UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER

Máster en Electrónica Industrial

Diseño y Verificación de Dispositivo Portable para la Adquisición de ECG Abdominal y Materno

Autor
Álvaro García Ávila

Directores
Almudena Rivadeneyra Torres
Víctor Toral López

Facultad de Ciencias

—

Granada, 8 de julio de 2024

Diseño y Verificación de Dispositivo Portable para la Adquisición de ECG Abdominal y Materno

Autor
Álvaro García Ávila

Directores
Almudena Rivadeneyra Torres
Víctor Toral López

DISEÑO Y VERIFICACIÓN DE DISPOSITIVO PORTABLE PARA LA ADQUISICIÓN DE ECG ABDOMINAL Y MATERNO

Álvaro García Ávila

PALABRAS CLAVE: Electrocardiograma, Electrocardiograma Fetal, PSoC, Fotopletismografía, SpO_2, Presión Sanguínea, Bioseñal, Instrumentación, Diseño Electrónico

RESUMEN

Este Trabajo de Fin de Máster tiene como objetivo el desarrollo de un instrumento portable que permita la adquisición del electrocardiograma, tanto fetal, en el contexto de una mujer embarazada, como de 12 derivaciones para el resto de casos. Dicho dispositivo cuenta también con puertos de uso genérico, los cuales permiten incluir la adquisición de nuevos parámetros biomédicos de manera modular. A modo de demostración de esta funcionalidad, se ha desarrollado la electrónica y el *firmware* necesarios para la adquisición del porcentaje de saturación de oxigeno en sangre (SpO_2). El instrumento cuenta con comunicación Bluetooth de Baja Energía, de forma que puede ser controlado de manera remota por un usuario desde, por ejemplo, un ordenador. Así mismo, esta comunicación permite a este mismo usuario la recepción de los resultados adquiridos para su posterior estudio y almacenamiento. Se ha desarrollado, por tanto, una aplicación en Python para lograr estas funcionalidades.

DESIGN AND VERIFICATION OF A PORTABLE DEVICE FOR THE ACQUISITION OF ABDOMINAL AND MATERNAL ECG

Álvaro García Ávila

KEYWORDS: Electrocardiogram, Fetal Electrocardiogram, PSoC, Photoplethysmography, SpO_2, Blood Pressure, Biosignal, Instrumentation, Electronic Design

ABSTRACT

The objective of this Master's Thesis is the development of a portable instrument that allows the acquisition of the electrocardiogram, both fetal, in the context of a pregnant woman, and 12 leads for the rest of cases. This device also has generic ports, which allow the acquisition of new biomedical parameters in a modular way. As a demonstration of this functionality, the electronics and firmware necessary for the acquisition of the percentage of oxygen saturation in blood (SpO_2) have been developed. The instrument has Bluetooth Low Energy communication, so that it can be remotely controlled by an user from, for example, a computer. Likewise, this communication allows this same user to receive the results acquired for later study and storage. Therefore, a Python application has been developed to achieve these functionalities.

Dña. **Almudena Rivadeneyra Torres**, Profesora del Área de Electrónica del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

D. **Víctor Toral López**, Investigador con Cargo a Proyecto del Departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Diseño y Verificación de Dispositivo Portable para la Adquisición de ECG Abdominal y Materno***, ha sido realizado bajo su supervisión por **Álvaro García Ávila**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de julio de 2024.

Los directores: Almudena Rivadeneyra Torres **Víctor Toral López**

Agradecimientos

Quisiera agradecer a todas aquellas personas sin las cuales no habría sido capaz de sacar este trabajo adelante y que han estado conmigo durante todo este tiempo.

En primer lugar a mi familia, que ha vivido conmigo no sólo la realización de este trabajo si no todas mis etapas estudiantiles, siempre han estado con su cariño y soporte incondicionales.

A mi pareja, que ha sufrido este trabajo como si fuese suyo, siempre apoyándome, mostrando su afecto, pasando momentos inolvidables juntos, y por supuesto leyendo y releyendo mil veces este documento junto a mi hasta encontrar ese sinónimo que teníamos en la punta de la lengua.

A mis tutores Almudena y Víctor por su constante ayuda e interés a lo largo de todo este trabajo y por sus muchos consejos durante todos estos meses.

Y por último a mis compañeros de laboratorio, que aunque siempre ocupados, no dudaron en sacar un hueco para ayudarme o hacer de sujetos de prueba, y sobre todo que hicieron esas largas sesiones en busca de fallos mucho más amenas.

Índice

Índice de Figuras

Índice de Tablas

Índice de Códigos

Capítulo 1

Introducción

1.1. Resumen del proyecto

El objetivo de este Trabajo de Fin de Máster es la creación de un dispositivo portátil que permita la adquisición del electrocardiograma de la mujer embarazada. Por tanto, se busca desarrollar un dispositivo con la suficiente resolución e inmunidad al ruido, como para ser capaz de adquirir tanto la señal del corazón de la madre como del feto, en lo que se conoce como electrocardiograma fetal (FECG), siendo el lograr esta medida el principal objetivo de este trabajo.

A esto hay que añadir que el dispositivo esté configurado de tal manera que se pueda reutilizar también, sin necesidad de añadir ningún tipo de *hardware* complementario, para la adquisición del electrocardiograma de 12 derivaciones. De manera adicional, se busca que el instrumento tenga reprogramabilidad y versatilidad, como para poder ser utilizado de manera simultanea en la adquisición de otros parámetros o biomarcadores que puedan resultar de interés a un profesional sanitario en este contexto. A este fin, el instrumento diseñado contará con puertos de entrada a los que será posible conectar *kits* de expansión que incluyan el *hardware* o la circuitería necesaria (sensorización, acondicionamiento analógico de la señal, etc) para la adquisición de estos otros parámetros de interés.

Una vez tomadas las muestras, se van a transmitir de manera inalámbrica, mediante Bluetooth, a un dispositivo que se encargue del procesado y almacenamiento de las mismas. Dicho dispositivo se puede tratar, por ejemplo, de un teléfono móvil inteligente o PC.

1.2. Objetivos a cumplir

Para lograr el proyecto propuesto en el apartado anterior, se puede dividir el trabajo a realizar en las siguientes tareas:

- Desarrollar un dispositivo portable que realice la adquisición del electrocardiograma (ECG) con la suficiente calidad como para adquirir tanto la señal de la madre como del feto. Dicho dispositivo debe de ser lo suficientemente reconfigurable como para adquirir otros parámetros como, por ejemplo, temperatura o saturación de oxígeno en sangre (SpO_2) y debe de permitir también adquirir un ECG típico de 12 derivaciones.
- Desarrollar el *firmware* necesario para el correcto funcionamiento y control del dispositivo de adquisición cardíaca.

- Crear una aplicación o *script* que permita el control remoto del dispositivo, sea capaz de recibir las muestras adquiridas desde el mismo y, finalmente, las procese (filtrado, etc) y almacene.
- Desarrollar algún *kit* de expansión para la adquisición de un biomarcador adicional. Como, por ejemplo, lo puede ser la temperatura, la presión sanguínea o la saturación de oxígeno en sangre.
- Realizar medidas, en personas que no estén en estado de gestación, para comprobar que los aspectos generales del dispositivo funcionan correctamente. Tras ello, en caso de que sea posible, se harán pruebas también en personas que si que estén embarazadas con la intención de verificar que, efectivamente, es posible recuperar la señal cardíaca del feto.

1.3. Fundamento teórico del ECG

En este apartado se resumen de las bases teóricas de la señal del electrocardiograma. Nótese que todas las técnicas mencionadas en este trabajo, se encuentran en el contexto de ser no invasivas.

El corazón está hecho de tejido cardíaco, que está compuesto por células llamadas cardiomiocitos. Estas células tienen la capacidad de polarizarse y despolarizarse en cada ciclo cardíaco, lo que provoca las contracciones necesarias para bombear la sangre por todo el cuerpo. La actividad eléctrica generada durante este proceso puede ser detectada por sensores colocados en el pecho o las extremidades. El electrocardiograma (ECG) es una representación gráfica de esta actividad eléctrica en función del tiempo obtenida a partir de dichos sensores [1, 10]. La señal cardíaca a menudo se denomina como la señal PQRSTU. En la Figura 1.1 se muestra una representación de esta señal.

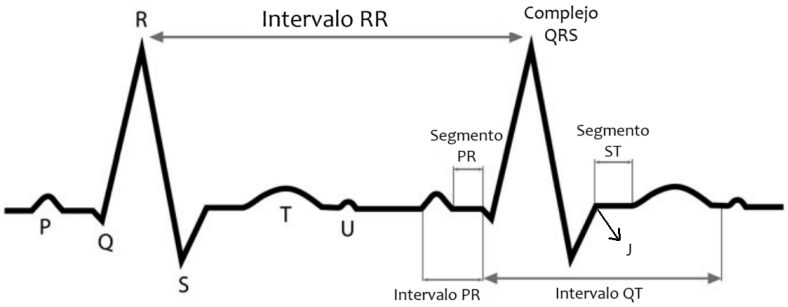


Figura 1.1: Ejemplo de señal cardíaca con los puntos de interés PQRSTU resaltados [1].

1.3.1. El corazón y la señal cardíaca

El corazón está compuesto por cuatro cámaras: las dos superiores son las aurículas y las dos inferiores son los ventrículos. En la Figura 1.2 se muestra una imagen de la anatomía del corazón, donde RA (*Right Atrium*) y RV (*Rigth Ventricle*) son respectivamente la aurícula y el ventrículo derechos. LA (*Left Atrium*) y LV (*Left Ventricle*) son la aurícula y el ventrículo izquierdos [10].

Respecto a la señal cardíaca, se puede interpretar de la siguiente manera [1, 10]:

- La onda P es el resultado de la despolarización de la aurícula.
- El complejo QRS corresponde a la despolarización ventricular y marca el inicio de la contracción ventricular.
- La onda T se produce debido a la repolarización del ventrículo. El intervalo QT se mide desde el principio del complejo QRS hasta el final de la onda T.
- Al final del ciclo cardíaco, se produce la señal U, cuyo origen aún no se comprende completamente, aunque existen varias teorías que sugieren que se debe a un potencial de repolarización.

A partir de la amplitud de estas señales y la duración de ciertos segmentos, se puede diagnosticar una variedad de anomalías. Entre los segmentos de interés se encuentran [1]:

- El intervalo temporal desde los puntos P a Q corresponde a la transmisión de impulsos eléctricos a través del nodo sinoauricular o nodo SA, es decir, el intervalo PQ se refiere al tiempo transcurrido desde la despolarización de la aurícula hasta el comienzo de la despolarización ventricular, que va desde el inicio de P hasta el inicio del complejo QRS.
- El intervalo QT se mide desde el final del complejo QRS, en el punto identificado como J, hasta el final de la onda T.
- El segmento ST se toma desde el final del complejo QRS hasta el comienzo de la onda T.
- El intervalo RR, que es el tiempo desde una onda R hasta la siguiente, representa un ciclo cardíaco y se utiliza para calcular el ritmo cardíaco instantáneo.

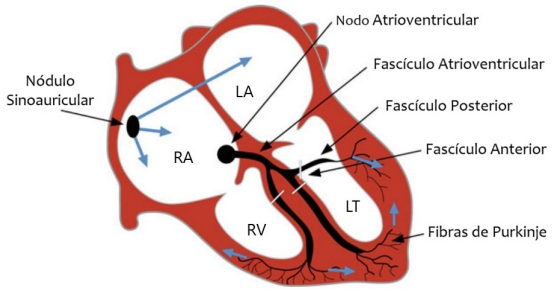


Figura 1.2: Cámaras del corazón [1].

1.3.2. Adquisición de la señal cardíaca

La instrumentación usada para la adquisición de un ECG debe de trabajar a frecuencias bajas, encontrándose la información más relevante de la señal del ECG en el rango de 1 a 30Hz. Por ello, se recomienda utilizar un ancho de banda de paso entre el rango de 0.5 a 150Hz, aunque en la bibliografía se pueden encontrar varios rangos mencionados como adecuados para el filtrado paso banda de la señal cardíaca. Por ejemplo, en [11] se recomienda un filtrado paso banda en el rango de 0.1 a 100Hz o uno paso baja con frecuencia de corte 30Hz entre otros. Por otra parte, para el caso de la población pediátrica, es preferible extender el ancho de banda hasta los 250Hz [12].

Este filtrado es necesario para lidiar con multitud de interferencias provenientes de otros equipos médicos que se pueden encontrar presentes en la inmediaciones del paciente, las propias interferencias generadas por el resto de músculos, el ruido de la alimentación (50 o 60 Hz) si el sistema se encuentra conectado directamente a la red eléctrica en lugar de utilizar baterías, ruido causado por un mal contacto entre la piel y los sensores, etc [1, 10].

La señal del corazón se adquiere de manera diferencial entre dos puntos del cuerpo. De forma habitual, se toman como lugares de adquisición el brazo derecho (RA o *Right Arm*), el izquierdo (LA o *Left Arm*) o la pierna izquierda (LL o *Left Leg*). Estas posiciones se conocen como el Triángulo de Einthoven (TE). En la pierna derecha se añade un electrodo que actúa a modo de referencia para reducir el ruido de la medida. A continuación, se exponen los pares que representan las tres posibles combinaciones de los puntos anteriores para extraer la señal del corazón de manera diferencial [10]:

$$I = V_{LA} - V_{RA}$$
$$II = V_{LL} - V_{RA}$$
$$III = V_{LL} - V_{LA}$$

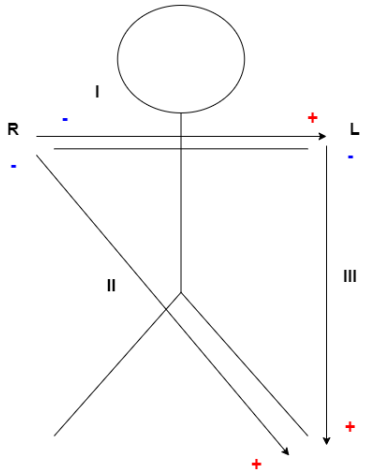


Figura 1.3: Triángulo de Einthoven.

Existen varios métodos para la adquisición del ECG, siendo una de uso muy extendido la conocida como ECG de 12 derivaciones o *12-lead ECG*, el cual mantienen los tres electrodos de las extremidades utilizados en el TE y de manera adicional añade varios puntos de adquisición en las costillas, conocidas como derivaciones precordiales o del plano horizontal. En este tipo de adquisición se representan las 3 señales diferenciales del TE mencionadas anterioremente, las precordiales y las derivaciones aumentadas. Estas últimas se extraen también del triángulo de Einthoven, salvo que en diferentes combinaciones [13]. El conjunto de todas estas medidas, es lo que proporciona las 12 derivaciones distintas que dan su nombre a la técnica.

1.3.3. El FECG

Para la extracción del FECG o electrocardiograma fetal en realidad lo que se realiza es un ECG abdominal (AECG). Para este, se colocan electrodos en la piel de la madre sobre la zona del abdomen. De ellos se extrae la señal cardíaca materna, sobre la cual se encuentra superpuesta la señal del feto [14].

La identificación de la señal fetal no resulta una tarea trivial, principalmente, debido a que la señal materna tiene mucha más amplitud. A lo que hay que sumar otros interferentes, tales como, contracciones musculares o del útero, actividad cerebral del feto y artefactos del movimiento, como lo podría ser la propia respiración de la madre. El principal interés, radica en ser capaces de eliminar todos estos interferentes sin degradar o filtrar por error la señal cardíaca del feto. Por ejemplo en [14], se proponen para el tratamiento de la señal cardíaca obtenida el uso de técnicas de *Discrete Wavelet Transform* (DWT), basadas en detección de umbrales o técnicas de aprendizaje no supervisado para la clasificación de objetos en *clusters*. El objetivo último de estos procedimientos es poder diferenciar de manera automatizada, al menos el complejo QRS, de la madre de el del feto.

En cuanto a la localización y cantidad de electrodos necesaria, no parece existir un consenso o estándar como si lo hay en el caso de los ECG más comunes como el de 12 derivaciones anteriormente mencionado. Algunos autores utilizan 4, 6 u 8 electrodos, mientra que otros llegan incluso a los 32. Además de que el lugar exacto de colocación también varía de autor a autor [2, 15, 16]. En la Figura 1.4 se pueden encontrar algunas propuestas de entre las existentes en la bibliografía.

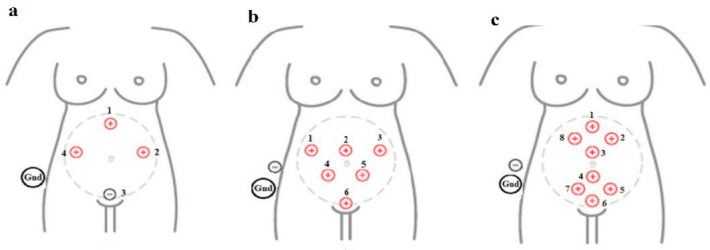


Figura 1.4: Algunos lugares propuestos para la colocación de los electrodos durante el FECG según [2].

1.3.3.1. FECG en este trabajo

Para este trabajo, se utilizan 17 electrodos, de los cuales los 16 primeros se emplean para la adquisición de la señal diferencial del corazón. Estos se configuran en parejas, es decir, se tienen 8 electrodos negativos y otros 8 positivos, estando cada positivo emparejado a un negativo concreto, con lo cual en total se pueden recuperar un máximo de 8 señales cardíacas fetales. El último electrodo se emplea a modo de referencia, es decir, se utiliza para fijar el cuerpo a un nivel de tensión conocido.

1.4. Motivación

La detección temprana de problemas cardíacos en fetos es crucial para gestionar el embarazo y el parto. Las enfermedades cardíacas congénitas o CHD (*Congenital Heart Diseases*) afectan a, aproximadamente, 1 de cada 125 neonatos y dentro del grupo de los afectados, hasta el 30 % resulta mortal [14]. Por tanto, ser capaz de medir la frecuencia cardíaca fetal o FHR (*Fetal Heart Rate*) es esencial para vigilar el embarazo, pudiendo ayudar a la detección de isquemias, arritmias y otras malformaciones del corazón. En este contexto, las técnicas de electrocardiografía fetal puede ayudar a detectar complicaciones antes del nacimiento, permitiendo intervenciones médicas o quirúrgicas rápidas después del parto si fuese necesario [14, 17].

A día de hoy, en España, la mayoría de estudios del FECG se dan en hospitales mediante las popularmente conocidas como "correas". Recibiendo este nombre debido a que se coloca un cinturón con los elementos de sensorización necesarios sobre el abdomen de la embarazada. El problema de estas mediciones, es que se convoca a la persona embarazada de manera ocasional al hospital, en concreto sobre la semana 40, de nuevo a la 41 y, finalmente, cada 2 o 3 días hasta el momento del parto. Estos estudios se realizan durante periodos de medición relativamente cortos, de alrededor de unos 20 minutos. Además hay que tener en cuenta que durante este proceso se precisa de personal sanitario presente y de espacio en las instalaciones del hospital [18].

Se pueden encontrar anunciados algunos dispositivos comerciales que, en principio, permitirían realizar esta monitorización continua y desde el hogar. Algunos ejemplos de estos dispositivos son: FEMOM [19], MERIDIAN M110 [20] o Nuvo [21]. La cuestión es que en la actualidad no es posible adquirir ninguno de estos productos ni existe mucha información al margen de la propia página oficial del respectivo fabricante. Por este motivo, se puede decir que en la práctica, a día de hoy, no existen dispositivos comerciales que faciliten la monitorización de la señal fetal desde el hogar o al menos de manera más cómoda para los involucrados que las "correas" tradicionales.

Teniendo en cuenta lo mencionado, este trabajo se ve motivado por el hecho de desarrollar un dispositivo capaz de cubrir este nicho de aplicación, para el cual en la actualidad no existen alternativas. Se pretende que el dispositivo pueda servir de ayuda a los profesionales médicos, al facilitar monitorización más prolongados al no ser realmente necesario, a no ser de que el personal médico lo considere oportuno, que todas la medidas se deban de realizar en el hospital. De igual manera, se pretende que este instrumento resulte más amigable para las pacientes que los sistemas tradicionales, ayudando con ello a reducir el estrés en el momento delicado que resulta el embarazo.

1.5. Herramientas *software* utilizadas

En este apartado se comentan brevemente los distintos programas o herramientas *software* que se han empleado durante la elaboración de este proyecto. En concreto las principales herramientas utilizadas han sido: KiCad, PSoC Creator, el lenguaje de programación Python, LTspice, Git y GitHub.

1.5.1. KiCad

KiCad es un *software* EDA (*Electronics Design Automation*) gratuito y de código abierto, que incluye el diseño de esquemáticos, distribución de componentes o *layout*, simulación electrónica (basada en SPICE) y visualización en 3D, con la intención de permitir el diseño de un dispositivo electrónico y generar los archivos de fabricación necesarios (*Bill of Materials*, archivos Gerber, etc) para la manufactura de la PCB (*Printed Circuit Board*) correspondiente [22]. En este trabajo, KiCad se ha utiliza en su versión 7.0.

El motivo principal de preferir esta herramienta sobre las muchas alternativas que existen actualmente (Altium Designer, EAGLE, EasyEDA, etc), es el hecho de que proporciona acceso a la totalidad de sus funcionalidades de manera gratuita, a pesar de ser una herramienta bastante potente. Entre otras de sus cualidades a destacar, se mencionan las extensas librerías de componentes que facilita a los diseñadores, documentación y manuales de calidad, lo que acelera en gran medida el aprendizaje de la herramienta y acceso a complementos tanto oficiales, como de terceros. Dichos complementos permiten añadir funcionalidades adicionales al *software*. Por poner un ejemplo, existen fabricantes de PCB que distribuyen de manera oficial sus propios complementos para facilitar la generación de los archivos de fabricación tal y como ellos los necesitan.

1.5.2. LTspice

Para aquellas etapas en las que ha sido necesario la simulación de un circuito analógico, se ha recurrido a la herramienta LTspice, el cual viene a ser un *software* gratuito para simulación de circuitos basado en SPICE (*Simulation Program with Integrated Circuit Emphasis*) [23].

De manera resumida, LTspice permite diseñar el esquema de un circuito electrónico, visualizar las formas de onda y los niveles de voltaje y de corriente en todos los nodos de este. Entre otras cosas, da la posibilidad de simular: barridos de tensión o frecuencia, respuesta en función del tiempo, operación en corriente continua, etc. Las simulaciones se pueden ejecutar usando componentes ideales o añadiendo no idealidades como, por ejemplo, limitaciones en el ancho de banda de un amplificador o los elementos parásitos en un componente pasivo como puede ser una resistencia, un condensador o una bobina. Para facilitar esto, se pueden encontrar librerías distribuidas en línea que ya incluyen el modelado de diversos componentes integrados. De manera oficial el *software* trae preinstaldas librerías con la gran mayoría de componentes de Analog Devices y de Linear Technology, al ser la herramienta propiedad de esta primera.

1.5.3. PSoC Creator

Dado que parte de este proyecto se basa en tecnologías PSoC (*Programable System on Chip*) de Infineon Technologies, para la programación de estos dispositivos se ha utilizado el IDE (*Integrated Design Environment*) oficial distribuido por Infineon, de uso también gratuito, PSoC Creator (en su versión 4.4) [24].

La utilización de este entorno facilita el desarrollo de *firmware* para estos dispositivos, ya que permite seleccionar, interconectar y configurar los varios componentes que se pueden encontrar en el interior de un PSoC de manera gráfica. Es decir, los componentes se añaden a un esquemático, el cual permite asignar las conexiones entre los distintos elementos, así como los pines de entrada y salida del dispositivo, y posteriormente, se puede acceder a cada componente desde dicho esquemático para realizar la configuración inicial

de cada componente de manera gráfica. Tras añadir y preconfigurar los componentes deseados, la herramienta genera las librerías necesarias para interactuar con ellos. Nótese que la programación del resto del código se realiza en lenguaje C.

Una cualidad a destacar de esta herramienta, es la facilidad de acceso de manera inmediata a una extensa documentación. Por dar un caso, con no más que seleccionar uno de los componentes del esquemático, se da al diseñador la posibilidad de acceder a la documentación oficial de dicho elemento. Dicha documentación suele venir acompañada además de uno o varios proyecto para ejemplificar el uso del mismo.

1.5.4. Python

El trabajo relativo al control remoto del dispositivo desde un ordenador, así como el procesado y visualización de los datos transmitidos por este, se ha implementado utilizando el lenguaje de programación Python (3.12). Python es un lenguaje de programación de alto nivel, interpretado, de propósito general y de código abierto, el cual resulta altamente versátil, con lo cual se puede encontrar utilizado en multitud de campos como: ciencia de datos, inteligencia artificial, desarrollo de juegos, automatización de tareas, etc. Este hecho posibilita que existan multitud de librerías de terceros para gran cantidad de finalidades [25].

1.5.5. Git y GitHub

Resulta relevante ser capaces de mantener una copia de seguridad de todos los archivos del proyecto y un control de las versiones del mismo. A este fin, se han usado en conjunto las herramientas Git y GitHub.

Git es un sistema de control de versiones que permite a los desarrolladores rastrear y gestionar los cambios en el código fuente de sus proyectos, así como cualquier otro archivo modificado dentro del mismo. Facilita el trabajo en equipo al permitir que cada miembro tenga una copia completa del historial del proyecto, permitiendo también la creación de ramas para trabajar en diferentes características y fusionarlas *a posteriori* [26].

GitHub, por otro lado, es una plataforma basada en la nube que aloja repositorios Git, proporcionando una interfaz web para trabajar con ellos. Está diseñada para facilitar la colaboración, ofreciendo herramientas como solicitudes para unificar distintas ramas de Git, revisiones de código y seguimiento de problemas. También integra funciones para la gestión de proyectos y la automatización de tareas, convirtiéndose en una herramienta esencial para el desarrollo colaborativo y el manejo eficiente de proyectos [27].

Capítulo 2

Diseño del dispositivo

El objetivo de este capítulo es tratar el desarrollo físico del dispositivo planteado en el Capítulo 1. Con ello nos venimos a referir a la selección de los componentes necesarios como para cumplir los objetivos planteados y como se distribuyen e interconectan estos.

- En el primer apartado, Requerimientos del instrumento, se mencionan las capacidades con las que a de contar el dispositivo, así como las limitaciones físicas impuestas.
- En cuanto a Arquitectura lógica del sistema, se presenta un diagrama conceptual de todos los elementos que forman el dispositivo.
- Un tratamiento en detalle de los componentes electrónicos más relevantes utilizados en el diseño de este instrumento se pueden consultar en: Módulo para control y comunicación, Módulo para muestreo y preprocesado de la señal cardíaca y Sistema de alimentación. Finalmente, se puede encontrar también información sobre otros componentes menores del diseño en Otros componentes. En todos estos apartados al margen de las características de los componentes seleccionados, se trata también el esquemático creado en KiCad para cada uno de ellos. A partir de dichos esquemáticos se creará posteriormente la PCB que implementa el instrumento.
- Respecto de la distribución o *layout* de la PCB que finalmente se ha diseñado, se puede consultar en Diseño y análisis de la PCB.
- Finalmente, en Listado de materiales (BOM) se puede consultar el listado de materiales o BOM (*Bill of Materials*) en el que se recopilan todos y cada uno de los componentes necesarios para la construcción de uno de los dispositivos diseñados en este trabajo.

Al final del capítulo, en la página 75, se puede encontrar el esquemático completo creado mediante KiCad.

2.1. Requerimientos del instrumento

En este apartado se buscan discutir los requerimientos y limitaciones a los que se debe de atener el dispositivo desarrollado, teniendo en cuanta, principalmente, que este se va a tratar de un dispositivo portable, el cual se va a usar para la adquisición de biomarcadores y bioseñales. Por tanto, un lista de los requerimientos mínimos a cumplir son:

- Suficiente resolución y resistencia al ruido, durante la adquisición del ECG, como para que sea posible vislumbrar la señal cardíaca del feto superpuesta a la de la madre. Esta resolución debe de ser de al menos 16 bits, aunque caso de ser posible una mayor resultaría preferente.

- Disponer de al menos 8 canales para la toma de la señal cardíaca. Es decir, tiene que ser capaz de realizar el proceso de adquisición de 8 FECGs de manera simultanea.
- La preamplificación de las muestras debe de ser reprogramable. De dicha manera, es posible realizar un ajuste más fino de la señal a muestrear, que haga un uso eficiente de la resolución completa de los convertidores ADC.
- Funcionar con baterías recargables de ion litio. De ser posible, el dispositivo debe de disponer de una autonomía de al menos 12 horas de operación. Dicha batería se puede recargar desde un conector tipo USB-C.
- Comunicación Bluetooth tanto para la recepción de instrucciones de control desde el lado del usuario, como para el envío de las muestras adquiridas.
- Lector de SD o microSD (*Micro Secure Digital*). Resulta de interés tener la posibilidad de generar un respaldo en una memoria extraíble, de manera que en caso de fallo en la comunicación inalámbrica, no se produzca una pérdida de información.
- Puertos de libre configuración que permitan añadir funcionalidades adicionales según se considere pertinente a lo largo del desarrollo del proyecto, como lo podrían ser, la medida de la presión sanguínea o la temperatura, por poner un ejemplo.
- El peso del dispositivo, con batería incluida, no debe de superar los 200 g. Aunque un peso de alrededor de unos 100 g resulta preferente.

2.2. Arquitectura lógica del sistema

En este apartado se plantea un diagrama de bloques para mostrar de manera abstracta los componentes que forman el instrumento completo. En la Figura 2.1 se puede consultar dicho diagrama. A continuación, se va a hacer una breve descripción de cada uno de los bloques.

El dispositivo principal entorno al cual gira el sistema (bloque MCU) es un *Programable System on Chip* (PSoC) de Infineon, en concreto se utiliza el módulo CYBLE-416045-02 [9]. Dicho elemento dispone de un PSoC 63 y de una antena impresa para permitir la comunicación mediante Bluetooth de Baja Energía [28]. Será el encargo, por ello, de transmitir las muestra vía BLE y de grabarlas también en la tarjeta SD.

El otro dispositivo integrado, que es componente principal para la adquisición de la señal del FECG, es el ADS1299 [3]. De manera resumida, es un empaquetado que permite la adquisición de 8 señales diferenciales, lo que en nuestro caso se traduce en 8 señales cardíacas. Adicionalmente, este módulo ya integra el filtrado y muestreo de cada una de las diferencias sin necesidad de añadir *hardware* adicional. A esto hay que sumar que los parámetros de dicha adquisición se puede configurar, hasta cierto punto, de manera externa. Esto quiere decir que parámetros como la preamplificación de la muestras antes del convertidor, la frecuencia de muestreo, etc son seleccionables de entre una lista de valores prefijada. El integrado está preparado para funcionar en una configuración maestro-esclavo, con lo cual, el PSoC 63 hará la funcionalidad de maestro y realizará solicitudes al esclavo para configurarlo y leer los resultados que este último proporcione.

El módulo CYBLE-416045-02 tiene conexión con dos puertos de libre configuración, los llamados “Puerto Genérico X”. Cada uno de ellos está compuesto por 6 pines macho de 2.54 mm de separación. En principio, estos puertos no tienen ninguna funcionalidad

predefinida. La idea básica, tal y como se comenta en Requerimientos del instrumento, es poder emplearlos para añadir funcionalidades que puedan resultar de interés durante algún caso de uso. La naturaleza flexible y reconfigurable del PSoC 63 resulta en gran medida de ayuda para esta tarea, ya que al poder usarse prácticamente cualquier pin para cualquier funcionalidad, los puertos se pueden emplear tanto como GPIOs (*General Purpose Input/Output*), como para comunicación SPI, UART, etc.

Respecto de la alimentación del instrumento, se realiza desde una única batería de ion litio de 3.7 V. El ADS1299 precisa de dos niveles de tensión diferentes para funcionar, a 3.3 V y 5 V, con lo cual, se incluyen dos sistemas de adaptación de voltaje. La regulación de 3.3 V también se aprovecha para suministrar al CYBLE-416045-02.

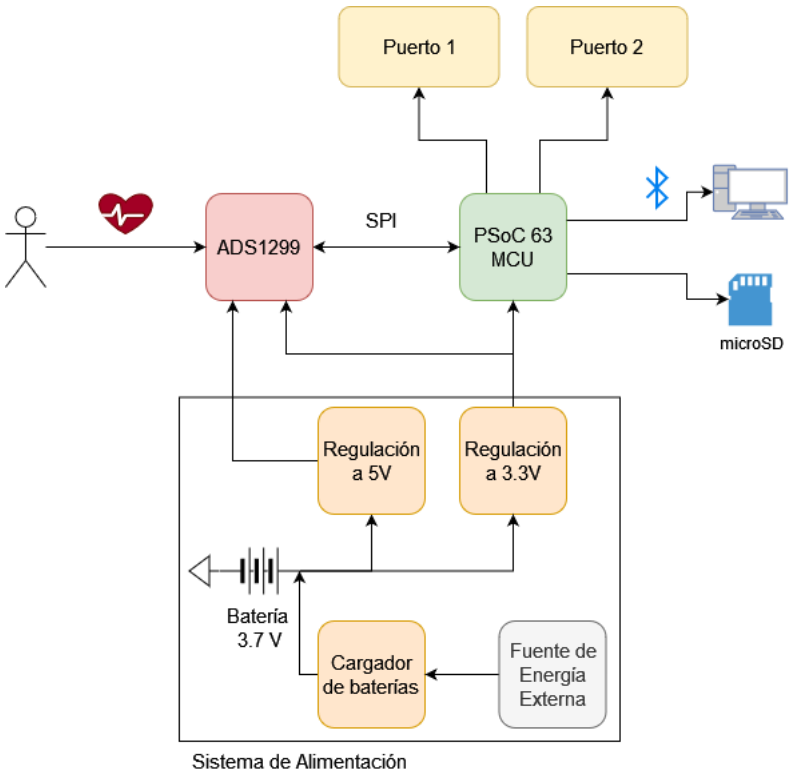


Figura 2.1: Diagrama de bloques conceptual del dispositivo.

De manera general, los pasos seguidos por el instrumento para la adquisición de un intervalo de la señal cardíaca son los siguientes:

1. El dispositivo se mantiene a la espera hasta recibir la instrucción de que inicie la adquisición. Hecho esto, se comprueba si los parámetros de muestreo del ADS1299 se han configurado desde que se encendió el instrumento. Caso de que el integrado aun se encuentre sin configurar, se inicia este proceso. Caso contrario, se puede iniciar la toma de muestras.

2. El PSoC recibe las muestras del ADS1299 y cuando ha almacenado una cierta cantidad de estas, las retransmite mediante Bluetooth antes. Al mismo tiempo, se grabarían estas mismas muestras en la tarjeta microSD. Este proceso se repite hasta que o bien se hayan transmitido una cierta cantidad de muestras o bien hasta que haya transcurrido una cierta cantidad de tiempo.

3. El instrumento vuelve al primer punto.

Aunque se han comentado brevemente algunos de los modelos de los integrado más relevantes utilizados, una descripción más en detalle de los mismos se puede encontrar en los subapartados siguientes.

2.2.1. Listado de componentes principales

En la Tabla 2.1 se ha elaborado un resumen de todos los módulos integrados que se utilizan en el diseño del instrumento desarrollado en este proyecto, de esta manera se puede consultar que dispositivo se ha utilizado para cada uno de los bloques representados en Figura 2.1. Un listado completo de todos y cada uno de los componentes electrónicos necesarios se puede encontrar en apartados posteriores.

Componentes	Modelo	Fabricante	Cantidad
Adquisición + Filtrado ECG	ADS1299	Texas Instruments	1
MCU + Comunicación	CYBLE-416045-02	Infineon	1
Regulador de Votlaje	LP2981A	Texas Instruments	1
Convertidor Boost	TPS61299	Texas Instruments	1
Tarjetero microSD	MEM2075-00-140-01-A	Global Connector Technology (GCT)	1
Puerto USB-C	USB4125-GF-A-0190	Global Connector Technology (GCT)	1
Cargador de Baterías	MAX77757	Analog Devices	1

Tabla 2.1: Listados de integrados y actuadores que forma el instrumento.

2.3. Módulo para control y comunicación

Como ya se ha comentado en el apartado Arquitectura lógica del sistema, el dispositivo encargado de realizar el control del instrumento y transmitir las muestras, va a ser el módulo **CYBLE-416045-02** [9].

Uno de los motivos de seleccionar este modelo, es el hecho de que cuenta con el protocolo de comunicación inalámbrica Bluetooth de manera nativa. Caso de que se utilizase un integrado para el control que no dispusiera de dicho protocolo, sería necesario añadir un componente extra que si que permitiese la comunicación. Esto, a parte de complicar

el sistema al tener que asegurar sincronía entre el controlador y el transmisor, incrementaría también la cantidad de componentes necesarios y con ello el coste y dimensiones del dispositivo final. De manera adicional, y siguiendo con la línea de reducir en la medida de los posible la cantidad de electrónica a añadir, se favorece también el uso de tecnologías PSoC debido a su naturaleza reprogramable y a la gran cantidad de componentes que aglutinan en su interior, lo que facilita dar una flexibilidad considerables a los dispositivos que lo incluyen. Por ejemplo, si desea en un momento concreto utilizar uno de los puertos genéricos que se mostraban en la Figura 2.1 para comunicación, como puede ser UART, se puede hacer con pequeñas modificaciones al *firmware* y sin necesidad de ningún tipo de electrónica adicional.

En el apartado PSoC Creator del Capítulo 1, ya se comentaban también algunas otras ventajas de esta tecnología, como puede ser la extensa documentación existente y la facilidad de trabajo con la herramienta de programación asociada, PSoC Creator.

Aunque existen otros modelos de PSoC que también cuenta con comunicación BLE, nominalmente la familia de integrados CY8C42xx-BL [29], estos utilizan BLE en su versión 4.2. Esta nueva iteración de Bluetooth presenta un incremento en la velocidad de transmisión, alcanzando los 2 Mbps, modos de largo alcance, propagación extendida, lo cual reduce la probabilidad de pérdida de información en entornos con abundancia de interferencias entre algunas otras mejoras [30, 31]. Estas características hacen que el uso de esta versión resulte preferible sobre sus predecesores.

2.3.1. Características CYBLE-416045-02

El módulo CYBLE-416045-02 es una solución, desarrollada por Infineon, basada en el integrado PSoC 63. Incluye la antena impresa, osciladores y componentes pasivos de manera que el PSoC 63 puede operar correctamente y sea capaz de utilizar comunicación inalámbrica Bluetooth de Baja Energía (BLE) 5.0. En la Tabla 2.2 se puede consultar una tabla con las principales características de este integrado. Nótese que esta tabla no pretende ser un compendio de las características completas del integrado, en su lugar solo se han incluido únicamente aquellas que puedan resultar de interés en el contexto de este trabajo.

Respecto de las interfaces de comunicación, se pueden utilizar los protocolos UART (*Universal Asynchronous Receiver / Transmitter*), I^2C (*Inter-Integrated Circuit*) y SPI (*Serial Peripheral Interface*). Para el uso de cualquiera de estos tres protocolos, el dispositivo cuenta con los denominados SCBs (*Serial Communication Blocks*), dichos bloques no son más que conjuntos de pines que se pueden configurar para cualquiera de los tres protocolos. En el caso de que se desee el uso de SPI, se puede controlar un máximo de 4 esclavos por SCB.

Por poner un ejemplo sobre como se pueden utilizar dichos bloques, el protocolo UART necesita de dos señales, una de transmisión (TX) y otra de recepción (RX). Al elegir uno de los SCB, el propio compilador asigna que pines corresponden a cada una de las señales. La tarea de balance del diseñador no es más que seleccionar el SCB que tenga asignado el conjunto de pines que más convengan en el diseño.

Por supuesto, los pines que tengan capacidades SCB, se pueden utilizar también como GPIOs, caso de que no se este utilizando un protocolo de comunicación asociado a dicho conjunto. Es decir, si los pines reservados para el SCB1 no están utilizando para comunicación, se podrían emplear en su lugar a modo de GPIOs.

Característica	Descripción
CPU	150-MHz Arm ® Cortex®-M4F
	100-MHz Cortex-M0+
Núcleos	2
SRAM	288 KB
Flash	1 MB
EEPROM	32 KB
Bluetooth	Si (Versión 5.0)
Comunicación UART	Si
Comunicación SPI	Si
Comunicación I2C	Si
Nº GPIO	36
Nº ADC	1 canal de 12 bits (SAR)
Nº DAC	1 canal de 12 bits
Nº Amplificadores Operacionales	2
Nº Comparadores	2
Nº Bloques Temporizadores / Contadores / PWM	8 de 32 bits 24 de 16 bits
Nº SCB	5
Rango Voltaje Alimentación	1.71 - 3.6 V
Empaquetado	43-pad SMT

Tabla 2.2: Listado de las características de interés del bloque CYBLE-416045-02 [9].

Respecto de los componentes analógicos que resulta de interés en este diseño, se desatacan un conversor analógico-digital (ADC) de 12 bits de resolución y otro conversor, en este caso digital-analógico (DAC), de también 12 bits. El disponer de estos componentes en el interior del dispositivo, facilita la reducción de componentes totales en el diseño total del instrumento. La utilidad de estos dos componentes será, en el caso del ADC la monitorización del estado de carga, también llamado SoC o *State of Charge*, de la batería de litio usada. En el caso del DAC, se puede llegar a utilizar para establecer un voltaje de referencia en el contexto de la adquisición de la señal cardíaca. A este respecto, se puede encontrar más información en el apartado Módulo para muestreo y preprocesado de la señal cardíaca.

2.3.2. Esquemático CYBLE-416045-02

En la Figura 2.2 se puede consultar el diseño esquemático del módulo CYBLE-416045-02. Lo primero que se destaca de dicho diseño es que se ha colocado un oscilador externo de 32.768 kHz para dar sincronización precisa durante operaciones de bajo consumo, tal y como indicado en [9], lo cual ayuda a reducir el consumo energético del integrado. Infineon pone a disposición de los desarrolladores una guía [32] con modelos de osciladores recomendados para operar con sus PSoCs. Se ha escogido el modelo FC-135 32.7680KA-A5 [33], el cual cumple las mismas características que los modelos recomendados. No se ha utilizado directamente ninguno de dichos componentes debido a que, en el momento de la elaboración de este trabajo, estos modelos de oscilador se encontraban fuera de *stock*.

2.3.2.1. Conexión con ADS1299

Respecto del resto de conexiones, el puerto 5 (P5.0-P5.4) se reserva para comunicación SPI. De manera resumida, en el contexto del protocolo SPI, se utilizan dos pines para transmisión de información, MISO Y MOSI, otro para el reloj que sincroniza la comunicación (SCLK) y el resto de pines se emplean en la selección del esclavo. Es decir, SPI tiene un configuración maestro-esclavo en la cual, un mismo maestro, puede ordenar a varios esclavos simultáneamente. Aunque sólo se puede comunicar con uno de ellos al mismo tiempo. Por este motivo, los pines SS0 y SS1 son los utilizados para indicar en cada momento a que esclavo se dirige el maestro. El pin SS0 (P5.3) se utiliza para indicar al módulo ADS1299 que las siguientes instrucciones que el maestro envíe van dirigidas a él. Por otra parte, se reserva también el SS1 (P5.4) para realizar la misma operación pero con la tarjeta microSD.

La última conexión entre el PSoC y el ADS1299, es el pin DRDY (P7.7). Cada vez que el nivel en este pin transiciona de alto a bajo, se está indicando desde el ADS1299 que una nueva conversión está lista [3].

2.3.2.2. Otras conexiones

En cuanto a los puertos de uso general o de libre configuración, de P9.0 a P9.5 se reservan para el primer puerto, mientras que de P10.0 a P10.6, excluyendo a P10.2, se usan para el segundo. El primer puerto se puede utilizar con el SCB2, mientras que el segundo se puede usar con el SCB1. Se han separado ambos bloques de comunicación en dos puertos distintos para que el desarrollo de nuevas a aplicaciones resulte más intuitivo.

Respecto de otros pines, P10.2 se utiliza para monitorizar la tensión de salida de la batería. Conociendo este dato se puede hacer una estimación del estado de carga de la misma. En cuanto a P9.6, se puede usar para generar un nivel de tensión que se utilice como referencia en ciertas adquisiciones del ADS1299. Por otro lado, P7.2 se reserva para un LED que indique el encendido de la placa. En cuanto a XRES, SWDIO y SWDCLK son los pines de la interfaz de programación SWD.

Adicionalmente, se incluyen 2 componentes, un LED y un pulsador, que en un principio no tienen utilidad predefinida. La idea es poder utilizar estos componentes en tareas de depuración. Eventualmente en una aplicación final, se les pueden asignar tareas según resulte más conveniente al desarrollador/usuario. El pulsador, en configuración de *Pull-Down*, se coloca en P0.4, mientras que el LED de depuración se sitúa en P7.1.

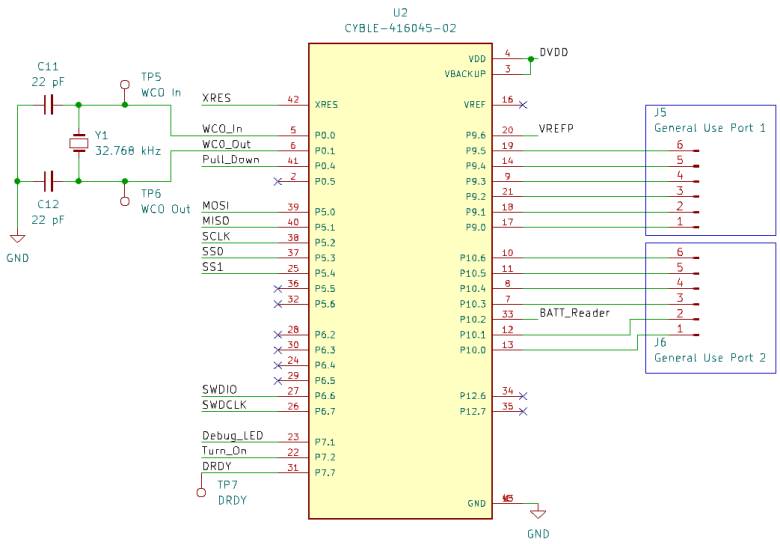


Figura 2.2: Diseño esquemático de las conexiones del módulo para control y comunicación CYBLE-416045-02.

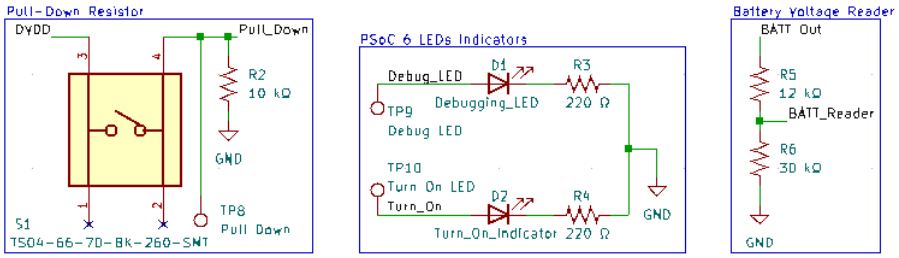


Figura 2.3: Componentes adicionales conectados al módulo CYBLE-416045-02.

2.4. Módulo para muestreo y preprocesado de la señal cardíaca

Se precisa de electrónica capaz de realizar la adaptación y muestreo de las señales cardíacas. En este proyecto, se ha escogido el SoC (*System on Chip*) **ADS1299** [3] de Texas Instruments el cual ya aglutina todas estas tareas. El dispositivo cuenta con 8 canales, pudiendo emplearse cada uno de ellos para una señal diferencial. Cada canal dispone a su vez de un ADC delta-sigma de 24 bits y un amplificador de ganancia programable o PGA. El rango de muestreo del dispositivo alcanza los 16 kHz, valor más que suficiente para la adquisición de la señal cardíaca debido a que el máximo ancho de banda a considerar en esta señal se encuentra entre los 150 y 250 Hz, este último valor para la realización de medidas pediátricas [12, 34]. En la Figura 2.4 se puede consultar el diagrama de bloques de este dispositivo.

Este dispositivo esta pensado para operar en una configuración maestro-esclavo, asumiendo el papel del esclavo. Para que el maestro pueda escribir en la configuración y para que pueda leer los resultados, el integrado dispone de una interfaz SPI.

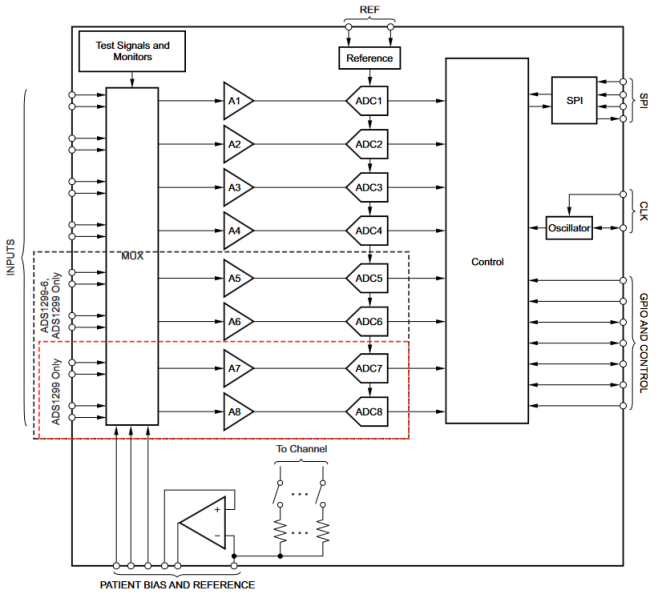


Figura 2.4: Diagrama de bloques ADS1299 [3].

2.4.1. Esquemático ADS1299

En la Figura 2.5 se puede consultar el diseño esquemático de ADS1299. En este apartado se detalla la configuración para la adquisición con este dispositivo.

El primer puerto de entrada (J1) que da acceso al integrado, está formado por las entradas BIASOUT y SRB1. BIASOUT se utiliza para poder dar una referencia externa al sistema desde un electrodo durante la adquisición del ECG. En la Figura 2.6, se puede observar como dicha señal permite, en esencia, dar una tensión de referencia al sistema, es decir, se introduce una cierta tensión sobre la piel del paciente para establecer todo el sistema a la misma tensión de referencia.

La segunda entrada existente en el puerto J1, es SRB1. La cual se puede utilizar para dar una referencia común únicamente a la parte negativa de la señal diferencial (INxN).

Esto es lo que permite que también sea compatible utilizar este instrumento en la adquisición del ECG de 12 derivaciones. En la Figura 2.7, se puede consultar un ejemplo de como quedaría la conexión interna del dispositivo caso de que se decida usar este pin para la entrada de la parte negativa de la señal diferencial. El hecho de que se desee usar esta configuración se le debe de indicar al circuito desde el maestro accediendo a los registros de configuración.

Se disponen también de pines para ajustar la referencia positiva y negativa de los ADCs de manera externa, es decir, los límites inferior y superior de estos. Para ello, se pueden utilizar los pines VREFN y VREFP para respectivamente, el límite inferior y superior. VREFN se cortocircuita a tierra, de manera que el límite inferior sea siempre 0 V. Por otra parte, se da la posibilidad de ajustar el límite superior si así se desea. Si se genera esta referencia de manera interna, la cual es la configuración por defecto, se genera este límite respecto de AVSS, lo que da un límite de 4.5 V. Sin embargo, también se puede dar este valor de tensión desde el PSoC. Para ello, se ha colocado una resistencia de 0 Ω llamada "RVref", la cual hace la conexión con un GPIO del PSoC que permite suministrar el valor de tensión deseado.

El motivo de hacer la conexión empleando un resistor de 0 Ω es que, si se decide que únicamente se desea emplear la referencia interna, se puede substraer el resistor para que no se introduzca una tensión desde el PSoC por accidente.

Por otra parte, el ADS1299 cuenta con 4 pines GPIO. Dado que no se les ha dado ningún uso, se cortocircuitan a tierra tal y como se indica en [3].

Para acabar, en la Figura 2.5 se puede ver también que el ADS1299 tiene dos entradas de alimentación AVDD y DVDD. La primera de ellas se emplea para suministra a la parte analógica del dispositivo a 5 V, mientras que la segunda suministra a la digital a 3.3 V.

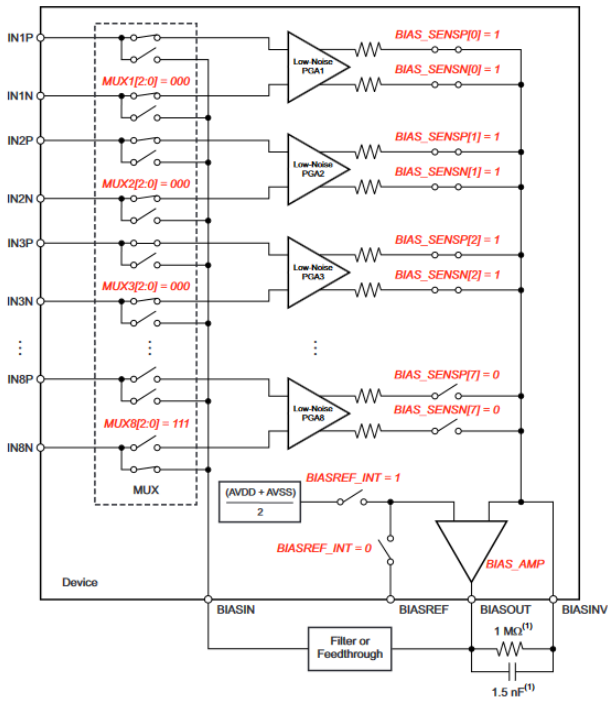


Figura 2.6: Conexiones y configuración del multiplexor del ADS1299 [3].

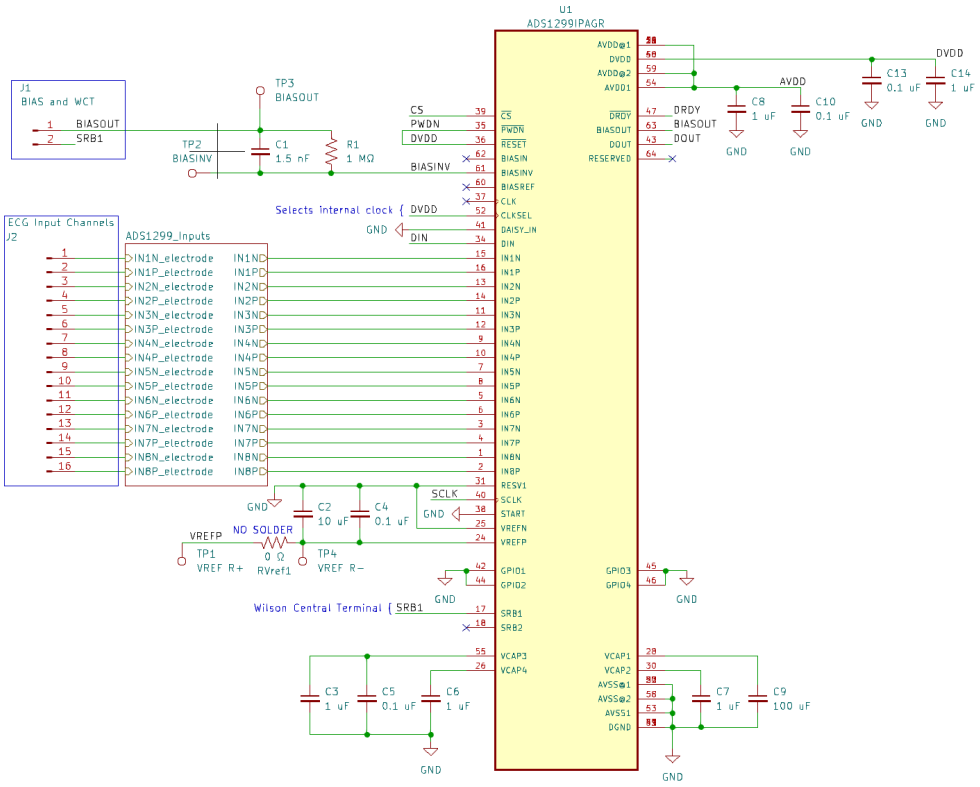


Figura 2.5: Diseño esquemático de las conexiones del módulo para adquisición de señales diferenciales ADS1299.

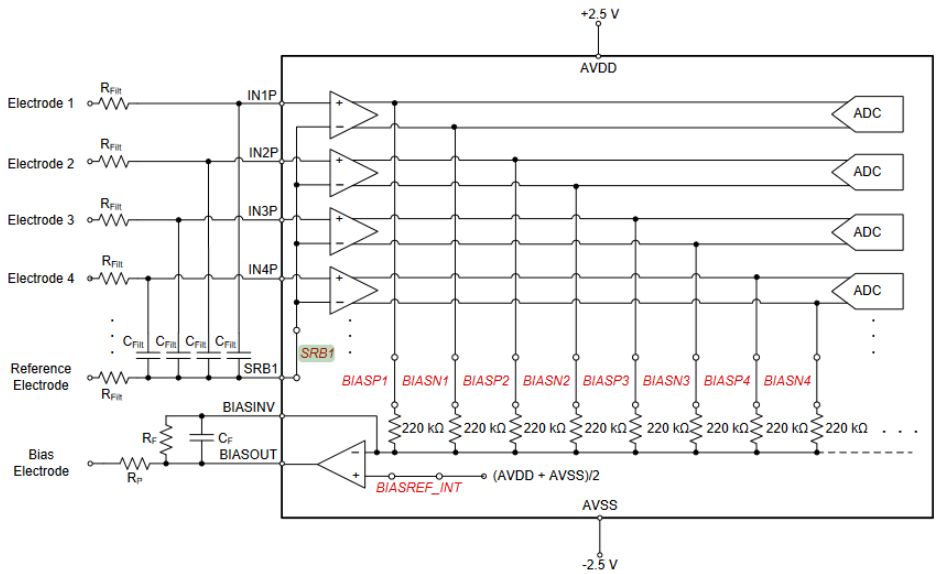


Figura 2.7: Diagrama de conexiones del multiplexor del ADS1299, caso de que la parte negativa de la señal diferencial se introduzca desde el pin SRB1 [3].

2.4.2. Filtrado paso baja diferencial y desacoplo del paciente

Para evitar el efecto del *aliasing*, se incluyen filtros pasivos paso baja diferenciales a la entrada del ADS1299. El *aliasing* es un fenómeno que ocurre cuando la frecuencia de muestreo de una señal no es lo suficientemente alta como para capturar adecuadamente las frecuencias presentes en la señal original, esto hace que al tratar de reconstruir la señal digitalizada, no se logre recuperar la original [35]. Para evitar este efecto, se realiza un filtrado paso baja de manera que se trate de eliminar las componentes de mayor frecuencia de la señal, limitándonos por tanto, al muestreo de únicamente un rango de frecuencia de interés. Este FPB (Filtro Paso Baja) es lo que se conoce como un filtro antialiasing.

Dado que se está trabajando con señales diferenciales, se ha diseñado un FPB pasivo de primer orden. En la Figura 2.8 se puede consultar el filtro planteado, formado por un condensador y dos resistencia para cada canal diferencial. Por lo tanto, se ha replicado esta estructura en 8 ocasiones.

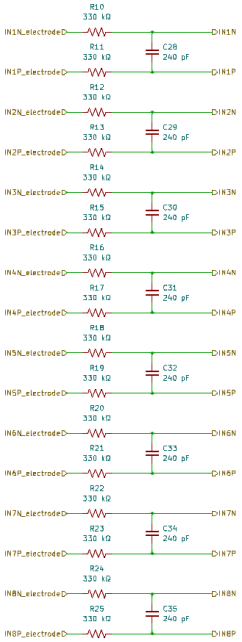


Figura 2.8: Diseño de un filtro paso baja diferencial para un total de 8 señales diferenciales.

Se establece una frecuencia de corte de 1 kHz, para lo cual se utiliza la Ecuación 2.1 mostrada en [36]. El rango de frecuencias de interés en el que se encuentra la señal cardíaca llega hasta los 250 Hz, con lo cual se debe de establecer la frecuencia de corte (f_c) por encima de este valor. Debido a que la señal fetal es muy pequeña, resulta de gran relevancia no atenuar componentes frecuenciales que se localicen dentro de este intervalo. Para ello, se sitúa f_c en 1 kHz, con lo cual se logra que los componentes dentro del rango de los 250 Hz mantengan ganancia unitaria, es decir, que no se vean atenuados.

$$f_c = \frac{1}{4\pi RC} \tag{2.1}$$

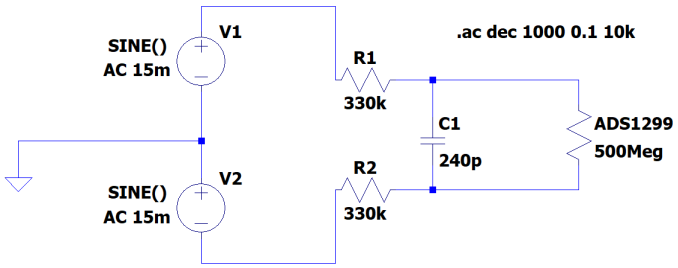
Previamente a que se seleccionen valores para el condensador y la resistencia, hay que tener en cuenta que estos pasivos tienen una segunda funcionalidad al margen de actuar como filtro antialiasing. En los sistemas médicos se debe de desacoplar al paciente de la

alimentación del dispositivo. Una manera de hacer esto, es situar una resistencia de gran valor entre el paciente y el instrumento, de manera que actúe de protección para evitar una descarga eléctrica. Las resistencias que forman parte del FPB pueden en este caso desempeñar este papel.

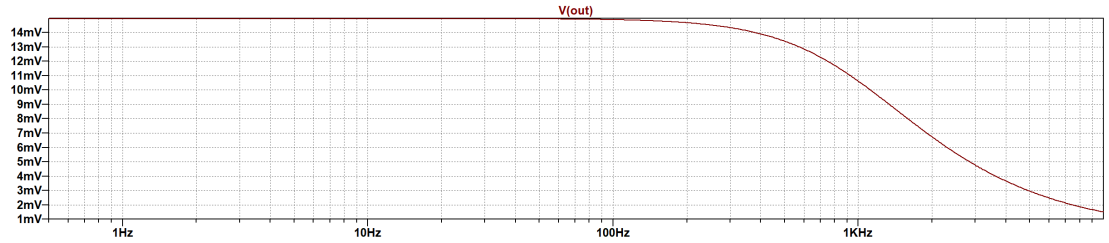
Se selecciona por tanto un valor de resistencia de 330 kΩ. Despejando de la Ecuación 2.1, se resuelve que se necesita un condensador de, aproximadamente 241 pF.

2.4.2.1. Simulación FPB diferencial

Para comprobar las características del filtro, se realiza una simulación del mismo utilizando el *software* LTspice. Se puede comprobar en la Figura 2.9 que se ha utilizado una resistencia de 500 MΩ para representar al ADS1299. Esto se debe a que como se indica [3], esta es la menor impedancia de entrada posible de los canales. Adicionalmente, dado que los 241 pF no son un valor comercial habitual para un condensador, se ha utilizado en su lugar uno de 240 pF.



(a) Esquema.



(b) Simulación.

Figura 2.9: Esquema y simulación de LTspice del FPB pasivo diferencial.

2.5. Sistema de alimentación

En esta sección se discute la electrónica necesaria para lograr la alimentación del instrumento. Esto abarca la selección de una batería para el suministro de energía, el sistema de carga de dicha batería y las adaptación analógica necesaria previa a la alimentación de los componentes que forman el instrumento.

2.5.1. Características de la batería de ion-litio a utilizar

Tal y como se mencionaba en Requerimientos del instrumento, uno de los requisitos a cumplir es que el dispositivo se alimente mediante baterías, dado que no deja de ser un instrumento portable. Respecto de la tensión de salida de la misma, se plantea que tenga un voltaje nominal de 3.7 V y una tensión de salida máxima de 4.2 V. Dado que

resulta deseable que el dispositivo tenga la máxima autonomía posible, la batería debe de disponer de la máxima capacidad de carga posible, alcanzando un compromiso entre el tamaño, peso y coste y la capacidad.

En la Tabla 2.3 se puede consultar una breve comparativa entre algunos modelos considerados. Finalmente, se ha optado por el modelo **VARTA 56457 201 016** [37] como posible candidata para la realización de pruebas con el dispositivo al considerarse esta un buen compromiso.

Modelo	Capacidad	Voltaje Nominal	Max Voltaje Carga
VARTA 56457 201 016	1000 mAh	3.7 V	4.2 V
JAUCH LP502030JH	260 mAh	3.7 V	4.2 V
VARTA 56456 302 012	2400 mAh	3.7 V	4.2 V
PANASONIC PICPAL36	1300 mAh	3.7 V	4.2 V
USE-702528-500PCBJST	500 mAh	3.7 V	4.2 V

Tabla 2.3: Comparativa de modelos de baterías de ion-litio

2.5.2. Cargador de baterías

Si se incluye una batería de litio, es lógico por tanto incluir un sistema de carga. El modelo de cargador integrado elegido ha sido el MAX77757 [38] de Analog Devices y se ha elegido teniendo en cuenta los requerimientos de la batería comentados en Características de la batería de ion-litio a utilizar.

Este dispositivo permite la carga de baterías de ion-litio de una celda, con un rango de entrada de 4.5-13.7 V y una corriente de entrada máxima de 3 A, con lo cual se da una cierta flexibilidad a la hora de tener una fuente alimentación desde la que cargar el sistema. Para realizar la carga desde, por dar un caso, la fuente estándar de un teléfono móvil, se ha añadido un cargador tipo USB-C. El cargador favorece este tipo de conexiones debido a que cuenta con un detector automático de conexiones USB-C, lo cual le permite configurar la carga de manera dinámica según el modelo de cargador identificado. En la Figura 2.10 se puede consultar un caso de aplicación típica para este cargador, el diseño realizado en este trabajo se ha basado en gran medida en esta imagen.

2.5.2.1. Esquemático MAX77757

Se pasa ahora a tratar el diseño esquemático realizado para este proyecto, el cual se puede consultar en la Figura 2.11. El integrado permite limitar la máxima corriente de carga a partir de una resistencia de configuración, la cual hemos denominado en el esquemático como “R_IFAST1”. Se elegido un valor para esta resistencia de 2.4 Ω lo cual, según el fabricante, limita la máxima corriente de carga a 0.5 A. El motivo de escoger este valor, es que aunque el utilizar una menor corriente de carga implica que la batería tarde más en precisamente cargarse, si que ayuda por otra parte a alargar la vida útil de la misma debido a que los tiempos rápidos de carga a mucha corriente favorecen el envejecimiento de las baterías [39].

En la Figura 2.11 se puede observar también que se han incluido dos LEDs para dar información al usuario sobre la carga. El primero de ellos, D3, indica si la batería se está cargando, mientras que D4 indica si la tensión de entrada al integrado es la adecuada para realizar la carga. Ambos LEDs son de color verde.

Nótese que el diseño se ha realizado de tal manera que es posible utilizar el instrumento a la par que la batería se está cargando. En principio, esto únicamente debería de implicar que la batería recibirá una corriente menor en situación de carga y uso simultáneo, con lo cual se cargará más lentamente.

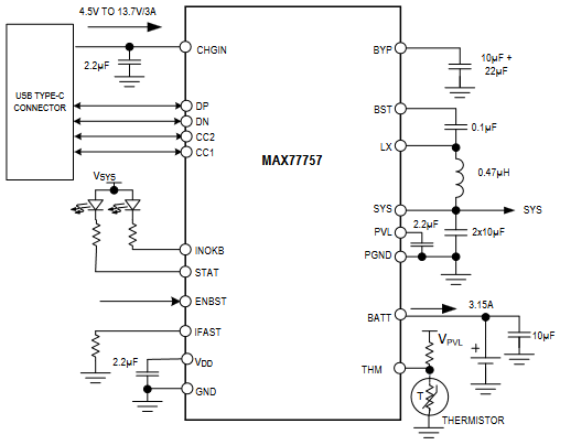


Figura 2.10: Diagrama de uso típico del cargador MAX77757.

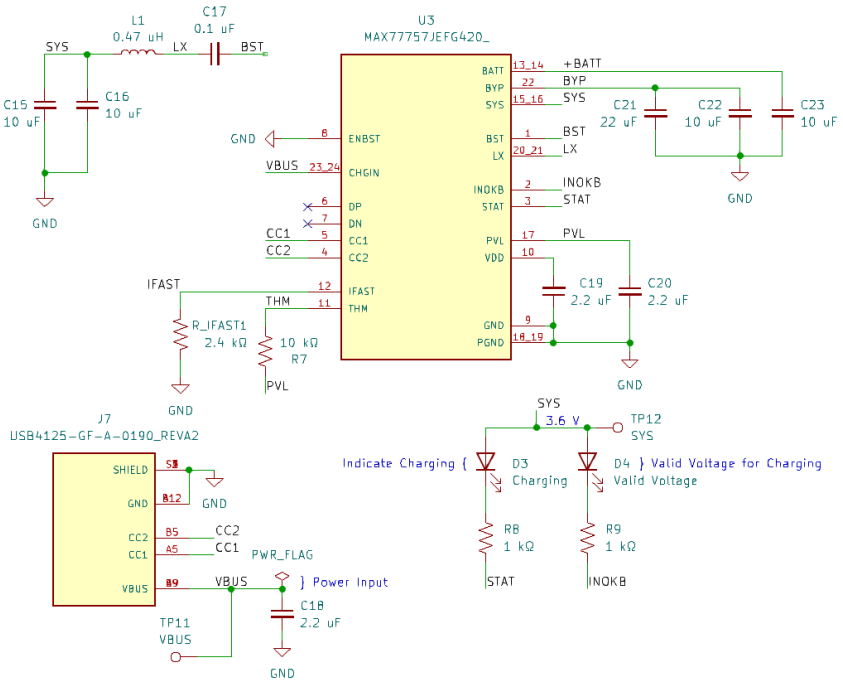


Figura 2.11: Esquemático cargador de baterías y de la entrada de potencia externa al sistema mediante conector tipo USB-C.

2.5.3. Regulación de tensión

Tal y como se ha comentado en el apartado Características de la batería de ion-litio a utilizar, se plantea utilizar una batería que de manera nominal proporcione 3.7V. Como se trató en Módulo para muestreo y preprocesado de la señal cardíaca, el ADS1299 precisa de dos voltajes de alimentación diferentes, mientras que el CYBLE-416045-02 únicamente requiere de uno. Para la parte analógica del ADS1299 se debe de regular la tensión hasta los 5V, mientras que para su parte digital, el CYBLE-416045-02 y el resto de componentes del instrumento se van a alimentar a 3.3V. En la Figura 2.12 se puede consultar el esquemático de los dos reguladores empleados.

Para la conversión a 5V, se ha utilizado el convertidor *boost* TPS61299 [40]. En cuanto a la regulación a 3.3V, se selecciona el LDO (*Low Dropout Regulators*) LP2981 [41]. En ambos casos se colocan condensadores a la entrada y salida de los reguladores. Estos condensadores se conocen como de *bypass* y su funcionalidad es doble. Por un lado, dado que los condensadores se "oponen" a cambios bruscos de voltaje, el primero de sus usos es suprimir fluctuaciones de la fuente de alimentación debido a cambios repentinos de corriente. De esta forma, si se produce una reducción en el nivel de tensión, el condensador se descarga para compensar dicho desnivel y si se produce un incremento, el condensador se carga.

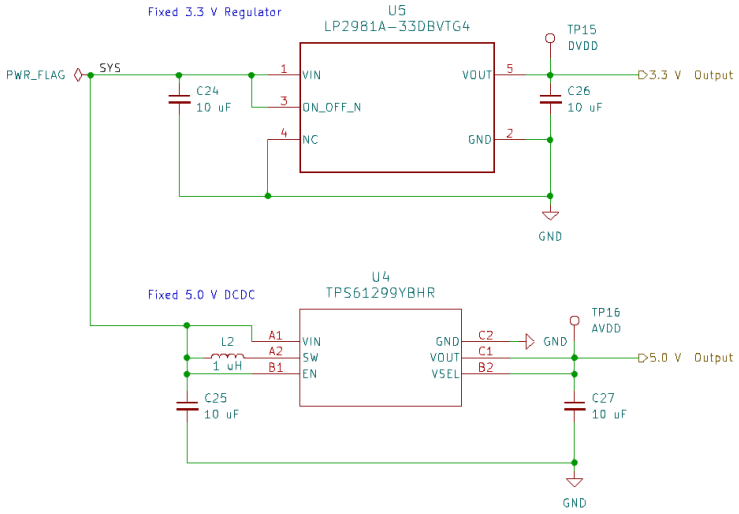


Figura 2.12: Esquemático de los reguladores para obtener tensiones de 3.3V y de 5V.

2.6. Otros componentes

2.6.1. Interruptor de encendido/apagado

Como resulta lógico, se debe de incluir un accionador que permita apagar el dispositivo. Para ello, se utiliza un interruptor de dos posiciones y de montaje en ángulo recto para abrir o cerrar el circuito. En concreto se ha utilizado el modelo JS102011JAQN [42], cuyo esquemático se puede consultar en Figura 2.13. Nótese que para el caso de que se quiera apagar el dispositivo (posición 2-3), se ha colocado una resistencia de 1kΩ. El motivo es evitar un posible cortocircuito al descargarse la corriente acumulada en los condensadores de *bypass* en el momento de apagar el dispositivo, disipándose en su lugar la potencia en esta resistencia.

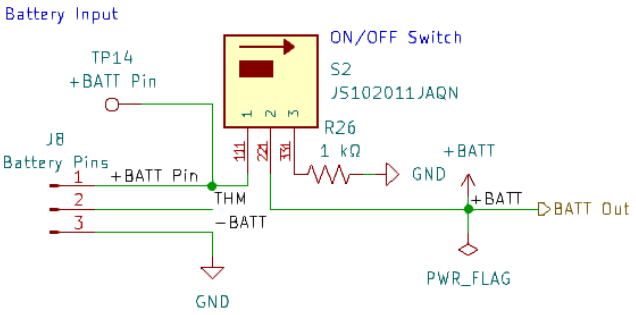


Figura 2.13: Esquemático del interruptor para apagado/encendido del dispositivo.

2.6.2. Tarjetero microSD

Se incluye un tarjetero microSD para, como se comento en Requerimientos del instrumento. se pueda crear un respaldo de las muestras que se vayan adquiriendo. En la Figura 2.14 se puede consultar el esquemático del tarjetero usado, el modelo es el MEM2075-00-140-01-A [43].

La lectura/escritura en la tarjeta microSD se realiza mediante protocolo SPI, por este motivo se pueden ver los pines SCLK, MOSI, MISO y SS1 que ya se mencionaban en la sección Esquemático CYBLE-416045-02. La escritura en la tarjeta se realiza desde el CYBLE-416045-02. Nósete que además este utiliza el mismo puerto SPI que el empleado para el control del ADS1299, teniendo asignado este último el pin SS0 y la microSD el SS1.

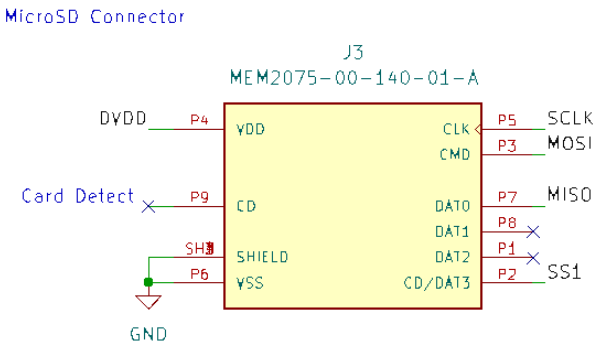


Figura 2.14: Esquemático del tarjetero microSD.

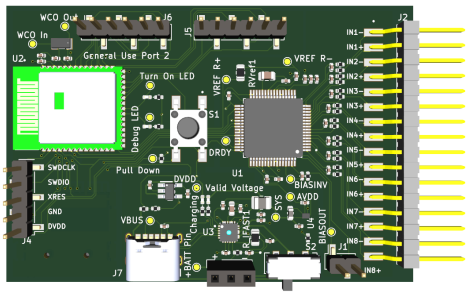
2.7. Diseño y análisis de la PCB

Vistos en los apartados anteriores los componentes a utilizar y los esquemáticos individuales de cada uno, es decir, los componentes pasivos que penden de ellos y las conexiones (o ausencia de ellas) de cada uno de sus pines, se procede ahora al diseño físico de una PCB que aglutine de la manera más eficiente posible todos estos componentes y conexiones sobre un espacio físico.

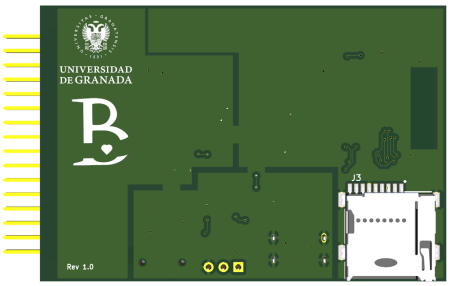
Para este proyecto se va a utilizar una PCB de doble cara. Esto quiere decir que la PCB tiene cobre en las caras inferior y superior, pudiendo usarse por lo tanto ambas para la soldadura de componentes y la colocación de pistas, existiendo una capa de material aislante entre las dos. Si se desean conectar eléctricamente dos puntos que se haya en caras opuestas de la placa se emplean vías. Estas consisten en perforaciones que conectan ambos lados de la PCB mediante un camino con cobre [44].

El uso de PCBs de dos caras resulta preferible en este caso sobre otras más complejas como las de 4 capas, en las cuales se tienen dos capas de cobre internas adicionales para la colocación de vías y pistas. Esto en potencia puede llegar a mejorar la densidad de componentes eléctricos, aunque a cambio de un mayor coste. En el contexto de este proyecto, una PCB de 2 caras resulta adecuada para lograr unas dimensiones lo bastante reducidas como para poder utilizar el dispositivo sin complicaciones.

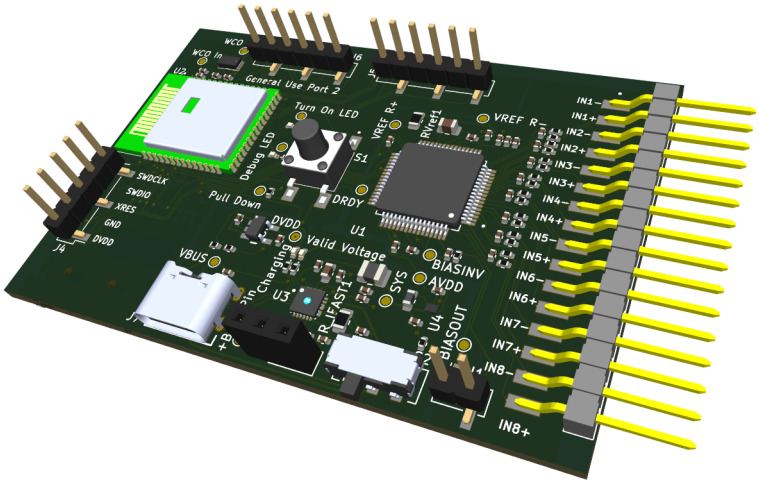
En la Figura 2.15 se puede consultar un modelo 3D del diseño final que se ha alcanzado para la placa, donde se muestran las caras inferior y superior de esta. En la sección a continuación, se trata en primer lugar las buenas prácticas y recomendaciones seguidas para lograr mejor diseño posible de la PCB. Tras ello, en la sección Mapa de componentes, se tratan en mayor detalle donde se ha situado cada uno de los componentes vistos en función de las recomendaciones mencionadas anteriormente.



(a) Vista superior.



(b) Vista inferior.



(c) Vista de perfil

Figura 2.15: Modelo 3D de la PCB que forma el dispositivo visto desde varios ángulos.

2.7.1. Buenas prácticas de diseño seguidas

2.7.1.1. Planos de tierra

Dado que en este sistema se utilizan tanto dispositivos analógicos como digitales, resulta recomendable separar los planos de tierra de ambos. Para ello, tal y como se puede observar en la Figura 2.16, se han creado hasta 3 planos de tierra conectados por "puentes". A la izquierda se ha situado el plano de referencia para los componentes digitales, a la derecha el de los analógicos y en la parte inferior se ha colocado un tercero para los elementos de potencia, es decir, el cargador y los reguladores de tensión. El motivo de hacer esta división es que si se conectan componentes analógicos a la tierra digital, a la señales analógicas del plano se les sumará ruido digital y viceversa [45]. Hay que tener en cuenta que aunque se hayan aislado en la medida de lo posible los 3 planos, se tienen que establecer conexiones entre ellos en la forma de "puentes" para que todo el sistema se encuentre a la misma referencia.

Es importante destacar que en este trabajo no se ha podido hacer una división perfecta entre componentes analógicos y digitales precisamente debido al uso de tecnologías SoCs. En concreto debido al empleo del CYBLE-416045-02 y del ADS1299, dado que ambos tiene parte digital (procesado de datos y comunicación SPI) y analógica (principalmente en la forma de ADCs). En [45] se recomienda incluir los componentes mixtos en el plano analógico, por este motivo se ha situado en él al ADS1299. Sin embargo, aunque el CYBLE-416045-02 sería también mixto, se ha situado en el plano digital. Se ha tomado esta decisión ya que la gran mayoría de las operaciones que realiza este componente son digitales, siendo las analógicas una parte muy pequeña de su actividad.

Finalmente, se puede apreciar que los planos generados ocupan la práctica totalidad de la parte inferior de la PCB. El motivo es que al hacer las superficies tan extensas, permiten también minimizar las interferencias electromagnéticas o EMI (*ElectroMagnetic Interference*), también llamadas RFI (*Radio Frequency Interference*). En esencia los planos metálicos actúan a modo de escudo, lo que lo hace menos susceptible a las interferencias externas. Así mismo, estos planos también hacen las veces de sumidero de calor, lo que ayuda a regular la temperatura de los componentes del instrumento [45, 46].

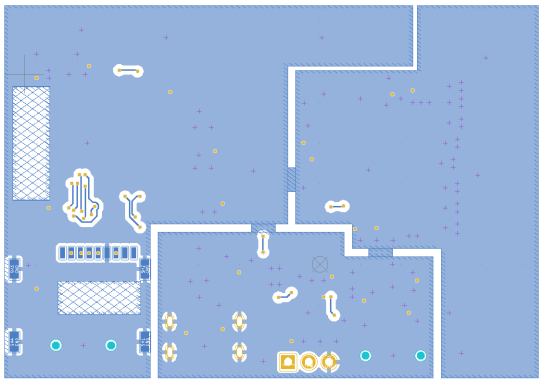


Figura 2.16: Distribución de los planos de tierra en el "Editor de placas" de KiCad.

2.7.1.2. Pistas entre planos

Cuando existen pistas que conectan dispositivos que se encuentran cada uno en uno de los planos, se pasan todas las pistas por encima de dichos puentes. Esto se puede consultar en la Figura 2.17, lo cual se ha hecho siguiendo las recomendaciones de [47]. El motivo de esta decisión de diseño es que la corriente al salir de un componente y volver por el plano de tierra genera un bucle que induce ruido en el sistema. El pasar las pistas por encima de los "puentes" se hace que este bucle sea lo más pequeño posible.

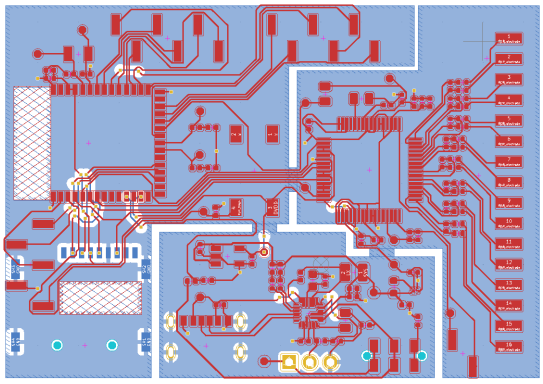


Figura 2.17: Distribución de los planos superior e inferior de cobre en el "Editor de placas" de KiCad.

2.7.1.3. *Keep-Out-Area* de la antena

Como ya se trató en su apartado correspondiente (Módulo para control y comunicación), el CYBLE-416045-02 cuenta con una antena impresa para la comunicación Bluetooth. Tal y como recomienda el fabricante [9], para maximizar el comportamiento de esta antena, se deben de seguir una serie de recomendaciones. La primera de ellas, es situar el CYBLE-416045-02 en uno de los extremos de la PCB, con la antena orienta hacia el borde. La segunda recomendación es despejar de zonas metálicas el área inmediatamente inferior a la antena en todas las capas de la PCB.

2.7.2. Mapa de componentes

En esta sección se resaltan los componentes que se pueden observar en la Figura 2.15. Los puertos genéricos se han colocado en la parte superior de la placa, mientras que en el extremo derecho se sitúan los canales de entrada diferenciales al ADS1299. Para esta última se han preferido pines macho de montaje horizontal, ya que hace el instrumento más ergonómico para la media del electrocardiograma.

En la parte inferior derecha se tiene los pines BIASOUT, que se puede emplear como electrodo de referencia en la toma del ECG, y el pin SRB1 que se puede usar como substituto para la parte negativa de las señales diferenciales. A la izquierda del interruptor de encendido/apagado se colocan los pines de entrada de la batería. Aunque en el modelo 3D mostrado se tienen pines hembra, en la práctica no se han soldados estos pines, si no que se dejan los agujeros para soldar en ellos directamente los cables de la batería.

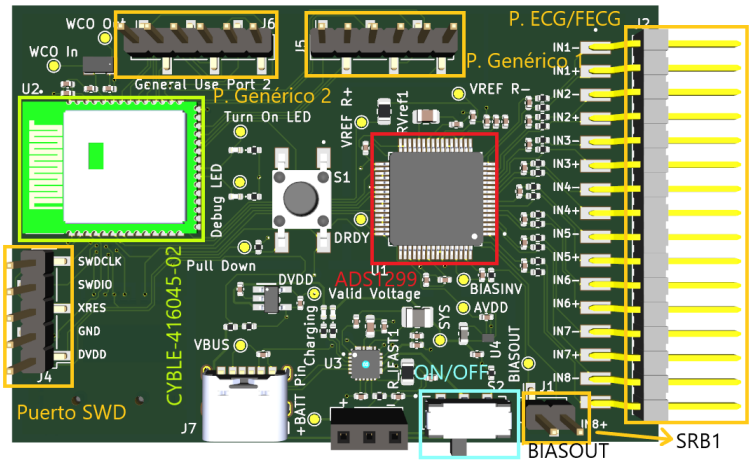


Figura 2.18: Vista superior de la PCB con los componentes principales resaltados.

2.7.3. Mapeo de los pines del CYBLE-416045-02

En este apartado se detallan, de forma breve, los pines utilizados del CYBLE-416045-02, de manera que si, por ejemplo, se desea crear un nuevo kit de expasión que haga uso de uno de los puertos de uso general, se pueda hacer un mapeo adecuado desde el lado de la programación. En la Tabla 2.4 se incluye un listado de a que pin del PSoC se encuentra conectado cada pin físico de los puertos genéricos. Se incluye también la Figura 2.19 de manera que quede claro de a que pin físico de está haciendo referencia en todo momento.

Puerto Genérico	Pin PSoC	Pin Puerto Físico
Puerto 1	P9.5	6
	P9.4	5
	P9.3	4
	P9.2	3
	P9.1	2
	P9.0	1
Puerto 2	P10.6	6
	P10.5	5
	P10.4	4
	P10.3	3
	P10.1	2
	P10.0	1

Tabla 2.4: Listado de la asignación de pines del CYBLE-416045-02 respecto de los puertos físcos

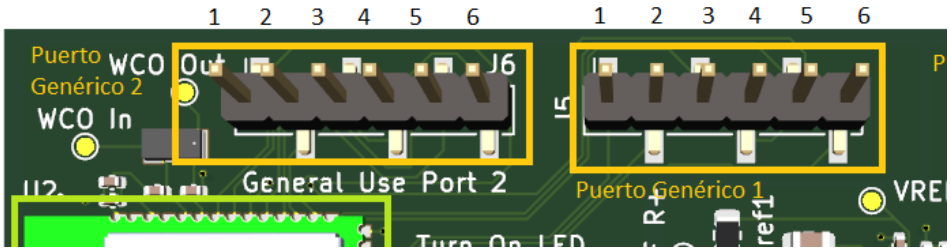


Figura 2.19: Vista superior de la PCB remarcando la enumeración de los pines de los puertos genéricos.

Capítulo 3

Firmware para adquisición del FECG

En este capítulo se puede encontrar la información relativa a la programación del *firmware* del módulo PSoC 63 para la adquisición de la señal cardíaca mediante el instrumento descrito en el Capítulo 2. Como se comentó también en el Capítulo 1. Introducción, este código se generará mediante el *software* PSoC Creator [24] en su versión 4.4. Cabe mencionar que el lenguaje de programación utilizado en todo momento para este fin será C.

El capítulo se ha distribuido de la siguiente manera:

- En el apartado Diagrama de flujo del *firmware* para FECG se puede consultar un diagrama de flujo con el funcionamiento conceptual del *firmware* que permite la adquisición del FECG.
- PSoC Creator facilita la implementación de esquemáticos para indexar los componentes *hardware* a utilizar de entre los que dispone el módulo PSoC 63. El esquemático diseñado para este proceso se encuentra por tanto en Diagramas de componentes en PSoC Creator.
- El *firmware* se ha diseñado de manera modular, para lo cual se han creado una serie de librerías en lenguaje C. La información relativa a dichas librerías se localiza en el apartado Librerías C para obtención del FECG.

3.1. Diagrama de flujo del *firmware* para FECG

En este aparatado se comenta brevemente el diagrama de flujo (Figura 3.1) del código C que controla el instrumento descrito en el Capítulo 2. En primer lugar, se da la inicialización de los componentes del CYBLE-416045-02, tras lo cual el módulo comienza a transmitir paquetes indicando que está disponible para que se conecte un cliente. Una vez este se conecta, el instrumento queda a la espera de instrucciones.

Si se envía la instrucción de comienzo de adquisición, el dispositivo ordena que el ADS1299 comience a captar muestras y empieza a transmitir los resultados al cliente hasta que se hayan transmitido suficientes datos. Por el contrario, se puede enviar también la orden de depuración del ADS1299. Para comprobar que el ADS1299 funciona correctamente, Texas Instruments incluye que una de las instrucciones que puede recibir sea una solicitud de varios identificadores del integrado. Cuando se recibe entonces esta orden, por parte del cliente, el PSoC 63 reenvía esta instrucción al ADS1299 y obtiene el resultado. Este proceso se emplea tanto para depurar la comunicación entre ADS1299 y PSoC 63,

como para asegurarse de que el *frontend* de Texas se encuentra operativo.

Finalmente, se comprueba si el cliente se ha desconectado tras la recepción de las muestras. Caso de que esto haya ocurrido, el instrumento vuelve a emitir que está listo para la conexión de un nuevo maestro. Nótese que si la desconexión ocurre en cualquier otro punto de la ejecución, llevará al mismo resultado.

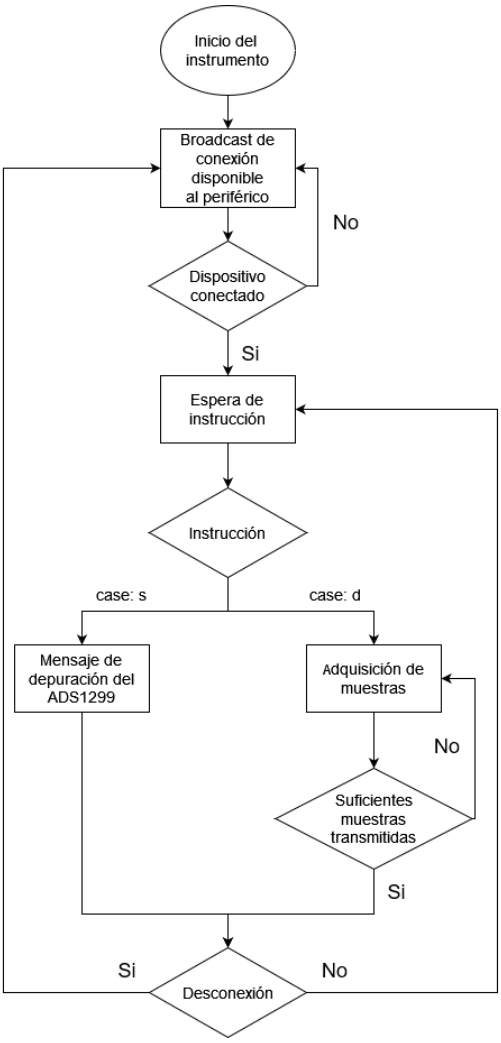


Figura 3.1: Diagrama de flujo del *firmware* del instrumento.

3.2. Diagramas de componentes en PSoC Creator

Tal y como ya se trató en la sección PSoC Creator del Capítulo 1, PSoC Creator facilita el uso y conexión de sus componentes internos mediante el empleo de una interfaz gráfica. En este apartado se pueden consultar los diagramas en los cuales se han incluido los componentes necesarios para la obtención de la señal cardíaca fetal.

En primer lugar se han incluido, en la Figura 3.2, los bloques necesarios para la comunicación del PSoC 63 con el resto de elementos del sistema. Como se ha tratado, el bloque "BLE" se emplea para la configuración de los recursos Bluetooth para la recepción de ordenes desde el cliente y para la transmisión de las muestras resultantes a este.

Adicionalmente, se incluyen otros dos bloques de comunicación. Por un lado, se incluye una interfaz SPI para dos esclavos. Esta será la que use el controlador para la comunicación con el ADS1299 para la obtención de las muestras y para el guardado de un respaldo de los datos en la tarjeta microSD. Por el otro lado, se ha añadido una interfaz de comunicación UART (*Universal Asynchronous Receiver-Transmitter*). Esta se emplea para la depuración del *firmware* durante el desarrollo del instrumento. Por ello, se suprimirá de la versión final del instrumento.

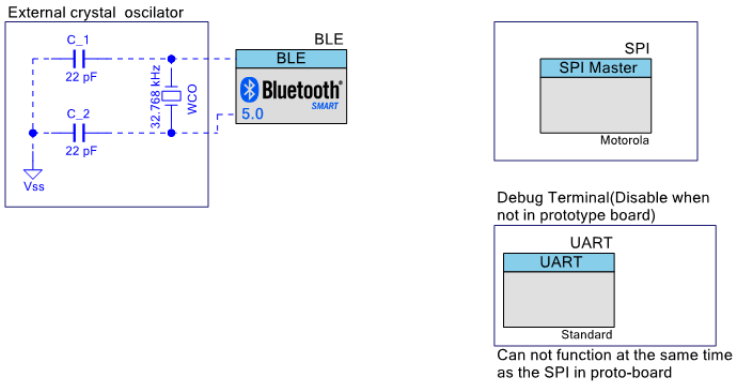


Figura 3.2: Esquemático de PSoC Creator con los bloques necesarios para la comunicación entre el PSoC y el resto de elementos.

En la Figura 3.3, se pueden consultar los componentes utilizados del PSoC 63. Se incluyen un conversor digital-analógico, el cual se puede emplear para generar un voltaje de referencia que establezca el límite superior de resolución de los ADCs incluidos en el ADS1299. A este respecto se puede encontrar más información en el apartado Esquemático ADS1299. Cabe recordar que no siempre será utilizado, ya que en la configuración del integrado se puede indicar el uso de un valor generado de manera interna. Sin embargo, el significado de poder contar con esta funcionalidad, caso de que se estime necesario, es el poder maximizar la precisión de las muestra obtenidas por los conversores.

Cuando se tiene un conversor analógico-digital, cuanto mayor sea el rango de valores que el dispositivo es capaz de muestrear, menor sera la sensibilidad de cada muestras, debido a que los bits de resolución se tienen que repartir entre un mayor rango de valores. En este trabajo, se va a partir de los límites por defecto de los conversores del ADS1299. Si se comprueba que no está aprovechando todo el rango de dichos conversores, se utilizará el DAC de la Figura 3.3, para reducir dicho rango y, con ello, incrementar la sensibilidad de las medidas.

En cuanto a los otros bloques que también se encuentran en esta imagen, se incluyen un ADC tipo SAR (*Successive Approximation Register*). Este será el que se encargue de monitorizar el valor de la tensión de salida de la batería del sistema, para poder realizar la estimación del estado de carga de la misma. Se incluyen también el pin de entrada del pulsador de *pull-down*, los pines de activación de los dos LEDs internos, usándose uno de ellos para indicar la transmisión activa de muestras mediante Bluetooth y, finalmente, el pin DRDY que indica, mediante un cambio de flanco, que el ADS1299 ha obtenido una nueva muestra.

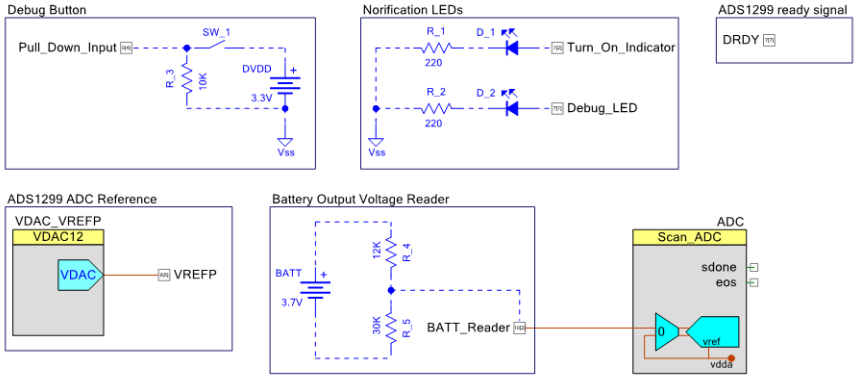


Figura 3.3: Esquemático de PSoC Creator con entradas y salidas varias para la obtención del FECG.

3.3. Librerías C para obtención del FECG

Uno de los objetivos del dispositivo desarrollado en este trabajo, es que sea reprogramable y, por tanto, sus funcionalidades se puedan ir ajustando según las necesidades del usuario final. Para ello, una buena práctica que facilita la reutilización de código y hace a su vez este más legible y sencillo de depurar, es la modularización del mismo. Es decir, se van a crear varias librerías en C, en donde se incluirán funciones que en conjunto realizan el proceso completo control del instrumento. En este apartado, se van a tratar las librerías que se han creado para implementar el *firmware* que permite la obtención del FECG.

3.3.1. Configuración Bluetooth

En este apartado se puede consultar la configuración y programación llevada a cabo en el lado del instrumento, como para lograr comunicación Bluetooth entre este y un controlador remoto.

3.3.1.1. Conceptos básicos de Bluetooth de Baja Energía

Antes de entrar en detalle sobre como se va a configurar la comunicación en este dispositivo, se va a aclarar algunos conceptos relativos a Bluetooth *Low Energy* (BLE) 5.0. Lo primero, es comentar que para identificar ciertas partes de la configuración se utilizan UUIDs (Identificador Único Universal). Estos son identificadores de 128 bits que diferencian una información de manera única. En este caso, se emplean para diferenciar los servicios y, a su vez, la distintas características que puede albergar cada servicio.

Las características se pueden definir, según la propia documentación oficial, como “Instancias de datos de estado que reflejan algún aspecto o capacidad del dispositivo” [4], teniendo cada una un UUID para diferenciarla. Un ejemplo aclaratorio de este concepto, puede ser el hecho de tener una característica tipo “Write”. Esto quiere decir que dicha característica está asociada a un valor guardado en la memoria del dispositivo BLE la cual el usuario puede editar. Si en cambio, se tuviese una tipo “Notify”, se puede hacer que cada vez que un parámetro o conjunto de ellos (en forma de *buffer*) cambie de valor, se le notifique al cliente y este pueda recuperar la nueva información.

En cuanto a los servicios, vienen a estar definidos como una agrupación de características asociadas y que proporcionan un contexto a partir del cual se puede determinar el significado y las normas de utilización de las características que contiene el servicio.

Esto reformulado viene a significar que se pueden agrupar características relacionadas en un mismo servicio para darles contexto [4].

Otro concepto que se va a emplear es el de la *Link Layer* (LL). Esta es la capa más baja del modelo de pila del protocolos Bluetooth. Es responsable de establecer y mantener la conexión física entre dispositivos BLE, así como de gestionar la transmisión de datos entre ellos. Para acabar, se menciona el MTU (*Maximum Transmission Unit*), el cual se refiere al máximo tamaño de un paquete de datos que puede ser transmitido entre dispositivos en una sola transacción, siendo su negociación parte del proceso de conexión entre dispositivos BLE. En versiones posteriores de Bluetooth, el tamaño máximo del MTU es de 23 bytes. BLE 5.0 y posteriores permiten incrementar este tamaño, en concreto el PSoC 63 soporta hasta los 512 bytes [48].

Mencionados todos conceptos, en los subapartados siguiente se trata como se han configurado cada uno de ellos.

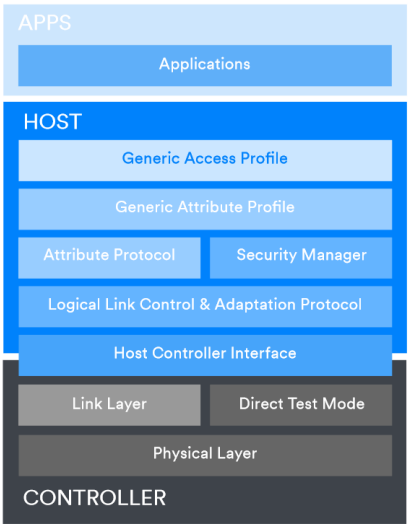


Figura 3.4: Capas de la comunicación BLE 5.0 [4].

3.3.1.2. Configuración BLE usada en PSoC 63

Infineon permite configurar el CYBLE-416045-02 en cuatro roles diferentes: *Peripheral*, *Central*, *Broadcaster* y *Observer*. Para este trabajo se utiliza la configuración de *Peripheral* o periférico, en la cual el dispositivo actúa a modo de esclavo, recibiendo instrucciones desde un dispositivo central o cliente que es que debe de iniciar la comunicación. En el contexto aquí presentado este cliente será el ordenador o dispositivo móvil utilizado por el usuario.

Se ha creado un servicio personalizado que cuenta con dos características, denominadas "DataAcquisition" y "ControllerOrders". La primera de ellas es de tipo "Notify", mientras que la segunda es "Write". El cliente se puede subscribir a la características "Notify", de manera que a medida que el procesador vaya contando con muestras de la adquisición, el cliente las vaya recibiendo. En cuanto a la característica de escritura, como su nombre indica, se utiliza para que el cliente pueda enviar ordenes al instrumento. Por ejemplo, el dispositivo se encuentra en reposo y no comienza la adquisición de muestras hasta que el

usuario envíe la orden pertinente desde el controlador remoto.

Para acabar, el dispositivo se configura de manera que se maximice la velocidad de transmisión de las muestras. Para ello, se habilita la velocidad de transmisión en capa física de 2 Mbps, se incrementa el tamaño de paquete de la LL por encima del máximo de 23 bytes que permitían las versiones anteriores de BLE y, finalmente, se utiliza un MTU de 512 bytes, el máximo permitido. Con todo esto, se puede transmitir un mayor volumen de datos y a mayor velocidad de la que sería posible en versiones 4.0.

La comunicación entre dos o más dispositivo en BLE se gestiona mediante eventos. Estos se refieren a las acciones o sucesos que ocurren durante la comunicación. Por ejemplo, cuando un dispositivo se conecte a al CYBLE-416045-02, se generará el evento correspondiente. En la práctica esto activa un *flag* específico en el código. La documentación completa con las *flags* generadas para cada evento BLE, para el caso concreto de utilizar el CYBLE-416045-02, se puede consultar en la documentación "Bluetooth Low Energy Middleware Library" [49], en la cual se puede encontrar también, por ejmeplo, la API referente al uso del módulo BLE.

Se ha creado una librería, llamada "BLE_handler", la cual incluye la gestión de los eventos y todos aquellos métodos creados para lidiar con la comunicación Bluetooth a nivel de *firmware*. En los siguientes apartados se explican algunas de las funcionalidades clave implementadas en esta librería.

3.3.1.3. Control de eventos BLE

El primer paso, es inicializar el módulo BLE con los parámetros que se mencionaban en el apartado Configuración BLE usada en PSoC 63. Esta configuración se introduce de manera gráfica, para acceder a la pestaña de configuración no hace falta más que hacer doble *click* sobre el bloque "BLE" de la Figura 3.2 en el esquemático de PSoC Creator. Hecho esto, utilizando el Código 3.1, el bloque que gestiona la comunicación BLE comienza a operar.

Nótese que en la línea 11, se introduce "callback" como parámetro. Esta no es más que otra función que se va invocar siempre que se produzca un nuevo evento BLE. Dicho método va a contener, por tanto, un listado de todos los eventos posibles y que debe de hacer cada vez que se produzca uno de ellos, si es que es necesario hacer algo. Es decir, de los eventos que se registran algunos simplemente son informativos y no es necesario ejecutar una lógica específica cuando se den.

En cuanto a "api_result", no es más que una estructura que se utiliza para guardar el resultado del evento producido. Por ejemplo, en este caso para hacer la depuración del código de manera más sencilla, se ha hecho que se notifique por puerto serie mediante comunicación UART, si el resultado de iniciar el bloque Bluetooth ha sido exitoso. Estos mensajes de depuración no son necesarios para la versión final del dispositivo y pueden llegar a ser eliminados *a posteriori*, ya que es una información que en ningún momento se va a mostrar tal cual al usuario.

```
1  void Ble_Init(void){
2          /*
3          Starts the BLE module
4
5          Inputs:
6          - None
7          Returns:
8          - None
9          */
10
11         api_result = Cy_BLE_Start(callback);
12
13         if(api_result != CY_BLE_SUCCESS)
14         {
15                 Cy_SCB_UART_PutString(UART_HW, "Cy_BLE_Start API
                       Error \r\n");
16         }
17         else
18         {
19                 Cy_SCB_UART_PutString(UART_HW, "BLE Stack
                       Initialized \r\n");
20         }
21 }
```

Código 3.1: Inicialización del módulo para la comunicación BLE

Se muestran ahora la función "callback" que gestiona la lógica que se considere necesaria cada vez que se de un evento. Nótese que este método resulta muy extenso, con lo cual sólo se van a mostrar en este documento algunos eventos relevantes a modo de ejemplo. En el Código 3.2 se muestra el caso de, por un lado, la lógica que se ejecuta cuando se pierde la conexión con un dispositivo que se encontraba conectado al CYBLE-416045-02. Por el otro, se puede ver también que instrucciones se ejecutan cuando se recibe un evento de solicitud de escritura en la característica "Write" correspondiente.

Cuando el dispositivo remoto se desconecta, se levanta la *flag* de la línea 14. Lo que se ha optado por hacer en esta situación, es reiniciar el *Advertising*. Dicho de otra manera, el instrumento vuelve a emitir al aire la indicación de que se encuentra buscado un dispositivo maestro con el que emparejarse. Esto se hace con la intención de poder volver a emparejarlo con el dispositivo original con el cual se perdió la comunicación o por el contrario emplear un nuevo controlador.

En cuanto al otro evento que se muestra en la línea 26, lo que se hace es comprobar que caracter ha escrito el usuario. Cada uno de estos se relaciona con una orden específica. Por ejemplo, en esta versión del código, si el usuario envía el caracter "s", se modifica una *flag* o bandera que hará que en el código principal comience la adquisición de muestras. Si en cambio, se recibe la letra "d", se inicia un proceso de *depuración* que compruebe si el ADS1299 funciona correctamente. Finalmente, si no se capta una letra asociada a una orden conocida, el dispositivo no realiza ninguna acción.

El *flag* que identifica a la instrucción, se puede leer en el código principal a partir del método mostrado en el Código 3.3. Se ha hecho de esta manera con la intención de que las variables que se utilicen en una librería sean independientes de las que se puedan encontrar en otra, de forma que no existan variables globales compartidas entre multitud de archivos, lo cual puede incrementar la tendencia a fallos del sistema.

```
void callback(uint32_t event, void* eventParam){
        /*
        Custom callback to handle the BLE events
        */

        uint8_t instr;  // Character with the order from the client

        switch(event){

                .
                .
                .

                case CY_BLE_EVT_GAP_DEVICE_DISCONNECTED:
                // This event indicates that the device has
                   disconnected from remote device or failed to
                   establish connection
                // Restart BLE advertisement after the device has
                   been disconnected
                {
                        Cy_SCB_UART_PutString(UART_HW, "
                           Disconnected \r\n");
                        negotiatedMtu = DEFAULT_MTU_SIZE;
                        Cy_SCB_UART_PutString(UART_HW, "Restarting
                           Advertising After Disconnection \r\n");
                        Cy_BLE_GAPP_StartAdvertisement(
                           CY_BLE_ADVERTISING_CUSTOM,
                           CY_BLE_PERIPHERAL_CONFIGURATION_0_INDEX)
                           ;
                        Cy_GPIO_Write(Debug_LED_PORT, Debug_LED_NUM
                           , 1);  // Debug LED OFF
                        break;
                }

                case CY_BLE_EVT_GATTS_WRITE_REQ:
                // This event indicates that the 'Write Request' is
                    received from GATT Client device (GATT: Write)
                // Identifies wich order the client as sent and
                   update the necessary flags
                {
                        Cy_SCB_UART_PutString(UART_HW, "Write
                           Request from Client \r\n");

                        cy_stc_ble_gatts_write_cmd_req_param_t
                           writeReqParameter;

                        writeReqParameter = *(
                           cy_stc_ble_gatts_write_cmd_req_param_t*)
                           eventParam;
                        api_result = Cy_BLE_GATTS_WriteRsp(
```

```
                                writeReqParameter.connHandle);

                        if(api_result == CY_BLE_SUCCESS){
                                Cy_SCB_UART_PutString(UART_HW, "
                                    Write Sucessful \r\n");
                        }
                        else{
                                Cy_SCB_UART_PutString(UART_HW, "
                                    Write Error \r\n");
                        }

                        instr = writeReqParameter.handleValPair.
                            value.val[0];

                        if(instr == 's'){
                                Cy_GPIO_Write(Debug_LED_PORT,
                                    Debug_LED_NUM, 0);  // Debug LED
                                     ON
                                instruction_flag =
                                    START_ACQUISITION;
                        }
                        else if(instr == 'd'){
                                instruction_flag =
                                    DEBUG_ADS1299_MESSAGE;
                        }
                        else{
                                instruction_flag = IDLE;
                        }
                        break;
                }

                .
                .
                .

        }
```

Código 3.2: Código C con ejemplos de algunos de los eventos contemplados en la función "callback".

```
uint8_t get_instruction_flag(void){
        /*
        Returns the flag that indicates if an acquisition should
            start

        Inputs:
        - None
        Returns:
        - ID of the instruction to execute
        */

        return instruction_flag;
}
```

Código 3.3: Función para leer en el código principal el identificador de la instrucción recibida.

3.3.1.4. Adaptación previa de las muestras

Como ya se ha tratado, las muestras de la señal cardíaca se adquieren con una resolución de 24 bits. Sin embargo, previamente a realizar la transmisión BLE, estas muestras se deben de dividir en grupos de 8 bits. Esto quiere decir, que cada muestras captada, se divide en tres grupos consecutivos de 1 byte cada uno, previamente a ser transmitidos.

Por otra parte, como se comentaba al principio de esta sección, la máxima cantidad de bytes que se pueden transmitir en una misma transición son 512. Lo que se va a hacer, por tanto, es realizar un control manual de la cantidad de bytes transmitida. En esencia, se gestiona que las muestras se envíen en lotes de 480 bytes. Dicho de otra forma:

1. Se capturan muestras de la señal cardíaca con una resolución de 24 bits, lo que es lo mismo que 3 bytes.
2. Se dividen las muestras en grupos consecutivos de 1 byte cada uno.
3. Cuando se tienen un total de 480 bytes, teniendo en cuenta la fragmentación del paso anterior, se transmiten al cliente.

En el Código 3.4 se puede consultar la función usada para el envío de muestras a través de la característica tipo "Notify" al cliente.

```
1  bool send_notification(uint8_t *data_buffer){
2          /*
3          Send notification to indicate to the client that the data
               in the "notify" characteristic as been updated
4
5          Inputs:
6          - None
7          Returns:
8          - True if the sending was sucessful or false if it was not
9          */
10
11         bool sending_sucessful = false;
12
13         // Update Notification packet with the data to be sent
14         notificationPacket.connHandle = appConnHandle;
15         notificationPacket.handleValPair.attrHandle =
               CUSTOM_SERVO_CHARO_HANDLE;
16         notificationPacket.handleValPair.value.val = data_buffer;
17         notificationPacket.handleValPair.value.len =
               NOTIFICATION_PKT_SIZE;
18
19         while(Cy_BLE_GATT_GetBusyStatus(appConnHandle.attId) ==
               CY_BLE_STACK_STATE_BUSY){
20                 Cy_BLE_ProcessEvents();
21         }
22
23         if(Cy_BLE_GATT_GetBusyStatus(appConnHandle.attId) ==
               CY_BLE_STACK_STATE_FREE){
24                 api_result = Cy_BLE_GATTS_Notification(&
                       notificationPacket);  // Notifies in the
                       characteristic
25                 sending_sucessful = true;
26         }
```

```
27          else{
28                  sending_sucessful = false;
29          }
30
31          return sending_sucessful;
32  }
```

Código 3.4: Función para transmisión de valores de 1 byte al cliente suscrito a la característica.

3.3.2. Métodos para control del ADS1299

Se ha creado una librería en C para poder gestionar la recepción de muestras desde el ADS1299 y la configuración de este. Como se trató en Módulo para muestreo y preprocesado de la señal cardíaca, el *frontend* para la adquisición de muestras está enfocado a actuar como un esclavo, comunicándose con el maestreo, en este caso el PSoC, mediante el protocolo SPI. En el Código 3.5 se puede consultar una función diseñada para la gestión de las comunicación mediante SPI.

En la Tabla 3.1 se muestran los comando aceptados por el integrado desde el maestreo. En el caso de la lectura o escritura de un registro o conjunto de ellos, el maestro debe de enviar un primer byte con la forma "001r rrrr", caso de lectura, o "010r rrrr", caso de escritura. El primer registro a partir del cual se va a empezar a leer/escribir, viene indicado por los bits "r rrrr". En caso de que se utilice uno de estos dos comandos, se entiende que el siguiente byte enviado por el maestro, indica cuantos registros se van a leer/escribir, menos uno, partiendo de la posición inicial indicada en el byte anterior.

Por poner un ejemplo, si se ha indicado que se van a leer dos registros, el ADS1299 devolverá, a través de la conexión DOUT, dos cadenas, cada una de 1 byte conteniendo la información de los registros leídos. Si por otra parte, lo que se hizo fue escribir en dos registros, el maestro debe de enviar dos cadenas de 1 byte cada una, siendo la primera cadena la que se va a almacenar en el primer registro y viceversa.

```
1  void handle_SPI_transfer(uint8_t *w_buffer, uint8_t *r_buffer,
       uint32 transfer_size){
2          /*
3          Manages the SPI communication between the master and the
               slave
4
5          Inputs:
6          - w_buffer: Array of bytes to send to slave
7          - r_buffer: Array of bytes where to store the response of
               the slave
8          - transfer_size: Number of bytes to be transmitted between
               master and slave
9          Returns:
10         - None
11         */
12
13         cy_en_scb_spi_status_t error_status;
14         error_status = Cy_SCB_SPI_Transfer(SPI_HW, w_buffer,
               r_buffer, transfer_size, &SPI_context);
15
```

```
16          if(error_status == CY_SCB_SPI_TRANSFER_BUSY){
17                  Cy_GPIO_Write(Debug_LED_PORT, Debug_LED_NUM, 0);
                        // Debug LED ON
18          }
19          else{
20                  Cy_GPIO_Write(Debug_LED_PORT, Debug_LED_NUM, 1);
                        // Debug LED OFF
21          }
22
23          while(Cy_SCB_SPI_GetTransferStatus(SPI_HW, &SPI_context) ==
                CY_SCB_SPI_TRANSFER_ACTIVE);  // Blocks until the
               communication has finished
24  }
```

Código 3.5: Función para control de comunicación SPI.

Comando	Descripción	1º BYTE	2º BYTE
Comandos del Sistema			
WAKEUP	Despertar del modo de espera	0000 0010 (02h)	
STANDBY	Entrar en modo de espera	0000 0100 (04h)	
RESET	Reiniciar el dispositivo	0000 0110 (06h)	
START	Iniciar y reiniciar (sincronizar) conversiones	0000 1000 (08h)	
STOP	Detener conversión	0000 1010 (0Ah)	
Comandos de Lectura de Datos			
RDATAC	Habilitar modo de Lectura de Datos Continua. Este es el modo predeterminado al encender	0001 0000 (10h)	
SDATAC	Detener modo de Lectura de Datos Continua	0001 0001 (11h)	
RDATA	Leer datos por comando; soporta múltiples lecturas.	0001 0010 (12h)	
Comandos de Lectura de Registro			
RREG	Leer n nnnn registros empezando en la dirección r rrrr	001r rrrr (2xh)	000n nnnn
WREG	Escribir n nnnn registros empezando en la dirección r rrrr	010r rrrr (4xh)	000n nnnn

Tabla 3.1: Comandos del integrado ADS1299

Capítulo 4

Desarrollo de soporte para adquisición de SpO2

Tal y como se comentaba en el Capítulo 1: Introducción, a pesar de ser el principal el objetivo del proyecto la medida de la señal cardíaca fetal, se han incluido también puertos de uso genérico en el instrumento desarrollado en el Capítulo 2: Diseño del dispositivo, con la intención de poder incluir electrónica externa adicional que permita la captura de otros biomarcadores de interés. A modo de demostración, en este capítulo se incluye el desarrollo de la electrónica requerida y la reprogramación del instrumento necesarios como para poder adquirir el porcentaje de saturación de oxígeno en sangre o SpO_2.

- En el apartado Introducción a la adquisición del SpO2 se puede consultar un breve resumen sobre los principios en los que se basa al adquisición de la saturación de oxígeno en sangre a partir de métodos lumínicos.
- La sección Diseño *hardware* de un sistema para la adquisición del SpO2 abarca la información relativa al *hardware* adicional que se ha tenido que desarrollar para poder adquirir este parámetro. Por otra parte, en Sección 4.3, se encuentra el diseño *firmware*.
- Resulta de interés implementar el control de la adquisición de las muestras mediante un sistema operativo de tiempo real. La información respecto a como se hecho esto se localiza en Gestión de la adquisición mediante un RTOS.

4.1. Introducción a la adquisición del SpO2

El porcentaje de saturación de oxígeno en sangre, comúnmente abreviado como SpO_2, es una medida de la cantidad de oxígeno que está siendo transportado por la hemoglobina en los glóbulos rojos en comparación con la cantidad total de hemoglobina que puede transportar oxígeno. Se expresa como un porcentaje y es una indicación clave de si el oxígeno está siendo adecuadamente suministrado a los tejidos del cuerpo [50].

En este trabajo se van a utilizar técnicas pletismográficas, también conocidas como PPG o *Photoplethysmography*, para la obtención del SpO_2. Esta es una técnica no invasiva que permite medir los cambios en el volumen sanguíneo en una zona del cuerpo, generalmente, en la punta de los dedos, el lóbulo de la oreja o el pie [51]. Para ello, se realiza el siguiente proceso [51, 52]:

1. **Emisión de luz**: Se emplean dos emisores lumínicos. Uno de ellos, emitirá en torno al espectro de la luz roja (longitudes de onda de alrededor de 660 nm) y el otro en el infrarrojo (alrededor de 940 nm). Hay que tener en cuenta que no se utilizan ambos de manera simultánea, si no que se mide en cada momento con, únicamente, uno de ellos activo.

2. **Absorción lumínica en el cuerpo**: La luz emitida atravesará la piel y los tejidos subyacentes. La hemoglobina oxigenada (HbO_2) y la hemoglobina desoxigenada (Hb) absorben estas longitudes de onda de manera diferente. Absorbiendo la HbO_2 más luz infrarroja, mientras que la Hb absorbe más la luz roja. El distinto comportamiento respecto a la absorción lumínica de ambos tipos de hemoglobina se puede aprovechar en el cálculo de la cantidad de HbO_2 o Hb presentes. Se puede consultar, en la Figura 4.1, el gráfico de la absorción espectral de la hemoglobina oxigenada y desoxigenada.

   La variación en las propiedades de absorción de las dos clases de hemoglobina hace que las mediciones fluctúen durante la sístole y la diástole del corazón. Esto se traduce en que la medición tenga una componente alterna (AC) y una continua (DC). La componente DC se debe a la presencia de músculo, huesos, grasa, etc., y aunque es relativamente constante, sufre pequeñas variaciones debido a factores como la respiración del paciente, entre otras razones. La componente AC se genera por la sangre que fluye a través de las arterias durante la sístole, ya que el aumento en el volumen de sangre modifica la absorción de luz. La fase ascendente de la componente AC ocurre durante la sístole, mientras que la fase descendente es causada por la diástole. En la Figura 4.2 se puede consultar la variación en la absorción lumínica a causa del bombeo del corazón, lo cual causa cambios en el volumen de sangre presente durante la sístole y la diástole.

3. **Detección de luz**: Se emplea uno, si de banda ancha, o dos fotodetectores para captar la cantidad de luz absorbida para cada una de las longitudes de onda. Dicho detector se puede situar de modo que capte la luz transmitida, colocándose por lo tanto en el lado opuesto del emisor de luz, o para captar la reflejada, situándose entonces el receptor junto al emisor. Según la zona del cuerpo donde se vaya a tomar la medida, se busca adquirir una u otra. Por ejemplo, en el caso de que esta se realice en la punta de los dedos o en el lóbulo de la oreja se adquiere la transmisión, mientras que en el resto de partes del cuerpo se mide la cantidad reflejada.

4. **Cálculo de la saturación**: Para calcular el valor de SpO_2 a partir de las muestras capturadas de ambos emisores, se debe de obtener previamente el índice de absorbancia (R). Este se obtiene a su vez a partir de la relación entre la componente AC y DC de la Figura 4.2. Por tanto, este valor se puede llegar a conocer a partir de la Ecuación 4.1. Donde a su vez la componente continua se puede obtener de la expresión Ecuación 4.2 y la alterna de la Ecuación 4.3 [53]. En ambos casos $m[i]$ hace referencia al vector de muestras extraído por el ADC y N al número total de muestras.

$$R = (\frac{PPG_{AC,Red}}{PPG_{DC,Red}})/(\frac{PPG_{AC,IR}}{PPG_{DC,IR}}) \tag{4.1}$$

$$PPG_{DC} = \frac{1}{N}\sum_{i=0}^{N} m[i] \tag{4.2}$$

$$PPG_{AC} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(m[i] - PPG_{DC})^2} \tag{4.3}$$

Una vez conocido el índice R, se puede calcular a partir de él el porcentaje de saturación de oxígeno en sangre. Para ello, existen varias curvas de calibración que permiten relacionar ambos valores. En este trabajo se utilizará la curva de calibración estándar de la pulsioximetría (Ecuación 4.4) ampliamente utilizada en la bibliografía [54, 55, 56]:

$$SpO_2 = 110 - 25R \tag{4.4}$$

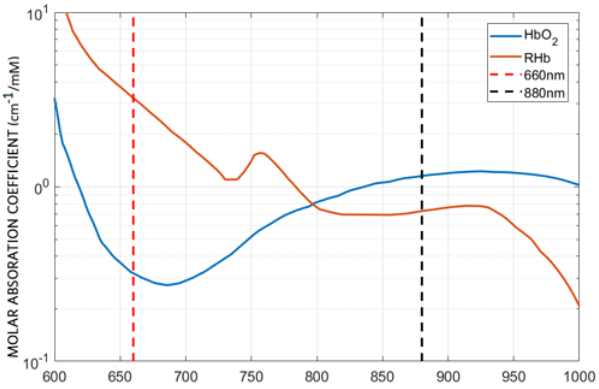


Figura 4.1: Diagrama de la absorción espectral de la hemoglobina [5].

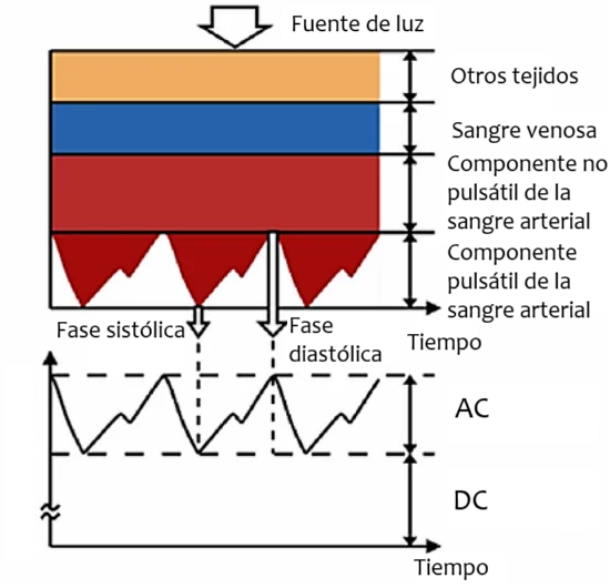


Figura 4.2: Variación en la absorción lumínica en una zona del cuerpo debido al cambio de volumen de la sangre durante el bombeo del corazón [6].

4.2. Diseño *hardware* de un sistema para la adquisición del SpO2

En esta sección se trata el diseño *hardware* del kit de expansión que añade la electrónica adicional que permite la adquisición del $SpO2$. Los componentes con los que debe de contar dicho kit son:

- **LEDs**. Uno cuya longitud de onda de emisión principal sea de 660 nm y otro de 940 nm. El haz de luz debe de ser lo más concentrado posible para asegurar una buena penetración en la piel.
- **Fotorreceptor/es**. Se puede incluir o bien un fotorreceptor de banda ancha capaz de captar las dos longitudes de onda del punto anterior, o bien dos receptores, cada uno con respectivamente una de la longitudes de onda como punto de sensibilidad central.
- **Conversor de corriente a tensión**. Se plantean utilizar fotodiodos a modo de fotorreceptores, no siendo este más que un dispositivo semiconductor que convierte la luz en corriente eléctrica. Dado que lo que el ADC muestrea son niveles de tensión y no de corriente, se tiene que hacer una conversión de la corriente generada en él, debido a la luz incidente, a tensión. A este fin, se utiliza un amplificador de transimpedancia (TIA).

Para incluir los LEDs y los fotodiodos, se utiliza el componente **SFH7072** [57] de OSRAM, el cual ya incluye dos emisores, de 660 nm y 950 nm, además de los dos fotodiodos necesarios para captar cada una de estas longitudes de onda.

En la Figura 4.3 se puede consultar un diagrama de bloques donde, a nivel conceptual, donde se muestran los subsistemas necesarios para la captura del porcentaje de saturación de oxígeno en sangre.

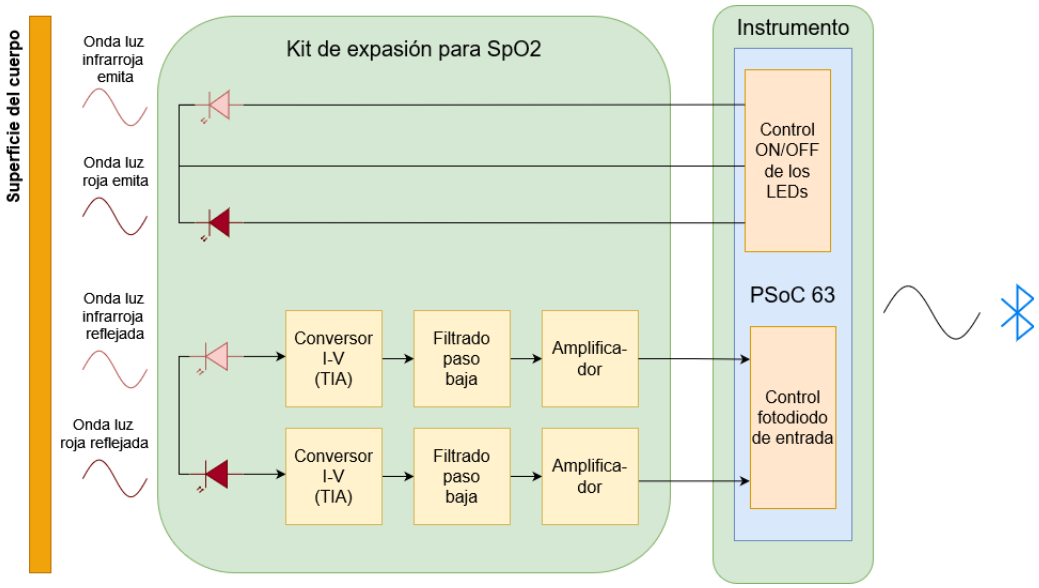


Figura 4.3: Diagrama de bloques del *hardware* necesario para realizar la adquisición del SpO_2.

4.2.1. Diseño de un TIA

En este apartado se puede consultar el diseño, de manera resumida, de un amplificador de transimpedancia. A modo de guía básica se puede utilizar la información disponible en [58]. Este circuito está formado por un un amplificador operacional, una resistencia (R_f), quien establece la conversión de corriente a tensión mediante la Ecuación 4.5 y un condensador (C_f), llamado de realimentación, el cual permite ajustar el ancho de banda del sistema.

$$V_o = R_f I_f \tag{4.5}$$

I_f es la corriente de salida del fotodiodo. En este caso no se espera que sea de más de 1.1 μA [57], con lo cual se selecciona una R_f de 1M Ω, de modo que la máxima tensión de salida proporcionada por el TIA será de 1.1 V.

Normalmente, el corazón late a un ritmo de entre 60 a 100 ppm (pulsaciones por minuto). Dado que el cambio en el volumen de la sangre y con ello el cambio en la tensión de salida del convertidor, dependerá del ritmo cardíaco, se puede realizar un filtrado paso baja con frecuencia de corte de 15Hz. De esta manera, se eliminan las frecuencias superiores al ritmo del corazón, ya que en este estudio dichas bandas sólo aportan ruido. El espectro de la salida también se va a limitar a partir del ajuste del ancho de banda del propio TIA, a partir del condensador de realimentación. Para seleccionar el valor del condensador de realimentación, se puede utilizar la Ecuación 4.6, donde f_c es la frecuencia de corte del sistema, es decir, la frecuencia a la cual la ganancia del amplificador sufre una caída de -3 dB. Con la intención de atenuar aún más las frecuencias superiores, se añade también un filtro analógico de primer orden a la salida del TIA.

$$C_f \leq \frac{1}{2\pi R_f f_c} \tag{4.6}$$

Se tiene que comprobar que ancho de banda debe de tener el operacional usado, de modo que el sistema sea estable. Esto se puede hacer a partir de la Ecuación 4.7, donde C_i es la capacidad total de entrada del TIA, calculada como la suma de la capacitancia del fotodiodo (C_{pd}) más las capacitancias parásitas, en modo común (C_{cm}) y en modo diferencial (C_d), del amplificador operacional.

$$GBW > \frac{C_i + C_f}{2\pi R_f C_f^2} \tag{4.7}$$

$$C_i = C_{pd} + C_d + C_{cm} \tag{4.8}$$

El modelo de amplificador operacional escogido para implementar el conversor de corriente a tensión, será el **OPA380** de Texas Instruments [59]. Los condensadores parásitos C_{cm} y C_d de este integrado son 3 pF y 1 pF. Adicionalmente, el amplificador cumple otra condición importante y es que dado que el kit de expansión se va alimentar desde el instrumento desarrollado en el Capítulo 2, está limitado a operar a las tensiones de alimentación de 3.3 V y de 5 V. Dado que el OPA380 funciona en el rango de 2.7 V a 5.5 V, es válido en estas condiciones.

Se escoge una frecuencia de corte de 1 kHz, con lo cual se calcula que C_f de tener una capacidad de como máximo 159 pF. Sin embargo, este no es un valor comercial, así que se utiliza finalmente uno de 150 pF. Como ya se comentó el SFH7072 cuenta con dos fotodiodos, uno de ellos tiene una capacidad de 55 pF, mientras que el otro la tiene de 4.2 pF [57]. Si se parte del peor caso, es decir, se usa el fotodiodo de mayor capacidad, el ancho de banda del operacional debe de ser de al menos 1.48 kHz, condición que el OPA380 cumple.

Por otra parte, el rango que puede muestrear el ADC embebido en el PSoC es en este proyecto de 0 a 6.6 V. Se puede incluir un amplificador consecutivo al filtro de salida del TIA de modo que se aproveche, dentro de lo posible, el máximo rango y por ello las muestras recuperadas sean de la máxima precisión posible. En cualquier caso, el límite hasta el que se puede amplificar, está limitado por la alimentación del amplificador, en este caso 3.3 V o 5 V, asumiendo *rail-to-rail*.

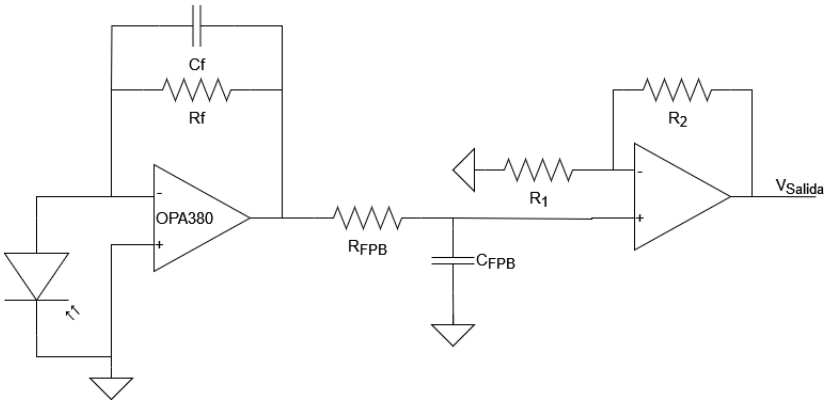


Figura 4.4: Diseño circuital del kit de expansión diseñado para la medida del SpO2.

4.3. Diseño *firmware* de la adquisición del SpO2

Para el diseño del *firmware* que gestionará esta nueva adquisición, se tienen que añadir, en primer lugar, los pines de la Figura 4.5. Se utiliza un pin analógico para captar la entrada del nivel de tensión proporcionado por el kit y dos digitales para activar o desactivar los LEDs. Dado que el PSoC 63 únicamente cuenta con un convertidor analógico-digital, se tiene que utilizar el mismo tanto para la medida de la tensión proporcionada por la batería y las muestras del *SpO*2. Para ello, este componente permite incluir un multiplexador a la entrada.

De manera adicional, se desarrolla una nueva librería, llamada “spo2.h”. Este archivo añade relativamente pocas nuevas funciones respecto de las que ya se habían utilizado para la adquisición del FECG. Por poner un ejemplo, se puede consultar que algunas de las variaciones son que, dado que se utiliza un convertidor de 12 bits de resolución en lugar de los 24 que se tenían para las muestras cardíacas, las muestras se dividen en grupos de 2 bytes en lugar de en grupos de 3. A esto hay que añadir un control del tiempo que ha estado encendido cada LED, para ir permutándolos cuando sea necesario.

La idea es disponer, en primer lugar de un *firmware* en el cual únicamente se capture la saturación de oxígeno. La principal funcionalidad de este será depurar los métodos utilizados tanto en el instrumento, como en el procesamiento externo mediante un PC, para asegurar que el cálculo de este parámetro se hace de manera adecuada. Una vez hecho esto, el siguiente paso es crear un nuevo *firmware* que aglutine las librerías tanto desarrolladas en el Capítulo 3, de manera que se puedan calcular obtener estos dos parámetros de forma simultánea.

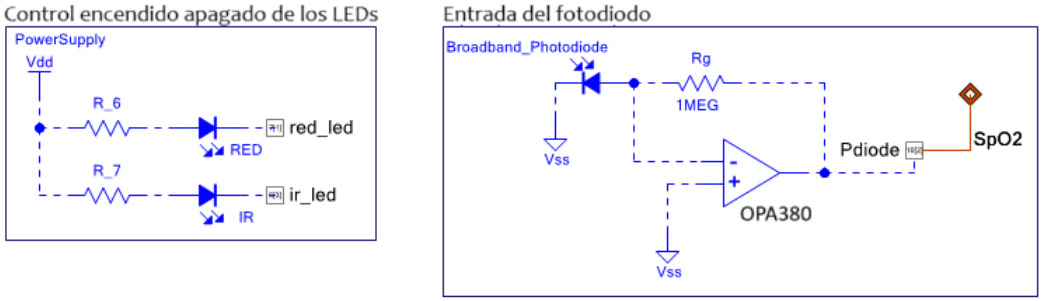


Figura 4.5: Esquemático de PSoC Creator con los pines de entrada/salida necesarios para la adquisición del SpO2. Se incluye también el diseño electrónico que se debe de incluir de manera externa previamente a los pines del PSoC.

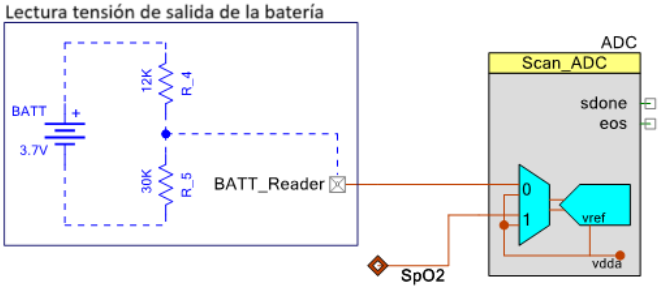


Figura 4.6: Esquemático de PSoC Creator con modificación hecha a la configuración del ADC para aceptar una segunda entrada que permita la adquisición del SpO2.

4.4. Obtención de la presión sanguínea a partir del SpO_2 y del ECG

A partir de la señal capturada durante la obtención de la saturación, se puede obtener también la presión sanguínea o BP (*Blood Pressure*). En la bibliografía existen varias técnicas para extraer este valor utilizando: un único PPG, dos PPGs o un PPG y un ECG [60]. En este trabajo, debido a que ya se ha incluido la parte relativa a la adquisición del ECG, se puede plantear la extracción de la BP de manera no invasiva a partir de dicho ECG y de un único PPG, habiéndose diseñado este último también en los apartados anteriores. Esto quiere decir que la BP se puede calcular en este proyecto sin necesidad de incluir *hardware* adicional, pudiéndose hacer los cálculos relativos en el equipo remoto una vez recuperadas las muestras del ECG y del SpO_2.

La técnica obtenida se conoce como *pulse transit time* (PTT) y, de manera resumida, consiste en extraer la BP a partir de la diferencia de tiempos entre el pico R del ECG y un punto de interés de la señal del PPG. Nótese que para que este cálculo sea correcto, ambas señales se deben de tomar dentro del mismo ciclo cardíaco. La relación que existe entre estas dos señales es que hay asociación entre el tiempo que tarda en llegar un pulso sanguíneo desde el corazón hasta las arterías de las extremidades y la BP. Cuando se produce un incremento en la presión sanguínea, el tiempo que la sangre tarda en alcanzar las extremidades se reduce, con lo cual se da una relación inversamente proporcional entre la diferencia de tiempo de los dos puntos medidos anteriormente y la BP [7].

Para la obtención de la presión a partir de la técnica elegida se pueden seguir los siguientes pasos [7, 61]:

1. Se hace realiza la captura del ECG y del SpO_2. Para el ECG se colocan los electrodos en el torso, mientras que para la saturación la medición se hace las extremidades, alejándolo, dentro de lo posible, lo máximo del lugar de medida del ECG.

2. Recuperadas las muestras de ambas medidas, se obtiene la diferencia de tiempo entre un punto de la señal del ECG y otro del PPG, tal y como se muestra en la Figura 4.7. En el caso del ECG, se suele escoger el pico R del complejo QRS. En el caso del PPG, la señal está compuesta por picos y valles. Se puede calcular la diferencia de tiempos desde el pico R hasta el punto más bajo del valle, el punto más alto del pico o el punto intermedio entre ambos.

3. Se establece una asociación entre la diferencia de tiempos medida, la que nos referiremos a partir de ahora como PTT, y la BP. En la bibliografía se pueden encontrar relaciones lineales:

$$BP = a \cdot PPT + b \tag{4.9}$$

Logarítmicas:

$$BP = a \cdot ln(PPT) + b \tag{4.10}$$

O modelos cuadráticos inversos:

$$BP = \frac{a}{PTT^2} + b \tag{4.11}$$

En cuanto a los coeficientes de calibración “a” y “b”, se deben de obtener para cada paciente/usuario a partir de un calibrado previo.

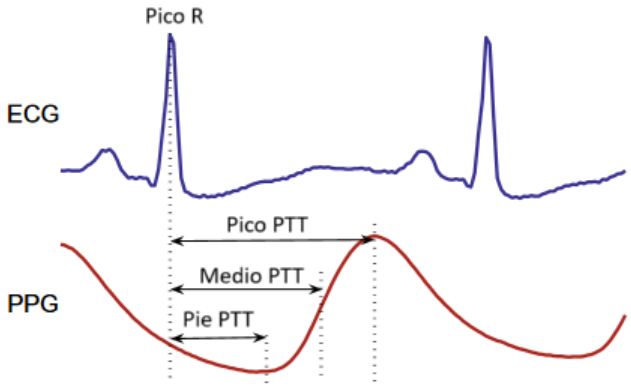


Figura 4.7: Obtención del BP a partir de la diferencia entre pico R del ECG y un punto de interés del PPG [7].

4.5. Gestión de la adquisición mediante un RTOS

Un sistema operativo de tiempo real (RTOS o *Real Time Operating System*) es un tipo de SO diseñado para gestionar recursos *hardware* en el entorno de, por ejemplo, un microcontrolador, de manera que se cumplan unas restricciones temporales estrictas. A diferencia de los sistemas operativos generales, un RTOS se enfoca en la ejecución de tareas dentro de un tiempo predecible y limitado, lo que es crucial en aplicaciones donde el tiempo de respuesta es crítico. En el contexto de este trabajo, es necesario que el controlador realice multitud de tareas de manera simultánea, como lo pueden ser: la adquisición del FECG, el *SpO2* y el nivel de batería, la gestión de la comunicación BLE, etc.

Resulta, por tanto, interesante disponer de este tipo de SO para la gestión de todas estas tareas, ya que al fin y al cabo, la gran mayoría de ellas se ejecutan sobre un único núcleo, con lo cual es necesario multiplexar el tiempo que cada tarea hace uso de los recursos del mismo.

En concreto, se ha utilizado FreeRTOS, para el cual PSoC 63 incluye extendido soporte, tanto en el propio PSoC Creator, como en la forma de proyecto de ejemplo oficiales. FreeRTOS es un *kernel* de sistema operativo en tiempo real de código abierto, diseñado para dispositivos embebidos y ofreciendo numerosas ventajas. Por un lado, cuenta con una API *(Application Programming Interface)* sencilla y bien documentada, además de una extensa base de usuarios, con lo cual no es complicado encontrar recursos adicionales creados por la comunidad. Por otro lado, tiene impacto mínimo sobre los recursos del sistema embebido sobre el que se utilice [62].

Figura 4.8: Logo FreeRTOS [8].

A continuación en esta sección, se va a tratar como se creado una implementación de la adquisición del porcentaje de saturación de oxígeno en sangre, a partir de la librerías de C y del diseño de PSoC Creator del apartado Diseño *firmware* de la adquisición del SpO2 de este mismo capítulo.

4.5.1. Adaptación del *firmware* para uso de FreeRTOS

Se han implementado un total de 3 tareas que hacen uso de las librerías C desarrolladas hasta el momento. Los esquemáticos de la Figura 4.6 y 4.6, así como los bloques para comunicaciones de la Figura 3.2, se reutilizan sin modificaciones, no siendo necesario para la implementación del SO añadir ningún elemento en el esquemático.

- **Gestion de eventos BLE**. La gestión de eventos mediante eventos se mantiene respecto del Capítulo 3. Sin embargo, es necesario añadir una nueva tarea que se encargue de gestionar dichos eventos a partir de interrupciones desde el bloque de comunicaciones BLE.
- **Adquisición de muestras (SpO_2)**. Almacena las muestras capturadas por el ADC de la Figura 4.6. Cuando se han alcanzado cierta cantidad de muestras guardadas, las transmite a la tarea para el envío de muestras.
- **Envío de muestras (SpO_2)**. Se encarga de gestionar el envío de fragmentos de los datos procesados al cliente.

Para enviar datos hacia o desde las tareas, se utilizan colas (*queues*) [63]. En esencia, estas son *buffers* FIFO (*First In, First Out*) que permiten sincronizar tareas, especialmente cuando una necesita esperar datos de otra tarea o rutina de servicio de interrupción (ISR), como ocurre en este caso. En la Figura 4.9 se puede consultar la comunicación entre las tareas de adquisición y envío. Tal y como se puede observar, la tarea para la adquisición de muestras cuenta con dos *buffers* de entrada. El primero de ellos, se emplea para la recepción de instrucción desde el cliente. Cuando la orden recibida coincida con la asociada al inicio de adquisición de muestras de la saturación, se inicializa el ADC. La segunda entrada está asociada a una rutina de interrupciones desde el ADC, la cual se activa cada vez que este finaliza la toma de una muestra. Cada vez que esto ocurra, se almacena dicho valor. Una vez que se han almacenado una cierta cantidad de valores, se envía al *buffer* de salida, el cual los transmitirá a la tarea para la gestión del envío BLE.

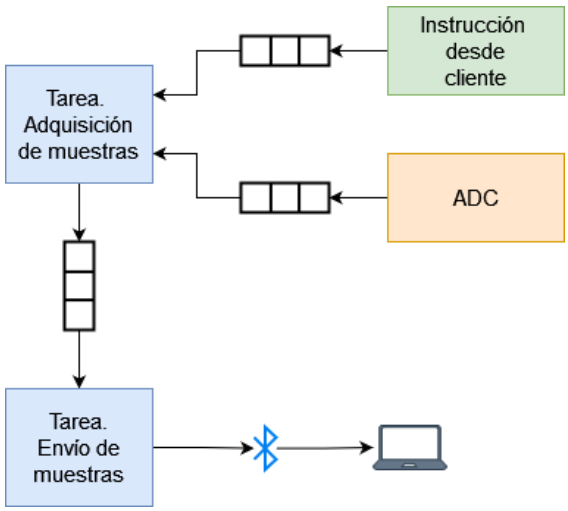


Figura 4.9: Diagrama de flujo entre las tareas de FreeRTOS para cálculo de SpO_2.

La tarea para gestión de eventos BLE se ha excluido de la imagen anterior, ya que no envía ni recibe valores directamente desde ninguna de las otras dos. En el Código 4.1, se puede consultar como se puede definir una de estas tareas.

```
.
.
.
void TASK_send_spo2_samples()
{
        // Array of data converted to 1 byte
        static uint8_t data_converted_1byte[
            DATA_BUFFER_SIZE_1BYTE_SPO2];

        for(;;)
        {
                if (xQueueReceive(send_spo2_samples_queue,
                    data_converted_1byte, portMAX_DELAY) == pdPASS)
                {
                        // Sending of the adapted data to the
                            client via Bluetooth packages
                        send_ble(data_converted_1byte,
                            DATA_BUFFER_SIZE_1BYTE_SPO2,
                            get_fragment_size());
                }
        }
}

.
.
.

int main(void)
{
        .
        .
        .
        // Create queues to send from and into the tasks
        send_spo2_samples_queue = xQueueCreate(
            DATA_BUFFER_SIZE_1BYTE_SPO2, sizeof(uint8_t));
        // Create tasks
        rtos_api_result |= xTaskCreate(TASK_send_spo2_samples,
            TASK_SEND_SPO2_NAME, TASK_SEND_SPO2_SIZE, NULL,
            TASK_SEND_SPO2_PRIORITY, NULL);

        if (pdPASS == rtos_api_result)
        {
                // Start the RTOS scheduler. This function should
                    never return
                vTaskStartScheduler();
        }
        .
        .
        .
}
```

Código 4.1: Tarea FreeRTOS para envío de muestras mediante BLE

Capítulo 5

Control y procesado desde dispositivo remoto

Para lograr el control del instrumento desde el cliente, es decir, desde en este caso un ordenador o portátil, se han creado varios *scripts* en el lenguaje de programación Python. En este capítulo se comenta dicho programa y las tareas que realiza.

5.1. Planteamiento del control y recuperación de las muestras

Se plantea emplear dos programas de Python, trabajando en paralelo, para realiza el control del instrumento y la recuperación y procesado de las muestras. La idea es que el primero de ellos transmita las instrucciones correspondientes al instrumento desarrollado en los apartados anteriores y reciba las muestras desde dicho dispositivo. Mientras, el segundo programa se encarga del procesado de dichas muestras y de su representación de cara al usuario.

Con la intención de comunicar ambos programas, a la par que se almacenan las muestras para posibles estudios a futuro, se genera un archivo de texto plano (.txt). Por una parte, el programa encargado de la recepción de la muestras irá copiando en dicho archivo los valores obtenidos. Por otra parte, el segundo programa irá extrayendo lotes de muestras del archivo, procesándolas y representando el resultado. La idea es que este proceso ocurra en tiempo real. Es decir, que a la par que se capturan las muestras, se vayan visualizando en pantalla.

En la Figura 5.1 se encontrar un diagrama que muestras la interacción entre estos dos programas de Python y el resto del proyecto.

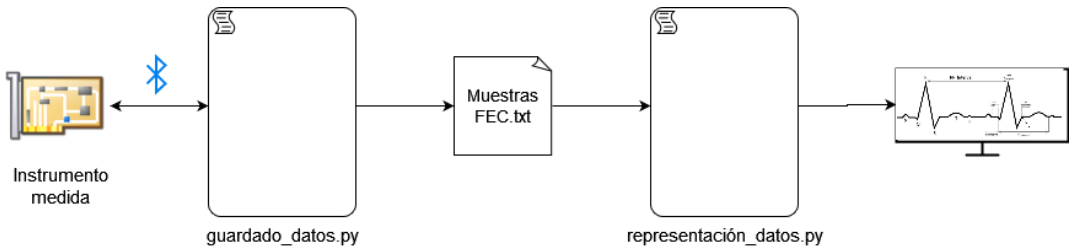


Figura 5.1: Diagrama de la interacción entre los dos proyectos de Python concurrentes.

5.2. Envío de ordenes y recepción de muestras

En primer lugar, se va a tratar en esta sección el primer *script* de la Figura 5.1, utilizado para el envío de ordenes hacia el instrumento y la recepción de los resultados transmitidos por el mismo como respuesta. La tareas que realiza este programa son, por orden de ejecución y asumiendo que el paso anterior de cada uno se ha ejecutado de manera correcta:

- Conexión a un dispositivo BLE, previamente conocida la MAC (*Media Access Control*) del mismo.
- Envío de la instrucción a dicho instrumento. Esta instrucción indica que medida se va a realizar de entre las disponibles (FECG, SpO_2, etc).
- Creación del archivo, en formato de texto plano, donde se van a almacenar los resultados. A dicho archivo se le añaden, previamente a recibir las muestras, unas cabeceras que indican información como el número de canales usados para la medida o la resolución (en bits) de las muestras.
- Se genera un segundo archivo de texto plano. La información que contiene este nuevo archivo es el camino absoluto usado para el documento que va a contener las muestras generado en el paso anterior. El significado de este paso es que el programa encargado del procesado de las muestras leerá la información contenida en él para saber el nombre y la localización del documento donde se están almacenando las muestras y que pueda acceder a ellas en tiempo real durante la representación.
- Como último paso, se inicia la recepción de las muestras desde el instrumento. Se puede configurar o bien que la comunicación se de hasta que se hayan capturado una cierta cantidad de muestras, o bien, se puede indicar que la recepción va a ocurrir *ad infinitum*.

5.2.1. Bucle principal del programa

En el Código 5.1, se puede encontrar un fragmento del *script* que permite realizar las tareas descritas anteriormente. En primer lugar, se realiza un escaneo de todos los dispositivos BLE disponibles, tras lo cual se trata de recuperar las características del dispositivo de interés, en este caso, el instrumento. Esto se realiza a modo de depuración, con la intención de comprobar que el módulo BLE está operativo y que las características se han descrito adecuadamente en el lado *firmware*.

Hecha esta primera comprobación, se envía una orden de escritura, con un mensaje que indica al programa que se de inicio a la adquisición de muestras. Tras ello, el *script* genera los dos documentos en texto plano mencionados anteriormente, se suscribe a la característica tipo “Notify” y comienza a almacenar los valores recibidos desde esta. En este ejemplo no se ha seleccionado un límite a las muestras que se quieren capturar.

La desconexión se realiza dentro de una estructura “finally”. Lo que esto implica es que aun cuando se pueda producir un error en cualquiera de los pasos anteriores, el *script* se asegura de que se realice una desconexión del instrumento. Así mismo, también se asegura de que se cierre el bucle de eventos donde se ejecutaron las tareas asíncronas.

```
# Handling of BLE communication
# ==============================================
try:
        # Creates loop for async tasks
        loop = asyncio.new_event_loop()

        # Create BLE client
        client = bleak.BleakClient(mac_address_bb, loop=loop)

        # Scan for near devices
        loop.run_until_complete(scan_for_devices())

        # Connect to the device
        print("Trying to connect to BLE device...")
        loop.run_until_complete(connect_and_pair(client, False))

        # Get the services and characteristics UUID (uncomment only
            if needed)
        loop.run_until_complete(get_services_and_characteristic(
           client))

        # Send start message
        print("Sending start message to device...")
        loop.run_until_complete(send_data_ble_device(client,
           UUID_ControllerOrders, order))

        # Creation of file where the samples will be stored
        last_path = generate_path_to_fecg_txt(filename)
        create_new_fecg_txt(last_path)
        add_headers_to_fecg_txt(last_path, num_channels=
           num_channels)

        # Create file for the sharing of information between files
        create_sharing_data_file(last_path)

        # Read data
        print("Starting capturing and saving of data...")
        loop.run_until_complete(read_data_ble_device(client,
           UUID_DataAcquisition, max_data_count=0))
        print("Data recovering finish")

except Exception as e:
        print(f"[Error] - Control and Recovery Main Script - {e}")

finally:
        # Disconnect from the BLE device
        print("Disconnecting from BLE device")
        loop.run_until_complete(disconnect_ble_device(client))
        loop.close()
```

Código 5.1: Bucle principal del programa de Python utilizado para la comunicación con el instrumento.

5.2.2. Librería para la conexión al instrumento

Para lograr la comunicación desde el PC se utiliza, principalmente, la librería "Bleak" de Python. Bleak es una biblioteca que proporciona una API para interactuar con dispositivos Bluetooth de baja energía en plataformas Windows a través de la API de este último [64].

A la hora de emplear esta librería, se ha creado una serie de métodos personalizados para conectarse/desconectarse a un dispositivo, recuperar la información sobre los servicios y características con las que cuenta y, finalmente, suscribirse a una característica para la recepción de datos o escribir en otra para el envío de las ordenes.

En el Código 5.2 se muestra a modo de ejemplo el método para suscribirse a la característica tipo "Notify" que se comentaba en Configuración BLE usada en PSoC 63. De manera resumida, en primer lugar se comprueba si hay una conexión existente con el dispositivo remoto, caso de que exista se subscribirá a la característica indicada por su UUID correspondiente. Para este ejemplo, se ha hecho que una vez que se haya recibido una cierta cantidad de paquetes de información, se de una desuscripción de la característica anterior y que el PC se desconecte del instrumento. Nótese que se ha diseñado de esta manera para facilitar la depuración del instrumento desde varios dispositivos de manera simultanea, en una versión final no sería ideal tener que reconectar el dispositivo si lo que se quiere es realizar varias medidas consecutivas.

En la línea 27 se utiliza un método "callback", llamado "notify_callback", al igual que ocurría en la sección Control de eventos BLE, esto no es más que un método que se invoca cada vez que se den cambios en la característica suscrita. En esta ocasión, este método recupera el paquete de información transmitido, estando simplemente este compuesto por una trama de varios bytes, y almacenándose esté en el archivo de texto plano correspondiente. Se cuenta también cuantos de estos paquetes se han recibido hasta el momento. En Código 5.3 se puede consultar este fragmento del programa.

```
1  async def read_data_ble_device(client, characteristic_uuid: str,
       max_data_count: int | float = 0):
2  """
3  Read bytes transmitted from the BLE device.
4
5  Inputs:
6  - client: Bleak client
7  - characteristic_uuid: UUID of the BLE characteristic to subscribe
       to
8  - max_data_count: Count of data expected to be received. If the
       value is equal or less than 0, the task runs forever
9  Returns:
10 - None
11 """
12
13 global _data_count
14 _data_count = 0
15
16 try:
17         # Checks connection
18         connected = await client.is_connected()
19
```

```
        if connected:
                print("Device Connected")
        else:
                raise Exception("Device not connected when
                    attempting to subscribe to characteristic")

        # Subscribe to notifications and define callback
        print("Subscribing to notification...")
        await client.start_notify(characteristic_uuid,
            notify_callback)
        print("Subscribed")

        # Keep the loop running until the desired amount of data is
            received
        if max_data_count > 0:
        while _data_count < max_data_count:
                await asyncio.sleep(0.1)  # Sleep to avoid busy
                    waiting

        # Keeps the loop running forever
        else:
                print("The program will read data forever")
                while True:
                        await asyncio.sleep(0.1)

except Exception as e:
        print(f"Error during subscription and receiving: {e}")

finally:
        # Checks connection
        connected = await client.is_connected()
        if not connected:
                print("Device already disconnected")
        else:
        # Unsubscribe from notifications when the loop is done
                await client.stop_notify(characteristic_uuid)
```

Código 5.2: Método de Python para subscribirse y leer de una característica BLE tipo "Notify".

```
async def notify_callback(sender: int, data: bytearray):
"""
Will be called when the BLE device notifies that there is data
ready. Saves the received samples into a .txt file
"""

global _data_count

# Save received data package in .txt file
file_path = get_last_generated_path()
data_converted = fecg_adapt_ble_samples_to_24bits([data])
add_data_set_to_fecg_txt(file_path, data_converted, num_channel=
    num_channels)

# Counts that one package of data has been received
_data_count += 1

```

```
17 print(f"Received notification ({_data_count} total): {data}")
```

Código 5.3: Método *callback* invocado cada vez que se actualiza la característica *notify*.

Como se puede observar en los dos fragmentos de código anteriores, los métodos relacionados con la comunicación Bluetooth son del tipo "async". En Python, los métodos "async" o asíncronos permiten realizar operaciones de manera concurrente, lo que significa que pueden ejecutar tareas en segundo plano y continuar con otras operaciones sin bloquear el hilo principal del programa [65].

También se puede observar que se hace uso de la palabra reservada "await". Esta se utiliza en el contexto de los métodos asíncronos para pausar la ejecución de la función hasta que la operación asíncrona que se está esperando se complete. Esto permite que otras tareas asíncronas se ejecuten mientras se espera, en lugar de bloquear la ejecución del hilo principal.

La biblioteca "Bleak" utiliza estos métodos asíncronos para manejar la comunicación Bluetooth Low Energy debido a la naturaleza inherentemente asíncrona de estas tareas. Las operaciones de BLE, como pueden ser: descubrir dispositivos, conectarse, leer/escribir características y recibir notificaciones, son no bloqueantes y pueden demorarse por factores externos. Los métodos asíncronos permiten realizar estas tareas sin bloquear el hilo principal, mejorando la capacidad de respuesta y la eficiencia del programa al permitir la concurrencia y simplificar el manejo de errores. Esto asegura una gestión eficiente de múltiples dispositivos BLE y operaciones concurrentes, manteniendo la aplicación receptiva y escalable.

5.2.3. Almacenamiento de las muestras

Con la intención de poder consultar las muestra adquiridas en estudios posteriores, los resultados obtenidos en milivóltios se almacenan en un archivo de texto plano (.txt). En el Código 5.4 se puede consultar el método de Python utilizado a este fin. En la primera columna se almacena el indicador del número de muestra, mientras que la segunda se pueda usar para indicar *offset* en la medida. A partir de la tercera columna, esta incluida, se almacenan las muestras. La tercera columna contiene las del primer canal, la cuarta las del segundo, etc.

```
1  def add_data_set_to_fecg_txt(path: str, data: list[int | float],
       num_channel: int = 1):
2  """
3  Adds data rows to an already existing .txt file according to the
       Biosignal data file format. If multiple channels
4  where use, the samples of each channel taken in the same time
       instant will be stored in the same row and in
5  different columns for each channel (first column for the first CH1,
        etc.)
6
7  Inputs:
8  - path: Absolute path to the .txt file where the samples will be
       added
9  - data: List of samples to copy to each row of the file. Each
       sample is copied in a different row
10 - num_channel: Number of channels use during the acquisition
```

```
Returns:
- None
"""

        global row_counter
        last_index = 0

        try:
                print("Adding data set")
                # Create data row
                with open(path, "a") as file:
                        while last_index < len(data):
                                data_row = []
                                for i in range(num_channel):
                                        data_row.append(data[
                                            last_index])
                                        last_index = last_index + 1

                        # Save row in the file
                        data_row_str = "\t".join(map(str, data_row)
                            )  # Convert the samples into str
                            separated by tabs
                        row = str(row_counter) + "\t" + "0" + "\t"
                            + data_row_str
                        row_counter = row_counter + 1
                        file.write(row + '\n')

        except Exception as e:
                print(f"[Error] - Synthetic ECG data set addition -
                    {e}")
```

Código 5.4: Método de Python para almacenar las muestras en un archivo de texto plano.

5.3. Procesado y representación

En este apartado se puede consultar como se recuperan las muestras almacenadas desde el instrumento y como se procesan y representan en un nuevo programa, funcionando en paralelo al utilizado para el almacenamiento de los resultados. Las tareas que realiza este nuevo programa son:

- Lectura de un set de muestras almacenadas en el fichero de texto plano correspondiente. Esto ocurre de manera simultanea a que el programa del apartado Envío de ordenes y recepción de muestras va escribiendo nuevas muestras.
- Conversión del valor proporcionado por el ADC a el correspondiente en milivóltios.
- Filtrado del fragmento de señal recuperado.
- Detección automática de picos R de la señal del electrocardiograma.
- Representación gráfica de la señal filtrada y de los picos R detectados en el paso anterior.

El proceso descrito se repetirá de manera constante siempre y cuando queden muestras sin utilizar en el archivo de texto.

5.3.1. Bucle principal del programa

En este apartado se puede encontrar el *script* principal de programa de Python que realiza las tareas anteriormente descritas.

En primer lugar, se lee el archivo de texto que contiene la dirección absoluta al documento sobre el que se están grabando las muestras. Tras lo cual, se recuperan también los parámetros de la adquisición, es decir, número de canales, resolución en bits de los mismos, la frecuencia de muestreo, etc. Algunos de estos parámetros se recuperan de la cabecera del documento de las muestras, mientras que las relativas a la configuración del ADC son previamente conocidas y ya se tienen almacenadas en el programa.

Hecho esto, se produce un bucle sin fin, en el cual se realizan las tareas descritas en el apartado anterior.

```
try:
        # Read the absolute path to the samples storage file
        path = read_sharing_data_file()

        # Recovery of the parameters of the acquisition
        fs_channel, adc_resolution, num_channels, date =
            get_headers(path)

        if test == "FECG":
                params = get_parameters(test)
                total_num_packages = params["total_num_packages"]
                adc_top_limit = params["adc_top_limit"]
                adc_bottom_limit = params["adc_bottom_limit"]
                amplification = params["amplification"]

        else:
                print(f"Unrecognized test {test}")
                raise Exception("[Error] - Unrecognized selected
                    test")

except Exception as e:
        print(f"[Error] - Acquisition parameters recovery - {e}")

try:
        # While there is data to be read in the file, this scripts
            will keep running
        while True:

                # Recovery and adaptation of samples from .txt file
                # ================================================
                raw_samples = get_samples_from_txt(path,
                        start_time=start_time,
                        end_time=end_time,
                        fs=fs_channel,
                        num_channels=num_channels)

                milivolts_samples = fecg_samples_to_milivots_bb(
                    raw_samples,
                        adc_resolution,
                        adc_top_limit,
```

```
                adc_bottom_limit,
                amplification)

        # WAVELET DENOISING
        # ================================================
        data_dwt_dn_set = []

        # Create list of list with the denoising of the samples of each channel
        for i in range(len(milivolts_samples)):
                data_dwt_dn = procesado_wavelet_noise(milivolts_samples[i],
                        fs_channel,
                        level_den,
                        metodo_den_tipo,
                        tipo_reescalado,
                        tipo_denoising,
                        tipo_proc)
                        data_dwt_dn_set.append(data_dwt_dn.tolist())

        # CLUSTERING
        # ================================================
        clusters_time_r_peak_set = []
        cluster_amp_r_peak_set = []

        # Create list of list with the clustering of the samples of each channel
        for i in range(len(data_dwt_dn_set)):
                clusters_time_r_peak, cluster_amp_r_peak, clusters_amp_diff, cluster_select = \
                        (deteccion_QRS_clustering(
                        data_dwt_dn_set[i],
                        num_clusters,
                        NUM_MUESTRAS_BUSQUEDA,
                        fs_channel))

        # That method returns 2 cluster, one with the noise and another with the ECG points.
        # We take only the ECG points (cluster 2 or list_name[1])
        clusters_time_r_peak_set.append(clusters_time_r_peak[1])
        cluster_amp_r_peak_set.append(cluster_amp_r_peak[1])

        # Data representation
        # ================================================
        # Time shift of the cluster points
        for i in range(len(clusters_time_r_peak_set)):
                clusters_time_r_peak_set[i] = [x + time_aux for x in clusters_time_r_peak_set[i]]

        plot_n_channels_dynamically_v2(samples=data_dwt_dn_set,
                cluster_amp_r_peak=cluster_amp_r_peak_set,
                clusters_time_r_peak=clusters_time_r_peak_set,
```

```
                        fs=fs_channel,
                        start_time=start_time)

                # Time displacement for the next set of data
                # ============================================
                time_aux = time_aux + time_shift  # Time
                    displacement in the cluster points in the next
                    loop
                start_time = start_time + time_shift
                end_time = end_time + time_shift

except Exception as e:
        print(f"[Error] - Data Representation Main Loop - {e}")
```

Código 5.5: Bucle principal del programa de Python utilizado para procesado y representación de las muestras.

5.3.2. Conversión de las muestras a milivóltios

Las muestras que se leen desde el archivo, indican el valor del ADC que se recuperó. Queda, por tanto, hacer una conversión de este valor al valor real en milivóltios a partir de los parámetros del ADC utilizado. En este caso, esto se realiza a partir de los parámetros de los ADCs embebidos en el ADS1299, para lo cual se puede emplear la Ecuación 5.1.

$$Amplitud(mV) = (\frac{Muestra_{ADC} \cdot V^{+}}{2^{N} - 1} + V^{-}) \cdot 1000 \tag{5.1}$$

Los parámetros de esta expresión son:

- **Muestra**$_{ADC}$. Valor digitalizado que finalmente se recupera de los ADCs embebidos en el ADS1299.
- $\mathbf{V^{+}}$ **y** $\mathbf{V^{-}}$. Respectivamente, el límite superior e inferior del ADC en voltios.
- **N**. Bits de resolución del ADC, 24 en este caso.

Finalmente se multiplica por 1000 para convertir el resultado de voltios a milivoltios.

5.3.3. Filtrado y detección de picos R

Para el filtrado de la señal, se utilizan técnicas de *Discrete Wavelet Transform* (DWT) con la intención de eliminar el ruido presente en las señales cardíacas. De manera muy resumida, esta técnica permite descomponer una señal en varios coeficientes. Si se identifican los coeficientes relativos al ruido, estos se pueden eliminar, consiguiendo, por lo tanto, filtrar la señal.

Por otra parte, la detección de picos R se realiza mediante una técnica de *clustering* o agrupación. Esto consiste en detectar todos los picos presentes en la señal, siendo considerado como pico un mínimo seguido de un máximo. Tras ello, se calcula la diferencia entre el mínimo y el máximo. Según el valor de dicha diferencia, se afirmará que el pico es, o bien, parte del ruido o que, por el contrario, se trata de un pico R de la señal cardíaca. Es decir, a partir de la diferencia, se incluirá el máximo en un *cluster* o conjunto o el otro.

En este trabajo se han utilizado librerías de Python que implementan el filtrado de las señales mediante la técnica DWT y la detección automática de los picos R. Dichas librerías se proporcionado ya desarrolladas y no han sido, por tanto, implementadas en este trabajo. La información relativa a dichas librerías se puede encontrar en: [14, 66].

5.4. Comprobación del funcionamiento del los programas junto con el *firmware*

Se desea comprobar que los *scripts* de Python que se han desarrollado a lo largo de este capítulo funcionan de manera correcta, a la par que se tratar de depurar también el funcionamiento del *firmware* para adquisición del ECG tratado en el capítulo Capítulo 3.

Para lograrlo, se utiliza el kit de prototipado CY8CPROTO-063-BLE, el cual se programará con el *firmware* del Capítulo 3. Utilizando un generador de señales, se configura una señal cardíaca artificial, la cual se muestreará desde el kit mencionado. La idea para depurar el proyecto, es comprobar que los programas de Python se logran comunicar con la placa y que las muestras se reciben, procesan y muestrean de manera correcta.

En la Figura 5.2 se puede consultar el resultado obtenido, donde se han tratado las muestras como si se dispusiese de dos canales (aunque sólo se ha empleado un único generador). Se puede observar que los máximos que representan al pico R de la señal cardíaca, se logran identificar y se indican mediante un punto rojo.

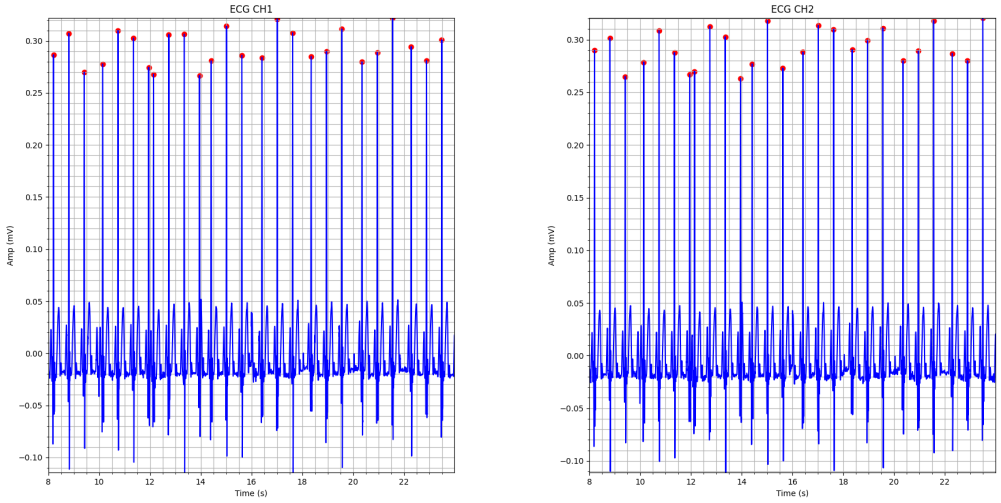


Figura 5.2: Señal de ECG creada a partir de un generador de señales, simulando 2 canales, capturada y procesada mediante los programas de Python desarrollados en este capítulo.

Capítulo 6

Conclusiones y líneas de trabajo futuras

6.1. Objetivos logrados

En este apartado se puede consultar que objetivos se han logrado cumplir de los que se plantearon en la sección Objetivos a cumplir del Capítulo 1, siendo estos:

- Se ha diseñado un instrumento de reducido tamaño capaz de realizar la adquisición tanto de la señal cardíaca mediante la técnica de las 12 derivaciones como de la señal cardíaca fetal.
- Se ha creado un *script* mediante el lenguaje de programación Python, el cual permite la comunicación con el instrumento anterior tanto para lograr el envío de instrucciones a este, como para la recepción de las muestras. Así mismo, este programa implementa cierto procesado de las muestras recibidas y el almacenamiento de las mismas en formato CSV.
- Se ha desarrollado el *firmware* necesario para la adquisición del FECG, incluyendo el control del *frontend* que toma las muestras y la gestión de la comunicación Bluetooth.
- Se ha implementado también el *firmware* que permite la obtención del SpO_2. Existen dos versiones de este *software*. Por un lado, con el control de las tareas a realizar planificadas por de manera manual, en lo que se conoce como *bare metal*, mientras que por el otro, se ha incluido el control de las tareas de manera más automatizada mediante el sistema operativo de tiempo real, FreeRTOS.
- Se ha planteado y diseñado la electrónica externa adicional, que hay que añadir para poder adquirir el SpO_2. Esta electrónica es un ejemplo de un posible kit de expansión que permite, de manera modular, añadir funcionalidades al dispositivo.

En conclusión, se han cumplido los objetivos principales que se plateaban en un comienzo para este proyecto.

6.2. Líneas de trabajo futuras

Este subapartado se plantean unas líneas de trabajo claras de cara a posibles iteraciones futuras sobre este proyecto, incluyendo todo aquello que no se ha podido llegar a desarrollar debido a las limitaciones de tiempo para esta Trabajo de Fin de Máster.

Cabe destacar que, aunque se encargo la fabricación de la PCB diseñada para el instrumento, esta no fue entrega antes de la finalización de este trabajo, con lo cual, una parte importante del trabajo pendiente viene condicionada a este hecho.

6.2.1. Instrumento

- **Depuración y verificación del diseño físico**. Hay que realizar pruebas sobre si existe algún error, a nivel electrónico, no contemplado sobre la primera versión diseñada del instrumento. Esto incluye, por ejemplo, la verificación del cargador de la batería, de los niveles de alimentación de los integrados, etc.
- **Encapsulado**. Es necesario plantear un encapsulado dentro del cual se pueda integrar el instrumento, con la intención de que este esté protegido de factores externos y que sea cómodo el hecho portarlo y emplearlo por parte de un usuario.

6.2.2. *Firmware*

- **Depuración de la librerías para comunicación con ADS1299**. Debido a no disponer del *frontend* encargado de la adquisición de las señales fetales, queda pendiente de comprobar si existen errores en la librería que gestiona la comunicación entre este y el PSoC 63.
- ***Firmware* para almacenamiento en microSD**. Se tiene que modificar el *firmware* que gestiona la adquisición, para añadir el almacenamiento de un respaldo de las muestras tomadas en la tarjeta microSD incluida.
- **Gestión de la adquisición del ECG y del SpO_2 de manera simultánea**. El funcionamiento del *firmware* para la adquisición de estos dos parámetros se ha verificado tomando en cada momento únicamente uno de ellos. Hay que aglutinar ambos, mediante el uso de FreeRTOS, en un único *firmware* que realice la adquisición simultánea.

6.2.3. Adquisición del SpO_2

- **PCB para kit de expansión**. Aunque se ha diseñado la circuitería necesaria para la adquisición del SpO_2, queda pendiente de diseñar una PCB que aglutine esta electrónica.

6.2.4. *Script* de Python

- **Diseño de interfaz gráfica**. Para facilitar y expandir las funcionalidades del *script* usado desde el lado del cliente, se tiene que añadir una interfaz gráfica, que permita el manejo del instrumento de manera más intuitiva y cómoda.
- **Cálculo de la presión sanguínea**. Tal y como se explicó en el Capítulo 4, a partir de las señales obtenidas en este proyecto se puede estimar la presión sanguínea. Para ello, queda por desarrollar una librería de Python que permita el cálculo de la BP a partir de las muestras del ECG y del SpO_2.

Bibliografía

[1] A. Gacek and W. Pedrycz, *ECG signal processing, classification and interpretation: a comprehensive framework of computational intelligence*. Springer Science & Business Media, 2011.

[2] N. Marchon, G. Naik, and R. Pai, "Ecg electrode configuration to extract real time fecg signals," *Procedia Computer Science*, vol. 125, pp. 501–508, 2018.

[3] Texas Instruments, "ADS1299-x Low-Noise, 4-, 6-, 8-Channel, 24-Bit, Analog-to-Digital Converter for EEG and Biopotential Measurements." ADS1299 Datasheet https://www.ti.com/lit/ds/symlink/ads1299.pdf?ts=1712921675119&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FADS1299%253Futm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Dasc-null-null-GPN_EN-cpc-pf-google-wwe%2526utm_content%253DADS1299%2526ds_k%253DADS1299%2526DCM%253Dyes%2526gad_source%253D1%2526gclid%253DCjwKCAjwt-OwBhBnEiwAgwzrUsOFn9c3CtUcATakAN7BllhOkeCNAAvVtd8GmgpZxjnk7dNOS_DPvhoCESgQAvD_BwE%2526gclsrc%253Daw.ds, Ene 2017. Rev. C.

[4] Martin Woolley, "Bluetooth Core Specification Version 5.2 Feature Overview." Bluetooth 5.2 Core Specification https://www.bluetooth.com/wp-content/uploads/2020/01/Bluetooth_5.2_Feature_Overview.pdf, Dic 2020. Ver 1.0.1.

[5] Analog Devices, "Guidelines for SpO2 Measurement." https://www.analog.com/en/resources/technical-articles/guidelines-for-spo2-measurement--maxim-integrated.html. [Online; Accedido: 29 de Mayo, 2024].

[6] V. Toral, A. García, F. J. Romero, D. P. Morales, E. Castillo, L. Parrilla, F. M. Gómez-Campos, A. Morillas, and A. Sánchez, "Wearable system for biosignal acquisition and monitoring based on reconfigurable technologies," *Sensors*, vol. 19, no. 7, p. 1590, 2019.

[7] Y. Choi, Q. Zhang, and S. Ko, "Noninvasive cuffless blood pressure estimation using pulse transit time and hilbert–huang transform," *Computers & Electrical Engineering*, vol. 39, no. 1, pp. 103–111, 2013.

[8] Wikipedia, "Archivo:Logo freeRTOS.png." https://es.m.wikipedia.org/wiki/Archivo:Logo_freeRTOS.png. (Accedido 3 de Junio, 2024).

[9] Infineon, "CYBLE-416045-02. EZ-BLE™ Creator Module." CYBLE-416045-02 Datasheet https://www.infineon.com/dgdl/Infineon-CYBLE-416045-02-DataSheet-v02_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee63f136e57, Feb 2019. Rev. A.

[10] J. D. Bronzino and D. R. Peterson, *Biomedical engineering fundamentals*. CRC press, 2014.

[11] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ecg analysis," *Biomedical Signal Processing and Control*, vol. 43, pp. 216–235, 2018.

[12] P. R. Rijnbeek, J. A. Kors, and M. Witsenburg, "Minimum bandwidth requirements for recording of pediatric electrocardiograms," *Circulation*, vol. 104, no. 25, pp. 3087–3090, 2001.

[13] D. Davis, P. Turner, L. Foster-Roesler, and D. Lynam, *Quick and accurate 12-lead ECG interpretation*. Philadelphia: Lippincott Williams & Wilkins, fourth edition. ed., 2005.

[14] E. Castillo, D. P. Morales, A. García, L. Parrilla, V. U. Ruiz, and J. A. Álvarez-Bermejo, "A clustering-based method for single-channel fetal heart rate monitoring," *PLoS One*, vol. 13, no. 6, p. e0199308, 2018.

[15] N. Marchon and G. Naik, "Electrode positioning for monitoring fetal ecg: A review," in *2015 International Conference on Information Processing (ICIP)*, pp. 5–10, IEEE, 2015.

[16] R. Martinek, R. Kahankova, H. Nazeran, J. Konecny, J. Jezewski, P. Janku, P. Bilik, J. Zidek, J. Nedoma, and M. Fajkus, "Non-invasive fetal monitoring: A maternal surface ecg electrode placement-based novel approach for optimization of adaptive filter control parameters using the lms and rls algorithms," *Sensors*, vol. 17, no. 5, p. 1154, 2017.

[17] M. Anisha, S. Kumar, E. E. Nithila, and M. Benisha, "Detection of fetal cardiac anomaly from composite abdominal electrocardiogram," *Biomedical Signal Processing and Control*, vol. 65, p. 102308, 2021.

[18] Hospital Clínic de Barcelona, "Pruebas y seguimiento durante el Embarazo." https://www.clinicbarcelona.org/asistencia/enfermedades/embarazo-y-parto/pruebas-y-seguimiento. [Online; Accedido: 23 de Abril, 2024].

[19] Biorithm, "FEMOM." https://www.bio-rithm.com/. [Online; Accedido: 23 de Abril, 2024].

[20] Mindchild, "About MERIDIAN M110 Fetal Monitor." https://www.mindchild.com/meridian-monitor.html. [Online; Accedido: 23 de Abril, 2024].

[21] Nuvo, "A paradigm shift in pregnancy monitoring." https://www.nuvocares.com/. [Online; Accedido: 23 de Abril, 2024].

[22] KiCad Development Team, "KiCad Docs: Introduction." https://docs.kicad.org/8.0/en/introduction/introduction.html. (Accedido 11 de Abril, 2024).

[23] Analog Devices, "LTspice: Fast • Free • Unlimited." https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html. [Online; Accedido: 11 de Abril, 2024].

[24] Infineon, "PSoC™ Creator." https://www.infineon.com/cms/en/design-support/tools/sdk/psoc-software/psoc-creator/. [Online; Accedido: 11 de Abril, 2024].

[25] Python Software Foundation, "Python." https://www.python.org/doc/. [Online; Accedido: 11 de Abril, 2024].

[26] Git, "Git –distributed-is-the-new-centralized." https://git-scm.com/. [Online; Accedido: 26 de Junio, 2024].

[27] GitHub, "GitHub Main Page." https://github.com/. [Online; Accedido: 26 de Junio, 2024].

[28] Infineon, "PSoC™ 63 MCU with Bluetooth® LE." PSoC™ 6 MCU: CY8C63x6, CY8C63x7 Datasheet https://www.infineon.com/dgdl/Infineon-PSoC_6_MCU_PSoC_63_with_BLE_Datasheet_Programmable_System-on-Chip_%28PSoC%29-DataSheet-v16_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee4efe46c37&utm_source=cypress&utm_medium=referral&utm_campaign=202110_globe_en_all_integration-datasheet&redirId=DS89, Abr 2023. Rev. R.

[29] Infineon, "PSoC™ 4 MCU with AIROC™ Bluetooth® LE." CY8C42xx-BL Datasheet https://www.infineon.com/dgdl/Infineon-PSoC_4_4200_BLE_Family_Datasheet_Programmable_System-on-Chip-DataSheet-v03_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee5f2a26ddd, Mar 2023. Rev. B.

[30] Martin Woolley, "Bluetooth® Core Specification Version 5.0 Feature Enhancements." Bluetooth 5.0 Core Specification https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf, Sep 2021. Ver 1.1.0.

[31] P. Bulić, G. Kojek, and A. Biasizzo, "Data transmission efficiency in bluetooth low energy versions," *Sensors*, vol. 19, no. 17, p. 3746, 2019.

[32] Infineon, "PSoC/PRoC BLE Crystal Oscillator Selection and Tuning Techniques." AN95089 https://www.mouser.es/datasheet/2/137/FC_255_en-1649561.pdf, Oct 2020. Rev. E.

[33] Epson Timing, "FC - 135R FC - 135 / FC - 255 kHz RANGE CRYSTAL UNIT." FC - 135R FC - 135 / FC - 255 Datasheet https://www.mouser.es/datasheet/2/137/FC_255_en-1649561.pdf.

[34] B. Young, "New standards for ecg equipment," *Journal of electrocardiology*, vol. 57, pp. S1–S4, 2019.

[35] D. Manolakis and J. G. Proakis, *Tratamiento Digital de Senales (4A. Ed)*. Mexico: Pearson,, 2003.

[36] F. Cenni, S. Mir, and L. Rufer, "Behavioral modeling and simulation of a chemical sensor with its microelectronics front-end interface," in *2009 3rd International Workshop on Advances in sensors and Interfaces*, pp. 92–97, IEEE, 2009.

[37] Varta, "3.7 V — 1,000 mAh nominal — 3.7 Wh — VKB: 56457 201 016." VARTA 56457 201 016 Datasheet https://www.farnell.com/datasheets/3770113.pdf, Ago 2019. Rev. 1.

[38] Analog Devices, "3.15A USB Type-C Autonomous Charger with JEITA for 1-Cell Li-ion/LiFePO4 Batteries." MAX77757 Datasheet https://www.analog.com/media/en/technical-documentation/data-sheets/max77757.pdf, Jun 2021. Rev. 2.

[39] A. Tomaszewska, Z. Chu, X. Feng, S. O'kane, X. Liu, J. Chen, C. Ji, E. Endler, R. Li, L. Liu, *et al.*, "Lithium-ion battery fast charging: A review," *ETransportation*, vol. 1, p. 100011, 2019.

[40] Texas Instruments, "TPS61299 95nA Quiescent Current, 5.5V Boost Converter with Input Current Limit and Fast Transient Performance." TPS61299 Datasheet https://www.ti.com/lit/ds/symlink/tps61299.pdf?ts=1713271496311&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS61299, Mar 2024. Rev. D.

[41] Texas Instruments, "LP2981 100-mA, Low-Dropout Regulator in SOT-23 Package." LP2981 Datasheet https://www.ti.com/lit/ds/symlink/lp2981.pdf?ts=1713340985458&ref_url=https%253A%252F%252Fwww.google.it%252F, Dic 2023. Rev. H.

[42] C & K, "JS Series Sub-Miniature Slide Switches." JS102011JAQN Datasheet https://www.mouser.es/datasheet/2/240/js-3050885.pdf, Sep 2022.

[43] GCT, "Micro SD Memory Card Connector,1.40mm Profile SMT, Push-Push, with Normally open Switch." MEM2075-00-140-01-A Datasheet https://www.mouser.es/datasheet/2/837/MEM2075-2887971.pdf, Sep 2020. Rev. A1.

[44] S. Bhunia and M. H. Tehranipoor, *Hardware security : a hands-on learning approach*. Cambridge, Massachusetts: Morgan Kaufmann Publishers, 2019.

[45] S. Pithadia and S. More, "Grounding in mixed-signal systems demystified, part 1," *Texas Instruments*, 2013.

[46] N. L. Eastman, "Considerations for mixed analog/digital pcb design," in *Wescon/96*, pp. 297–301, IEEE, 1996.

[47] S. Pithadia and S. More, "Grounding in mixed-signal systems demystified, part 2," *Texas Instruments*, 2013.

[48] Infineon, "Bluetooth Low Energy (BLE PDL)." PSoC Creator Component Datasheet Bluetooth Low Energy (BLE PDL) https://www.infineon.com/dgdl/Infineon-Component_Bluetooth_Low_Energy_(BLE_PDL)_2.20-Software%20Module%20Datasheets-v02_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0eb8494b2d2d, Dec 2019. Rev. 2.20.

[49] Infineon, "Cypress PSoC 6 Bluetooth Low Energy Middleware Library 3.60." https://infineon.github.io/bless/ble_api_reference_manual/html/index.html. [Online; Accedido: 23 de Abril, 2024].

[50] B. B. Hafen and S. Sharma, "Oxygen saturation," *National Library of Medicine (NIH)*, 2018.

[51] P. A. Kyriacou and J. Allen, *Photoplethysmography: technology, signal analysis and applications*. Academic Press, 2021.

[52] T. Tamura, "Current progress of photoplethysmography and spo2 for health monitoring," *Biomedical engineering letters*, vol. 9, no. 1, pp. 21–36, 2019.

[53] C. L. Petersen, T. P. Chen, J. M. Ansermino, and G. A. Dumont, "Design and evaluation of a low-cost smartphone pulse oximeter," *Sensors*, vol. 13, no. 12, pp. 16882–16893, 2013.

[54] O. Tsiakaka, B. Gosselin, and S. Feruglio, "Source detector spectral pairing-related inaccuracies in pulse oximetry: Evaluation of the wavelength shift," *Sensors*, vol. 20, no. 11, p. 3302, 2020.

[55] J. Koseeyaporn, P. Wardkein, A. Sinchai, P. Kainan, and P. Tuwanut, "Pulse oximetry based on quadrature multiplexing of the amplitude modulated photoplethysmographic signals," *Sensors*, vol. 23, no. 13, p. 6106, 2023.

[56] Y. Chen, Y. Zheng, S. Johnson, R. Wiffen, and B. Yang, "A comparative study of accuracy in major adaptive filters for motion artifact removal in sleep apnea tests," *Medical & Biological Engineering & Computing*, vol. 62, no. 3, pp. 829–842, 2024.

[57] Osram, "SFH7072." OSRAM SFH7072 Datasheet https://look.ams-osram.com/m/682b32d8c8dd3713/original/SFH-7072.pdf, Ago 2023. Rev. 1.6.

[58] Texas Instruments, "Transimpedance Amplifier Circuit." Analog Engineer's Circuit - Transimpedance Amplifier Circuit https://www.ti.com/lit/an/sboa268a/sboa268a.pdf?ts=1717067537238&ref_url=https%253A%252F%252Fwww.google.com%252F, Ene 2019. Rev. A.

[59] Texas Instruments, "OPA380." Precision, High-Speed Transimpedance Amplifier https://www.ti.com/lit/ds/symlink/opa380.pdf?ts=1716994926736, Sep 2007. Rev. G.

[60] P. H. Charlton, P. A. Kyriacou, J. Mant, V. Marozas, P. Chowienczyk, and J. Alastruey, "Wearable photoplethysmography for cardiovascular monitoring," *Proceedings of the IEEE*, vol. 110, no. 3, pp. 355–381, 2022.

[61] M. Sharma, K. Barbosa, V. Ho, D. Griggs, T. Ghirmai, S. K. Krishnan, T. K. Hsiai, J.-C. Chiao, and H. Cao, "Cuff-less and continuous blood pressure monitoring: a methodological review," *Technologies*, vol. 5, no. 2, p. 21, 2017.

[62] FreeRTOS, "The FreeRTOS™ Kernel." https://www.freertos.org/RTOS.html. (Accedido 31 de Mayo, 2024).

[63] FreeRTOS, "FreeRTOS - Queue Management." https://www.freertos.org/a00018.html. (Accedido 31 de Mayo, 2024).

[64] Henrik Blidh, "Bleak Documentation." https://bleak.readthedocs.io/en/latest/#. [Online; Accedido: 24 de Abril, 2024].

[65] Python, "asyncio — Asynchronous I/O (Python 3.12.3)." https://docs.python.org/3/library/asyncio.html. [Online; Accedido: 28 de Mayo, 2024].

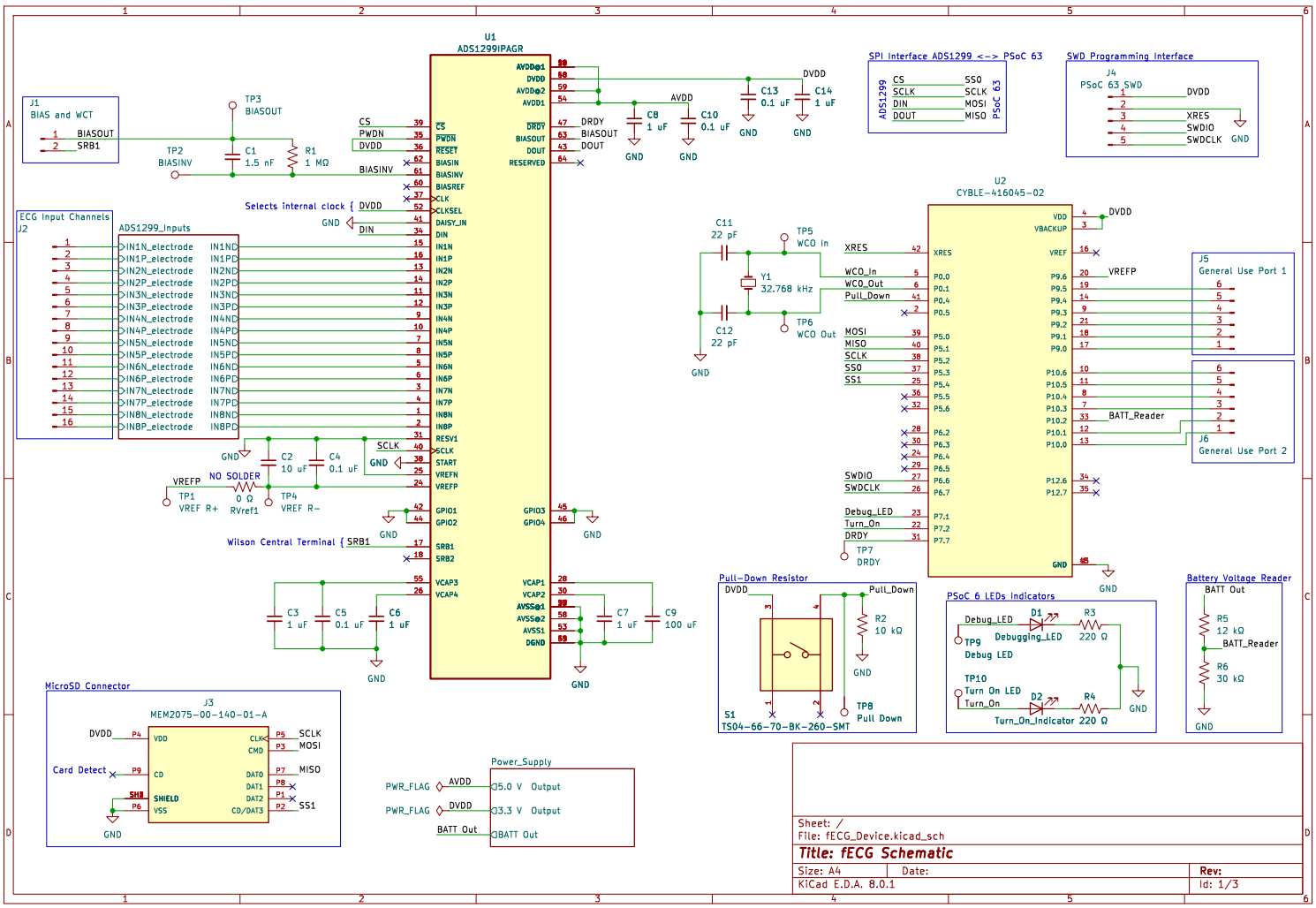
[66] E. Castillo, D. P. Morales, G. Botella, A. Garcia, L. Parrilla, and A. J. Palma, "Efficient wavelet-based ecg processing for single-lead fhr extraction," *Digital Signal Processing*, vol. 23, no. 6, pp. 1897–1909, 2013.

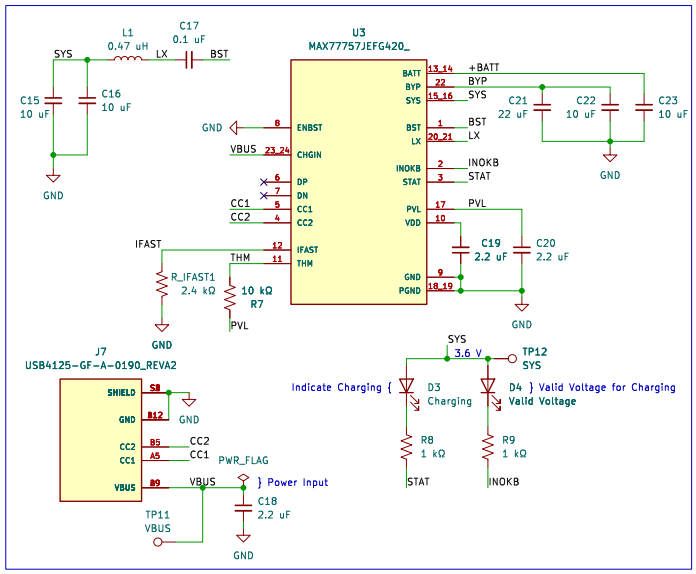
[67] AISLER, "Quick and affordable manufacturing for your Electronic Project." https://aisler.net/. (Accedido 4 de Junio, 2024).

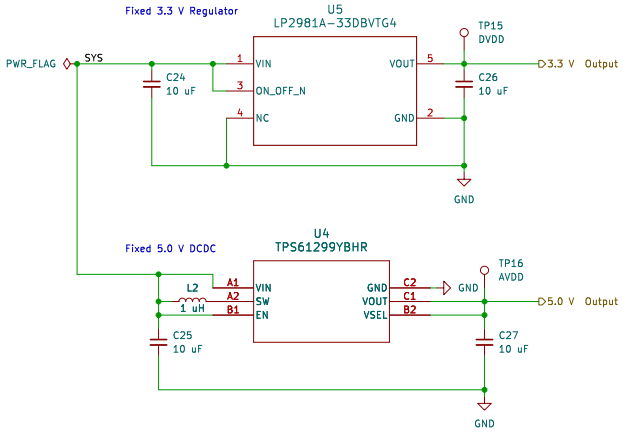
Apéndice A

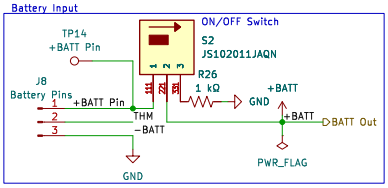
Esquemáticos del instrumento diseñado

En este anexo se pueden encontrar todos los esquemáticos diseñados en KiCad para la implementación de la PCB del instrumento diseñado en este trabajo.

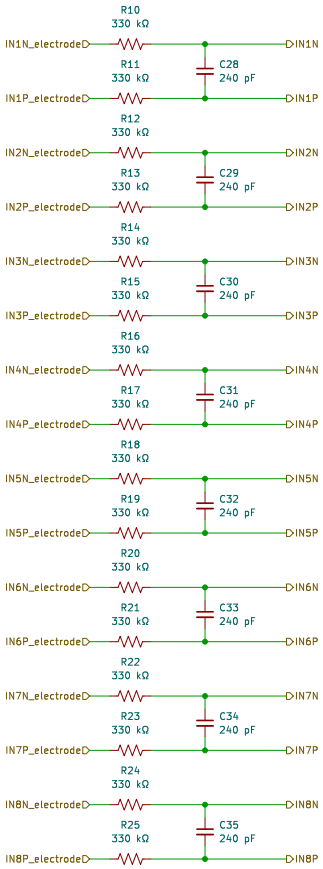

U1
ADS1299IPAGR
J1
BIAS and WCT
BIASOUT
SRB1
TP3
BIASOUT
TP2
BIASINV
C1
1.5 nF
R1
1 MΩ
CS
PWDN
DVDD
BIASINV
Selects internal clock
DVDD
GND
DIN
ECG Input Channels
J2
ADS1299_Inputs
IN1N_electrode
IN1P_electrode
IN2N_electrode
IN2P_electrode
IN3N_electrode
IN3P_electrode
IN4N_electrode
IN4P_electrode
IN5N_electrode
IN5P_electrode
IN6N_electrode
IN6P_electrode
IN7N_electrode
IN7P_electrode
IN8N_electrode
IN8P_electrode
C2
10 uF
C4
0.1 uF
SCLK
VREFP
NO SOLDER
0 Ω
RVref1
TP1
VREF R+
TP4
VREF R−
Wilson Central Terminal
SRB1
C3
1 uF
C5
0.1 uF
C6
1 uF
C8
1 uF
C10
0.1 uF
C13
0.1 uF
C14
1 uF
C7
1 uF
C9
100 uF
DRDY
BIASOUT
DOUT
AVDD
DVDD
SPI Interface ADS1299 <-> PSoC 63
ADS1299
PSoC 63
CS
SCLK
DIN
DOUT
SSO
SCLK
MOSI
MISO
SWD Programming Interface
J4
PSoC 63 SWD
XRES
SWDIO
SWDCLK
U2
CYBLE-416045-02
C11
22 pF
C12
22 pF
Y1
32.768 kHz
TP5
WCO In
TP6
WCO Out
XRES
WCO_In
WCO_Out
Pull_Down
MOSI
MISO
SCLK
SSO
SS1
SWDIO
SWDCLK
Debug_LED
Turn_On
DRDY
TP7
DRDY
J5
General Use Port 1
J6
General Use Port 2
BATT_Reader
Pull-Down Resistor
R2
10 kΩ
S1
TS04-66-70-BK-260-SMT
TP8
Pull Down
PSoC 6 LEDs Indicators
Debug_LED
D1
R3
220 Ω
Debugging_LED
TP9
Debug LED
TP10
Turn On LED
Turn_On
D2
R4
Turn_On_Indicator 220 Ω
Battery Voltage Reader
BATT Out
R5
12 kΩ
R6
30 kΩ
MicroSD Connector
J3
MEM2075-00-140-01-A
Card Detect
Power_Supply
PWR_FLAG
AVDD
5.0 V Output
DVDD
3.3 V Output
BATT Out
Sheet: /
File: fECG_Device.kicad_sch
Title: fECG Schematic
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.1
Id: 1/3







Sheet: /Power_Supply/	
File: Power_Supply.kicad_sch	
Title: fECG Power	
Size: A4	Date:
KiCad E.D.A. 8.0.1	Id: 2/3



Sheet: /ADS1299_Inputs/
File: ADS1299_Inputs.kicad_sch

Title: Differential Input Filters

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.1		Id: 3/3

Apéndice B

Listado de materiales (BOM)

En este apartado de puede consular el listado de materiales o BOM (*Bill o Materials*), este no es más que un listado con la totalidad de los componentes necesarios para la fabricación del instrumento. Se proporciona, el identificador dentro del esquemático, la cantidad total necesitada, el fabricante, el código de identificación del producto ("Mfg Part"), el valor, si es un elemento pasivo o una breve descripción si no, el empaquetado utilizado y finalmente el tipo de soldadura a la PCB.

Nótese que en la mayoría de los casos, los componentes elegidos son del tipo SMD (*Surface Mounted Device*), es decir, son elementos que se montan sobre la superficie de la PCB sin tener que atravesarla. Esto permite que el elemento se suelde sobre, únicamente, una de las caras sin afectar a la contraria. Esto permite que, llegado el caso, se pudieran utilizar ambas caras de la PCB para el montaje de componentes, reduciendo con ello el tamaño final. En el contexto de este trabajo, el principal interés de preferir este tipo de soldadura, al margen de que los componentes que la utilizan suelen ser de menor tamaño, es que el plano de referencia sea lo más continuo posible.

Item	Ref	Qty	Manufacturer	Mfg Part	Valor / Descripción	Package	Type
1	C1	1	Wurth Elektronik	885012205007	1.5 nF	C_0402_1005Metric	SMD
2	C10, C13, C17, C4, C5	5	KEMET	C0402C104K8RAC	0.1 uF	C_0402_1005Metric	SMD
3	C11, C12	2	Murata	GRM1555C1E220JA01D	22 pF	C_0402_1005Metric	SMD
4	C14, C3, C6, C7, C8	5	KEMET	C0402C105M9PAC	1 uF	C_0402_1005Metric	SMD
5	C15, C16, C2, C22, C23, C24, C25, C26, C27	9	Murata	GRM155R60J106ME05J	10 uF	C_0402_1005Metric	SMD
6	C18, C19, C20	3	Murata	GRM155C80J225ME95D	2.2 uF	C_0402_1005Metric	SMD
7	C21	1	Murata	GRM188R60G226MEA0D	22 uF	C_0603_1608Metric	SMD
8	C28, C29, C30, C31, C32, C33, C34, C35	8	Murata	GRM1885C1H241JA01D	240 pF	C_0402_1005Metric	SMD
9	C9	1	Samsung Electro-Mechanics	CL31A107MQHNNWE	100 uF	C_0805_2012Metric	SMD
10	D1	1	Inolux	743-IN-S42BT5Y	Yellow LED	LED_0402_1005Metric	SMD
11	D2	1	Inolux	IN-S42BT5G	Green LED	LED_0402_1005Metric	SMD
12	D3	1	Inolux	IN-S42BT5G	Green LED	LED_0402_1005Metric	SMD
13	D4	1	Inolux	IN-S42BT5G	Green LED	LED_0402_1005Metric	SMD
14	J1	1	Harwin	M20-8770242	VERTICAL SMT HEADER	PinHeader_1x02_P2.54mm_Vertical_SMD_Pin1Right	SMD
15	J2	1	Samtec	TSM-116-02-L-SH	Horizontal SMT HEADER	SAMTEC_TSM-116-02-L-SH	SMD
16	J3	1	Global Connector Technology (GCT)	MEM2075-00-140-01-A	MicroSD Connector	GCT_MEM2075-00-140-01-A	SMD
17	J4	1	Amphenol	10129380-905002BLF	1x5 Male Pin Headers	PinHeader_1x05_P2.54mm_Vertical_SMD_Pin1Left	SMD
18	J5	1	Amphenol	10129380-906002BLF	1x6 Male Pin Headers	PinHeader_1x06_P2.54mm_Vertical_SMD_Pin1Right	SMD
19	J6	1	Amphenol	10129380-906002BLF	1x6 Male Pin Headers	PinHeader_1x06_P2.54mm_Vertical_SMD_Pin1Right	SMD
20	J7	1	Global Connector Technology (GCT)	USB4125-GF-A-0190	USB Type C Connectors (Female)	GCT_USB4125-GF-A-0190_REVA2	SMD
21	J8	1	-	-	-	-	DNP
22	L1	1	Samsung Electro-Mechanics	CIGT252010LMR47MNE	0.47 uH	L_1008_2520Metric	SMD
23	L2	1	Cyntec	HTEK16080H-1R0MSR	1 uH	L_0603_1608Metric	SMD
24	R1	1	Wurth Elektronik	560112110033	1 MΩ	R_0402_1005Metric	SMD
25	R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25	16	Wurth Elektronik	560112110088	330 kΩ	R_0402_1005Metric	SMD
26	R2, R7	2	Wurth Elektronik	560112110020	10 kΩ	R_0402_1005Metric	SMD
27	R3, R4	2	Wurth Elektronik	560112110217	220 Ω	R_0402_1005Metric	SMD
28	R5	1	Wurth Elektronik	560112110038	12 kΩ	R_0402_1005Metric	SMD
29	R6	1	Vishay	RCS040230K0FKED	30 kΩ	R_0402_1005Metric	SMD
30	R8, R9	2	Vishay	CRCW04021K00FKED	1 kΩ	R_0402_1005Metric	SMD
31	RVref1	1	Vishay	CRCW08050000Z0EC	0 Ω	R_0805_2012Metric	SMD
32	R_IFAST1	1	Vishay	CRCW08052K40FKEA	2.4 kΩ	R_0805_2012Metric	SMD
33	S1	1	CUI Devices	TS04-66-70-BK-260-SMT	SWITCH TACTILE SPST-NO 0.05A 12V	SW_TS04-66-70-BK-260-SMT	SMD
34	S2	1	C&K	JS102011JAQN	90^{o} Slide Switch	SW_JS102011JAQN	SMD/THR
35	U1	1	Texas Instruments	ADS1299IPAG	-	TQFP-64	SMD
36	U2	1	Infineon	CYBLE-416045-02	-	XCVR_CYBLE-416045-02	SMD
37	U3	1	Analog Devices	MAX77757JEFG420+	Battery Charger (4.2 V)	FC2QFN	SMD
38	U4	1	Texas Instruments	TPS61299YBHR	Fixed 5 V DC-DC Boost	WCSP	SMD
39	U5	1	Texas Instruments	LP2981A-33DBVTG4	Fixed 3.3 V Linear Regulator	SOT-23	SMD
40	Y1	1	Epson Timing	Q13FC1350000400	32.768 kHz Crystal Oscillator (Cl = 12.5 pF)	XTAL_FC-135_32.7680KA-A0	SMD

Tabla B.1: BOM del instrumento

Apéndice C

Costes de fabricación

En este anexo se puede consultar el coste total de fabricación de una tirada de prototipaje del instrumento diseñado en el Capítulo 2. La PCB se ha solicitado al fabricante AISLER [67]. El coste total de fabricación asciende a, incluyendo la propia PCB, los componentes, el ensamblaje de los mismo y los gastos de envío, **739.97EUR**, siendo el coste individual de cada PCB de **241.66EUR**. La factura correspondiente se puede encontrar en la Página 84.

Es importante tener en cuenta que los costes se ven influenciados a la alta debido a que al ser un tirada pequeña para prototipaje, el precio de cada dispositivo individual se incrementa. En caso de tiradas a gran escala para, por ejemplo, comercialización, el precio unitario se reduciría de manera considerable.

En la Tabla C.1 y en la Tabla C.2, se puede consultar el coste estimado de cada componente individual necesario para la fabricación de cada instrumento. Esto conlleva un precio en componentes de, al menos, **97EUR** por instrumento. Nótese que esta estimación de precios esta actualizada al 1 de Junio de 2024 y puede variar a fechas posteriores. Adicionalmente, este precio también se verá modificado según la cantidad de dispositivos fabricados, ya que el precio de los componentes electrónicos se reduce a mayor cantidad se adquieran de una sola vez.

Componente	Modelo	Uds	Coste/Unidad (EUR)	Coste Total (EUR)
Condensadores	885012205007	1	0.092026	0.092026
	C0402C104K8RAC	5	-	-
	GRM1555C1E220JA01D	2	0.092026	0.184052
	C0402C105M9PAC	5	0.093	0.465
	GRM155R60J106ME05J	9	0.093	0.837
	GRM155C80J225ME95D	3	0.093	0.279
	GRM188R60G226MEA0D	1	0.1196338	0.1196338
	GRM1885C1H241JA01D	8	0.13	1.04
	CL31A107MQHNNWE	1	0.3957118	0.3957118
LED Amarillo	743-IN-S42BT5Y	1	0.2852806	0.2852806
LEDs Verdes	IN-S42BT5G	3	0.3312936	0.9938808
Conectores macho superficial	M20-8770242	1	0.2392676	0.2392676
	TSM-116-02-L-SH	1	2.5951332	2.5951332
	10129380-905002BLF	1	0.184052	0.184052
	10129380-906002BLF	2	0.0414117	0.0828234
Conector microSD	MEM2075-00-140-01-A	1	1.8221148	1.8221148
Conector USB	USB4125-GF-A-0190	1	0.34877854	0.34877854
Bobinas	CIGT252010LMR47MNE	1	0.2116598	0.2116598
	HTEK16080H-1R0MSR	1	0.2208624	0.2208624

Tabla C.1: BOM de los componentes del instrumento con precios incluidos (Parte 1 de 2)

Componente	Modelo	Uds	Coste/Unidad (EUR)	Coste Total (EUR)
Resistencias	560112110033	1	0.092026	0.092026
	560112110088	16	0.08098288	1.29572608
	560112110020	2	0.092026	0.184052
	560112110217	2	0.092026	0.184052
	560112110038	1	0.092026	0.092026
	RCS040230K0FKED	1	0.093	0.093
	CRCW04021K00FKED	2	0.0092026	0.0184052
	CRCW08050000Z0EC	1	0.092026	0.092026
	CRCW08052K40FKEA	1	0.0046013	0.0046013
Switches	TS04-66-70-BK-260-SMT	1	0.1196338	0.1196338
	JS102011JAQN	1	0.23374604	0.23374604
Integrado adquisición muestras ECG	ADS1299IPAG	1	61.8322694	61.8322694
Integrado control y comunicación	CYBLE-416045-02	1	14.4204742	14.4204742
Cargador baterías	MAX77757JEFG420+	1	3.5245958	3.5245958
Reguladores tensiones	TPS61299YBHR	1	2.714767	2.714767
	LP2981A-33DBVTG4	1	2.0797876	2.0797876
Oscilador de cuarzo	Q13FC1350000400	1	0.2147	0.2147
Coste Total			**93.1747**	**97.5947**

Tabla C.2: BOM de los componentes del instrumento con precios incluidos (Parte 2 de 2)

AISLER

AISLER Germany GmbH
Philipsstraße 8 - 52068 Aachen - Germany

Dpto Electrónica Tec. Computadores
c/o [redacted]
Facultad de Ciencias, Av. Fuente Nueva , sn
18071 Granada
Spain

Invoice

Invoice #	[redacted]
Invoice Date	28-05-2024
Delivery Date	28-05-2024
Ordered by	[redacted]
Payment Due At	**11-06-2024**
Grand Total	**€ 739,97**

Bill to	Ship to	Fulfilled by
Dpto Electrónica Tec. Computadores c/o [redacted] Facultad de Ciencias, Av. Fuente Nueva , sn 18071 Granada Spain [redacted]	Dpto Electrónica Tec. Computadores c/o [redacted] Facultad de Ciencias, Av. Fuente Nueva , sn 18071 Granada Spain [redacted]	AISLER Germany GmbH Philipsstraße 8 52068 Aachen Germany [redacted]

#	Item	Qty	Rate	Total
1	Assembled PCBs (Simple) Amazing Assembly with simple board configuration Project: ELHJDSVJ Our Order No: 2024-22963	3x	€ 241,66	€ 724,98
2	Tracked Shipping Tracked Shipping Project: ELHJDSVJ Our Order No: 2024-22963	1x	€ 14,99	€ 14,99
			Net Total	**€ 739,97**