Desarrollo de dispositivos e-Health de bajo coste para Raspberry Pi

Low Cost e-Health devices development for Raspberry Pi

Xavier Gallofré Nieva
Jesús F. López San José
Álvaro Pérez Liaño

Proyecto de Sistemas Informáticos, Facultad de Informática
Universidad Complutense de Madrid

Departamento de Arquitectura de Computadores y Automática
Curso 2013/2014

Directores:
Luis Piñuel Moreno
Joaquín Recas Piorno

Abstract

Cardiovascular monitoring requires the use of expensive equipment, often unaffordable for certain low-income areas. The main complications in the process of reducing these costs stem from the difficulty of finding small, high-performance platforms with a low energy consumption able to acquire electrocardiographic signals in real time. To this purpose, this document presents a low-cost, low-consumption alternative that uses the Raspberry Pi, a rising platform nowadays. In order to allow the processing of the signal and its acquisition, morphological and transfer function-based filtering have been used, which have proven to be highly effective for ECG filtering. A chip, made by Texas Instruments, the ADS1198, has been used in the development of this alternative and has proven to be a low-cost, high-efficiency alternative.

The goal of this device is to provide a cheap, broad spectrum solution in order to sample and analyse signals in all environments, specially those with economic constraints.

Keywords ECG, ARM, Raspberry Pi, ADS1198, low cost

La monitorización cardiovascular requiere el uso de equipos de coste elevado, siendo este inasumible en muchos casos para ciertos ámbitos de bajos recursos económicos. Las principales complicaciones a la hora de reducir dichos costes se centran en encontrar plataformas de pequeño tamaño, alto rendimiento y bajo consumo energético que puedan capturar señales electrocardiográficas en tiempo real. Este documento presenta un dispositivo de bajo coste que cumple los objetivos antes citados, haciendo uso para ello de una plataforma ahora mismo en auge, la Raspberry Pi. Para permitir el procesamiento de la señal y capturar sus puntos clave se han usado tanto filtros morfológicos como filtros basados en funciones de transferencia, probándose altamente efectivos para ECGs. Se ha hecho uso además de un chip fabricado por Texas Instruments, el ADS1198, siendo este de coste reducido y de rendimiento elevado.

La meta final del dispositivo es facilitar el muestreo y análisis de señales en todos los entornos, sobre todo en aquellos con limitaciones económicas.

Palabras clave ECG, ARM, Raspberry Pi, ADS1198, bajo coste

Xavier Gallofré Nieva, Jesús F. López San José, Álvaro Pérez Liaño authorize Complutense University of Madrid to spread and use this report and the associated code, multimedia content and its results with academical and non-comercial purposes, provided that its authors shall be explicitly mentioned.

17 de junio de 2014

Xavier Gallofré Nieva Jesús F. López San José Álvaro Pérez Liaño

A Luis y Joaquín, por tener paciencia y enseñarnos

A mi padre, por saber guiarme y apoyarme.
A mi madre, por ponerme en el camino.
A todos mis amigos, sin vuestra compañía esto no habría sido lo mismo.

A mi abuela Felisa, por su confianza y mecenazgo, por regalarme un pasado.
A Encarnación y Jesús, mis padres, por su apoyo incondicional, por darme futuro.
A mis amigos y a Sonia, por vuestra valiosa compañía, por llenar el presente.

A mi familia, porque desde pequeño me habéis apoyado en lo que me gusta,
lo que me ha traído finalmente hasta aquí.
A mis amigos, tanto los que llevan toda la vida como los que han llegado después,
por hacer más interesante el camino.
A Sara, por todo el apoyo que me has dado tanto en las buenas como en las malas,
porque sin tus ánimos todo sería mucho más difícil,
porque consigues sacar lo mejor de mí.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En esta memoria se presenta el desarrollo de un prototipo de dispositivo holter diseñado sobre la Raspberry Pi. Se exponen los pasos que se han llevado a cabo, investigación que ha sido necesaria y los resultados obtenidos.

Este capítulo explica con detalle en que consiste este proyecto y cuál ha sido la motivación para llevarlo a cabo. Se exponen además las razones que han motivado las elecciones de las plataformas utiliziadas y la situación actual de los holter comerciales.

1.1. Motivación

Los monitores holter se suelen utilizar para discriminar el correcto funcionamiento del corazón en períodos de actividad normal, de ejercicio, estrés o reposo.
También permiten a un equipo médico diagnosticar problemas con el ritmo cardíaco (taquicardia, bradicardia, palpitaciones, etc), o determinar las implicaciones en un paciente del uso ne nuevos fármacos en su tratamiento.
La diferencia principal con los electrocardiogramas convencionales estriba en que estos solo recogen, de media, entre 1 y 3 minutos de muestra, mientras que los dispositivos holter son empleados para registrar y almacenar durante habitualmente 24 horas los latidos del paciente, para ser analizados posteriormente por el equipo médico.

La captura de esta señal electrocardiográfica ha sido terreno de múltiples avances tecnológicos a lo largo de los últimos 60 años, pese a que los principios de electrocardiografía dinámica, inicialmente ideados y desarrollados por Normal Holter y colaboradores (Gengerelli y Glasscock) no han variado apenas.
Estos avances tecnológicos han permitido con el tiempo mejorar la calidad del registro, aumentar la duración de la grabación y agilizar el análisis a posteriori de las muestras capturadas.
Además, también se ha facilitado con las mejoras tecnológicas el uso del paciente, puesto que al reducir el tamaño y consumo energético de los dispositivos se han visto beneficiados factores como la autonomía y la ergonomía.

Sin duda, el peor inconveniente de los monitores holter, tanto para los usuarios como para los hospitales es el alto coste de los dispositivos. Para ilustrar este concepto mostramos a continuación, en la tabla 1.1, características de algunos monitores holter comerciales.

Como se puede apreciar, el precio de estos dispositivos es bastante elevado. Esto es así porque se necesita cierta potencia de procesamiento para realizar la labor de captura de señales (y análisis en algunos casos), pero sin elevar drásticamente el consumo energético y con un tamaño mínimo de dispositivo.

Sin embargo, en los últimos años podemos ver cómo existe un mundo, cada vez más en auge, de tecnología de sistemas empotrados, cuyas premisas precisamente se centran en la relación capacidad de cómputo - bajo consumo energético. Esto se manifiesta en nuestro entorno con las tecnologías integradas en teléfonos móviles, navegadores, tablets, etc.
Además, existen iniciativas, cada vez más numerosas y de mayor calado, que abogan por dispositivos reducidos, con estas mismas características de bajo consumo y respetable

Algunos monitores holter comerciales

Nombre	**$ aprox.**	**Características**
Holtech Hcaa 348	$ 8.000	3 canales. ECGs 24-48 h. Resolución: 225 muestras por segundo por canal, 8 bits por muestra. Consumo: 2 pilas AA cada 48h. Incluye software para PC de análisis asistido.
Cardiovex Veccsa	$ 10.000	3 canales. ECGs 24-48h. Resolución: 250 muestras por segundo por canal, 8 bits por muestra. Consumo: 1 pila AA cada 48h. Incluye software muy completo para PC de análisis asistido y base de datos de pacientes.
Healforce Prince 180B	$ 2.000	3 canales. ECGs 24h (30 segundos por test). Resolución: 250 muestras por segundo por canal, 8 bits por muestra. Consumo: 2 pilas AAA cada 3000 ECGs de 30 segundos. Incluye software de control del dispositivo para PC.

Cuadro 1.1: Características principales de algunos monitores holter comerciales

capacidad de cómputo, y que tienen a sus espaldas grandes y enriquecedoras comunidades de desarrollo open source, que pretenden llevar la tecnología y el conocimiento a todos los rincones del mundo.

Es precisamente Raspberry Pi, una de estas iniciativas de gran fama internacional, la que encarna a la vez nuestro interés por el desarrollo en un entorno de bajo coste, para todos, y la capacidad para realizar un producto realmente beneficioso, con unas características similares o, en algunos aspectos, superiores a los monitores holter de tirada comercial.

Con todo esto en mente, en este proyecto pretendemos ofrecer, en lo que abarca la monitorización holter, quizá no una solución final, pero sí un buen punto de partida para

poder acercar estas tecnologías de monitorización cardíaca a entornos económicamente poco favorecidos, con un interés principal de facilitar el acceso a la salud al máximo número de personas posible.

1.2. Visión general del documento

Esta memoria pretende ofrecer una visión de conjunto de este proyecto, así como dejar claras las implicaciones del mismo y los temas cubiertos por las investigaciones sin las cuales todo esto no se podría haber llevado a cabo. Vamos a comentar de forma general la estructura del documento que tenemos entre las manos.

Como se ha visto, inicialmente se encuentra un capítulo de *Introducción*, en el que podemos perfilar las características y motivaciones principales de este proyecto, así como obtener una idea inicial en la que iremos profundizando en la secciones subsiguientes.

El capítulo *Antecedentes* nos ofrece una base de estudio sobre la que fundamentar el proyecto y las decisiones que más adelante tendremos que tomar. Este capítulo de estudio y análisis se centra en dos pilares fundamentales para nuestro proyecto: la base científicomédica de la captura y análisis electrocardiográfico y la tecnología hardware que hace posible esta empresa.

Con esa base clara, podemos ver cómo en el capítulo siguiente, *Decisiones de Diseño* se presentan diferentes alternativas para resolver problemas o conseguir metas y los procesos de decisión que nos llevan a tomar una u otra vía en cada caso. Estas decisiones tienen repercusión en el desarrollo del proyecto, por lo que se les ha querido dar aquí un espacio para su reflexión.

El siguiente capítulo se centra en la fase de *Desarrollo* del proyecto. Está a su vez dividido en dos secciones; la primera, con gran contenido técnico, se centra en el diseño y la arquitectura del producto, y la segunda ofrece una visión esquemática y, por claridad, con cierta informalidad del modelo de proceso que ha permitido llevar un control y buen rumbo en el desarrollo del proyecto.

En el último capítulo se ofrece una visión clara de los *Resultados* obtenidos en este proyecto, así como de las medidas de calidad que se han podido establecer con el producto finalizado. Por otro lado, se presenta también un compendio de algunas de las posibles expansiones de nuestro proyecto, así como algunos usos futuros que hacen de ésta una

aplicación con vida por delante.

Para concluir, disponemos de un conjunto de *Apéndices*, en el que se suministran, entre otros, un manual de usuario, esquemáticos y coste para la miniaturización del dispositivo, explicaciones detalladas sobre temas como estructuras auxiliares empleadas, etc. Al final de este capítulo de apéndices podemos encontrar un glosario de términos, que puede ayudar a resolver algunas dudas que puedan surgir sobre términos concretos a lo largo de la lectura de este documento.

Capítulo 2

Antecedentes

Para el correcto desarrollo de este proyecto se ha llevado a cabo una investigación que cubre desde aspectos médicos como la composición y funcionamiento de un corazón, hasta aspectos más técnicos como protocolos de comunicación y las plataformas utilizadas.

En este capítulo se citan los antecedentes investigados, exponiendo un breve resumen de cada uno de ellos. De este modo se pretende dar una visión global de los aspectos básicos que conforman el proyecto, facilitando así la comprensión del resto de la memoria.

2.1. Electrocardiograma (ECG/EKG)

El electrocardiograma o señal electrocardiográfica es un registro a lo largo del tiempo de la actividad eléctrica global del corazón, no sólo de la zona de conducción eléctrica.

Dicho registro se lleva a cabo gracias a diferencias de potencial existentes debido a la actividad contráctil del corazón. Esta señal es de gran importancia ya que, gracias a un estudio de su forma y variaciones temporales, sirve como método para diagnosticar diversas enfermedades cardiovasculares, siendo un ejemplo de ellas las arritmias cardíacas.

2.1.1. Estructura del ECG

El trazado normal de un ECG [Fig 2.1] mientras se registra un latido del corazón, consta de una serie característica de ondas y complejos:

- Onda P
- Complejo QRS
- Onda T

Las señales más comunes suelen estar dentro del rango de amplitudes que va desde los $0{,}5mV$ hasta los $5mV$, contando, además, con una componente continua de hasta $\pm 300mV$, efecto del contacto de los electrodos con la piel, y otra de $1{,}5V$, causa de la diferencia de potencial existente entre los electrodos y la masa.

2.1.2. Electrofisiología cardiaca

El corazón está constituido por cuatro cámaras diferenciadas: dos aurículas (izquierda y derecha) y dos ventrículos (izquierdo y derecho) [Fig 2.2].

Su funcionamiento es siempre el mismo: la aurícula derecha recibe la sangre venosa del cuerpo a través de la vena cava superior y la envía al ventrículo derecho. Para que dicha sangre pueda oxigenarse, el ventrículo derecho la envía a través de la arteria pulmonar a los pulmones, retornando al corazón, a través de la vena pulmonar, a la aurícula izquierda. Para finalizar con el ciclo, la sangre pasa de dicha aurícula al ventrículo izquierdo, el cual la distribuye por todo el cuerpo gracias a la arteria aorta, para volver posteriormente a la aurícula derecha y así cerrar el ciclo.

Para que todo este funcionamiento periódico se realice de manera síncrona y sin errores, el corazón dispone de un sistema de conducción eléctrica constituido por fibras de músculo

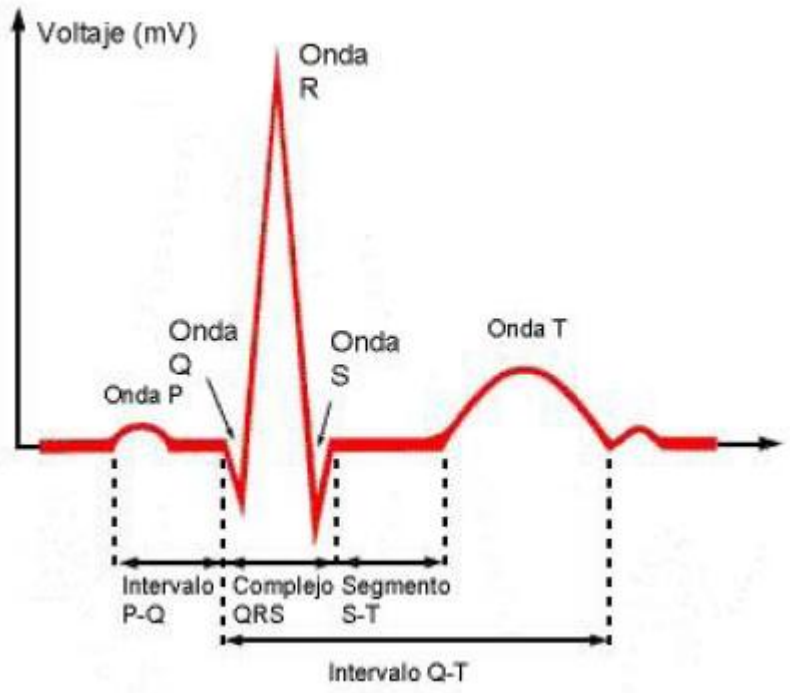


Figura 2.1: Estructura de la señal de ECG durante un ciclo cardiaco

cardiaco cuya especialidad es la transmisión de impulsos eléctricos. Dicho sistema es autoexcitable, razón que explica que no tengamos ningún control sobre los latidos del corazón. Se compone de los siguientes elementos reflejados en la figura 2.3:

- Nodo sinusal o sinoatrial (NSA).
- Tractos internodales.
- Nodo auriculoventricular (NVA).
- Haz de His, con sus ramas derecha e izquierda para cada ventrículo.
- Fibras de Purkinje.

A continuación se describe el proceso más detalladamente y relacionando cada etapa del ciclo cardiaco con sus eventos eléctricos asociados:

1. **Contracción auricular**, provocada por una despolarización de las mismas debido a la transmisión del impulso eléctrico generado en el NSA a través de los tractos internodales. Con esta fase, da comienzo la onda P en el registro electrocardiográfico,

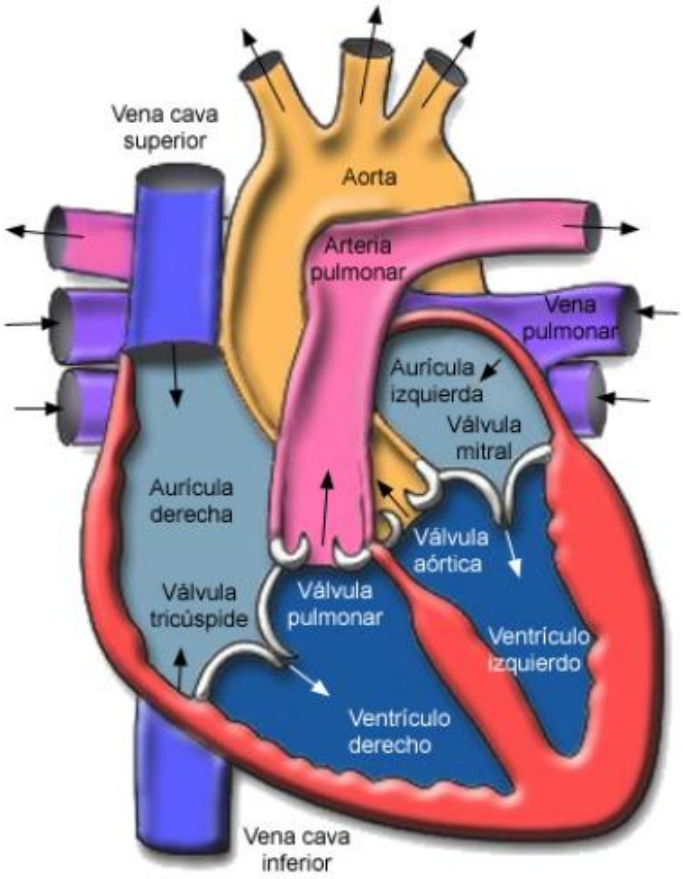


Figura 2.2: Estructura física del corazón con sus partes más importantes

y se produce el paso de la sangre venosa por las aurículas, produciendo un llenado rápido de los ventrículos.

2. **Contracción isovolumétrica**, debido al cierre de las válvulas existentes entre las aurículas y los ventrículos, con lo que sube la presión intraventricular. El origen de esta contracción está en una despolarización de los ventrículos, ya que se ha seguido transmitiendo el impulso eléctrico por el NAV y el Haz de His. Con esta fase, da comienzo el complejo QRS del ECG.

3. **Eyección rápida de sangre**, ya que la presión intraventricular es superior a la existente en las arterias, por lo que se abren las válvulas ventriculares expulsando la sangre rápidamente. En esta fase, como sigue produciéndose la despolarización ventricular, todavía se producirá el complejo QRS.

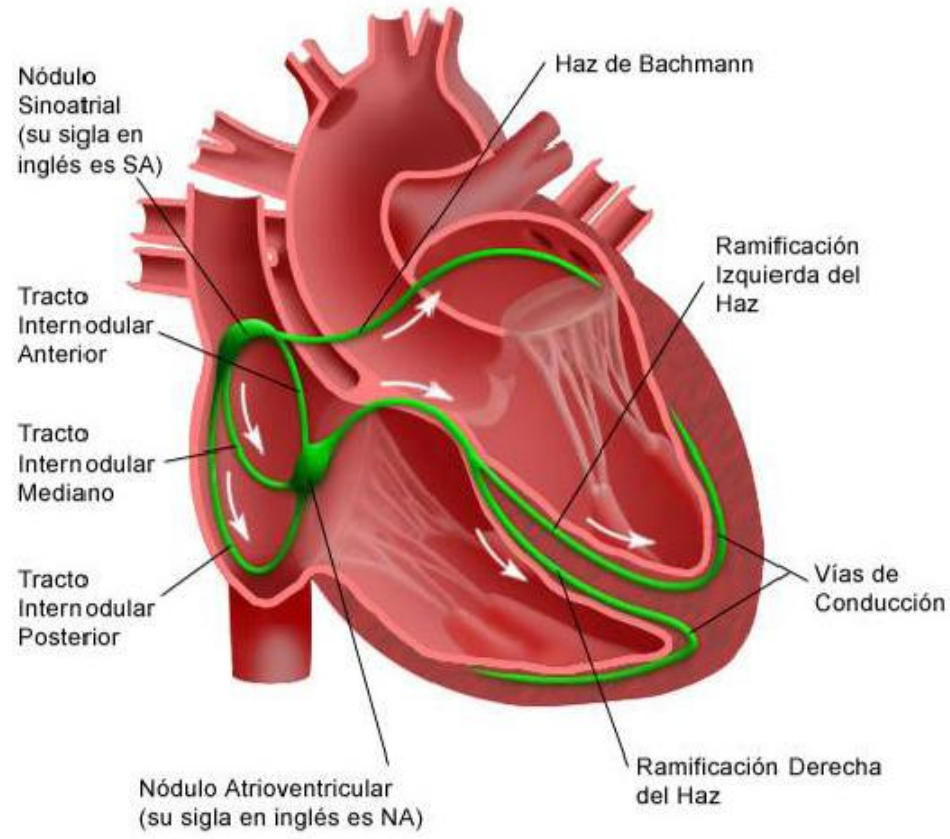


Figura 2.3: Estructura eléctrica del corazón con sus partes más importantes

4. **Eyección reducida de sangre**, debido a que baja la presión intraventricular. Además, se produce una progresiva repolarización ventricular, dando lugar a la aparición de la onda T.

5. **Relajación isovolumétrica**, en la que las válvulas que estaban abiertas entre los ventrículos y las arterias se cierran.

6. **Llenado rápido**, en el cual la presión ventricular cae por debajo de la auricular, por lo que se abren las válvulas auriculoventriculares, provocando el llenado rápido de los ventrículos ya comentado.

7. **Llenado lento**, donde los ventrículos siguen llenándose de forma más pausada que en la fase anterior, a la vez que se expanden. Una consecuencia de esta fase es que baja la presión arterial.

2.1.3. Procedimiento del registro del ECG

El proceso físico en sí que se lleva a cabo es medir la actividad eléctrica del corazón entre varios puntos corporales, ya que mientras el corazón pasa por los estados de polarización y despolarización, el cuerpo en su conjunto se comporta como un volumen conductor, propagando la corriente eléctrica.

El equipo de registro consta de una serie de electrodos (en concreto, 10) que se conectan a la piel del paciente y de un equipo de registro. Los electrodos se colocan en unas posiciones predeterminadas y universales, lo que nos permite obtener el registro del llamado sistema de las 12 derivaciones de Einthoven, que son de tres tipos:

- *Frontales* (I, II y III). Son bipolares. Se miden entre dos electrodos, uno positivo y otro negativo.

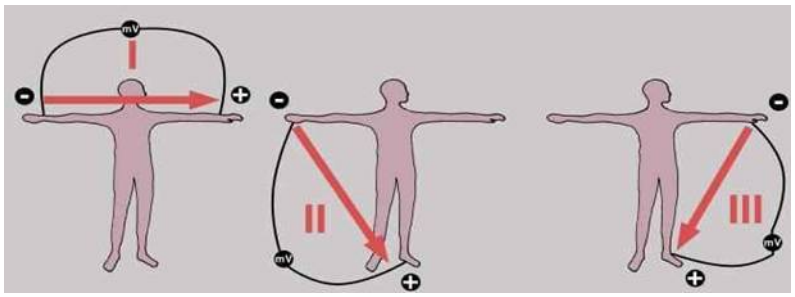


Figura 2.4: Medición de las derivaciones frontales

- *Frontales aumentadas* (aVR, aVL y aVF). Son unipolares. Se miden entre un electrodo positivo y 0.

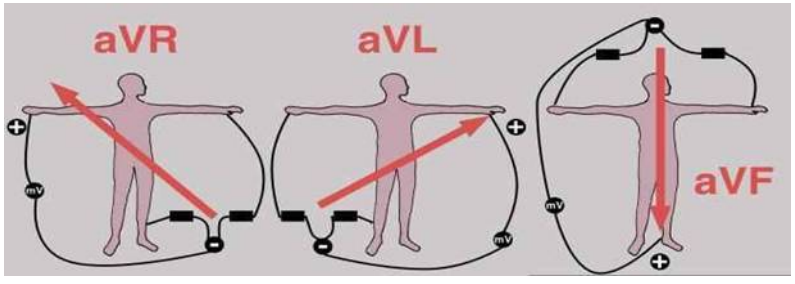


Figura 2.5: Medición de las derivaciones aumentadas

- *Precordiales* (V1-V6). Son unipolares.

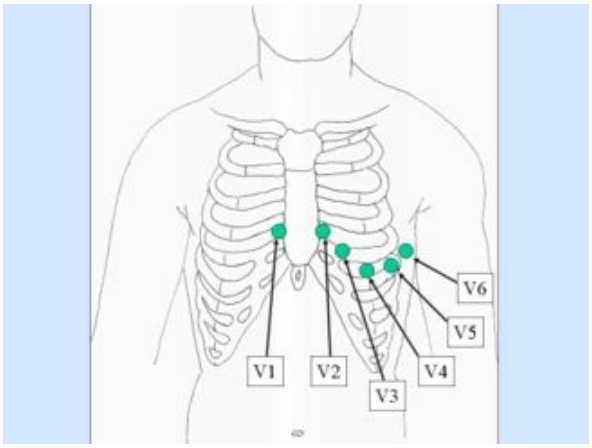


Figura 2.6: Medición de las derivaciones precordiales

Con cada derivación lo que hacemos es obtener una visión parcial, es decir, información específica de una parte del corazón, con lo que si disponemos de todas tendremos una información completa.

2.1.4. Usos del ECG

A continuación se presentan los usos más comunes:

- La más importante, determinar si el corazón funciona correctamente o sufre alguna anomalía.
- Indicar bloqueos coronarios arteriales.
- Detección de alteraciones electrolíticas de potasio, sodio, calcio o magnesio.
- Detección de anormalidades conductivas.
- Mostrar la condición física de un paciente durante un test de esfuerzo.
- Ofrecer información sobre las condiciones físicas del corazón.

2.1.5. Monitor Holter

Un monitor Holter es un sistema portable de guardado y monitorización continua de electrocardiogramas, que registra la señal durante 24 y 48 horas. Se utiliza para detectar y evaluar arritmias intermitentes así como hipertensión.

Su portabilidad permite a los pacientes hacer vida normal aunque se les pide que registren que tipo de actividades realizan para poder compararlo con los registros obtenidos.

Sus capacidades de registro no superan la semana ya que se limitan a registrar datos que tienen que se procesados con posterioridad, lo que lo convierte en una herramienta dificil de gestionar debido a la cantidad de información que genera. Por este motivo sólo se utiliza para detectar determinados tipos de cardiopatías.

2.2. Tecnologías

Para llevar a cabo la captura de señales electrocardiográficas y su posterior procesamiento, se han tenido que investigar diversas tecnologías. Se ha necesitado disponer de un módulo de captura, otro de procesamiento y una interfaz de conexión entre ambos.

Cómo módulo de procesamiento se hace uso de la placa Raspberry Pi, aprovechando la gran aceptación que está teniendo entre el público y su bajo coste. En cuanto a la captura se ha optado por utilizar el chip ADS1198, que permite unas velocidades de captura elevadas y al igual que la Raspberry, tiene un coste reducido. Por último, para conectar ambos dispositivos, se ha recurrido a una interfaz SPI, sienda esta la única opción de conexión compartida por ambos dispositivos.

2.2.1. Bus SPI

El bus SPI (*Serial Periferical Interface*) [8] es un estándar de comunicación serie síncrono desarrollado por la empresa Motorola. La transmisión y recepción de datos se realiza en buses separados, permitiéndose la comunicación en ambos sentidos de manera simultánea, por tanto es un protocolo englobado en la categoría full duplex. Su uso está extendido en comunicaciones de corta distancia, como pueden ser las involucradas en circuitos integrados, tarjetas SD o sensores.

La transferencia consta siempre de un dispositivo principal, conocido como maestro y al menos otro, denominado esclavo. El maestro es el encargado de enviar las señales de reloj y de seleccionar el esclavo activo en cada instante de la comunicación [Fig 2.7].

Las líneas involucradas en la comunicación son las siguientes:

- **MISO** (Master Input Slave Output): Bus de salida de datos de los esclavos y entrada al maestro.

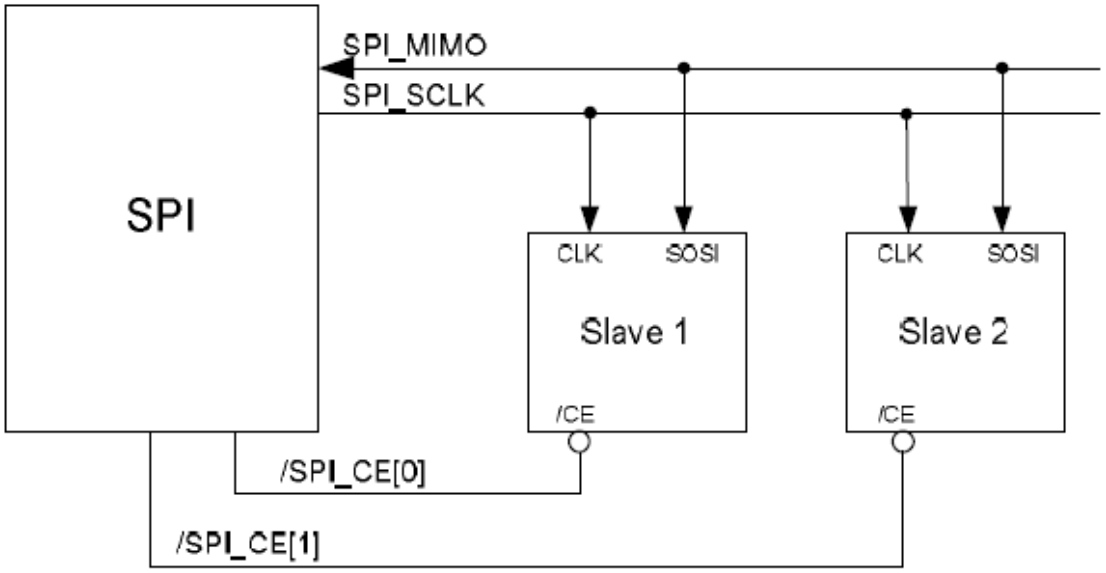


Figura 2.7: Comunicación SPI

- **MOSI** (Master Output Slave Input): Bus de salida de datos del maestro y entrada a los esclavos.
- **SCLK** (Clock): Pulso de reloj generado por el maestro.
- $\overline{\textbf{CS}}$ (Chip Select): Bus de salida del maestro y entrada a los esclavos, se encarga de seleccionar el esclavo activo en la comunicación[1].

Para procesar una lectura desde un esclavo, el maestro ha de realizar una escritura en el bus, provocando así la generación de la señal de reloj que desencadenará la escritura de datos por parte del dispositivo deseado.

Las transferencias se realizan en tamaño de byte, por lo que si se quiere realizar, por ejemplo, una escritura de un byte que provoque una lectura de otro byte, sería necesario generar 16 pulsos de reloj y mantener la línea $\overline{\text{CS}}$ a 0 durante ese tiempo [Fig 2.8].

El dispositivo maestro ha de conocer las siguientes características de cada esclavo partícipe en la comunicación:

- **Frecuencia máxima de transferencia**: La velocidad de la comunicación con cada dispositivo vendrá limitada por este valor, siendo imposible enviar o recibir datos a más velocidad.

[1] Normalmente los dispositivos trabajan con la línea CS negada

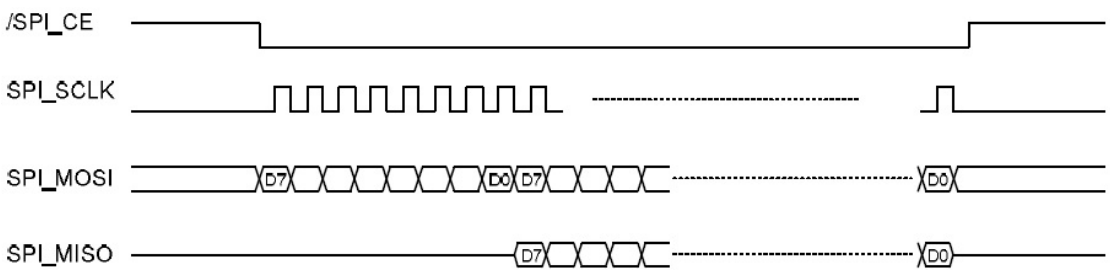


Figura 2.8: Cronograma SPI

- **Polaridad** (CPOL): Determina la polaridad del reloj [Fig 2.9].

- **Fase** (CPHA): Determina el flanco de reloj donde se desencadenará la escritura [Fig 2.9].

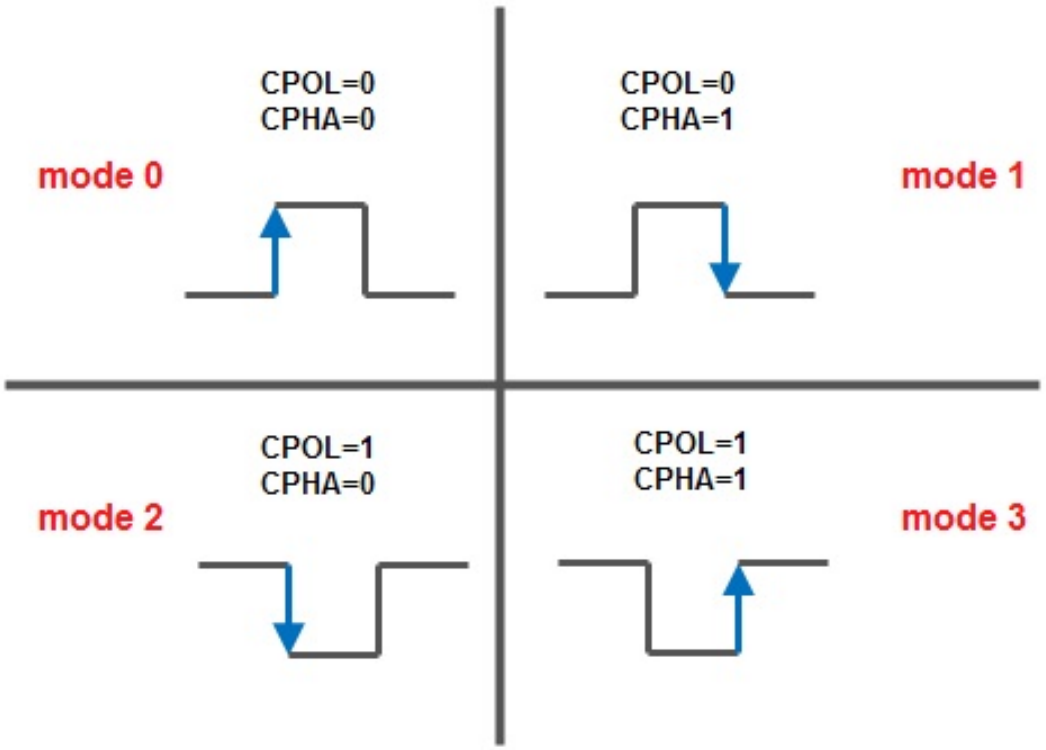


Figura 2.9: Configuraciones posibles de polaridad y fase en SPI

2.2.2. Raspberry Pi

La Raspberry Pi[2] es una placa de bajo coste desarrollada por la Universidad de Cambridge para promover la enseñanza de ciencias de la computación en entornos académicos. Actualmente se pueden encontrar dos modelos en el mercado, el modelo A y el modelo B, diferenciándose en características técnicas que influyen en su precio de venta.

Ambos modelos están constituidos por el System-on-Chip de Broadcom **BCM2835** [2], que utiliza como procesador un **ARM1176JZF-S** [1] a 700MHz (Arquitectura versión 6 de ARM) y un procesador gráfico VideoCore IV. Asimismo constan de salidas de vídeo RCA y HDMI, un puerto USB, salida de Audio y pines de entrada/salida de propósito general. Como dispositivo de almacenamiento de datos no volátil es necesario utilizar una tarjeta SD.

El modelo A consta de una memoria RAM de 256 MB, mientras que la memoria del modelo B asciende a 512 MB, doblando de esta manera la memoria disponible en la versión A. En ambos casos la memoria RAM se encuentra compartida con la GPU.

La versión B cuenta además con un puerto USB adicional y conectividad de Red mediante una interfaz 10/100 Ethernet (RJ-45).

El consumo energético es de 500mA (2.5W) en el modelo A y 700mA (3.5W) en el B.

La fundación Raspberrypi.org está siendo la encargada de dar soporte a la placa, generando distribuciones de Linux, apoyando el desarrollo y fomentando el crecimiento de una comunidad en torno al dispositivo.

2.2.3. Chip ADS1198

El chip ADS1198 [3] en un Front-End de Texas Instruments de bajo consumo para mediciones de señales ECG, se caracteriza por tener ocho canales de 16 bits cada uno[3], una frecuencia de muestreo de hasta 8 KHz, un amplificador de ganancia programable, referencia interna y un oscilador integrado. Cada canal consta de un multiplexor que permite la lectura desde ocho entradas diferentes, siendo las más relevantes las entradas de temperatura, electrodos y señal de test generada internamente[4].

[2]La página web www.raspberrypi.org contiene toda la información sobre la placa, así como enlaces a proyectos y foros

[3]Una versión más moderna del chip, el ADS1298, dispone de una resolución de 24 bits por canal

[4]La señal de test es idónea para realizar comprobaciones iniciales de funcionamiento

El chip consta de una interfaz SPI para permitir la comunicación con otros dispositivos. Además de las líneas típicas de SPI, se proporciona una línea adicional, **$\overline{\text{DRDY}}$**, que índica cuando se tienen nuevos datos válidos preparados para enviar. La lectura de datos [Fig 2.10] se realiza siempre para los 8 canales, devolviendo adicionalmente una cabecera con información de la configuración del chip.

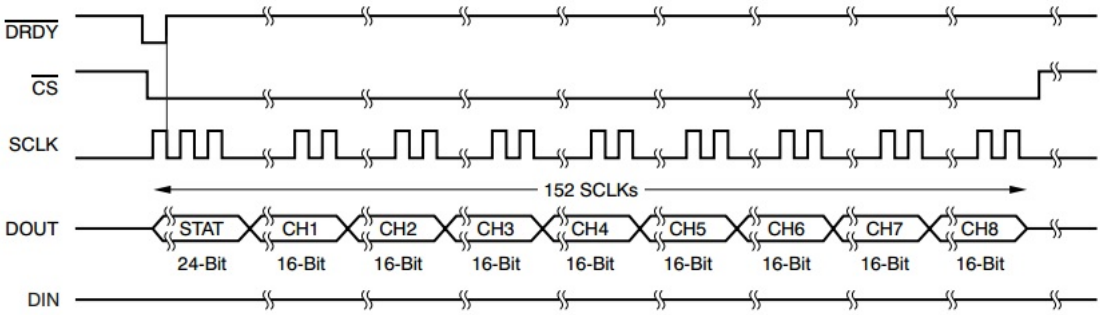


Figura 2.10: Salida del bus SPI durante lectura del chip ADS1198

El chip permite obtener la alimentación de manera unipolar o bipolar:

- **Unipolar**: La alimentación unipolar se realiza mediante una entrada de 5V y otra de 3V, coincidiendo estas con las proporcionadas por la Raspberry Pi mediante pines de propósito general.
- **Bipolar**: El modo bipolar requiere entradas de ±2.5V. Este modo no ha sido utilizado por disponer de manera más sencilla de las entradas necesarias para el unipolar.

Además de las entradas de SPI, el chip cuenta con 3 entradas de especial importancia para su funcionamiento:

- **START**: Mediante esta entrada permitimos el inicio de las conversiones. Adicionalmente podemos dejar la entrada a 0 y realizar el mismo comportamiento mediante SPI, haciendo uso de un comando específico.
- **$\overline{\text{RESET}}$**: Esta entrada fuerza el estado de reset del chip. Podemos dejar la entrada a 1 y hacer uso de un comando específico mediante SPI que realiza la misma función.
- **$\overline{\text{PWDN}}$**: Esta última entrada enciende o apaga el chip, por lo que si está a 0 el chip se encontrará desconectado y a 1 estará conectado y funcionando.

Internamente el ADS1198 cuenta con 25 registros que permiten al usuario configurar todas las características programables del chip. Gran parte de la configuración permitida está relacionada con aspectos propios de la captura de señales, como las ganancias, el uso de un oscilador interno o el voltaje de referencia utilizado para la captura. El resto de configuraciones posibles permiten variar la frecuencia de captura, las entradas de los canales o modificar las señales de test internas en amplitud y frecuencia.

Dado su elevado rendimiento, alto nivel de integración y bajo consumo, el ADS1198 permite el desarrollo de instrumentación médica de prestaciones elevadas, tamaño reducido y bajo coste.

Para el propósito del proyecto hemos usado un kit de desarrollo, donde se incluye el chip integrado en una placa, permitiendo configurar la forma de alimentación o tener acceso a las entradas de datos de forma más cómoda y sencilla.

Características técnicas:

- 8 canales de alta resolución.
- Bajo consumo: 0.55mW/canal.
- Frecuencia de muestreo: desde 125Hz a 8kHz.
- Ganancia programable: 1, 2, 3, 4, 6, 8, o 12.
- Alimentación: Unipolar o Bipolar.
  - Analógica: 2.7V a 5.25V.
  - Digital: 1.65V a 3.6V.
- Oscilador y referencia internos.
- Señales de test integradas.
- Comunicación mediante interfaz SPI.
- Rango de temperatura operativo: 0 °C a 70 °C.

Aplicaciones:

- Monitorización de pacientes.
- Adquisición de señales de alta precisión.

Capítulo 3

Decisiones de Diseño

Este capítulo abarca todas las decisiones que se han tomado a la hora de realizar el diseño del dispositivo. Los siguientes apartados justifican decisiones que se han tomado a lo largo del desarrollo del proyecto, abarcando aspectos tanto técnicos como funcionales.

Se ha tomado como premisa investigar todas las opciones posibles para los filtros aplicados, sistema operativo utilizado, flujo de ejecución de la aplicación o el manejo de gráficos.

Por último, y para dar paso al siguiente capítulo, se hace una conclusión general presentando el producto que vamos a desarrollar.

3.1. Elección de filtros

Para facilitar la delineación de la señal es necesario aplicar un filtro paso banda que se consigue combinando un filtro paso baja con una frecuencia de corte de 40 Hz más un paso alta, necesarios para quitar el ruido de alta frecuencia de origen muscular y la componente continua respectivamente.

Se ha optado por buscar alternativas a los filtros basados en función de transferencia, ya que requieren una elevada complejidad de cálculo en comparación con los filtros morfológicos.

En el artículo *Ecg signal conditioning by morphological filtering* [7] se ha demostrado que el filtro morfológico que proponen como alternativa al paso alto ofrece unos resultados muy buenos para señales de electrocardiogramas, siendo esta la razón que justifica el uso de este filtro en nuestro proyecto.

Sin embargo, la propuesta para el filtro paso bajo no funciona tan bien, por lo que se ha implementado un filtro paso bajo basado en una función de transferencia.

3.2. Raspbian

El sistema operativo que utilizamos es **Raspbian**[1], siendo este una distribución de Linux que ofrece la propia RaspberryPi Foundation. Esta distribución es de instalación fácil, es gratuita y consta de una comunidad activa de usuarios, encajando de este modo perfectamente con los ideales del proyecto.

Raspbian está basado en Debian Wheezy (Debian 7.0 [6]) y ha sido optimizado para la Raspberry Pi. Consta de más de 35.000 paquetes, software pre-compilado y su desarrollo sigue en activo. Destacan su alta estabilidad y elevado rendimiento, habiéndose optimizado el cálculo en coma flotante mediante hardware.

3.3. Xenomai y Linux CNC

Antes de optar definitivamente por el uso de Raspbian, estimamos necesario ver si había alguna alternativa más potente, preferiblemente una que facilitara el desarrollo en tiem-

[1] En la página web www.raspbian.org se puede descargar la distribución, así como conocer detalles de la misma

po real. Lo que más se adecuaba a nuestras necesidades era el Framework **Xenomai**[2], framework pensado para cooperar con el Kernel de Linux en proyectos con requerimientos temporales delicados.

La ejecución de este framework se permite en Kernels de Raspberry Pi, pero por desgracia su instalación no es sencilla, haciéndose necesario incluirlo durante la compilación del propio Kernel. Decididos a considerar todas las opciones, pensamos que era necesario comparar el rendimiento de Raspbian frente al de un sitema operativo que hiciera uso de este framework.

La compilación la realizamos utilizando una distribución de Linux pensada para el manejo de micro controladores, **Linux CNC**[3]. Esta distribución cuenta con un desarrollo mucho menor que Raspbian, pero al estar pensada para ámbitos donde la temporización es crucial, concluimos que sería la mejor opción de cara a un test de rendimiento.

La prueba se consistió en un código que ejecutaba rutinas mediante temporizadores, simulando lecturas de SPI a 8KHz y tratando posteriormente los resultados obtenidos. Lo interesante de esta ejecución era comprobar cuantas muestras se perdían en cada distribución por señales que no llegaban a poder procesarse a tiempo.

Aunque los resultados completos de las ejecuciones pueden consultarse en el apéndice F, a continuación se exponen las medias obtenidas:

- Con la versión de Linux CNC que hace uso de Xenomai se perdía una media de un 0,14 % de los temporizadores mandados por el sistema.

- La versión de Raspbian sin embargo perdía un 0,66 % de las muestras, valor ligeramente superior al anterior.

Considerando el esfuerzo de la compilación de una distribución que carece de gran parte de la funcionalidad de una que está en desarrollo activo y viendo además que en ambos casos los valores de muestras péridas no llegan siquiera al 1 %, optamos por seguir la línea de desarrollo inicial haciendo uso de Raspbian.

[2]La información completa del framework puede consultarse en su página web www.xenomai.org
[3]La página web de Linux CNC www.linuxcnc.org contiene toda la información de la distribución

3.4. Temporizadores e interrupciones

Para realizar la captura de datos mediante SPI teníamos disponibles dos opciones, usar interrupciones Kernel o realizar capturas periódicas mediante el uso de temporizadores y señales.

Las interrupciones Kernel obligan a crear un módulo que las maneje, haciéndose necesario poder compilar e insertar el módulo en el Kernel actual o compilar uno propio, introduciendo el módulo en el proceso de compilación[4]. Dada la imposibilidad de introducir un módulo al Kernel por limitaciones de las distribuciones Linux disponibles, sólo queda la opción de compilar uno que lo contenga desde el principio.

Una señal en Linux no es más que un mecanismo para informar a un proceso de diversos eventos, provocados por el mismo o por otros procesos [4]. Es similar a una interrupción lógica, pues al llegar la señal, el proceso interrumpe su ejecución normal y se procede al tratamiento de la señal, haciendo uso para este fin de una función de desviación. Un temporizador es un método para procesar la captura de señales de forma periódica, permitiendo de esta manera la ejecución de rutinas de tratamiento a intervalos definidos por el usuario.

El afán del proyecto es formar parte del avance actual de la Raspberry, evitando quedarse estancado en versiones antiguas de las distribuciones disponibles, por lo que la opción de distribuir la aplicación con un Kernel propio se ha descartado.

El uso de temporizadores y señales se puede realizar en espacio de usuario, no siendo necesario hacer uso del espacio del Kernel. Para que los temporizadores no consuman gran cantidad de tiempo de procesamiento, la rutina que ejecuten no debe requerir un cómputo excesivo, por esto, la aplicación únicamente plantea su uso para la captura de señales y renderizar la pantalla, en caso de estar esta última conectada.

3.5. Threads

Para el desarrollo de este producto se estudiaron y ensayaron dos modelos de diseño utilizando hilos de ejecución, aparte del diseño puramente secuencial en un solo hilo y

[4]Raspbian no permite la compilación de módulos Kernel, debido a que las cabeceras necesarias no se encuentran disponibles en la distribución. Ciertas versiones, ya antiguas, si constan de estas cabeceras, pero hemos optado por no considerarlas al no ser tan estables como las versiones actuales y tener ciertos elementos, como el módulo SPI, no integrados de base

proceso.

En primer lugar fue estudiada la posibilidad de separar el procesado secuencial de la onda en módulos que realizaran el trabajo de forma dividida, tomando y recibiendo datos de unas bandejas intermedias cuya ocupación podría variar flexiblemente dentro de un rango, para absorber las posibles fluctuaciones de tiempo invertido en el procesado de cada estación [Fig 3.1].

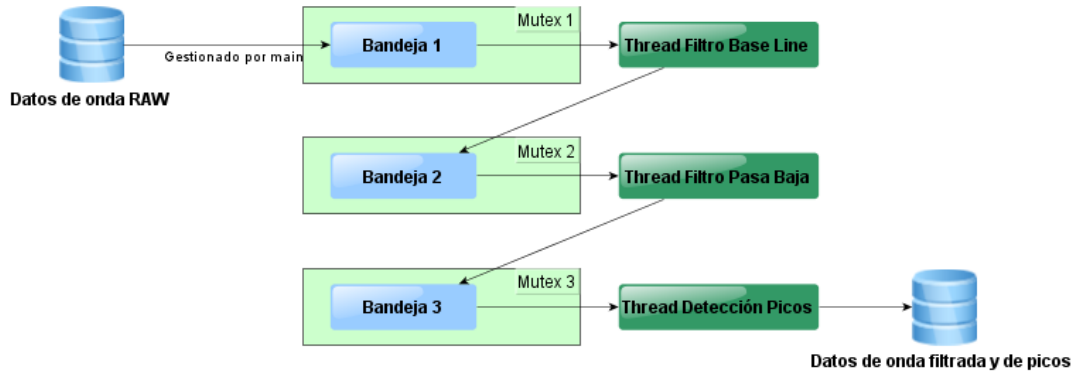


Figura 3.1: Posible diseño modular con hilos

Sabíamos desde el principio que ni el tipo de procesado era muy paralelizable ni disponíamos de más de un núcleo en nuestro dispositivo, pero fue nuestro interés investigar la viabilidad de una modularización basada en threads.

A pesar de ofrecer un pipeline correctamente ensamblado, los resultados analizados no fueron beneficiosos en absoluto. El tiempo desperdiciado en cada operación de bloqueo con los *mutex*, del orden de milisegundos, hacía imposible el procesado de una señal con un período de captura del orden de microsegundos. De hecho, el experimento indicaba claramente que aunque dispusiéramos de un procesador con varios núcleos y la aplicación corriese en un sistema operativo a tiempo real (acelerando así el tiempo de mutex), las bajas propiedades de paralelismo inherentes a este procesado secuencial impedirían cualquier posible beneficio al emplear estos métodos.

En segundo lugar, probamos a lanzar solo un hilo extra para la captura de la señal y el control de la lectura por SPI. Realizamos un diseño sin exclusividades mutuas para evitar interbloqueos a modo de prueba, y analizamos los resultados. La conclusión fue otra vez similar; mientras el planificador realizaba sus tareas en la escala de los milisegundos,

nuestro temporizador para lectura intentaba tener tiempo de procesado con un periodo de hasta tres órdenes de magnitud mas pequeño, lo cual era inviable.

Si bien es cierto que el primer modelo con hilos sería inviable bajo cualquier circunstancia, como hemos comentado, por la imposibilidad de paralelizar este procesado secuencial, la segunda posibilidad, basada en un modelo de productor-consumidor podría ser factible en un sistema de dos o más núcleos con un sistema operativo a tiempo real con un planificador *preemptive*.

En conclusión, el modelo que finalmente realizaríamos sería uno totalmente secuencial en el que el sistema operativo no invirtiera tiempo cambiando el contexto entre módulos.

3.6. Gráficos

Cuando se trata de la forma en que representaremos gráficamente información en una pantalla, se presentan ante nosotros múltiples posibilidades.

Por un lado, Raspberry Pi dispone de soporte gráfico de famosas librerías como OpenGL ES, OpenVC o EGL, que en estos momentos están bastante integradas y permiten la realización de aplicaciones gráficas sin mucho esfuerzo. Además, el Sistem On Chip (SOC) de la Raspberry Pi, el BCM2835, incluye una unidad de procesamiento gráfico (GPU o VideoCore) con compatibilidad para estas librerías.

Por otro lado, nos encontramos con la opción de dibujar de una forma más directa, rasterizando directamente las primitivas gráficas, escribiendo a bajo nivel en la región de memoria de vídeo que será utilizada para representar los píxeles en pantalla.

Aunque en primera instancia pudiera parecer que la mejor opción es la primera de las citadas, las pruebas realizadas determinan que si bien la paralelización de procesado gráfico que realiza la GPU incorporada en el SOC es beneficiosa para realizar aplicaciones gráficas de alta complejidad, no resultan sin embargo ser la mejor opción para un producto de tiempo real como el que nos ocupa. Para comprender esto hay que tener en cuenta tres factores:

- En primer lugar, estas librerías son complejas, enfocadas en aplicaciones principalmente gráficas y no son ligeras ni en memoria ni en procesado (no debemos olvidar que los gráficos no son la prioridad de nuestro producto, y por ello el tiempo invertido en representar imágenes debe ser despreciable). Con estas librerías el ratio

entre el consumo de ciclos de CPU y el número de píxeles modificados por fotograma es demasiado alto, ya que están optimizadas para cálculos complejos (en los que el beneficio empieza a ser evidente), no ligeras variaciones en la pantalla (no necesitamos refrescar la pantalla completa, solo escribir algunos nuevos valores).

- En segundo lugar, este tipo de librerías involucran toda una tubería gráfica o pipeline (que incluye transformaciones matriciales, proyecciones, texturizados, etc), que resulta demasiado aparatosa cuando lo único que necesitamos es variar algunos píxeles de la pantalla de forma lo más veloz posible. No nos podemos permitir, además, el consumo de ciclos de CPU que involucra, por pocos que sean, una tubería de este tipo, cuando nos encontramos ante una aplicación a tiempo real de alta frecuencia.

- Por último, una de las mayores desventajas del uso de cualquiera de estas librerías gráficas es que se apoyan en el interfaz gráfico de usuario (GUI) de Linux, llamado X Window, y que por el hecho de estar funcionando implica una sobrecarga al sistema excesiva en nuestro ámbito, desde el punto de vista de ciclos de reloj (por ser esta una aplicación de tiempo real) y de consumo de energía (por ser un sistema empotrado).

Es por esto que nuestra opción predilecta resulta ser la creación de una librería gráfica a bajo nivel, que nos permita representar información con libertad pero que no colisione con el interés por un procesado de los datos a tiempo real.

El modo de realizar esta librería gráfica es apoyándonos en la herramienta que nos ofrecen los sistemas Linux para comunicarnos con el segmento de memoria de vídeo, conocido como Framebuffer.

Con el Framebuffer correctamente configurado, basta con realizar una escritura en memoria para dibujar un píxel en la pantalla. Además solo dibujaremos en cada fotograma lo estrictamente necesario, y no realizaremos refrescos completos de pantalla para dedicar aun menos tiempo al procesado gráfico.

3.7. Conclusiones

Una vez realizada la labor de investigación previa, se plantea un proyecto que cumpla los siguientes objetivos:

- **Filtrado en tiempo real**: Mediante el uso de filtros en el dominio del tiempo, se permite la filtración de la señal en tiempo real. Esto permitirá muestrear y

analizar la señal durante su captura, siendo este un aspecto que supondrá una mejora sobre los dispositivos actuales, que sólo realizan el análisis a posteriori mediante un software específico.

- **Calidad de captura y análisis**: La captura esperada será de un 1KHz, permitiendo muestrear incluso la señal de un marcapasos, señal que actualmente no es capturada por la totalidad de los dispositivos. El estándar de calidad se mantendrá lo más alto posible, buscándose tener un ratio de muestras perdidas despreciable.
- **Facilidad de adquisición y uso**: Haciendo uso de plataformas de fácil adquisición se permite a cualquier usuario disponer del dispositivo. Asimismo se plantea el uso de un sistema operativo de uso sencillo y con alta aceptación entre el público.
- **Bajo coste**: Los dispositivos utilizados en el proyecto no tendrán un coste excesivo. El ideal del proyecto es llevar las más altas prestaciones al mayor número de individuos, siendo imperativo para ello que el precio de adquisición sea lo más reducido posible.

El modelo de comportamiento de la aplicación [Fig 3.2] implica las siguientes características:

- **Múltiples entradas de datos**: La captura de datos podrá hacerse desde un paciente o mediante un fichero, dándose de esta manera la posibilidad de procesar datos en tiempo real o manejar datos ya capturados.
- **Diseño modular**: Para permitir la posible expansión del proyecto, se hará uso de una arquitectura modular. Esta idea facilita además hacer uso de cada módulo en proyectos externos, ampliando así el alcance del desarrollo.
- **Almacenamiento permanente de datos**: Todas las capturas realizadas podrán almacenarse en dispositivos de almacenamiento no volátil, en este caso, la tarjeta SD de la Raspberry Pi.
- **Interacción con el usuario**: La interacción con el usuario será sencilla e intuitiva, se pretende que el aprendizaje para el uso de la aplicación sea mínimo.

Figura 3.2: Diagrama general de la aplicación

Capítulo 4

Desarrollo

Este capítulo explica cómo se han implementado los diferentes módulos y funcionalidades que cubre el proyecto. Se hace una exposición tanto del código desarrollado en base a la investigación llevada a cabo, como del proceso y la planificación que se ha seguido a lo largo de los meses que ha durado el proyecto.

4.1. Diseño y Arquitectura

La estructura de la arquitectura está formada por tres secciones independientes:

- **Comunicación mediante SPI**.
- **Procesamiento de datos**.
- **Visualización por pantalla**.

Existe además una cuarta sección que actúa como nexo y controla el flujo de ejecución.

4.1.1. Diseño Modular

El diseño está basado en la idea de obtener módulos o librerías cuya funcionalidad no estuviese condicionada al resto del proyecto. Esta máxima se persigue con el fin de permitir la ampliación de la funcionalidad del código generado, así como su uso en futuros proyectos.

Una ventaja adicional de este modelo es la facilidad para la paralelización del desarrollo, pudiendo trabajarse en distintos módulos de manera simultánea.

Comunicación entre módulos

El paso de datos y las etapas de ejecución están definidas por la existencia de datos nuevos o la posibilidad de almacenar más datos, y para esto hacemos uso de dos bufferes de datos, implementadas tal cómo se explica en el anexo E.2.2 e interactuando con el resto de la aplicación según la representación de la figura 4.1.

En el primer buffer almacenamos los datos en bruto según los adquirimos de la fuente de entrada, de manera que el módulo de comunicación SPI se comunica con la aplicación escribiendo los datos capturados en este buffer.

En el segundo buffer se guardan los resultados de aplicar los filtros a las muestras del primer buffer.

El módulo de visualización por pantalla tiene acceso a los dos buffers de forma que puede mostrar cualquier señal, tanto aquellas sin filtrar (primer buffer), como las filtradas (segundo buffer). De la misma manera el procesamiento de datos accede a los dos buffers

para aplicar los filtrados y delinear la señal.

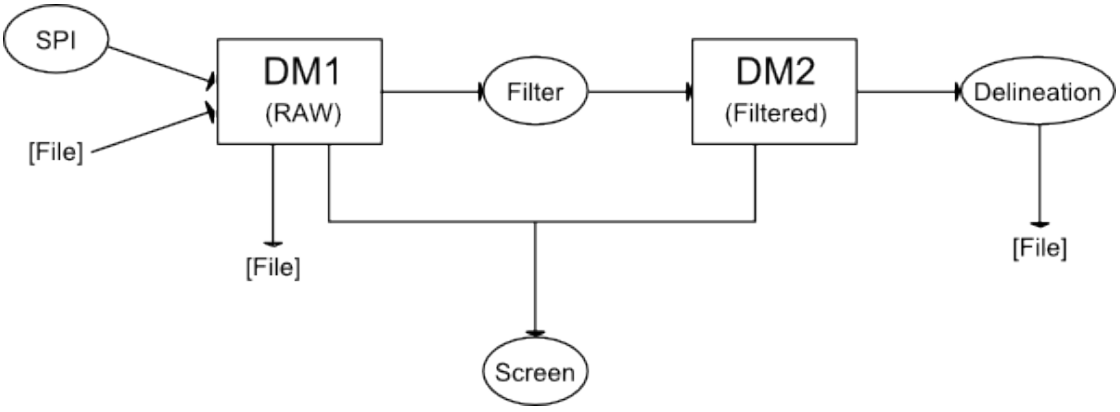


Figura 4.1: Diagrama de interconexión de modulos

4.1.2. Flujo de Ejecución

La ejecución en primer lugar requiere una inicialización de los módulos previamente mencionados:

- Configurar la placa ADS1198 para que comience la transmisión de datos.
- Abrir los ficheros necesarios.
- Preparar la pantalla para mostrar la señal.
- Configurar los filtros que se van a utilizar.
- Inicializar los buffers de datos.

Una vez configurado todo, la ejecución entra en un bucle en el que confluyen los siguientes apartados y del que sólo se sale cuando acaba la ejecución para proceder a cerrar todos los módulos de manera segura.

Entrada de datos

Hay dos opciones excluyentes para volcar muestras al primer buffer de arrays intermedios (datos RAW) o también referenciado como buffer de bloques:

- En el primer caso, obtenemos los datos del chip ADS1198 mediante el módulo de comunicación SPI, escribiéndose en el primer buffer según se va muestreando. Esta ejecución la realiza la rutina ejecutada por el temporizador, por lo que su flujo es independiente del bucle principal.

- En el segundo caso leemos de un fichero de entrada una muestra por cada iteración del bucle, volcándose igualmente al primer buffer.

En ambos casos la escritura se realiza dato a dato y una vez completado el bloque actual se convertirá en un bloque válido para el procesamiento.

Procesamiento de señal

En cada iteración del bucle se comprueba el estado de los buffers de datos para ver si hay suficientes datos cómo para procesarlos. En caso de haber un bloque disponible en el primer buffer (datos RAW) procedemos a guardar el bloque en el fichero de datos, a aplicar los filtros y a escribir los datos resultantes en el segundo buffer. Al acabar marcamos el bloque cómo leido ya que hemos acabado de realizar operaciones.

Si hay bloques preparados en el segundo buffer (datos filtrados) los llevamos al módulo de delineación para extraer la información y posteriormente guardamos en fichero los datos del análisis.

Interacción con el usuario

El acceso a la información de la captura de la señal y el procesamiento viene determinado por la configuración con la que se haya lanzado la ejecución. Las formas que tenemos de consultar los resultados sería mediante salida por pantalla o por fichero.

La salida por pantalla permite cambiar el aspecto de la señal haciendo distintos zooms y elegir el canal que se esta representando mediante teclado.
La otra forma de obtener los datos es mediante ficheros que vendrán dados a razón de una muestra de la señal por cada fila en el caso del fichero de la señal RAW y con el formato descrito en la tabla 4.1 por fila para los ficheros de delineación.

En el caso de detectar el marcapasos vendrá indicado como tres columnas añadidas a la información anterior con el formato expuesto en la tabla 4.2.

Q	R	S	T_{on}	T	T_{off}	P_{on}	P	P_{off}

Cuadro 4.1: Formato de fichero para la delineación

M_{on}	M	M_{off}

Cuadro 4.2: Extensión del formato de fichero para la delineación

Por último, independientemente de la configuración, el teclado nos va a permitir cerrar la aplicación de manera controlada para no perder datos y que no de errores de memoria.

4.1.3. Librería SPI y captura de la señal

La comunicación SPI sirve de interfaz entre los dos dispositivos utilizados en nuestro proyecto, la Raspberry Pi y el chip ADS1198. Para realizar un módulo lo más general posible, se ha optado por generar una librería que abarque la parte relacionada con la Raspberry Pi, siendo este el elemento que actúa como maestro en la comunicación. Como el chip ADS1198 es el elemento que actua como esclavo, se ha creado una interfaz propia para su conexión.

Esta librería facilita una conexión futura con otros dispositivos, haciéndose necesario únicamente desarrollar una interfaz concreta para cada dispositivo a conectar.

En los siguientes apartados se expone con más detalle cada elemento desarrollado, explicándose además el uso de señales y temporizadores para la captura de los datos.

Librería SPI

La Raspberry Pi hace uso de los pines de propósito general para realizar la comunicación mediante SPI. Por este motivo se ha hecho necesario trabajar en funciones que permitan la modificación del comportamiento de los pines, enriqueciendo además la funcionalidad que se presta.

Desde el punto de vista de la transmisión de información se han desarrollado funciones para que únicamente sea necesario especificar los datos a enviar, evitando así que el usuario requiera de un conocimiento del protocolo SPI.

La configuración de la conexión si requiere un poco más de conocimiento, pues se necesitan introducir tanto la velocidad del envío de datos, como la polaridad y la fase del reloj. La señal de chip select es propia de la Raspberry Pi, siendo necesario especificar que pin deseamos que ejerza esta función.

Como funciones generales, se ofrece un servicio de inicialización y otro desconexión, haciendo de este modo más cómoda la manipulación de los pines. Este modelo de ejecución enmascara la gestión de memoria interna, evitando así que se puedan quedar recursos de memoria sin liberar al finalizar la ejecución.

Internamente se ha necesitado hacer uso de la función *mmap* de Linux para realizar el mapeo de las regiones de memoria a utilizar.

El cuadro 4.3 resume las funcionalidades prestadas por la librería.

Funcionalidad de la librería de comunicación	
Reserva de memoria y mapeo de las regiones de memoria utilizadas Liberación de memoria y desmapeo de las regiones de memoria utilizadas	
Funciones GPIO	**Funciones SPI**
Configuración de pines (entrada/salida) Lectura en pines de entrada Escritura en pines de salida	Configurar conexión (CPHA, CPOL, CS) Configurar la señal de reloj (SCLK) Envío de tramas de datos

Cuadro 4.3: Resumen de la funcionalidad de la librería de comunicación

Interfaz ADS1198

El chip ADS1198 se configura mediante el envío de comandos por SPI, todos ellos disponibles desde las funciones de la interfaz implementada. Los comandos se dividen en los siguientes grupos:

- **Comandos de sistema**
  - **WAKE UP**: Despierta la placa de un estado de standby, en caso de estar ya despierta no tiene ningún efecto.
  - **STANBY**: Provoca la entrada de un estado de stanby, este estado es útil para ahorrar energía.
  - **RESET**: Resetea la placa dejando los registros a su valor inicial.

  - **START**: Empieza la conversión de datos, en caso de estar ya empezada fuerza la vuelta a empezar.
  - **STOP**: Para la conversión de datos, en caso de estar parado no tiene ningún efecto.

- **Comandos de lectura de datos**
  - **RDATAC**: Activa el modo de lectura continua de muestras.
  - **SDATAC**: Para el modo de lectura continua de muestras.
  - **RDATA**: Lectura de datos bajo demanda, sólo tiene efecto si el chip no está en modo de lectura contínua.

- **Comandos de lectura/escritura de registros**
  - **RREG**: Lectura de datos de registro.
  - **WREG**: Escritura de datos en registro.

Para agilizar la configuración se proporcionan funciones completas de inicialización. Se permiten capturas de datos de manera continua o bajo demanda. Hemos creído conveniente abstraer la configuración de los registros internos, ya que es una labor tediosa y que requiere conocimiento avanzado del chip, cosa que el usuario no tiene por qué tener.

En cuanto a la configuración propia de SPI del ADS1198, es necesario configurar la fase (CPHA) a 1 y la polaridad (CPOL) positiva, es decir, a 0. La velocidad de la conexión se establece a 15.625MHz cuando se está en modo de lectura continua y a 3.90MHz en el envío de comandos de sistema o de lectura/escritura de registros. La diferencia de velocidades está motivada por el tiempo de decodificación que necesita la placa a la hora de procesar un comando, siendo este de 1.96 μsegundos. Si forzamos el envío a una frecuencia superior a 4Mhz, el tiempo de decodificación será superior al de llegada de cada byte, por lo que se haría necesario introducir esperas entre bytes.

Captura de datos

La captura de datos se hace mediante el uso de un temporizador de Linux[1]. En la creación del timer hacemos uso de la constante CLOCK_REALTIME, que fuerza la ejecución con un reloj de tiempo real, permitiendo así una mayor precisión temporal.

[1]Las funciones que ha sido necesario utilizar para la creación y configuración del temporizador han sido timer_create y timer_settime. Ambas funciones se encuentran detalladas en la página man7.org

La frecuencia de las llamadas se define en función de la velocidad de muestreo deseada, cumpliéndose la relación de la ecuación:

$$Intervalo(ns) = 1/(Frecuenciademuestreo) * 10^9$$

Es posible que un temporizador no se ejecute debido a que la señal enviada por el sistema operativo se pierda, para contabilizar el número de ejecuciones perdidas Linux proporciona la función *timer_getoverrun*, que nos sirve para definir una medida de calidad en función de las muestras que no podemos procesar.

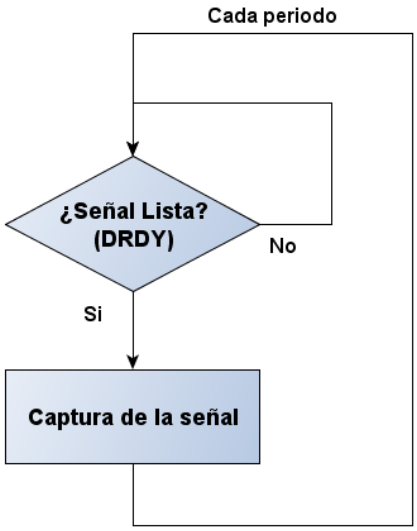


Figura 4.2: Diagrama de captura mediante temporizador

El diagrama 4.2 ilustra el funcionamiento del temporizador para la captura de las señales. Una vez se procede a la adquisición de la señal, se ha de esperar de forma activa a que la señal $\overline{\text{DRDY}}$ esté a 0, indicando así que el chip ADS1198 tiene lista una nueva captura.

4.1.4. Filtrado de la señal

Para la correcta delineación de un electrocardiograma es recomendable filtrar la señal para que el ruido que pueda tener no interfiera en la detección de picos.

Tal como se indicaba en las decisiones de diseño se ha optado por un filtro morfológico para quitar la componente continua y un filtro paso bajo basado en una función de

transferencia.

El resultado de aplicar ambos filtrados a la señal superior de la figura 4.3 es la señal sin componente continua y sin ruido que aparece en la misma figura en la parte inferior.

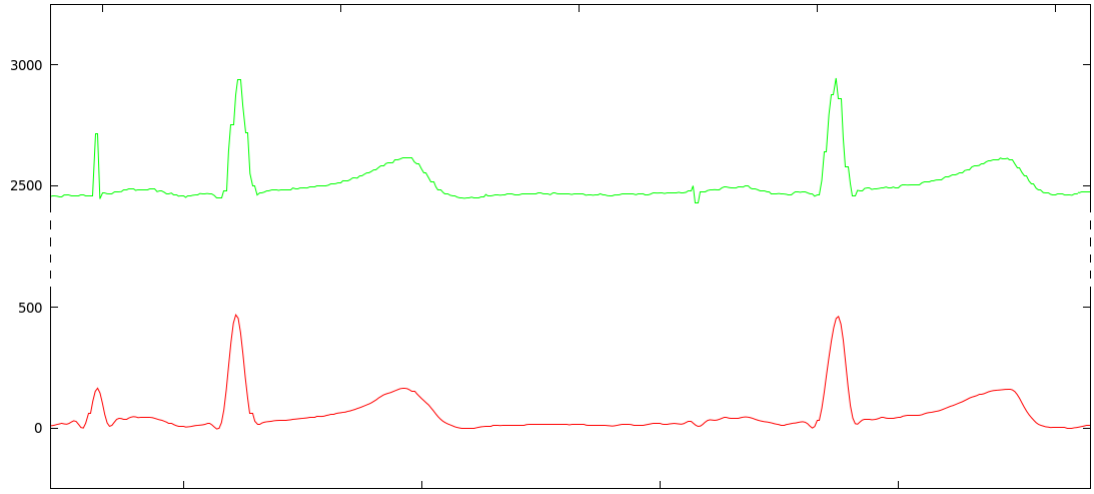


Figura 4.3: Ejemplo de aplicación del filtrado completo

Baseline

Para filtrar la componente continua de la señal se utiliza un filtro morfológico *baseline*[7]. En primer lugar se realiza una apertura (eq 4.3) a la señal f_o para quitarle los picos y a la resultante un cierre (eq 4.4) para quitarle los hoyos resultando una señal f_b que nos va a definir la linea sobre la que oscila la señal original.

La corrección baseline resulta al restarle a la señal original f_o la señal sin picos ni hoyos f_b. Se puede ver un ejemplo completo en el que se ven todos los pasos y las señales resultantes en la figura 4.4.

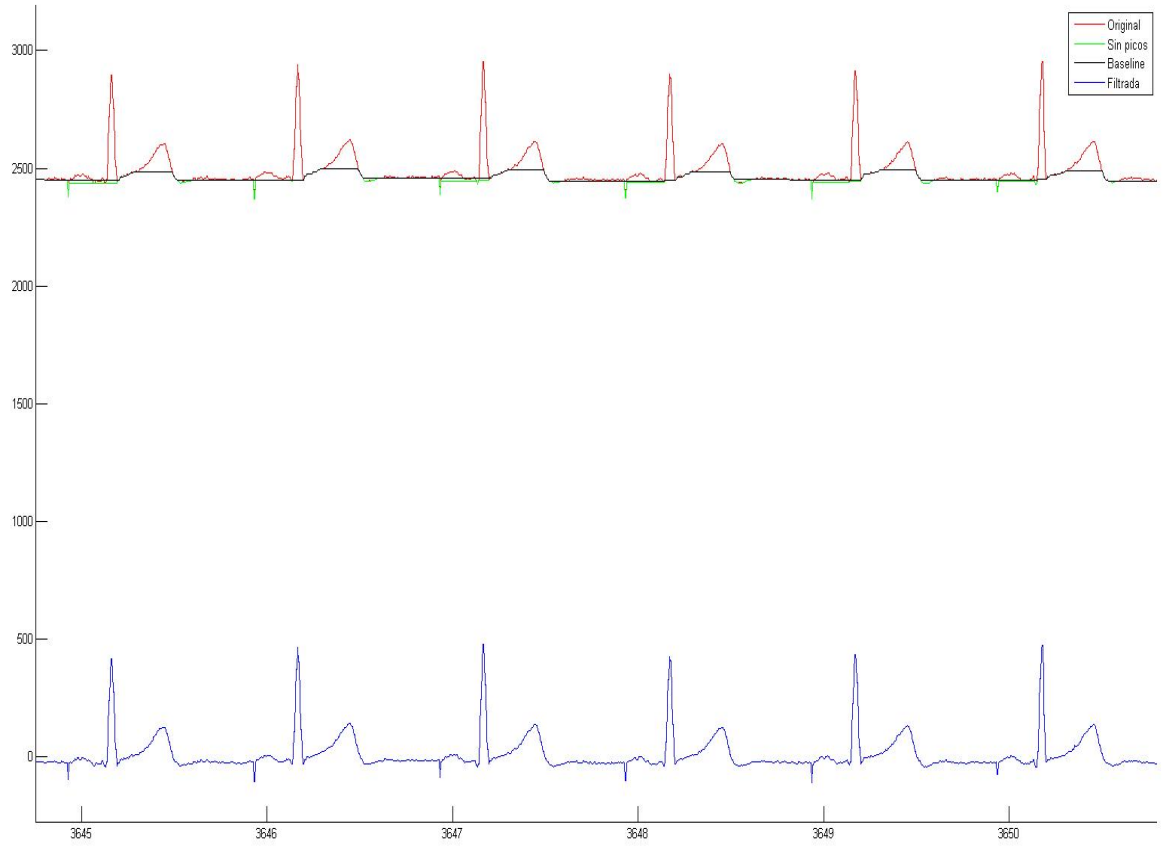


Figura 4.4: Diferentes señales resultantes en el proceso del aplicado de Baseline

Siendo $f(n), n = 0, 1, ..., N-1$ una señal discreta de N puntos, y $B(m), m = 0, 1, ..., M-1$ una estructura simetica de M puntos, se definen las operaciones:

$$erosion : (f \ominus B)(n) = \min_{m=0,...,M-1} \left\{ f\left(n - \frac{M-1}{2} + m\right) - B(m) \right\} \tag{4.1}$$

$$dilatacion : (f \oplus B)(n) = \max_{m=0,...,M-1} \left\{ f\left(n - \frac{M-1}{2} + m\right) + B(m) \right\} \tag{4.2}$$

$$apertura : f \circ B = f \ominus B \oplus B \tag{4.3}$$

$$cierre : f \bullet B = f \oplus B \ominus B \tag{4.4}$$

Para la ejecución en tiempo real se utilizan cuatro buffers circulares con ventanas en las que se van calculando máximos y mínimos [E.1.1]. Hay un buffer para cada operación que hay que realizar (dilataciones (eq 4.2) y erosiones (eq 4.1)) y el resultado de cada una de estas operaciones va a ofrecer el dato con el que operará el siguiente buffer.

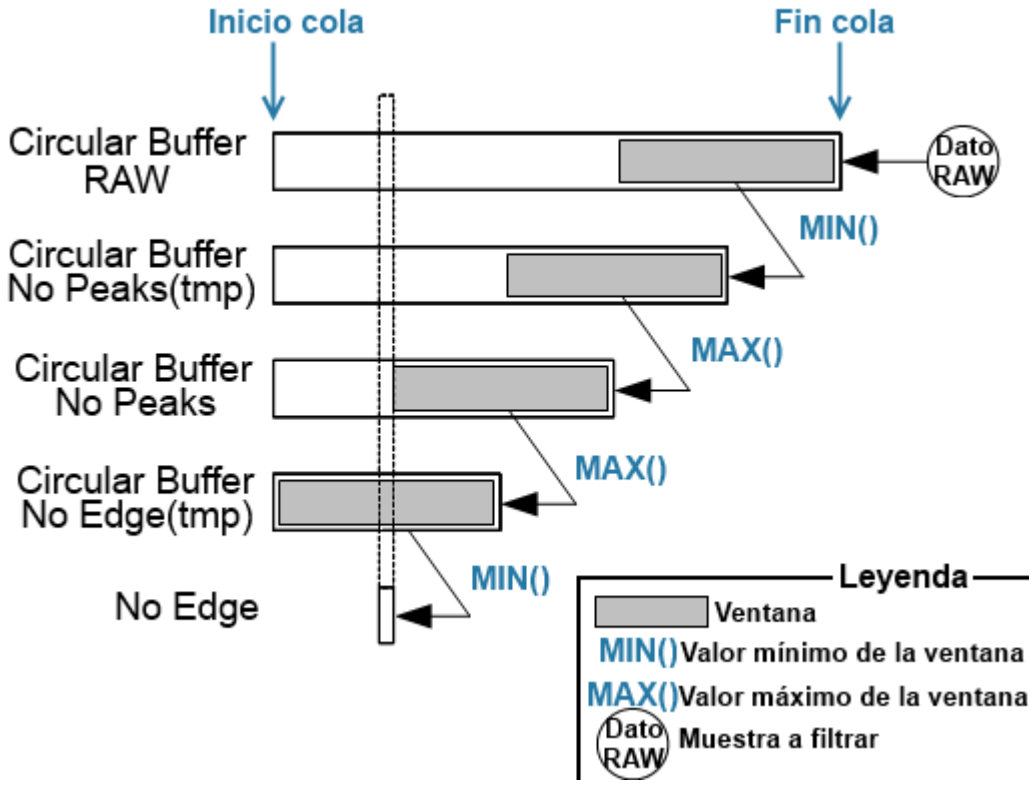


Figura 4.5: Estructura del filtrado Baseline

En la figura 4.5 se muestran alineados de forma que se ve que posiciones equivalen a las mismas muestras despues de diferentes operaciones aplicadas. Los datos que correspon-

den al mismo instante de tiempo son los recogidos por el rectángulo que atraviesa las cuatro colas circulares en la figura

Como necesitamos restar el resultado de las operaciones morfológicas a la señal original, necesitamos guardar suficientes datos como para poder compararlos posteriormente. Por este motivo, si los buffers fuesen todos del mismo tamaño, se perderían los datos originales. La relación de tamaños es de $size + \frac{size}{2}i$ siendo $i \in [3.,0]$ el número de buffer.

El filtrado correspondiente a un dato no aparece hasta que se han operado muestras por el equivalente de la mitad de la ventana. Como al principio de la ejecución todavia no hay datos insertados el filtrado devuelve datos no validos hasta que se han introducido $\frac{4}{2}size - 1$ muestras, correspondientes a los cuatro saltos que existen entre las colas circulares por las que pasa una muestra.

FiltFilt

Para quitar el ruido se usa un filtro paso bajo basado en una función de transferencia, que desfasa la señal, por lo que utilizamos la técnica de FiltFilt que consiste en aplicar el filtro de izquierda a derecha y posteriormente de derecha a izquierda sobre el resultado para anular el desfase aplicado por el filtro. Para el analisis en tiempo real se aplica sobre una ventana, implementada mediante un array que cuando esta lleno ejecuta los filtrados [Fig 4.6].

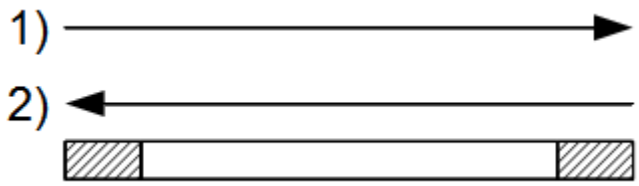


Figura 4.6: Estructura del filtrado Filtfilt

Los diferentes coeficientes para las funciones de transferencia los hemos precalculado con Matlab con el diseño de filtros usando el método de la ventana[2]. Por ejemplo, el filtro a 8KHz con frecuencia de corte 40Hz es el representado por el diagrama de bode de la figura 4.7 en el que podemos apreciar el desfase de la señal.

[2]fir1: Fir filter design using the window method

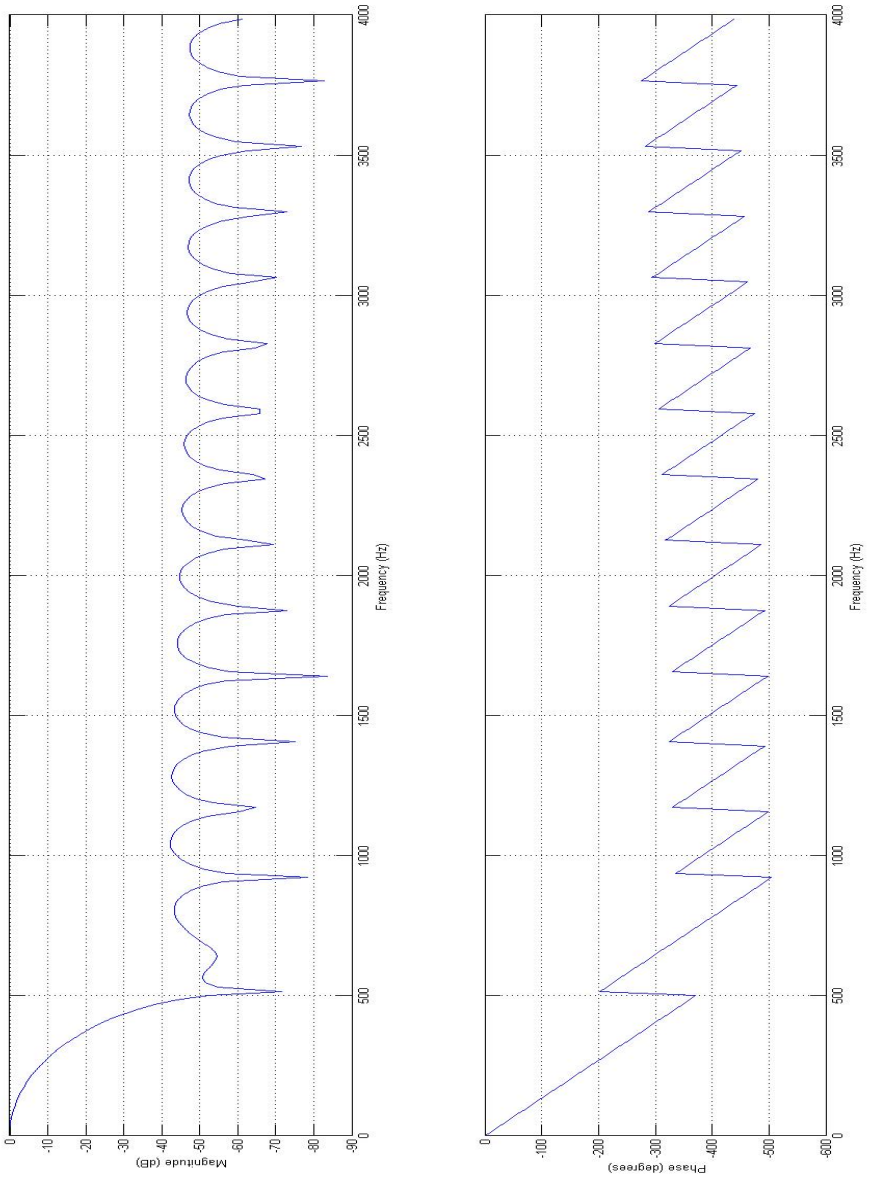


Figura 4.7: Diagrama de bode de un filtro con frecuencia de corte 40Hz a 8KHz

Se ha optado por filtros de orden 33, lo que deja datos no válidos tanto al principio cómo al final de la ventana ya que el filtro necesita como mínimo 33 datos para ofrecer resultados.

Estos datos no válidos estan representados por las zonas sombreadas en la figura 4.6, y hay que rescatar esos datos y reutilizarlos para permitir la continuidad de la señal. Para ello encadenamos las ventanas como se indica en la figura 4.8, reutilizando los datos originales correspondientes a las dos zonas sombreadas donde se encadenan las ventanas. La señala filtrada es la que corresponde a la unión de las partes válidas de las ventanas.

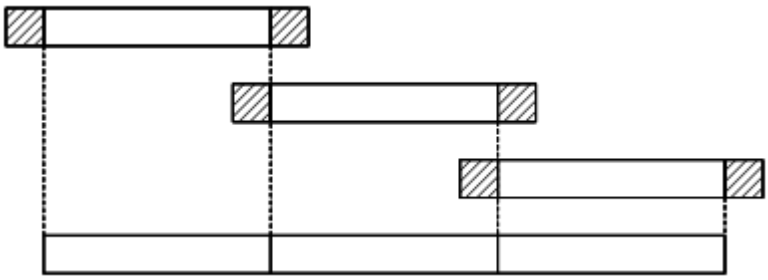


Figura 4.8: Encadenamiento de resultados del filtrado Filtfilt

Aun con el encadenamiento hay que tener en cuenta que este método deja un número de datos inválidos al principio de la señal por valor de $coeficientes+1$ debido a las primeras muestras en las que no hay suficiente historia como para aplicar el filtro correctamente.

El filtro necesita tener acceso a los últimos n valores que ha tomado la señal, y los m últimos valores que ha dado como resultado el filtro. Por eso está implementado con dos bufferes circulares [E.1] que almacena los valores que se filtran y sus resultados.

4.1.5. Delineación de la señal

Para delinear la señal nos apoyamos en su crecimiento y decrecimiento, calculando sus derivadas y definiendo unos marcadores que indiquen en que zonas crece o decrece de manera abrupta.

Esta información, además de ubicar los posibles picos, define unos patrones característicos para distintos picos que va a ser una de las bases para su identificación.

Se han realizado pruebas con otro método de detección basado en la morfología de los picos, que presentaba muy buenos resultados iniciales pero que finalmente no se ha implementado por no conseguir funcionalidad suficiente, aunque se presenta más adelante como parte del desarrollo como base de una posible ampliación.

En la figura 4.9 se puede observar un ejemplo del cálculo de los marcadores de crecimiento (linea verde) y los picos detectados (cruces azules) en un intervalo RR[3].

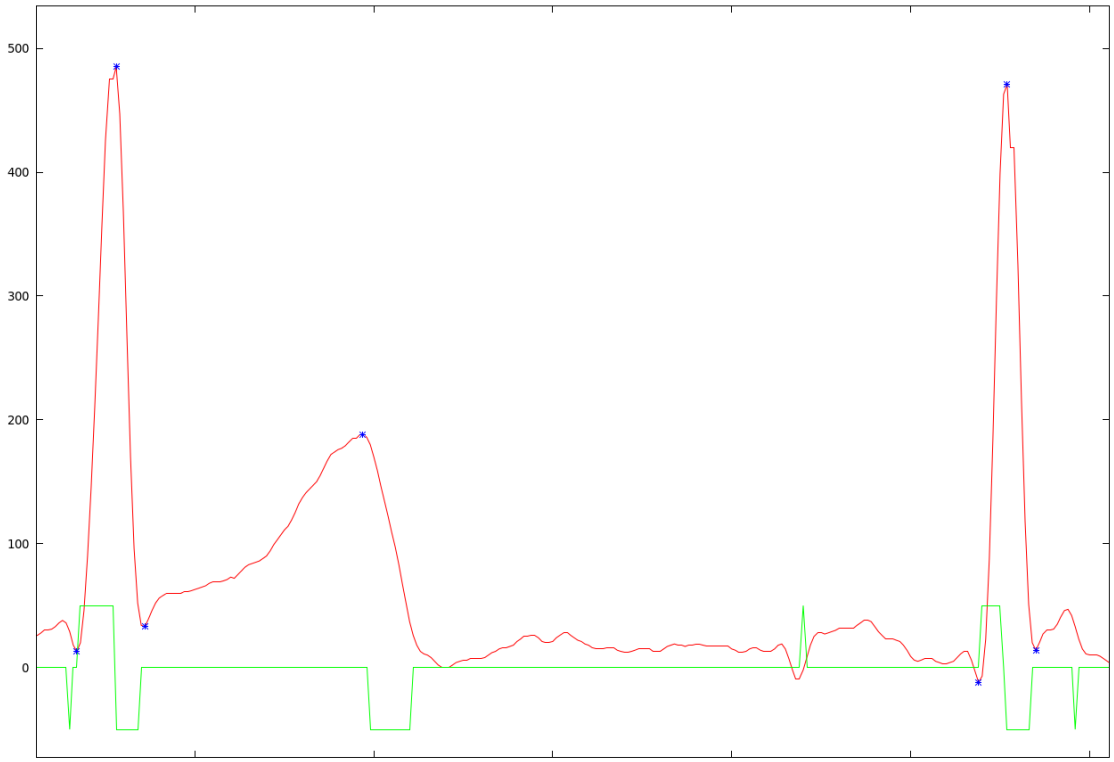


Figura 4.9: Cálculo de los marcadores de crecimiento y picos localizados

Cálculo de las derivadas y marcadores de crecimiento

Por cada dato de la señal vamos a calcular la derivada del valor anterior basándonos en la definición geométrica y teniendo en cuenta que el diferencial de tiempo es siempre constante, por lo que se calcula como la diferencia entre la muestra nueva y la anterior. Calculamos la segunda derivada de manera análoga con los datos guardados de la primera derivada.

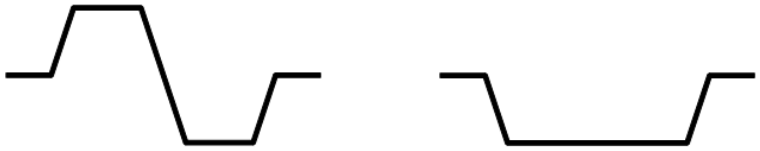
[3]Un intervalo RR es la señal definida desde un pico R hasta el pico R siguiente

Con las derivadas calculadas definimos unos marcadores de crecimiento que nos indican en que segmentos de la curva crece o decrece superando un umbral (para descartar crecimientos leves o ruido).

Detección de picos basado en marcadores de crecimiento

Las zonas donde los marcadores indican crecimiento o decrecimiento corresponden a los diferentes segmentos donde pueden existir picos y la forma que toman estos marcadores nos da información de que tipo de pico puede tratarse. Además, la posición relativa de los picos nos permite ubicar unos respecto a otros.

Observando diferentes ejemplos de marcadores de crecimiento se llega a la conclusión de que se dan ciertos patrones [Fig 4.10] para los picos R y T, por lo que la delineación de la señal se basa en la aparición de estos patrones.



(a) Patrón característico del pico R y del marcapasos.

(b) Patrón característico del pico T.

Figura 4.10: Patrones caracteríticos de los picos

El patrón que identifica los picos R también identifica el marcapasos, por lo que hay que hacer un análisis posterior basado en la amplitud del pico para diferenciarlos, ya que es mayor en el pico R. Su posición respecto a otros picos así como el tiempo transcurrido y el orden en el que se suceden también ayuda a la detección.

Por otra parte, el patrón característico del pico T indica la bajada del pico, lo que nos marca el offset de este pico. Para calcular la posición del pico nos desplazamos hacia la izquierda en la señal mientras esta tenga valores mayores.

Para el cálculo de los onset y offset de los picos (incluidos el caso particular de los picos Q y S) desde la posición de cada pico buscamos a izquierda y a derecha mientras los valores sean menores llegando a los puntos donde empiezan y acaban los picos.

Determinación de picos basada en la morfología

Los marcadores de crecimiento no es la única información útil que podemos rescatar de la señal, si no que si encuadramos los picos en un rectángulo, el aspecto de este (la proporción entre altura y anchura) nos ofrece información sobre que tipo de pico se trata. Así, rectángulos con un aspecto alto corresponderan a picos del tipo R o del marcapasos mientras que un aspecto unidad o bajo indicarán picos T ó P.

Por otra parte, el rectángulo que resulta nos da más información si nos fijamos en su area, ya que rectángulos de area muy pequeña corresponderan a ruidos de la señal que no hayan sido eliminados y nos permitirá descartarlos.

4.1.6. Librería Gráfica

Durante la ejecución, si la aplicación se encuentra realizando un procesado con la salida por pantalla activada, se puede ver una representación del ECG capturado a tiempo real, en cualquiera de los canales que estén seleccionados (o la señal filtrada si está disponible). En esta sección se presenta el modo en que se consigue representar esa información en un monitor.

Framebuffer

Como fue comentado previamente, nuestra decisión de diseño comprendía la realización de una librería gráfica a bajo nivel, fundamentada en el empleo del Framebuffer que nos ofrece Linux.

La memoria de vídeo se encuentra en un segmento de la memoria principal y es allí donde se encuentran almacenados los píxeles[4] que se representarán en pantalla. El Framebuffer es un dispositivo virtual que nos permite comunicarnos con esta memoria y utilizarla.

Esta región de memoria funciona como si fuera un Array. Se dispone de un puntero que apunta al principio, y es posible saber cuánto ocupa la región completa. Además, como una imagen es realmente una matriz (la pantalla también lo es, de píxeles), se dispone de información que, a su vez, indica la longitud de cada línea, con lo que es posible calcular el desplazamiento (offset) concreto para cada píxe l[Fig 4.11].

[4]La palabra pixel proviene de la unión de las palabras picture y cell, que significan celda o célula de imagen. Las pantallas están compuestas por miles de estos píxeles que emiten luz de los colores rojo, azul y verde, que los conos y bastones del fondo del ojo perciben para ver la imagen representada.

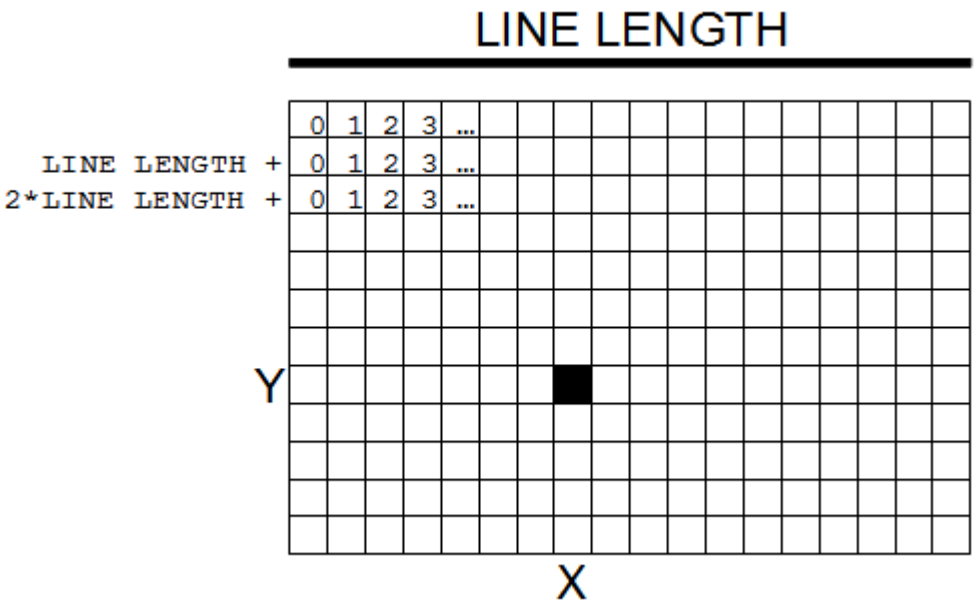


Figura 4.11: Desplazamiento de un pixel en memoria

Lo único que falta por tener en cuenta en este cálculo es el espacio que realmente ocupa cada píxel. Si este espacio en memoria es diferente a un byte, se debe tener en cuenta, pues será un factor de escala para cada desplazamiento.

Con lo visto, la siguiente ecuación calcula el desplazamiento de un píxel concreto respecto al origen de su región de memoria, dadas las coordenadas X e Y y los bits que ocupa cada píxel, y el tamaño de una línea, es la siguiente:

$$Pixel\ offset = x * Bytes\ Per\ Pixel + y * LINE\ LENGTH$$

Además, el Framebuffer de Linux actúa como un dispositivo virtual, por lo que de él se puede extraer toda la información necesaria para trabajar de forma directa con memoria y desplazamientos.

Dibujado en pantalla

Se ha visto previamente la ecuación que permite conseguir el desplazamiento para escribir información en un byte de color o píxel concreto. Es cierto que de esta forma ya podemos escribir información directa en ese píxel, pero antes es necesario comprender qué representa esa información.
El concepto de densidad de color se refiere a la cantidad de colores representables por la

información contenida en un píxel. De este modo, si se dispone de 8 bits (un byte) para representar colores en cada píxel, es posible representar un total de 2^8 colores, es decir, 256. Cuanta más bits en memoria ocupe cada píxel, más información podrá contener y más realista será la imagen obtenida con la combinación de esos colores.

La Raspberry Pi ofrece soporte para densidades de color de 8 bits (indexada con paleta de 256 colores), 16 bits (de los cuales 5 bits son para rojo, 6 para verde y 5 para azul) y 24 bits (8 bits para cada componente de color, rojo, verde y azul).
Aunque nuestra librería gráfica da soporte para densidades indexadas de 8 bits[5] (256 colores), se ha decidido, por motivos de calidad gráfica, dejar por defecto una densidad *truecolor* de 24 bits (16.777.216 colores). Esto implica que al escribir un píxel mediante el uso de la fórmula de desplazamiento de píxel vista anteriormente, se deben escribir tres bytes consecutivos en memoria que indiquen en orden las tres componentes de color (rojo, verde y azul)[Fig 4.12].

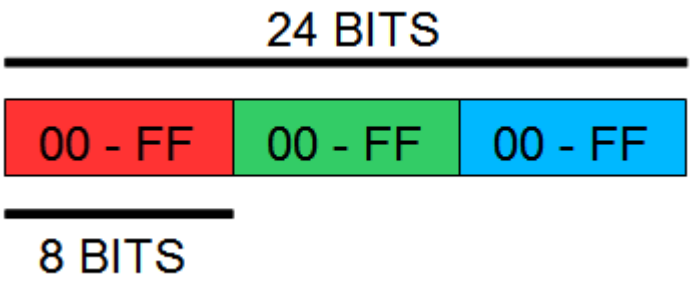


Figura 4.12: Densidad de color de 24 bits

Del mismo modo que son escritos colores en cada píxel, se podrían borrar escribiendo, por ejemplo, el color de fondo. Sin embargo también se ofrece la posibilidad de un borrado rápido de toda la pantalla mediante la llamada al sistema *memset*, que escribe en toda la región de Framebuffer el mismo valor para cada byte. Del mismo modo también se ofrece funcionalidad para borrar filas y columnas concretas, para poder realizar la labor de refresco de una forma mucho más veloz.

Puesto que esta librería pretende dar soporte especialmente a la rasterización de ondas electrocardiográficas, a parte de puntos (píxeles) se debe poder representar otra pri-

[5] Antiguamente esta era la densidad utilizada, ya que no se disponía de procesadores gráficos con mayor ancho de palabra. Se disponía de paletas (o índices) de colores referenciados por el valor del pixel, y estas paletas se cambiaban dinámicamente para ofrecer el efecto de mayor número de colores.

mitiva, la línea recta (para unir dos puntos de la onda). En nuestro caso, cada punto representado de la onda en la pantalla siempre se encontrará en la posición inmediatamente contigua, en el eje X, al punto anterior.
Es decir, que para representar las líneas que unen dos puntos consecutivos de la onda, solo habrá que avanzar una posición en el eje X.
Esta premisa es muy importante, porque permite llevar a cabo una simplificación realmente beneficiosa en tiempo del algoritmo para representación de líneas de Bresenham[5][Fig 4.13].
Como siempre, el interés es dibujar consumiendo el mínimo número de ciclos necesarios.

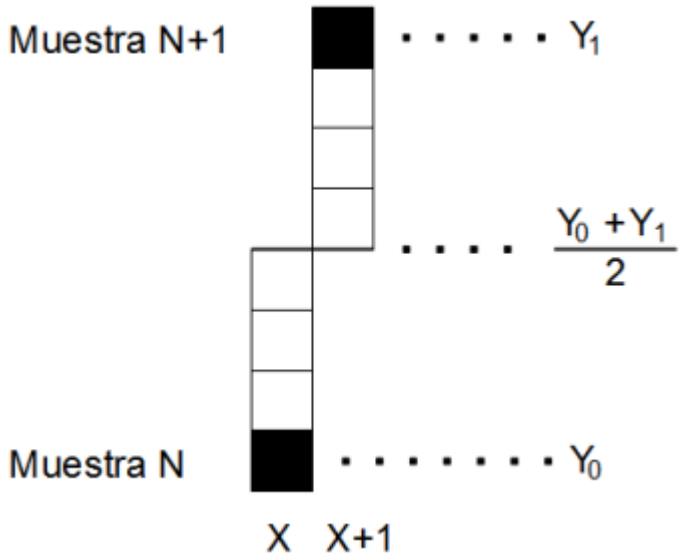


Figura 4.13: Algoritmo para líneas verticales

Por último, también se ofrece la posibilidad de dibujar otra primitiva más, el rectángulo relleno, con el objetivo de que se pueda diseñar el fondo del entorno visual de una forma sencilla e intuitiva.
Como este tipo de primitivas son costosas en tiempo, se deben utilizar solo en la inicialización del módulo.

No se ofrece soporte para refresco de pantalla o doble buffer, puesto que no son empleados por la aplicación. En su lugar, se representa todo el entorno gráfico al comienzo de la inicialización del módulo, y cada vez que es necesario representar un punto de la onda o un cambio en la frecuencia cardíaca, por ejemplo, se borra solamente el espacio de pantalla

absolutamente necesario, y siempre se dibuja lo mínimo posible. Ese es el compromiso del módulo gráfico para no restar tiempo de ejecución al resto de la aplicación.

Rasterizado de texto

Se ofrecen dos posibilidades a la hora de representar texto en pantalla, cada una para un empleo determinado.

En primer lugar, una fuente de texto, de tamaño 8x8, representable en tamaño original y doble.
Para utilizarla basta con llamar a la función correspondiente que se encarga de representar en la posición de pantalla indicada el string pasado por parámetro.
Esta es la fuente más sencilla de utilizar, pero también la más costosa en tiempo (no demasiado, pero más que la segunda posibilidad), y es por esto que solo es empleada en el diseño del entorno y el fondo visual, así como para representar valores que no van a variar constantemente o no lo van a hacer a alta frecuencia.

En segundo lugar hemos diseñado, con el propósito específico de consumir el mínimo tiempo posible, un sistema de numeración basado en los displays hardware de 7 segmentos. De esta forma, la región de pantalla a borrar para actualizar este valor es mínima.
Empleamos estos dígitos de 7 segmentos en la representación de la frecuencia de ritmo cardíaco (Beats Per Minute), ya que esta puede variar en cualquier momento y puede hacerlo muy a menudo.

Conector de pantalla

Es importante poder conectar la librería gráfica al programa principal, integrándola en el módulo de pantalla para representación gráfica. En este apartado trataremos este conector.

Cuando se inicia este módulo, además de lo comentado en el apartado de inicialización (anchura, altura y densidad de color), se deben tener en cuenta los parámetros de la aplicación referentes a:

- Frecuencia de muestreo de la señal
- Fotogramas por segundo a los que queremos dibujar
- Intervalo de tiempo mostrado por pantalla

La función principal de dibujado de esta librería(la cual se declara ante el compilador como *inline* para evitar consumo de CPU ejecutando preámbulo y epílogo para mantener el marco de ejecución) se invoca a la misma frecuencia que el muestreo de la señal, llevando así el módulo gráfico la cuenta de las muestras y el tiempo, aunque solo dibujará realmente cuando corresponda en base a su propia frecuencia descrita en fotogramas por segundo.

Teniendo estos elementos en cuenta, es posible calcular cuántas muestras se deben dibujar por cada fotograma, y cuántas muestras deben pasarse por alto sin dibujar, es decir, el inframuestreo de pantalla, que determina la resolución de la onda visible y será modificable dinámicamente por la aplicación para permitir al usuario ver la señal con más o menos zoom en el eje de tiempo.

Es posible calcular todos estos factores con las siguientes fórmulas:

$$Samples\ On\ Screen = frequency(Hz) * time\ interval$$

$$Samples\ Per\ Pixel = Samples\ On\ Screen / display\ width$$

De este modo, *Samples Per Pixel* representa el número de muestras que deberemos pasar por alto antes de dibujar una muestra en concreto, regulando así la frecuencia de dibujado.
Además, se dibujarán tantas muestras seguidas como correspondan en bloque, a la frecuencia de dibujado en fotogramas por segundo del módulo gráfico.

Este conector del módulo de representación gráfica tiene acceso a la memoria de datos de la aplicación principal, y mediante un sencillo interfaz es capaz de recibir de la aplicación punteros a los nuevos bloques de datos a dibujar, cuando haya que cambiarlos, llevando internamente todo el control y el desplazamiento entre los mismos.
La aplicación podrá decidir si especificar un bloque de datos crudos o filtrados, y podrá elegir también el canal que representar.

Además de la onda ECG, se mostrará por pantalla información extra en una barra lateral, para que el usuario pueda tener acceso inmediato a ella [Fig 4.14]:

- Frecuencia cardíaca (Beats Per Minute)
- Señal activa (canal o filtrado que se encuentra activo)
- Frecuencia de muestreo (en Hz)

Figura 4.14: Barra lateral de información

- Fotogramas por segundo
- Muestras perdidas/mal capturadas en el último segundo
- Escala en eje X (intervalo de tiempo mostrado) e Y (mV, escala grabada en el margen derecho)

4.2. Proceso de desarrollo

Este sección detalla las etapas por las que ha pasado el proyecto, desde su concepción hasta la finalización del mismo. La evolución de los distintos elementos implicados en el desarrollo no es mas que el fruto de una planificación inicial dividida en cuatro etapas o iteraciones. Las iteraciones, así como sus contenidos principales, se exponen en la figura 4.15.

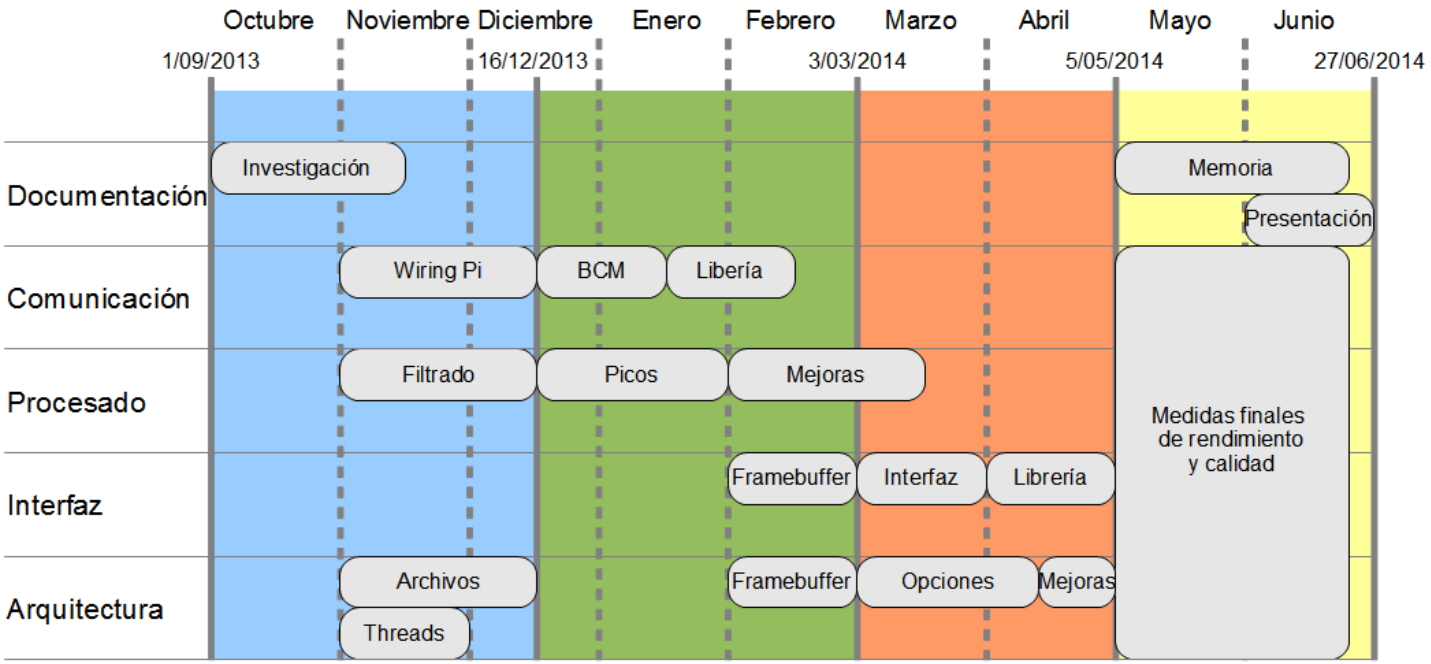


Figura 4.15: Esquema de la planificación anual

Las líneas de trabajo se han agrupado lo máximo posible, quedando sólo las cuatro expuestas a continuación:

- **Documentación**: En este apartado incluimos tanto la investigación realizada como la propia documentación del proyecto. Sólo se incluye este apartado en las iteraciones primera y última, siendo estas donde se ha hecho el trabajo de documentación más extenso. Esto no quiere decir que durante el resto de iteraciones no se hayan realizado las actividades estructurales necesarias, documentación incluida, pues se ha seguido una política de comentarios en el código estricta, así como la continua documentación de manera informal de cada paso dado durante el desarrollo del proyecto.

- **Comunicación**: Esta línea involucra todos los aspectos de conexión entre la Raspberry Pi y el chip ADS1198.

- **Procesado**: Sección que aborda el ámbito de procesado y análisis de la señal.

- **Interfaz**: La interfaz gráfica, que hace uso de una pantalla, se expone en este apartado.

- **Arquitectura**: Los elementos del desarrollo destinados a unificar los distintos módulos realizados se exponen en esta última sección.

A continuación se detalla cada una de las etapas, explicando en cada línea el trabajo realizado. Nos parece necesario añadir en cada apartado de las iteraciones los problemas que nos han surgido, pues son parte importante del proceso de desarrollo y ponen de manifiesto el esfuerzo llevado a cabo para solventarlos.

4.2.1. Iteración 1: Investigación y arranque del proyecto

Documentación

El principio del proyecto engloba toda la investigación llevada a cabo y explicada en el capítulo 2. Se hizo necesario adquirir todo el conocimiento que habíamos de aplicar durante el desarrollo, así como el realizar pruebas que nos dieran bases sólidas sobre las decisiones de diseño tomadas. Este proceso nos sirvió como toma de contacto con las tecnologías a utilizar y nos permitió investigar la información explicada en la sección 2.1 sobre ondas electrocardiográficas.

Comunicación

La comunicación mediante el uso de la interfaz SPI se planteaba inicialmente utilizando la librería Wiring Pi[6]. El uso de este recurso planteaba una comodidad que podía librarnos de tener que realizar nosotros este trabajo, pudiendo invertir ese tiempo en otros aspectos del proyecto.

Al principio no tuvimos problemas con el uso de la librería, que aún haciéndose necesario hacer uso de scripts de configuración mediante el bash de Linux, permitía realizar pruebas iniciales de comunicación. El problema llegó cuando se hizo necesaria empezar la captura de muestras desde el chip ADS1198, la librería no nos daba soporte para las velocidades que teníamos que manejar, por lo que su uso quedó descartado y se hizo necesario buscar una alternativa.

Desde el punto de vista del dispositivo físico, el chip ADS1198 nos dio bastantes problemas iniciales. Como se comenta en el apéndice D la información sobre la señal de RESET no es coherente, estando escrita como **RESET** o $\overline{\textbf{RESET}}$ en diferentes secciones del manual o la propia placa. Dado que la comunicación involucra más líneas, como son **MOSI**, **MISO**, **CS**, **CLK**, **START** y **DRDY**, la combinatoria para saber por qué no conseguíamos capturar señales desde de la placa era altísima.

Finalmente, tras varias semanas probando múltiples combinaciones y cambiando la librería de Wiring Pi por la BCM2835[7], conseguimos configurar la conexión SPI para leer los registros de la placa, no pudiendo todavía capturar señales.

Procesado

Para poder empezar a procesar la señal teníamos que realizar un filtrado previo, eliminando dos aspectos fundamentales: La componente continua y el ruido.

- **Filtro Baseline**: Este es el primer filtro que se llevó a cabo. Su implementación requirió la creación de estructuras combinando buffers circulares de una manera específica tal como se define en la sección 4.1.4, añadiendo complejidad a la implementación.
- **Filtro FiltFilt**: El desarrollo de este filtrado obligó a hacer uso del programa Matlab, sin el cual el calculo de coeficientes habría sido más tedioso. La imple-

[6]Wiring Pi es una librería desarrollada para Raspberry Pi que permite modificar tanto los GPIOs como establecer una conexión SPI. La página del proyecto es www.wiringpi.com

[7]Toda la información relativa a esta librería se encuentra disponible en la página web www.airspayce.com/mikem/bcm2835/

mentación no fue trivial, ya que se hizo necesario testear de manera concienzuda el funcionamiento del filtro y su correcta ejecución.

Arquitectura

Una de las investigaciones, en el ámbito de la arquitectura, que se llevaron a cabo durante esta iteración abarcaba todos los test de división de tareas mediante hilos (threads).

Como podemos ver en la sección 3.5, en primer lugar se realizó un diseño secuencial basado en una segmentación de las tareas de captura, filtrado y procesado de la señal. Los resultados de este diseño no fueron beneficiosos, dado el tiempo invertido en artefactos de control como los *mutex*, muy superior al permisible dada la frecuencia de captura.

En segundo lugar se realizó un diseño basado en dos hilos; uno para el procesado y el otro para la captura. También resultó infructuosa esta vía de trabajo, dado el tiempo consumido por el planificador en el cambio de contexto.

Como no se podía capturar señales desde la placa, tuvimos que empezar el desarrollo del sistema de ficheros, tanto la captura de datos como el guardado de los mismos.

Para poder visualizar los ficheros de señales, hicimos uso de la herramienta *gnuplot* de Linux[8]. Esta aplicación se ejecuta desde la línea de comandos y permite representar una gráfica en función de la información de un fichero.

4.2.2. Iteración 2: Estructura básica de la aplicación

Comunicación

Una vez se permitía la comunicación básica con la placa empezamos el estudio de por qué seguíamos sin poder capturar señales.

La librería BCM2835 resultó muy útil de manera inicial, pero nos percatamos de que a la hora de configurar la captura de las señales nos daba problemas al no permitir establecer de forma correcta las líneas de SPI.

Viendo que se estaba demorando mucho la captura y que esto podía ser un problema, decidimos que lo más práctico sería hacer nuestra propia librería, aportando así código

[8]La página web www.gnuplot.info contiene información completa sobre la aplicación

abierto a la comunidad[9].

Durante la creación de la librería se puso especial atención a que el consumo de recursos consumidos por su utilización no fuera elevado, siendo este un requerimiento necesario para poder llevar a cabo un sistema de tiempo real. Asimismo se hicieron multitud de test para probar la pérdida de datos durante la conexión con el chip ADS1198, resultando satisfactorios por su baja tasa de errores de muestreo.

Procesado

Una vez comprobado que el filtro Filt Filt funcionaba correctamente y que el Baseline eliminaba de forma correcta la componente continua de la señal, se empezó con el análisis de picos.

Al igual que la parte de filtrado, se requirió de una serie de cálculos previos, teniendo que realizarse un modelo teórico que nos diera unas bases sobre el código que había que generar. Una vez visto que la delineación podía realizarse mediante las derivadas primera y segunda, como se expone en la sección 4.1.5, se procedió a la implementación del código.

Nos enorgullecemos de poder decir que las ideas realizadas son en su gran parte nuestras, estando llevadas a cabo por la necesidad de poder realizar una serie de operaciones complejas con el menor cómputo posible.

Interfaz

En esta iteración se llevaron a cabo una serie de tests enfocados en la representación por pantalla de información. Tras varios prototipos, se llegó a una funcionalidad que utilizaba el Framebuffer de Linux como medio de comunicación directa con la memoria de vídeo, y se establecieron las bases de desarrollo en las que se asentaría el trabajo de interfaz.

Arquitectura

Una vez conseguida la captura de señales, se integró esta funcionalidad al programa, teniendo así las dos entradas de datos funcionando (tanto entrada por fichero como por placa).

[9]Tanto la librería Wiring Pi como la BCM2835 no permiten ver su implementación

4.2.3. Iteración 3: Presentación y puesta a punto

Procesado

La puesta a punto del procesado se realizó cambiando estructuras internas de almacenamiento, sobre todo mejorando su rendimiento.

Es en la ejecución de tests cuando descubrimos problemas de rendimiento a velocidades de muestreo muy elevadas (8KHz), donde comprobamos que el análisis de la señal, junto con la captura en tiempo real, provocan una ralentización del sistema. Considerando inadmisible una captura con unas pérdidas del orden del 5 % y un análisis posterior que no cumplía nuestras exigencias, se opta por eliminar la opción de analizar señales a 8KHz, dejando únicamente la opción de capturar y almacenar a esa frecuencia.

Interfaz

Se llevó a cabo en esta iteración el trabajo correspondiente al diseño y desarrollo del interfaz de conexión entre la aplicación y la librería gráfica que posteriormente se desarrolló.

Con los prototipos y funcionalidades a los que la librería debía dar soporte definidos, se comenzó el desarrollo completo de la librería gráfica.
Cabe destacar el esfuerzo que supone realizar una librería gráfica a bajo nivel desde cero, partiendo simplemente de la escritura en memoria para representar píxeles.
Finalmente la librería gráfica daría soporte a la aplicación para renderizado de la onda electrocardiográfica a tiempo real, así como información extra tanto de las características de la onda como de las estadísticas del procesado.

Arquitectura

La última tarea a realizar en este apartado era permitir la interacción con el usuario, haciéndose uso del terminal de Linux para este fin. Una función de Linux de gran utilidad fue *getopt*, que permite procesar los argumentos y opciones que el usuario puede introducir mediante teclado.

4.2.4. Iteración 4: Memoria y presentación

Documentación

Ya habiendo acabado el grueso del código, se procedió a documentar de manera formal todo el proyecto. Esta memoria es el producto final de esta sección de documentación.

Comunicación

Una vez funcionando todo de manera conjunta, se hicieron pruebas de captura durante periodos largos de tiempo, del orden de horas. Estas últimas comprobaciones sirvieron para demostrar que tanto el temporizador como la captura no consumían un tiempo excesivo de la ejecución, permitiéndose filtrar y analizar la señal en tiempo real, así como su visualización por pantalla.

Procesado

Las pruebas finales de procesado se ejecutaron de manera conjunta con las de comunicación, ya que ambas trabajan de manera conjunta. Era necesario estar completamente seguros de que durante ejecuciones prolongadas no hubiera posibles problemas de rendimiento por la realización de los filtrados y el análisis. Las pruebas resultaron satisfactorias, dejando así esta parte concluida.

Interfaz

Finalmente se realizaron una serie de test de calidad, en los que se sometía a la aplicación al máximo rendimiento posible (pantalla, filtrado y análisis, así como escritura de todos los ficheros de salida activados) y se variaban los fotogramas por segundo de la aplicación y la resolución.

Los resultados indican que hasta unos 180 fotogramas por segundo la carga de trabajo para el procesador es despreciable, siendo esta una frecuencia de refresco muy superior a la necesaria. Dado que la mayoría de monitores tienen una frecuencia de refresco de 60Hz, se estableció así la frecuencia de la aplicación en 60 fotogramas por segundo.

Las pruebas con la resolución demostraron a su vez que aumentando la misma no se obtiene peor rendimiento, sin importar la resolución que sea (llegando incluso a representar 1920x1080 píxeles, puesto que nuestra librería gráfica no redibuja toda la pantalla cada refresco, si no que solo se dibujan los puntos estrictamente necesarios cada fotograma. Por tanto la resolución preestablecida se eligió en base a la máxima compatibilidad posible con los monitores del mercado y no en base a una limitación de rendimiento. Finalmente la resolución fue establecida en 600x400 píxeles, dando soporte así también a monitores de bajas dimensiones, ideales para entornos portátiles.

Arquitectura

Como última comprobación, dejamos probar la aplicación a usuarios que desconocían su funcionamiento, probando así que no hubiera errores de ejecución.

También se mejoró el sistema de ficheros, añadiendo comprobaciones de espacio durante la ejecución, para evitar llenar la memoria no volátil y dejar al sistema sin espacio en disco.

Capítulo 5

Resultados

Esta memoria concluye con la presentación del holter como producto acabado, cuál es su funcionalidad actual y las diferentes medidas de calidad obtenidas para la posible comparación con otras herramientas existentes.

Al tratarse de un dispositivo que ofrece muchas posibilidades de uso, se proponen varias ideas para su mejora y expansión, junto a algunas pautas sobre cómo llevarlas a cabo.

5.1. Producto final

Una vez terminado el desarrollo del proyecto, se plantea el resultado final al que se ha llegado, utilizándose como referencia los objetivos iniciales planteados.

- **Filtrado en tiempo real**: Este aspecto se ha cubierto completamente, los objetivos propuestos han quedado cumplidos. Se hace necesario destacar que los algoritmos de filtrado son propios del proyecto, si bien es cierto que se han basado en ideas ya probadas, su planteamiento y desarrollo han corrido a cuenta nuestra.
- **Calidad de captura y análisis**: La idea inicial era capturar y analizar una señal de 1KHz, pudiendo así hacer un dispositivo a la altura de los disponibles en el mercado. Estamos orgullosos de haber sobrepasado este umbral permitiendo una captura y análisis a tiempo real de señales de 4KHz. Esto un punto muy positivo sobre las expectativas iniciales, dado que se ha cubierto completamente la captura de la señal de marcapasos. Adicionalmente se permite la captura a 8KHz, pudiendo analizarse la señal posteriormente sin problema.
- **Facilidad de adquisición y uso**: Se ha hecho énfasis en que la interacción con el usuario sea sencilla y directa. La entrada de instrucciones se realiza haciendo uso de un terminal Linux, mediante comandos intuitivos y simples. Para que se pueda utilizar en mayor número de lugares, la interfaz de salida por pantalla se puede visualizar en cualquier dispositivo con entrada HDMI o RCA.
- **Bajo coste**: Durante el desarrollo no se ha incurrido en costes adicionales, si no que se ha hecho un esfuerzo por intentar mantener lo más bajo posible el coste total del proyecto.
- **Múltiples entradas de datos**: Se han conseguido implementar las dos entradas de datos previstas, siendo estas por fichero o desde paciente.
- **Diseño modular**: La arquitectura final es completamente modular. Este punto es otro de los pilares del proyecto, permitiendo que el código pueda ser utilizado en más aplicaciones y facilitando la expansión del dispositivo mediante la adición de otros elementos[1].
- **Almacenamiento permanente de datos**: Las capturas y análisis se almacenan a medida que van siendo realizadas. Para no generar ficheros de tamaño excesivo, se ha optado por generar ficheros nuevos de forma periódica[2].

[1] Los dispositivos que se consideran de mayor utilidad se citan en la sección 5.3

[2] Adicionalmente se comprueba que el espacio en disco sea suficiente, avisando al usuario en caso contrario y cerrándose la aplicación de forma controlada

El diagrama 5.1 muestra de manera gráfica el comportamiento de la apliación, mientras que la figura 5.2 ilustra la interfaz que el usuario tiene disponible para visualizar las capturas que se realizan.

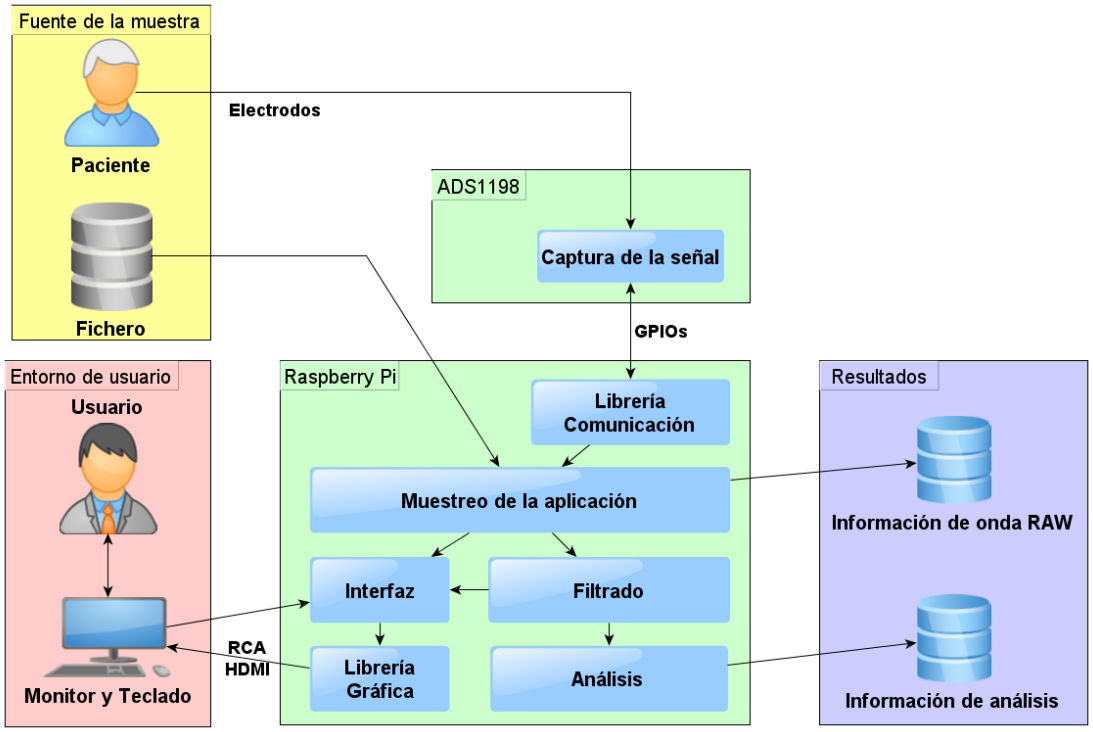


Figura 5.1: Diagrama del producto final

Como se puede ver, se han cumplido y superado las expectativas iniciales. No podemos si no estar orgullosos del resultado obtenido, reflejándose en él el trabajo y esfuerzo de un año de desarrollo.

5.2. Medidas de calidad

La figura 5.3 ilustra las medidas temporales, de media, de los distintos segmentos de la onda electrocardiográfica, puediendo verse que el complejo QRS ocupa en su totalidad en torno a 0.12 s.

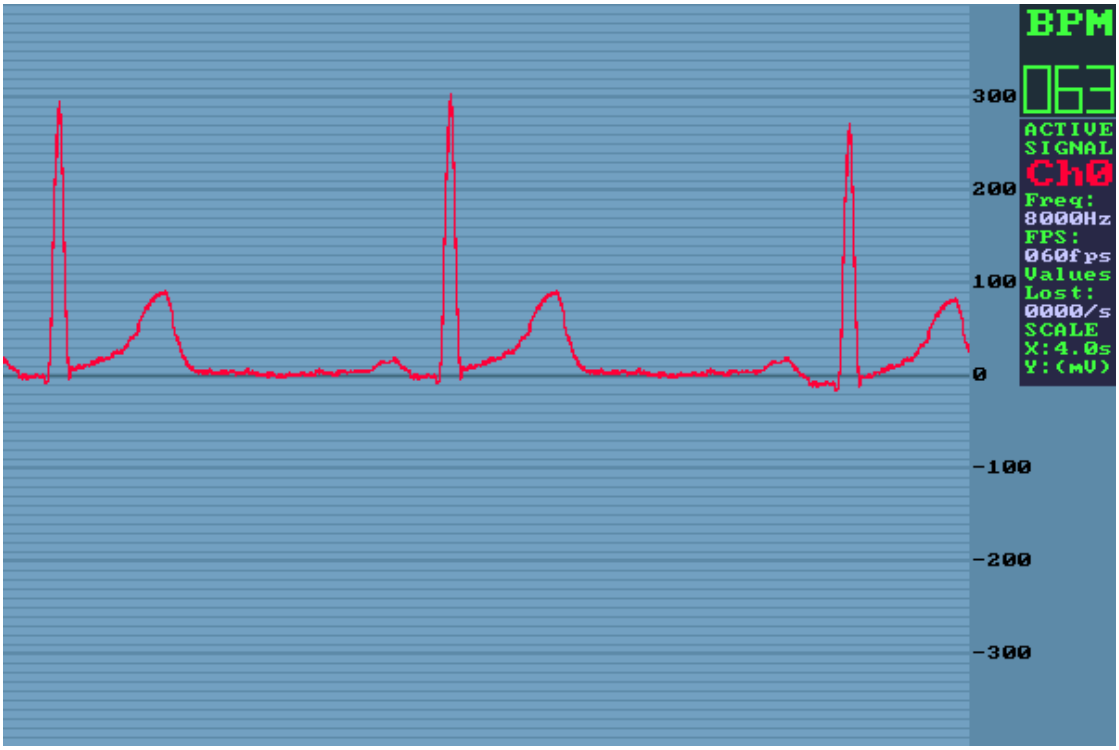


Figura 5.2: Interfaz visual de la apliación

De cara a la obtención de datos, debemos asegurarnos por tanto de no llegar nunca a ese volumen de pérdida, siendo necesario perder menos de un 12 % de las muestras cada segundo. En función de la frecuencia de captura el volumen de datos varía, tal y como se aprecia en la ecuación siguiente.

$$Muestras = 0{,}12 * Frecuencia$$

Aplicando la relación a las frecuencias disponibles obtenemos los valores de la tabla 5.1. En el peor de los casos posibles, con un volumen de pérdida igual o superior a lo que vemos en la tabla 5.1, todos estos valores serían consecutivos, haciendo todo el complejo QRS imposible de detectar.
El dispositivo desarrollado tiene una pérdida de capturas[3] de un orden de magnitud inferior, nunca sobrepasando el 5 % de muestras perdidas en un segundo, lo que da margen más que suficiente para permitir detectar estos complejos.

[3]Las perdidas incluyen tanto los temporizadores perdidos por el sistema (cerca del 1 %) como la captura de muestras erróneas por el chip ADS1198 (en torno al 1 %). A la velocidad máxima de muestreo se pierden un mayor número de temporizadores, de manera puntual puede darse el caso de perder un 2-3 % durante un segundo.

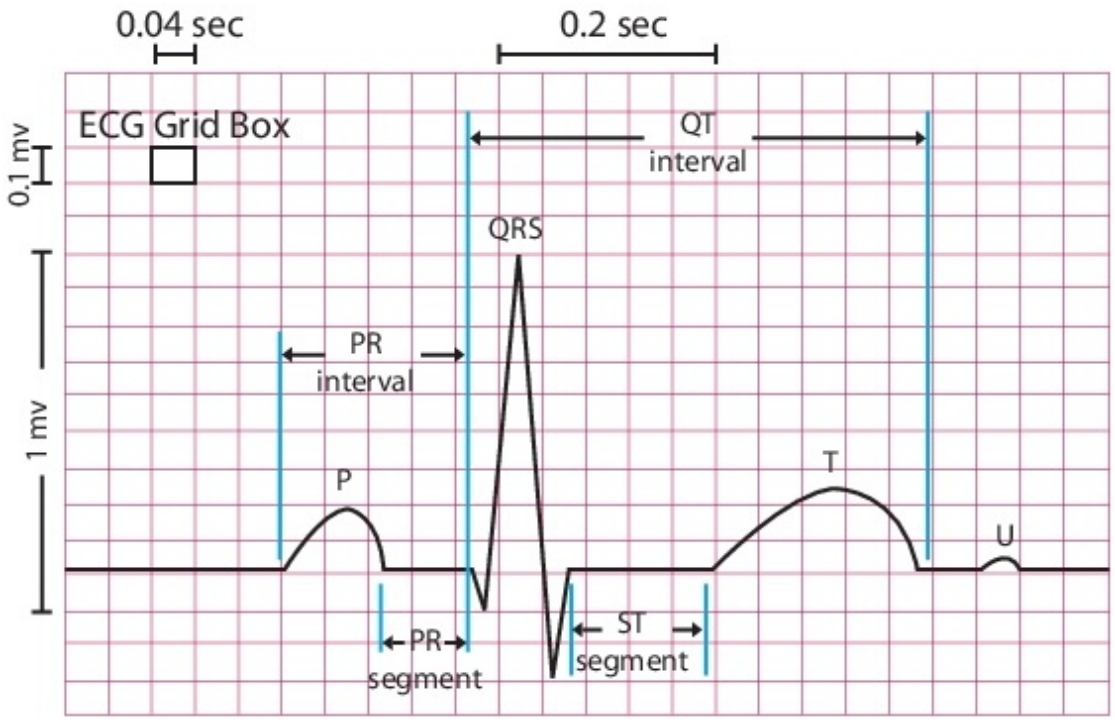


Figura 5.3: Temporización de una onda ECG

5.3. Expansión y uso futuro

Tal como se indica en la descripción de este proyecto, se ha desarrollado de un entorno para dar una primera funcionalidad estable en lo que a captura de datos y representación se refiere, con la posibilidad de adaptarlo de múltiples formas a otros proyectos.

5.3.1. Registro de una unidad de medición inercial (IMU)

Una IMU[4] consiste en un dispositivo que mide la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos.

Frecuentemente en los monitores Holter se integra una IMU que permite registrar y estimar si el paciente está en movimiento, haciendo algún tipo de ejercicio o en reposo para poder contrastarlo con el electrocardiograma que se está registrando, ya que comportamientos inusuales en el ECG pueden corresponderse a una actividad determinada, ofreciendo más información para el análisis.

[4]Inertial Measuremente Unit

Frecuencia de muestreo(Hz)	Pérdida máxima de muestras por segundo
8000	960 $muestras/s$
4000	480 $muestras/s$
2000	240 $muestras/s$
1000	120 $muestras/s$
500	60 $muestras/s$
250	30 $muestras/s$
125	15 $muestras/s$

Cuadro 5.1: Máximo número de muestras perdidas permisibles por segundo

Las IMUs suelen tener protocolos de comunicación I^2C o SPI, caso en el que habría que compaginarlo con la adquisición de datos del chip ADS1198, pero no sería necesarío registrar tantos datos ya que al registrar movimiento no hay tanta variación en el tiempo por lo que los guardaríamos en torno a una frecuencia de 20Hz.

5.3.2. Registro de la pulsioximetría

La Pulsioximetría es un método no invasivo, que permite determinar el porcentaje de saturación de oxígeno de la hemoglobina en sangre de un paciente con ayuda de métodos fotoeléctricos.

Registrar esta información puede alertar de otras patologías ya que al bajar de ciertos porcentajes son síntomas claros de posibles problemas respiratorios y en combinación con el electrocardiograma ofrece un diagnostico más completo.

Para realizar esta técnica, se coloca el pulsioxímetro, en una parte del cuerpo que sea relativamente translúcida y tenga un buen flujo sanguíneo, por ejemplo los dedos de la mano o del pie o el lóbulo de la oreja. El pulsioxímetro emite luces con longitudes de onda, roja e infrarroja que pasan secuencialmente desde un emisor hasta un fotodetector a través del paciente. Se mide la absorbancia de cada longitud de onda causada por la sangre arterial (componente pulsátil), excluyendo sangre venosa, piel, huesos, músculo, grasa. Con estos datos será posible calcular la saturación de oxígeno en sangre.

5.3.3. Miniaturización del dispositivo

Una de las posibilidades contempladas, para la que se proponen esquemáticos en el apéndice B, es la miniaturización del dispositivo utilizando el chip ADS1198 directamente y eliminando los componentes de la placa sobre la que va integrado que no se están utilizando.

De esta manera, se consigue reducir el tamaño aproximadamente al de la Raspberry Pi añadiendo un conector para los electrodos y el PCB[5] sobre el que iría montado.

Se podría disminuir el tamaño más aun con el nuevo producto de la Raspberry Pi, el Compute Module[6] que reduce su area y su altura al eliminar sus conectores y reducirlo a una única placa como un módulo de computación básica, lo que dejaría el area del dispositivo aproximadamente a la mitad.

5.3.4. Detección de picos por reconocimiento de patrones

La detección de picos implementada se basa en el crecimiento/decrecimiento de la señal y en el área que ocupan los picos, por lo que desconocemos la forma que tienen. Además, los picos P de muy baja amplitud no pueden ser recogidos por estos métodos, o al menos no sin dificultad.

Esta información sobre la forma no sólo es relevante para diagnósticos médicos si no que puede permitir la detección de los picos de una manera alternativa que, por su forma o por su amplitud, no son detectados por su crecimiento.

Teniendo una pequeña colección de picos de ejemplos, se podría implementar un reconocimiento basado en patrones que nos indicase sí una región con una señal desconocida incluye un pico e incluso, sí se trata de una cardiopatía o alguna anomalía, podria registrarlo para su posterior consulta.

5.3.5. Detección de cardiopatías

Con toda la información resultante de los diferentes análisis, se puede automatizar la detección de diferentes cardiopatías que vengan indicadas por ejemplo por la ausencia,

[5] Printed Circuit Board - Circuito impreso

[6] Raspberry Pi Compute Module - http://www.raspberrypi.org/raspberry-pi-compute-module-new-product/

el alargamiento o desfase de algún pico.

Para facilitar el tratamiento de una gran cantidad de datos, es conveniente que el dispositivo ofrezca toda la información posible, por lo que reportar un registro de cardiopatías conocidas o sucesos extraños junto al instante de tiempo en el que han ocurrido sería un gran avance para el estudio de enfermedades cardiovasculares.

De manera parecida, se podrían configurar estudios midiendo puntos y formas características de las ondas para que alertase de desviaciones no normales donde pudiese centrarse un cardiólogo sin necesidad de revisar manualmente toda la señal.

Apéndices

Apéndice A

Análisis presupuestario

En el cuadro B.1 se listan los precios de los elementos utilizados[1].

Referencia	Producto	Precio
ADS1198ECGFE-PDK	TEXAS INSTRUMENTS ADS1198ECGFE-PDK - ADS1198, ANALOG FRONT END, ECG EEG	246.43 €
RASPBERRY PI	RASPBERRY PI, MODEL B, 512MB WITH 8GB SD CARD	41.12 €
Coste total		**287.55** €

Cuadro A.1: Precios de los componentes utilizados

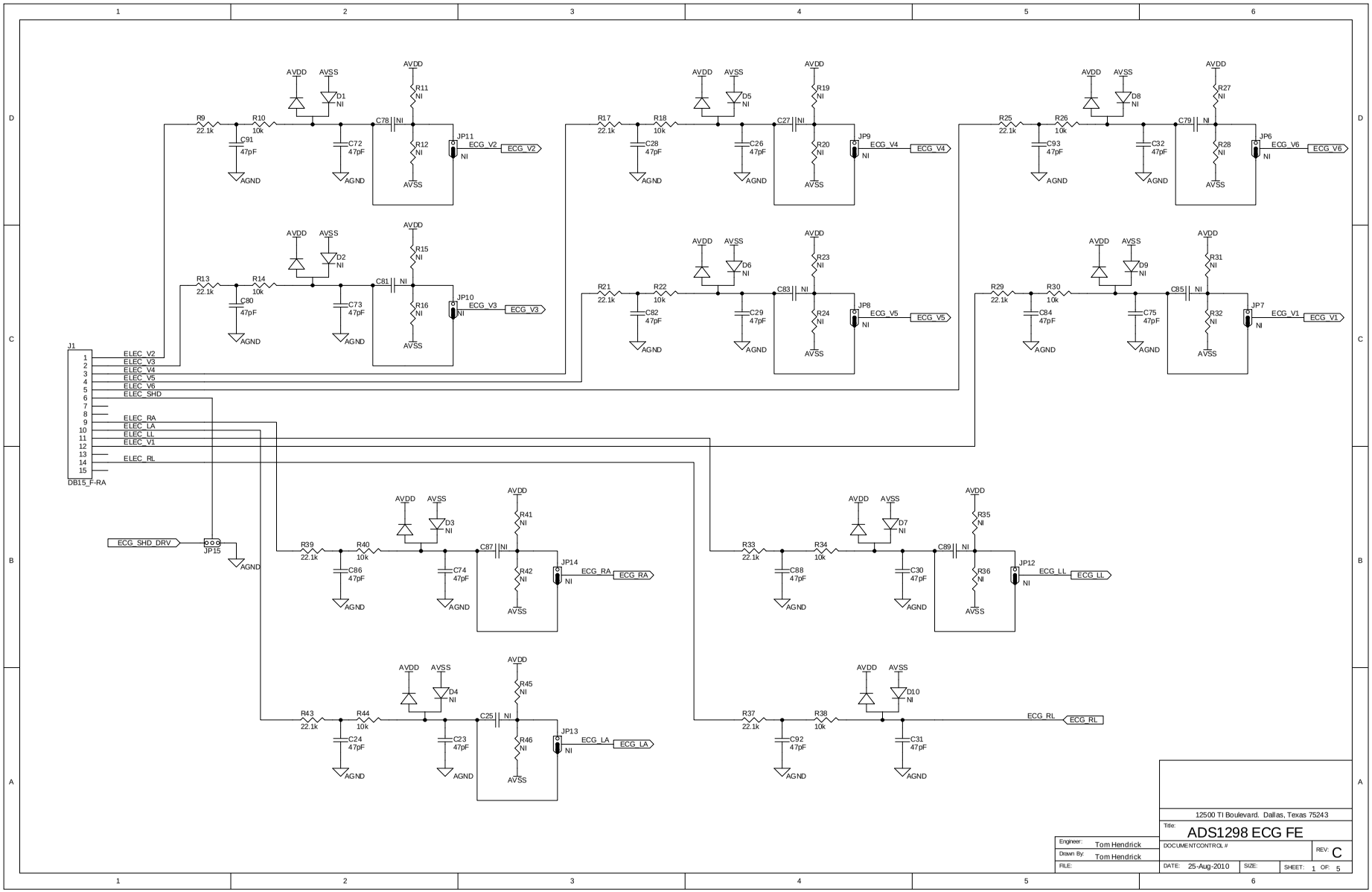
[1]Los precios se listan según la página www.farnell.com a día 17 de junio de 2014

Apéndice B

Esquemáticos reducidos y BOM

Dado que se busca la reducción de los costes de producción, a continuación se adjuntan los esquemáticos reducidos de la placa donde está contenido el chip ADS1198[1]. Se han eliminado de los esquemáticos originales todos aquellos elementos que no son imprescindibles para el correcto funcionamiento del dispositivo desarrollado.

[1]En los esquemáticos dados por Texas Instruments podemos ver que la referencia es para el chip ADS1298, esto no es un error, ambas placas son idénticas, la única variación es el propio chip



Engineer:	Tom Hendrick
Drawn By:	Tom Hendrick
FILE:	

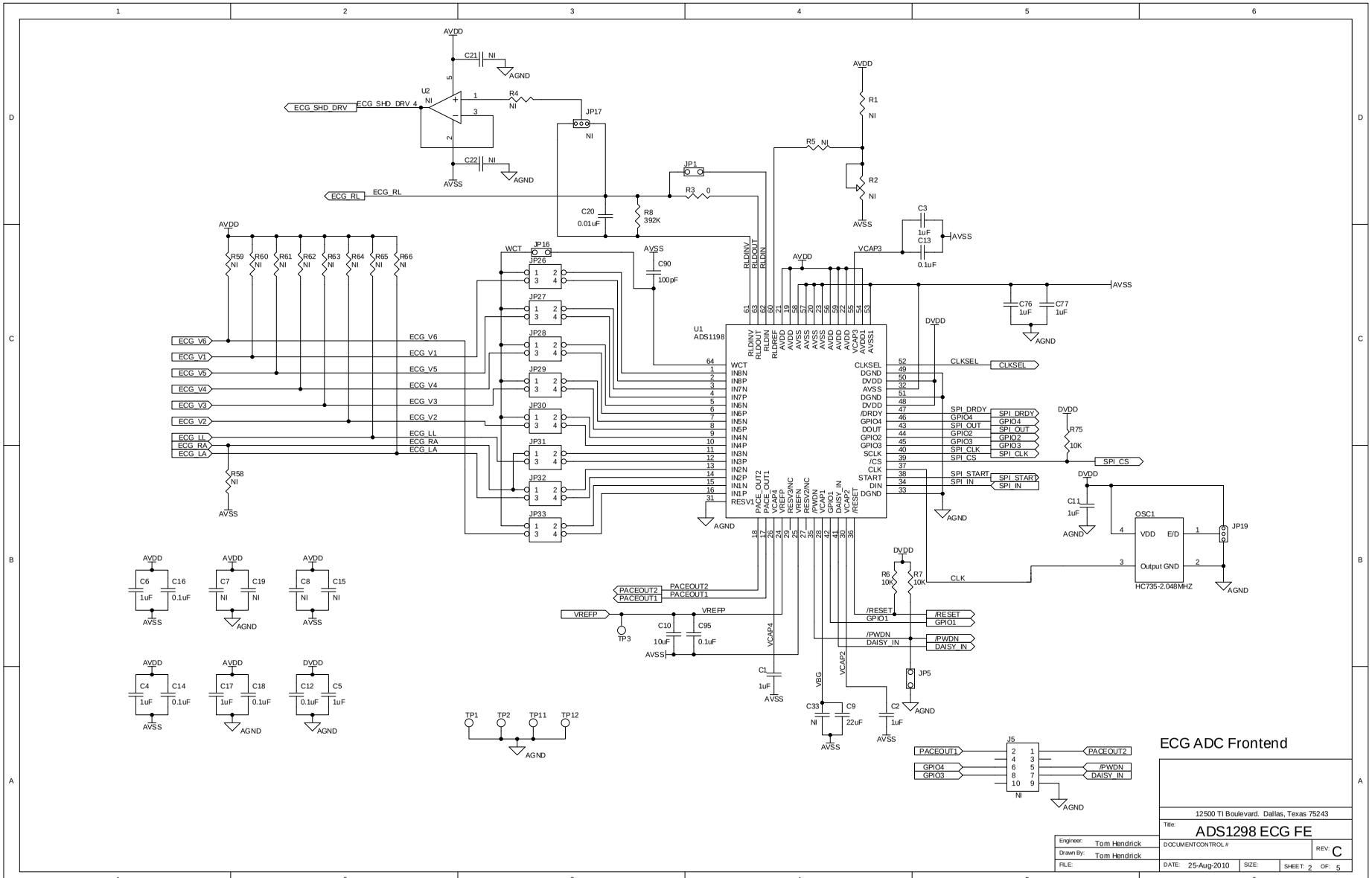
12500 TI Boulevard. Dallas, Texas 75243

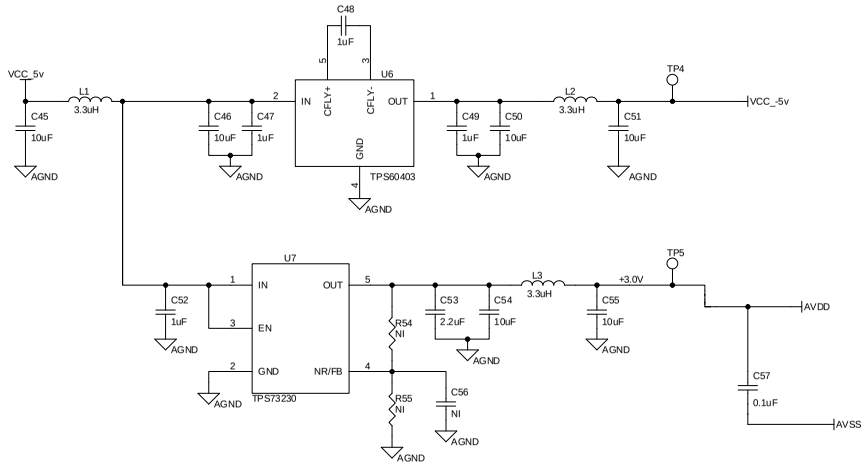
Title: ADS1298 ECG FE

DOCUMENTCONTROL #

REV: C

DATE: 25-Aug-2010 SIZE: SHEET: 1 OF: 5


ECG ADC Frontend
12500 TI Boulevard. Dallas, Texas 75243
Title: ADS1298 ECG FE
Engineer: Tom Hendrick
Drawn By: Tom Hendrick
FILE:
DOCUMENTCONTROL #
REV: C
DATE: 25-Aug-2010
SIZE:
SHEET: 2 OF: 5
U1 ADS1198
U2
OSC1
HC735-2.048MHZ
J5



ECG Power Supplies

Engineer:	Tom Hendrick
Drawn By:	Tom Hendrick
FILE:	

12500 TI Boulevard. Dallas, Texas 75243

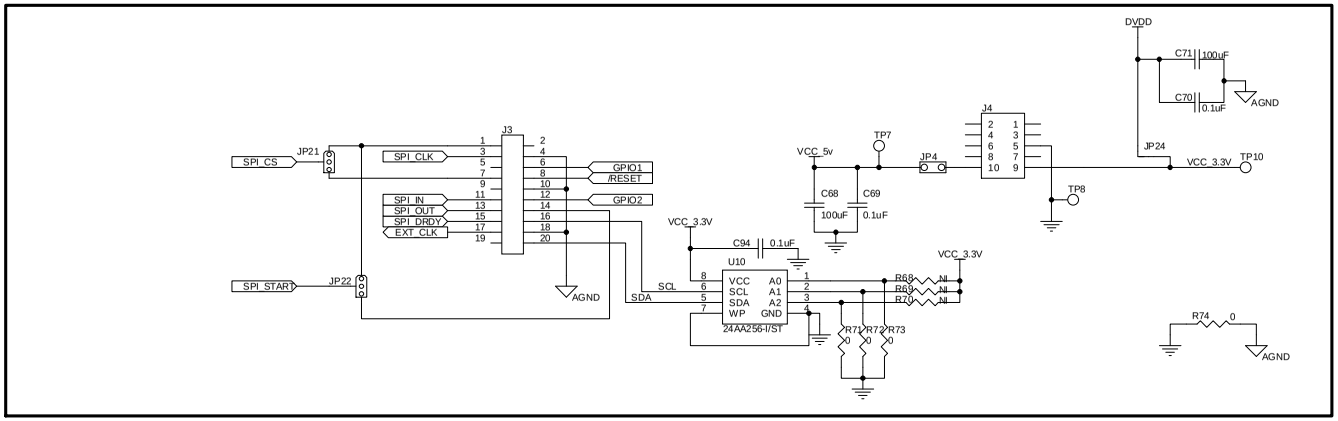
Title: ADS1298 ECG FE

DOCUMENTCONTROL #

REV: C

DATE: 25-Aug-2010 SIZE: SHEET: 4 OF: 5

MDK Interface Connectors



NOTE: J3, J4 female connectors should be populated from the bottom side !

Engineer: Tom Hendrick	12500 TI Boulevard. Dallas, Texas 75243
Drawn By: Tom Hendrick	Title: ADS1298 ECG FE
FILE:	DOCUMENTCONTROL # REV C
	DATE: 25-Aug-2010 SIZE: SHEET: 5 OF: 5

A modo ilustrativo se adjunta el **BOM**[2] con los elementos requeridos para la construcción de la placa[3]. Los precios unitarios son en base a un pedido mínimo de 4000 unidades de cada elemento citado, teniendo en cuenta que muchos de los componentes requeridos no se venden en lotes menores a esta cantidad.

El precio unitario del dispositivo completo sería de 49,69599 $, que al cambio en euros saldría a 36,68 €[4]. Teniendo en cuenta que la placa de desarrollo original de Texas Instruments sale por 246.43 €, la reducción de costes es bastante notable..

[2]El BOM, *Bill of Materials* en inglés, es el listado de los componentes utilizados en la fabricación de un dispositivo

[3]Los precios se han consultado en las páginas www.digikey.com y www.newark.com. La placa de desarrollo *Fusion PCB Service - 2 layers* se encuentra disponible en la página www.seedstudio.com

[4]Según el cambio dolar-euro a día 17 de junio de 2014

Cantidad	Descripción	Precio unitario
15	Capacitor, ceramic 1μF 25V 10 % X5R	0,00844 $
1	Capacitor, ceramic 22μF 6.3V 10 % X5R	0,12800 $
11	Capacitor, ceramic 10μF 10V 10 % X5R	0.08109 $
10	Capacitor, ceramic 0.1μF 50V 10 % X7R	0,00759 $
1	Capacitor, ceramic 10nF 50V 10 % X7R	0,00357 $
20	Capacitor, ceramic 47pF 50V 5 % C0G	0,00552 $
4	Capacitor, ceramic 2.2μF 6.3V 10 % X5R	0,03161 $
2	Capacitor, ceramic 100μF 10V 20 % X5R	0,76500 $
1	Capacitor, ceramic 1000pF 50V 5 % X7R	0,018 $
1	Connector, DSUB Rcpt 15-position R/A PCB SLD	1,66509 $
2	10x2x0.1 female (installed from bottom side)	1,39 $
1	5x2x0.1 female (installed from bottom side)	1,39 $
4	2x2x0.1, 2-pin dual row header	0,187 $
5	Ferrite bead 470Ω	0,035 $
5	Resistor, 0.0Ω 1/10W 5 % SMD	0,004 $
14	Resistor 10.0kΩ 1/10W 1 % SMD	0,005 $
1	Resistor, 392kΩ 1/10W 1 % SMD	0,005 $
10	Resistor, 22.1kΩ 1/10W 1 % SMD	0,004 $
1	Resistor, 47.5kΩ 1/10W 1 % SMD	0,006 $
1	Resistor, 43.2kΩ 1/10W 1 % SMD	0,004 $
1	Resistor, 49.9kΩ 1/10W 1 % SMD	0,008 $
1	Resistor, 46.4kΩ 1/10W 1 % SMD	0,002 $
1	IC ADC 16BIT SPI 8KSPS	29,20 $
1	IC UNREG CHRG PUMP V INV	0,471 $
1	IC LDO REG 250MA 3.0V	0,64 $
1	IC EEPROM 256KBIT 400KHZ	0,74 $
1	OSC 2.0480 MHz 3.3V HCMOS SMT	0,672 $
1	Fusion PCB Service - 2 layers	1,049 $
Precio total de fabricación de una placa		**42,69599 $**

Cuadro B.1: BOM de la placa reducida

Apéndice C

Manual de usuario

C.1. Requisitos previos y conexión

Este programa (de ahora en adelante *ecg*) requiere una serie de configuraciones hardware sencillas para poder funcionar correctamente.

- El programa debe encontrarse en un sistema Raspbian, corriendo sobre Raspberry Pi.
- La Raspberry Pi debe estar conectada a una fuente de alimentación de 5 Voltios que suministre al menos 500mA de corriente al sistema.
- La placa ADS1198 debe estar conectada con la Raspberry Pi mediante los pines de ambos dispositivos siguiendo el esquema de conexión indicado en el cuadro C.1.
- Se deberá conectar el cable de captura en el puerto designado para ello del ADS1198, y los parches se conectarán al paciente (o simulador de pacientes) como indiquen las instrucciones del fabricante.
- Si se desea utilizar salida por pantalla, ésta deberá estar conectada a la Raspberry Pi por puerto digital (HDMI) o analógico (RCA).

C.2. Opciones de ejecución

A la hora de ejecutar el programa *ecg*, debemos hacerlo con permisos de super-usuario. Esto se debe a las operaciones en espacio kernel que se realizan para las tareas de comunicación, captura y mostrado por pantalla.
Para ejecutar el programa *ecg*, suponiendo que nos encontramos en la ruta de instalación, debemos introducir en la consola el siguiente comando:

Raspberry Pi	ADS1198			Uso
	J3	J4	Jumpers	
1		9	JP24 en posición 2-3	3.3V
2		10	JP4 conectado	5.0V
6		5		GND
7	8			RST
13	15			DRDY
18	14		JP22 en posición 2-3	START
19	11			MOSI
21	13			MISO
23	3			SCLK
24	1		JP21 en posición 1-2	CS

Cuadro C.1: Conexión de pines entre Raspberry Pi y ADS1198

```
sudo ./ecg [opcion1 (argumento1)] [opcion2 (argumento2)] ... [opcionN
(argumentoN)]
```

Como vemos, podemos insertar opciones que definirán el modo de ejecución. Las opciones pueden ser una combinación de las siguientes:

- `--file ARG` o `-f ARG`: Opción de lectura desde fichero (en lugar de desde electrodos). Como argumento, ARG debe ser la ruta de un fichero a leer.
  Esta opción permite filtrar/analizar una señal dada por fichero. No es posible mostrar la onda por pantalla al no tratarse esta de una muestra a tiempo real.

- `--frequency ARG` o `-F ARG`: Frecuencia de lectura desde la placa. Como argumento, ARG debe ser concretamente uno de los siguientes selectores, que indican la frecuencia de muestreo del ECG (número de muestras capturadas por segundo).

  - `8000`
  - `4000`
  - `2000`
  - `1000`
  - `500`
  - `250`
  - `125`

- `--channels ARG` o `-c ARG`: Número de canales de los que se captura la señal. Como argumento, ARG debe ser un número entre 1 y 8.
- `--screen` o `-s`: Esta opción habilita la salida por pantalla.
- `--time ARG` o `-t ARG`: Número de segundos de muestreo. Como argumento, ARG debe ser un valor numérico indicando el tiempo de captura, expresado en segundos.
- `--test` o `-T`: Habilita la generación de una onda de pulso cuadrado en la placa ADS1198 para comprobar el funcionamiento de los componentes.
- `--help` o `-h`: Se muestra un menú de ayuda con todas estas opciones descritas.
- `--raw` o `-r`: Se guarda en ficheros la información de la onda capturada. Estos ficheros adquirirán un nombre automático indicando la fecha y hora de captura. Además, la aplicación cierra y cambia de fichero cada 5 minutos para evitar un tamaño excesivo de los mismos o errores de escritura. En caso de no encontrar espacio en la unidad de almacenamiento, se indica este error al usuario y la aplicación termina el procesado.
- `--analysis` o `-a`:Se filtra y analiza la señal, guardándose en ficheros la información con los resultados del análisis. Del mismo modo que en la opción anterior, estos ficheros adquirirán un nombre automático y la aplicación cambiará de fichero cada 5 minutos. También se realizará la comprobación de espacio en disco.
- `--verbose` o `-v`: Se proporciona información adicional durante la ejecución.

Las opciones por defecto de la aplicación, si no introducimos ninguna, son las siguientes:

- Lectura: desde electrodos (placa ADS1198)
- Frecuencia de muestreo: 4000Hz
- Numero de canales: 8
- Tiempo de captura: 1 día
- Salida por pantalla: no
- Guardar en fichero onda capturada: no
- Guardar en fichero análisis: no

C.3. Interfaz de usuario

C.3.1. Pantalla

A parte de la onda electrocardiográfica, la pantalla[Fig C.1] muestra información al usuario acerca de la onda y de la propia ejecución:

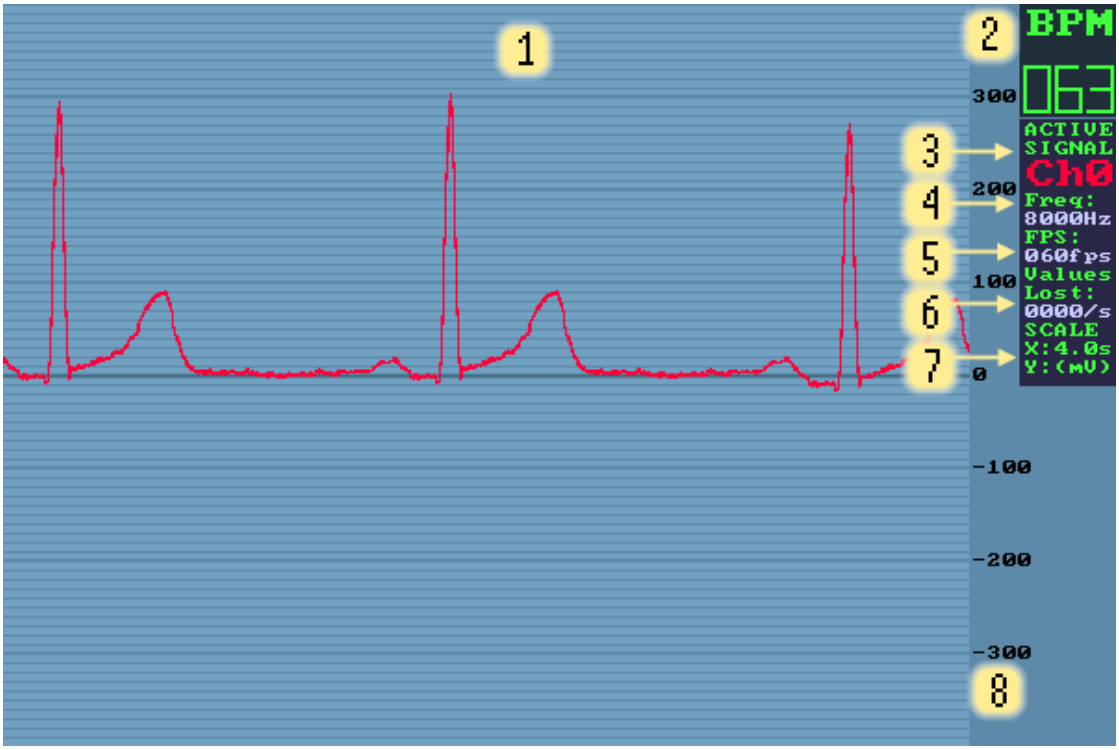


Figura C.1: Pantalla de la aplicación

- 1. Onda ECG sin filtrar/filtrada.
- 2. Frecuencia cardíaca, expresada en pulsos por minuto (BPM, Beats Per Minute).
- 3. Señal activa. Puede variar entre los 8 canales de captura (Ch0..9) y la señal filtrada(Filter).
- 4. Frecuencia de muestreo, expresada en Hz.
- 5. Fotogramas por segundo mostrados por pantalla.
- 6. Valores perdidos en la captura o tratamiento de la señal por segundo.
- 7. Escala de los ejes. Horizontal: tiempo. Vertical: Voltaje.

- 8. Valores en mV del eje vertical.

C.3.2. Teclado

El usuario tiene control sobre la aplicación utilizando el teclado:

- `Ctrl + C`: Cerrar aplicación de inmediato. Como el resto de aplicaciones, este comando de teclado envía una señal al proceso indicando que debe terminar. La aplicación *ecg* captura esta señal y se encarga de finalizar limpiamente todo el procesado, en el mismo instante en que se pulsa, sin haber terminado el tiempo de procesado definido al comienzo de la ejecución.
- `1`: Canal anterior. Selecciona durante la ejecución mostrar por pantalla el canal anterior al actual, ciclando entre los disponibles. [La pantalla debe estar activada].
- `2`: Canal siguiente. Selecciona durante la ejecución mostrar por pantalla el canal siguiente al actual, ciclando entre los disponibles. [La pantalla debe estar activada].
- `3`: Señal filtrada. Si se encuentra disponible en el modo de ejecución actual, esta tecla alterna entre mostrar la onda cruda capturada y la onda filtrada. [La pantalla debe estar activada].
- `4`: Zoom in. Esta tecla permite aumentar el zoom (en el eje del tiempo) sobre la onda mostrada. [La pantalla debe estar activada].
- `5`: Zoom out. Esta tecla permite disminuir el zoom (en el eje del tiempo) sobre la onda mostrada. [La pantalla debe estar activada].

Apéndice D

Fe de erratas

En el manual de usuario del chip ADS1198 hemos enontrado una errata, en la página 43. En dicha página se nos muestra el cuadro D.1 donde podemos ver que la señal de RESETB, correspondiente al pin J3 número 8, está sin negar. Esto es un fallo, ya que esta señal si está negada internamente. El cuadro correcto sería el D.2.

El fallo también se reproduce en la placa donde está contenido el chip ADS1198, habiendo escrito en el pin 8 de J3: **RESET**, cuando debería ser $\overline{\textbf{RESET}}$, en la figura [Fig D.1] podemos ver una reproducción de la placa donde se aprecia el error (esquina inferior derecha).

Signal	J3 Pin Number		Signal
START/$\overline{\text{CS}}$	1	3	CLKSEL
CLK	3	4	GND
NC	5	6	GPIO1
CS	7	8	RESETB
NC	9	10	GND
DIN	11	12	GPIO2
DOUT	13	14	NC/START
DRDYB	15	16	NC
NC	17	18	GND
NC	19	20	NC

Cuadro D.1: Table 12. Serial Interface Pinout (Erróneo)

Signal	J3 Pin Number		Signal
START/$\overline{\text{CS}}$	1	3	CLKSEL
CLK	3	4	GND
NC	5	6	GPIO1
CS	7	8	$\overline{\text{RESETB}}$
NC	9	10	GND
DIN	11	12	GPIO2
DOUT	13	14	NC/START
DRDYB	15	16	NC
NC	17	18	GND
NC	19	20	NC

Cuadro D.2: Table 12. Serial Interface Pinout (Correcto)

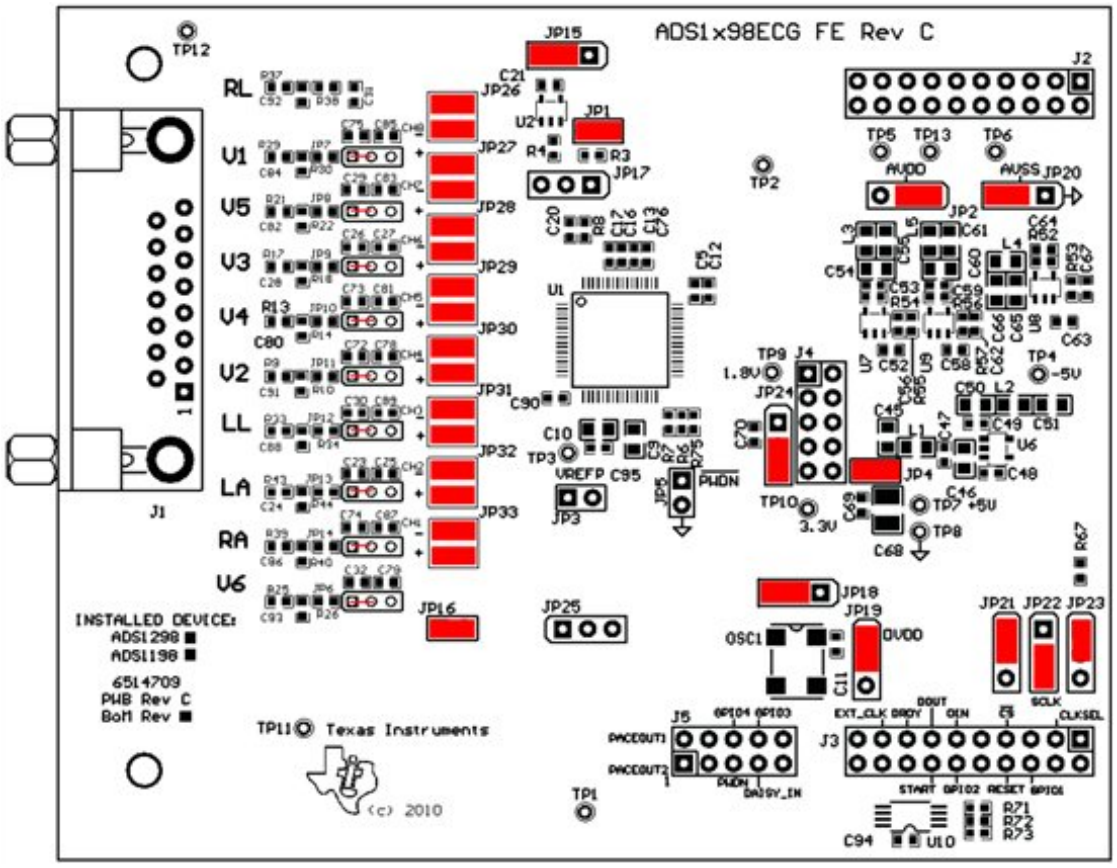


Figura D.1: Placa del chip ADS1198

Apéndice E

Estructuras auxiliares

Para el desarrollo del proyecto hemos preparado una serie de estructuras para aumentar la eficiencia del programa, reservando toda la memoria dinámica al principio de la ejecución para que no se viese afectado el rendimiento del análisis en tiempo real.

E.1. Buffer Circular

La operación que más se repite en el tratamiento de las muestras es el almacenamiento de sus valores según transcurre el tiempo y la eliminación de los datos más antiguos una vez ya han sido utilizados o haya pasado un márgen de tiempo.

Por ello, en las operaciones que normalmente utilizariamos un array, usamos una cola FIFO con implementación de buffer circular, donde cada vez que se lee un dato devuelve el más antiguo que coincide con la posición del primero, y cada vez que se inserta se hace al final.

E.1.1. Modificación: Ventana de máximos y mínimos

Para la aplicación del filtro baseline es necesario calcular los máximos y mínimos en una ventana del buffer circular posicionada al final lo que se traducía en una operación de coste de orden $\Theta(n)$ en el tamaño de la ventana.

Añadiendo información sobre la posición y el valor del máximo y mínimo conseguimos actualizarlos con coste de orden $\Theta(1)$ en el caso de que se encontrasen dentro de la ventana, aunque mantengamos el coste $\Theta(n)$ cada vez que desaparece de la ventana.

La implementación se basa en la actualización de máximos y mínimos cada vez que se introduce un dato para lo que se comprueba si los valores anteriores pertenecen a la ventana, y en caso afirmativo se actualizan.
El otro caso posible es que el máximo o el mínimo ya no perteneciese a la ventana por lo que hay que recalcularlos recorriendo toda la ventana.

E.2. Buffer de datos

El buffer de datos consiste en una cola circular de arrays, de manera que ofrece secciones independientes de la señal, lo que permite bloques de memoria que pueden ser pasados por referencia y evita la necesidad de reservar memoria dinámicamente durante la ejecución.

Este buffer se inicializa con la previsión del número máximo de muestras simultaneas que se calcula que no va a alcanzar la ejecución del programa. Para ello, se reserva un número de bloques determinado de un tamaño fijo.

Al igual que en un buffer circular podemos leer bloques del principio o escribir en el bloque del final, pero con una restricción: la lectura de un bloque sólo se puede llevar a cabo una vez la escritura de ese bloque haya finalizado.

E.2.1. Lectura de bloques

Para leer el siguiente bloque del buffer se usa la función `dm_nextBlockToRead()` que devuelve el primer bloque de la cola siempre que exista y `null` en caso contrario. Una vez finalizada la lectura hay que indicarlo mediante la función `dm_readFinished()` para indicar que se puede descartar ese bloque.

Se puede dar el caso, y se da frecuentemente, en el que se solicitan bloques para la lectura cuando la escritura va por la mitad del bloque y se indica que no existe ningún bloque disponible para leer.

E.2.2. Escritura de bloques

Para escribir existen dos opciones:

- `dm_nextBlockToWrite()`: Pedir un bloque entero e indicar mediante la función `dm_writeFinished()` cuando hemos acabado con la escritura para que marque el bloque como válido y se continue la escritura en el siguiente bloque.
- `dm_writeData(int data)`: Escribirlos dato a dato de manera que cuando se llene un bloque la memoria gestiona automáticamente el cambio al siguiente bloque.

Apéndice F

Raspbian frente a CNC

Aquí se muestran los resultados de la prueba de rendimiento consistente en capturar datos a 8KHz durante 1 minuto. Se contabilizan el número de muestras perdidas debido a señales del sistema que no han sido capturadas por el temporizador.

	Raspbian		**Linux CNC**	
Número de test	**Muestras perdidas**	**Porcentaje de pérdidas**	**Muestras perdidas**	**Porcentaje de pérdidas**
1	1871	1,33 %	304	0,38 %
2	961	1,20 %	56	0,07 %
3	434	0,54 %	144	0,18 %
4	890	0,54 %	106	0,13 %
5	437	1,33 %	304	0,38 %
6	462	0,58 %	95	0,12 %
7	421	0,53 %	79	0,10 %
8	442	0,55 %	76	0,09 %
9	514	0,63 %	164	0,20 %
10	459	0,57 %	118	0,15 %
11	430	0,54 %	69	0,09 %
12	489	0,61 %	104	0,13 %
13	473	0,58 %	131	0,16 %
14	446	0,56 %	93	0,08 %
15	498	0,62 %	66	0,1 %
16	448	0,55 %	84	0,09 %
17	407	0,51 %	69	0,27 %
18	466	0,58 %	218	0,12 %
19	455	0,57 %	94	0,09 %
20	399	0,50 %	74	0,13 %
Media	**570,1**	**0,66**	**122.4**	**0,1455**

Cuadro F.1: Temporizadores no procesados por minuto (Raspbian frente a Linux CNC)

Apéndice G

Glosario de términos

En esta sección se encuentra un glosario con explicaciones o definiciones de algunos de los términos involucrados en el desarrollo de este proyecto.

Acelerómetro Instrumento destinado a medir la aceleración.

Debian Sistema operativo libre desarrollado por una comunidad de usuarios y desarrolladores.

FIFO Estructura de datos donde los elementos entran y salen en el mismo orden.

Front End Parte encargada de recoger entradas de datos y procesarlas para poder tratarlas posteriormente.

Giróscopo Dispositivo para medir, mantener o cambiar la orientación en el espacio de un aparato.

GPIO Entrada salida de propósito general. Es un pin genérico en un chip, cuyo comportamiento puede ser controlado en tiempo de ejecución.

GPU Unidad de procesamiento gráfico. Es un procesador dedicado al procesamiento de gráficos u operaciones de punto flotante.

HDMI Interfaz multimedia de alta definición. Es una norma de vídeo y audio cifrado y sin compresión.

Kernel Software que constituye el núcleo de un sistema operativo.

RAM Memoria de acceso aleatorio. Es allí donde se cargan todas las instrucciones que ejecutan el procesador y otras unidades de cómputo. Se puede leer o escribir en una posición de memoria con un tiempo de espera igual para cualquier posición, no siendo necesario seguir un orden para acceder a la información.

Rasterización Proceso por el cual una imagen se convierte en un conjunto de píxeles para ser mostrados por un medio de salida digital.

RAW El formato RAW contiene la totalidad de los datos de la señal capturada, sin ningún tipo de procesamiento.

RCA Conector común en el mercado audiovisual.

System-on-Chip También llamado SOC, es un circuito integrado con todos sus componentes ubicados en un sólo chip.

USB Estándar que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores, periféricos y dispositivos electrónicos.

Videocore Chip de procesamiento gráfico desarrollado por la empresa Broadcom.

Bibliografía

[1] ARM. *ARM1176JZFS Technical Reference Manual*, 2009.

[2] Broadcom Corporation. *BCM2835 ARM Peripherals*. Broadcom Europe Ltd. 406 Science Park Milton Road Cambridge CB4 0WW, 2012.

[3] Texas Instruments. *Low-Power, 8 channel, 16-Bit Analog Front_end for Biopotential Measurements*, 2011.

[4] Robert Love. *Linux system programming*. Sebastopol (California) : O'Reilly Media, 2013.

[5] Alan Watt. *3D Computer Graphics*. Pearson Education Limited, 2000.

[6] Graham J Williams. *Debian GNU/Linux Desktop Survival Guide*. Togaware.

[7] S. M. Krishnan Y. Sun, K. L. Chan. Ecg signal conditioning by morphological filtering. *Computers in Biology and Medicine*, 32(6):465–479, Nov. 2002.

[8] Djordje Šaponjić Zoran Milivojević. *Programming dsPIC (Digital Signal Controllers) in C*.