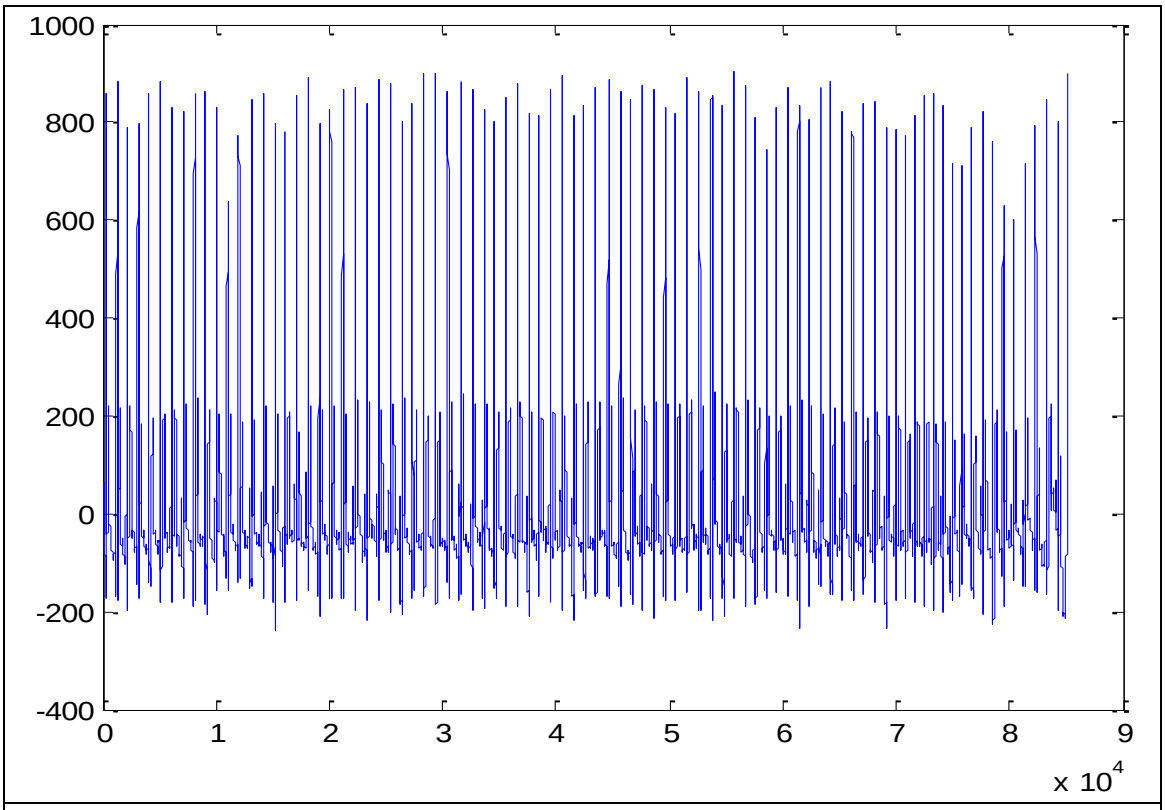


Fig. 4.23 - Señal filtrada

El filtrado se lleva a cabo mediante la aplicación sucesiva de un filtro pasa altos (0,5 Hz) y pasa bajos (20 Hz) implementados mediante aproximación Butterworth de orden 4. Con esto se consigue eliminar las oscilaciones de la línea base (y la eliminación de los niveles de continua) y los términos de alta frecuencia (incluyendo la señal de red).

```
load('s0306lrem.mat');
fl=0.5;
fh=20;
Fs=1000;

ch=[1];
der1= val(ch(1),:);

fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency
[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

latido = 4150;
offset_in= 202;
offset_out= 340;
inicio= latido - offset_in;
fin= latido + offset_out;
der1=der1(inicio:fin);
```

```
plot(der1); title(ch(1));
```

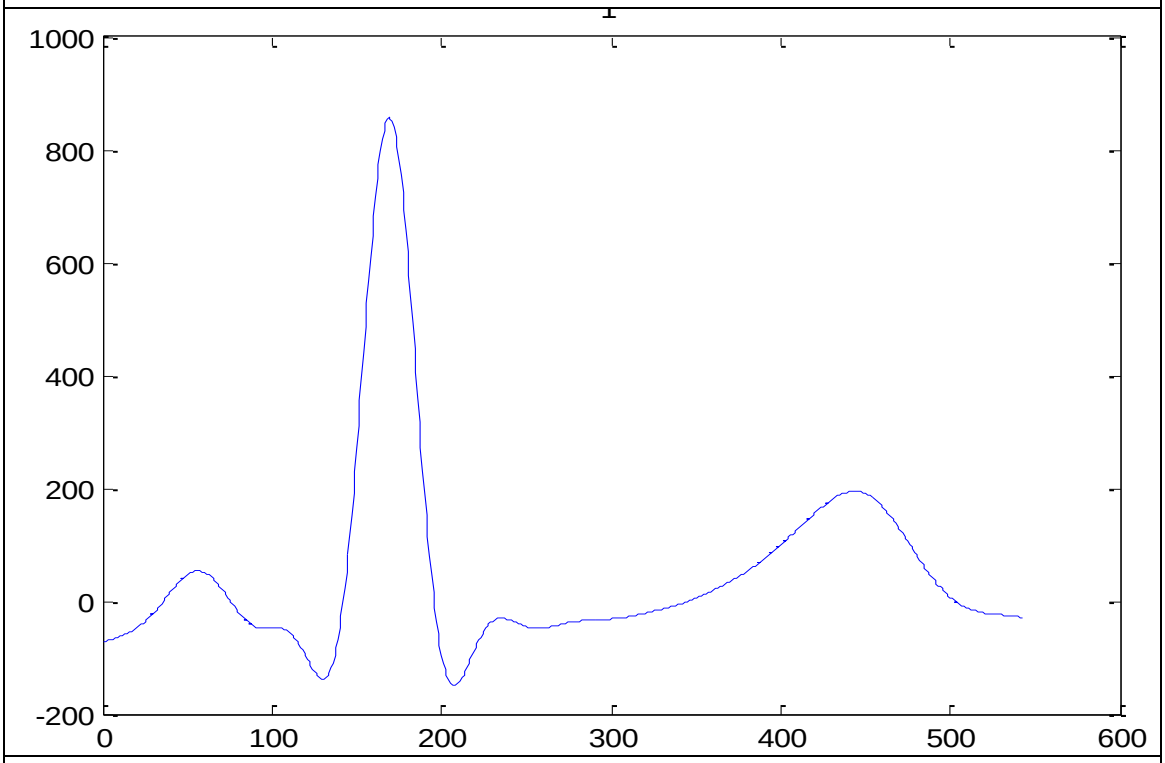


Fig. 3.24 - Latido seleccionado

Seguidamente, para poder seleccionar el tramo de señal a grabar como tabla, se implementa una rutina de navegación y extracción de latidos. Como punto de partida se escoge el valor 4150, al cual posee un offset inferior (202) y un offset superior (340), con esto se obtiene el latido seleccionado.

```
load('s0306lrem.mat');

fl=0.5;
fh=20;
Fs=1000;

ch=[1];
der1= val(ch(1),:);

fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency
[bh,ah] = butter(4, fh_Norm, 'high');    %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');    %# 4th order filter

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);


latido = 4150;
offset_in= 202;
offset_out= 340;
inicio= latido - offset_in;
```

```
fin= latido + offset_out;
der1=der1(inicio:fin);

ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

plot(der1); title(ch(1));
```

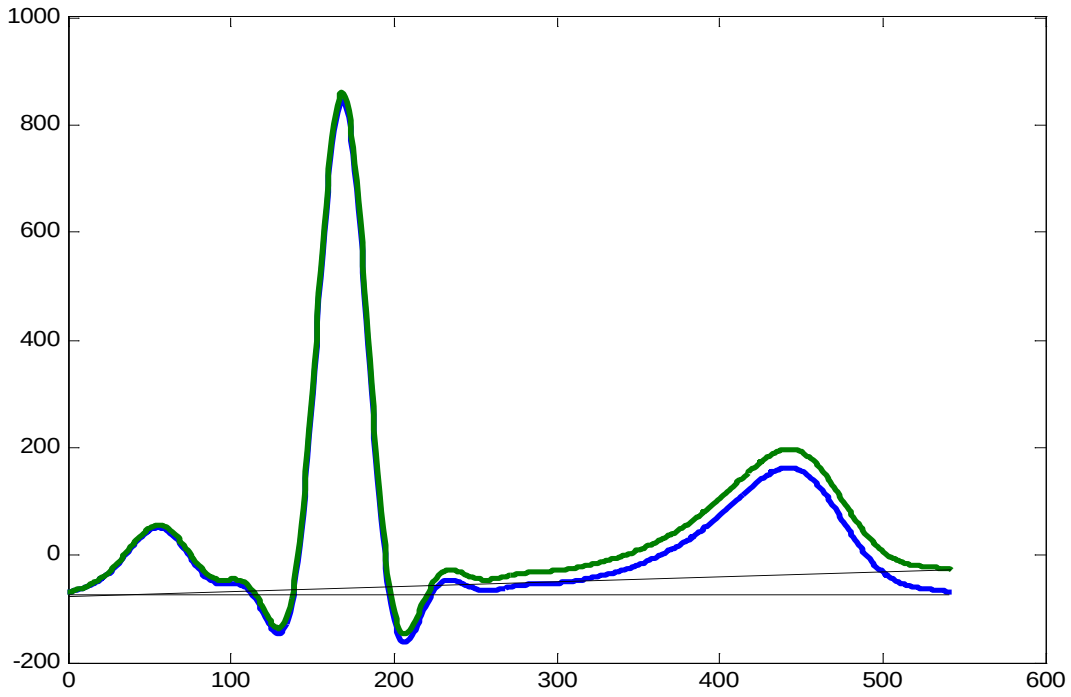


Fig. 3.25 - Detalle del proceso de restauración de la línea base

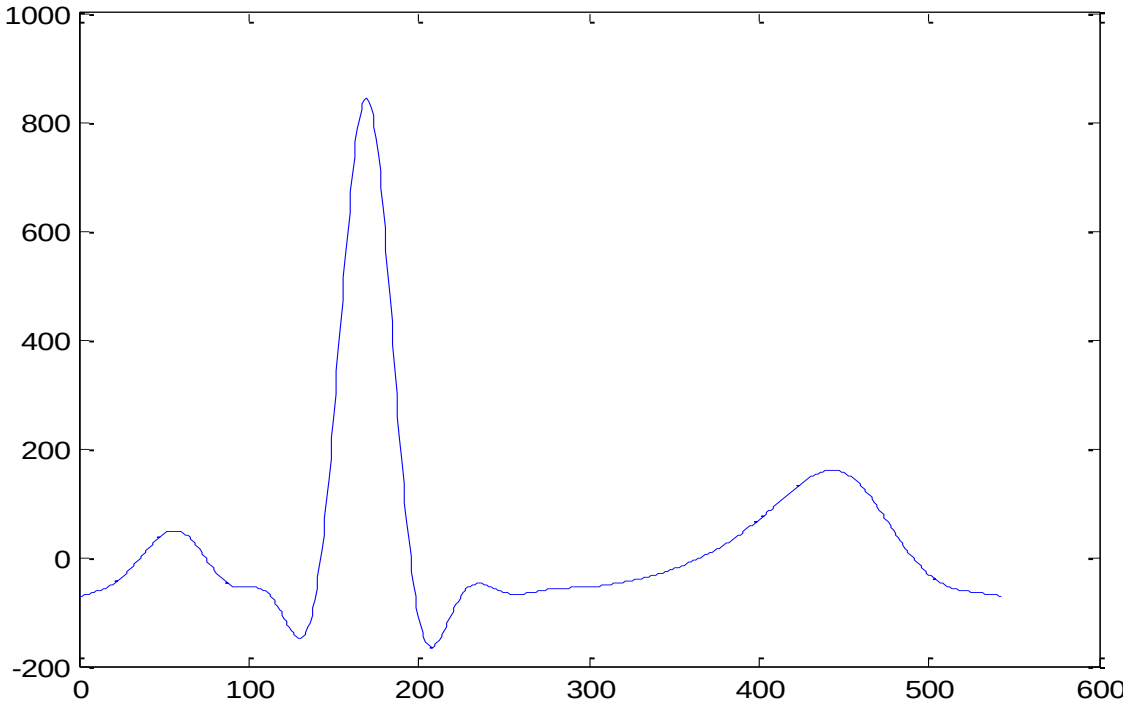


Fig. 3.26 - Latido seleccionado con línea base corregida

Acto seguido se implementa una subrutina encargada de hacer coincidir el primer y último valor de la señal. Esto se realiza trazando una recta desde el principio de la señal hasta su final y, seguidamente, restando cada valor de la señal a cada valor correspondiente de la recta. Se puede traducir como un giro de la señal, manteniendo fijo el primer valor y girando la señal hasta que el primer y último valor coincidan.

```
load('s0306lrem.mat');

fl=0.5;
fh=20;
Fs=1000;

ch=[1];
der1= val(ch(1),:);

fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency
[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);


latido = 4150;
offset_in= 202;
offset_out= 340;
inicio= latido - offset_in;
fin= latido + offset_out;
der1=der1(inicio:fin);

ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

mmax=max(der1);
mmin=min(der1);
der1_ADC= fix(4095.*((der1 - mmin)/(mmax - mmin)));

plot(der1_ADC); title(ch(1));
```

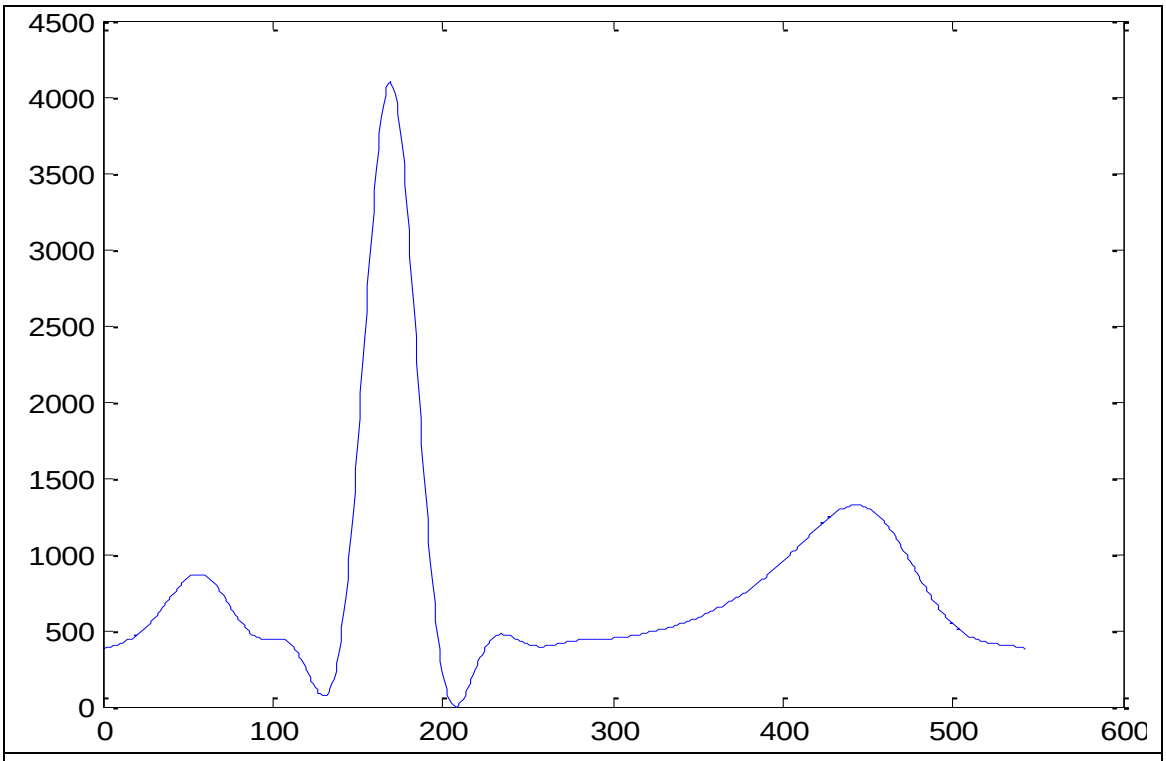


Fig. 3.27 - Versión discreta del latido seleccionado con línea base corregida

Este cuarto paso es el último propiamente dicho de procesado de la señal, puesto que pasada esta etapa la señal se encuentra definida en el rango en el que puede operar el convertidor D/A. Con la subrutina añadida en este paso se realiza un cambio de escala para acabar con un rango de valores de [0-4095].

```
load('s0306lrem.mat');

fl=0.5;
fh=20;
Fs=1000;

ch=[1];
der1= val(ch(1),:);

fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency
[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);


latido = 4150;
offset_in= 202;
offset_out= 340;
inicio= latido - offset_in;
fin= latido + offset_out;
der1=der1(inicio:fin);
```

```
ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

mmax=max(der1);
mmin=min(der1);
der1_ADC= fix(4095.*((der1 - mmin)/(mmax - mmin)));

plot(der1_ADC); title(ch(1));

i=0;
texto='const int16 der1={';
for i=1:ml,
    texto=sprintf('%s, %i', texto, der1_ADC(i));
end
texto=sprintf('%s };\n', texto);
texto_d1=texto;

disp(texto_d1);
```

SOFTWARE_FINAL_ECG_MATLAB

Este programa es el finalmente usado, incorpora todo lo previamente explicado, añadiéndole en la parte final del programa una nueva subrutina. Esta subrutina es la encargada de mostrar la señal como una tabla de valores, mostrando cada valor en formato texto.

Este proceso está realizado sobre la primera onda del conjunto de señales perteneciente a la señal de electrocardiograma. Dicho proceso es efectuado sobre el resto de las 14 señales provistas por physiobank. A continuación se muestran las tablas de valores de las 15 señales:

```
const int16 Der_I[543]  ={381, 384, 387, 390, 393, 396, 399, 403, 407,
411, 415, 420, 424, 429, 435, 441, 447, 454, 461, 469, 477, 485, 495,
504, 515, 526, 537, 549, 562, 575, 588, 602, 616, 630, 645, 660, 675,
690, 705, 720, 734, 749, 763, 776, 789, 801, 813, 823, 833, 842, 849,
856, 861, 865, 867, 868, 868, 866, 863, 859, 853, 846, 837, 827, 816,
803, 790, 775, 760, 744, 728, 710, 693, 675, 657, 639, 622, 604, 588,
571, 556, 541, 527, 514, 502, 491, 481, 473, 465, 459, 454, 450, 447,
445, 444, 443, 443, 443, 444, 445, 445, 446, 445, 444, 442, 439, 435,
430, 423, 414, 404, 392, 378, 362, 345, 326, 306, 284, 261, 238, 215,
191, 168, 146, 126, 107, 92, 80, 72, 68, 70, 78, 93, 115, 145, 183,
231, 288, 354, 431, 518, 614, 721, 838, 965, 1101, 1245, 1396, 1555,
1720, 1889, 2062, 2236, 2412, 2586, 2759, 2927, 3089, 3244, 3391,
3527, 3651, 3762, 3859, 3940, 4005, 4053, 4083, 4095, 4088, 4063,
4020, 3960, 3882, 3789, 3680, 3558, 3423, 3276, 3120, 2956, 2785,
2610, 2431, 2251, 2071, 1893, 1718, 1547, 1382, 1224, 1074, 932, 800,
677, 565, 463, 372, 292, 222, 162, 113, 73, 43, 21, 6, 0, 0, 5, 16,
32, 51, 73, 98, 125, 152, 181, 210, 238, 266, 292, 318, 342, 364, 384,
402, 418, 432, 443, 453, 461, 467, 471, 473, 474, 473, 472, 469, 466,
461, 456, 451, 446, 440, 435, 429, 424, 419, 414, 410, 406, 403, 400,
398, 396, 395, 394, 394, 394, 395, 396, 397, 399, 401, 403, 405, 408,
410, 413, 415, 418, 420, 423, 425, 427, 429, 431, 433, 434, 436, 437,
438, 439, 440, 441, 441, 442, 442, 443, 443, 443, 444, 444, 444, 445,
```

```
445, 446, 446, 447, 448, 449, 449, 450, 452, 453, 454, 456, 457, 459,
461, 463, 465, 467, 469, 471, 473, 476, 478, 481, 483, 486, 488, 491,
493, 496, 499, 501, 504, 507, 509, 512, 515, 518, 521, 524, 527, 530,
533, 537, 540, 544, 548, 551, 555, 559, 564, 568, 572, 577, 582, 587,
591, 596, 602, 607, 612, 617, 623, 628, 634, 639, 645, 651, 656, 662,
668, 674, 680, 686, 693, 699, 706, 712, 719, 726, 733, 740, 748, 755,
763, 771, 779, 787, 796, 805, 813, 822, 832, 841, 850, 860, 869, 879,
889, 899, 909, 919, 930, 940, 951, 961, 972, 982, 993, 1004, 1015,
1026, 1036, 1047, 1058, 1070, 1081, 1092, 1103, 1114, 1125, 1136,
1147, 1157, 1168, 1178, 1189, 1199, 1209, 1219, 1228, 1237, 1246,
1254, 1263, 1270, 1277, 1284, 1291, 1296, 1302, 1306, 1310, 1314,
1316, 1318, 1320, 1320, 1320, 1319, 1317, 1315, 1311, 1307, 1302,
1296, 1290, 1282, 1274, 1264, 1255, 1244, 1232, 1220, 1207, 1193,
1179, 1164, 1148, 1132, 1115, 1098, 1080, 1063, 1044, 1026, 1007, 988,
968, 949, 930, 910, 891, 871, 852, 833, 814, 795, 777, 759, 741, 724,
707, 690, 674, 658, 643, 628, 614, 601, 587, 575, 563, 551, 540, 530,
520, 511, 502, 494, 486, 479, 472, 466, 460, 455, 450, 445, 441, 437,
433, 430, 427, 424, 422, 419, 417, 415, 413, 412, 410, 408, 407, 405,
404, 402, 401, 399, 398, 396, 395, 393, 391, 389, 387, 385, 383, 381};
```

Array_valores_derivación_I

```
const int16 Der_II[543] ={552, 560, 568, 576, 585, 593, 602, 610, 619,
628, 637, 645, 654, 663, 672, 680, 689, 697, 705, 713, 720, 727, 733,
739, 744, 749, 753, 757, 760, 762, 764, 765, 765, 764, 763, 761, 759,
756, 752, 748, 744, 739, 733, 728, 722, 715, 709, 702, 695, 688, 681,
673, 666, 658, 650, 643, 635, 627, 619, 611, 603, 594, 586, 578, 569,
560, 551, 542, 533, 524, 514, 505, 495, 485, 476, 466, 456, 447, 437,
428, 418, 409, 401, 393, 385, 377, 370, 364, 358, 353, 349, 345, 341,
339, 337, 335, 334, 334, 334, 334, 334, 334, 335, 335, 335, 335, 334,
332, 329, 326, 321, 315, 308, 300, 290, 279, 266, 252, 236, 220, 202,
183, 163, 143, 123, 103, 84, 65, 48, 32, 19, 9, 2, 0, 1, 8, 20, 38,
63, 94, 133, 180, 235, 298, 369, 449, 537, 634, 739, 852, 972, 1100,
1234, 1374, 1519, 1669, 1822, 1978, 2135, 2293, 2451, 2607, 2760,
2909, 3054, 3193, 3324, 3448, 3563, 3669, 3764, 3848, 3920, 3981,
4028, 4063, 4085, 4095, 4091, 4074, 4045, 4004, 3951, 3888, 3813,
3729, 3636, 3535, 3427, 3312, 3191, 3066, 2938, 2807, 2674, 2540,
2406, 2273, 2141, 2012, 1886, 1764, 1646, 1532, 1424, 1320, 1223,
1132, 1046, 967, 893, 826, 765, 710, 661, 617, 578, 544, 515, 491,
470, 453, 439, 429, 421, 416, 412, 410, 410, 411, 413, 415, 418, 422,
425, 428, 431, 434, 437, 439, 441, 442, 443, 443, 442, 441, 440, 438,
436, 434, 431, 428, 425, 421, 418, 415, 411, 408, 405, 402, 399, 396,
394, 392, 390, 389, 387, 386, 386, 385, 385, 386, 386, 387, 388, 389,
390, 392, 394, 395, 397, 399, 401, 404, 406, 408, 410, 412, 414, 416,
418, 420, 422, 424, 425, 427, 428, 430, 431, 433, 434, 435, 437, 438,
439, 440, 441, 443, 444, 445, 447, 448, 450, 452, 454, 456, 458, 460,
462, 465, 468, 471, 474, 477, 480, 483, 487, 491, 495, 499, 503, 507,
511, 516, 520, 525, 529, 534, 539, 544, 549, 553, 558, 563, 568, 573,
578, 584, 589, 594, 599, 604, 609, 615, 620, 626, 631, 637, 642, 648,
654, 660, 666, 673, 679, 686, 693, 700, 707, 715, 722, 730, 739, 747,
756, 765, 774, 784, 794, 804, 814, 825, 836, 847, 859, 870, 882, 894,
907, 919, 932, 945, 959, 972, 986, 999, 1013, 1027, 1041, 1056, 1070,
1085, 1100, 1115, 1130, 1145, 1160, 1175, 1191, 1206, 1222, 1238,
1253, 1269, 1285, 1301, 1316, 1332, 1348, 1364, 1379, 1394, 1410,
1425, 1440, 1454, 1469, 1483, 1496, 1509, 1522, 1534, 1546, 1557,
1567, 1577, 1586, 1594, 1601, 1607, 1613, 1617, 1621, 1623, 1624,
1625, 1624, 1621, 1618, 1614, 1608, 1601, 1593, 1584, 1573, 1562,
1549, 1535, 1520, 1504, 1487, 1469, 1451, 1431, 1411, 1390, 1368,
1346, 1323, 1300, 1277, 1254, 1230, 1206, 1182, 1158, 1134, 1111,
1088, 1064, 1042, 1019, 997, 976, 955, 935, 915, 896, 877, 859, 842,
826, 810, 795, 780, 766, 753, 741, 729, 718, 707, 697, 687, 679, 670,
662, 655, 648, 642, 636, 630, 625, 620, 615, 610, 606, 602, 599, 595,
```

```
592, 589, 586, 583, 581, 579, 576, 574, 572, 570, 568, 567, 565, 563,
562, 561, 559, 558, 557, 556, 555, 554, 553, 552};
```

Array_valores_derivación_II

```
const int16 Der_III[543] ={676, 685, 695, 704, 714, 723, 733, 743,
753, 763, 772, 782, 791, 800, 809, 818, 826, 833, 840, 846, 851, 855,
859, 861, 862, 863, 862, 859, 856, 852, 846, 839, 831, 821, 811, 800,
787, 774, 761, 746, 731, 716, 700, 684, 668, 652, 636, 620, 605, 589,
575, 561, 547, 534, 522, 511, 500, 490, 480, 472, 464, 457, 450, 444,
439, 434, 429, 425, 421, 417, 414, 411, 408, 404, 401, 398, 395, 392,
388, 385, 381, 377, 373, 369, 365, 361, 358, 354, 350, 346, 343, 340,
337, 334, 332, 331, 329, 328, 328, 327, 327, 328, 328, 329, 330, 331,
332, 333, 334, 334, 333, 332, 331, 328, 324, 320, 314, 307, 299, 289,
278, 265, 252, 237, 220, 203, 185, 166, 147, 128, 108, 89, 71, 54, 38,
25, 14, 5, 1, 0, 3, 11, 24, 43, 68, 99, 137, 182, 235, 295, 363, 439,
523, 614, 713, 820, 934, 1055, 1183, 1316, 1455, 1599, 1746, 1897,
2050, 2205, 2360, 2514, 2667, 2817, 2964, 3106, 3241, 3370, 3492,
3604, 3706, 3798, 3879, 3947, 4003, 4046, 4076, 4092, 4095, 4083,
4059, 4021, 3970, 3907, 3833, 3747, 3652, 3547, 3434, 3314, 3187,
3056, 2921, 2782, 2642, 2501, 2360, 2220, 2082, 1947, 1815, 1688,
1566, 1449, 1338, 1234, 1136, 1044, 960, 882, 812, 748, 691, 640, 596,
557, 524, 497, 474, 455, 441, 431, 423, 419, 417, 417, 418, 421, 425,
430, 436, 441, 447, 452, 457, 462, 466, 469, 472, 474, 475, 476, 476,
475, 473, 471, 468, 464, 461, 457, 452, 448, 443, 439, 434, 429, 425,
421, 417, 413, 410, 407, 405, 403, 401, 400, 399, 399, 399, 399, 400,
401, 402, 404, 406, 408, 410, 412, 415, 417, 420, 422, 425, 427, 430,
432, 434, 436, 438, 440, 442, 444, 445, 447, 448, 449, 450, 451, 452,
453, 454, 455, 455, 456, 457, 458, 459, 460, 461, 463, 464, 466, 468,
470, 472, 474, 477, 479, 482, 485, 489, 492, 496, 500, 504, 508, 512,
516, 520, 525, 529, 534, 538, 543, 548, 552, 557, 561, 566, 570, 575,
579, 583, 588, 592, 596, 601, 605, 609, 614, 618, 623, 627, 632, 637,
642, 647, 653, 658, 664, 670, 676, 683, 689, 696, 704, 711, 719, 727,
735, 744, 753, 762, 771, 781, 791, 801, 812, 823, 834, 845, 856, 868,
880, 892, 904, 917, 930, 943, 956, 969, 983, 996, 1010, 1024, 1038,
1052, 1066, 1081, 1095, 1110, 1125, 1140, 1155, 1170, 1185, 1200,
1216, 1231, 1246, 1262, 1277, 1293, 1308, 1323, 1339, 1354, 1369,
1384, 1398, 1413, 1427, 1441, 1454, 1467, 1480, 1492, 1504, 1514,
1525, 1534, 1543, 1551, 1558, 1565, 1570, 1575, 1578, 1580, 1581,
1582, 1580, 1578, 1575, 1570, 1565, 1558, 1550, 1540, 1530, 1518,
1506, 1492, 1478, 1462, 1446, 1428, 1410, 1392, 1372, 1352, 1332,
1311, 1289, 1268, 1246, 1224, 1202, 1180, 1158, 1137, 1115, 1094,
1073, 1053, 1033, 1013, 994, 975, 957, 940, 923, 907, 892, 877, 863,
849, 836, 824, 813, 802, 791, 781, 772, 763, 755, 747, 740, 733, 727,
721, 715, 710, 705, 700, 695, 691, 687, 683, 679, 676, 672, 669, 666,
663, 660, 658, 655, 653, 651, 649, 647, 645, 643, 641, 640, 639, 638,
637, 636, 635, 634, 634, 634, 634, 634, 634};
```

Array_valores_derivación_III

```
const int16 Der_avr[543] ={3602, 3596, 3589, 3582, 3575, 3568, 3560,
3553, 3545, 3538, 3530, 3522, 3514, 3506, 3498, 3490, 3482, 3474,
3465, 3457, 3450, 3442, 3434, 3427, 3420, 3413, 3406, 3400, 3394,
3388, 3383, 3378, 3373, 3369, 3366, 3362, 3359, 3357, 3355, 3353,
3352, 3351, 3351, 3351, 3351, 3352, 3354, 3355, 3357, 3360, 3363,
3366, 3370, 3375, 3379, 3385, 3391, 3397, 3404, 3411, 3419, 3427,
3436, 3445, 3455, 3465, 3476, 3487, 3499, 3510, 3522, 3535, 3547,
3560, 3573, 3585, 3598, 3610, 3623, 3635, 3646, 3657, 3668, 3678,
3687, 3696, 3704, 3712, 3718, 3724, 3729, 3733, 3736, 3739, 3741,
3742, 3743, 3743, 3743, 3743, 3742, 3742, 3742, 3742, 3742, 3744,
3746, 3749, 3753, 3758, 3765, 3773, 3782, 3793, 3806, 3820, 3835,
3852, 3871, 3890, 3911, 3932, 3953, 3975, 3996, 4016, 4035, 4053,
4068, 4080, 4089, 4094, 4095, 4090, 4079, 4063, 4039, 4008, 3970,
```

```
3923, 3867, 3803, 3730, 3647, 3556, 3455, 3346, 3228, 3102, 2968,
2828, 2681, 2528, 2371, 2211, 2048, 1884, 1719, 1556, 1395, 1237,
1084, 937, 798, 667, 545, 434, 334, 246, 170, 108, 60, 26, 6, 0, 7,
29, 64, 112, 173, 245, 328, 421, 523, 633, 750, 874, 1002, 1134, 1270,
1407, 1545, 1683, 1820, 1956, 2089, 2218, 2344, 2465, 2581, 2692,
2797, 2896, 2989, 3075, 3155, 3229, 3296, 3357, 3411, 3460, 3504,
3541, 3574, 3602, 3626, 3645, 3661, 3674, 3683, 3690, 3695, 3697,
3698, 3698, 3696, 3694, 3691, 3687, 3684, 3680, 3676, 3673, 3670,
3667, 3664, 3663, 3661, 3660, 3660, 3660, 3661, 3662, 3664, 3666,
3668, 3671, 3673, 3676, 3680, 3683, 3686, 3689, 3692, 3695, 3698,
3701, 3703, 3705, 3707, 3709, 3711, 3712, 3713, 3713, 3714, 3714,
3714, 3714, 3713, 3713, 3712, 3711, 3709, 3708, 3707, 3705, 3704,
3702, 3700, 3698, 3697, 3695, 3693, 3692, 3690, 3688, 3686, 3685,
3683, 3682, 3680, 3679, 3678, 3676, 3675, 3674, 3672, 3671, 3670,
3669, 3668, 3666, 3665, 3664, 3663, 3661, 3660, 3658, 3657, 3655,
3653, 3651, 3649, 3647, 3645, 3643, 3640, 3638, 3635, 3632, 3629,
3626, 3623, 3620, 3617, 3613, 3610, 3606, 3602, 3598, 3594, 3590,
3586, 3582, 3577, 3573, 3568, 3564, 3559, 3555, 3550, 3545, 3540,
3535, 3530, 3525, 3520, 3515, 3510, 3504, 3499, 3493, 3488, 3482,
3477, 3471, 3465, 3460, 3454, 3448, 3441, 3435, 3429, 3422, 3415,
3409, 3401, 3394, 3387, 3379, 3371, 3363, 3355, 3346, 3337, 3328,
3319, 3310, 3300, 3290, 3279, 3269, 3258, 3247, 3236, 3224, 3212,
3200, 3188, 3176, 3163, 3151, 3138, 3125, 3111, 3098, 3085, 3071,
3057, 3043, 3029, 3015, 3001, 2987, 2972, 2958, 2943, 2928, 2913,
2898, 2884, 2869, 2854, 2839, 2824, 2809, 2794, 2779, 2764, 2750,
2735, 2721, 2707, 2693, 2679, 2666, 2653, 2640, 2628, 2617, 2605,
2595, 2585, 2576, 2567, 2559, 2552, 2545, 2540, 2535, 2532, 2529,
2527, 2527, 2527, 2529, 2531, 2535, 2540, 2546, 2553, 2561, 2571,
2581, 2593, 2605, 2619, 2634, 2650, 2666, 2684, 2702, 2721, 2741,
2762, 2783, 2804, 2826, 2849, 2872, 2895, 2918, 2942, 2965, 2989,
3012, 3035, 3058, 3081, 3104, 3126, 3148, 3169, 3190, 3211, 3230,
3250, 3268, 3287, 3304, 3321, 3337, 3353, 3367, 3382, 3395, 3408,
3420, 3432, 3443, 3453, 3463, 3472, 3481, 3489, 3497, 3504, 3510,
3517, 3523, 3528, 3533, 3538, 3542, 3547, 3550, 3554, 3557, 3561,
3564, 3566, 3569, 3571, 3574, 3576, 3578, 3580, 3582, 3584, 3585,
3587, 3589, 3590, 3592, 3593, 3595, 3596, 3597, 3599, 3600, 3601,
3602};
```

Array_valores_derivación_avr

```
const int16 Der_avl[543] ={2797, 2789, 2780, 2772, 2764, 2756, 2748,
2740, 2731, 2724, 2716, 2709, 2702, 2695, 2689, 2684, 2679, 2676,
2673, 2671, 2671, 2672, 2674, 2677, 2682, 2688, 2696, 2706, 2717,
2729, 2744, 2759, 2776, 2795, 2815, 2835, 2857, 2880, 2903, 2927,
2952, 2977, 3002, 3026, 3051, 3075, 3099, 3122, 3144, 3165, 3184,
3203, 3220, 3236, 3250, 3263, 3274, 3283, 3291, 3297, 3302, 3305,
3306, 3306, 3305, 3302, 3299, 3294, 3289, 3283, 3276, 3269, 3261,
3254, 3246, 3238, 3231, 3224, 3217, 3210, 3205, 3199, 3195, 3191,
3188, 3185, 3183, 3182, 3181, 3181, 3181, 3182, 3183, 3184, 3186,
3187, 3189, 3190, 3191, 3191, 3191, 3191, 3190, 3189, 3186, 3183,
3180, 3175, 3170, 3165, 3159, 3152, 3146, 3139, 3132, 3125, 3118,
3112, 3107, 3103, 3100, 3099, 3099, 3101, 3105, 3112, 3121, 3133,
3148, 3166, 3188, 3212, 3240, 3272, 3306, 3344, 3384, 3428, 3473,
3521, 3571, 3622, 3674, 3725, 3777, 3827, 3875, 3921, 3963, 4001,
4033, 4060, 4080, 4091, 4095, 4088, 4072, 4045, 4007, 3957, 3896,
3822, 3736, 3638, 3527, 3406, 3273, 3130, 2978, 2817, 2648, 2474,
2295, 2113, 1929, 1746, 1564, 1385, 1211, 1044, 885, 735, 597, 471,
358, 260, 176, 109, 57, 22, 2, 0, 12, 40, 83, 140, 209, 291, 383, 485,
595, 712, 834, 962, 1092, 1224, 1357, 1490, 1621, 1749, 1874, 1995,
2111, 2221, 2326, 2424, 2515, 2599, 2675, 2745, 2808, 2863, 2912,
2954, 2990, 3021, 3045, 3065, 3079, 3090, 3097, 3100, 3101, 3099,
3095, 3090, 3083, 3076, 3068, 3059, 3051, 3043, 3035, 3028, 3022,
```

```
3016, 3012, 3008, 3006, 3004, 3004, 3004, 3005, 3007, 3010, 3013,
3018, 3022, 3027, 3032, 3038, 3043, 3049, 3054, 3060, 3065, 3070,
3074, 3078, 3082, 3085, 3088, 3090, 3092, 3093, 3094, 3094, 3094,
3093, 3092, 3091, 3090, 3088, 3086, 3084, 3081, 3079, 3076, 3074,
3071, 3069, 3067, 3064, 3062, 3060, 3059, 3057, 3055, 3054, 3053,
3052, 3051, 3051, 3050, 3050, 3050, 3050, 3050, 3050, 3050, 3051,
3051, 3051, 3051, 3051, 3051, 3051, 3050, 3050, 3049, 3048, 3047,
3046, 3045, 3043, 3042, 3040, 3038, 3035, 3033, 3031, 3028, 3025,
3023, 3020, 3017, 3014, 3011, 3009, 3006, 3003, 3001, 2998, 2996,
2993, 2991, 2989, 2987, 2985, 2984, 2982, 2980, 2979, 2977, 2976,
2974, 2973, 2971, 2970, 2968, 2966, 2964, 2962, 2960, 2958, 2956,
2953, 2950, 2947, 2944, 2940, 2937, 2933, 2929, 2924, 2920, 2915,
2910, 2905, 2900, 2895, 2889, 2884, 2878, 2872, 2865, 2859, 2852,
2846, 2839, 2832, 2825, 2818, 2810, 2803, 2795, 2788, 2780, 2772,
2764, 2756, 2747, 2739, 2730, 2722, 2713, 2704, 2696, 2687, 2678,
2669, 2660, 2650, 2641, 2632, 2622, 2613, 2603, 2593, 2584, 2574,
2564, 2555, 2545, 2535, 2526, 2516, 2507, 2497, 2488, 2479, 2470,
2462, 2453, 2445, 2438, 2430, 2423, 2417, 2411, 2406, 2401, 2396,
2393, 2390, 2387, 2385, 2384, 2384, 2385, 2386, 2388, 2390, 2393,
2397, 2402, 2407, 2414, 2420, 2427, 2435, 2444, 2452, 2462, 2471,
2481, 2492, 2502, 2513, 2524, 2535, 2546, 2557, 2568, 2579, 2590,
2601, 2611, 2622, 2632, 2641, 2650, 2659, 2668, 2676, 2684, 2691,
2698, 2705, 2711, 2717, 2722, 2727, 2732, 2736, 2740, 2744, 2747,
2750, 2753, 2756, 2758, 2760, 2763, 2765, 2767, 2768, 2770, 2772,
2774, 2775, 2777, 2778, 2780, 2782, 2783, 2785, 2786, 2788, 2790,
2791, 2793, 2794, 2795, 2797, 2798, 2799, 2800, 2801, 2802, 2803,
2803, 2803, 2804, 2804, 2803, 2803, 2802, 2802, 2801, 2800, 2798,
2797};
```

Array_valores_derivación_avl

```
const int16 Der_avf[543] ={594, 603, 612, 621, 630, 639, 648, 657,
667, 676, 686, 695, 704, 713, 722, 731, 740, 748, 755, 762, 769, 775,
780, 785, 789, 792, 794, 795, 795, 795, 793, 791, 787, 783, 778, 772,
766, 758, 750, 742, 733, 723, 714, 703, 693, 682, 672, 661, 650, 640,
629, 619, 609, 599, 589, 580, 570, 561, 553, 544, 536, 528, 521, 513,
506, 498, 491, 484, 477, 470, 463, 456, 449, 442, 435, 428, 421, 414,
407, 400, 393, 386, 379, 373, 366, 360, 355, 349, 344, 340, 336, 332,
329, 326, 324, 323, 321, 321, 320, 320, 321, 321, 322, 322, 323, 323,
323, 322, 321, 319, 316, 312, 307, 301, 294, 285, 275, 264, 251, 237,
222, 206, 188, 170, 151, 132, 113, 94, 75, 58, 42, 28, 16, 7, 1, 0, 2,
9, 22, 40, 65, 96, 134, 180, 233, 293, 362, 439, 524, 616, 717, 825,
940, 1062, 1191, 1325, 1465, 1609, 1757, 1907, 2060, 2214, 2368, 2521,
2672, 2820, 2965, 3104, 3238, 3364, 3483, 3594, 3695, 3786, 3866,
3934, 3991, 4036, 4068, 4088, 4095, 4089, 4070, 4039, 3996, 3942,
3876, 3800, 3714, 3619, 3516, 3405, 3288, 3166, 3039, 2908, 2775,
2640, 2504, 2369, 2234, 2101, 1971, 1844, 1721, 1602, 1488, 1379,
1277, 1180, 1089, 1005, 927, 855, 790, 730, 677, 630, 589, 552, 521,
495, 473, 455, 440, 429, 421, 416, 412, 411, 411, 413, 415, 419, 422,
426, 431, 435, 439, 442, 446, 448, 451, 452, 453, 454, 453, 453, 451,
450, 447, 445, 441, 438, 435, 431, 427, 423, 420, 416, 412, 409, 406,
403, 400, 398, 395, 394, 392, 391, 391, 390, 390, 391, 391, 392, 393,
395, 396, 398, 400, 402, 404, 406, 409, 411, 414, 416, 418, 421, 423,
425, 427, 429, 431, 433, 434, 436, 438, 439, 440, 442, 443, 444, 445,
446, 448, 449, 450, 451, 452, 454, 455, 457, 458, 460, 462, 464, 466,
469, 471, 474, 477, 480, 483, 487, 490, 494, 498, 502, 506, 511, 515,
519, 524, 529, 533, 538, 543, 548, 553, 558, 563, 567, 572, 577, 582,
587, 592, 597, 602, 607, 612, 617, 622, 627, 633, 638, 643, 649, 654,
660, 666, 672, 678, 685, 692, 698, 706, 713, 721, 728, 737, 745, 754,
763, 772, 781, 791, 801, 812, 822, 833, 845, 856, 868, 880, 892, 904,
917, 930, 943, 956, 969, 983, 997, 1011, 1025, 1039, 1054, 1068, 1083,
1098, 1113, 1128, 1143, 1158, 1174, 1189, 1205, 1221, 1236, 1252,
```

```
1268, 1284, 1300, 1316, 1332, 1348, 1364, 1380, 1396, 1411, 1427,
1442, 1457, 1471, 1486, 1500, 1513, 1527, 1539, 1551, 1563, 1574,
1584, 1593, 1602, 1610, 1617, 1623, 1628, 1632, 1634, 1636, 1637,
1636, 1635, 1632, 1628, 1622, 1616, 1608, 1599, 1589, 1578, 1565,
1552, 1537, 1522, 1505, 1487, 1469, 1450, 1430, 1409, 1388, 1366,
1344, 1321, 1298, 1275, 1251, 1228, 1204, 1181, 1158, 1135, 1112,
1089, 1067, 1045, 1024, 1003, 983, 963, 944, 926, 908, 891, 874, 858,
843, 829, 815, 802, 790, 778, 767, 756, 746, 737, 728, 720, 712, 704,
697, 691, 685, 679, 674, 669, 664, 659, 655, 651, 647, 644, 641, 637,
634, 632, 629, 626, 624, 622, 620, 618, 616, 614, 612, 611, 610, 608,
607, 606, 605, 604, 603, 603, 602, 602, 602};
```

Array_valores_derivación_avf

```
const int16 Der_v1[543] ={3797, 3798, 3797, 3797, 3795, 3794, 3791,
3788, 3785, 3781, 3776, 3771, 3765, 3759, 3752, 3745, 3737, 3729,
3721, 3712, 3703, 3694, 3685, 3675, 3666, 3656, 3647, 3638, 3629,
3620, 3612, 3604, 3596, 3589, 3582, 3576, 3570, 3565, 3560, 3556,
3552, 3549, 3547, 3545, 3544, 3543, 3543, 3543, 3544, 3545, 3546,
3549, 3551, 3554, 3557, 3561, 3565, 3569, 3574, 3579, 3584, 3589,
3595, 3601, 3607, 3614, 3620, 3627, 3634, 3641, 3648, 3655, 3662,
3669, 3676, 3683, 3690, 3697, 3703, 3709, 3715, 3720, 3725, 3729,
3733, 3736, 3739, 3741, 3742, 3743, 3743, 3742, 3740, 3738, 3736,
3732, 3728, 3724, 3719, 3714, 3708, 3703, 3697, 3692, 3686, 3681,
3677, 3673, 3670, 3668, 3667, 3667, 3669, 3672, 3677, 3684, 3692,
3702, 3714, 3728, 3744, 3762, 3782, 3803, 3825, 3849, 3874, 3899,
3924, 3950, 3975, 3999, 4021, 4041, 4059, 4074, 4085, 4092, 4095,
4091, 4082, 4066, 4043, 4013, 3975, 3928, 3873, 3810, 3737, 3655,
3565, 3466, 3358, 3242, 3118, 2987, 2849, 2705, 2556, 2403, 2246,
2087, 1927, 1766, 1607, 1449, 1294, 1143, 998, 859, 728, 605, 492,
388, 296, 215, 146, 90, 48, 18, 2, 0, 10, 35, 72, 122, 184, 258, 342,
437, 541, 654, 774, 900, 1032, 1169, 1309, 1452, 1596, 1741, 1885,
2028, 2169, 2306, 2440, 2570, 2695, 2814, 2927, 3034, 3134, 3228,
3315, 3394, 3467, 3533, 3592, 3645, 3691, 3731, 3766, 3795, 3819,
3839, 3854, 3866, 3874, 3879, 3881, 3881, 3880, 3877, 3873, 3867,
3862, 3856, 3850, 3844, 3838, 3833, 3828, 3824, 3821, 3819, 3817,
3816, 3816, 3817, 3819, 3821, 3824, 3828, 3832, 3837, 3842, 3847,
3852, 3858, 3864, 3870, 3876, 3881, 3887, 3892, 3897, 3902, 3907,
3911, 3915, 3919, 3922, 3925, 3927, 3929, 3931, 3933, 3934, 3935,
3936, 3936, 3937, 3937, 3937, 3937, 3937, 3936, 3936, 3935, 3935,
3935, 3934, 3934, 3933, 3933, 3933, 3933, 3933, 3933, 3933, 3933,
3933, 3933, 3934, 3934, 3935, 3935, 3936, 3937, 3938, 3938, 3939,
3940, 3941, 3942, 3943, 3944, 3945, 3946, 3947, 3948, 3949, 3950,
3951, 3952, 3953, 3954, 3955, 3955, 3956, 3957, 3958, 3959, 3960,
3961, 3962, 3963, 3963, 3964, 3965, 3966, 3967, 3968, 3970, 3971,
3972, 3973, 3974, 3976, 3977, 3978, 3980, 3981, 3982, 3984, 3985,
3987, 3988, 3990, 3991, 3993, 3994, 3996, 3997, 3999, 4000, 4002,
4003, 4004, 4006, 4007, 4008, 4009, 4011, 4012, 4013, 4014, 4014,
4015, 4016, 4017, 4017, 4018, 4018, 4019, 4019, 4019, 4019, 4019,
4019, 4019, 4018, 4018, 4017, 4017, 4016, 4015, 4014, 4012, 4011,
4009, 4007, 4006, 4004, 4001, 3999, 3996, 3994, 3991, 3988, 3984,
3981, 3977, 3973, 3969, 3965, 3961, 3956, 3952, 3947, 3941, 3936,
3931, 3925, 3919, 3913, 3907, 3901, 3894, 3888, 3881, 3874, 3867,
3860, 3852, 3845, 3838, 3830, 3822, 3815, 3807, 3800, 3792, 3785,
3777, 3770, 3763, 3756, 3749, 3742, 3736, 3730, 3724, 3718, 3713,
3708, 3703, 3699, 3695, 3691, 3688, 3685, 3683, 3681, 3679, 3678,
3677, 3677, 3677, 3677, 3678, 3679, 3681, 3682, 3684, 3687, 3689,
3692, 3695, 3698, 3702, 3705, 3709, 3712, 3716, 3720, 3724, 3728,
3732, 3736, 3740, 3743, 3747, 3751, 3754, 3758, 3761, 3764, 3768,
3771, 3773, 3776, 3779, 3781, 3783, 3786, 3788, 3789, 3791, 3793,
3794, 3795, 3796, 3797, 3798, 3799, 3800, 3800, 3801, 3801, 3802,
3802, 3802, 3802, 3802, 3802, 3802, 3802, 3802, 3802, 3802, 3802,
```

```
3802, 3801, 3801, 3801, 3800, 3800, 3800, 3799, 3799, 3798, 3798,
3797};
```

Array_valores_derivación_v1

```
const int16 Der_v2[543] ={3019, 3020, 3021, 3022, 3022, 3023, 3023,
3022, 3022, 3021, 3020, 3019, 3017, 3016, 3014, 3012, 3009, 3007,
3005, 3002, 2999, 2997, 2994, 2991, 2989, 2986, 2984, 2981, 2979,
2977, 2976, 2974, 2973, 2972, 2971, 2970, 2970, 2970, 2970, 2971,
2971, 2972, 2973, 2975, 2976, 2978, 2979, 2981, 2983, 2984, 2986,
2988, 2990, 2992, 2993, 2995, 2996, 2997, 2999, 3000, 3001, 3001,
3002, 3002, 3003, 3003, 3003, 3003, 3003, 3003, 3002, 3002, 3002,
3001, 3001, 3000, 3000, 2999, 2999, 2998, 2997, 2997, 2996, 2995,
2994, 2993, 2992, 2991, 2989, 2988, 2985, 2983, 2980, 2977, 2973,
2969, 2964, 2959, 2953, 2946, 2939, 2932, 2924, 2915, 2907, 2897,
2888, 2878, 2868, 2859, 2849, 2840, 2832, 2824, 2818, 2812, 2808,
2806, 2805, 2807, 2810, 2817, 2826, 2838, 2852, 2870, 2892, 2916,
2944, 2975, 3010, 3047, 3088, 3131, 3177, 3224, 3274, 3325, 3376,
3428, 3479, 3529, 3577, 3623, 3666, 3704, 3738, 3766, 3788, 3803,
3810, 3808, 3798, 3778, 3748, 3707, 3656, 3594, 3522, 3438, 3344,
3239, 3124, 3000, 2867, 2727, 2579, 2425, 2266, 2104, 1939, 1773,
1606, 1442, 1280, 1123, 972, 827, 691, 564, 449, 344, 253, 174, 110,
60, 25, 5, 0, 9, 33, 71, 122, 186, 261, 348, 444, 549, 662, 781, 905,
1034, 1165, 1297, 1431, 1563, 1694, 1823, 1948, 2069, 2185, 2295,
2399, 2497, 2588, 2673, 2750, 2820, 2882, 2938, 2987, 3029, 3066,
3096, 3120, 3139, 3154, 3164, 3170, 3173, 3173, 3171, 3166, 3160,
3152, 3143, 3134, 3125, 3115, 3106, 3097, 3089, 3082, 3075, 3070,
3066, 3062, 3060, 3059, 3059, 3061, 3063, 3066, 3070, 3075, 3080,
3086, 3092, 3099, 3107, 3114, 3121, 3129, 3137, 3144, 3151, 3158,
3165, 3171, 3178, 3183, 3189, 3194, 3198, 3202, 3206, 3210, 3213,
3215, 3218, 3220, 3222, 3224, 3225, 3226, 3228, 3229, 3230, 3231,
3232, 3233, 3234, 3235, 3236, 3237, 3238, 3240, 3241, 3243, 3245,
3247, 3249, 3251, 3254, 3256, 3259, 3262, 3265, 3268, 3271, 3275,
3278, 3282, 3285, 3289, 3293, 3297, 3301, 3304, 3308, 3313, 3317,
3321, 3325, 3329, 3333, 3338, 3342, 3347, 3351, 3356, 3360, 3365,
3370, 3375, 3380, 3385, 3390, 3395, 3401, 3406, 3412, 3418, 3424,
3430, 3436, 3443, 3449, 3456, 3463, 3470, 3478, 3485, 3493, 3501,
3509, 3517, 3525, 3534, 3543, 3551, 3561, 3570, 3579, 3588, 3598,
3608, 3618, 3628, 3638, 3648, 3659, 3669, 3680, 3691, 3702, 3713,
3724, 3735, 3746, 3757, 3769, 3780, 3791, 3803, 3814, 3825, 3837,
3848, 3859, 3871, 3882, 3893, 3904, 3915, 3925, 3936, 3946, 3956,
3966, 3976, 3985, 3994, 4003, 4012, 4020, 4028, 4035, 4042, 4049,
4055, 4061, 4067, 4072, 4077, 4081, 4084, 4087, 4090, 4092, 4093,
4094, 4095, 4094, 4093, 4092, 4090, 4087, 4083, 4079, 4074, 4069,
4063, 4056, 4049, 4041, 4032, 4023, 4013, 4002, 3991, 3979, 3967,
3954, 3940, 3926, 3912, 3897, 3881, 3865, 3849, 3832, 3815, 3797,
3780, 3762, 3744, 3725, 3707, 3688, 3670, 3651, 3632, 3614, 3595,
3577, 3558, 3540, 3522, 3504, 3487, 3469, 3453, 3436, 3419, 3403,
3388, 3373, 3358, 3343, 3329, 3315, 3302, 3289, 3277, 3265, 3253,
3242, 3231, 3221, 3211, 3201, 3192, 3183, 3175, 3167, 3159, 3152,
3144, 3138, 3131, 3125, 3119, 3114, 3108, 3103, 3098, 3094, 3089,
3085, 3081, 3078, 3074, 3071, 3067, 3064, 3062, 3059, 3056, 3054,
3051, 3049, 3047, 3045, 3043, 3041, 3040, 3038, 3036, 3035, 3034,
3032, 3031, 3029, 3028, 3027, 3026, 3025, 3023, 3022, 3021, 3020,
3019};
```

Array_valores_derivación_v2

```
const int16 Der_v3[543] ={2265, 2267, 2269, 2271, 2273, 2275, 2276,
2278, 2279, 2280, 2281, 2281, 2282, 2282, 2283, 2283, 2283, 2283,
2283, 2282, 2282, 2281, 2281, 2280, 2279, 2279, 2278, 2277, 2276,
2276, 2275, 2274, 2274, 2273, 2273, 2272, 2272, 2272, 2272, 2272,
2272, 2272, 2272, 2272, 2273, 2273, 2274, 2274, 2274, 2275, 2275,
```

```
2275, 2275, 2275, 2275, 2275, 2274, 2274, 2273, 2272, 2271, 2270,
2269, 2267, 2265, 2263, 2261, 2259, 2257, 2255, 2253, 2251, 2248,
2246, 2244, 2242, 2240, 2238, 2237, 2235, 2234, 2233, 2232, 2232,
2231, 2231, 2230, 2230, 2230, 2230, 2230, 2230, 2229, 2228, 2227,
2226, 2224, 2221, 2218, 2214, 2209, 2203, 2196, 2189, 2180, 2171,
2160, 2149, 2137, 2124, 2110, 2096, 2081, 2066, 2051, 2036, 2022,
2009, 1996, 1985, 1977, 1970, 1966, 1965, 1967, 1973, 1983, 1998,
2018, 2043, 2073, 2109, 2151, 2199, 2253, 2314, 2379, 2451, 2528,
2611, 2698, 2789, 2883, 2980, 3080, 3180, 3280, 3380, 3477, 3572,
3662, 3747, 3825, 3895, 3957, 4008, 4048, 4077, 4092, 4095, 4082,
4055, 4014, 3957, 3884, 3797, 3695, 3580, 3450, 3309, 3156, 2992,
2820, 2641, 2456, 2267, 2076, 1884, 1694, 1506, 1324, 1148, 981, 823,
676, 542, 421, 314, 222, 146, 86, 41, 12, 0, 2, 19, 50, 94, 150, 217,
294, 380, 473, 572, 677, 785, 896, 1008, 1120, 1231, 1341, 1447, 1550,
1649, 1743, 1832, 1915, 1991, 2061, 2125, 2182, 2233, 2277, 2316,
2348, 2374, 2395, 2411, 2423, 2430, 2434, 2434, 2431, 2426, 2419,
2411, 2401, 2390, 2379, 2368, 2356, 2345, 2335, 2325, 2316, 2308,
2301, 2295, 2291, 2288, 2286, 2285, 2285, 2286, 2289, 2292, 2296,
2301, 2306, 2312, 2319, 2326, 2333, 2340, 2347, 2355, 2362, 2369,
2376, 2383, 2389, 2395, 2401, 2406, 2411, 2415, 2419, 2423, 2426,
2429, 2432, 2435, 2437, 2439, 2440, 2442, 2443, 2444, 2445, 2446,
2447, 2448, 2449, 2450, 2452, 2453, 2454, 2456, 2457, 2459, 2461,
2463, 2465, 2467, 2470, 2472, 2475, 2478, 2481, 2484, 2487, 2491,
2494, 2498, 2502, 2506, 2510, 2514, 2518, 2522, 2526, 2530, 2535,
2539, 2544, 2548, 2553, 2557, 2562, 2567, 2572, 2577, 2582, 2587,
2592, 2597, 2603, 2608, 2614, 2619, 2625, 2631, 2637, 2643, 2650,
2656, 2663, 2670, 2677, 2684, 2691, 2699, 2706, 2714, 2722, 2730,
2738, 2746, 2755, 2764, 2772, 2781, 2791, 2800, 2809, 2819, 2828,
2838, 2848, 2858, 2869, 2879, 2890, 2900, 2911, 2922, 2933, 2944,
2956, 2967, 2979, 2990, 3002, 3014, 3026, 3038, 3050, 3063, 3075,
3087, 3100, 3112, 3125, 3137, 3149, 3162, 3174, 3186, 3199, 3211,
3223, 3235, 3246, 3258, 3269, 3280, 3291, 3302, 3312, 3323, 3332,
3342, 3351, 3360, 3369, 3377, 3385, 3392, 3399, 3406, 3412, 3417,
3422, 3427, 3431, 3435, 3438, 3440, 3442, 3444, 3445, 3445, 3444,
3444, 3442, 3440, 3437, 3433, 3429, 3424, 3419, 3413, 3406, 3398,
3390, 3381, 3371, 3361, 3350, 3338, 3326, 3313, 3299, 3285, 3270,
3254, 3238, 3222, 3205, 3187, 3169, 3151, 3132, 3113, 3093, 3074,
3054, 3034, 3014, 2994, 2973, 2953, 2933, 2913, 2893, 2873, 2853,
2834, 2815, 2796, 2777, 2759, 2741, 2724, 2707, 2690, 2674, 2658,
2643, 2628, 2614, 2600, 2587, 2574, 2562, 2550, 2539, 2528, 2518,
2508, 2498, 2490, 2481, 2473, 2465, 2458, 2451, 2444, 2438, 2432,
2427, 2421, 2416, 2412, 2407, 2403, 2399, 2395, 2392, 2388, 2385,
2382, 2379, 2377, 2374, 2372, 2370, 2367, 2365, 2363, 2362, 2360,
2358, 2356, 2355, 2353, 2352, 2351, 2349, 2348, 2347, 2345, 2344};
```

Array_valores_derivación_v3

```
const int16 Der_v4[543] ={1016, 1018, 1019, 1021, 1022, 1024, 1025,
1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1035, 1036,
1037, 1037, 1038, 1038, 1038, 1039, 1039, 1039, 1040, 1040, 1040,
1040, 1041, 1041, 1041, 1041, 1042, 1042, 1042, 1043, 1043, 1044,
1044, 1045, 1046, 1046, 1047, 1047, 1048, 1048, 1049, 1049, 1050,
1050, 1050, 1050, 1050, 1049, 1048, 1047, 1046, 1045, 1043, 1041,
1039, 1036, 1034, 1031, 1028, 1024, 1021, 1017, 1013, 1009, 1005,
1001, 998, 994, 990, 987, 984, 981, 978, 976, 975, 973, 972, 972, 972,
972, 973, 975, 976, 978, 980, 983, 985, 987, 990, 992, 993, 994, 995,
994, 993, 991, 987, 982, 976, 968, 959, 948, 935, 921, 905, 888, 870,
850, 829, 807, 785, 763, 741, 719, 699, 680, 663, 649, 638, 631, 629,
631, 639, 653, 674, 703, 739, 783, 836, 898, 969, 1049, 1139, 1237,
1345, 1461, 1585, 1716, 1854, 1998, 2146, 2299, 2454, 2610, 2765,
2920, 3071, 3217, 3357, 3488, 3611, 3723, 3822, 3907, 3978, 4033,
4071, 4092, 4095, 4079, 4045, 3993, 3923, 3835, 3731, 3612, 3478,
```

```
3331, 3172, 3004, 2827, 2644, 2456, 2266, 2074, 1884, 1696, 1513,
1335, 1165, 1004, 853, 712, 583, 467, 363, 273, 196, 132, 81, 43, 17,
3, 0, 7, 23, 48, 80, 119, 164, 213, 266, 322, 379, 438, 497, 555, 612,
667, 720, 770, 817, 860, 900, 936, 968, 996, 1020, 1040, 1057, 1070,
1079, 1086, 1090, 1090, 1089, 1086, 1080, 1074, 1066, 1057, 1047,
1037, 1027, 1017, 1007, 997, 988, 980, 972, 965, 959, 953, 949, 946,
943, 941, 941, 941, 941, 943, 945, 948, 951, 954, 958, 962, 966, 971,
975, 980, 984, 988, 993, 997, 1000, 1004, 1007, 1010, 1013, 1015,
1017, 1019, 1021, 1022, 1024, 1024, 1025, 1026, 1026, 1027, 1027,
1027, 1027, 1027, 1027, 1028, 1028, 1028, 1028, 1028, 1029, 1029,
1030, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1038, 1039, 1041,
1043, 1044, 1046, 1048, 1050, 1052, 1054, 1057, 1059, 1061, 1064,
1066, 1069, 1071, 1074, 1076, 1079, 1082, 1085, 1088, 1090, 1093,
1096, 1100, 1103, 1106, 1109, 1113, 1116, 1120, 1123, 1127, 1131,
1135, 1139, 1143, 1147, 1152, 1156, 1161, 1166, 1170, 1175, 1180,
1185, 1190, 1196, 1201, 1207, 1212, 1218, 1224, 1230, 1236, 1242,
1248, 1255, 1261, 1268, 1275, 1282, 1289, 1296, 1304, 1311, 1319,
1327, 1335, 1344, 1352, 1361, 1369, 1378, 1387, 1397, 1406, 1416,
1425, 1435, 1445, 1455, 1465, 1476, 1486, 1496, 1507, 1517, 1528,
1539, 1549, 1560, 1571, 1581, 1592, 1602, 1613, 1623, 1634, 1644,
1654, 1664, 1674, 1683, 1693, 1702, 1711, 1720, 1729, 1738, 1746,
1754, 1762, 1769, 1777, 1784, 1790, 1797, 1803, 1808, 1814, 1818,
1823, 1827, 1830, 1834, 1836, 1838, 1840, 1841, 1842, 1841, 1841,
1839, 1838, 1835, 1832, 1828, 1823, 1818, 1812, 1806, 1798, 1790,
1782, 1772, 1762, 1752, 1741, 1729, 1717, 1704, 1691, 1677, 1663,
1648, 1633, 1618, 1602, 1586, 1570, 1554, 1538, 1522, 1505, 1489,
1473, 1457, 1440, 1424, 1409, 1393, 1378, 1363, 1348, 1333, 1319,
1305, 1292, 1279, 1266, 1253, 1241, 1230, 1219, 1208, 1197, 1187,
1178, 1169, 1160, 1151, 1143, 1135, 1128, 1121, 1114, 1108, 1102,
1097, 1091, 1086, 1081, 1077, 1072, 1068, 1065, 1061, 1058, 1054,
1051, 1049, 1046, 1043, 1041, 1039, 1037, 1035, 1033, 1031, 1030,
1028, 1027, 1025, 1024, 1023, 1021, 1020, 1019, 1018, 1017, 1016};
```

Array_valores_derivación_v4

```
const int16 Der_v5[543] ={418, 420, 422, 423, 425, 427, 428, 430, 432,
433, 435, 437, 438, 440, 442, 443, 445, 447, 449, 450, 452, 453, 455,
457, 458, 460, 461, 463, 464, 466, 467, 469, 470, 471, 473, 474, 476,
477, 479, 480, 482, 483, 485, 486, 488, 490, 491, 493, 494, 496, 497,
498, 499, 500, 501, 501, 501, 501, 501, 500, 499, 497, 495, 493, 490,
487, 484, 480, 476, 471, 466, 461, 456, 450, 444, 438, 432, 426, 420,
414, 409, 403, 398, 394, 390, 386, 383, 380, 378, 377, 376, 376, 377,
378, 380, 382, 384, 387, 390, 393, 396, 399, 402, 404, 405, 406, 406,
405, 402, 398, 392, 385, 376, 366, 353, 339, 323, 305, 285, 264, 242,
218, 194, 169, 144, 120, 96, 74, 53, 35, 20, 9, 2, 0, 3, 13, 29, 53,
85, 125, 174, 233, 301, 378, 465, 562, 669, 785, 910, 1043, 1184,
1333, 1487, 1647, 1812, 1979, 2148, 2318, 2488, 2655, 2819, 2978,
3131, 3277, 3414, 3541, 3656, 3760, 3851, 3929, 3992, 4040, 4073,
4092, 4095, 4082, 4056, 4014, 3960, 3892, 3811, 3720, 3618, 3506,
3387, 3260, 3127, 2990, 2849, 2705, 2560, 2414, 2270, 2127, 1987,
1850, 1717, 1589, 1467, 1350, 1240, 1137, 1040, 950, 868, 793, 724,
663, 608, 560, 518, 482, 451, 425, 404, 388, 375, 366, 360, 357, 355,
356, 358, 362, 366, 371, 376, 382, 387, 392, 397, 401, 405, 408, 410,
412, 413, 413, 412, 411, 409, 406, 403, 400, 396, 392, 388, 383, 378,
374, 369, 364, 360, 356, 351, 348, 344, 341, 338, 335, 333, 331, 330,
328, 327, 327, 326, 326, 327, 327, 328, 328, 329, 330, 332, 333, 334,
336, 337, 339, 340, 341, 343, 344, 346, 347, 348, 349, 350, 351, 352,
353, 354, 355, 356, 357, 357, 358, 359, 360, 360, 361, 362, 362, 363,
364, 364, 365, 366, 367, 367, 368, 369, 370, 371, 372, 373, 374, 375,
376, 377, 378, 379, 381, 382, 383, 385, 386, 387, 389, 390, 392, 394,
395, 397, 399, 401, 403, 405, 407, 409, 411, 414, 416, 419, 422, 424,
427, 430, 433, 436, 440, 443, 446, 450, 453, 457, 461, 465, 469, 473,
```

```
477, 481, 485, 490, 494, 499, 503, 508, 513, 518, 524, 529, 534, 540,
546, 552, 558, 564, 571, 577, 584, 591, 598, 606, 614, 621, 629, 638,
646, 655, 664, 673, 682, 691, 701, 710, 720, 730, 741, 751, 761, 772,
783, 794, 805, 816, 827, 838, 849, 860, 872, 883, 895, 906, 917, 929,
940, 951, 963, 974, 985, 996, 1007, 1018, 1028, 1038, 1048, 1058,
1068, 1077, 1086, 1095, 1103, 1110, 1118, 1124, 1131, 1136, 1142,
1146, 1150, 1153, 1155, 1157, 1158, 1158, 1157, 1156, 1154, 1151,
1147, 1142, 1136, 1130, 1123, 1115, 1106, 1097, 1087, 1076, 1064,
1052, 1040, 1026, 1013, 999, 984, 970, 954, 939, 924, 908, 892, 876,
861, 845, 829, 814, 798, 783, 768, 753, 738, 724, 710, 696, 683, 670,
657, 645, 633, 622, 611, 600, 590, 580, 570, 561, 553, 544, 536, 529,
522, 515, 508, 502, 496, 491, 485, 480, 476, 471, 467, 463, 459, 456,
453, 449, 447, 444, 441, 439, 437, 435, 433, 431, 430, 428, 427, 426,
425, 424, 423, 422, 422, 421, 420, 420, 419, 419, 419, 418};
```

Array_valores_derivación_v5

```
const int16 Der_v6[543] ={406, 408, 410, 411, 413, 415, 416, 418, 420,
421, 423, 424, 426, 427, 428, 430, 431, 432, 434, 435, 436, 437, 438,
439, 440, 441, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453,
454, 455, 456, 457, 459, 460, 461, 462, 464, 465, 466, 467, 468, 469,
470, 471, 471, 472, 472, 472, 471, 470, 469, 468, 466, 464, 462, 459,
456, 453, 449, 445, 441, 436, 432, 427, 422, 417, 412, 407, 402, 398,
393, 389, 385, 382, 379, 377, 375, 374, 373, 373, 374, 375, 376, 379,
381, 384, 388, 391, 395, 399, 402, 405, 408, 410, 411, 412, 411, 409,
405, 400, 393, 385, 374, 362, 348, 332, 314, 294, 273, 250, 226, 201,
176, 151, 125, 101, 78, 56, 37, 22, 10, 2, 0, 3, 13, 30, 54, 87, 129,
180, 241, 312, 393, 485, 587, 699, 820, 952, 1092, 1240, 1396, 1559,
1727, 1899, 2074, 2251, 2427, 2603, 2775, 2943, 3105, 3260, 3405,
3539, 3661, 3770, 3865, 3945, 4008, 4054, 4083, 4095, 4088, 4065,
4024, 3966, 3892, 3804, 3701, 3585, 3457, 3319, 3172, 3017, 2857,
2692, 2524, 2354, 2185, 2017, 1852, 1691, 1535, 1385, 1242, 1107, 980,
861, 753, 653, 563, 483, 413, 352, 300, 257, 222, 195, 176, 163, 156,
155, 159, 167, 178, 193, 210, 229, 249, 270, 291, 312, 333, 353, 372,
390, 406, 421, 434, 445, 454, 462, 468, 472, 474, 475, 475, 473, 470,
466, 461, 456, 450, 443, 436, 430, 423, 416, 409, 402, 396, 391, 385,
381, 376, 373, 370, 367, 365, 364, 363, 363, 363, 363, 364, 365, 367,
369, 371, 373, 375, 378, 380, 383, 386, 388, 391, 393, 395, 397, 400,
402, 403, 405, 407, 408, 409, 410, 412, 412, 413, 414, 415, 415, 416,
416, 417, 417, 418, 418, 419, 420, 420, 421, 421, 422, 423, 424, 425,
426, 427, 428, 429, 431, 432, 434, 435, 437, 438, 440, 442, 444, 446,
448, 450, 452, 454, 457, 459, 461, 464, 466, 469, 472, 474, 477, 480,
483, 486, 489, 492, 495, 498, 502, 505, 508, 512, 515, 519, 523, 527,
531, 535, 539, 543, 547, 551, 556, 560, 564, 569, 574, 579, 584, 589,
594, 599, 604, 610, 616, 621, 627, 633, 640, 646, 653, 659, 666, 673,
681, 688, 696, 704, 712, 720, 728, 737, 746, 755, 764, 774, 783, 793,
803, 813, 823, 833, 844, 854, 865, 876, 887, 898, 909, 920, 931, 942,
953, 964, 976, 987, 998, 1009, 1020, 1031, 1042, 1053, 1064, 1075,
1085, 1096, 1106, 1117, 1127, 1137, 1146, 1156, 1165, 1174, 1183,
1192, 1200, 1208, 1215, 1223, 1229, 1236, 1242, 1247, 1252, 1257,
1260, 1264, 1266, 1268, 1270, 1270, 1270, 1270, 1268, 1266, 1263,
1259, 1254, 1249, 1243, 1236, 1228, 1220, 1211, 1201, 1191, 1179,
1168, 1155, 1142, 1129, 1115, 1101, 1086, 1071, 1056, 1041, 1025,
1009, 993, 977, 961, 945, 929, 913, 898, 882, 867, 852, 837, 822, 808,
794, 781, 767, 755, 742, 730, 719, 707, 696, 686, 676, 666, 657, 649,
640, 632, 625, 617, 610, 604, 598, 592, 586, 581, 576, 572, 567, 563,
560, 556, 553, 550, 547, 544, 542, 539, 537, 535, 533, 532, 530, 529,
528, 527, 526, 525, 524, 523, 522, 522, 521, 521, 520, 520, 520, 519};
```

Array_valores_derivación_v6

```
const int16 Der_vx[543] ={409, 410, 411, 412, 413, 414, 414, 415, 416,
417, 417, 418, 419, 419, 420, 421, 422, 423, 423, 424, 425, 427, 428,
```

```
429, 430, 432, 434, 435, 437, 439, 442, 444, 447, 449, 452, 455, 458,
461, 464, 468, 471, 474, 478, 481, 485, 488, 492, 495, 498, 501, 503,
506, 508, 509, 511, 512, 512, 512, 512, 511, 510, 508, 505, 503, 499,
495, 491, 486, 481, 476, 470, 464, 458, 451, 445, 438, 432, 425, 419,
414, 408, 403, 398, 394, 391, 388, 386, 384, 383, 383, 383, 384, 386,
388, 390, 393, 396, 399, 402, 405, 407, 410, 411, 412, 411, 410, 407,
403, 397, 390, 381, 370, 357, 342, 325, 307, 286, 265, 241, 217, 192,
167, 142, 117, 93, 70, 49, 32, 17, 6, 0, 0, 5, 17, 36, 63, 99, 143,
198, 261, 335, 420, 514, 619, 734, 858, 992, 1135, 1286, 1443, 1608,
1777, 1950, 2126, 2303, 2479, 2654, 2826, 2992, 3152, 3304, 3446,
3577, 3696, 3801, 3891, 3966, 4024, 4065, 4089, 4095, 4083, 4053,
4007, 3943, 3864, 3770, 3662, 3541, 3408, 3266, 3114, 2956, 2792,
2624, 2454, 2282, 2111, 1942, 1776, 1614, 1458, 1308, 1166, 1031, 905,
789, 682, 584, 497, 419, 351, 293, 243, 203, 171, 146, 129, 119, 115,
116, 122, 132, 146, 162, 181, 201, 223, 245, 267, 289, 311, 331, 351,
369, 385, 400, 412, 423, 432, 440, 445, 448, 450, 450, 449, 447, 443,
438, 433, 426, 419, 412, 405, 397, 389, 382, 375, 368, 361, 355, 349,
344, 340, 336, 333, 330, 328, 327, 326, 325, 326, 326, 327, 329, 331,
333, 335, 338, 341, 343, 346, 349, 352, 355, 358, 361, 363, 366, 368,
370, 372, 374, 376, 377, 379, 380, 381, 382, 383, 383, 384, 384, 385,
385, 385, 385, 386, 386, 386, 386, 386, 387, 387, 387, 388, 388, 389,
389, 390, 391, 391, 392, 393, 395, 396, 397, 399, 400, 402, 403, 405,
407, 409, 411, 413, 416, 418, 420, 423, 425, 428, 431, 434, 436, 439,
442, 445, 449, 452, 455, 458, 462, 465, 469, 472, 476, 480, 483, 487,
491, 495, 499, 503, 507, 512, 516, 520, 525, 529, 534, 539, 544, 549,
554, 559, 565, 571, 576, 582, 588, 595, 601, 608, 615, 622, 629, 637,
644, 652, 660, 668, 677, 686, 694, 703, 713, 722, 732, 742, 752, 762,
772, 782, 793, 803, 814, 825, 836, 847, 858, 869, 880, 891, 903, 914,
925, 936, 948, 959, 970, 981, 992, 1003, 1014, 1024, 1035, 1046, 1056,
1066, 1076, 1086, 1096, 1106, 1115, 1124, 1133, 1141, 1150, 1158,
1166, 1173, 1180, 1187, 1193, 1199, 1204, 1209, 1213, 1217, 1220,
1223, 1225, 1226, 1227, 1227, 1226, 1225, 1222, 1219, 1216, 1211,
1206, 1199, 1193, 1185, 1176, 1167, 1157, 1146, 1135, 1123, 1110,
1097, 1083, 1068, 1053, 1038, 1022, 1006, 990, 973, 956, 939, 922,
905, 887, 870, 853, 836, 819, 802, 786, 770, 754, 738, 723, 708, 694,
680, 666, 653, 640, 627, 615, 604, 593, 582, 572, 562, 553, 544, 535,
527, 519, 512, 505, 498, 492, 486, 480, 475, 470, 465, 461, 457, 453,
449, 446, 443, 440, 437, 434, 432, 430, 428, 426, 424, 422, 421, 419,
418, 417, 416, 415, 414, 413, 412, 412, 411, 410, 410, 409};
```

Array_valores_derivación_vx

```
const int16 Der_vy[543] ={655, 664, 674, 684, 694, 704, 715, 725, 736,
746, 757, 768, 778, 789, 799, 809, 818, 828, 837, 845, 853, 860, 866,
872, 876, 880, 883, 885, 885, 885, 884, 882, 879, 875, 870, 864, 857,
850, 841, 833, 823, 814, 804, 793, 783, 772, 761, 751, 740, 729, 719,
709, 699, 689, 680, 671, 662, 654, 646, 638, 630, 622, 615, 607, 600,
592, 585, 578, 570, 562, 554, 546, 538, 529, 521, 512, 503, 494, 484,
475, 466, 457, 448, 439, 430, 422, 413, 406, 398, 392, 385, 380, 374,
370, 366, 362, 360, 357, 355, 354, 352, 351, 350, 349, 348, 347, 345,
343, 340, 336, 332, 326, 319, 312, 302, 292, 280, 267, 252, 237, 220,
202, 183, 163, 144, 123, 104, 84, 66, 49, 33, 20, 10, 3, 0, 0, 6, 17,
34, 56, 86, 122, 166, 217, 277, 344, 419, 503, 594, 694, 801, 915,
1037, 1165, 1299, 1438, 1582, 1730, 1881, 2033, 2188, 2342, 2495,
2647, 2796, 2940, 3081, 3215, 3342, 3462, 3574, 3676, 3768, 3850,
3920, 3979, 4027, 4062, 4084, 4095, 4092, 4078, 4052, 4014, 3965,
3906, 3836, 3756, 3668, 3572, 3469, 3359, 3244, 3124, 3001, 2874,
2746, 2617, 2487, 2358, 2231, 2105, 1982, 1862, 1746, 1634, 1527,
1425, 1328, 1236, 1151, 1071, 997, 929, 866, 809, 758, 712, 671, 635,
603, 576, 553, 533, 517, 504, 493, 485, 479, 475, 472, 471, 470, 471,
472, 473, 475, 476, 478, 479, 481, 482, 482, 483, 483, 482, 481, 480,
478, 476, 474, 471, 468, 465, 462, 459, 456, 453, 450, 447, 444, 442,
```

```
439, 437, 435, 434, 432, 431, 431, 430, 430, 430, 431, 432, 433, 434,
436, 437, 439, 441, 443, 446, 448, 450, 453, 455, 458, 460, 463, 465,
467, 469, 472, 474, 475, 477, 479, 480, 482, 483, 484, 486, 487, 488,
489, 490, 491, 492, 493, 494, 496, 497, 498, 500, 502, 504, 506, 508,
510, 513, 516, 519, 522, 525, 529, 533, 537, 541, 546, 550, 555, 560,
565, 570, 575, 580, 586, 591, 597, 603, 608, 614, 620, 626, 632, 637,
643, 649, 655, 661, 667, 673, 679, 685, 692, 698, 704, 711, 717, 724,
731, 738, 745, 753, 760, 768, 776, 785, 793, 802, 811, 821, 830, 840,
851, 861, 872, 883, 895, 907, 919, 931, 944, 957, 970, 984, 998, 1012,
1026, 1040, 1055, 1070, 1085, 1100, 1115, 1131, 1146, 1162, 1178,
1194, 1210, 1226, 1243, 1259, 1276, 1292, 1309, 1326, 1342, 1359,
1376, 1393, 1409, 1426, 1443, 1459, 1476, 1492, 1508, 1524, 1540,
1556, 1571, 1587, 1602, 1616, 1630, 1644, 1657, 1670, 1683, 1694,
1706, 1716, 1726, 1735, 1744, 1751, 1758, 1764, 1769, 1773, 1776,
1778, 1779, 1779, 1778, 1776, 1772, 1768, 1762, 1755, 1747, 1738,
1728, 1716, 1704, 1690, 1675, 1660, 1643, 1625, 1607, 1587, 1567,
1546, 1524, 1502, 1479, 1456, 1432, 1408, 1384, 1359, 1335, 1310,
1285, 1261, 1236, 1212, 1188, 1164, 1141, 1118, 1096, 1074, 1053,
1033, 1013, 993, 975, 957, 939, 923, 907, 892, 877, 864, 851, 838,
827, 816, 805, 796, 787, 778, 771, 763, 756, 750, 744, 739, 734, 729,
725, 721, 717, 713, 710, 707, 704, 701, 698, 696, 693, 690, 688, 686,
683, 681, 679, 676, 674, 672, 670, 668, 666, 664, 662, 660, 658, 656,
655};
```

Array_valores_derivación_vy

```
const int16 Der_vz[543] ={880, 876, 873, 871, 869, 867, 866, 865, 865,
865, 866, 867, 869, 871, 874, 878, 882, 886, 891, 896, 902, 908, 914,
920, 927, 934, 941, 948, 954, 961, 968, 975, 981, 987, 993, 999, 1004,
1009, 1014, 1018, 1022, 1025, 1028, 1031, 1033, 1035, 1037, 1038,
1039, 1040, 1041, 1041, 1041, 1041, 1040, 1040, 1039, 1038, 1037,
1036, 1034, 1033, 1031, 1030, 1028, 1026, 1024, 1021, 1019, 1017,
1014, 1011, 1009, 1006, 1003, 1000, 997, 994, 991, 988, 985, 983, 981,
978, 977, 975, 974, 973, 973, 974, 974, 976, 978, 981, 984, 989, 994,
999, 1006, 1013, 1021, 1029, 1038, 1048, 1058, 1068, 1078, 1089, 1099,
1110, 1120, 1129, 1138, 1146, 1153, 1159, 1163, 1165, 1165, 1163,
1159, 1152, 1143, 1130, 1115, 1096, 1074, 1049, 1021, 990, 956, 919,
879, 836, 792, 746, 698, 650, 601, 553, 505, 459, 415, 375, 337, 304,
277, 255, 240, 232, 232, 241, 258, 285, 323, 370, 428, 496, 575, 664,
764, 873, 992, 1119, 1255, 1398, 1547, 1702, 1861, 2024, 2188, 2353,
2517, 2680, 2839, 2994, 3143, 3284, 3418, 3541, 3655, 3757, 3846,
3923, 3986, 4034, 4069, 4089, 4095, 4086, 4063, 4026, 3976, 3914,
3840, 3755, 3660, 3556, 3445, 3326, 3203, 3074, 2943, 2809, 2675,
2540, 2407, 2275, 2146, 2022, 1901, 1786, 1676, 1572, 1475, 1385,
1301, 1224, 1155, 1092, 1036, 987, 945, 908, 878, 852, 832, 817, 806,
798, 794, 793, 794, 798, 803, 809, 817, 825, 833, 842, 850, 858, 865,
872, 878, 883, 887, 890, 893, 894, 894, 893, 891, 888, 885, 881, 876,
870, 865, 858, 852, 845, 838, 831, 824, 818, 811, 804, 798, 792, 786,
781, 775, 771, 766, 762, 758, 754, 751, 748, 745, 743, 741, 738, 737,
735, 733, 731, 730, 728, 727, 725, 724, 722, 720, 719, 717, 715, 713,
711, 708, 706, 704, 701, 698, 695, 693, 690, 687, 684, 680, 677, 674,
671, 668, 665, 661, 658, 655, 652, 649, 646, 643, 640, 637, 634, 631,
628, 625, 622, 619, 616, 613, 610, 607, 604, 600, 597, 594, 590, 587,
583, 579, 575, 571, 567, 562, 558, 553, 548, 543, 537, 532, 526, 520,
514, 508, 502, 495, 489, 482, 475, 467, 460, 452, 445, 437, 429, 421,
413, 405, 396, 388, 379, 370, 362, 353, 344, 335, 326, 317, 307, 298,
289, 279, 270, 261, 251, 242, 233, 224, 214, 205, 196, 187, 178, 169,
161, 152, 144, 136, 127, 120, 112, 104, 97, 90, 83, 77, 70, 64, 58,
52, 47, 42, 37, 32, 28, 24, 20, 16, 13, 10, 7, 5, 3, 2, 1, 0, 0, 0, 0,
1, 3, 5, 8, 11, 15, 19, 24, 30, 36, 43, 51, 59, 68, 77, 88, 98, 110,
122, 134, 147, 161, 175, 189, 204, 219, 235, 251, 267, 284, 300, 317,
334, 351, 367, 384, 401, 418, 434, 451, 467, 482, 498, 513, 528, 542,
```

```
557, 570, 583, 596, 609, 621, 632, 643, 654, 664, 674, 683, 692, 701,
709, 717, 725, 732, 739, 745, 752, 758, 763, 769, 774, 780, 785, 789,
794, 799, 803, 807, 811, 815, 819, 823, 827, 830, 834, 837, 840, 844,
847, 849, 852, 855, 857, 860, 862, 864, 866, 868, 870, 872, 873, 875,
876, 878, 879, 880, 881, 882, 882, 883, 884};
```

Array_valores_derivación_vz

3.4.3 - Generación de señal de EEG con Matlab

La explicación de funcionamiento del software final usado será explicada paso a paso para su adecuado entendimiento. Se parte del registro 'ch01_01_edfm' de la base de datos CAP Sleep Research, señal de EEG con una amplitud máxima de 1588 µV muestreada a 256 Hz y 16 bits, de 10s de duración (921600 muestras).

```
load('chb01_01_edfm');
der1= val();
plot(der1);
```

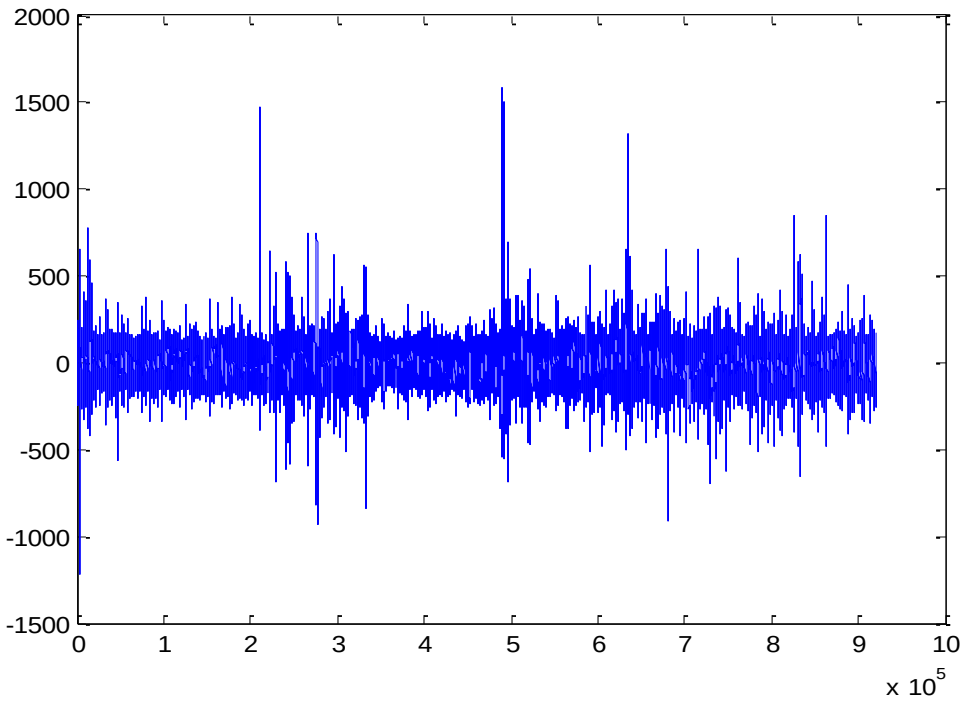


Fig. 3.28 - Señal de partida sin procesar

Como se puede comprobar en la imagen, la señal presenta una gran duración, además de poseer un contenido espectral muy elevado, a causa de ello, se hace preciso el procesado y filtrado de esta señal. Además, la frecuencia de muestreo es distinta (256 Hz) a la perseguida en la aplicación (1000 Hz), por lo que es necesario el remuestreo de la misma.

```
load('chb01_01_edfm');

fl=0.5;
fh=30;
Fs=1000;
fh_Norm = fl/(Fs/2);                    %# normalized cutoff frequency

der1=val();

[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');    %# 4th order filter
der1= resample(der1, 1000,256);

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

plot(der1);
```

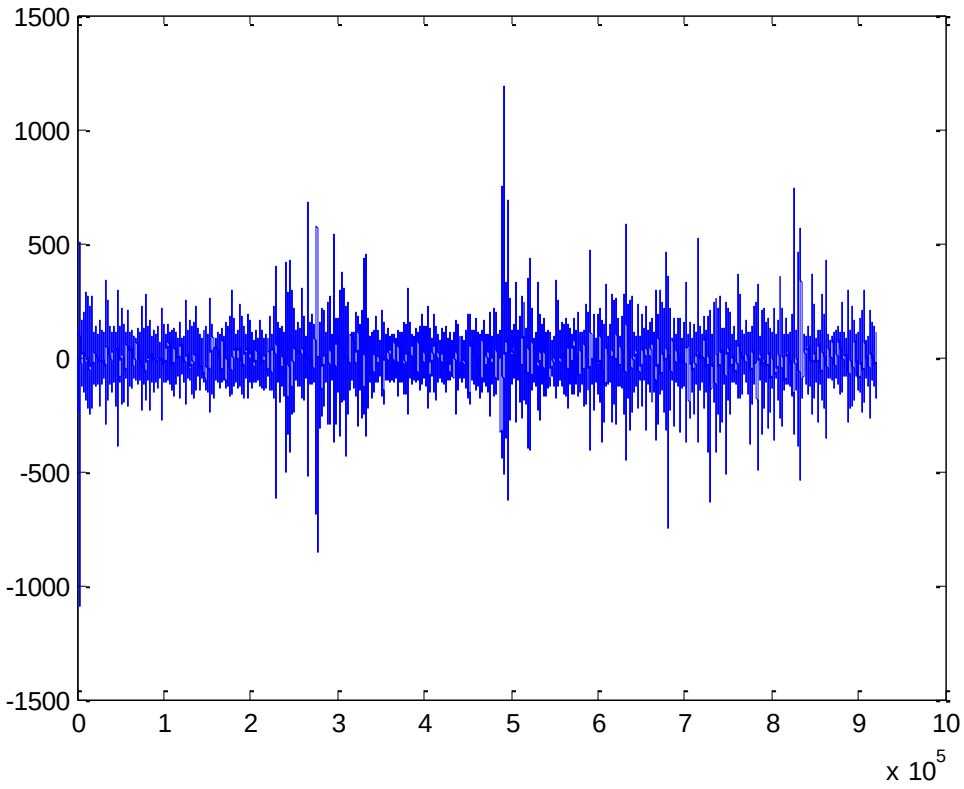


Fig. 3.29 - Señal filtrada

El filtrado se lleva a cabo mediante la aplicación sucesiva de un filtro pasa altos (0,5 Hz) y pasa bajos (30 Hz) implementados mediante aproximación Butterworth de orden 4. Con esto se consiguen eliminar los términos de alta frecuencia.

```
load('chb01_01_edfm');

fl=0.5;
fh=30;
```

```
Fs=1000;
fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency

der1= val();

[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter
der1= resample(der1, 1000,256);

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

der1=der1(1,1800000:1820000);
der1=der1(1,500:1499);
der1=der1(1:543);

plot(der1);
```

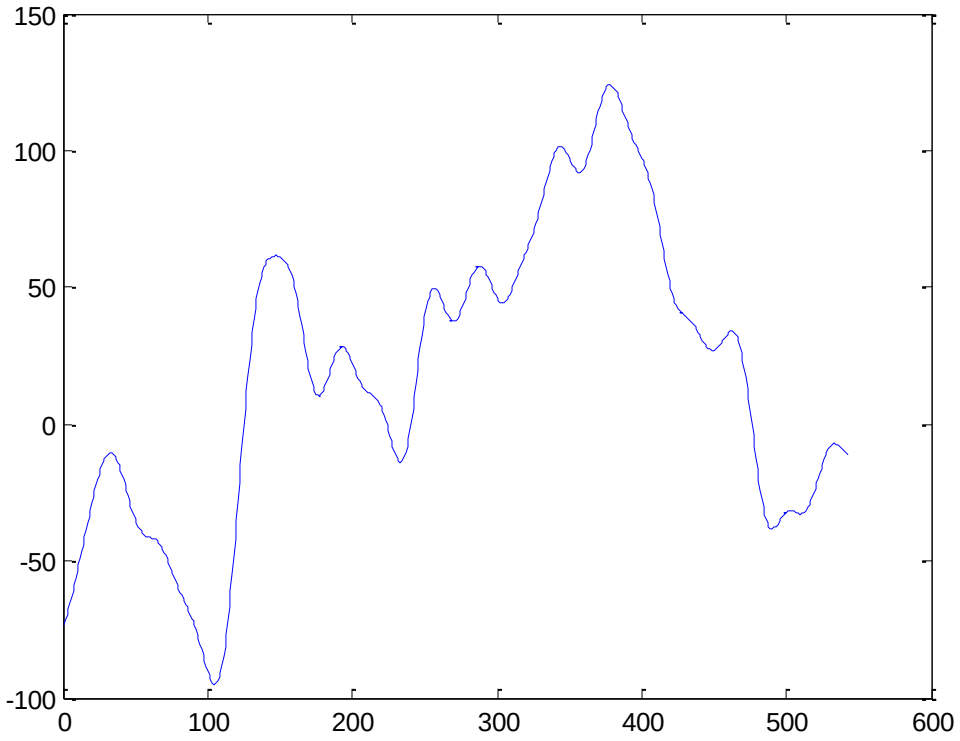


Fig. 3.30 - Fracción de señal seleccionada

Seguidamente, para poder seleccionar el tramo de señal a grabar como tabla, se implementa una rutina de navegación y extracción de señal. En principio se escoge una franja de la señal, seguidamente se aumenta la acotación, y por último se seleccionan los 547 valores que conforman la tabla de valores, al igual que la señal de electrocardiograma.

```
load('chb01_01_edfm');

fl=0.5;
fh=30;
Fs=1000;
fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency

der1= val();

[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter
der1= resample(der1, 1000,256);

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

der1=der1(1,1800000:1820000);
der1=der1(1,500:1499);
der1=der1(1:543);

ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

plot(der1);
```

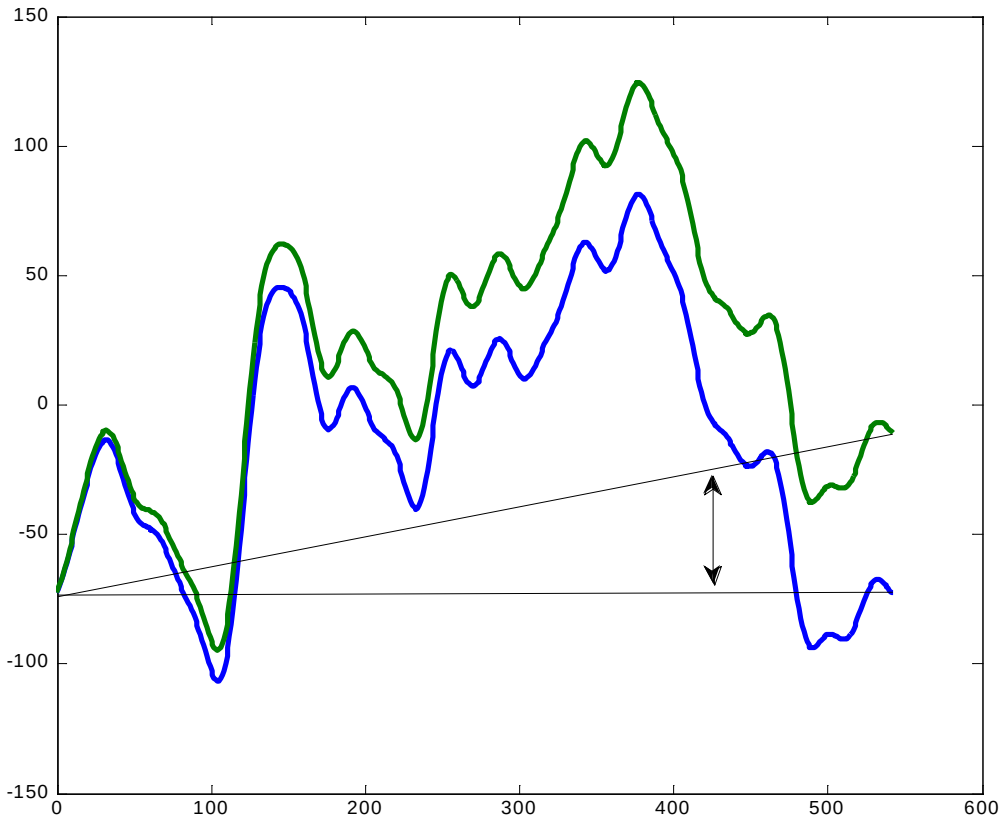


Fig. 3.31 - Detalle del proceso de restauración de la línea base

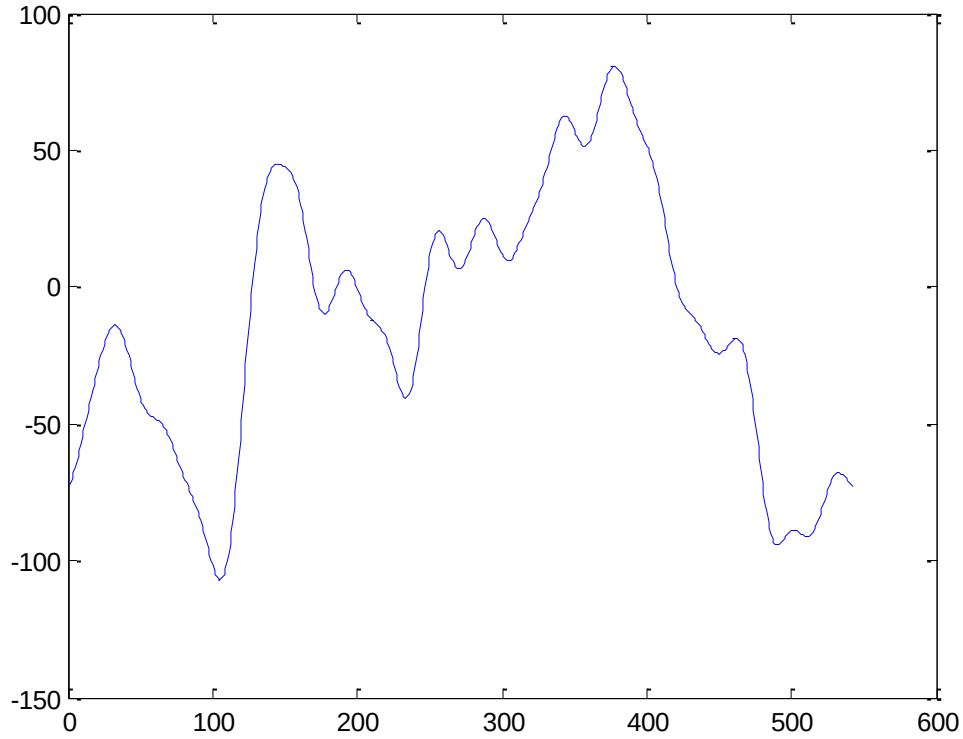


Fig. 3.32 - Fracción de señal seleccionada con línea base corregida

Acto seguido se implementa una subrutina encargada de hacer coincidir el primer y último valor de la señal. Esto se realiza trazando una recta desde el principio de la señal hasta su final y, seguidamente, restando cada valor de la señal a cada valor correspondiente de la recta. Se puede traducir como un giro de la señal, manteniendo fijo el primer valor y girando la señal hasta que el primer y último valor coincidan.

```
load('chb01_01_edfm');

fl=0.5;
fh=30;
Fs=1000;
fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency

[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter
der1= resample(der1, 1000,256);

der1= val();

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

der1=der1(1,1800000:1820000);
der1=der1(1,500:1499);
der1=der1(1:543);
```

```
ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

mmax=max(der1);
mmin=min(der1);
der1_ADC= fix(4095.*((der1 - mmin)/(mmax - mmin)));

plot(der1_ADC)
```

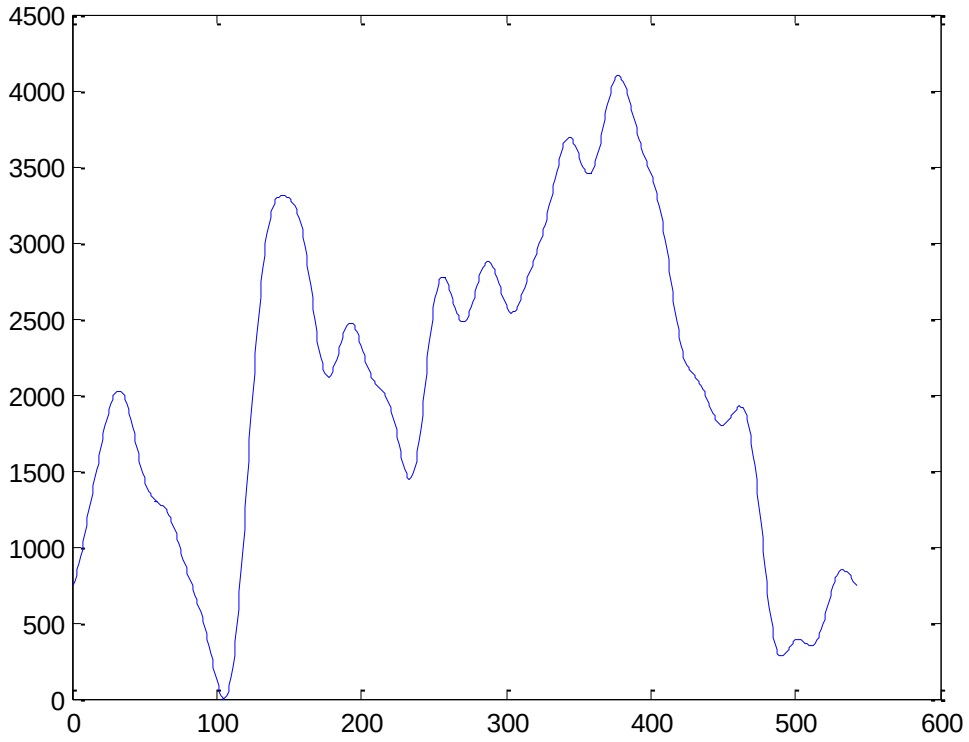


Fig. 3. 33 - Versión discreta de la fracción seleccionada con línea base corregida

Esta cuarto paso es el último propiamente dicho de procesado de la señal, puesto que pasada esta etapa la señal se encuentra definida en el rango en el que puede operar el convertidor D/A. Con la subrutina añadida en este paso se realiza un cambio de escala para acabar con un rango de valores de [0-4095].

```
load('chb01_01_edfm');

fl=0.5;
fh=30;
Fs=1000;
fh_Norm = fl/ (Fs/2);                    %# normalized cutoff frequency

[bh,ah] = butter(4, fh_Norm, 'high');   %# 4th order filter
fl_Norm = fh/ (Fs/2);                    %# normalized cutoff frequency
[bl,al] = butter(4, fl_Norm, 'low');   %# 4th order filter
```

```
der1= val();

der1= resample(der1, 1000,256);

der1 = filtfilt(bh, ah, der1);
der1 = filtfilt(bl, al, der1);

der1=der1(1,1800000:1820000);
der1=der1(1,500:1499);
der1=der1(1:543);

ml = length(der1);
mvec = linspace(der1(1),der1(ml),ml);
mvec = mvec - der1(1);
der1 = der1 - mvec;

mmax=max(der1);
mmin=min(der1);
der1_ADC= fix(4095.*((der1 - mmin)/(mmax - mmin)));

plot(der1_ADC)

i=0;
texto='const int16 EEG_form [543]={';
for i=1:ml,
    texto=sprintf('%s, %i', texto, der1_ADC(i));
end
texto=sprintf('%s };\n', texto);
texto_d1=texto;
disp(texto_d1);
```

SOFTWARE_FINAL_EEG_MATLAB

Este programa es el finalmente usado, incorpora todo lo previamente explicado, añadiéndole en la parte final del programa una nueva subrutina. Esta subrutina es la encargada de mostrar la señal como una tabla de valores, mostrando cada valor en formato texto.

A continuación se muestra la tabla de valores de la señal de EEG.

```
    const int16 EEG_form [543] ={ 737, 771, 808, 849, 892, 937, 984,
1033, 1083, 1135, 1187, 1239, 1292, 1344, 1397, 1449, 1501, 1552,
1602, 1651, 1699, 1745, 1790, 1831, 1871, 1906, 1938, 1966, 1989,
2007, 2019, 2025, 2025, 2019, 2007, 1990, 1967, 1938, 1906, 1869,
1829, 1787, 1742, 1697, 1652, 1607, 1563, 1522, 1483, 1448, 1417,
1390, 1366, 1347, 1331, 1318, 1308, 1300, 1293, 1286, 1280, 1273,
1264, 1254, 1241, 1226, 1208, 1188, 1164, 1139, 1111, 1081, 1050,
1018, 986, 953, 920, 888, 857, 826, 796, 767, 739, 711, 684, 657, 629,
600, 569, 537, 503, 467, 428, 386, 343, 297, 251, 206, 161, 119, 81,
49, 24, 7, 0, 3, 18, 46, 86, 139, 205, 284, 374, 476, 588, 709, 839,
975, 1117, 1262, 1411, 1560, 1709, 1857, 2001, 2141, 2275, 2403, 2524,
2636, 2739, 2834, 2918, 2993, 3059, 3115, 3163, 3202, 3234, 3259,
3278, 3292, 3301, 3307, 3310, 3310, 3308, 3304, 3299, 3291, 3282,
3271, 3256, 3239, 3218, 3192, 3161, 3125, 3083, 3034, 2980, 2920,
2854, 2785, 2712, 2638, 2563, 2490, 2420, 2354, 2294, 2242, 2198,
2163, 2137, 2122, 2115, 2118, 2130, 2148, 2173, 2203, 2236, 2272,
```

```
2307, 2342, 2375, 2404, 2429, 2448, 2461, 2468, 2468, 2462, 2450,
2432, 2409, 2382, 2352, 2320, 2287, 2253, 2221, 2191, 2163, 2138,
2116, 2098, 2082, 2069, 2058, 2049, 2039, 2030, 2019, 2006, 1990,
1971, 1947, 1920, 1888, 1852, 1812, 1769, 1723, 1677, 1631, 1586,
1545, 1509, 1479, 1458, 1446, 1445, 1455, 1479, 1514, 1562, 1622,
1692, 1771, 1858, 1950, 2046, 2143, 2239, 2331, 2418, 2498, 2570,
2632, 2683, 2724, 2753, 2771, 2778, 2776, 2765, 2747, 2723, 2694,
2662, 2630, 2597, 2566, 2538, 2515, 2496, 2484, 2478, 2478, 2486,
2499, 2519, 2545, 2574, 2608, 2643, 2679, 2715, 2750, 2782, 2811,
2835, 2854, 2867, 2874, 2875, 2870, 2859, 2842, 2821, 2795, 2766,
2735, 2703, 2671, 2640, 2612, 2587, 2567, 2551, 2542, 2538, 2540,
2547, 2560, 2578, 2599, 2624, 2651, 2679, 2709, 2739, 2768, 2797,
2826, 2853, 2880, 2907, 2933, 2960, 2988, 3017, 3048, 3082, 3117,
3155, 3195, 3238, 3282, 3327, 3373, 3420, 3465, 3508, 3549, 3586,
3619, 3646, 3667, 3682, 3690, 3691, 3685, 3673, 3656, 3634, 3608,
3581, 3553, 3525, 3500, 3479, 3462, 3451, 3447, 3450, 3460, 3477,
3501, 3532, 3569, 3610, 3656, 3704, 3753, 3803, 3852, 3899, 3943,
3983, 4017, 4046, 4068, 4084, 4093, 4095, 4089, 4078, 4060, 4037,
4010, 3978, 3944, 3907, 3869, 3831, 3792, 3755, 3719, 3684, 3652,
3621, 3591, 3564, 3537, 3510, 3484, 3456, 3427, 3396, 3362, 3325,
3284, 3239, 3190, 3136, 3078, 3017, 2952, 2884, 2815, 2745, 2675,
2607, 2542, 2480, 2423, 2372, 2325, 2285, 2250, 2221, 2197, 2177,
2161, 2147, 2135, 2123, 2112, 2099, 2086, 2071, 2054, 2035, 2015,
1993, 1970, 1947, 1924, 1901, 1879, 1859, 1842, 1827, 1815, 1807,
1802, 1802, 1804, 1810, 1820, 1832, 1845, 1861, 1876, 1891, 1904,
1915, 1922, 1924, 1921, 1910, 1892, 1866, 1831, 1788, 1735, 1674,
1604, 1527, 1443, 1353, 1259, 1162, 1063, 964, 867, 773, 685, 602,
528, 463, 407, 361, 326, 300, 284, 277, 277, 283, 293, 307, 323, 339,
355, 368, 379, 387, 392, 393, 391, 387, 380, 372, 364, 357, 351, 347,
346, 349, 356, 368, 384, 404, 429, 458, 489, 524, 560, 598, 635, 672,
707, 739, 769, 794, 815, 831, 843, 849, 851, 848, 841, 832, 819, 805,
790, 775, 760, 747, 737};
```

Array_valores_EEGform

3.5 - Comunicación SPI

SPI (Serial Peripherical Interface) es un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas transmite la información entre los diferentes dispositivos conectados al bus, que pueden actuar como transmisor y receptor simultáneamente, por lo que este tipo de comunicación serie es full duplex. Dos de estas líneas trasfieren los datos (una en cada dirección) y la tercera línea es la de reloj. El bus emplea un simple registro de desplazamiento para transmitir la información.

Tras esta breve introducción sobre el protocolo de comunicación, se procede a explicar la implementación de dicho protocolo en el microcontrolador.

```
void MCP4921_init(void)
{
   setup_timer_2(T2_DIV_BY_1,24,1); //Para generar un reloj de 250 KHz
   output_low(MCP4921_SELECT);
   output_low(MCP4921_CLK);
   output_low(MCP4921_DI);
   i=input(MCP4921_DO);
   setup_spi(SPI_MASTER  |  SPI_CLK_T2|  SPI_L_TO_H  |  SPI_SS_DISABLED
|SPI_XMIT_L_TO_H);
}

void MCP4921_Send(int16 Sample)
{
   int16 SampleBuffer;
   int8 Data = 0;
   output_low(MCP4921_SELECT);

   SampleBuffer= Sample;
   Data=(SampleBuffer>>8) & 0x0F;
   Data |= 0x30;
   spi_write(Data);

   SampleBuffer= 0b0000000011111111 & Sample;
   Data = SampleBuffer;
   spi_write(Data);

   output_high(MCP4921_SELECT);
}
```

Programa de comunicación SPI

La primera parte del código se encarga de configurar la forma en la que se realiza la comunicación SPI, 2 pines de salida para los datos SPI, 1 pin de señal de reloj (controlado por el timer 2 a 250KHz) y 1 pin de salida para la selección del chip. El sistema empezará configurando todas sus líneas de comunicación a nivel bajo.

El cálculo del valor de precarga del timer2 es el siguiente:

Tiempo = 125 kHz = 2 µs

Frecuencia del µC = 48 MHZ

Divisor frecuencia = 1*4

Frecuencia resultante = 12 MHz

Periodo resultante = 0,0833 µs

Conteo necesario = 2 / 0,0833µ = 24

Temporización efectiva = 24*0,0833 µs = 1,9992 µs

setup_timer_2(T2_DIV_BY_1,24,1)	La function de reloj se efectúa a través del timer2, con un preescaler de 1, un conteo efectivo de 24 y un postcaler de 1
SPI_MASTER	El dispositivo actúa como maestro
SPI_L_TO_H	La comunicación se efectúa en flanco ascendente

SPI_SS_DISABLED	La selección de esclavo deshabilitada
SPI_XMIT_L_TO_H	La comunicación se comienza en el flanco ascendente de la señal de reloj, cuando la comunicación entre dispositivos es autorizada

La segunda parte se encarga de la comunicación SPI propiamente dicha. Este protocolo de comunicación trabaja con un tamaño de 8 bits (1 byte), pero la variable que se desea transferir posee una longitud de 12 bits, debido a esto el envío de los datos realizarse mediante 2 particiones.

Para llevar a cabo la comunicación se crea una variable de 16 Bits, variable manipulada, junto con otra variable de 8 Bits, variable enviada. En primer lugar se realiza un desplazamiento a derechas de los 4 Bits más significativos, mediante el uso de una máscara se mantiene intactos los 4 Bits del dato y se establecen los Bits de configuración de la comunicación SPI (4 Bits menos significativos), acto seguido se produce el envío. A continuación son enviados los 8 Bits restantes del dato que, también, se aplica sobre ellos una máscara para mantenerlos intactos.

Las patillas encargadas de la comunicación SPI son nombradas en el fichero de cabecera de todos los programas.

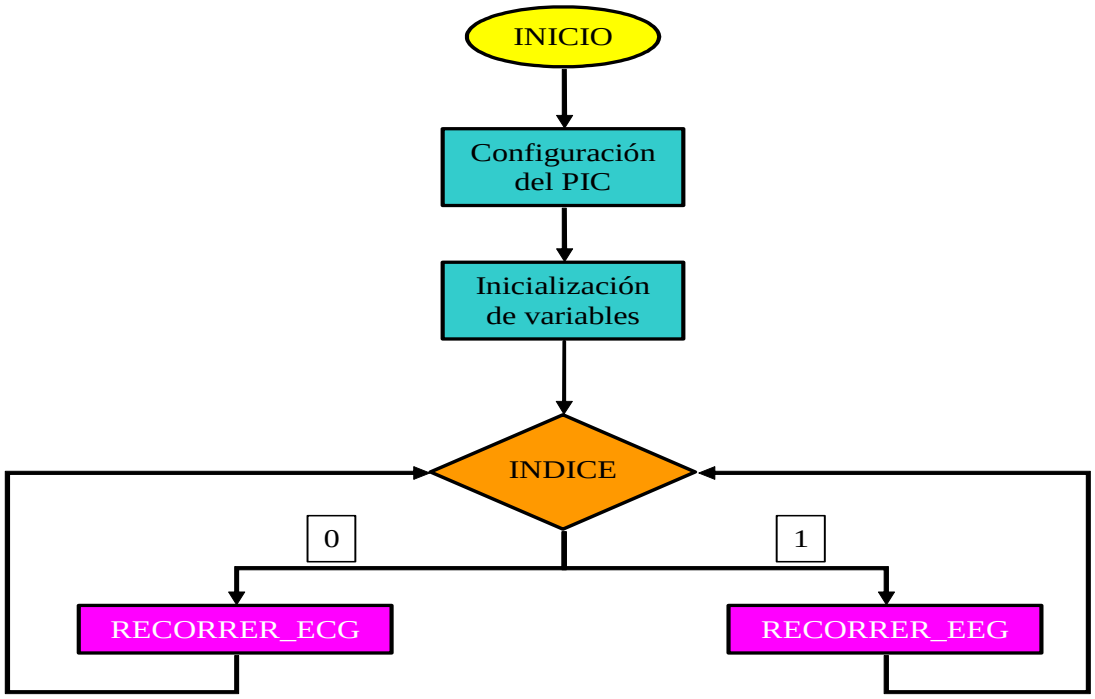
```
#define MCP4921_SELECT PIN_B5
#define MCP4921_DI     PIN_C7
#define MCP4921_DO     PIN_B4
#define MCP4921_CLK    PIN_B6
```

Identificación de las patillas SPI

3.6 - Integración del software final

Este programa es el utilizado en la programación del microcontrolador, ya que realiza todas las tareas necesarias en el proyecto.

El proyecto tiene la siguiente estructura de programación:



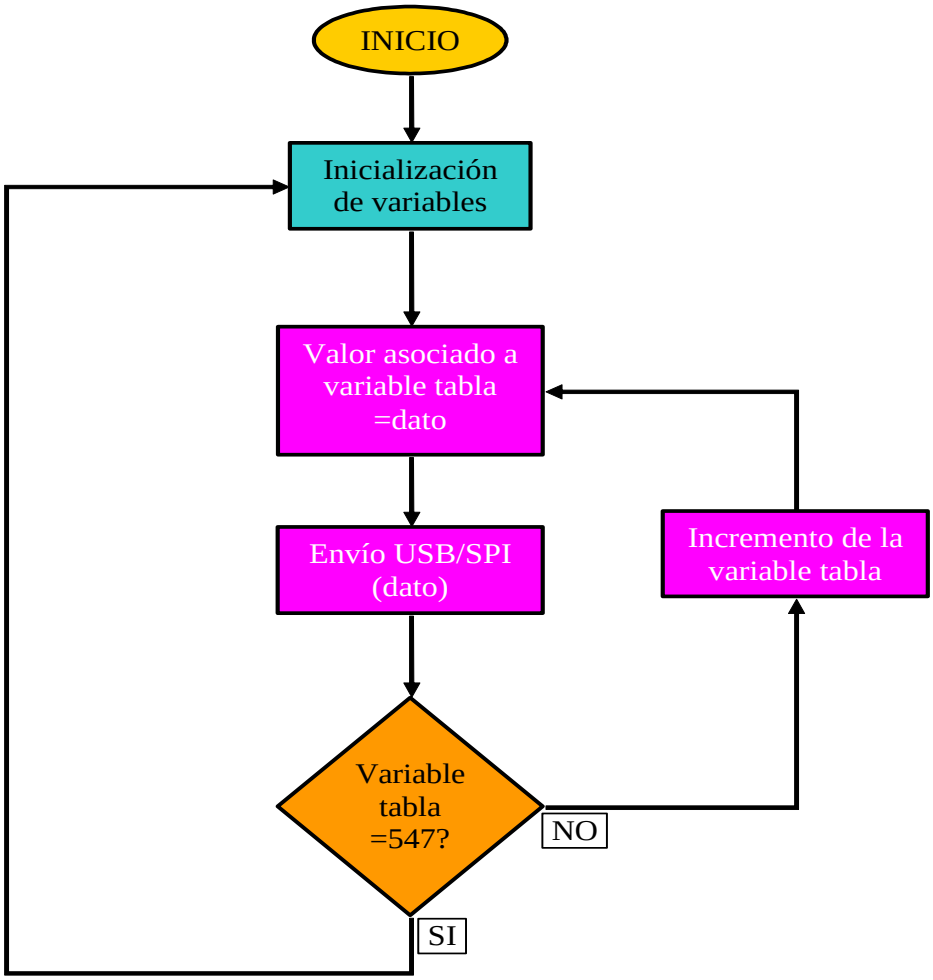
FLUJOGRAMA PRINCIPAL

El programa comienza por la configuración de todos los parámetros presentes en el microcontrolador y la inicialización y declaración de todas las variables utilizadas en el programa del PIC.

Acto seguido se produce la evaluación de la variable índice; en base al valor de ésta, se produce la comunicación de una señal u otra.

El envío de la señal debe finalizarse para que se pueda alternar la utilización de la señal.

El funcionamiento de 'recorrer_EEG' es el que se contempla en el siguiente diagrama de flujo.



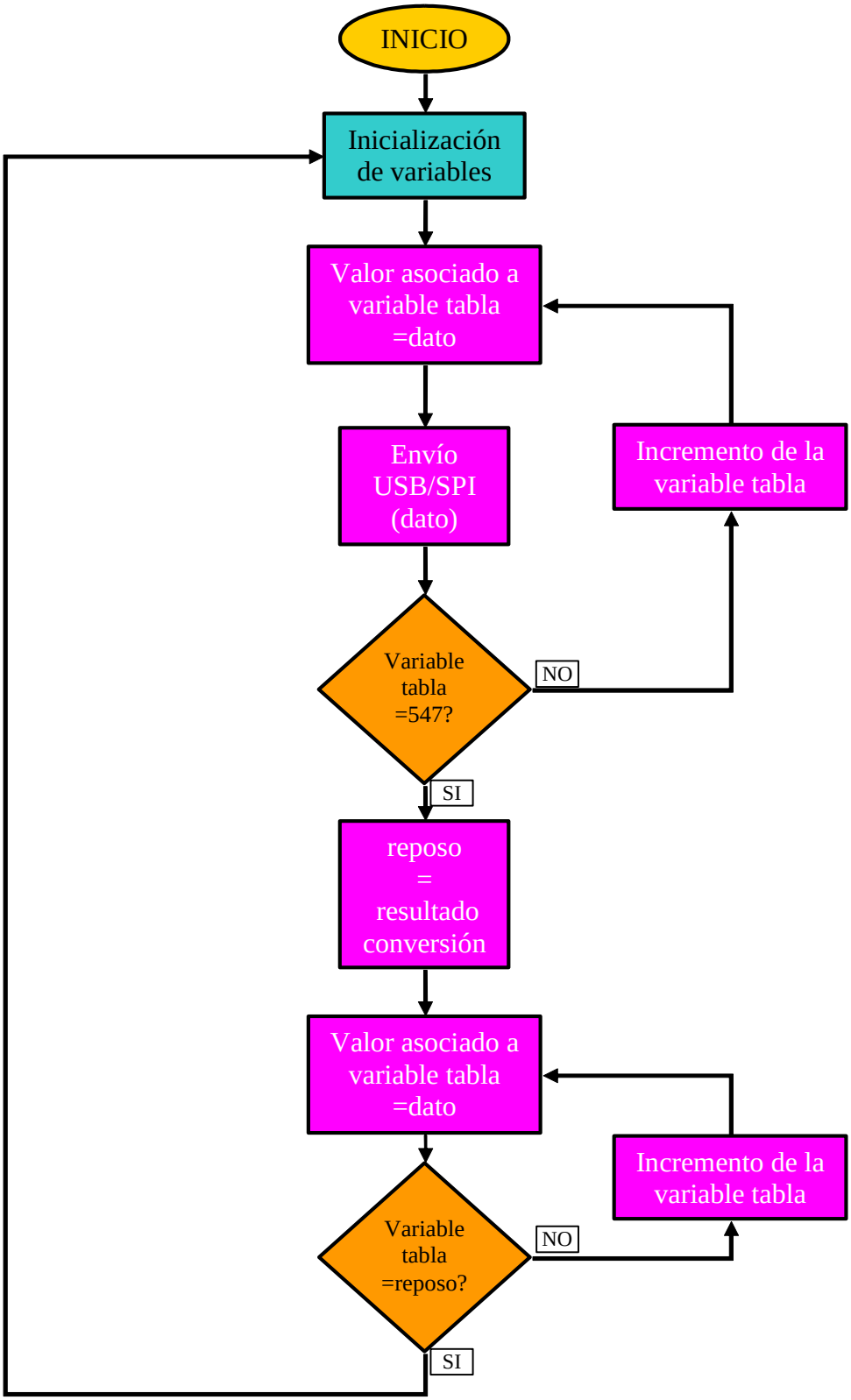
FLUJOGRAMA DE RECORRER_EEG

En primer lugar son inicializadas las variables que actúan en su funcionamiento.

Acto seguido se comienza el recorrido por la tabla de valores depositada en la memoria RAM del micro. Cada uno de los valores de la tabla es enviado simultáneamente a dos destinos diferentes. El primero de los destinos es el PC, usando la comunicación USB. El segundo destino es el convertidor Digital-Analógico, usando el protocolo de comunicación SPI. El envío de un valor respecto a otro está retrasado un tiempo de 1 milisegundo.

En total la tabla de valores posee una cantidad de 547 muestras, que se envían de forma seguida sin interrupción.

El modo de funcionamiento de 'recorrer_ECG', incluido en el diagrama de flujo principal, se explica en el siguiente flujograma.



FLUJOGRAMA DE RECORRER_ECG

En su inicio se produce la inicialización de todas las variables que componen el funcionamiento de este bloque.

Acto seguido se comienza el recorrido por la tabla de valores depositada en la memoria RAM del micro. Cada uno de los valores de la tabla es enviado simultáneamente a dos destinos diferentes. El primero de los destinos es el PC, mediante el uso de la comunicación USB. El segundo destino es el convertidor Digital-Analógico, realizado mediante el protocolo de comunicación SPI. El envío de un valor respecto a otro está retrasado un tiempo de 1 milisegundo. A continuación de esto, se produce un tiempo de latencia entre ciclo y ciclo, en el cual se produce el envío del último valor de la señal, espaciado 1 milisegundo cada envío, valor que coincide con el primero del siguiente ciclo de la señal.

Este tiempo de latencia viene marcado por el valor de resistencia del potenciómetro, así pues, si el valor de dicho potenciómetro es máximo (resistencia máxima) el tiempo de latencia es mínimo, prácticamente 0. Si el valor del potenciómetro es mínimo, el tiempo de latencia es el máximo, alrededor de un segundo.

Esto se lleva a cabo mediante la conversión A/D del valor de tensión proporcionado por el potenciómetro. Esto genera un número determinado de bits, que puede tener un valor entre 0 y 1023, cuyo este valor es guardado en la variable 'xpot'.

El número de veces que será enviado el último valor de la señal está regulado por el valor que posea la variable reposo. Este valor es calculado de la siguiente manera:

$$reposo = 1024 - xpot$$

De manera que si 'xpot' tiene un valor de 1023, el tiempo de espera es de 1 milisegundo, de igual manera si 'xpot' tiene un valor de 0, el tiempo de latencia es de 1024 milisegundos. La variable 'xpot' puede poseer un valor entero entre 0 y 1023, lo que ofrece la posibilidad de 1024 valores diferentes.

- Primer caso: Valor máximo resistivo del potenciómetro

543ms (duración de la señal)

Reposo = 1024 – 1023 = 1ms

Duración de señal = 543 + 1 = 544ms = 1,838 ciclos/s

BPM (latidos por minuto) = 1,838 * 60 = 110, 28 bpm ≈ 110 bpm

- Segundo caso: Valor mínimo resistivo del potenciometro

543ms (duración de la señal)

Reposo = 1024 – 0 = 1024ms

Duración de señal = 543 + 1024 = 1557ms = 0,642 ciclos/s

BPM (latidos por minuto) = 0,642 * 60 = 38,52 bpm ≈ 38 bpm

Esto hace posible que la señal oscile entre 110 bpm (latidos por minuto) y 38 bpm, dando lugar a un gran rango de valores intermedios.

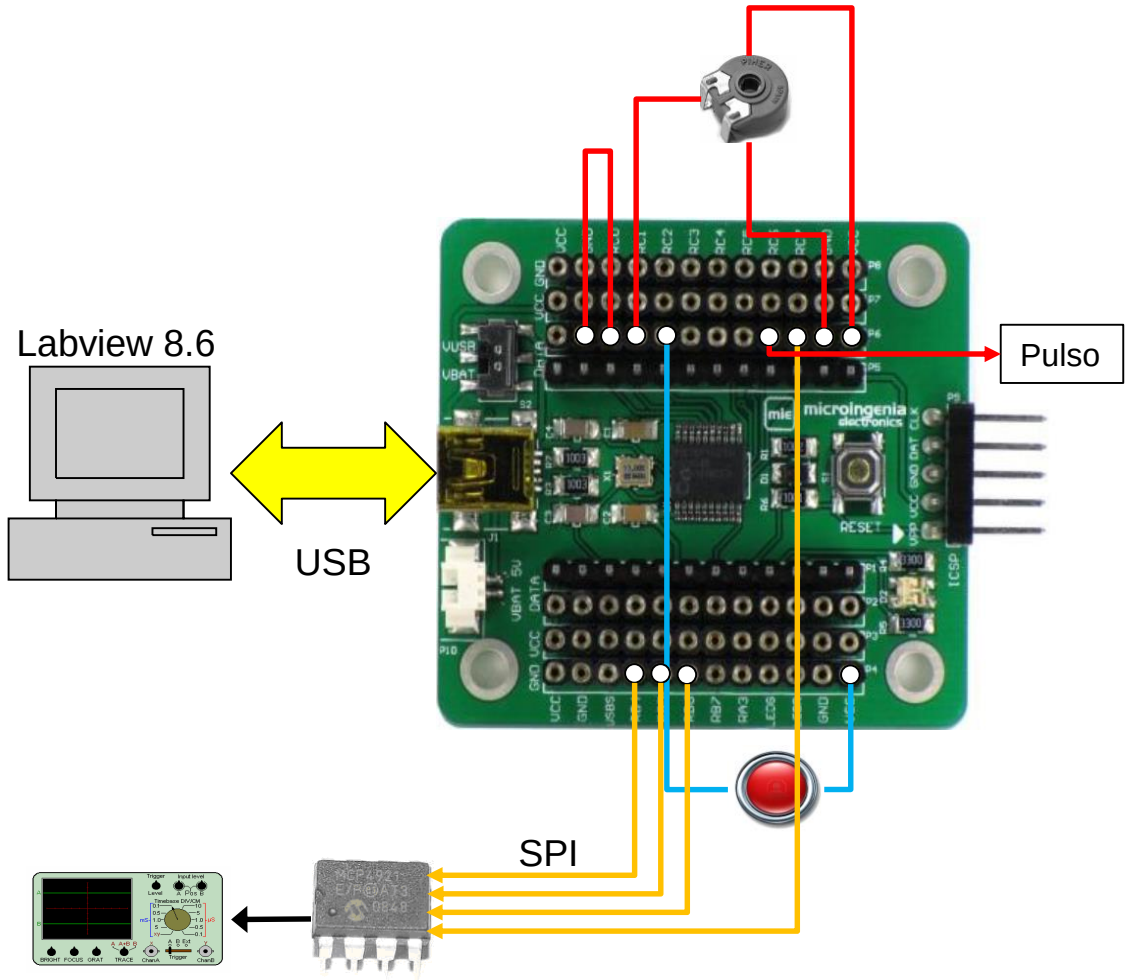


Fig. 3.34 - Conexiones placa de desarrollo 3.6

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

signed int16 i=0, j=0, xpot=0, reposo=0;
int8 indice = 0;

const int16 DerI_form[543]={
381, 384, 387, 390, 393, 396, 399, 403, 407, 411,
415, 420, 424, 429, 435, 441, 447, 454, 461, 469,
477, 485, 495, 504, 515, 526, 537, 549, 562, 575,
588, 602, 616, 630, 645, 660, 675, 690, 705, 720,
734, 749, 763, 776, 789, 801, 813, 823, 833, 842,
849, 856, 861, 865, 867, 868, 868, 866, 863, 859,
853, 846, 837, 827, 816, 803, 790, 775, 760, 744,
728, 710, 693, 675, 657, 639, 622, 604, 588, 571,
556, 541, 527, 514, 502, 491, 481, 473, 465, 459,
454, 450, 447, 445, 444, 443, 443, 443, 444, 445,
445, 446, 445, 444, 442, 439, 435, 430, 423, 414,
404, 392, 378, 362, 345, 326, 306, 284, 261, 238,
```

```
215, 191, 168, 146, 126, 107, 92, 80, 72, 68,
70, 78, 93, 115, 145, 183, 231, 288, 354, 431,
518, 614, 721, 838, 965, 1101, 1245, 1396, 1555, 1720,
1889, 2062, 2236, 2412, 2586, 2759, 2927, 3089, 3244, 3391,
3527, 3651, 3762, 3859, 3940, 4005, 4053, 4083, 4095, 4088,
4063, 4020, 3960, 3882, 3789, 3680, 3558, 3423, 3276, 3120,
2956, 2785, 2610, 2431, 2251, 2071, 1893, 1718, 1547, 1382,
1224, 1074, 932, 800, 677, 565, 463, 372, 292, 222,
162, 113, 73, 43, 21, 6, 0, 0, 5, 16,
32, 51, 73, 98, 125, 152, 181, 210, 238, 266,
292, 318, 342, 364, 384, 402, 418, 432, 443, 453,
461, 467, 471, 473, 474, 473, 472, 469, 466, 461,
456, 451, 446, 440, 435, 429, 424, 419, 414, 410,
406, 403, 400, 398, 396, 395, 394, 394, 394, 395,
396, 397, 399, 401, 403, 405, 408, 410, 413, 415,
418, 420, 423, 425, 427, 429, 431, 433, 434, 436,
437, 438, 439, 440, 441, 441, 442, 442, 443, 443,
443, 444, 444, 444, 445, 445, 446, 446, 447, 448,
449, 449, 450, 452, 453, 454, 456, 457, 459, 461,
463, 465, 467, 469, 471, 473, 476, 478, 481, 483,
486, 488, 491, 493, 496, 499, 501, 504, 507, 509,
512, 515, 518, 521, 524, 527, 530, 533, 537, 540,
544, 548, 551, 555, 559, 564, 568, 572, 577, 582,
587, 591, 596, 602, 607, 612, 617, 623, 628, 634,
639, 645, 651, 656, 662, 668, 674, 680, 686, 693,
699, 706, 712, 719, 726, 733, 740, 748, 755, 763,
771, 779, 787, 796, 805, 813, 822, 832, 841, 850,
860, 869, 879, 889, 899, 909, 919, 930, 940, 951,
961, 972, 982, 993, 1004, 1015, 1026, 1036, 1047, 1058,
1070, 1081, 1092, 1103, 1114, 1125, 1136, 1147, 1157, 1168,
1178, 1189, 1199, 1209, 1219, 1228, 1237, 1246, 1254, 1263,
1270, 1277, 1284, 1291, 1296, 1302, 1306, 1310, 1314, 1316,
1318, 1320, 1320, 1320, 1319, 1317, 1315, 1311, 1307, 1302,
1296, 1290, 1282, 1274, 1264, 1255, 1244, 1232, 1220, 1207,
1193, 1179, 1164, 1148, 1132, 1115, 1098, 1080, 1063, 1044,
1026, 1007, 988, 968, 949, 930, 910, 891, 871, 852,
833, 814, 795, 777, 759, 741, 724, 707, 690, 674,
658, 643, 628, 614, 601, 587, 575, 563, 551, 540,
530, 520, 511, 502, 494, 486, 479, 472, 466, 460,
455, 450, 445, 441, 437, 433, 430, 427, 424, 422,
419, 417, 415, 413, 412, 410, 408, 407, 405, 404,
402, 401, 399, 398, 396, 395, 393, 391, 389, 387,
385, 383, 381};

const int16 EEG_form[543]={
737, 771, 808, 849, 892, 937, 984, 1033, 1083, 1135,
1187, 1239, 1292, 1344, 1397, 1449, 1501, 1552, 1602, 1651,
1699, 1745, 1790, 1831, 1871, 1906, 1938, 1966, 1989, 2007,
2019, 2025, 2025, 2019, 2007, 1990, 1967, 1938, 1906, 1869,
1829, 1787, 1742, 1697, 1652, 1607, 1563, 1522, 1483, 1448,
1417, 1390, 1366, 1347, 1331, 1318, 1308, 1300, 1293, 1286,
1280, 1273, 1264, 1254, 1241, 1226, 1208, 1188, 1164, 1139,
1111, 1081, 1050, 1018, 986, 953, 920, 888, 857, 826,
796, 767, 739, 711, 684, 657, 629, 600, 569, 537,
503, 467, 428, 386, 343, 297, 251, 206, 161, 119,
81, 49, 24, 7, 0, 3, 18, 46, 86, 139,
205, 284, 374, 476, 588, 709, 839, 975, 1117, 1262,
1411, 1560, 1709, 1857, 2001, 2141, 2275, 2403, 2524, 2636,
2739, 2834, 2918, 2993, 3059, 3115, 3163, 3202, 3234, 3259,
3278, 3292, 3301, 3307, 3310, 3310, 3308, 3304, 3299, 3291,
3282, 3271, 3256, 3239, 3218, 3192, 3161, 3125, 3083, 3034,
```

```
2980, 2920, 2854, 2785, 2712, 2638, 2563, 2490, 2420, 2354,
2294, 2242, 2198, 2163, 2137, 2122, 2115, 2118, 2130, 2148,
2173, 2203, 2236, 2272, 2307, 2342, 2375, 2404, 2429, 2448,
2461, 2468, 2468, 2462, 2450, 2432, 2409, 2382, 2352, 2320,
2287, 2253, 2221, 2191, 2163, 2138, 2116, 2098, 2082, 2069,
2058, 2049, 2039, 2030, 2019, 2006, 1990, 1971, 1947, 1920,
1888, 1852, 1812, 1769, 1723, 1677, 1631, 1586, 1545, 1509,
1479, 1458, 1446, 1445, 1455, 1479, 1514, 1562, 1622, 1692,
1771, 1858, 1950, 2046, 2143, 2239, 2331, 2418, 2498, 2570,
2632, 2683, 2724, 2753, 2771, 2778, 2776, 2765, 2747, 2723,
2694, 2662, 2630, 2597, 2566, 2538, 2515, 2496, 2484, 2478,
2478, 2486, 2499, 2519, 2545, 2574, 2608, 2643, 2679, 2715,
2750, 2782, 2811, 2835, 2854, 2867, 2874, 2875, 2870, 2859,
2842, 2821, 2795, 2766, 2735, 2703, 2671, 2640, 2612, 2587,
2567, 2551, 2542, 2538, 2540, 2547, 2560, 2578, 2599, 2624,
2651, 2679, 2709, 2739, 2768, 2797, 2826, 2853, 2880, 2907,
2933, 2960, 2988, 3017, 3048, 3082, 3117, 3155, 3195, 3238,
3282, 3327, 3373, 3420, 3465, 3508, 3549, 3586, 3619, 3646,
3667, 3682, 3690, 3691, 3685, 3673, 3656, 3634, 3608, 3581,
3553, 3525, 3500, 3479, 3462, 3451, 3447, 3450, 3460, 3477,
3501, 3532, 3569, 3610, 3656, 3704, 3753, 3803, 3852, 3899,
3943, 3983, 4017, 4046, 4068, 4084, 4093, 4095, 4089, 4078,
4060, 4037, 4010, 3978, 3944, 3907, 3869, 3831, 3792, 3755,
3719, 3684, 3652, 3621, 3591, 3564, 3537, 3510, 3484, 3456,
3427, 3396, 3362, 3325, 3284, 3239, 3190, 3136, 3078, 3017,
2952, 2884, 2815, 2745, 2675, 2607, 2542, 2480, 2423, 2372,
2325, 2285, 2250, 2221, 2197, 2177, 2161, 2147, 2135, 2123,
2112, 2099, 2086, 2071, 2054, 2035, 2015, 1993, 1970, 1947,
1924, 1901, 1879, 1859, 1842, 1827, 1815, 1807, 1802, 1802,
1804, 1810, 1820, 1832, 1845, 1861, 1876, 1891, 1904, 1915,
1922, 1924, 1921, 1910, 1892, 1866, 1831, 1788, 1735, 1674,
1604, 1527, 1443, 1353, 1259, 1162, 1063, 964, 867, 773,
685, 602, 528, 463, 407, 361, 326, 300, 284, 277,
277, 283, 293, 307, 323, 339, 355, 368, 379, 387,
392, 393, 391, 387, 380, 372, 364, 357, 351, 347,
346, 349, 356, 368, 384, 404, 429, 458, 489, 524,
560, 598, 635, 672, 707, 739, 769, 794, 815, 831,
843, 849, 851, 848, 841, 832, 819, 805, 790, 775,
760, 747, 737};

void MCP4921_init(void)
{
   setup_timer_2(T2_DIV_BY_1,24,1); //Para generar un reloj de 250 KHz
   output_low(MCP4921_SELECT);
   output_low(MCP4921_CLK);
   output_low(MCP4921_DI);
   i=input(MCP4921_DO);
   setup_spi(SPI_MASTER  |  SPI_CLK_T2|  SPI_L_TO_H  |  SPI_SS_DISABLED
|SPI_XMIT_L_TO_H);
}

void MCP4921_Send(int16 Sample)
{
   int16 SampleBuffer;
   int8 Data = 0;

   output_low(MCP4921_SELECT);

   SampleBuffer= Sample;
   Data=(SampleBuffer>>8) & 0x0F;
   Data |= 0x30;
```

```
   spi_write(Data);


   SampleBuffer= 0b0000000011111111 & Sample;
   Data = SampleBuffer;
   spi_write(Data);

   output_high(MCP4921_SELECT);
}

void recorrerECG(void)
{
   xpot=read_adc();
   reposo=1024-xpot;
   if (reposo <0) reposo =0;
   if (reposo >1024) reposo = 1024;

   for(i=0;i<543;i++)
      {
         j = DerI_form[i];
         MCP4921_Send(j);
               if(i<180) output_high(PIN_C6); //generar minipulso
               else output_low(PIN_C6);
         printf(usb_cdc_putc, "%li \r\n" ,j);
         delay_ms(1);
      }
   for(i=0;i<reposo;i++)
      {
         MCP4921_Send(j);
         printf(usb_cdc_putc,"%li \r\n",j);
         delay_ms(1);
      }
}

void recorrerEEG(void)
{
      for(i=0;i<543;i++)
      {
         j = EEG_form[i];
         MCP4921_Send(j);
         printf(usb_cdc_putc, "%li \r\n" ,j);
         delay_ms(1);
      }
}

void main(void)
{
    ext_int_edge(2,L_TO_H);
    enable_interrupts(INT_EXT2);
    enable_interrupts(GLOBAL);

    usb_cdc_init();
    usb_init();
    usb_wait_for_enumeration();

    MCP4921_init();

    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(sAN5|VSS_VDD );
    set_adc_channel(5);
```

```
   while(TRUE)
   {
      switch(indice)
      {
         case 0:
            recorrerECG();
         break;

         case 1:
            recorrerEEG();
         break;

         default:
            indice=0;
         break;
      }
   }
}
#INT_EXT2
void interrupt_RC2()//accionar pulsador conectado a RC2 para que se
lea una u otra señal de las tablas de memoria (ECG / EEG)
{
   indice++;
   delay_ms(175);
}
```

SOFTWARE FINAL

El programa comienza con una serie de configuraciones específicas para la placa entrenadora utilizada en el proyecto. Tras la configuración, inicializa las variables globales y las tablas de valores pertenecientes a cada una de las señales con las cuales se trabaja.

A continuación se crean los prototipos de las funciones que se van a utilizar a lo largo del programa. En primer lugar se crea la función encargada de inicializar la comunicación SPI, seguida de la función que gestiona la comunicación en sí misma.

Acto seguido, se elaboran las funciones encargadas de recorrer las tablas de valores. También se debe recalcar que, al comienzo de cada ciclo de la señal de ECG se produce la activación de la patilla RC6 durante una duración aproximada de 180ms, para ser usada como frecuencímetro.

Una vez finalizada la creación de las funciones utilizadas, se procede a implementar la función 'main' que se ejecuta en el PIC. Dentro de ella, en primer lugar, se configura la interrupción producida por el pulsador, a continuación, la comunicación USB y SPI y, por último, la conversión A/D. En segundo lugar se produce la evaluación de la variable 'indice' de forma ininterrumpida, variable encargada de escoger la tabla de valores que se manipula.

En último lugar se crea la función que se ejecuta cuando se produce la interrupción por el accionamiento del pulsador, la espera de 175ms tiene como fin atenuar el ruido al ser apretado el pulsador.

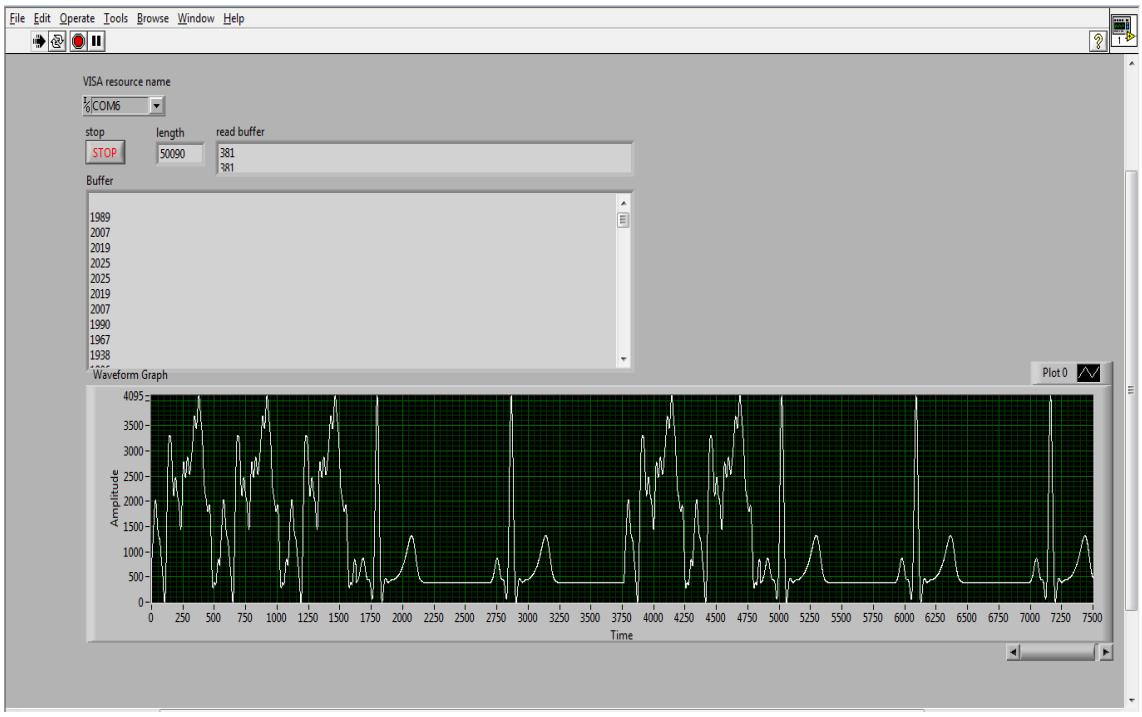


Fig. 3.35 - Captura de envío del software final

Obsérvese el cambio de señal generada con la activación del pulsador

Capítulo 4

Desarrollo hardware

4.1 - Planteamiento del desarrollo

Para la implementación física del sistema propuesto, se lleva a cabo el diseño de un prototipo hardware final haciendo uso del paquete Eagle v6.40. Así, se parte de un esquemático inicial fruto de las pruebas realizadas en la etapa de desarrollo software, que es depurado y aumentado para adaptarlo a los requisitos del proyecto. Finalmente, se confecciona una placa de circuito impreso haciendo uso de la herramienta incluida en Eagle a tal efecto.

4.2 - Easily Applicable Graphical Layout Editor (EAGLE)

Dentro de los múltiples programas de diseños de PCB en el mercado, se ha optado por la utilización de EAGLE (Easily Applicable Graphical Layout Editor). Todo el desarrollo hardware que compone este proyecto, tanto esquemáticos como PCBs, está hecho este programa.

Fig. 4.1 - Logotipo de Eagle®

Este software de diseño PCB es capaz de diseñar un esquema electrónico y una placa de circuito impreso (PCB), es un software de diseño de paquetes PCB que consta de un editor de esquemas, un editor de PCB y un módulo de trazador. El software viene con una amplia biblioteca de componentes, pero ésta puede ampliarse, puesto que está incluido un editor para el diseño de nuevas piezas o para modificar las ya existentes. Esta desarrollado por CadSoft (www.cadsoftusa.com), y está disponible en tres versiones, tanto para windows como para Linux. En este proyecto se hace uso de la versión 'light', puesto que sus restricciones son suficientes para poder realizar este proyecto.

- Restricciones de la licencia Eagle Light Edition.
  - Área de diseño de placa limitada a 100 x 80 mm.
  - Sólo dos capas de diseño (top y bottom).
  - Tan sólo se puede crear un esquema por proyecto.
  - Soporte limitado vía e-mail o mediante la utilización de los foros.
  - Uso limitado a aplicaciones que no produzcan beneficios o para placas de evaluación.
  - Existe el permiso de distribución de EAGLE con distribuciones de Linux con colecciones de software o CD-ROMs, mediante información a la empresa desarrolladora.

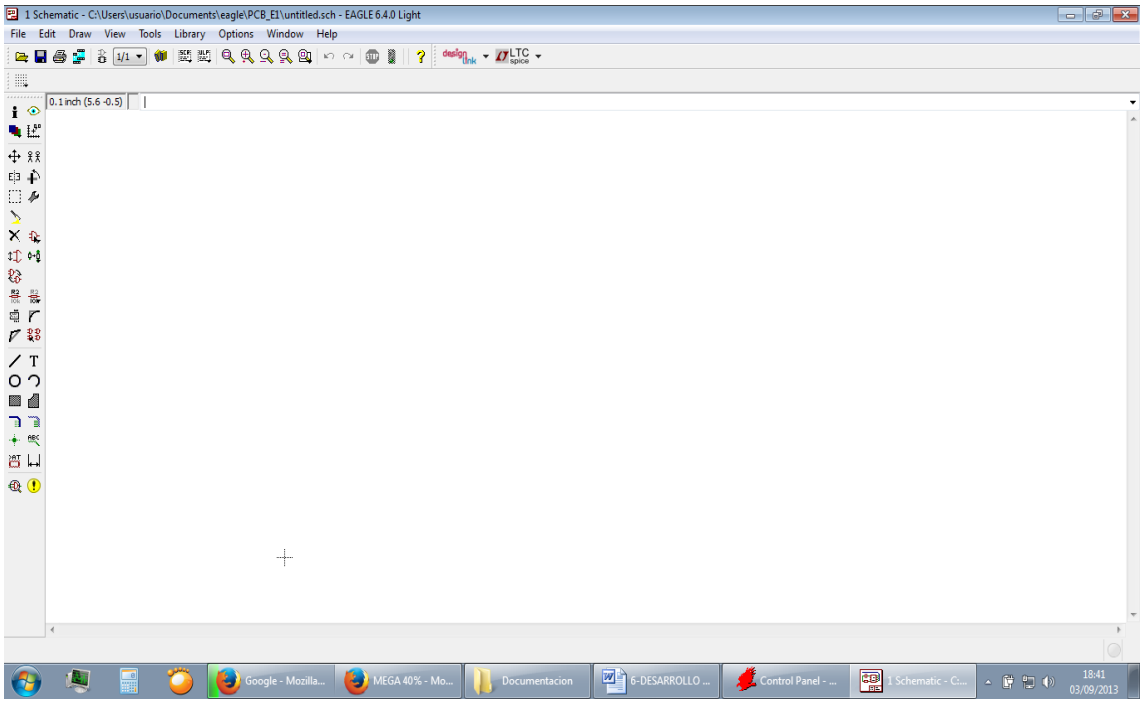


Fig. 4.2 - Entorno de desarrollo de Eagle

En este proyecto no se ha podido encontrar el 100% de los componentes necesarios en el interior de las librerías facilitadas por EAGLE, por lo que se han tenido que importar librerías descargadas de la red.

En total se han tenido que descargar 2 librerías:

1. En el interior de la primera librería (18F14K50.lbr) se encuentra almacenado el PIC 18F14K50, tanto su esquemático como se PCB.
2. En el interior de la segunda (adamfuit.lbr) se encuentra el convertidor MCP4921, además de una gran cantidad de componentes. Se puede encontrar en su interior tanto su esquemático como se PCB.

4.3 - Características principales del PIC 18F14K50

El microcontrolador que se ha elegido para la realización del proyecto es el PIC18F14K50, de la casa Microchip. Esta familia ofrece las ventajas de los microcontroladores PIC18, es decir, alto rendimiento de computación a un precio económico, con la adición de alta resistencia y memoria 'flash'. Además de estas características, la familia PIC18F1XK50/PIC18LF1XK50 presenta mejoras de diseño que hacen a estos microcontroladores una opción lógica para este proyecto, ya que se trata de una aplicación sensible a la energía que, además, requiere un alto rendimiento.

Las características principales del microcontrolador son las siguientes:

Compatibilidad USB:

a) USB 2.0
b) Baja velocidad (1.5 Mb/s) y alta velocidad (12 Mb/s)
c) Soporta transferencias de tipo control, interrupción, asíncrona y Bulk
d) 256 bytes de memoria RAM para USB
e) Detección de conexión USB mediante D+/D-
f) Soporta hasta 16 Endpoints (8 bidireccionales)

Modos de administración de energía:

a) Run: CPU on, periféricos on
b) Idle: CPU off, periféricos on
c) Sleep: CPU off, periféricos off
d) Oscilador de arranque (2 velocidades)

Periféricos:

a) 3 interrupciones externas programables
b) 7 interrupciones independientes de cambios de voltaje de entrada
c) 7 resistencias de pull-ups programables
d) Módulo ECCP (Enhanced Capture/Compare/PWM)
e) Módulo MSSP (Master Synchronous Serial Port) que soporta comunicación SPI de 3 hilos y comunicación I2C™ con modo esclavo y maestro
f) Módulo EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter).
g) 9 canales de converisión Analógica-Digital, 10 bits de resolución
h) Comparador analógico dual

Estructura del oscilador

a) Divisor de frecuencia para CPU
b) 4 tipos de cristales resonadores (4 - 48 MHz)
c) Oscilador interno (31 KHz – 16 MHz)
d) 4X Phase Lock Loop (PLL)
e) Oscilador secundario utilizando timer1 a 32 KHz
f) Respaldo de frecuencia de reloj

Características especiales del microcontrolador

a) Arquitectura optimizada para compilación en C
b) Niveles de prioridad para las interrupciones
c) Perro guardián (WDT) de 4ms hasta 131s
d) Auto programable bajo software de control
e) Multiplicador hardware de 8 bits
f) Alimentación 3V In-Circuit Serial Programming™ (ICSP™) usando 2 pines
g) Rango de voltaje de alimentación (1,8V – 5,5V)
h) Brown-out Reset (BOR) programable
i) Dos velocidades de arranque

Device	Program Memory		Data Memory		I/O(1)	10-bit A/D (ch)(2)	ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-bit	USB
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C™				
PIC18F13K50/ PIC18LF13K50	8K	4096	512(3)	256	15	11	1	Y	Y	1	2	1/3	Y
PIC18F14K50/ PIC18LF14K50	16K	8192	768(3)	256	15	11	1	Y	Y	1	2	1/3	Y

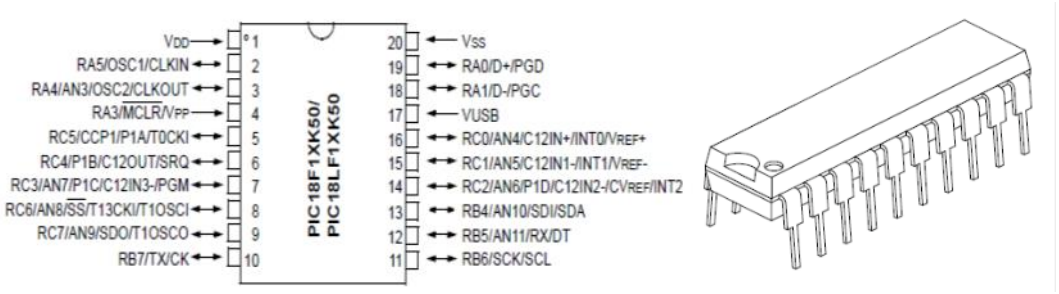


Fig. 4.3 - Microcontrolador 18F14K50

Para su conexión al PC a través del puerto USB se hace preciso seleccionar el conector adecuado según las recomendaciones del estándar.

El Universal Serial Bus (**USB**) es un estándar de comunicación síncrono con un bus serie bidireccional de bajo coste, que define los cables, conectores y protocolos usados en el bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos o dispositivos electrónicos. Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultánea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido.

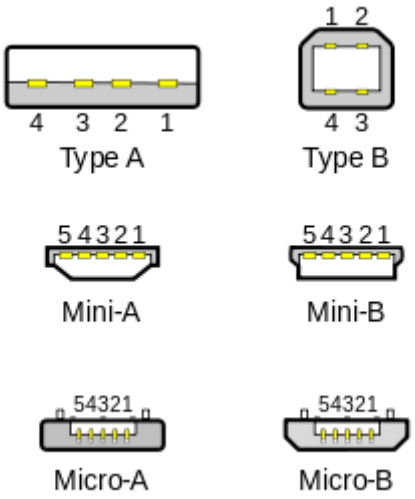


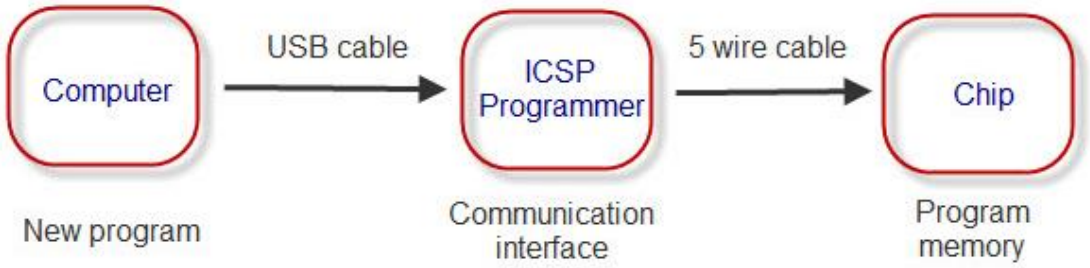
Fig. 4.4 - Tipos de conectores USB

El PIC hace uso del estándar de comunicación USB 2.0, con una tasa de transferencia de 60 MB/s (480 Mbit/s), totalmente compatible con los dispositivos USB 1.0, puesto que es capaz de ajustar su velocidad entre la velocidad de éste y la nueva incluida en él. Para su comunicación se utilizan 4 líneas:

- *VCC* (PIN 1) es la línea de alimentación USB, típicamente 5V.
- *GND* (PIN 4) es la línea de masa o tierra de la comunicación.
- *D+* (PIN 3) es la segunda línea de comunicación de datos.
- *D-* (PIN 2) es la primera línea de comunicación de datos.

De forma similar, para hacer uso de la funcionalidad ICSP, se hace preciso definir el conector adecuado.

Los PICs se programan utilizando cinco señales, dos de ellas dedicadas a la comunicación síncrona, y las tres líneas restantes destinadas a proporcionar la energía al microcontrolador. La señal de reloj está siempre controlada por el programador.



Comunicación de programación típica

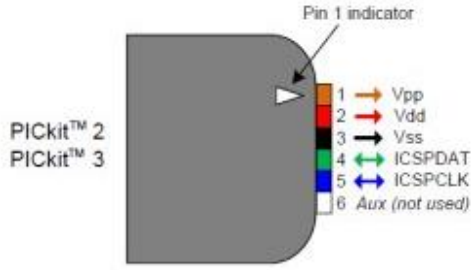


Fig. 4.5 - Conectores ICSP

Líneas que componen la programación por ICSP:

- *VPP* (PIN 1) es el voltaje necesario para preparar el PIC para que éste funcione en modo programación. Esta señal debe estar conectada al pin MCLR del micro; dependiendo de si su valor es GND o VDD, el microcontrolador se encuentra en modo programación o no.
- *VDD* (PIN 2) es la entrada de energía positiva al PIC.
- *VSS* (PIN 3) es la entrada de energía negativa al PIC y la referencia de 0 voltios para las señales restantes.
- *ICSPCLK* (PIN 5) es la línea de reloj de la comunicación serie. El cometido de esta línea es producir oscilaciones entre GND y VDD, cuya frecuencia es elegida por el programador.
- *ICSPDAT* (PIN 4) es la línea de datos serie. Su interfaz es bidireccional, por lo que los datos pueden ser enviados del PIC al programador y viceversa. Los cambios que se producen en la línea son entre GND y VDD, siendo producida la transmisión en el flanco descendente.

4.4 - Características principales del MCP4921

Los dispositivos de la serie MCP492X de Microchip Technology son convertidores digitales-analógicos de 12 bits (DAC), de baja potencia, bajo DNL (Differential Non-Linearity), interfaz SPI, y con la opción de buffer de salida 2X.

Los MCP492X proporcionan una alta precisión y rendimiento con bajo ruido intrínseco para aplicaciones industriales donde la calibración o la compensación de las señales (por temperatura, presión y humedad) son necesarias.

Estos dispositivos utilizan una arquitectura de cadena resistiva, con sus ventajas inherentes; bajo error DNL, coeficiente temperatura medición bajo y estabilización rápida en el tiempo. También incluye doble Buffer de entrada, lo que permite actualizaciones simultáneas utilizando el pin LDAC (MCP4921 no dispone de él). Estos dispositivos también incorporan un Power-On Circuit Reset (POR) que asegura un buen funcionamiento de la puesta en marcha del dispositivo.

Características

a) 12 bits de resolución
b) ±0.2 LSB DNL
c) ±2 LSB INL (Integral Non-Linearity)
d) 1 canal (4921) o 2 canales (4922)
e) Salida Rail-to-Rail
f) Interfaz de comunicación SPI
g) Toma de valores simultáneos (dual, 4922)
h) Ganancia de salida seleccionable (1X o 2X)
i) Tiempo de establecimiento 4.5 µs
j) Modo multiplicador a 450 KHz
k) Entrada externa de V_{REF}
l) Rango de voltaje máximo trabajando con una fuente de energía de 2.7V a 5.5V

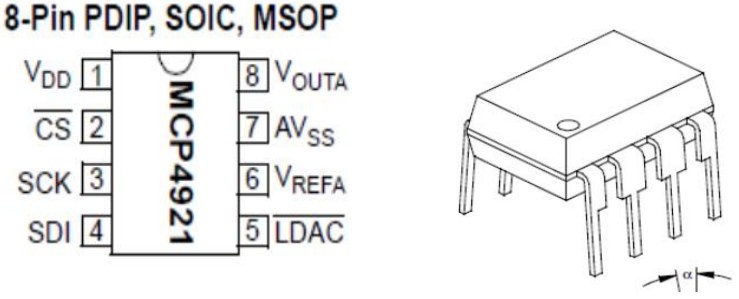


Fig. 4.6 - Convertidor MCP4921

4.5 - Pruebas iniciales

Para el desarrollo del proyecto se ha decidido fabricar una placa de pruebas previa a la elaboración de la placa de circuito impreso final, para corregir posibles errores en la distribución del circuito y realizar las pruebas del software necesarias en ella.

Para la elaboración de la placa de pruebas, se ha optado por la implementación sobre una protoboard, realizada según el diseño del esquemático que se iba a utilizar para la realización del PCB definitivo.

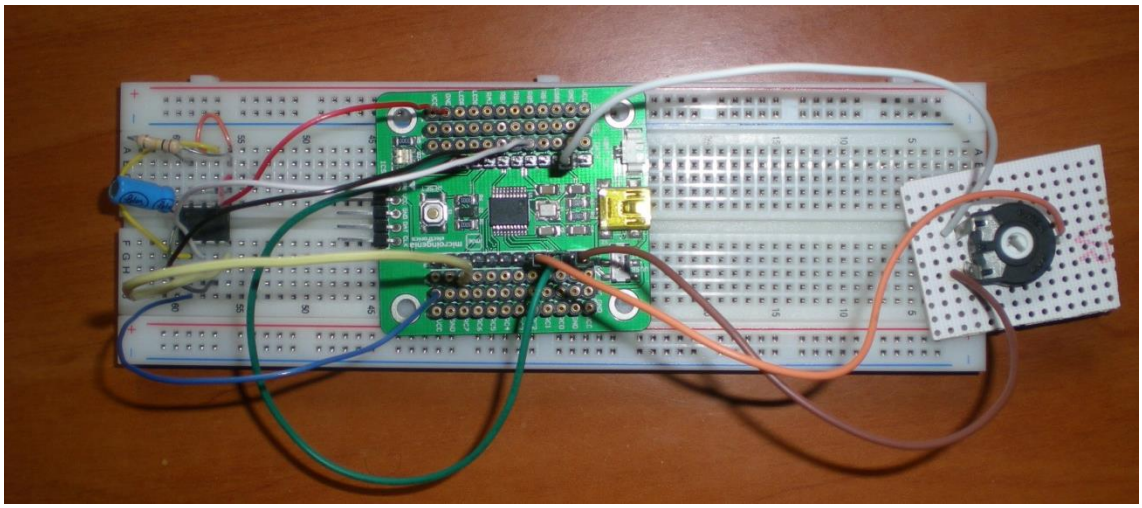


Fig. 4.7 - Protoboard de prueba

4.5.1 - Prueba 1

Como primer test para comprobar el correcto funcionamiento de la comunicación SPI, se plantea un ejercicio básico en el uso de convertidores; el salto de su nivel más bajo (0 bits) a su nivel más alto (4095 bits), y viceversa, con una duración entre salto de 1 milisegundo.

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

signed int16 j=0, i=0;

void MCP4921_init(void)
{
   setup_timer_2(T2_DIV_BY_1,24,1); //Para generar un reloj de 250 KHz
   output_low(MCP4921_SELECT);
   output_low(MCP4921_CLK);
   output_low(MCP4921_DI);
```

```
   i=input(MCP4921_DO);
   setup_spi(SPI_MASTER  |  SPI_CLK_T2|  SPI_L_TO_H  |  SPI_SS_DISABLED
|SPI_XMIT_L_TO_H);
}

void MCP4921_Send(int16 Sample)
{
   int16 SampleBuffer;
   int8 Data = 0;
   output_low(MCP4921_SELECT);

   SampleBuffer= Sample;
   Data=(SampleBuffer>>8) & 0x0F;
   Data |= 0x30;
   spi_write(Data);

   SampleBuffer= 0b0000000011111111 & Sample;
   Data = SampleBuffer;
   spi_write(Data);

   output_high(MCP4921_SELECT);
}

void main ()
{
   usb_cdc_init();
   usb_init();
   usb_wait_for_enumeration();

   MCP4921_init();

   while(1)
   {
      j=0;
      MCP4921_Send(j);
      printf(usb_cdc_putc, "%ld \r\n" ,j);
      delay_ms(1);
      j=4095;
      MCP4921_Send(j);
      printf(usb_cdc_putc, "%ld \r\n" ,j);
      delay_ms(1);
   }
}
```

Programa Prueba-1

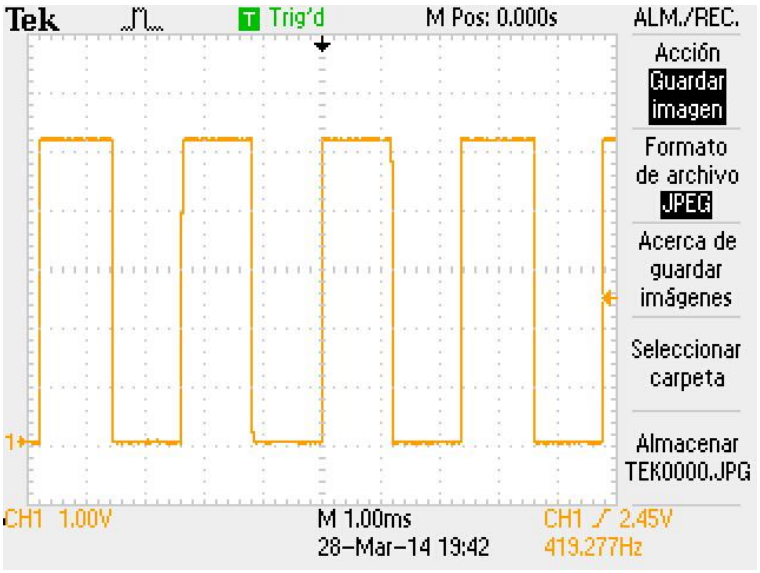


Fig. 4.8 - Resultado prueba-1

4.5.2 - Prueba 2

Como segundo test para comprobar la linealidad del convertidor D/A, se plantea un segundo ejercicio para el uso de convertidores; el paso en rampa desde su nivel más bajo (0 bits) a su nivel más alto (4095 bits).

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

signed int16 j=0,i=0;

void MCP4921_init(void)
{
   setup_timer_2(T2_DIV_BY_1,24,1); //Para generar un reloj de 250 KHz
   output_low(MCP4921_SELECT);
   output_low(MCP4921_CLK);
   output_low(MCP4921_DI);
   i=input(MCP4921_DO);
   setup_spi(SPI_MASTER  |  SPI_CLK_T2|  SPI_L_TO_H  |  SPI_SS_DISABLED
|SPI_XMIT_L_TO_H);
}

void MCP4921_Send(int16 Sample)
{
   int16 SampleBuffer;
   int8 Data = 0;
   output_low(MCP4921_SELECT);

   SampleBuffer= Sample;
   Data=(SampleBuffer>>8) & 0x0F;
   Data |= 0x30;
   spi_write(Data);
```

```
    SampleBuffer= 0b0000000011111111 & Sample;
    Data = SampleBuffer;
    spi_write(Data);

    output_high(MCP4921_SELECT);
}

void main ()
{
    usb_cdc_init();
    usb_init();
    usb_wait_for_enumeration();
    MCP4921_init();
    int16 x=0;

    while(1)
    {
       for(x=0;x<4095;x++)
          {
          MCP4921_Send(x);
          printf(usb_cdc_putc, "%ld \r\n" ,x);
          delay_ms(1);
          }
    }
}
```

Programa Prueba-2

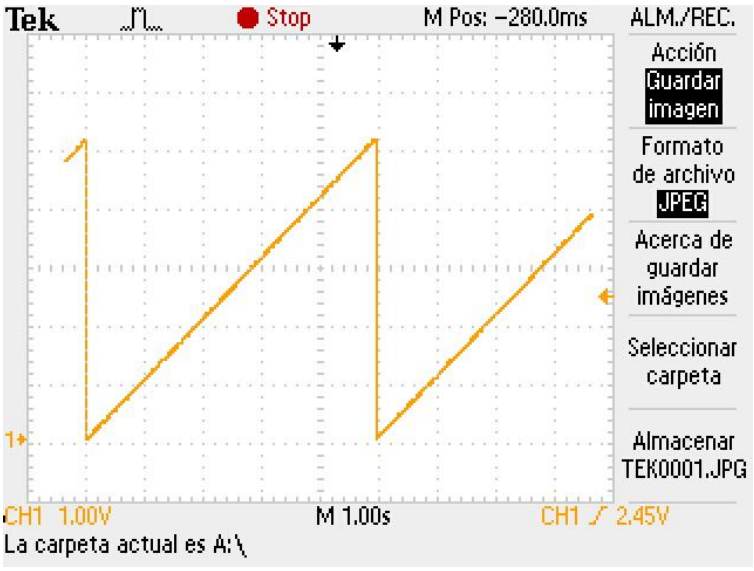


Fig. 4.9 - Resultado prueba-2

4.5.3 - Prueba software final

En este punto se ha programado el PIC con el software final para comprobar la correcta comunicación SPI, y su adecuado funcionamiento.

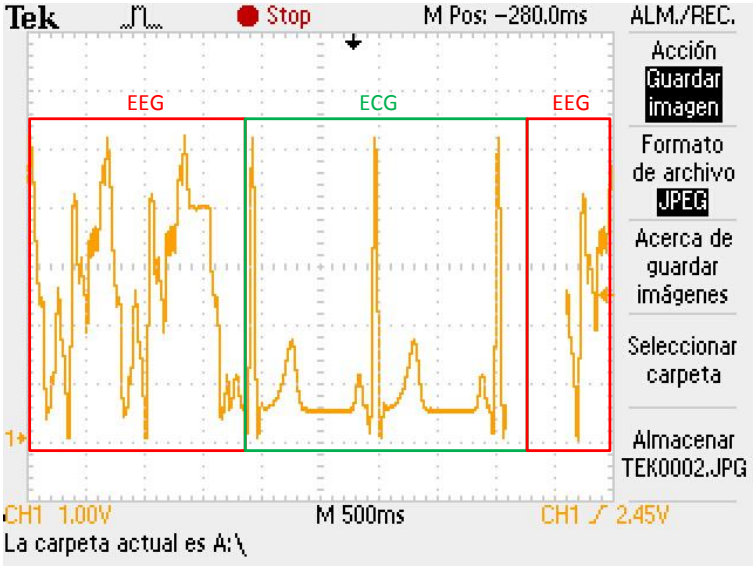


Fig. 4.10 - Resultado de la prueba del software final

4.6 - Esquemático de la solución propuesta

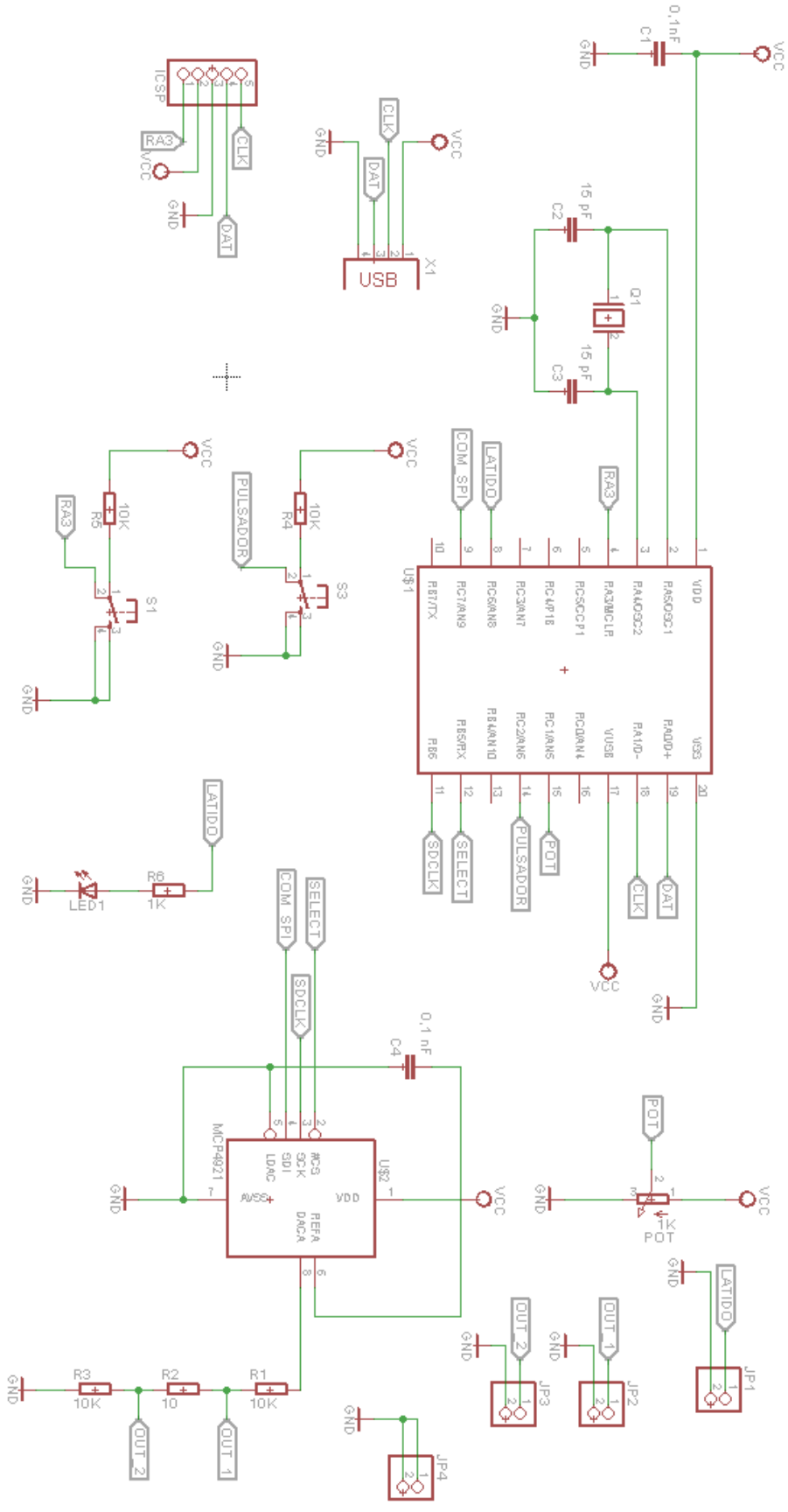


Fig. 4.11 - Esquemático del simulador

Como se puede observar en el esquema, el microcontrolador tiene conectado un cristal de 48 Mhz, que marcará la frecuencia principal de trabajo, con sus respectivos condensadores.

Se ha configurado la patilla /MCLR para poder realizar en tiempo de ejecución del programa un reinicio del mismo y para indicar al microcontrolador en el arranque que entre en modo programación, cargando su bootloader, pudiéndose generar mediante el uso de comunicación ICSP o USB.

Se ha elaborado el montaje de un pulsador externo normalmente abierto, que tras ser pulsado forzará un 0 lógico en la patilla RC2 que provocará la acción del dispositivo. Para ello, se ha dispuesto una resistencia de 10k entre alimentación y la patilla, que mantendrá la misma forzada a un 1 lógico hasta que se active el pulsador.

Se ha realizado el montaje de un potenciómetro de 1K, en el cual sus extremos están conectados a masa y a alimentación, y su patilla central está conectada a la patilla RC1.

Se han configurado las patilas RC7 (comunicación de datos), RB5 (habilitación de comunicación) y RB6 (señal de reloj) del PIC para su comunicación SPI, las cuales son conectadas al pin correspondiente del convertidor.

Se han colocado cuatro tiras de pines conectadas a distintas señales del dispositivo, con objeto de efectuar las oportunas medidas de éstas.

Se ha dispuesto la patilla RC6 como frecuencímetro de la señal de ECG, al cual se le ha añadido un diodo LED para su visualización sin el uso de ningún equipo de medida.

Todo ello junto con los condensadores necesarios para el correcto funcionamiento tanto del convertidor, como del microcontrolador y como el cristal oscilador.

4.7 - PCB, montaje y puesta en marcha

Para la elaboración del PCB se ha elegido su realización mediante una fresadora de prototipado rápido.

4.7.1 - Elaboración del layout del PCB

Tras comprobar que el diseño de la placa y el programa funciona correctamente, se ha procedido a diseñar el layout para realizar la placa de circuito impreso mediante la fresadora de prototipado rápido.

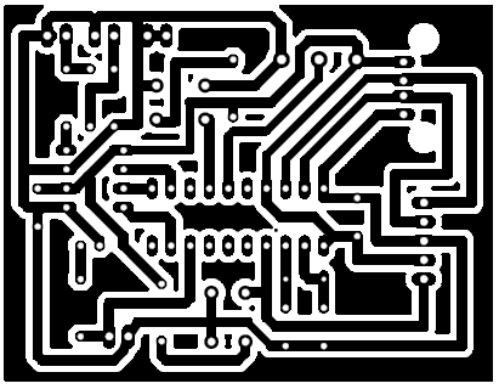


Fig. 4.12 - Layout del PCB

Como se puede observar en la figura, se han dispuesto todos los componentes buscando bastante similitud con el esquemático, a fin de facilitar la comprensión entre ambos.

Se ha intentado utilizar el suficiente espacio como para que la distribución de los componentes sea compacta pero limpia. El ruteado de la placa se ha realizado a una sola cara, con pistas de 0.05 pulgadas de ancho, salvo en algunas partes del circuito, donde por necesidad las pistas son de 0.04 pulgadas.

4.7.2 - Elaboración PCB por fresadora

Para realizar la placa fresada, en primer lugar se han generado con EAGLE tres archivos Gerber. Uno de ellos contiene el contorno de la placa (CAM-Dimension), otro de ellos contiene el layout de las pistas y pads (CAM- Bottom, Pads, Nets), y finalmente el que contiene el layout de las perforaciones (CAM-Drills,Holes).

Para generar los archivos gerber con EAGLE debe seguirse este procedimiento:

En CAM processor se carga el job llamado 'EXCELLON', dentro de él se configuran los parámetros para generar el archivo gerber de las perforaciones.

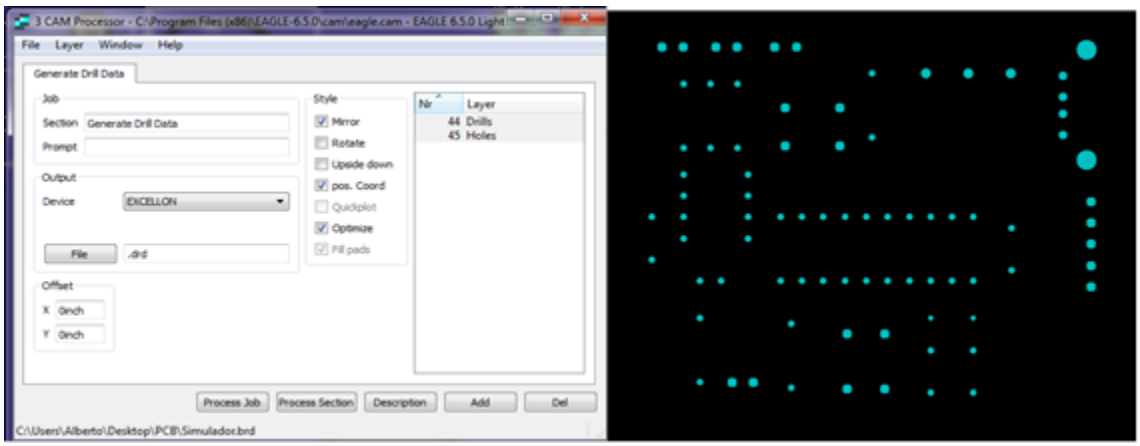


Fig. 4.13 - Gerber taladrado

Acto seguido se carga el job llamado ‘gerber274camx’, dentro de él se configuran los parámetros para generar los archivos gerber del borde de la placa y de la capa ‘bottom’.

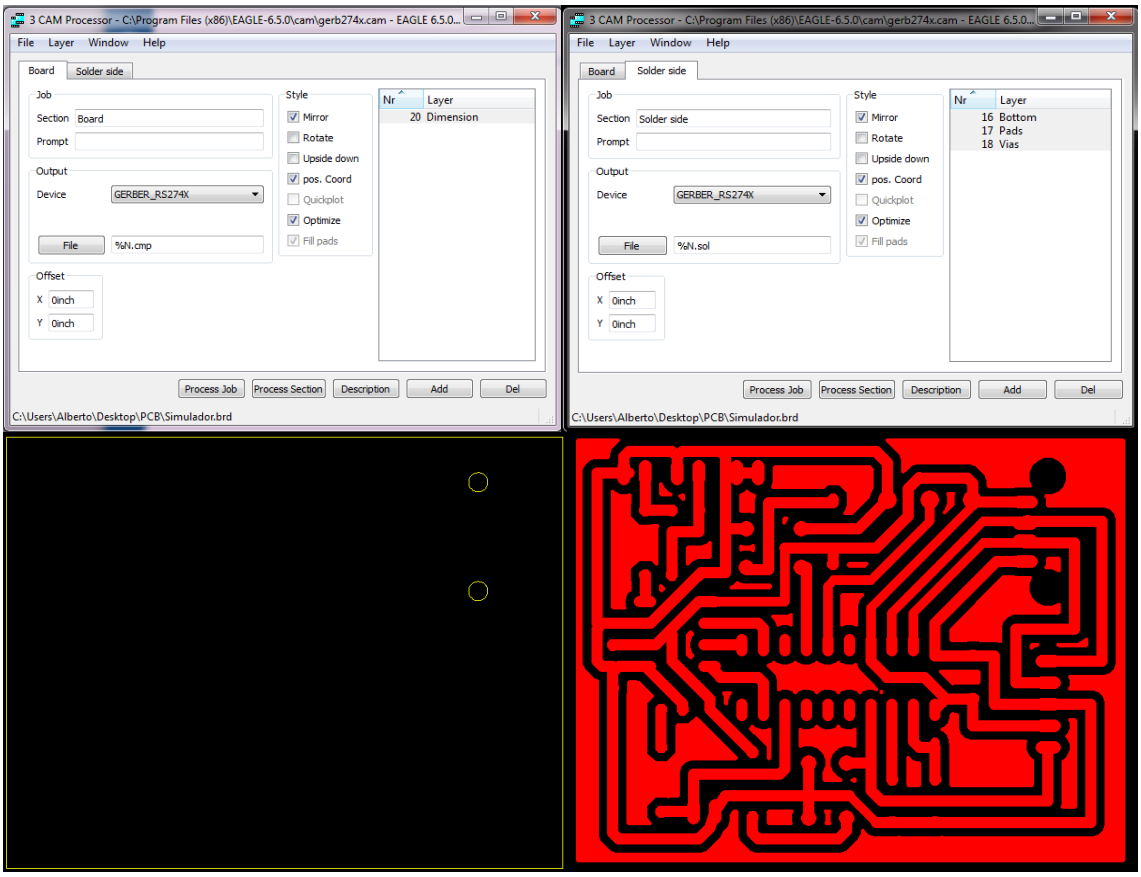


Fig. 4.14 - Gerber perímetro y pistas

Tras preparar los archivos por capas, se ha mandado el proyecto al programa controlador de la fresadora.

Taladrado.drd → Capa DrillUnplated

Pistas.top → Capa Toplayer

Perímetro.asb → Capa Boardoutline

Tras calibrar la fresadora y preparar el portaherramientas con las herramientas adecuadas, se ha iniciado el fresado de la placa.

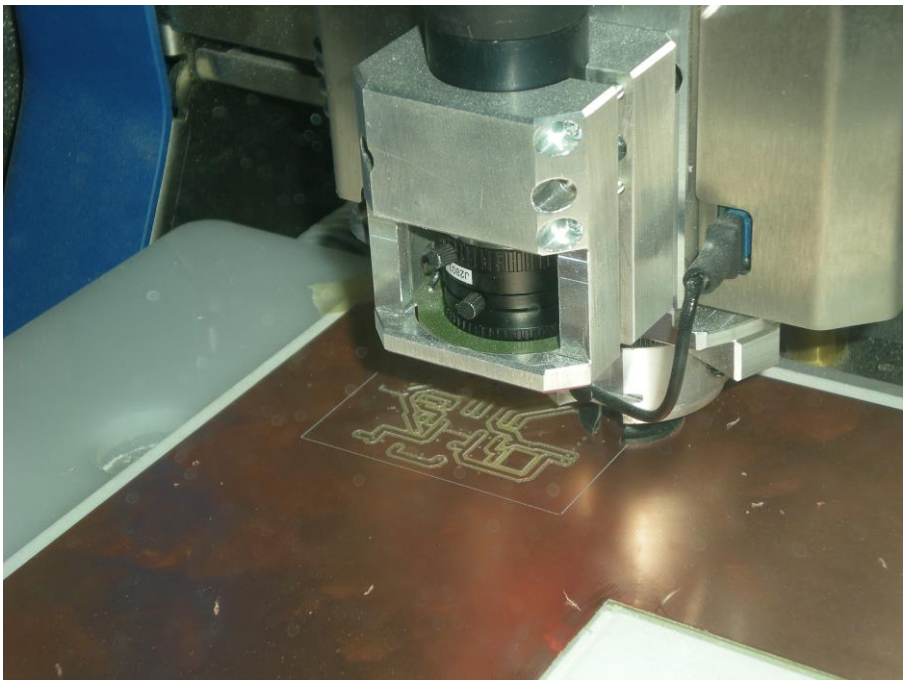


Fig. 4.15 - Fresado de la placa

Tras el fresado del PCB, se ha procedido al soldado de los componentes y a la comprobación de su funcionamiento.

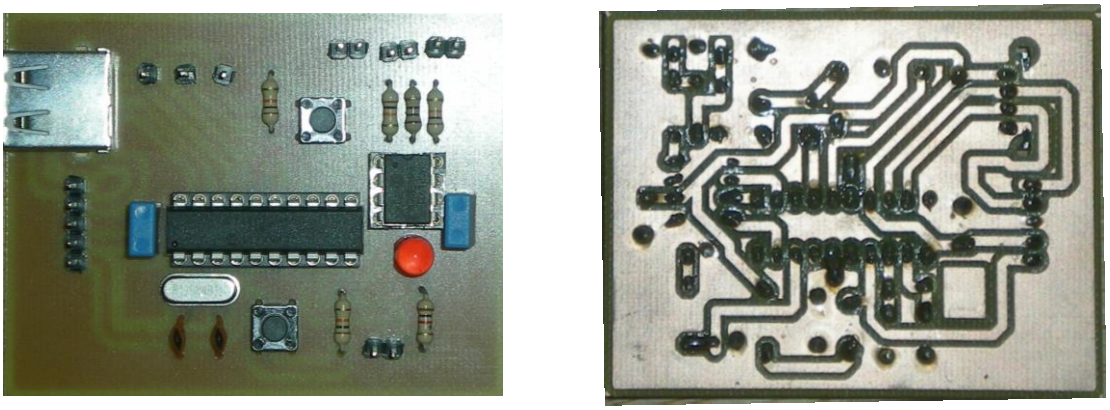


Fig. 4.16 - PCB finalizado, cara de componentes y pistas

Se ha decidido realizar una segunda versión de este simulador, que incluye todos los componentes antes mencionados, junto con elementos de mejora:

- Conexión y circuito acondicionador de batería externa
- LED de alimentación
- Switch selector de alimentación USB o batería
- 1 pulsador de uso de usuario
- 1 pulsador de bootloader

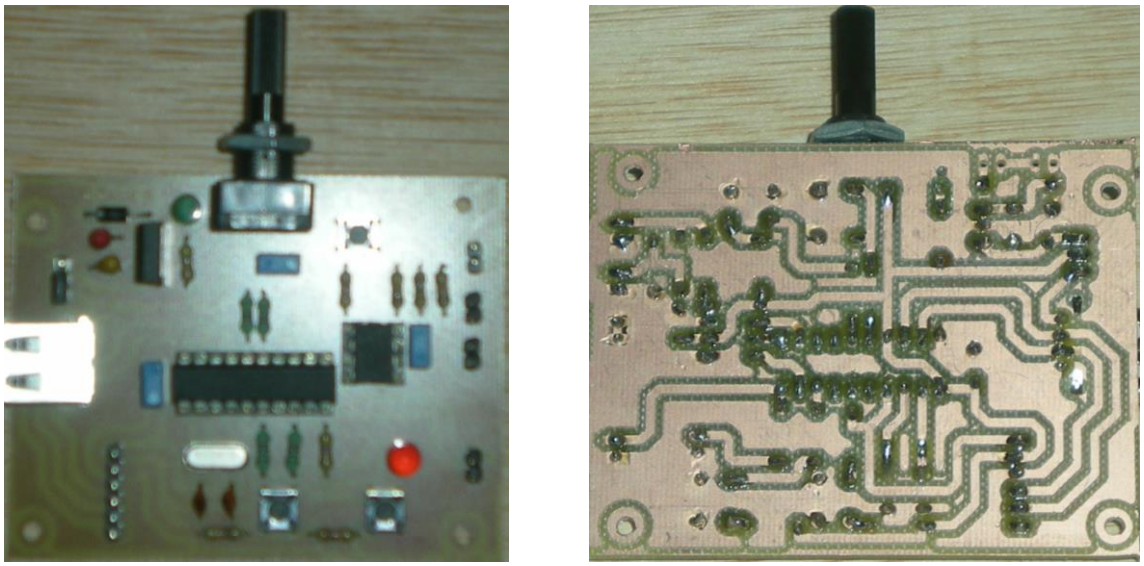


Fig. 4.17 - PCB_v2 finalizado, cara de componentes y pistas

Para la comprobación del correcto funcionamiento de la placa versión2 se ha procedido a programar en el PIC el programa bootloader por ICSP, y posteriormente insertar en el microcontrolador el software final, con los ajustes necesarios para poder ser usado en esta versión, a través del bootloader.

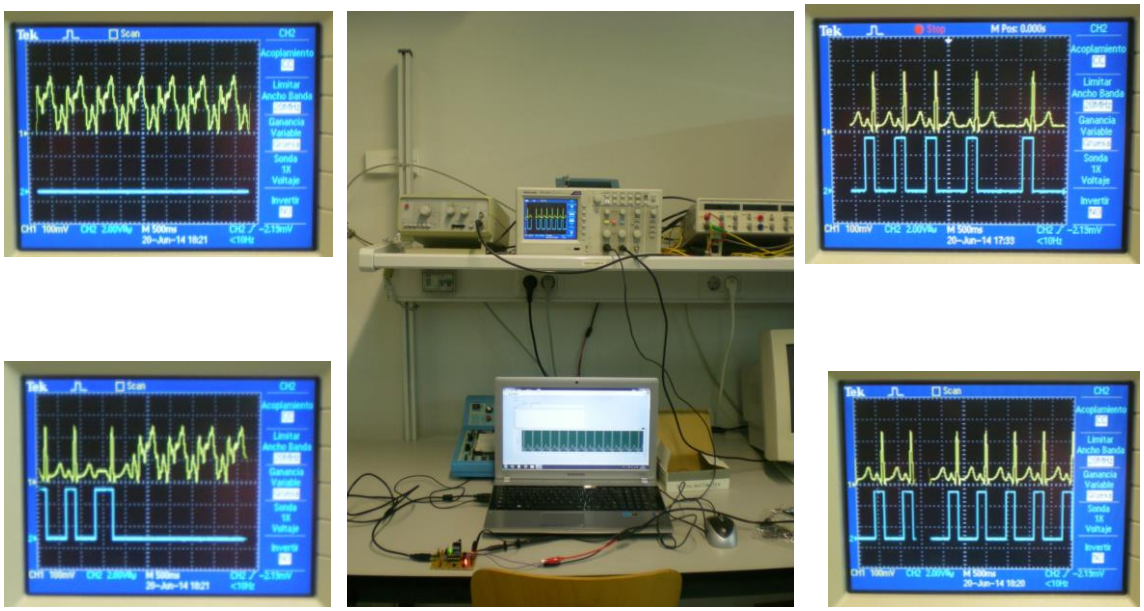


Fig. 4.18 - Puesta en marcha de placa

Capítulo 5

Conclusiones y trabajos futuros

5.1 Conclusiones

En este proyecto se ha llevado a cabo la implementación software, el diseño y la construcción de una placa de circuito impreso que tiene como función recrear una serie de bipotenciales presentes en el cuerpo humano, en este caso se ha trabajado con señales producidas por el cerebro (EEG) y por el corazón (ECG).

Finalmente, destacar que con el trabajo llevado a cabo en este proyecto, aparte de suponer una aplicación real, se han adquirido conocimientos en el diseño electrónico, y más específicamente, en el diseño de simuladores electrónicos en el ámbito clínico.

5.2 Trabajos futuros

Como ocurre siempre que se aborda un proyecto de este tipo, siempre es posible plantear líneas para la mejora y ampliación del trabajo realizado. Entre éstas, cabe destacar las siguientes por su interés:

- La inclusión de una tarjeta de memoria externa, que originaría la posibilidad de incluir cuantas señales se requieran. Esto acarrea un *handicap*, realizar la comunicación entre el microcontrolador y dicha memoria.
- Asimismo, también sería interesante optimizar el código fuente usado, haciéndolo más fluido y simplificado.

Se ha de señalar que el uso de este simulador no sólo es aplicable al ámbito clínico, si no que cualquier tipo de señal cíclica es apta de ser usada en este dispositivo. Como ejemplo, este

dispositivo puede ser utilizado como un generador de señales (sinusoidales, triangulares, cuadradas, etc.), de baja potencia y de frecuencia variable.

Como comentario final, señalar que la placa del presente proyecto es una versión 1.0, en la cual el objetivo era obtener un soporte para comprobar el correcto funcionamiento del ejercicio programado y para poder ver el resultado de la programación. Como futuro trabajo se deja la creación de una versión 2.0 en la cual se ve reducido el tamaño de la placa, utilizando componentes smd.

Anexo A

Bibliografía y páginas web

[1] Compilador C CCS y Simulador Proteus para Microcontroladores PIC.

Eduardo García Breijo

ISBN-10: 8426714293 ISBN-13: 978-8426714299

[2] Bioinstrumentación

John G. Webster

ISBN-10: 0471263273 ISBN-13: 978-0471263272

[3] Adafruit Menta ECG Simulator

[4] Tutorial EAGLE Version 6

[5] Design and development of medical electronic instrumentation

David Prutchi y Michael Norris

ISBN 0-471-67623-3

[6] http://physionet.org/physiobank/database/ptbdb/

[7] http://physionet.org/physiobank/database/capslpdb/

[8] Matlab: una introducción con ejemplos prácticos.

Amos Gilat

ISBN-10: 84-291-5035-8 ISBN-13: 978-84-291-5035-3

Anexo B

Presupuesto

Costes de materiales de ejecución

Descripción	Cant.	$PVP_{unitario}$	Total
PCB cobre, simple cara 300x300	1	3,00 €	3,00 €
Cristal 12.0 MHZ	1	0,42 €	0,42 €
RESISTENCIA 1K	1	0,10 €	0,10 €
RESISTENCIA 10K	4	0,09 €	0,34 €
RESISTENCIA 10	1	0,09 €	0,09 €
CONDENSADOR 15 PF	2	0,26 €	0,52 €
CONDENSADOR 0,1UF	2	0,22 €	0,44 €
CONECTOR USB	1	0,90 €	0,90 €
LED rojo	1	0,15 €	0,15 €
POTENCIOMETRO 1K	1	0,40 €	0,40 €
PULSADOR	2	1,00 €	2,00 €
CONECTORES MACHO 20	1	0,70 €	0,70 €
PIC18F14K50	1	2,71 €	2,71 €
MCP4921	1	1,43 €	1,43 €
		Materiales Ejecución	**13,20€ /UNIDAD**

Costes de mano de obra

Descripción	€/hora	Horas	Total
INGENIERO DISEÑO HARDWARE	15,00€	80	1.200,00 €
INGENIERO DISEÑO FIRMWARE	15,00€	30	450,00 €
INGENIERO DISEÑO SOFTWARE	15,00€	8	120,00 €
		Mano de obra	**1.650,00€**

Costes de materiales de desarrollo

Descripción	Cant.	$PVP_{unitario}$	Total
ESTACIÓN TRABAJO INFORMATICO	1	2.000,00 €	2.000,00 €
KIT DE PROGRAMACION µC	1	600,00 €	600,00 €
KIT DE SOLDADOR	1	100,00 €	100,00 €
LICENCIA PICC	4	600,00 €	600,00 €
FRESADORA LPFK 602	1	12.000,00 €	12.000,00 €
		Materiales Desarrollo	**15.300,00 €**

Costes totales

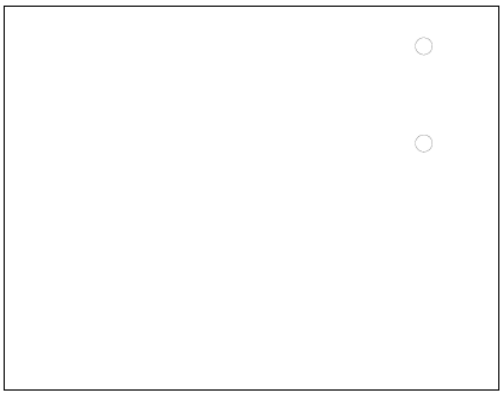
Descripción	Total
MATERIALES EJECUCIÓN	13,20 €
MANO DE OBRA	1.650,00 €
MATERIALES DESARROLLO	15.300,00 €
	16.963,20€

Supuesto de fabricación

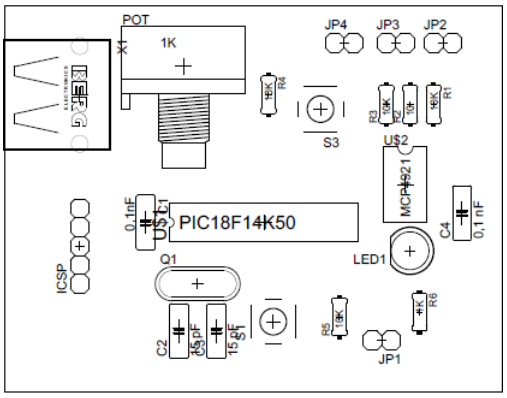
PCB Prototipo	Precio
MATERIALES EJECUCIÓN	13,20 € / UNIDAD
TOTAL 1 UNIDAD PROTOTIPO	13,20 € x 0,5 = 6,60 €
TOTAL 100 UNIDADES PROTOTIPO	660,00 €
BENEFICIO INDUSTRIAL (20%)	132,00 €
	1,32 € / UNIDAD
PVP (IVA 21%)	**798,60 €** **7,99 € / UNIDAD**

Anexo C

Planos

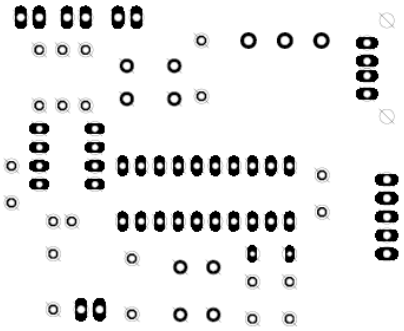


Autor:	Alberto García Salinas	
Título:	Geber - Perímetro	
Fecha:	28/05/14	
UNIVERSIDAD POLITÉCNICA DE CARTAGENA		

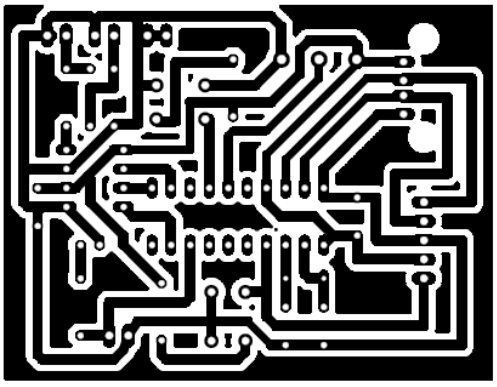


Autor:	Alberto García Salinas
Título:	Posicionamiento componentes
Fecha:	28/05/14

UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Autor:	Alberto García Salinas	
Título:	Gerber – Pads y taladrado	
Fecha:	28/05/14	
UNIVERSIDAD POLITÉCNICA DE CARTAGENA		



Autor:	Alberto García Salinas	
Título:	Layout PCB	
Fecha:	28/05/14	
UNIVERSIDAD POLITÉCNICA DE CARTAGENA		

Anexo D

Datasheet

Los dataheets de los principales dispositivos que componen el proyecto se encuentran en un archivo adjunto a este proyecto.

Anexo E

Código fuente

En este apartado se muestra el código de programación, en lenguaje C, finalmente usado para ser volcado en el microcontrolador.

```
#include "config18F14K50Trainer.c"
#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

signed int16 i=0, j=0, xpot=0, reposo=0;
int8 indice = 0;

const int16 DerI_form[543]={
381, 384, 387, 390, 393, 396, 399, 403, 407, 411,
415, 420, 424, 429, 435, 441, 447, 454, 461, 469,
477, 485, 495, 504, 515, 526, 537, 549, 562, 575,
588, 602, 616, 630, 645, 660, 675, 690, 705, 720,
734, 749, 763, 776, 789, 801, 813, 823, 833, 842,
849, 856, 861, 865, 867, 868, 868, 866, 863, 859,
853, 846, 837, 827, 816, 803, 790, 775, 760, 744,
728, 710, 693, 675, 657, 639, 622, 604, 588, 571,
556, 541, 527, 514, 502, 491, 481, 473, 465, 459,
454, 450, 447, 445, 444, 443, 443, 443, 444, 445,
445, 446, 445, 444, 442, 439, 435, 430, 423, 414,
404, 392, 378, 362, 345, 326, 306, 284, 261, 238,
215, 191, 168, 146, 126, 107, 92, 80, 72, 68,
70, 78, 93, 115, 145, 183, 231, 288, 354, 431,
518, 614, 721, 838, 965, 1101, 1245, 1396, 1555, 1720,
1889, 2062, 2236, 2412, 2586, 2759, 2927, 3089, 3244, 3391,
3527, 3651, 3762, 3859, 3940, 4005, 4053, 4083, 4095, 4088,
4063, 4020, 3960, 3882, 3789, 3680, 3558, 3423, 3276, 3120,
2956, 2785, 2610, 2431, 2251, 2071, 1893, 1718, 1547, 1382,
1224, 1074, 932, 800, 677, 565, 463, 372, 292, 222,
162, 113, 73, 43, 21, 6, 0, 0, 5, 16,
32, 51, 73, 98, 125, 152, 181, 210, 238, 266,
292, 318, 342, 364, 384, 402, 418, 432, 443, 453,
461, 467, 471, 473, 474, 473, 472, 469, 466, 461,
456, 451, 446, 440, 435, 429, 424, 419, 414, 410,
406, 403, 400, 398, 396, 395, 394, 394, 394, 395,
396, 397, 399, 401, 403, 405, 408, 410, 413, 415,
418, 420, 423, 425, 427, 429, 431, 433, 434, 436,
437, 438, 439, 440, 441, 441, 442, 442, 443, 443,
443, 444, 444, 444, 445, 445, 446, 446, 447, 448,
```

```
449, 449, 450, 452, 453, 454, 456, 457, 459, 461,
463, 465, 467, 469, 471, 473, 476, 478, 481, 483,
486, 488, 491, 493, 496, 499, 501, 504, 507, 509,
512, 515, 518, 521, 524, 527, 530, 533, 537, 540,
544, 548, 551, 555, 559, 564, 568, 572, 577, 582,
587, 591, 596, 602, 607, 612, 617, 623, 628, 634,
639, 645, 651, 656, 662, 668, 674, 680, 686, 693,
699, 706, 712, 719, 726, 733, 740, 748, 755, 763,
771, 779, 787, 796, 805, 813, 822, 832, 841, 850,
860, 869, 879, 889, 899, 909, 919, 930, 940, 951,
961, 972, 982, 993, 1004, 1015, 1026, 1036, 1047, 1058,
1070, 1081, 1092, 1103, 1114, 1125, 1136, 1147, 1157, 1168,
1178, 1189, 1199, 1209, 1219, 1228, 1237, 1246, 1254, 1263,
1270, 1277, 1284, 1291, 1296, 1302, 1306, 1310, 1314, 1316,
1318, 1320, 1320, 1320, 1319, 1317, 1315, 1311, 1307, 1302,
1296, 1290, 1282, 1274, 1264, 1255, 1244, 1232, 1220, 1207,
1193, 1179, 1164, 1148, 1132, 1115, 1098, 1080, 1063, 1044,
1026, 1007, 988, 968, 949, 930, 910, 891, 871, 852,
833, 814, 795, 777, 759, 741, 724, 707, 690, 674,
658, 643, 628, 614, 601, 587, 575, 563, 551, 540,
530, 520, 511, 502, 494, 486, 479, 472, 466, 460,
455, 450, 445, 441, 437, 433, 430, 427, 424, 422,
419, 417, 415, 413, 412, 410, 408, 407, 405, 404,
402, 401, 399, 398, 396, 395, 393, 391, 389, 387,
385, 383, 381};

const int16 EEG_form[543]={
737, 771, 808, 849, 892, 937, 984, 1033, 1083, 1135,
1187, 1239, 1292, 1344, 1397, 1449, 1501, 1552, 1602, 1651,
1699, 1745, 1790, 1831, 1871, 1906, 1938, 1966, 1989, 2007,
2019, 2025, 2025, 2019, 2007, 1990, 1967, 1938, 1906, 1869,
1829, 1787, 1742, 1697, 1652, 1607, 1563, 1522, 1483, 1448,
1417, 1390, 1366, 1347, 1331, 1318, 1308, 1300, 1293, 1286,
1280, 1273, 1264, 1254, 1241, 1226, 1208, 1188, 1164, 1139,
1111, 1081, 1050, 1018, 986, 953, 920, 888, 857, 826,
796, 767, 739, 711, 684, 657, 629, 600, 569, 537,
503, 467, 428, 386, 343, 297, 251, 206, 161, 119,
81, 49, 24, 7, 0, 3, 18, 46, 86, 139,
205, 284, 374, 476, 588, 709, 839, 975, 1117, 1262,
1411, 1560, 1709, 1857, 2001, 2141, 2275, 2403, 2524, 2636,
2739, 2834, 2918, 2993, 3059, 3115, 3163, 3202, 3234, 3259,
3278, 3292, 3301, 3307, 3310, 3310, 3308, 3304, 3299, 3291,
3282, 3271, 3256, 3239, 3218, 3192, 3161, 3125, 3083, 3034,
2980, 2920, 2854, 2785, 2712, 2638, 2563, 2490, 2420, 2354,
2294, 2242, 2198, 2163, 2137, 2122, 2115, 2118, 2130, 2148,
2173, 2203, 2236, 2272, 2307, 2342, 2375, 2404, 2429, 2448,
2461, 2468, 2468, 2462, 2450, 2432, 2409, 2382, 2352, 2320,
2287, 2253, 2221, 2191, 2163, 2138, 2116, 2098, 2082, 2069,
2058, 2049, 2039, 2030, 2019, 2006, 1990, 1971, 1947, 1920,
1888, 1852, 1812, 1769, 1723, 1677, 1631, 1586, 1545, 1509,
1479, 1458, 1446, 1445, 1455, 1479, 1514, 1562, 1622, 1692,
1771, 1858, 1950, 2046, 2143, 2239, 2331, 2418, 2498, 2570,
2632, 2683, 2724, 2753, 2771, 2778, 2776, 2765, 2747, 2723,
2694, 2662, 2630, 2597, 2566, 2538, 2515, 2496, 2484, 2478,
2478, 2486, 2499, 2519, 2545, 2574, 2608, 2643, 2679, 2715,
2750, 2782, 2811, 2835, 2854, 2867, 2874, 2875, 2870, 2859,
2842, 2821, 2795, 2766, 2735, 2703, 2671, 2640, 2612, 2587,
2567, 2551, 2542, 2538, 2540, 2547, 2560, 2578, 2599, 2624,
2651, 2679, 2709, 2739, 2768, 2797, 2826, 2853, 2880, 2907,
2933, 2960, 2988, 3017, 3048, 3082, 3117, 3155, 3195, 3238,
3282, 3327, 3373, 3420, 3465, 3508, 3549, 3586, 3619, 3646,
```

```
3667, 3682, 3690, 3691, 3685, 3673, 3656, 3634, 3608, 3581,
3553, 3525, 3500, 3479, 3462, 3451, 3447, 3450, 3460, 3477,
3501, 3532, 3569, 3610, 3656, 3704, 3753, 3803, 3852, 3899,
3943, 3983, 4017, 4046, 4068, 4084, 4093, 4095, 4089, 4078,
4060, 4037, 4010, 3978, 3944, 3907, 3869, 3831, 3792, 3755,
3719, 3684, 3652, 3621, 3591, 3564, 3537, 3510, 3484, 3456,
3427, 3396, 3362, 3325, 3284, 3239, 3190, 3136, 3078, 3017,
2952, 2884, 2815, 2745, 2675, 2607, 2542, 2480, 2423, 2372,
2325, 2285, 2250, 2221, 2197, 2177, 2161, 2147, 2135, 2123,
2112, 2099, 2086, 2071, 2054, 2035, 2015, 1993, 1970, 1947,
1924, 1901, 1879, 1859, 1842, 1827, 1815, 1807, 1802, 1802,
1804, 1810, 1820, 1832, 1845, 1861, 1876, 1891, 1904, 1915,
1922, 1924, 1921, 1910, 1892, 1866, 1831, 1788, 1735, 1674,
1604, 1527, 1443, 1353, 1259, 1162, 1063, 964, 867, 773,
685, 602, 528, 463, 407, 361, 326, 300, 284, 277,
277, 283, 293, 307, 323, 339, 355, 368, 379, 387,
392, 393, 391, 387, 380, 372, 364, 357, 351, 347,
346, 349, 356, 368, 384, 404, 429, 458, 489, 524,
560, 598, 635, 672, 707, 739, 769, 794, 815, 831,
843, 849, 851, 848, 841, 832, 819, 805, 790, 775,
760, 747, 737};

void MCP4921_init(void)
{
   setup_timer_2(T2_DIV_BY_1,24,1); //Para generar un reloj de 250 KHz
   output_low(MCP4921_SELECT);
   output_low(MCP4921_CLK);
   output_low(MCP4921_DI);
   i=input(MCP4921_DO);
   setup_spi(SPI_MASTER  |  SPI_CLK_T2|  SPI_L_TO_H  |  SPI_SS_DISABLED
|SPI_XMIT_L_TO_H);
}

void MCP4921_Send(int16 Sample)
{
   int16 SampleBuffer;
   int8 Data = 0;

   output_low(MCP4921_SELECT);

   SampleBuffer= Sample;
   Data=(SampleBuffer>>8) & 0x0F;
   Data |= 0x30;
   spi_write(Data);


   SampleBuffer= 0b0000000011111111 & Sample;
   Data = SampleBuffer;
   spi_write(Data);

   output_high(MCP4921_SELECT);
}

void recorrerECG(void)
{
   xpot=read_adc();
   reposo=1024-xpot;
   if (reposo <0) reposo =0;
   if (reposo >1024) reposo = 1024;

   for(i=0;i<543;i++)
```

```
      {
         j = DerI_form[i];
         MCP4921_Send(j);
               if(i<180) output_high(PIN_C6); //generar minipulso
               else output_low(PIN_C6);
         printf(usb_cdc_putc, "%li \r\n" ,j);
         delay_ms(1);
      }
   for(i=0;i<reposo;i++)
      {
         MCP4921_Send(j);
         printf(usb_cdc_putc,"%li \r\n",j);
         delay_ms(1);
      }
}

void recorrerEEG(void)
{
      for(i=0;i<543;i++)
      {
         j = EEG_form[i];
         MCP4921_Send(j);
         printf(usb_cdc_putc, "%li \r\n" ,j);
         delay_ms(1);
      }
}

void main(void)
{
    ext_int_edge(2,L_TO_H);
    enable_interrupts(INT_EXT2);
    enable_interrupts(GLOBAL);

    usb_cdc_init();
    usb_init();
    usb_wait_for_enumeration();

    MCP4921_init();

    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(sAN5|VSS_VDD );
    set_adc_channel(5);

   while(TRUE)
   {
      switch(indice)
      {
         case 0:
            recorrerECG();
         break;

         case 1:
            recorrerEEG();
         break;

         default:
            indice=0;
         break;
      }
   }
}
```

```
#INT_EXT2
void  interrupt_RC2()//accionar  pulsador  conectado  a  RC2  para  que  se
lea una u otra señal de las tablas de memoria (ECG / EEG)
{
   indice++;
   delay_ms(175);
     }
```

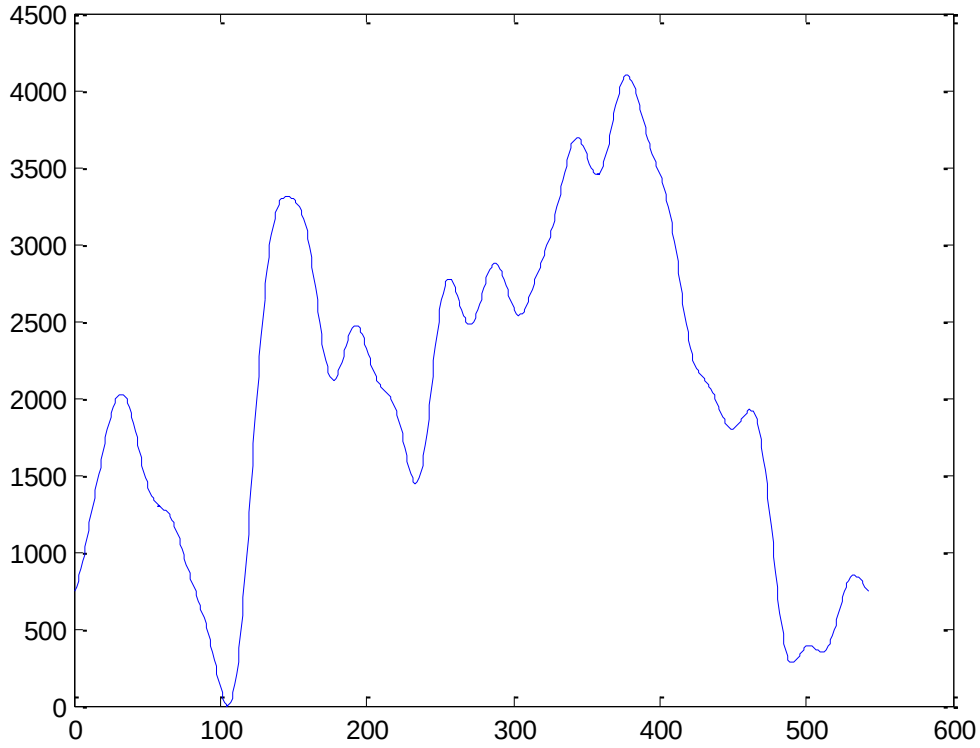
Anexo F

Señales

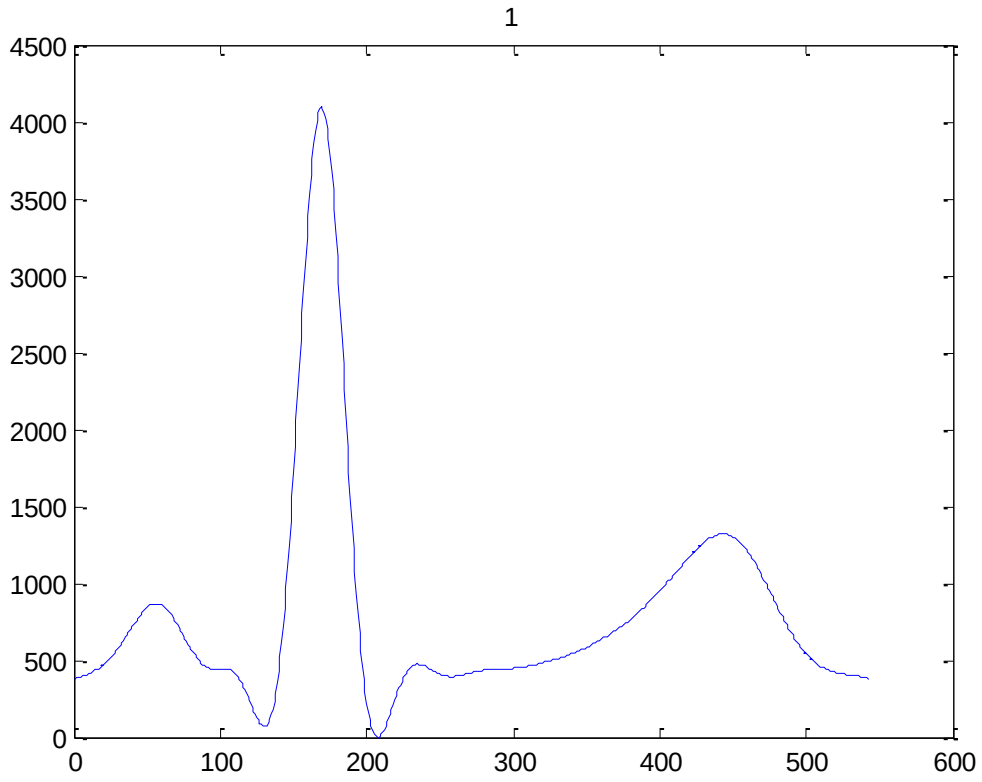
En este anexo del proyecto se muestran las señales de los distintos tipos de biopontenciales con los que se trabaja.

Estas señales son: las 15 derivaciones propias de un electrocardiograma (ECG), una señal de electroencefalograma (EEG) y, además, una señal de referencia usada al principio de comenzar con el presente proyecto, cuyo fin es comprobar la correcta lectura de la misma.

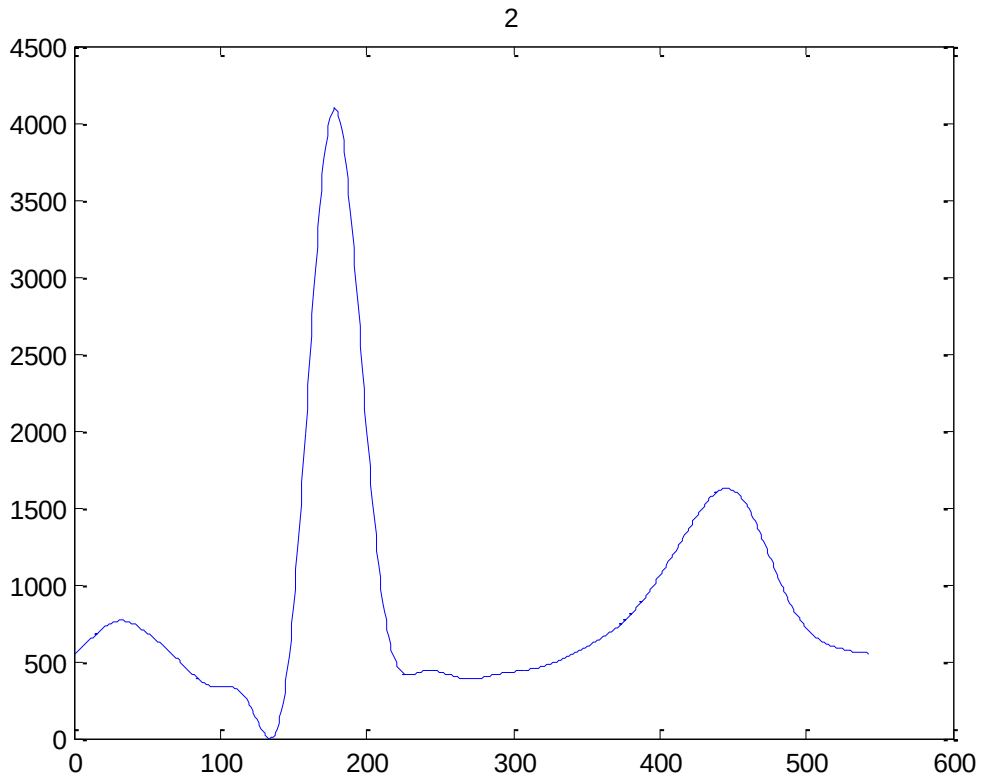
F.1 Gráficas



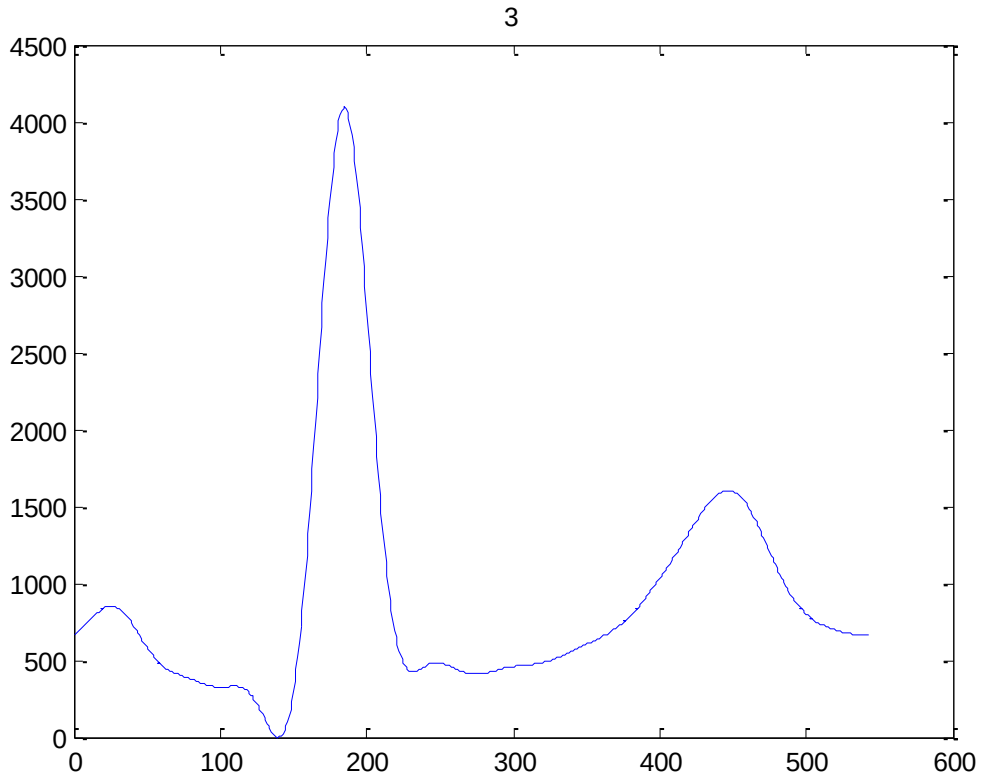
Señal de electroencefalograma



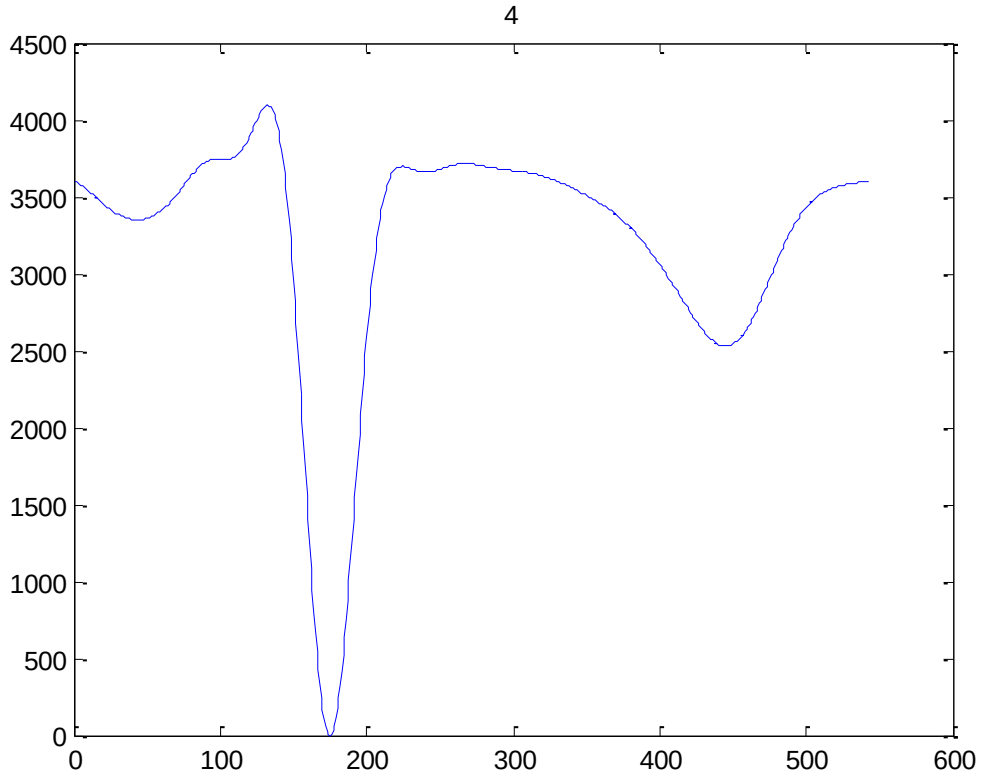
Señal de la derivación I de un ECG



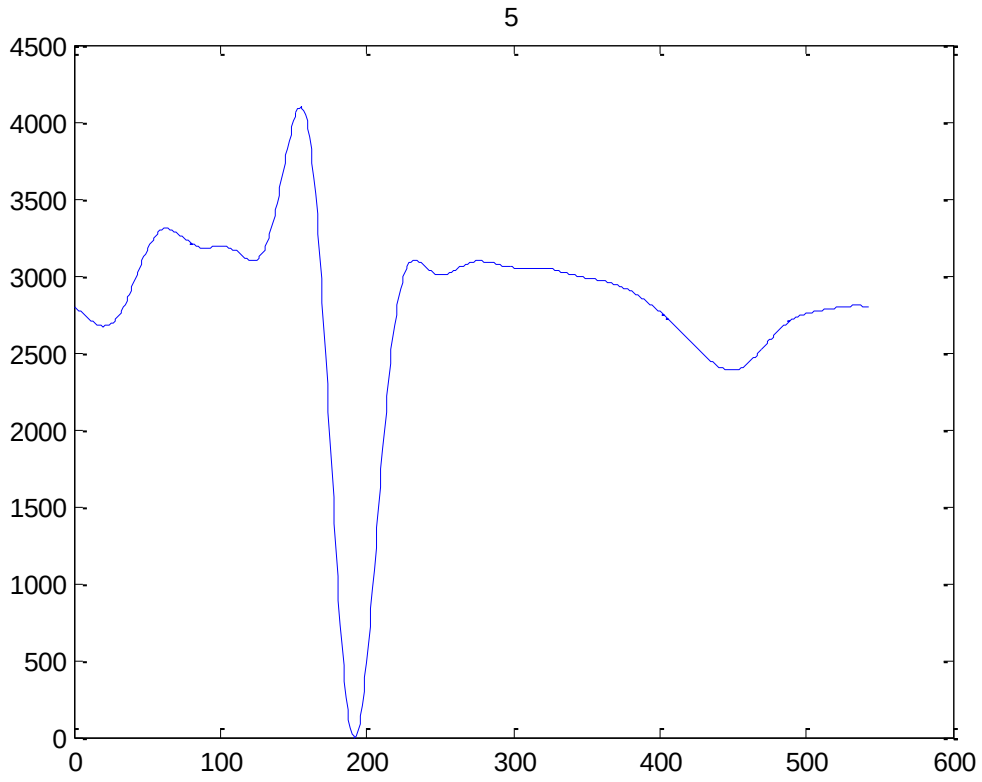
Señal de la derivación II de un ECG



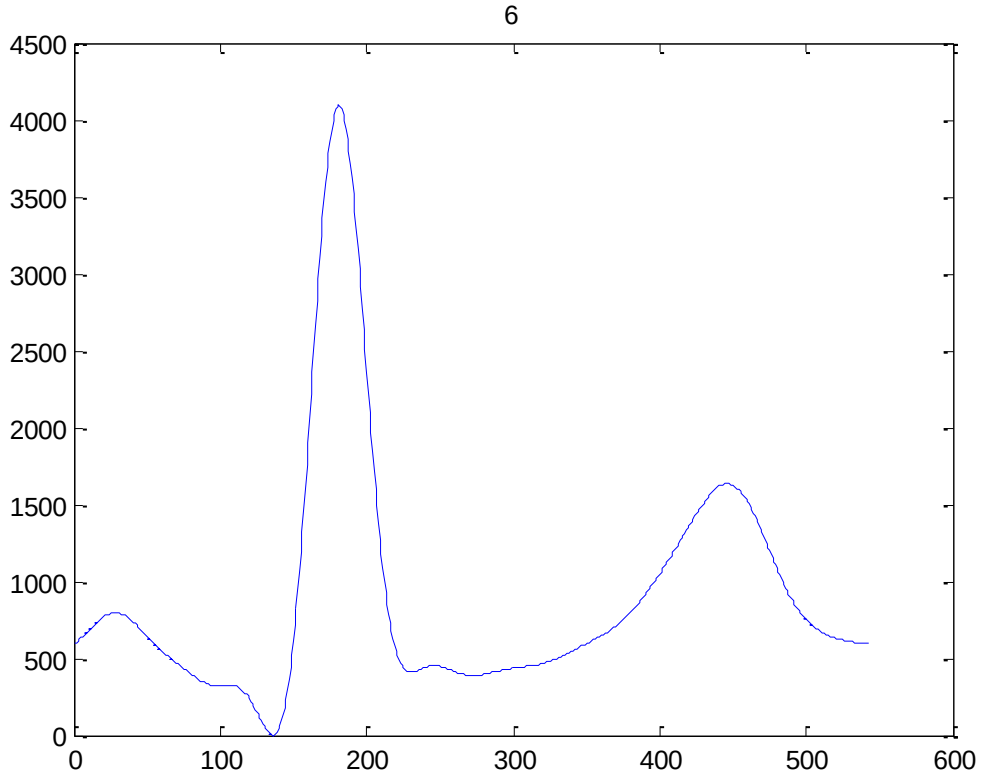
Señal de la derivación III de un ECG



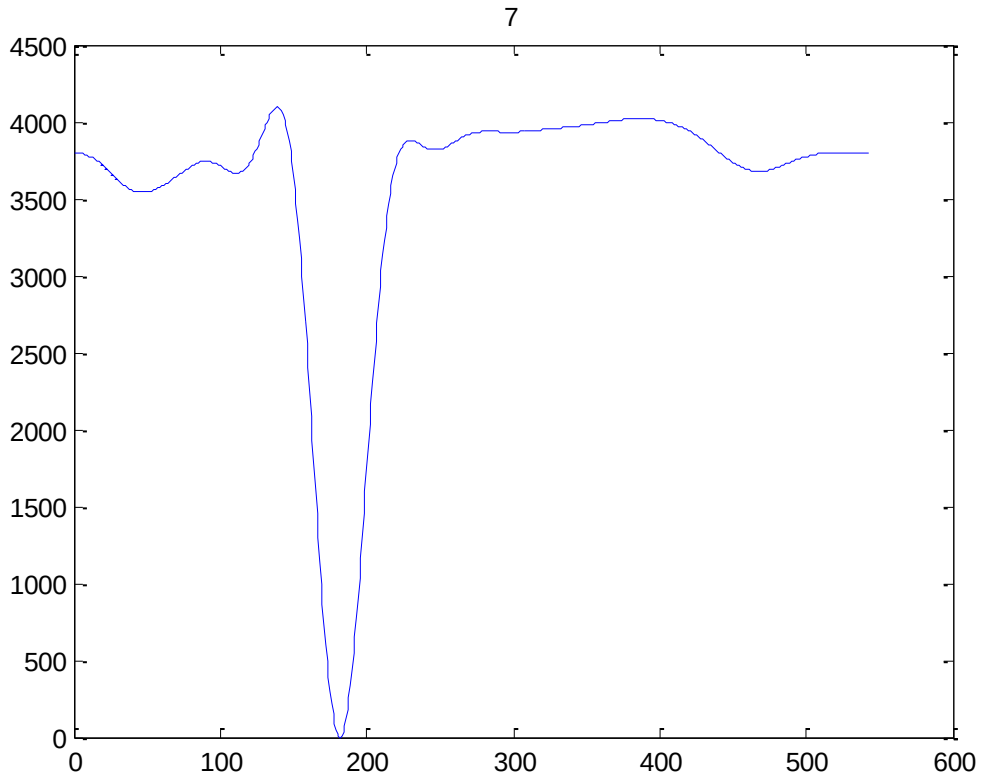
Señal de la derivación avr de un ECG



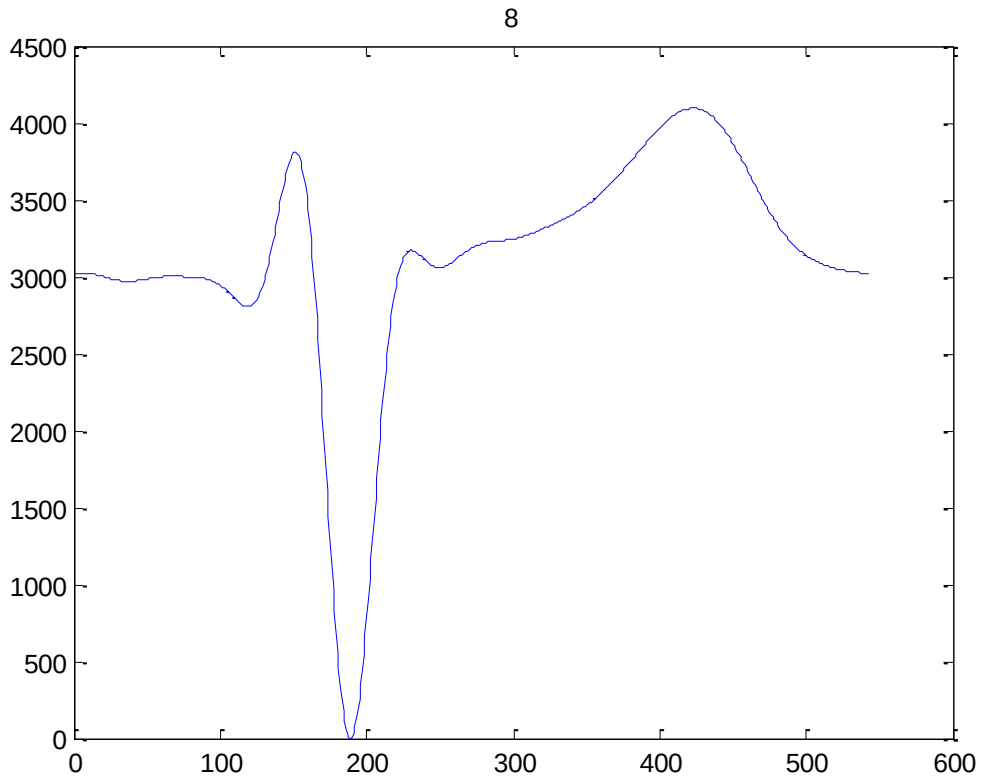
Señal de la derivación avl de un ECG



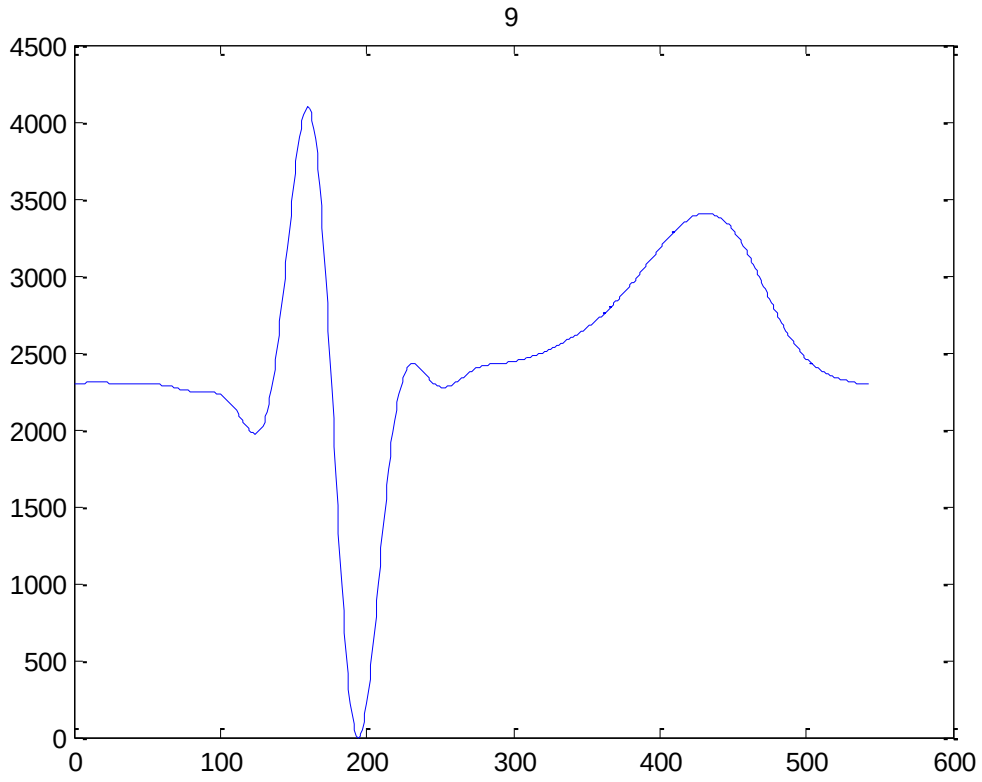
Señal de la derivación avf de un ECG



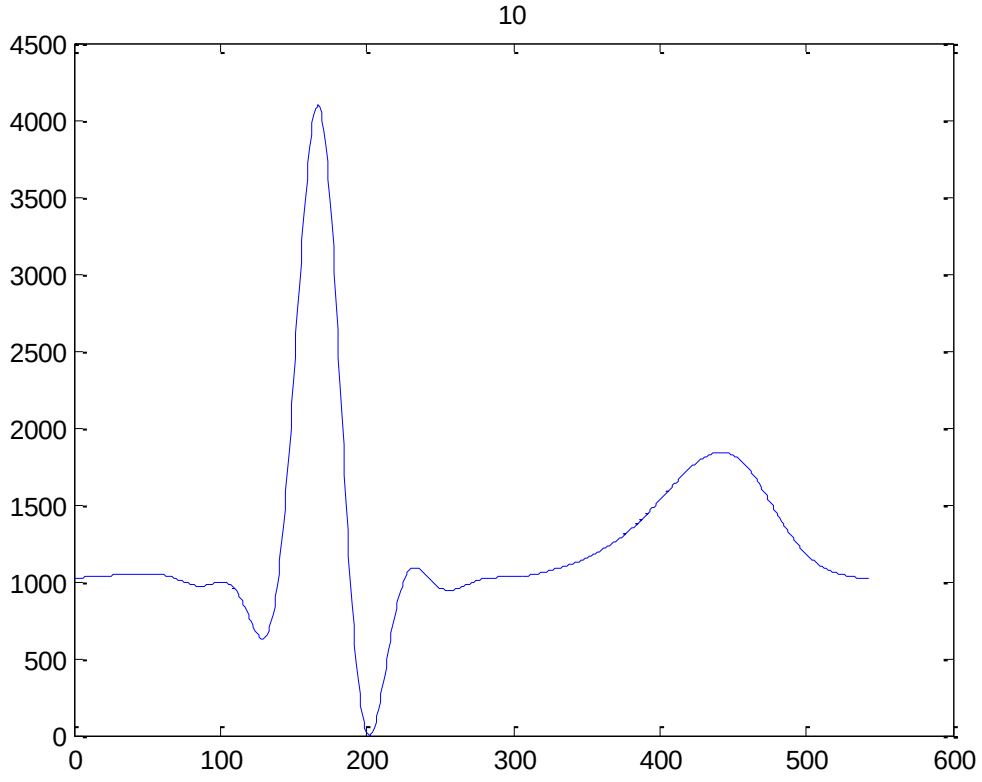
Señal de la derivación v1 de un ECG



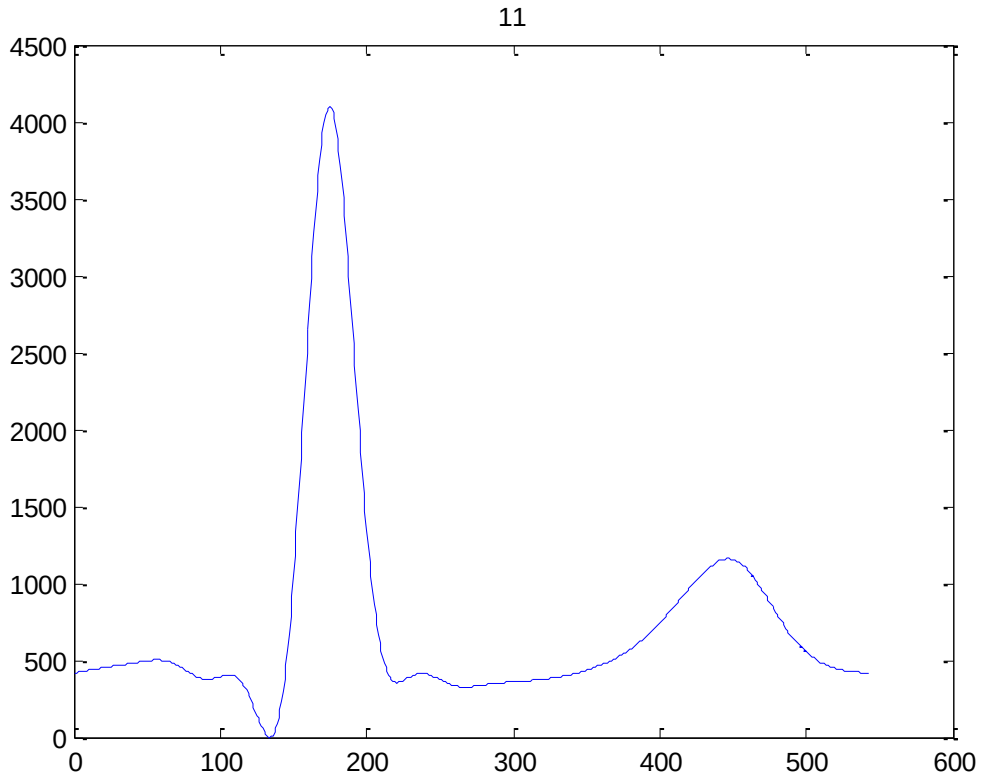
Señal de la derivación v2 de un ECG



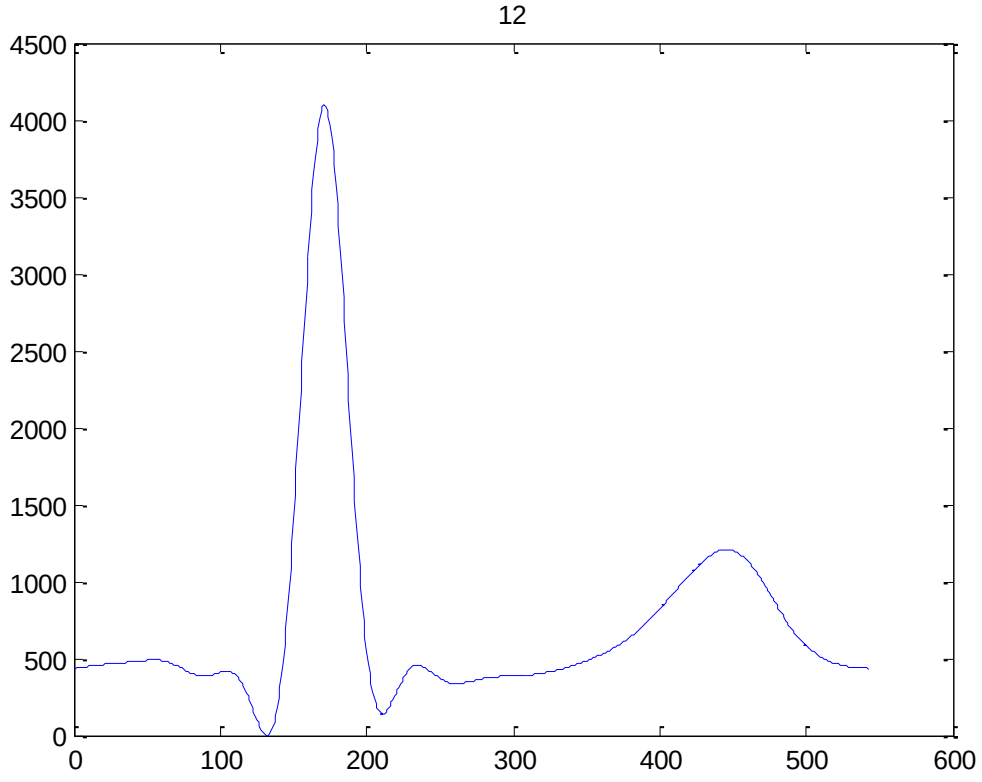
Señal de la derivación v3 de un ECG



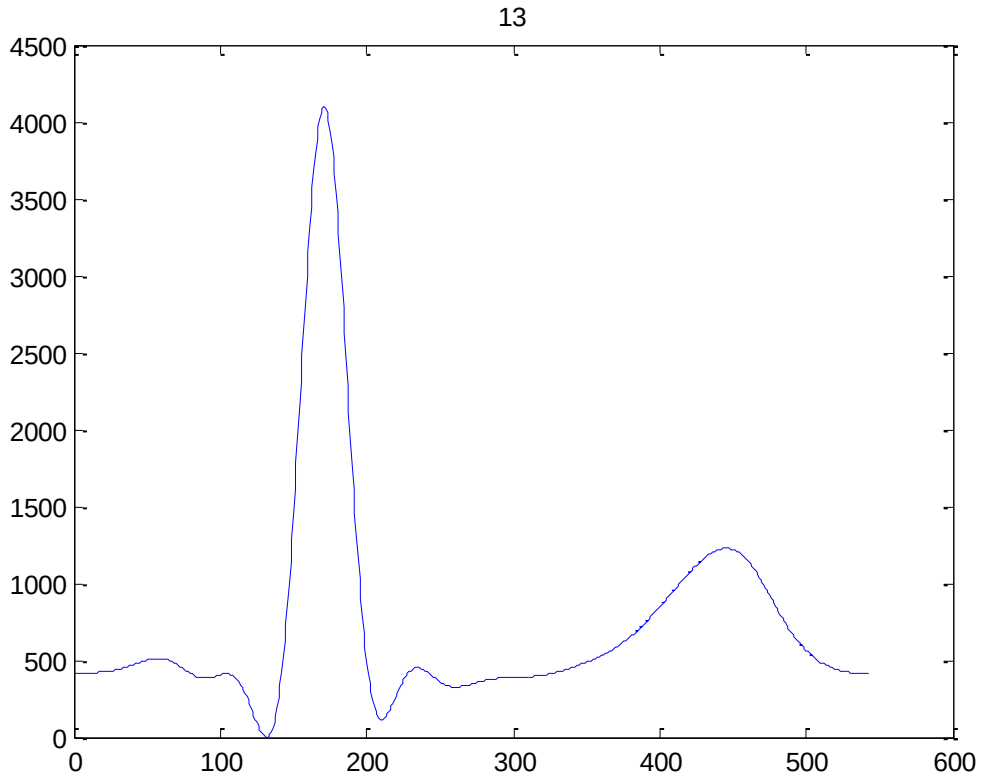
Señal de la derivación v4 de un ECG



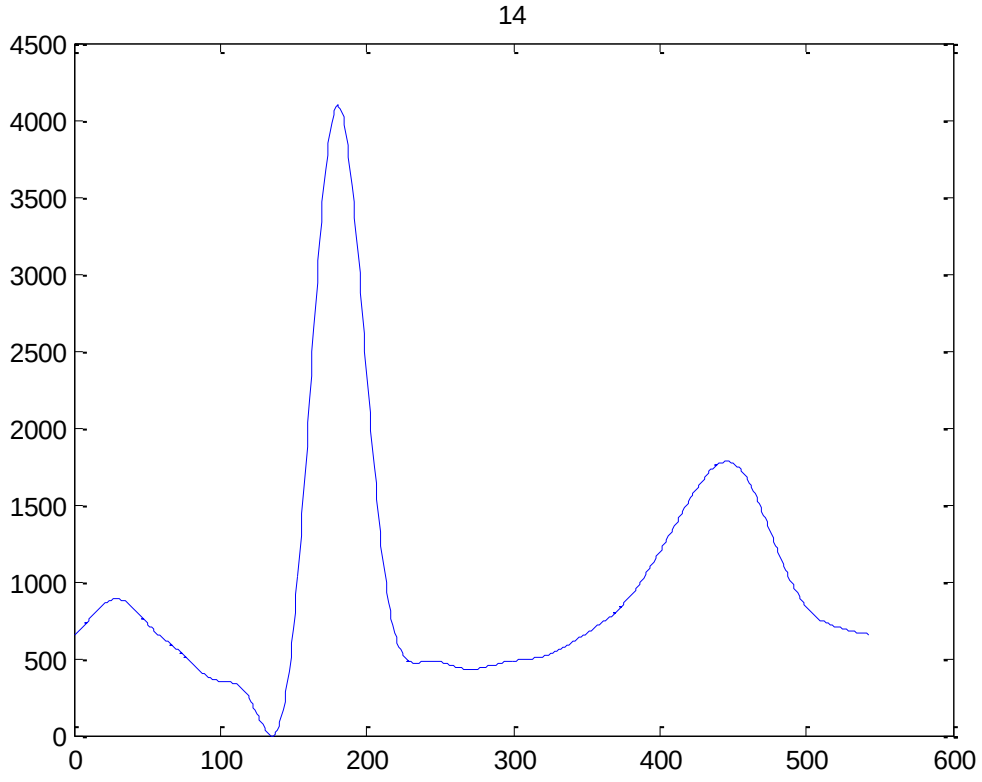
Señal de la derivación v5 de un ECG



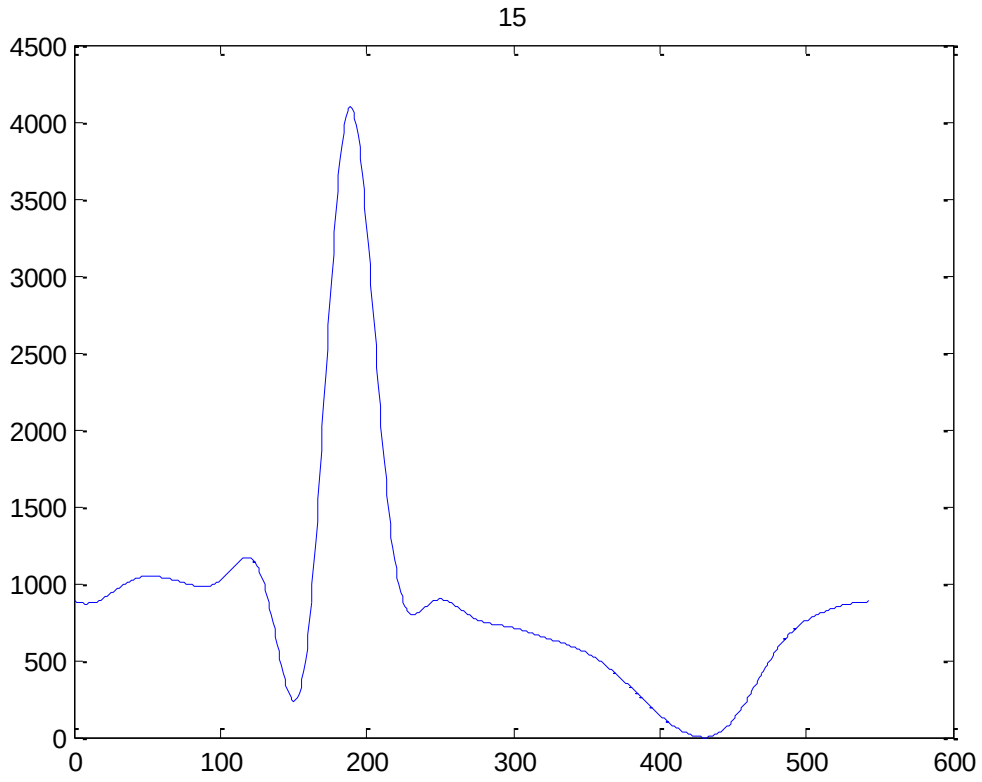
Señal de la derivación v6 de un ECG



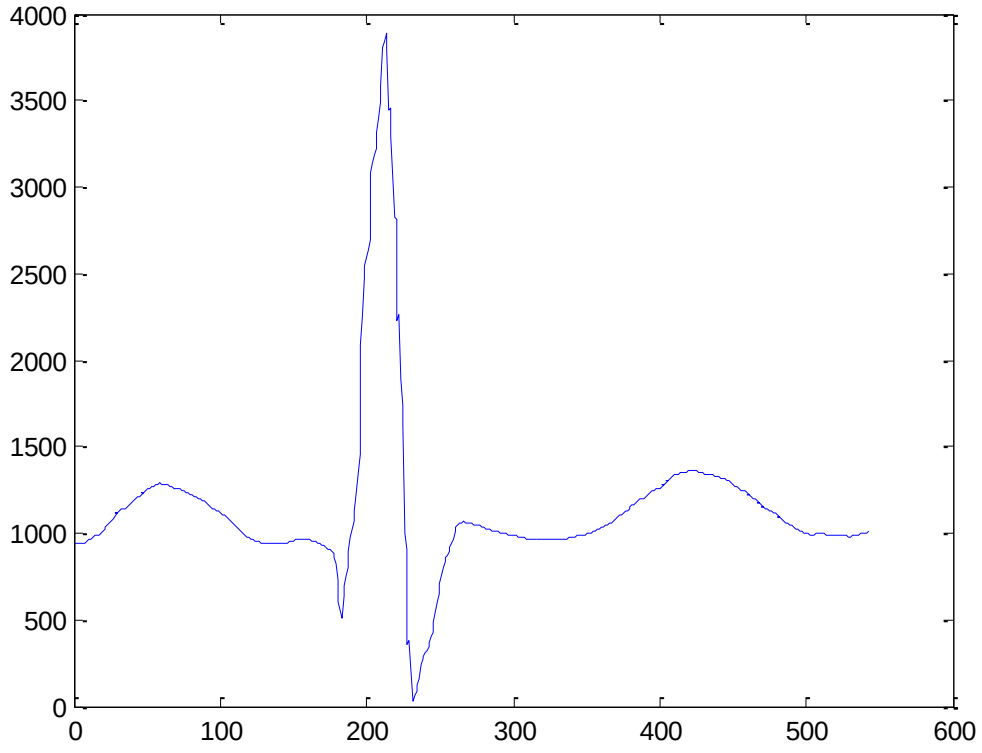
Señal de la derivación vx de un ECG



Señal de la derivación vy de un ECG



Señal de la derivación vz de un ECG



Señal de calibración

Anexo G

Índice de figuras