Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Diseño y Desarrollo de un Sistema Ambulatorio para la Adquisición de Señales Cardíaca y Respiratoria

Autor: Antonio López Marín
Tutor: Juan José Ramos Castro
Ponente: Laura María Roa Romero

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Diseño y Desarrollo de un Sistema Ambulatorio para la Adquisición de Señales Cardíaca y Respiratoria

Autor:

Antonio López Marín

Tutor:

Juan José Ramos Castro

Profesor titular en la Universidad Politécnica de Cataluña

Ponente:

Laura María Roa Romero

Catedrática en la Universidad de Sevilla

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera:

Diseño y Desarrollo de un Sistema Ambulatorio para la Adquisición de Señales Cardíaca y Respiratoria

Autor: Antonio López Marín

Tutor: Juan José Ramos Castro

Ponente: Laura María Roa Romero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi familia, en especial a mis padres y a
mi hermana, que siempre me han apoyado
en las decisiones que he tomado a lo largo de mi vida.
Sin su confianza y cariño no habría podido llegar tan lejos.

A Francisco Ruiz, mi compañero de piso y de proyecto,
porque sin su ayuda este trabajo nunca habría sido realizado
y por estar siempre dispuesto a hacer las cosas lo mejor posible.

A mis antiguos compañeros de piso, Alejandro, David y Fernando.
Aunque estemos separados, nunca olvidaré todos los años compartidos
superando los retos que se presentaban en la carrera.

A mis tutores del proyecto en Barcelona,
Mireya Fernández y Juan Ramos,
por guiarme y ayudarme en todo momento mientras lo realizaba.
También agradecer a mi tutora en la ESI, Laura Roa,
por ofrecerse sin dudar para que pudiera seguir adelante con él.

A todos mis compañeros y profesores del laboratorio,
que me han ayudado siempre que me ha hecho falta,
en especial a Alfonso Méndez, siempre dispuesto a echarme
una mano en lo que no sabía de electrónica.

A los compañeros y profesores de la Universidad de Sevilla,
que me ayudaron mientras estudié allí y me enseñaron todo lo que sé.

Antonio López Marín

Barcelona, Septiembre de 2015

Resumen

La medida y seguimiento de las constantes vitales como el pulso y la respiración son una pieza clave en la detección prematura de problemas de salud. Su análisis en tiempo real permite diagnosticar enfermedades y prevenir situaciones de peligro para el paciente.

Existen varios dispositivos en el mercado que, en mayor o menor medida, dan solución a esta necesidad. Sin embargo, ninguno resulta de utilidad para las investigaciones llevadas a cabo en este laboratorio.

Se ha diseñado un dispositivo de tamaño reducido alimentado a batería capaz de: capturar las señales cardíaca y respiratoria de un paciente mediante dos electrodos; enviar las señales muestreadas en tiempo real a un dispositivo externo mediante Bluetooth Low Energy; y recargar su batería de manera inalámbrica mediante el protocolo Qi.

Con una resolución de 24 bits y una frecuencia de muestreo de 1000 Hercios, el dispositivo aquí desarrollado puede obtener tanto ECG como ritmo respiratorio y transmitirlos a través de un enlace inalámbrico de hasta 10 metros.

El dispositivo desarrollado, además de mejorar anteriores versiones, cumple con las necesidades para las que fue diseñado y abre la puerta al diseño de futuros prototipos con mayores prestaciones y comodidad de uso.

Abstract

Measuring and monitoring vital signs such as pulse and breathing is a key factor on the early detection of health problems. The real-time analysis of these signals allows to diagnose conditions and prevent life-threatening episodes on patients.

There are several commercially available devices that, to some extent, solve this need. However, none of them suits the research carried out in this laboratory.

We have designed a battery-powered compact device capable of: capturing the heart and respiratory signals from a patient using two electrodes; send the sampled signals in real time to an external device via Bluetooth Low Energy; and recharge its battery wirelessly by means of the Qi protocol.

Using its 24-bit resolution and sampling frequency of 1000 Hertz, the developed device can obtain both electrocardiogram and breathing and transmit them over a 10-meter wireless link.

This device not only improves earlier versions but it also fulfils the proposed requirements and its development opens the way to future design of prototypes with higher performance and wearing comfort.

Índice

1 INTRODUCCIÓN

Desde que Norman Holter y Bruce del Mar colaborasen para crear el primer monitor cardíaco comercial en 1949 [1], se han producido grandes avances en el desarrollo de este tipo de dispositivos. El auge de los denominados *wearables* y la creciente accesibilidad a herramientas de desarrollo han aumentado tanto la rapidez como la facilidad de implementación.

El proyecto aquí presentado tiene como objetivo el diseño de un **Sistema ambulatorio de actividad cardíaca y respiratoria**, a utilizar junto con una banda elástica comercial o electrodos convencionales. A su vez, se enmarca en diversas investigaciones llevadas a cabo por el ***Grupo de Instrumentación Electrónica y Biomédica (IEB)*** del Departamento de Ingeniería Electrónica de la Universidad Politécnica de Cataluña.

Dichos estudios buscan relacionar la variabilidad de la frecuencia cardíaca con las actividades diarias del paciente, ya sea para su diagnóstico o para la detección precoz de condiciones de salud anómalas. Por otro lado el análisis del ritmo respiratorio, además de diagnosticar posibles enfermedades, permite detectar niveles de fatiga y somnolencia al conducir o hacer un seguimiento en situaciones donde el paciente deba prestar la máxima atención.

1.1 Contexto

Una ***señal biométrica*** o *bioseñal* se puede denominar como aquella que cuantifica alguna característica de un ser vivo. El dispositivo a desarrollar capturará dos *bioseñales* por el contacto con la piel de un paciente mediante electrodos.

Por una parte, la ***señal cardíaca*** y el ***electrocardiograma*** (ECG). Este último se define como un registro de la actividad eléctrica en el miocardio (el tejido muscular del corazón). El latido del corazón registra en el ECG una señal como la de la Figura 1-1. Esta señal se caracteriza por tener un espectro de frecuencias entre 0.05 y 150 Hz, amplitud entre 0.1 y 10 mV y una frecuencia típica de ciclo entre 60 y 100 pulsaciones por minuto [2].

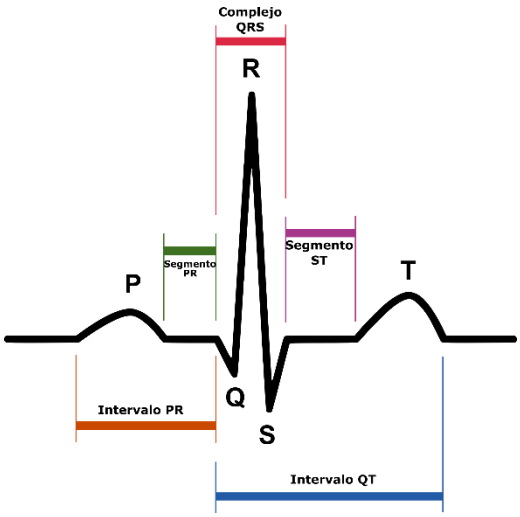


Figura 1-1. Ciclo ECG típico

Se define una ***derivación*** de ECG como la medida de señal entre dos electrodos colocados sobre la piel, para la que se han establecido a su vez diferentes derivaciones estandarizadas. Las más comunes son las llamadas ***derivaciones periféricas***:

- *Derivación I*: Medida de potencial eléctrico entre brazo derecho (RA) e izquierdo (LA).
- *Derivación II*: Medida de potencial eléctrico entre brazo derecho (RA) y pierna izquierda (LL).
- *Derivación III*: Medida de potencial eléctrico entre brazo izquierdo (LA) y pierna izquierda (LL).

En este proyecto hará uso de la **Derivación I** (Lead I), con la entrada positiva conectada al brazo izquierdo (LA) y la negativa al derecho (RA) [2]. El signficado eléctrico de cada una de estas derivaciones se puede visualizar en el *Triángulo de Einthoven*, mostrado en la Figura 1-2.

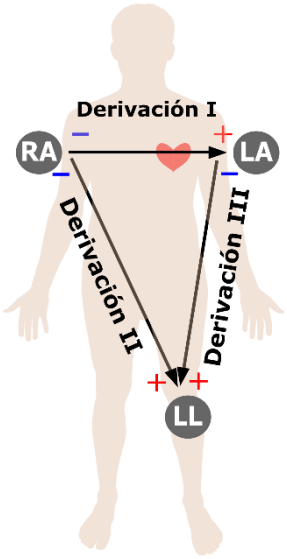


Figura 1-2. Derivaciones periféricas

Ley de Einthoven: El potencial de la derivación II será la suma de los potenciales detectados en las derivaciones I y III [2]. Puede demostrarse fácilmente aplicando la 2ª Ley de Kirchoff al triángulo.

El análisis del *electrocardiograma* obtenido mediante cualquiera de estas derivaciones aporta valiosa información como el ritmo cardiaco, detección de anomalías en la actividad del corazón o enfermedades asociadas al mismo.

Por otro lado, el ***ritmo respiratorio*** puede deducirse a partir del ***Volumen total*** (TV) o Tidal, que es la cantidad de aire que entra y sale de los pulmones. Tiene un valor aproximado de **500 ml** o **7 ml/kg** en adultos y empieza desde un valor fijo *Resting Expiatory Level* (REL) [3]. El volumen total o TV se obtendrá mediante el análisis de la *impedancia bioeléctrica* entre los dos electrodos.

La ***impedancia bioeléctrica*** es una técnica que consiste en estimar la impedancia de un objeto mediante un barrido de frecuencias que va de los 50 a los 100 kHz. Para nuestro análisis, su cálculo en el tórax del paciente [3] resulta de gran utilidad. A partir de la misma se podrán obtener los cambios en el volumen de aire (1-2Ω/L) y por tanto, la señal respiratoria, con una frecuencia comprendida entre 0.05 y 2 Hz y variaciones no mayores de 10 ohmios [2].

Existen dos métodos para obtener la impedancia, mediante técnicas de *dos* o *cuatro electrodos*. Aun sabiendo que se introducirán errores en la medida [4], se usará una técnica con ***dos electrodos***, ya que el interés de esta aplicación se centra no tanto en el valor de la impedancia como en su variación (ritmo respiratorio), permitiendo reducir la complejidad de uso del sistema para el paciente.

1.2 Objetivos del Proyecto

El objetivo principal de este proyecto final de carrera consiste en desarrollar un dispositivo portable capaz de adquirir de manera ambulatoria las señales cardíaca y respiratoria de un paciente y que éstas sean transmitidas a un dispositivo externo (*Monitor*) para su procesado y/o almacenamiento.

El envío de las señales capturadas se hará por ***transmisión inalámbrica*** en ***tiempo real*** y el dispositivo a su vez deberá ser de ***tamaño reducido***, ***bajo consumo*** y con un sistema de ***carga inductiva*** para la batería.

En base a este objetivo, se desprenden una serie de tareas específicas:

- Estudiar el estado del arte, esto es, encontrar dispositivos con funciones similares que se encuentren a la venta. En función de sus características y el conocimiento previo en el desarrollo de este tipo de dispositivos, definir las especificaciones del sistema a diseñar.
- Clasificar tecnologías (comunicación, procesamiento, carga inductiva) existentes que puedan resultar útiles para el diseño del dispositivo. Más tarde, estudiar los productos ofrecidos por los fabricantes y determinar aquellos que mejor se ajusten a las necesidades del proyecto.
- Implementar un primer dispositivo en placa de pruebas con aquellos productos seleccionados y desarrollar tanto código como interconexiado de los diferentes módulos que conforman el sistema.
- Tras confirmar la viabilidad del primero, implementar un segundo dispositivo en placa de circuito integrado con componentes de montaje superficial, a su vez integrado en una pequeña caja cuyas únicas conexiones externas sean las correspodientes a los electrodos.
- Comprobar el correcto funcionamiento del segundo prototipo y realizar pruebas tanto simuladas como con pacientes reales para extraer resultados y conclusiones de cara a futuras líneas de investigación.

1.3 Estado del arte

Los denominados *Sistemas Holter* han evolucionado a lo largo del tiempo. En los últimos años han aparecido dispositivos de un coste asequible que ofrecen tanto monitorización clínica del paciente como otras aplicaciones relacionadas con el deporte y la salud. Esto ha hecho que sumadas a la comodidad, se valoren otras características como intercompatibilidad entre marcas o facilidad de uso.

Se presentan a continuación una serie de productos a la venta, enmarcados en el tipo de dispositivo que hemos definido en los Objetivos del Proyecto.

Figura 1-3. CardioLeaf ULTRA de Clearbridge VitalSigns

El **CardioLeaf** [5] es un dispositivo de bajo consumo con electrodos integrados que cuenta con autonomía y capacidad de almacenamiento de la señal cardiaca para 7 días.

Mediante una aplicación para Smartphone u ordenador se pueden recopilar los datos para que un médico cualificado los estudie y formule un diagnóstico.

El fabricante no aporta datos técnicos acerca de la captura de los datos, además de que el dispositivo no captura el ritmo respiratorio.

Figura 1-4. BioPatch de Zephyr

El dispositivo **BioPatch** [6] permite tanto transmitir como almacenar bioseñales como el ECG y el ritmo respiratorio, además de poseer diferentes algoritmos de procesamiento internos, comunicación Bluetooth y ofrecer una aplicación para Android e iOS.

Afortunadamente este fabricante sí aporta información técnica acerca de la captura de los datos:

- Frecuencia de muestro ECG a 1000 Hz
- Conversión analógico-digital ECG de 12 bits
- Frecuencia de muetreo Ritmo respiratorio a 25 Hz

Sin embargo, la transmisión de los datos no se realiza en tiempo real, si no en *bloques de 30 segundos*.

Figura 1-5. VitalJacket de Biodevices

La **VitalJacket** [7] está formada por una camiseta con electrodos integrados y un dipositivo del tamaño de un teléfono móvil que realiza la captura. Cuenta con autonomía y capacidad de almacenamiento de la señal cardiaca de 3 día, además de:

- Frecuencia de muestro ECG a 500 Hz
- Conversión analógico-digital ECG de 10 bits

Al igual que el anterior, no ofrece la posibilidad de realizar la transmisión en tiempo real.

1.4 Requisitos y Especificaciones del Proyecto

Una vez presentados los productos disponibles en el mercado, se pueden establecer una serie de especificaciones que deberá cumplir nuestro dispositivo, según la Tabla 1–1. Éstas se han definido teniendo en cuenta tanto el *Estado del arte* como trabajos previos dentro del propio laboratorio.

Tabla 1–1. Especificaciones del dispositivo

Ancho de banda	0.01 – 250 Hz
Rango de voltaje [Abs.]	0.5 – 4 mV
Resolución	24 bits
Frecuencia de muestreo	500 muestras/s
Autonomía	24 horas
Dimensiones	≤ (50x50x20) mm
Frecuencia transmisión	Banda ISM
Alcance de transmisión	1 – 5 metros
Alimentación	1.8 – 3.3 V
Carga de la batería	Inalámbrica

El diseño del sistema se llevará a cabo en base al diagrama de bloques del sistema de la Figura 1-6, que cuenta con los siguientes bloques funcionales:

1. ***Adquisición de datos***, que a partir de los electrodos capture las señales cardíaca y respiratoria.
2. ***Control y procesamiento***, que gestione el sistema y la comunicación con los demás bloques.
3. ***Comunicación inalámbrica***, que envíe los datos al *Monitor* en tiempo real.
4. ***Alimentación y carga inalámbrica***, que gestione de forma autónoma la batería del sistema.

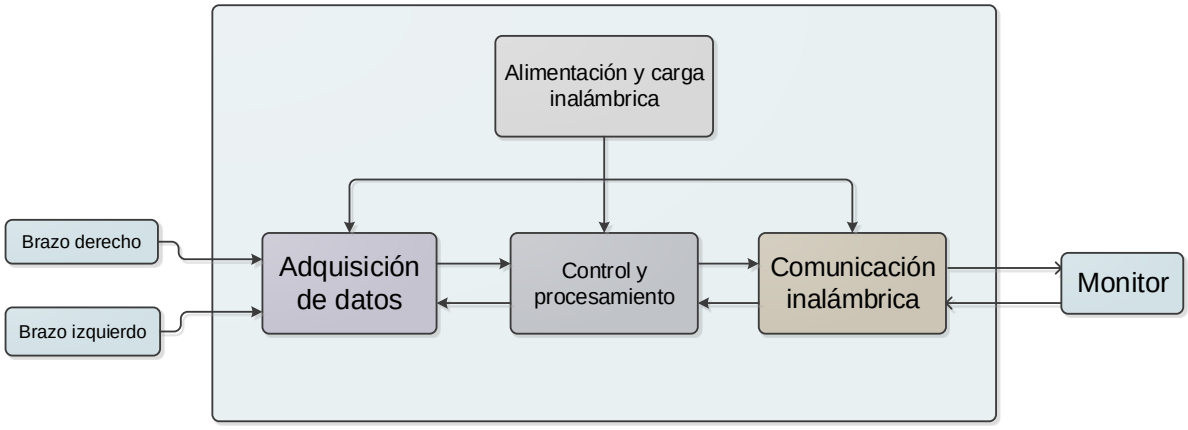


Figura 1-6. Diagrama de bloques conceptual

La implementación del ***Monitor***, aun estando fuera del ámbito de este proyecto, será parte imprescindible para comprobar el funcionamiento del dispositivo aquí desarrollado. El diseño de dicho Monitor se llevará a cabo por otro alumno en paralelo desarrollando una aplicación para Android. Ambos proyectos por tanto, compartirán parte de la sección de resultados.

1.5 Estructura de la memoria

Este documento se divide en varios capítulos que separan cada una de las fases del proyecto:

- Capítulo 2 (Diseño del Sistema), donde se definirá el sistema y componentes pertenecientes a cada uno de los bloques de la Figura 1-6.
- Capítulo 3 (Implementación del Sistema), donde se detalla cómo se ha implementado cada uno de esos bloques.
- Capítulo 4 (Caracterización del Sistema), donde se realizan diversas pruebas del dispositivo para comprobar que funciona correctamente.
- Capítulo 5 (Resultados), donde se realizarán pruebas de distinto tipo tanto simuladas como con pacientes reales.
- Capítulo 6 (Estudio económico), para conocer tanto coste de este proyecto como hipotéticos costes de fabricación a gran escala.
- Capítulo 7 (Conclusiones), para valorar el proyecto en su conjunto.

Además de una serie de anexos con documentación extendida referente a dichos capítulos:

- Anexo A – Servicio y Perfil BLE
- Anexo B – Ficheros BLE112-A
- Anexo C – Código PIC18F24K20
- Anexo D – Esquemáticos del circuito
- Anexo E – Lista de materiales

2 DISEÑO DEL SISTEMA

Todo diseño requiere de una etapa de estudio previo en la que se pone en tela de juicio la viabilidad del propio proyecto y a su vez se busca la solución que más se ajuste a las necesidades del mismo. Para ello, el capítulo quedará dividido en varios apartados que se corresponden con cada uno de los bloques definidos en los ***Requisitos y Especificaciones del Proyecto***. Cabe decir que la solución a la que se quiere llegar en este proyecto ya ha sido planteada con anterioridad [8], por lo que en primera instancia el sistema es realizable.

La elección de componentes debe responder tanto a factores de diseño como coste, tamaño o consumo, como a la posibilidad del propio laboratorio de conseguirlos y la disponibilidad de las herramientas para su programación e implementación en el sistema final. Además, cualquier decisión tomada en un apartado implicará restricciones en el diseño del resto.

2.1 Adquisición de datos

La adquisición de las señales de interés puede realizarse mediante diferentes métodos. Por una parte dependiendo de la señal o el tipo de prueba, la posición de los electrodos resulta determinante. Usando una banda elástica sobre el tórax con los electrodos lo suficientemente separados, se pueden obtener ambas señales de una manera cómoda para el paciente.

Sin embargo, el sistema podría ser utilizado en otro tipo de diagnóstico. Por ejemplo, para el análisis del ritmo respiratorio en situaciones de conducción se prefiere usar electrodos textiles colocados en la parte del volante donde el conductor coloca las manos [9].

En cualquier caso, este apartado se centra en el sistema electrónico de adquisición propiamente dicho, para el que se presentan dos alternativas:

- Desarrollar un sistema de adquisición ***desde cero*** a partir de amplificadores y componentes pasivos. Se pueden incluir etapas de filtrado y de amplificación adaptados según se estime conveniente.
- Utilizar un ***circuito integrado*** preparado que automatice el proceso de adquisición.

A pesar de que la primera opción puede considerarse la óptima, el esfuerzo de diseño y tiempo necesarios hacen que nos decantemos por usar un circuito integrado. En el mercado, destacan tanto por volumen de ventas como por cantidad de documentación publicada el chip **ADS1292R** de *Texas Instruments* [10] y el **ADAS1000** de *Analog Devices* [11].

En la Tabla 2–1 se presenta una comparativa de ambos dispositivos. Se han escogido aquellas versiones que ofrecen tanto captura de señal cardíaca como respiratoria. Puede comprobarse que tanto uno como otro cumplen los requisitos referentes a la adquisición: frecuencia de muestreo, resolución y ancho de banda. Sin embargo, el modelo de *Texas Instruments* lo hace con un precio y consumo menores, además de tratarse de un producto con una mayor madurez en el mercado. Utilizando el protocolo serie SPI se podrán obtener dos canales indepedientes de electrocardiograma y señal respiratoria.

Tabla 2–1 Comparativa entre circuitos de adquisición

	ADAS1000	**ADS1292R**
Tipo de entrada	Diferencial	Diferencial
Número de canales	5	2
Resolución	24 bits	24 bits
Frecuencia de muestreo	2 MSa/s (máximo)	8 kSa/s (máximo)
Ancho de banda	65 kHz	8.5 kHz
Arquitectura	SAR	Sigma-Delta
Interfaz de comunicación	SPI	SPI
Impedancia de entrada DC	1000 MΩ (mínimo)	1000 MΩ (mínimo)
Potencia disipada	8.2 mW/canal	0.48 mW/canal
CMRR	-105 dB (mínimo)	-105 dB (mínimo)
SNR	100 dB	107 dB
Corriente total consumida	4.97 mA (2 canales)	**325 µA** (2 canales)
Precio (unidad)	31.09€	**10.35€**

Para comprobar el correcto funcionamiento del chip y dado que se encuentra en el laboratorio, se utilizará la placa de evaluación **ADS1292ECG-FE** de *Texas Instruments* [10] mostrada en la Figura 2-1. Mediante el software de evaluación que lo acompaña, se pueden analizar las prestaciones del chip en profundidad.

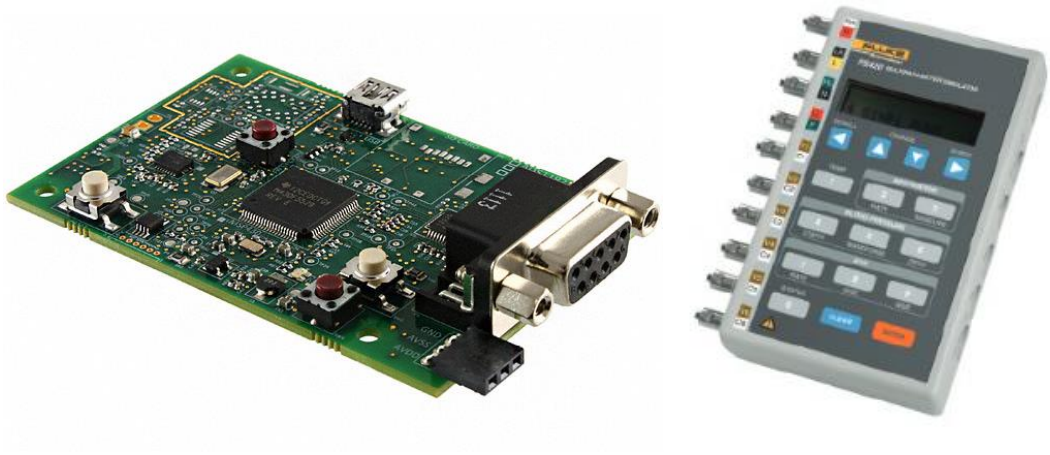


Figura 2-1. Texas Instruments ADS1292ECG-FE y Metron PS-420

La señal capturada de la placa vendrá dada por un Simulador de paciente **PS-420** de *Metron* [12] como el que aparece en la Figura 2-1. Esta herramienta permite simular diversas señales de origen cardíaco para los distintos tipos de derivaciones estandarizadas, además de generar una impedancia variable que simule el ciclo respiratorio de un paciente real.

Por último y no menos importante, al ser un producto de ámbito clínico que pudiera ponerse a la venta, requiere cumplir ciertos estándares para material de tipo médico.

El estándar **IEC 60601-2-25** [13] especifica los requisitos básicos y de rendimiento para equipos de electrocardiograma. En nuestro caso, el conjunto del dispositivo a desarrollar junto con la aplicación *Monitor* debe cumplir dicho estándar. El circuito integrado *ADS1292R* se ha diseñado en base a esta especificación, por lo que no será necesario comprobar que se cumplen en el dispositivo final.

2.2 Control y procesamiento

La gestión del sistema en su totalidad será la parte principal del esfuerzo de diseño en este proyecto, la cual deberá cumplir una serie de funciones que se desprenden del diagrama de bloques ya definido:

- Recibir las muestras del chip ADS1292R y procesarlas si es necesario.
- Enviar, mediante el uso del módulo de comunicación, las muestras al *Monitor*.
- Realizar tareas y comprobaciones adicionales que afecten al funcionamiento del sistema.

En base a los requisitos definidos en la Tabla 1–1 y otros implícitos en los *Objetivos del Proyecto*, se definen el la Tabla 2–2 una serie de especificaciones que debe cumplir este bloque.

Tabla 2–2 Requisitos del microcontrolador

Frecuencia de procesamiento	1-8 MIPS
Categoría	Bajo consumo
Comunicación	SPI, UART
Módulos necesarios	ADC

La ***frecuencia de procesamiento*** necesaria es uno de los parámetros más importantes. Podemos estimar un valor de compromiso en base al siguiente planteamiento:

- La frecuencia de muestreo es de 500 muestras por segundo, esto es, una muestra cada 2 milisegundos.
- Si x milisegundos se define como el tiempo necesario para la captura y procesamiento de las muestras, $(2 - x)$ milisegundos será el disponible para realizar el resto de operaciones antes de recibir la siguiente muestra.
- Suponemos que en un caso hipotético se ejecutan 1000 instrucciones para estas operaciones. Para ello es necesaria una frecuencia de procesamiento de 1 $MIPS$, es decir 1 milisegundo.

De ahí que se haya elegido un rango de 1 a 8 $MIPS$ como frecuencia de procesamiento. Este es, sin embargo, un parámetro que sólo puede determinarse a ciencia cierta en el momento de la implementación, por lo que es preferible ser conservador en su elección.

El dispositivo será de ***bajo consumo*** alimentado con una batería, por lo que resulta conveniente monitorizar el estado real de la misma mediante un conversor analógico-digital. El protocolo asíncrono ***UART*** es un estándar de comunicaciones muy utilizado en sistemas gobernados por microcontroladores. El módulo *ADS1292R* añade a su vez como requisito el uso del protocolo ***SPI***.

El módulo de evaluación **ADS1292ECG-FE** lleva integrado un microcontrolador de la serie **MSP430** de *Texas Instruments* [10] de bajo consumo. No obstante, se ha preferido utilizar un microcontrolador PIC de *Microchip* [14]. Dentro de la serie de microcontroladores de 8 bits de alto rendimiento **PIC18F**, se pueden encontrar varios con la tecnología XLP (*eXtreme Low Power*). De todos ellos se ha elegido la familia ***PIC18FxxK20***, con las siguientes características:

- Oscilador interno a 16 MHz calibrado al ±1%.
- PLL incorporado para obtener hasta 64 MHz, esto es, 16 MIPS.
- Tensión de alimentación entre 1.8 y 3.6 V.
- Módulo analógico-digital de 10 bits.
- Módulos SPI y UART.

Para facilitar el desarrollo de esta parte y la elección del dispositivo final se hará uso de la placa de evaluación **DM164130-4** como la de la Figura 2-2, la cual incorpora un **PIC18F45K20**, uno de los microcontroladores más completos de la familia escogida.

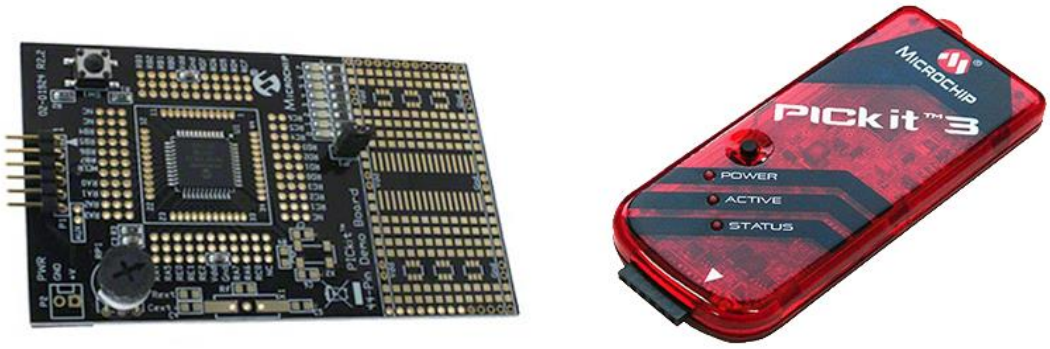


Figura 2-2. Microchip DM164130-4 y PICkit 3

La placa dispone de un interruptor de botón, un potenciómetro conectado al módulo ADC y 8 LED, para una comprobación del código en tiempo de ejecución mucho más sencilla.

Para su programación será necesario utilizar un programador **PICkit 3** como el de la Figura 2-2, además del entorno de programación **MPLAB X** junto con el compilador **XC8**, todo ello proporcionado por *Microchip* en su página web con licencia gratuita [14].

2.3 Transmisión de la señal

Desde el comienzo del diseño se ha dado por hecho que la transmisión se realizaría de la manera inalámbrica. Las comunicaciones inalámbricas de corto alcance se caracterizan por ser rápidas, eficientes e ideales para un volumen de datos no muy elevado. Dentro de las diferentes frecuencias de las **bandas ISM** encontramos:

- ***Bandas sub-GHz***, en 434 y 868 MHz.
- ***Bandas de GHz***, en 2.4 y 5.8 GHz.

Ya que la vida útil de la batería está fijada en 24 horas, el consumo deberá ser moderado. Por tanto, quedan descartadas tanto *WiFi*, como *WiMAX* o *telefonía móvil*, siendo la banda de 2.4 GHz la única adecuada. En esta banda podemos encontrar diferentes tecnologías de bajo consumo como son *Bluetooth Smart*, *ZigBee* o *ANT+*.

Los criterios de selección se basarán en:

- Consumo de transmisión y recepción
- Interoperabilidad entre diferentes fabricantes

El estándar ***Bluetooth Smart***, también denominado Bluetooth LE (*Low Energy*) es una revisión de la versión de Bluetooth tradicional o Classic y sigue la denominada *Especificación del Sistema Bluetooth* [15]. Al igual que su antecesora esta tecnología se basa en un sistema de perfiles, renovado para una transmisión de datos de más sencilla y eficaz. La tecnología ***ZigBee*** se caracteriza por una transmisión ligera, con aplicaciones interoperables entre diferentes fabricantes. La tecnología propietaria ***ANT+***, de características similares, requiere de una certificación específica por parte del fabricante, por lo que queda descartada a priori para nuestro proyecto.

Tabla 2–3 Comparativa entre módulos BLE y ZigBee

	Bluegiga BLE112	**Digi Xbee ZigBee**
Alcance	10 m	10 m
Velocidad efectiva	270 Kbps	64 Kbps
Tiempo de wake-up	6 milisegundos	30 milisegundos
Consumo máximo TX	36 mA	45 mA
Consumo máximo RX	25 mA	50 mA
Consumo máximo en reposo	235 µA	10 µA

Se han escogido dos módulos de comunicación, BLE112 [16] (*Bluetooth Smart*) y Xbee ZigBee [17] para realizar una comparativa de ambas tecnologías, reflejada en la Tabla 2–3. Tanto por compatibilidad y presencia en el mercado como por características técnicas, la opción que más se ajusta a los requisitos del proyecto es Bluetooth Low Energy.

2.3.1 Bluetooth Smart

Este capítulo estará dedicado a decribir *Bluetooth Low Energy*, también denominado Bluetooth Smart. Este estudio está basado en la *Especificación del estándar Bluetooth* [15].

2.3.1.1 Perfiles

Cualquier dispositivo Bluetooth LE basa su funcionamiento en el uso de *perfiles*. Estos permiten a fabricantes y desarrolladores de software crear productos interoperables bajo un estándar común.

Un *perfil* contiene una descripción de requisitos, servicios ofrecidos e información necesaria para realizar la conexión. Existen ***perfiles estandarizados***, gestionados por *Bluetooth SIG*, que se identifican por un UUID de 16 bits. A su vez, los desarrolladores pueden crear sus propios perfiles identificándolos con UUID de 128 bits.

2.3.1.2 Generic Access Profile (GAP)

El perfil GAP, presente en versiones anteriores de Bluetooth, es el encargado de la conexión y la gestión de las capas de la pila Bluetooth. Este perfil define cuatro roles de comunicación:

- ***Maestro***: puede comunicarse con uno o más esclavos.
- ***Esclavo***: puede comunicarse con un único maestro.
- ***Avisador***: anuncia paquetes con información.
- ***Escáner***: recibe los paquetes de los avisadores.

El dispositivo objetivo de este proyecto tomaría los roles de ***Esclavo*** y ***Avisador***, mientras que el *Monitor* que recibe los datos asumiría los papeles de *Maestro* y *Escáner*. Cada dispositivo Bluetooth LE implementa además un *Application Profile*. Todos los *Application Profile* requieren la implementación de GAP y suelen ser particuales según la aplicación desarrollada.

2.3.1.3 Attribute Protocol (ATT)

Se define un protocolo ATT para almacenar la información de un servicio de Bluetooth Smart. Dicha información se almacena en bloques llamados **atributos** que constan de:

- ***Handle*** (dirección de memoria de 16 bits)
- ***UUID*** (identificador único de 16 o 128 bits)
- ***Valor*** (contenido del atributo)

Además, el protocolo se basa en una arquitectura donde:

- El *servidor* recopila datos y los expone a través de atributos (Nuestro dispositivo)
- El *cliente* recolecta la información de uno más servidores (*Monitor*)

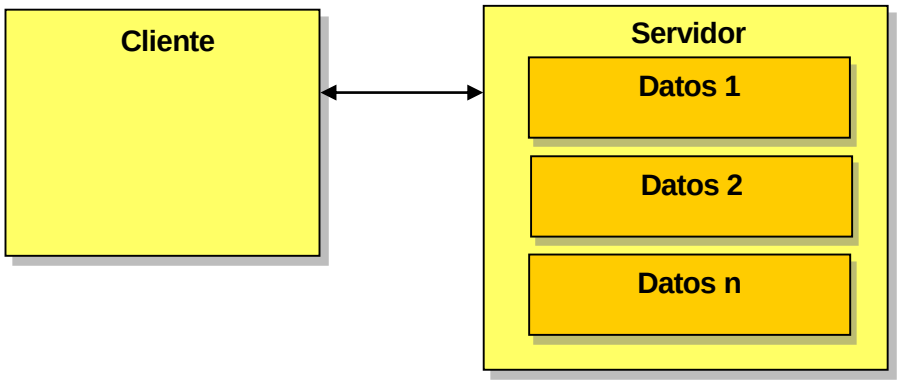


Figura 2-3. Roles de los dispositivos

Los ***atributos*** poseen las propiedades de *lectura*, *escritura*, o *lectura y escritura* asociadas. Realizar operaciones sobre los atributos puede requerir de otras como *autentificación*, *autorización* o *encriptación y emparejamiento*.

En base a dichas propiedades, el protocolo define una serie de **métodos**. Aquellos definidos para la comunicación de *Servidor a Cliente* son:

1. ***Notificación***: Envía el nuevo valor de un atributo (hasta 20 bytes).
2. ***Indicación***: Envía el nuevo valor y espera confirmación del cliente.
3. ***Error***: Envía información sobre un error.

2.3.1.4 Generic Attribute Profile

A partir del ATT se define el denominado GATT, el cual engloba los atributos en ***servicios*** y expone los valores de los datos en las denominadas ***características***, tal y como se muestra en la Figura 2-4.

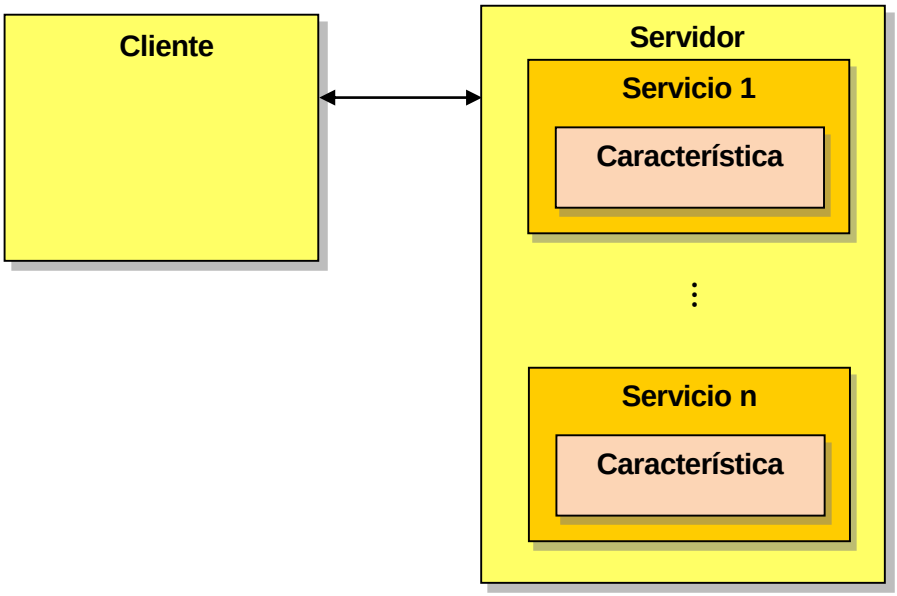


Figura 2-4. Arquitectura GATT

El **perfil GATT** permite que servidor y cliente puedan interactuar, de manera que los *servicios* del primero sean accesibles para el segundo y proceder así al intercambio de infomación. Dichos servicios se clasifican en:

- Servicios *primarios*, que ofrecen la funcionalidad principal del servicio.
- Servicios *secundarios*, que contienen funcionalidades adicionales al primario.

Las propiedades de los servicios se denominan *características* y se componen de los siguientes campos, como se muestra en la Figura 2-5:

- ***Declaración***, *propiedades* del valor (lectura y escritura), *UUID* y *handle*.
- ***Valor***, *dato* de la característica.
- ***Descriptor***, *información adicional* de la característica.

Los perfiles creados a partir de GATT son los utilizados por Bluetooth para el intercambio de información entre cliente y servidor. Un perfil GATT típico sigue la esctructura como en la Figura 2-6.

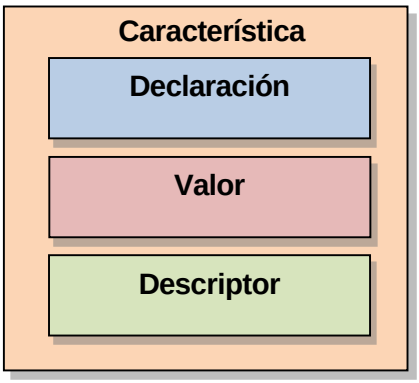


Figura 2-5. Composición de una característica

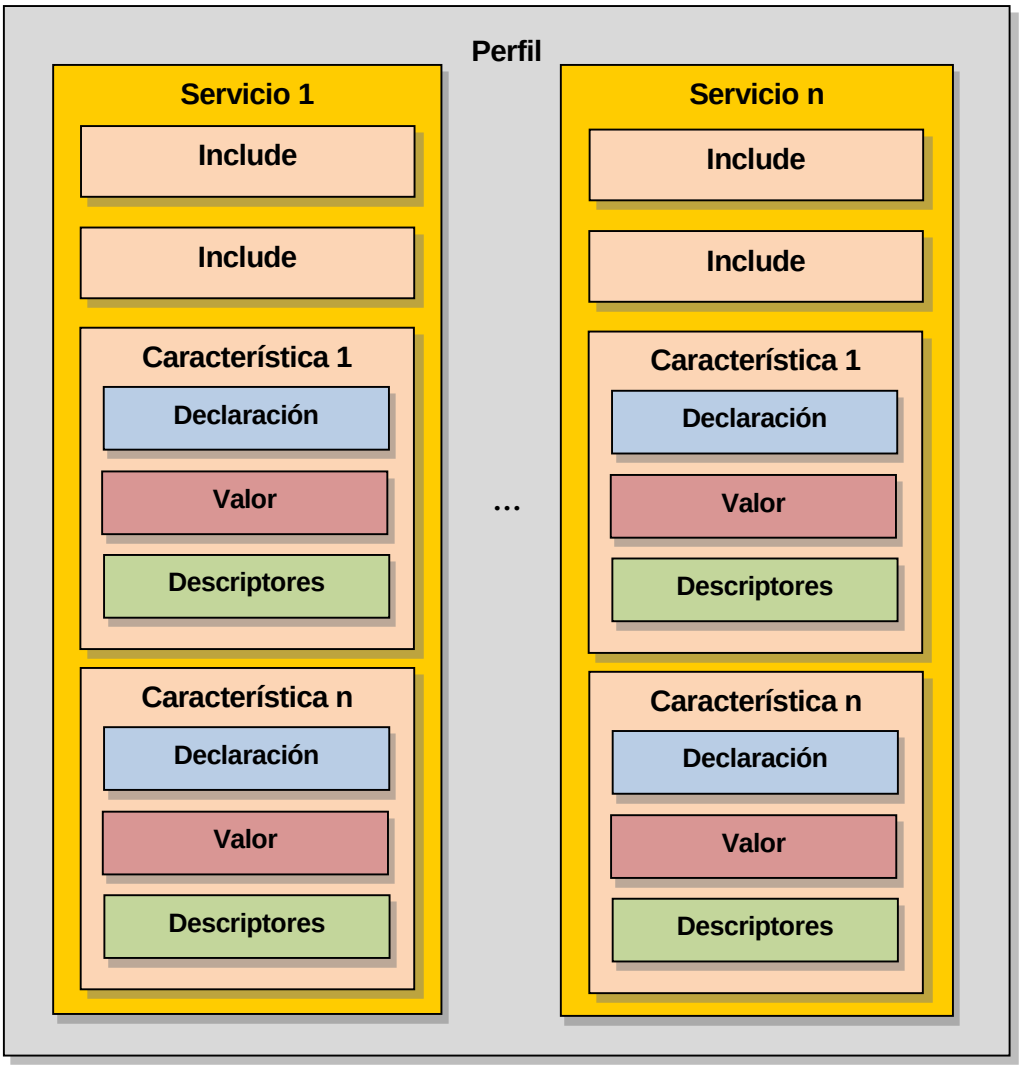


Figura 2-6. Niveles jerárquicos de un perfil GATT

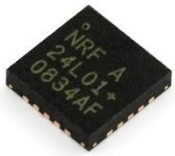
De ahora en adelante, se supondrá que el *Monitor* es un dispositivo **Android 5.0** que implementa la especificación Bluetooth Low Energy, que acabamos de explicar en detalle.

Una vez elegida la tecnología, el siguiente paso es decidir qué producto utilizar de los que la implementan. En el mercado se presentan dos categorías de dispositivos.

1. Los ***SoC*** (*System On a Chip*) son circuitos integrados que implementan la tecnología como tal, pero que requieren de toda una circuitería adicional para funcional correctamente: antena, componentes pasivos, osciladores. Además, el sistema final deberá obtener las certificaciones FCC, IC y ETSI.
2. ***Módulos preparados*** ofrecidos por diversos fabricantes realizan ese diseño, por lo que ya incorporan los *SoC* y toda la circuitería, además de haber pasado las certificaciones mencionadas para su uso en aplicaciones comerciales.

La Tabla 2–4 muestra una comparativa de diferentes transmisores que pueden adquirirse a día de hoy. Todos ellos ofrecen la gestión de la pila Bluetooth, GATT, L2CAP [15], abstrayendo en mayor o menor medida la configuración de Bluetooth LE al resto del sistema.

Tabla 2–4 Comparativa de transmisores Bluetooth Smart

	SoC (Circuitos integrados)		Módulos preparados	
	CC2541	nRF24L01+	BLE113-A	RN4020
				
Empresa	Texas Instruments	Nordic Semi.	Bluegiga	Microchip
Versión BLE	4.0	4.1	4.0	4.1
Flash	128 KB	No	128 KB	64 KB
RAM	8 KB	No	8 KB	No
Comunicación	UART (2 Mbps)	SPI (8 Mbps)	UART (2 Mbps)	UART (2 Mbps)
Potencia TX	0 dBm	0 dBm	0 dBm	0 dBm
Corriente TX	18.2 / 14.3 mA	11.3 mA	20.7 / 14.3 mA	16 mA
Sensibilidad RX	-93 dBm	-93 dBm	-93 dBm	-93 dBm
Corriente RX	20.2 mA	12.3 mA	21.9 mA	16 mA
Precio	5.71€	3.60€	13.63€	10.04€

De manera similar a lo que ocurría en el apartado de *Adquisición de datos*, la solución óptima sería utilizar un circuito integrado y diseñar la circuitería adicional acorde a nuestro dispositivo. No obstante en este caso, utilizar un **módulo preparado** se traduce en disminuir la carga de diseño, abstraer lo máximo posible el protocolo de comunicación utilizado y evitar la necesidad de pasar certificaciones específicas para nuestro dispositivo.

Atendiendo a la gran cantidad de documentación ofrecida y al soporte técnico que ofrece el fabricante, se ha decidido utilizar el módulo **BLE113** de *Bluegiga* [16] (que a su vez implementa el SoC **CC2541** de *Texas Instruments*). El laboratorio dispone del programador **CC Debugger** de *Texas Instruments* para poder programar el módulo. Ambos se muestran en la Figura 2-7.

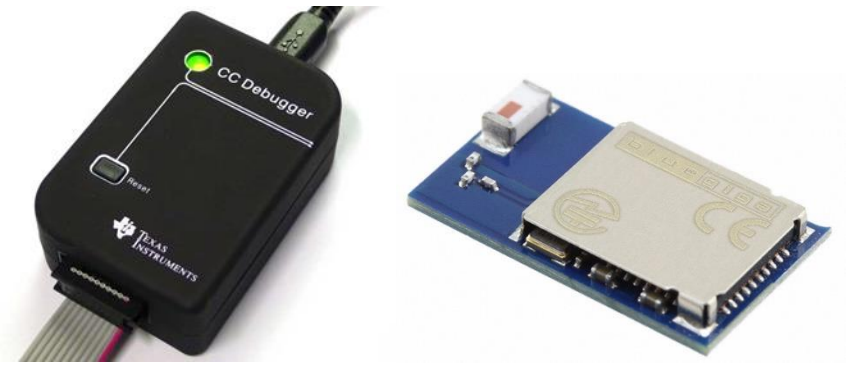


Figura 2-7. Texas Instruments CC Debugger y Bluegiga BLE113-A

A pesar de que el fabricante ofrece una placa de evaluación **DKBLE113** [16], se ha optado por utilizar el módulo directamente para realizar las pruebas. A la hora de comprar los componentes no se ha podido conseguir el modelo BLE113 y se ha adquirido el modelo ***BLE112*****,** de características similares. Según el fabricante, la migración del proyecto al modelo BLE113 es sencilla sin ser necesario modificar el código implementado [18].

2.3.2 Bluegiga Smart SDK

El fabricante nos indica que se pueden desarrollar aplicaciones siguiendo dos tipos de arquitecturas:

- Arquitectura ***standalone***: la pila Bluetooth, perfiles y código a ejecutar dentro del módulo.
- Arquitectura ***hosted***: la pila Bluetooth y los perfiles están en el módulo. La implementación del código debe realizarse en un dispositivo externo (**microcontrolador**, ordenador, etc).

Para ello ofrece un ***kit de desarrollo software*** (SDK) con las siguientes características:

- Implementación de la pila Bluetooth LE en el módulo.
- Pseudolenguaje ***BGScript*** para uso en arquitectura *standalone*.
- Protocolo ***BGAPI*** para el control externo en una arquitectura *hosted*.
- Librería ***BGlib*** en C para desarrollar aplicaciones que utilicen el protocolo *BGAPI*.
- Herramienta para la creación de perfiles basados en GATT.

Las dos arquitecturas permiten al módulo trabajar de manera indepediente ejecutando una aplicación interna escrita en lenguaje BGScript, o bien ser controlado desde un dispositivo externo a través del protocolo BGAPI.

Además, la programación del módulo permite establecer la configuración del dispositivo en la memoria interna haciendo uso de distintos archivos:

- *gatt.xml* Información sobre el perfil, servicios y características implementadas por el dispositivo.
- *hardware.xml* Configuración relacionada con los periféricos, módulos internos y opciones de energía.
- *config.xml* Parámetros de la aplicación relacionados con la conexión Bluetooth LE.
- *bgscript.xml* Código fuente de la aplicación BGScript a implementar en el dispositivo.

Estos ficheros, junto con un fichero principal *project.bgproj* pueden ser compilados e instalados en la memoria del módulo BLE113 mediante el uso de la herramienta ***BLE SW Update Tool***. Todo este kit es ofrecido de manera gratuita en la página web del fabricante [16].

2.3.2.1 BGScript

Este lenguaje de programación implementa todas las funcionalidades del estandar mediante comandos definidos en una API. BGScript permite comunicarse con periféricos a través de los puertos UART, SPI o I^2C del módulo. Su sintaxis similar a ***BASIC*** permite que el módulo realice actividades en función de eventos generados por el propio sistema.

Esta opción resulta interesante, ya que permitiría aunar los bloques *Control y procesamiento* y *Transmisión de la señal* en un sólo dispositivo. Sin embargo, según indicaciones del propio fabricante [18] cada instrucción ejecutada en BGScript equivale a un tiempo de ejecución de 1 milisegundo. Esto es incompatible con la restricción añadida en el apartado *Adquisición de datos*, donde el tiempo para el procesado y envío debe ajustarse a poco menos de 2 milisegundos.

2.3.2.2 BGAPI

Este protocolo binario diferencia tres tipos de mensajes: comandos, respuestas y eventos. Los ***comandos***, con una longitud máxima de 64 bytes, son enviados desde el dispositivo externo al módulo Bluetooth. Una vez los atiende, el módulo envía ***respuestas*** con información de interés al ordenante. Los ***eventos*** son otro tipo de mensajes mandados desde el módulo Bluetooth que informan de situaciones de interés relacionadas con la conexión Bluetooth o el propio sistema.

La principal ventaja de BGAPI es que desprende al módulo de carga de procesamiento, incrementando el rendimiento de la conexión a costa de precisar de un dispositivo externo para controlar el sistema. En la Figura 2-8 se muestran las primitivas utilizadas en este protocolo para la comunicación entre la aplicación (alojada en el dispositivo externo) y la pila Bluetooth (módulo).

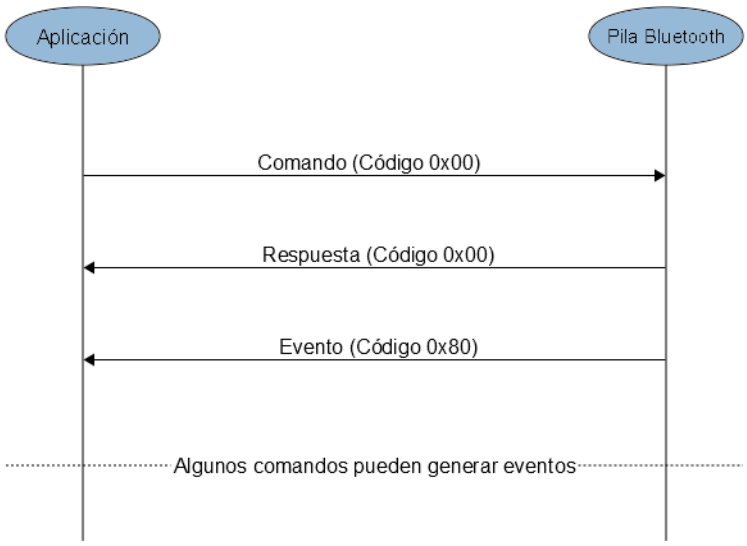


Figura 2-8. Protocolo de comunicación BGAPI

2.3.2.3 Configuración de la conexión

Los ***parámetros de conexión*** son otro factor importante en un enlace Bluetooth:

- ***Intervalo de conexión***: Cada cuánto tiempo el *cliente* solicita datos al *servidor*. Puede tomar un valor entre 7.5 milisegundos y 4 segundos.
- ***Latencia***: Establece un tiempo por el cual el *servidor* puede ignorar peticiones del *cliente*. Esto permite al *servidor* mantenerse en un estado de baja energía si así lo precisa.
- ***Timeout de conexión***: Tiempo a partir de la última transmisión para el que el enlace se considerará desconectado.

La ***tasa de transmisión*** según la especificación de Bluetooth LE es de 1 Mbps para la modulación RF. Debido a la sobrecarga de la propia especificación, la tasa teórica máxima apenas alcanza los **270 Kbps**. Existen otros aspectos que reducen este número, pero dependen de cada implementación concreta. Además hay que tener en cuenta que BLE sólo permite un máximo de **20 bytes por envío** (valor de un carácterística).

Una vez conocidas las principales características y restricciones de Bluetooth Smart y el módulo BLE113, es necesario ver cómo afectan a nuestro diseño particular.

1. Nuestro dispositivo será un ***Servidor*** que implementará un ***perfil*** *personalizado* con una serie de ***servicios***, algunos *estandarizados* y otros de creación propia. El monitor a su vez hará las veces de *Cliente*.
2. *Bluegiga* nos hace saber [18] que la **tasa máxima** alcanzada con sus módulos es de **60 Kbps**. Sabiendo que las muestras de dos canales del ADS1292R equivalen a 6 bytes, se podrían enviar hasta 3 muestras distintas de una sola vez, esto es, cada 6 milisegundos. Suponiendo que se utilizan los 20 bytes, esto equivale a una tasa de envío de $160\ bits/6\ ms = \mathbf{26.67}\boldsymbol{kbps}$, que no supera la máxima indicada por el fabricante.
3. El **intervalo de conexión** debe ser lo suficientemente pequeño debido a la moderada tasa de tranmisión de nuestra aplicación. El valor mínimo de 7.5 milisegundos será el punto de partida a partir del cual se probarán valores más elevados. El resto de *parámetros de conexión* se decidirán más adelante, en la etapa de implementación.

2.4 Alimentación y carga inalámbrica

Otra parte importante en el diseño será el sistema que alimente a todos los dispositivos. Como ya conocemos las características principales del resto del sistema se puede buscar una solución acorde al mismo. En la Tabla 2–5 se presentan las características de alimentación de cada uno de estos dispositivos.

Tabla 2–5 Requisitos de alimentación de los circuitos integrados

	ADS1292R	**PIC18F24K20**	**BLE112**
Tensión (Analógica)	2.7 – 5.25 V	–	2 – 3.6 V
Tensión (Digital)	1.8 – 3.6 V	1.8 – 3.6 V	2 – 3.6 V
Corriente consumida	325 µA	3-5 mA	20 mA

De la misma se desprende que la alimentación puede dividirse en parte analógica y digital (Cada parte podría ser alimentada a una tensión diferente, reduciendo el consumo total del circuito). Sin embargo, para disminuir la dificultad en la implementación se ha optado por utilizar una tensión común a todo el circuito.

En nuestro caso, haremos uso de una batería recargable de litio, con una tensión nominal de 3.7 V, pudiendo alcanzar valores de **4 – 4.2 V** cuando se encuentra al máximo de su capacidad. Por tanto, se plantea como una necesidad el uso de un **regulador de voltaje**, que sirva tanto para obtener una **tensión de 3V** como para **estabilizarla** cuando se produzcan picos de consumo por parte de los componentes.

Existes diferentes alternativas en el mercado, pero se ha optado por un modelo conocido en el laboratorio por su especial utilidad en aplicaciones sensibles al ruido como esta. El **TPS73201** de *Texas Instruments* [10] permite obtener una tensión estable a partir de una tensión entre 1.7 y 5.5 V.

A partir de la corriente consumida se puede estimar la capacidad mínima de la batería necesaria, sabiendo que necesitamos una autonomía de **24 horas**. Teniendo en cuenta este número y que la batería debe ser de tamaño reducido, se ha escogido el modelo OL-40B de Energizer, con una capacidad de 710 mAh.

$$Capacidad = (20\ mA + 5\ mA + 0.325\ mA) * 24h\ \backsimeq 600\ mAh$$

2.4.1 Estándar de carga inductiva Qi

El siguiente paso en este apartado del diseño es hallar una solución para el sistema de carga inalámbrica, también denominada ***carga inductiva***. Antes de realizar la búsqueda, es conveniente conocer el funcionamiento básico de esta tecnología.

El estándar de carga por inducción ***Qi*** establece un protocolo de ***transmisión de energía*** entre dos dispositivos, un transmisor y un receptor, por medio de ***bobinas*** que se encuentran a distancias de 5 a 40 mm.

La principal ventaja de ser un protocolo estándar es la ***intercompatibilidad***, esto es, que un transmisor de cualquier fabricante podrá funcionar sin mayor problema con el receptor de cualquier otro. Además, un integrado que cumpla la especificación permite utilizar esta tecnología sin necesidad de programación adicional. En la Figura 2-9 se muetra un diagrama de bloques que explica de manera sencilla su funcionamiento.

La comunicación entre transmisor y receptor puede interpretarse como: a) una señal modulada en amplitud (AM) a 2 kHz para transmitir la energía de TX a RX y b) una portadora con una frecuencia variable entre 100 y 200 kHz para la comunicación de RX a TX. Además, como es utilizado para ambas operaciones, el sistema final precisa de un único medio de transmisión.

Las funciones y comportamientos que define el protocolo son los siguientes:

- El sistema de carga (***transmisor***) no consumirá una cantidad significativa de potencia en reposo (cuando no haya ningún dispositivo situado sobre él).
- El *transmisor* puede detectar la presencia de un objeto situado sobre el mismo y comprobar si es un dispositivo ***receptor*** que cumple la especificación Qi.
- Una vez el dispositivo (*receptor*) se ha situado, el *transmisor* enviará una cantidad variable de potencia basado en sus propias especificaciones.
- El *receptor* comunica en todo momento la potencia que necesita utilizando el mismo acoplamiento magnético con el *transmisor* como canal de comunicación.

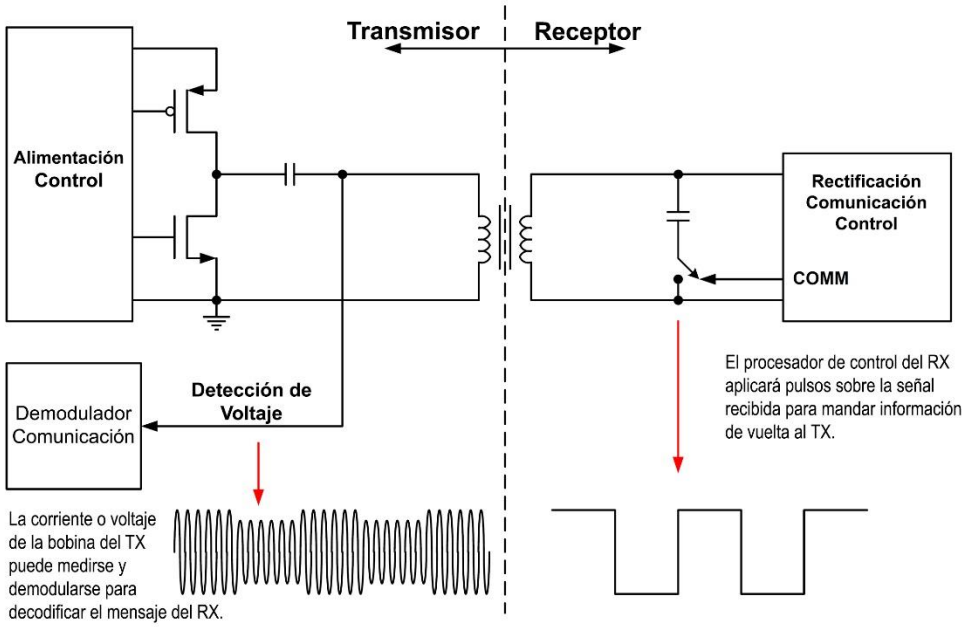


Figura 2-9. Diagrama de bloques implementado en el protocolo Qi

En el mercado existen diversos fabricantes que están apostando por ofrecer soluciones de este tipo, pero tanto por la madurez como por la cantidad de información disponible, la serie **bqTESLA** de *Texas Instruments* es la más adecuada en este caso.

De dicha serie, el receptor **bq51050B** [10], está preparado para la carga de baterías de litio como la nuestra. Además, por sus características, es un módulo autónomo que no tiene que ser controlado, liberando al microcontrolador de esta tarea.

Para poder comprobar el funcionamiento del dispositivo se ha adquirido una placa de evaluación **bq51050BEVM-764** del mismo fabricante [10], como la que aparece en la Figura 2-10. Esta placa contiene el receptor que necesitamos junto con toda la circuitería necesaria para su funcionamiento, incluida la bobina de recepción y una serie de jumpers y potenciómetros para configurarlo. En cuanto al transmisor, basta con adquirir un cargador de cualquier fabricante que cumpla la especificación, como el cargador genérico que aparece en la misma Figura 2-10.

Figura 2-10. Texas Instruments bq51050BEVM-764 y Cargador Qi genérico

3 IMPLEMENTACIÓN DEL SISTEMA

Una vez establecidos todos los pasos necesarios a seguir en el diseño es hora de llevar a cabo una implementación real. Para comprobar la viabilidad del sistema se desarrollará un ***primer prototipo***, usando una placa de pruebas y probando cada uno de las secciones por separado, en la medida de lo posible.

En base a esta primera prueba, se propondrán mejoras al diseño inicial y se implementará un ***segundo prototipo*** fabricado en circuito impreso y con componentes de menor tamaño. Este prototipo final será el utilizado en las pruebas realizadas más adelante.

3.1 Primer prototipo

3.1.1 Texas Instruments ADS1292R

Para comenzar el análisis de esta parte haremos uso de la placa de evaluación **ADS1292ECG-FE** como la que aparece en la Figura 3-1. Esta dispone de un software de evaluación que nos ayudará a entender el proceso a seguir para obtener las muestras y cómo configurarlo correctamente.

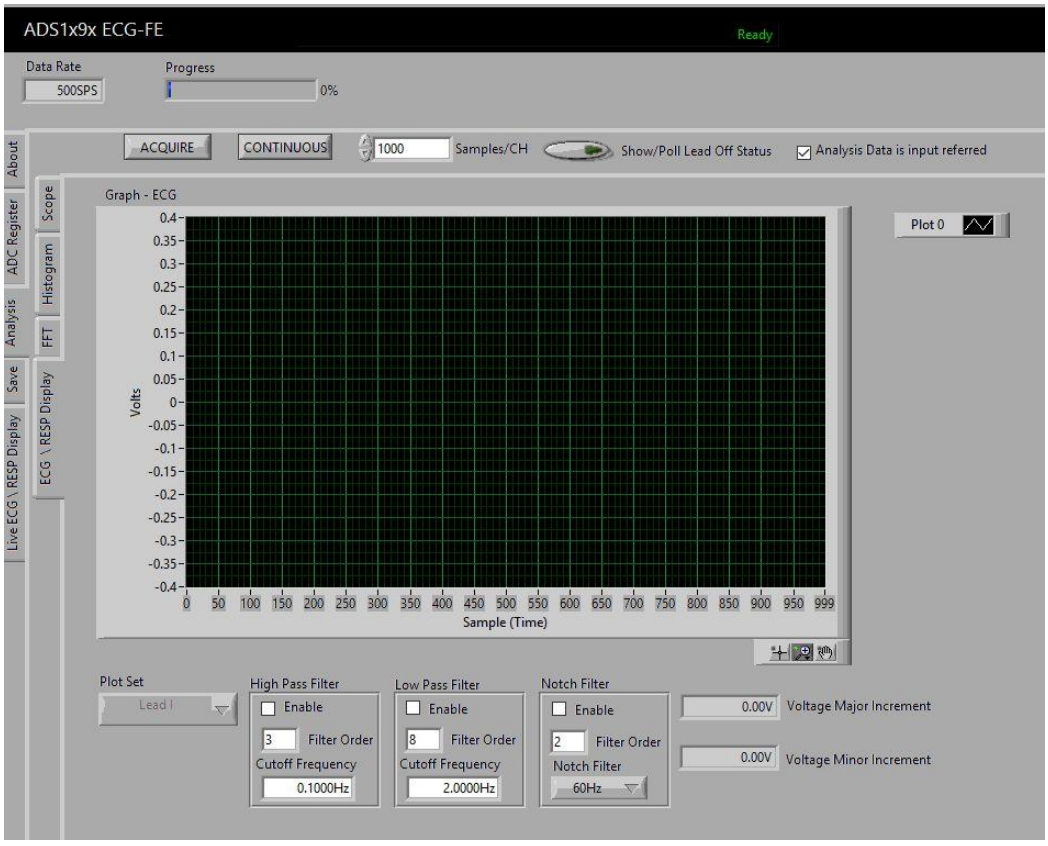


Figura 3-1. Software de la placa ADS1292ECG-FE

Haciendo uso del software de la Figura 3-1 y en base al diagrama de bloques de la Figura 3-2 junto con más información obtenida del datasheet del módulo ***ADS1292R*** [10] podemos extraer los datos y referencias que nos servirán a la hora de implementar el circuito y escribir el código correspondiente al microcontrolador.

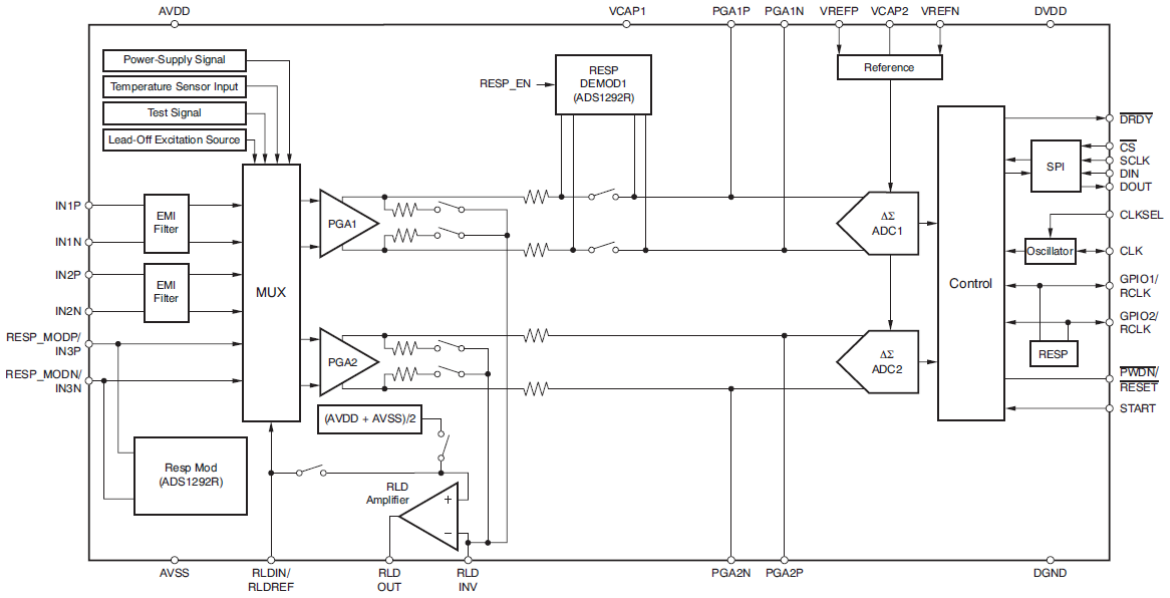


Figura 3-2. Diagrama de bloques funcional del ADS1292R

Las principales características del módulo son las siguientes:

- ***Canal 1*** (par de pines **IN1x**), correspondiente a la **respiración**. La señal de excitación que genera la *bioimpendancia eléctrica* tiene una frecuencia de 32 o 64 kHz, por lo que se debe implementar un **filtro HP**.
- ***Canal 2*** (par de pines **IN2x**), correspondiente al **electrocardiograma**. La señal de interés se encuentra a baja frecuencia (0.01 – 150 Hz). Por tanto, se necesitará un **filtro LP**.
- ***Amplificador RLD***, para reducir la interferencia del modo común. Es necesario implementar circuitería pasiva para que funcione correctamente.
- ***Reloj interno a 512 kHz***, con posibilidad de facilitar uno externo. El uso del interno reduce el consumo del circuito, por lo que utilizar uno externo queda descartado.
- ***Control externo via hardware***, mediante el uso de los pines *DRDY*, *CS*, *START* y *PWDN/RESET*.
- ***Comunicación via SPI***, mediante el uso de los pines *SCLK*, *DIN* y *DOUT*.

En vez de implementar el circuito, en este primer prototipo se podría utilizar la placa de pruebas **ADS1292ECG-FE** y los pines accesibles del integrado para su conexión con el microcontrolador. Sin embargo, al no poder disponer de ella a tiempo completo en el laboratorio se ha decidido utilizar parte un prototipo utilizado en un proyecto más antiguo. Este prototipo se obtuvo a partir del esquemático de la propia placa de evaluación, por lo que su funcionamiento debe ser equivalente.

Sólo queda conocer la forma de obtener los datos desde el circuito integrado a través de SPI. Para ello acudimos a la documentación, de donde se pueden extraer varios diagramas como los que aparecen en la Figura 3-3 y la Figura 3-4.

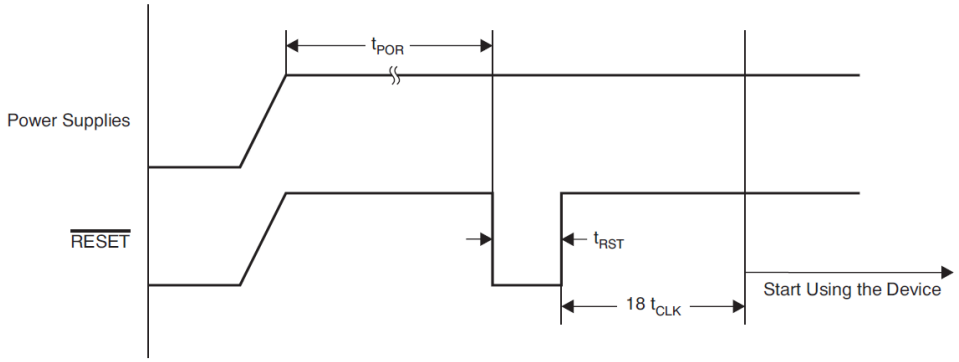


Figura 3-3. Secuencia de inicio del ADS1292R

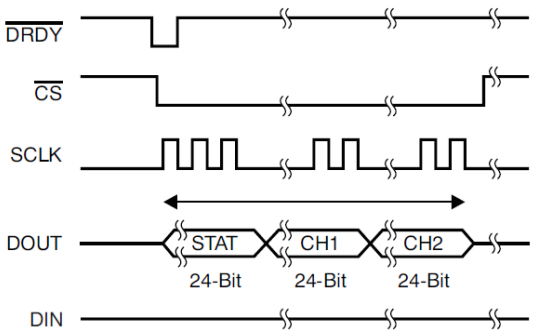


Figura 3-4. Salida del bus SPI del ADS1292R

Los pasos a seguir para poder utilizar el dispositivo después de ser conectado a la alimentación son:

- ***Reiniciar*** el dispositivo por la entrada ***RESET***, tal y como se hace en la Figura 3-3.
- ***Configurar*** los registros internos, estableciendo un estado por defecto.
- ***Iniciar*** o ***detener*** la captura y envío de muestras a la frecuencia establecida en la comunicación. El diagrama temporal del envío queda representado en la Figura 3-4.

Existen dos formas de controlar el envío de muestras, suponiendo que el **modo continuo** se encuentra activado, esto es, el envío de cada muestra capturada sin necesidad de pedirla explícitamente:

- Mediante la señal ***START***. Siempre que se encuentre a nivel bajo las muestras serán enviadas por el bus SPI. Cuando se quiera parar el proceso, basta con ponerla a nivel alto.
- Mediante ***comandos*** por SPI. Haciendo uso del mismo bus, el microcontrolador puede mandar las órdenes de inicio o parada mediante comandos preestablecidos.

Dado que puede ser necesario mandar más comandos además de los de inicio y parada se utilizará el segundo método, liberando al microcontrolador de utilizar una salida para controlar la señal *START*.

El chip ADS1292R permite una configuración flexible del proceso de muestreo de los canales y otras funciones que ofrece el propio chip. Por ello es conveniente establecer las diferentes opciones de configuración desde el punto de vista del **Cliente**, por ejemplo:

- ***Tasa de transmisión***: 125, 250, 500 o 1000 muestras/s.
- ***Ganancia de amplificación***: 1, 2, 3, 4, 6, 8, 12.
- ***Corriente de detección del sensor***: 6 o 22 nA.
- ***Señal de Test***: a enviar en lugar de las señales biométricas.

3.1.2 Bluegiga BLE112-A

Como ya vimos en la parte de diseño se ha decidido utilizar el módulo comercial directamente en el primer prototipo. Para comenzar esta parte es necesario realizar un circuito de adaptación que haga los pines accesibles para el resto de componentes del sistema como el de la Figura 3-5.

Figura 3-5. Software de evaluación de Bluegiga

Para la programación del módulo se hará uso del programador CC Debugger. En la comunicación entre módulo y microcontrolador se utilizará ***BGAPI***, un protocolo binario definido en la documentación. Para poder estudiarlo y así poder implementarlo más adelante en el microcontrolador se ofrece a su vez el software **bleGUI**. Tanto software como documentación pueden encontrarse en la página web del fabricante correspondiente al *BLE112* [16].

Haciendo uso de la documentación y de la aplicación bleGUI podemos saber qué mensajes BGAPI pueden ser de utilidad en la comunicación entre módulo Bluetooth y microcontrolador.

Por un lado los *eventos,* mensajes enviados desde el módulo hacia el microcontrolador cuando:

- ***ble_evt_system_boot***: el sistema se ha iniciado y está listo para comunicarse.
- ***ble_evt_connection_status*** y ***ble_evt_connection_disconnected***: el *Cliente* ha establecido o ha finalizado la conexión con el dispositivo, respectivamente.
- ***ble_evt_hardware_io_port_status***: se ha producido algún cambio en el puerto del módulo.
- ***ble_evt_attributes_value***: el *Cliente* ha escrito el valor de una característica de la base de datos GATT.

Además de *comandos*, órdenes enviadas desde el microcontrolador al módulo para que:

- ***ble_cmd_gap_set_mode*** y ***ble_cmd_gap_set_adv_parameters***: el dispositivo sea visible al Cliente y este pueda conectarse.
- ***ble_cmd_connection_update***: se actualicen los parametros de la conexión en curso.
- ***ble_cmd_attributes_write***: se actualice el valor de una característica en la base de datos GATT y se notifique/indique al *Cliente* si es necesario.
- ***ble_cmd_attributes_send***: se notifique/indique al *Cliente* el valor de una característica sin escribirla en la base de datos GATT.

El módulo BLE112 permite **control de flujo por hardware** (RTS, CTS) en la comunicación UART. Sin embargo, el PIC18F no lo implementa y se ha descartado su simulación por software. En su lugar, se utilizará el ***modo paquete*** al realizar envíos al módulo, esto es, indicar la longitud total con un byte al principio de la transmisión.

Existen diferentes modos de consumo presentados en la Tabla 3–1. Salvo el *Modo Activo*, el resto de ***modos de bajo consumo*** restringen el uso de componentes del módulo a mayor numeración. Sabiendo que cualquiera de estos modos impide hacer uso de la UART [18], nuestro sistema deberá mantenerse en en ***Modo Activo*** para la comunicación con el microcontrolador.

Sí resulta interesante mantenerse en un modo de bajo consumo mientras no haya datos a transmitir. Para ello se dispone de una **interrupción hardware** externa que permite forzar el cambio al *Modo Activo* y volver a uno de bajo consumo más tarde.

Tabla 3–1 Modos de actividad del chip BLE112

	Activo	**PM1**	**PM2**	**PM3**
Consumo de corriente	6.7 mA	235 µA	0.9 µA	0.4 µA
Transición a "Activo"	–	4 µs	120 µs	120 µs

Siguiendo las recomendaciones en la web del fabricante [18], el dispositivo incluirá dos perfiles. El primero y obligatorio es el ***Generic Access Profile***, incluído en todos los dispositivos:

- ***Device name***, con una cadena fija *WIER Monitor*.
- ***Appearance***, con un valor fijo correspondiente a *Heart Rate belt*.

Además se incluirá un **perfil** personalizado con los siguientes servicios y características:

- **Device Information**, con información adicional del dispositivo.
  - ***Manufacturer Name String***, con una cadena fija *BCNdevices*.
  - ***Model Number String***, con una cadena fija *WIER Monitor.*
  - ***Firmware Revision String***, con una cadena variable a modificar.
- **Battery Service**, servicio para indicar la batería.
  - ***Battery Level***, con un valor variable entre 0 y 100 a modificar.
- **EKG & RI Service**, servicio para enviar las muestras del ADS1292R.
  - ***Multisample measurement***, contiene las muestras e información adicional.
  - ***Status / Configuration,*** valor editable desde el Monitor para configurar el dispositivo.

El servicio **EKG & RI** consta de dos *características*. La primera de ellas, ***Multsisample measurement*** sigue una estructura como la que aparece la Figura 3-6. Tiene un tamaño de ***20 bytes***, posee la propiedad de ***lectura*** y permite la suscripción mediante ***notificación***. En su interior se almacenan 3 muestras de cada uno de los canales, un total de 18 bytes. Los dos bytes restantes pertencen a un campo denominado ***ID / Contacto*** que contiene tanto un número de secuencia como un indicador de contacto del sensor. Su estructura queda reflejada en la Figura 3-7.

Figura 3-6. Estructura de la característica Multisample Measurement

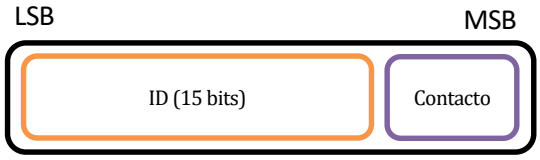


Figura 3-7. Estructura del campo ID / Contacto

La segunda *característica* es la llamada **Status / Configuration**, con una estructura como la de la Figura 3-8. Tiene un tamaño de **1 byte**, propiedad de ***escritura*** y permite establecer la configuración de la transmisión con las opciones que se definieron en el apartado 3.1.1 (*Texas Instruments ADS1292R*).

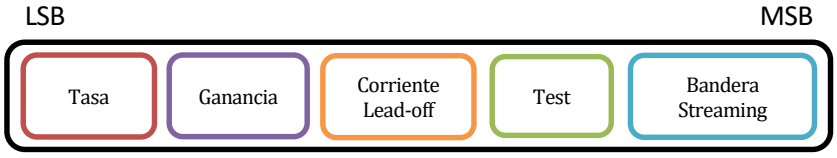


Figura 3-8. Estructura de la característica Status / Configuration

Se ha añadido una opción adicional, **Streaming Flag**, para que el cliente controle el inicio y parada del envío de datos. Una vez sea escrita la característica, el microcontrolador deberá enviar las muestras a través de la **Multisample Characteristic** con la nueva configuración aplicada.

Para obtener información detallada sobre el perfil y servicio creados para esta aplicación se puede acudir al **Anexo A**, con un formato de documentación similar al que se ofrece para los perfiles y servicios estandarizados por el *Bluetooth SIG*.

Una vez hemos dedicido cómo configurar el módulo Bluetooth, sólo queda programarlo adecuadamente. Para ello, el fabricante aporta una serie de documentos *Profile Toolkit Guide* y *Module Configuration Guide* [16], con los que se puede definir tanto los perfiles utilizados como la configuración en ficheros XML. Dichos ficheros se encuentran en el **Anexo B** y están escritos en base a la configuración que acabamos de presentar. La herramienta BLE SW Update tool junto con el programador CC Debugger permitirán escribir estos archivos en el módulo Bluetooth de una manera rápida y sencilla.

3.1.3 Microchip PIC18F45K20

El microcontrolador es la parte de mayor carga de diseño de la implementación, ya que su funcionamiento dependerá a su vez de los componentes **ADS1292R** y **BLE112**.

Para poder entender el funcionamiento y acelerar la implemetación del código a posteriori se han realizado las pruebas pertenecientes a la documentación de la placa de desarrollo DM164130-4 como la de la Figura 2-2.

El código de esta primera versión puede encontrarse en su totalidad en el **Anexo C**. Dicho código pertenece al segundo prototipo, pero no ha requerido ningún cambio relevante a mencionar de uno a otro. En el mismo puede encontrarse una definición más detallada de las funciones implementadas.

Para la comunicación con el módulo BLE112-A mediate BGAPI, se ha utilizado la librería **BGlib** en C proporcionada por el fabricante [16] y adaptada para que funcionase en nuestro microcontrolador y compilador particulares.

La configuración del PIC18F se ha implementado en función de los Requisitos y Especificaciones del Proyecto, además de las restricciones adicionales derivadas del uso de los módulos ADS1292R y BLE112-A:

- **Oscilador interno** a 8 MHz con PLL activado → Frecuencia de reloj a 32 MHz
- **Conversor analógico-digital**, utilizando el potenciómetro integrado (simulando la batería)
- **Timer 0** activado con desbordamiento cada 1 segundo
- **Módulo UART** con una tasa de transmisión de 1 Mbps
- **Módulo SPI** con una tasa de transmisión de 512 Kbps o 2 Mbps

Además se han activado una serie de interrupciones tanto internas como externas. Las interrupciones de prioridad baja podrán ser interrumpidas por la de prioridad alta y no al contrario:

- **Recepción UART**: (*Prioridad alta*) Se recibirá el mensaje completo del módulo BLE112-A y se atenderá la petición antes de volver a la ejecución normal del código.
- **Flanco descendiente en RB1**: (*Prioridad baja*) Se recibirán las muestras del módulo ADS1292R, para su posterior procesamiento, en caso de que el envío esté activado y la señal *DRDY* esté conectada al pin RB1.
- **Desbordamiento de Timer 0**: (*Prioridad baja*) Se incrementará el temporizador del sistema 1 segundo y se realizarán actividades adicionales que dependan del mismo.

Para que todo el sistema funcione coordinadamente es necesario definir una máquina de estados para controlarlo, la cual consta de los siguientes estados:

- **Advertising** (***Fast Connection***): El módulo Bluetooth se encuentra disponible emitiendo avisos con frecuencia.
- **Advertising** (***Power Saving***): El módulo Bluetooth se encuentra disponible emitiendo avisos con intervalos de tiempo más elevados para reducir el consumo.
- **Connection**: Se realizan operaciones previas al establecimiento del enlace con el Cliente.
- **Disconnection**: Se realizan operaciones posteriores al cese del enlace con el Cliente.
- **Linked**: El enlace se encuentra activo.
- **Streaming**: Las muestras del ADS1292R se notifican al Cliente a través del perfil Bluetooth.
- **Configuration**: El cliente quiere una nueva configuración para el envío.

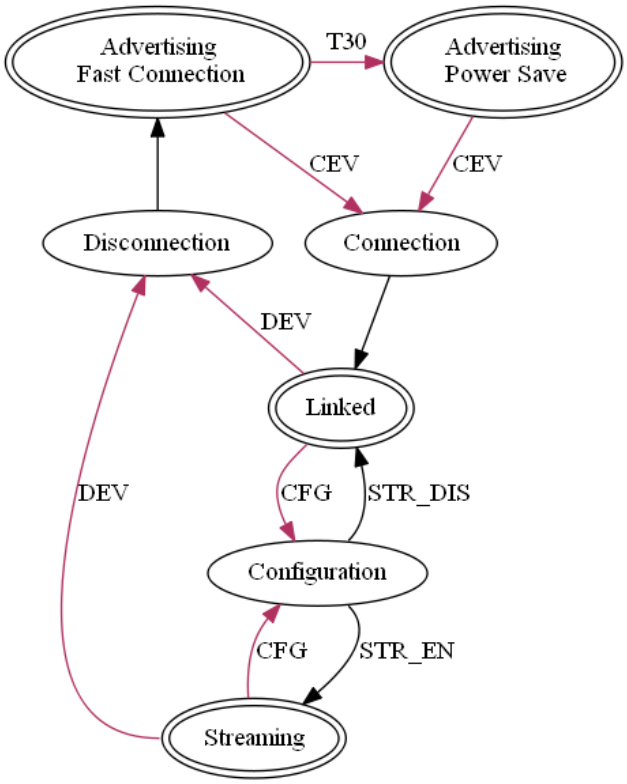


Figura 3-9. Máquina de estados del sistema

El diagrama queda representado por la Figura 3-9 y las condiciones de cambio de estado en la Tabla 3–2. A su vez, dichos estados han sido clasificados en:

- Estados **inestables**: Realizan una serie de tareas antes de cambiar a otro estado diferente.
- Estados **estables** (*doble elipse*): Seguirán funcionando indefinidamente hasta que una interrupción provoque un cambio de estado.

Tabla 3–2 Cambios de estado de la máquina de estados

Siglas	Significado
CEV	Evento de conexión
DEV	Evento de desconexión
CFG	Configuración recibida
STR_EN	Envío activado
STR_DIS	Envío desactivado
T30	Temporizador alcanza 30 s.

En base a esta máquina de estados, se presenta un diagrama de flujo simplificado de las operaciones que realiza el microcontrolador en cada uno de esos estados. Comenzando con la Figura 3-10, el sistema prepara todo lo necesario antes de dar inicio a la máquina de estados, que comienza con el estado *Advertising (Fast Connection)*.

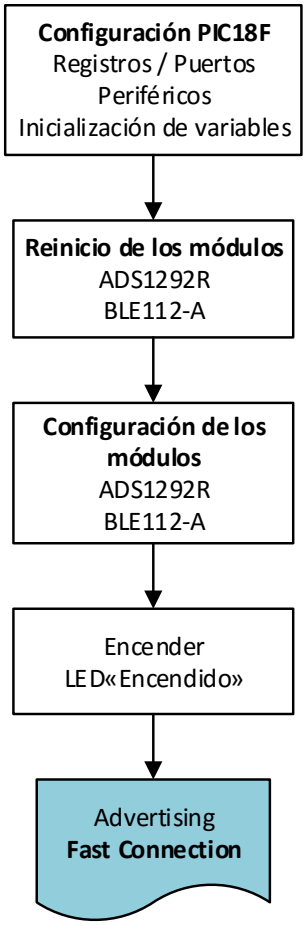


Figura 3-10. Diagrama de flujo del inicio del sistema.

En este nuevo estado el módulo Bluetooth actúa de avisador realizando anuncios con un intervalo de tiempo reducido entre ellos, facilitando una conexión rápida con el Monitor a costa de un mayor consumo.

En el diagrama de la Figura 3-11 se muestran además las interrupciones relevantes para este estado:

- Interrupción de Timer 0: A los 30 segundos se cambiará al estado *Advertising (Power Saving)*.
- Nuevo cliente: Se cambiará al estado *Connection*.

A su vez, en la Figura 3-12 se muestra el estado *Advertising (Power Saving)*, el cual sólo finaliza cuando se produce una interrupción relacionada con la conexión de un nuevo cliente, pasando al estado *Connection*.

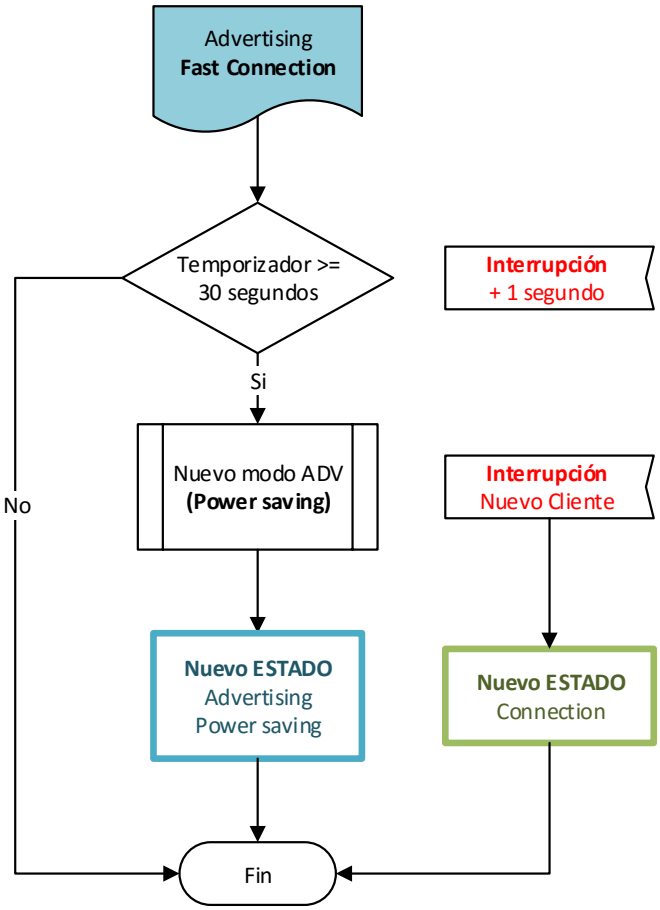


Figura 3-11. Diagrama de flujo del estado Advertising (Fast Connection)

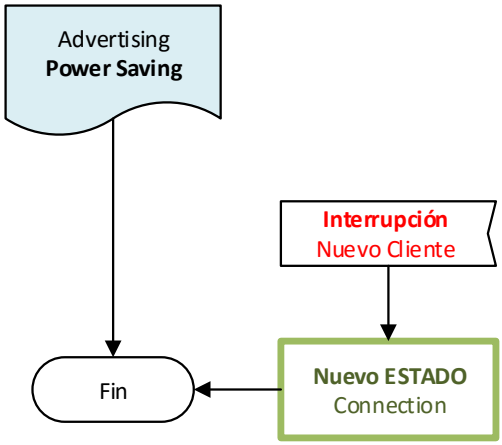


Figura 3-12. Diagrama de flujo del estado Advertising (Power Saving)

Dos de los estados que hemos denominado **inestables** son *Connection* y *Disconnection*. Su principal función es la de preparar al sistema antes de volver al estado estable.

Del estado *Connection* se pasará al estado *Linked,* no sin antes actualizar los parámetros de conexión de la conexión Bluetooth establecida.

A partir del estado *Disconnection* a su vez, se pasará al estado *Advertising (Fast Connection)* dejando los módulos ADS1292R y BLE112-A en un estado equivalente al del inicio de la máquina de estados.

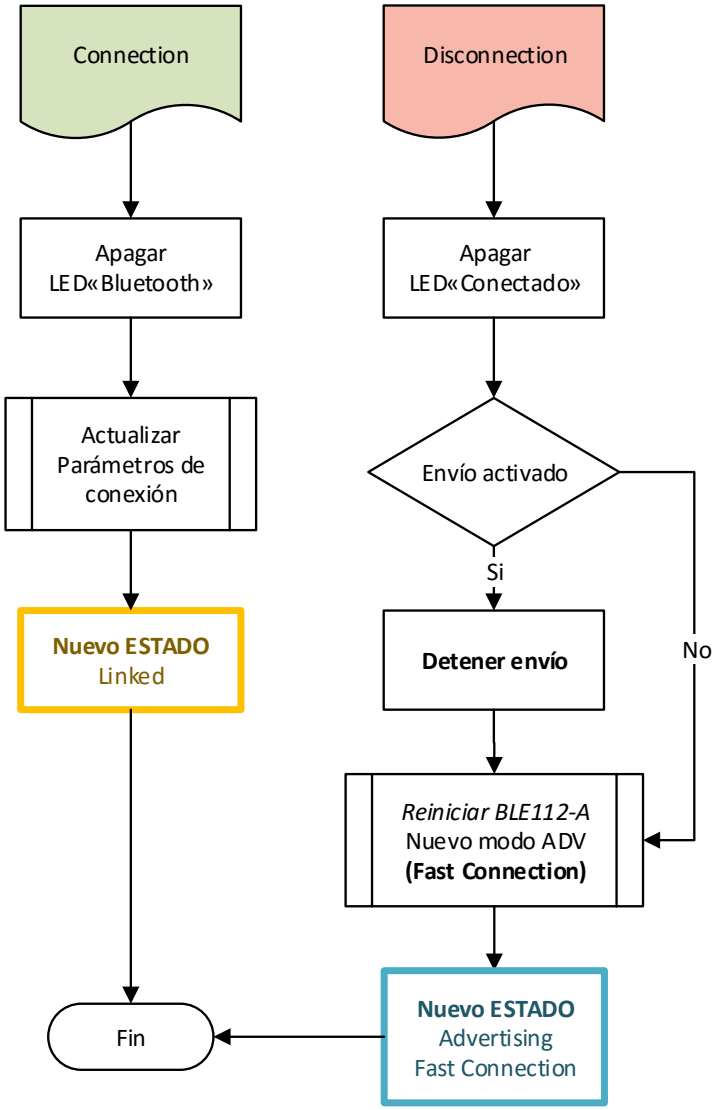


Figura 3-13. Diagrama de flujo de los estados Connection y Disconnection

Una vez se ha establecido el enlace con el Monitor de manera apropiada, el estado *Linked* será un estado intermedio en el que la conexión se ha producido pero el envío está desactivado. Se definen dos interrupciones posibles que pueden cambiar a un nuevo estado:

- Nueva configuración: Se cambiará al estado *Configuration*.
- Cliente desconectado: Se cambiará al estado *Disconnection*.

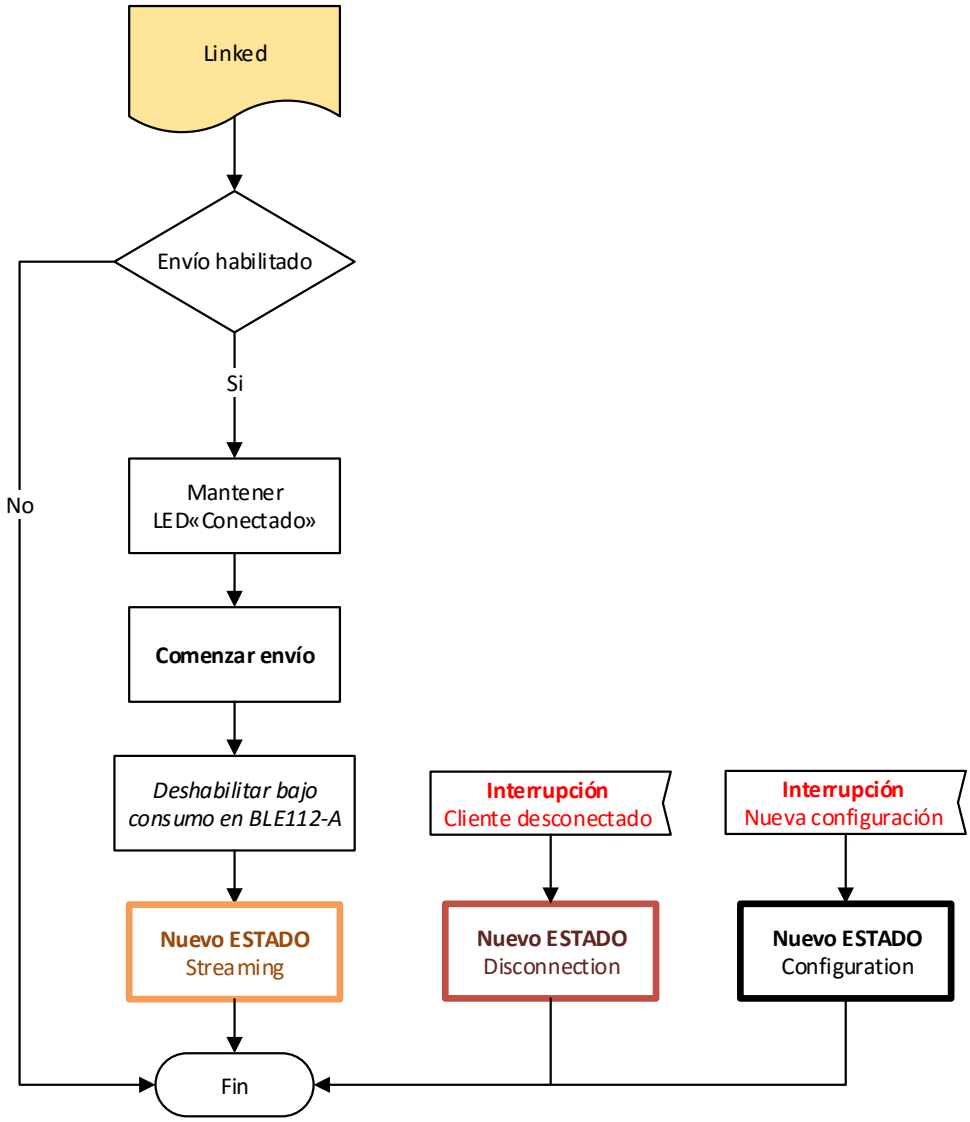


Figura 3-14. Diagrama de flujo del estado Linked

El cambio al estado *Streaming* también está gobernado por una interrupción de manera *indirecta*, ya que dependerá del análisis de la configuración enviada por el Cliente y procesada en el estado *Configuration*.

El estado inestable *Configuration* reflejado en la Figura 3-14 será el encargado de leer la palabra de configuración enviada por el cliente a través de la escritura de la característica *Status / Configuration*. Dicha palabra de configuración tiene la estructura definida en la Figura 3-8. La *Bandera Streaming* permitirá, tras volver al estado *Linked*, permanecer en el mismo o cambiar al estado *Streaming*.

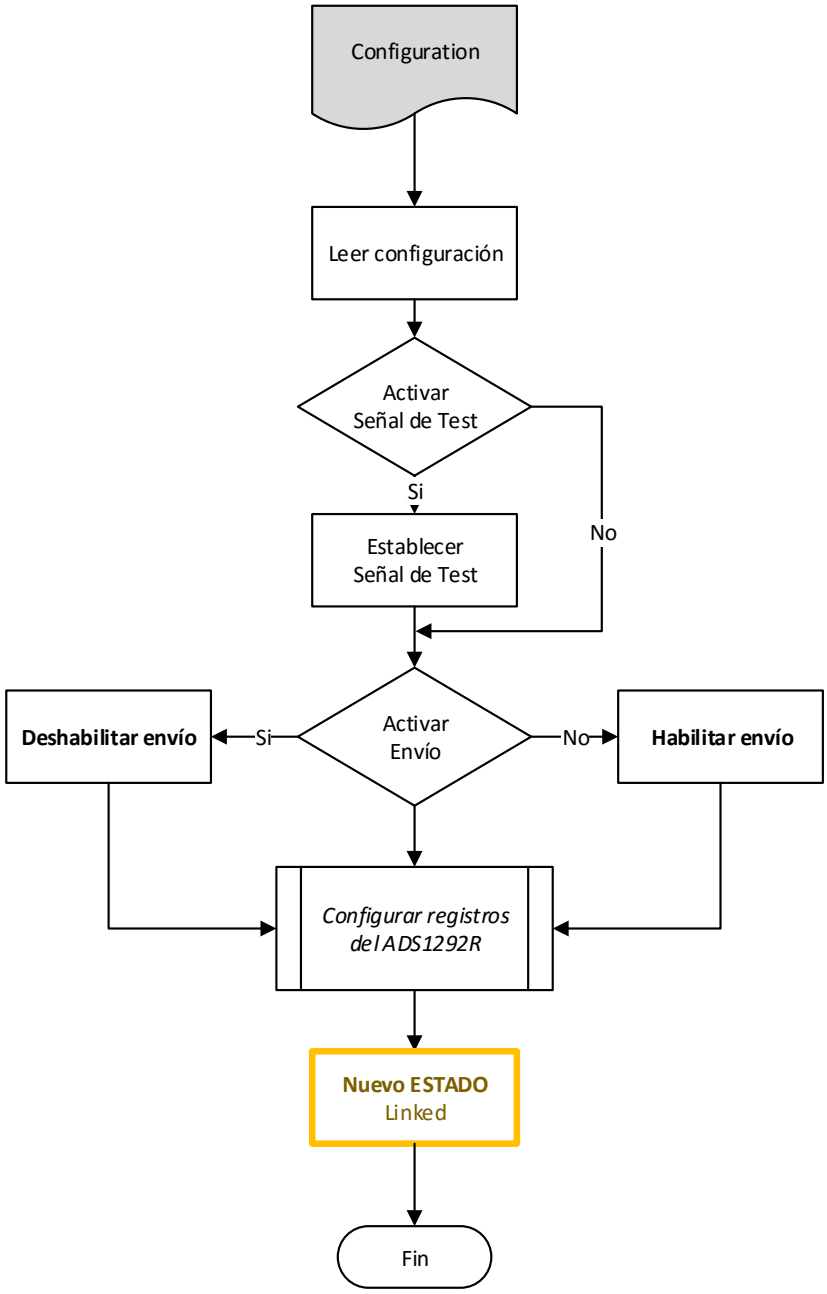


Figura 3-15. Diagrama de flujo del estado Configuration

Por último, el estado más importante en el que la conexión está activa y se están enviando las muestras al Monitor es *Streaming*. Éste comparte las mismas interrupciones que *Linked* y realiza las siguientes operaciones:

- Espera activa hasta que llegue una muestra.
- Se procesa la muestra (Conversión de formato CA2 a Exceso K) [19].
- Si han llegado 3 muestras se puede formar un paquete, el cual se envía al *Monitor* junto con la información del Contacto del Sensor.

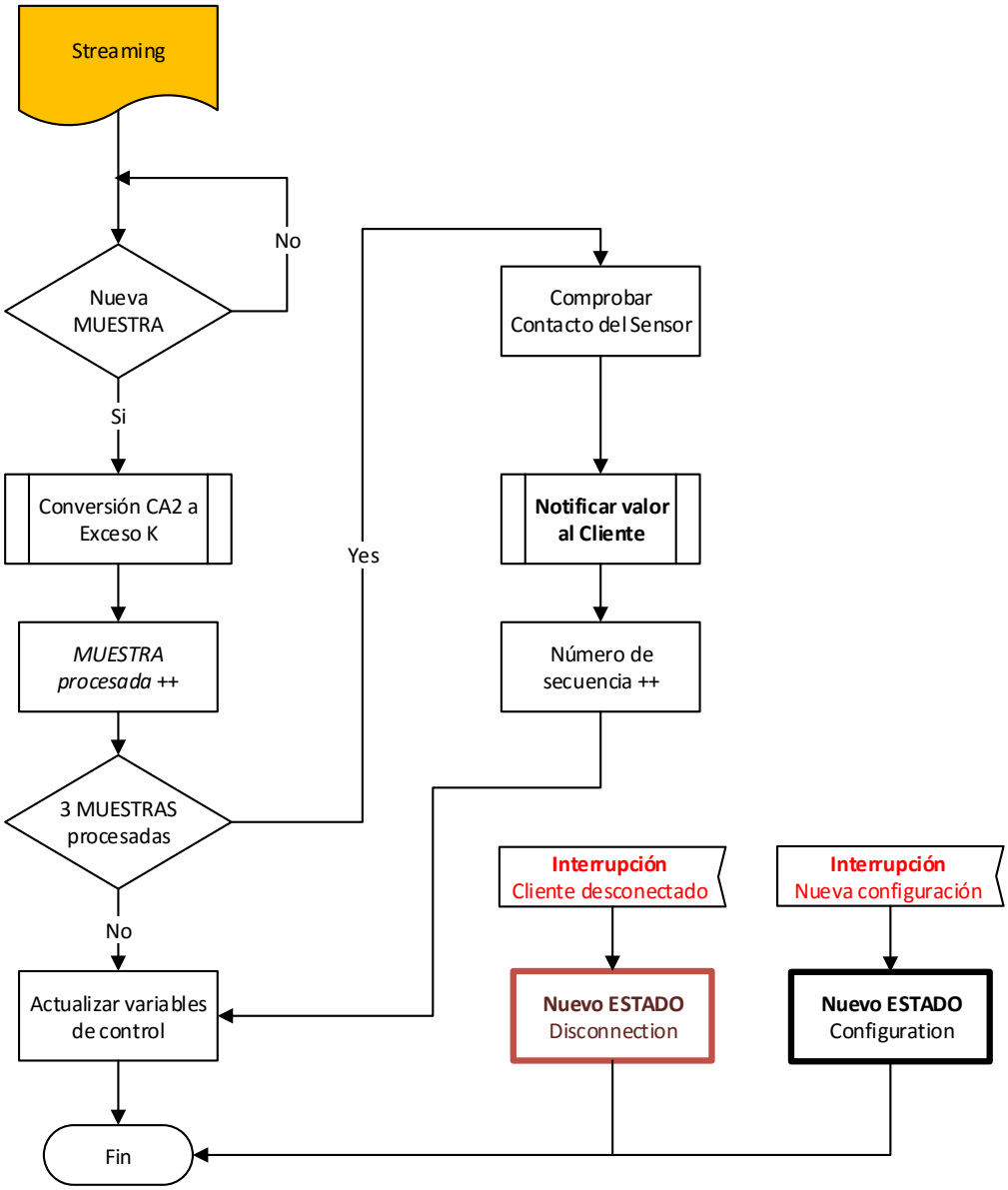


Figura 3-16. Diagrama de flujo del estado Streaming

Además, las interrupciones se han implementado siguiendo los diagramas de flujo de la Figura 3-17. Es necesario recordar, aunque no se haya mostrado en el diagrama, que la interrupción de *prioridad baja* puede ser a su vez interrumpida por la de *prioridad alta*, y nunca al contrario.

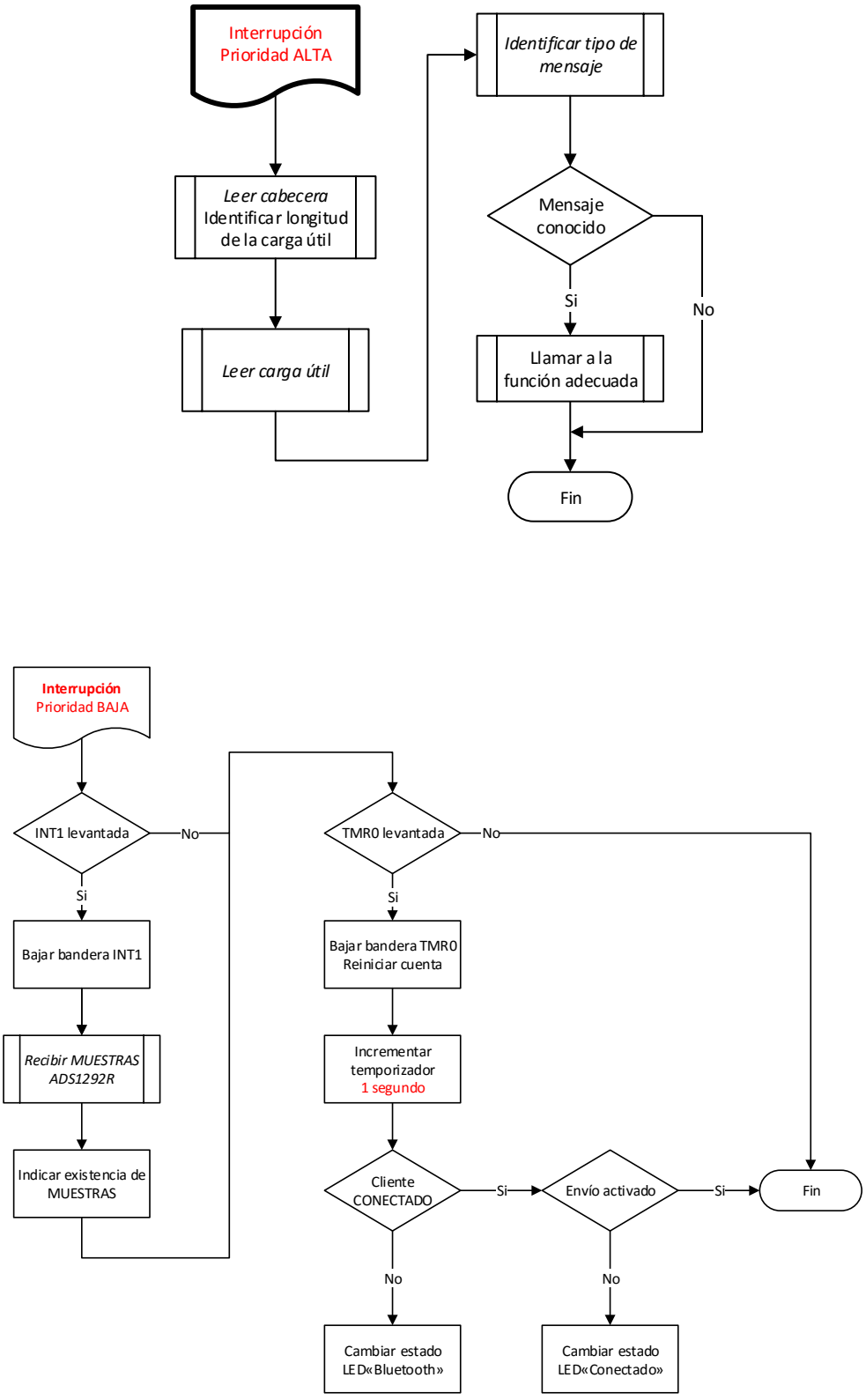


Figura 3-17. Diagramas de flujo de las interrupciones del sistema

3.1.4 Texas Instruments bq51050B

El chip elegido para la carga inductiva pertenece a la serie **bqTESLA** de *Texas Instruments*. Para comprobar su funcionamiento se decidió adquirir la placa de evaluación **bq51050BEVM-764**. La documentación que acompaña a dicha placa permite entender cómo se realiza la carga de la batería y bajo qué circunstancias.

Las principales características de este chip son las siguientes:

- Carga de baterías de litio de hasta 4.2V a una corriente máxima de 1A.
- Posibilidad de personalizar el perfil de carga mediante el uso de potenciómetros.
- Detener la carga de la batería bajo ciertas condiciones de temperatura.
- Posibilidad de utilizarlo junto con un sistema de alimentación externa (cableado).

En la Figura 3-18 se muestra el perfil de carga de la batería que realiza el chip integrado, el cuál se ha comprobado siguiendo las indicaciones de la documentación. En el eje horizontal se refleja el porcentaje de capacidad presente en la batería, mientras que el vertical es compartido por distintos voltajes y corrientes característicos del sistema.

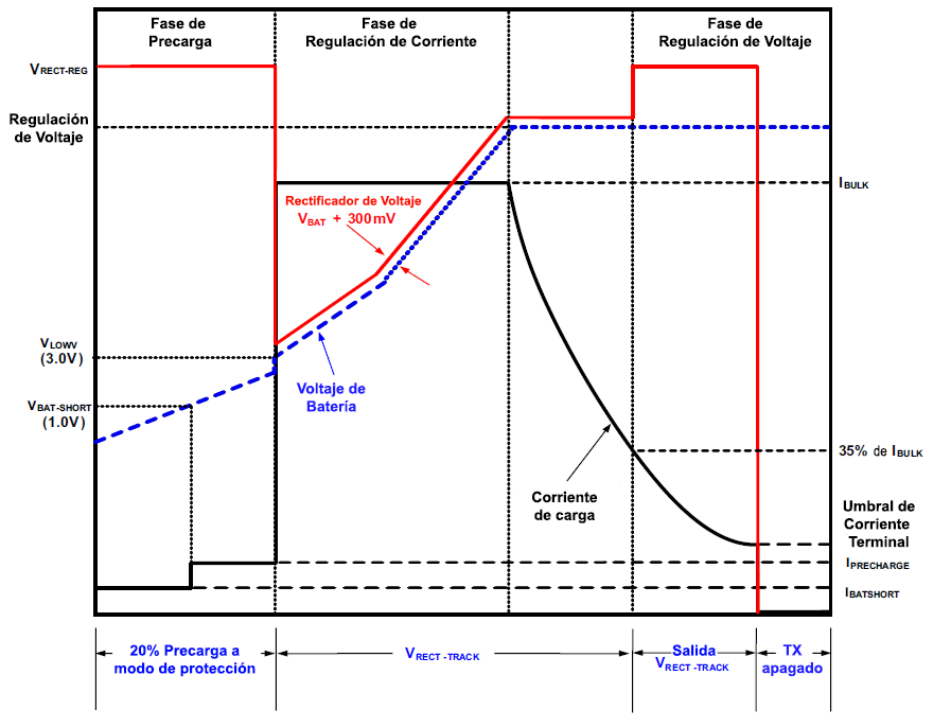


Figura 3-18. Perfil de carga de batería del chip BQ51050B

Sin embargo, para alimentar el primer prototipo se ha utilizado la batería junto con un regulador de voltaje **MCP1702** de *Microchip* [14] a 3.3 V, dejando el uso de este dispositivo para el prototipo final. A su vez se han añadido condensadores de desacople tanto en el regulador de voltaje como en las entradas de alimentación de cada uno de los componentes.

3.1.5 Sistema completo

La Figura 3-19 muestra el primer prototipo al completo. Durante la realización de las pruebas se ha utilizado una fuente lineal para no depender de la carga y descarga de la batería. En aras de comprobar la funcionalidad del dispositivo se ha hecho uso de la aplicación **nRF Master Control Panel** desarrollada en Android por *Nordic Semiconductor* [20], junto con la que nos ofrecía *Bluegiga*, **bleGUI** [16].

Además, durante el desarrollo del prototipo se han realizado pruebas con la aplicación *Monitor*, desarrollada en paralelo. Estas han permitido tanto descubrir errores en la implementación del código como asegurar el correcto funcionamiento del sistema completo.

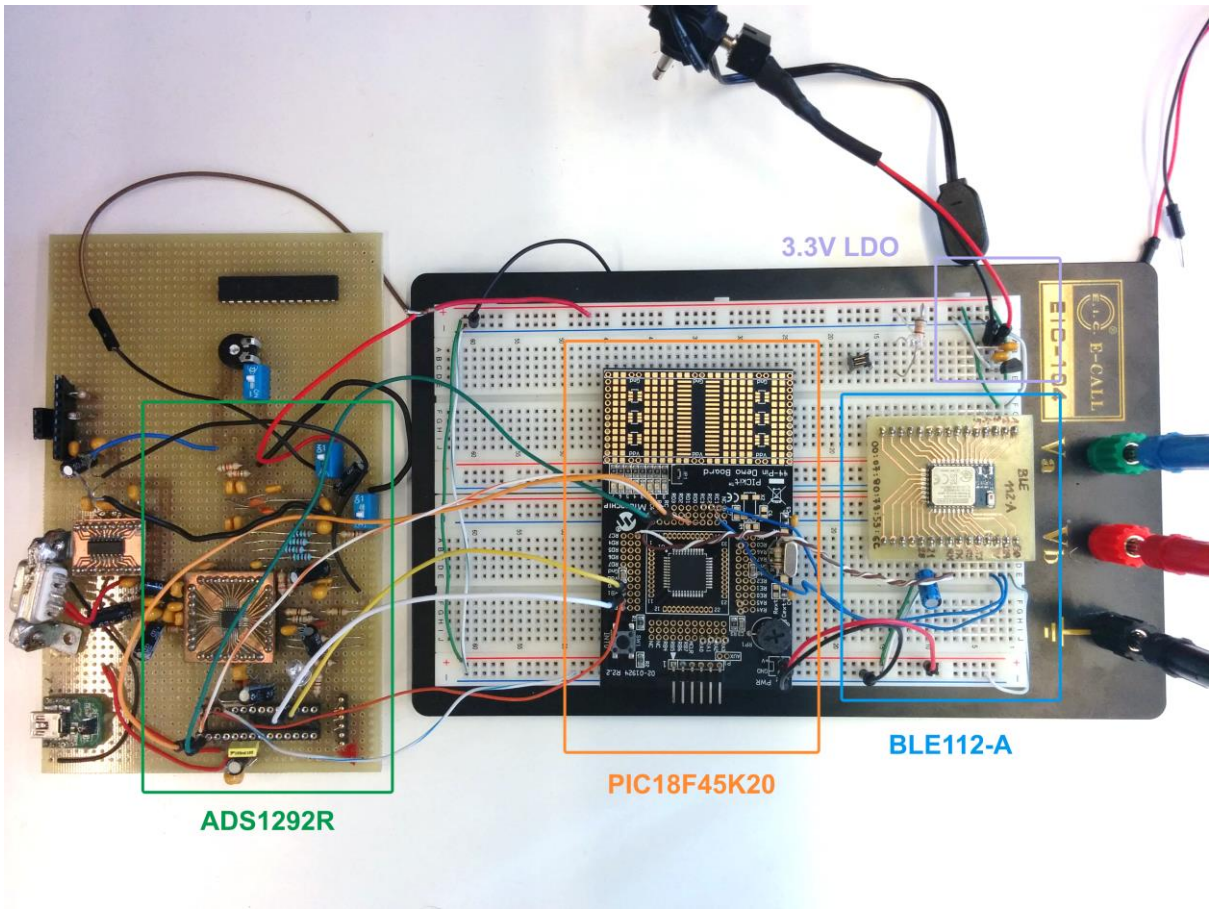


Figura 3-19. Primer prototipo funcional

3.1.6 Resultado final

A partir de este primer prototipo se pueden tomar una serie de decisiones técnicas que afectarán al diseño y desarrollo del dispositivo final.

- El envío funciona correctamente de **125 a 1000 muestras por segundo**. Si la ***frecuencia de muestreo*** es conocida, no es necesario enviar el *timestamp* correspodiente a cada muestra, ni que los envíos estén completamente sincronizados. Esto permite utilizar el **oscilador interno** del microcontrolador, con un margen de error asumible (1%).
- La ***memoria*** utilizada por el programa **no alcanza los 16KB**, el modelo **PIC18F24K20** ofrece las prestaciones suficientes para el siguiente prototipo utilizando un encapsulado más pequeño.
- El ***alcance de la conexión*** es **mayor a los 5 metros** propuestos, pero este resultado debe tomarse en consideración sólo en el segundo prototipo ya que podría vverse reducido.

3.2 Segundo prototipo

Tras diversas pruebas y correciones, podemos confirmar que el primer prototipo funciona correctamente. El siguiente paso será diseñar el prototipo final que cumpla todos los objetivos de este proyecto final de carrera.

En este capítulo se abordará el diseño e implementación del dispositivo que debe cumplir las especificaciones marcadas en los *Requisitos y Especificaciones del Proyecto*. Para el desarrollo de la placa de circuito impreso se hará uso del software de diseño electrónico **Altium Designer 15** [21] en su versión de evaluación, la cual permite utilizar todas las funcionalidades que ofrece la versión completa.

Se ha requerido de un tiempo entre 2 y 3 semanas para aprender a utilizarlo y saber qué herramientas son útiles para este proyecto en particular. En la propia web del desarrollador se ofrecen tutoriales para un aprendizaje rápido del software y una sección con las preguntas más habituales respondidas.

Por otro lado, antes de diseñar el circuito hay que establecer las limitaciones de carácter técnico como el tamaño o el número de capas de la placa. El ***tamaño*** vendrá dado por la caja donde va a estar alojado el circuito.

En la Figura 3-20, se muestra la caja elegida, teniendo en cuenta el tamaño de la batería y la bobina de recepción del módulo de carga inalámbrica, es el modelo **Minitec B9004858** del fabricante *OKW*. Según las especificaciones, esta caja está preparada para un circuito impreso de tamaño **51x34 mm**.

Figura 3-20. Encapsulado del segundo prototipo

3.2.1 Altium Designer

El primer paso en *Altium* consiste en crear un proyecto que englobará todos los ficheros de nuestro circuito. Se distinguen dos tipos de ficheros:

- ***SchDoc***: archivo que contiene el esquemático de un circuito. Puede contener a su vez o estar relacionado con otros ficheros *SchDoc*.
- ***PcbDoc***: archivo que contiene el *layout*, las capas y disposición de los componentes del circuito impreso.

Aunque pueden desarrollarse por separado, la principal ventaja de este software es la posibilidad de sincronizar los cambios producidos en un fichero *SchDoc* con otro *PcbDoc* y viceversa, lo que reduce notablemente el tiempo de trabajo. Otra funcionalidad importante es la de la corrección de errores, esto es, el programa puede compilar los dos tipos de fichero siguiendo unas reglas establecidas para confirmar que no haya incoherencias y que la placa puede ser fabricada en base a dichas reglas.

Por otro lado, el programa permite usar librerías de componentes, tanto para esquemáticos como para circuitos. Incluido en el mismo, existen gran cantidad de librerías en formato ***LibPkg*** con componentes genéricos que pueden ser utilizados sin necesidad de introducirlos manualmente en el programa para cada nuevo proyecto. Pero dada la considerable cantidad de componentes utilizada en este programa se ha decidido implementar una librería *LibPkg* propia. Para ello se ha de crear un proyecto adicional, que contiene dos tipos de archivos:

- ***SchLib***: librería de componentes a utilizar en los ficheros *SchDoc.*
- ***PcbLib***: librería de componentes a utilizar en los ficheros *PcbDoc*.

Los componentes que se han implementado en esta librería provienen tanto de creación propia, a partir de la documentación de los componentes, de material ofrecido por el programa y de algunas librerías de acceso libre encontradas en Internet. Por último, es necesario compilar el proyecto para obtener un fichero *LibPkg*, a utilizar en el proyecto principal del circuito.

Para simplificar el diseño del esquemático del dispositivo, se ha decidido crear una jerarquía de archivos *SchDoc*. El fichero principal ***Main.SchDoc*** contiene bloques generales que se corresponden con los de la Figura 1-6, el diagrama de bloques conceptual del sistema. Para cada uno de los bloques se ha creado un fichero diferente donde se ha realizado el esquemático usando los componentes creados con anterioridad.

En el **Anexo D** se encuentran todos los esquemáticos a los que se harán referencia en este Apartado, por lo que cualquier referencia a los mismos puede consultarse en dicho anexo.

3.2.1.1 ADS1292R.SchDoc

El bloque correspondiente a la adquisición de las señales está compuesto por el chip *ADS1292R* junto con toda la circuitería adicional necesaria. Para acelerar el proceso, el esquemático ha sido diseñado a partir del circuito en el *Apéndice C.2* de la documentación de la placa de evaluación **ADS1292ECG-FE** [10]. Eliminando componentes innecesarios como *Test point* y aplicando algunas simplificaciones se ha obtenido un circuito con las siguientes características:

- Alimentación analógica y digital *VDD* a 3V con tierras separadas *VSS* y *DGND*.
- Posibilidad de conectar la pierna derecha para mejorar el CMRR.
- Salidas (*DOUT*, *DRDY*) y entradas (*SCLK*, *DIN*, *CS*, *PWDN*) del ADS1292R conectadas al PIC18F24K20.

3.2.1.2 BLE112.SchDoc

El módulo Bluetooth, sin embargo, no requiere de gran cantidad de componentes para funcionar. Basta con añadir condensadores de desacoplamiento en las entradas de alimentación, las conexiones necesarias para los pines de programación y aquellas para conectarse con el PIC18F:

- *TX* y *RX* para la conexión UART.
- *RST* y *WUP*, para reiniciar el módulo y despertarlo, respectivamente.

3.2.1.3 PIC18F24K20.SchDoc

El bloque de control está gestionado por el microcontrolador PIC18F24K20. Haciendo uso de 21 de los 28 pines disponibles, se gestiona todo el sistema, la conexión con los módulos ADS1292R y BLE112 y el cálculo del voltaje de la batería con un circuito relativamente sencillo, presentando en la Figura 3-21.

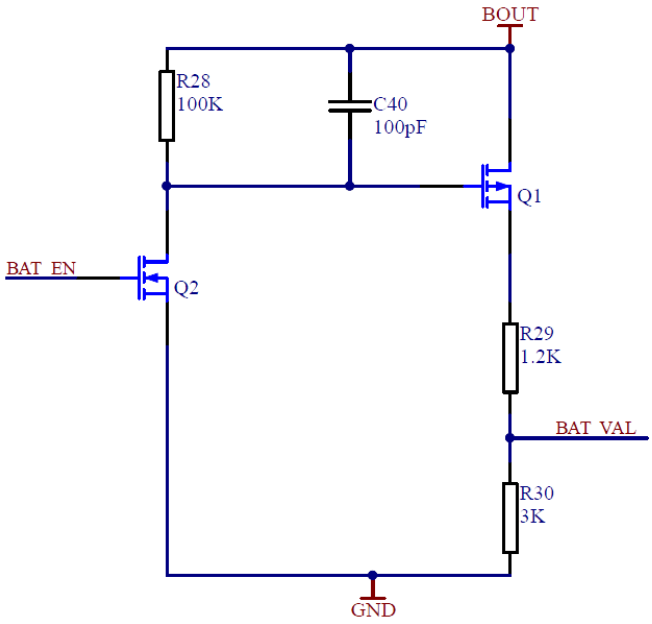


Figura 3-21. Esquemático del circuito de la batería

Este circuito permite, mediante el uso de transistores Mosfet, activar la tensión visible para el puerto analógio-digital del microcontrolador sólo en el momento en que se quiera hacer la medida, evitando drenaje continuo de corriente en el divisor de tensión.

Dicho divisor multiplica la tensión de la batería por un cociente $3/4.2$ de manera que los valores visibles desde el microcontrolador irán de 0 a 3 V (La tensión de referencia usada en la conversión analógico-digital).

También se han incluido tres LED (Rojo, Azul y Amarillo) limitados en corriente por resistencias en serie, controlados según el estado del sistema como ya se vio en los diagramas de flujo del apartado *Microchip PIC18F45K20*.

Todos los pines de entrada y salida de este bloque se han mencionado en los dos anteriores, ya que corresponden con las conexiones con los módulos de adquisición y Bluetooth.

3.2.1.4 POWER.SchDoc

Este bloque corresponde a toda la circuitería necesaria para obtener todos los voltajes necesarios:

- Voltaje de la batería, normalmente entre 3 y 4.2 V, según la capacidad actual de la misma.
- Voltaje regulado a un valor fijo de 3V, que alimentará a todos los componentes.

Para regular el voltaje de la batería a 3V se utilizará el **TPS73201** de *Texas Instruments* [10], junto con la circuitería necesaria para configurarlo, indicada en la documentación.

3.2.1.5 BQ51050B.SchDoc

El módulo de carga inductiva de la serie **bqTesla** es una de las partes más importantes de esta implementación pues no participaba en el primer prototipo. Este bloque debe funcionar de manera totalmente autónoma, cargando la batería si fuera necesario al detectar la presencia de un *Transmisor Qi*.

Texas Instruments ofrece el esquemático de una versión simplificada de su receptor, a modo de demostración del pequeño tamaño en el que se puede integrar su producto. La **TIDA-00329** [10], de un tamaño de 5x5 mm y 4 capas nos servirá como referencia en nuestro diseño. La documentación incluye los esquemáticos del circuito como los de la Figura 3-22, los cuales se han modificado ligeramente para hacerlos coincidir con nuestra implementación.

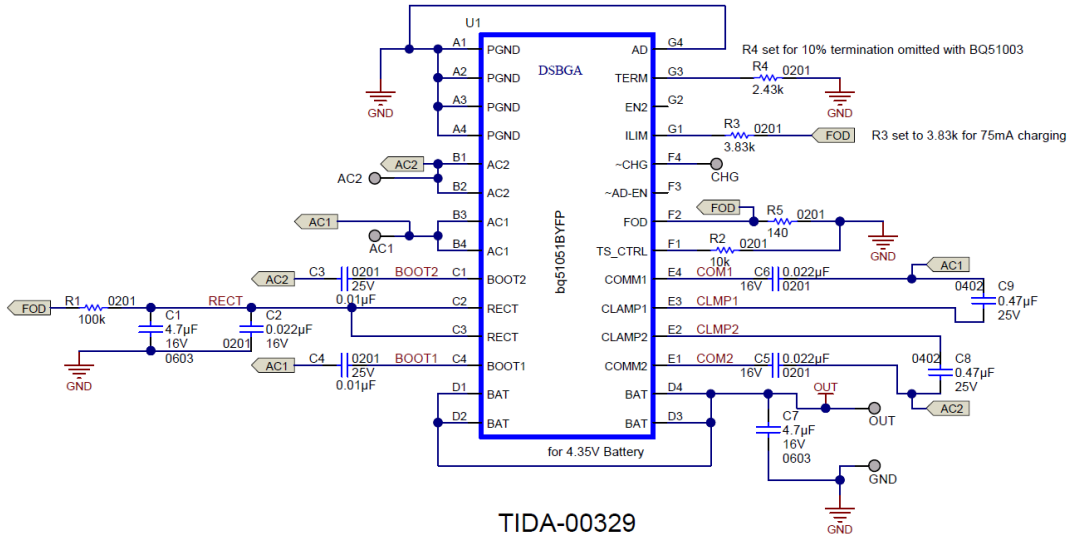


Figura 3-22. Esquemático del TIDA-00329

La única modificación ha sido incluir en el puerto *CHG* el mismo LED y resistencia que se puede encontrar en el esquemático de la placa **bq51050BEVM-764** y así tener información visual del estado de carga.

3.2.2 Disposición en el circuito impreso

Una vez se han definido todos los ficheros correspondientes al esquemático de nuestro dispositivo, el siguiente paso consiste en decidir su disposición en la placa de circuito impreso. Dada la gran cantidad de componentes para el espacio reducido del que se dispone, se ha decidido fabricar una placa de 4 capas, algo que no es posible en el propio laboratorio y que se encargará a una empresa externa.

Por tanto, deben fijarse una serie de restricciones o reglas en nuestro diseño según lo que nos indique el fabricante. De esta manera, el programa puede comprobar en cualquier momento si las reglas se están cumpliendo y asegurar un diseño que no se corresponda con el esquemático o sea irrealizable.

- Ancho y separación mínima de las pistas: 180 µm
- Diámetro mínimo del taladro: 0.25 mm
- Corona mínima alrededor del taladro: 0.2 mm
- Margen mínimo de la máscara antisoldante: 75 µm
- Trazo mínimo de la máscara antisoldante: 0.1 mm
- Margen mínimo entre trazo y marca de componente: 0.1 mm
- Trazo mínimo del marcaje de componentes: 0.125 mm
- Taladros pasantes en todas las capas

El sistema de reglas del proyecto permite aplicar todas estas restricciones y mediante el uso del DRC (*Design Rule Checker*) pueden comprobarse en cualquier momento del diseño.

La distribución de los componentes se ha llevado a cabo siguiendo las recomendaciones de los distintos fabricantes, realizando modificaciones para facilitar las conexiones entre los componentes.

Durante el proceso de diseño de la placa, se vio conveniente utilizar 4 capas debido a la gran cantidad de vías (conexiones entre distintas capas) y trazados complejos que suponía el uso de 2 capas, y con la siguiente estructura:

1. ***Top Layer***: contiene parte de los componentes y todos los conectores externos.
2. ***Power Layer***: plano de VDD a 3 V, además de pistas internas entre vías de la primera y última capa.
3. ***Ground Layer***: plano de VSS, además de pistas internas entre vías de la primera y última capa.
4. ***Bottom Layer***: contiene el resto de componentes y los LED de estado del sistema.

El cobre sobrante de las capas *Top* y *Bottom* se ha utilizado para los 3 planos de masa VSS, GND y BGND. Todos ellos están conectados por resistencias de 0 ohmios en un único punto del circuito con el objetivo de aislar las los distintos planos del circuito (analógico, digital y batería) y reducir el ruido entre ellas [22].

Por otra parte, Altium permite exportar el circuito en formato 3D *STEP*. Además, la librería de componentes que hemos creado incluye modelos en este formato para cada uno de dichos componentes, de manera que puedan usarse en futuros proyectos.

Este ha sido un trabajo realizado en paralelo al diseño de la placa para obtener un modelo 3D equivalente del circuito, haciendo uso de los programas **FreeCAD** [23] e **Inkscape** [24] tanto para crear como para editar y adaptar modelos distribuidos gratuitamente en Internet [25].

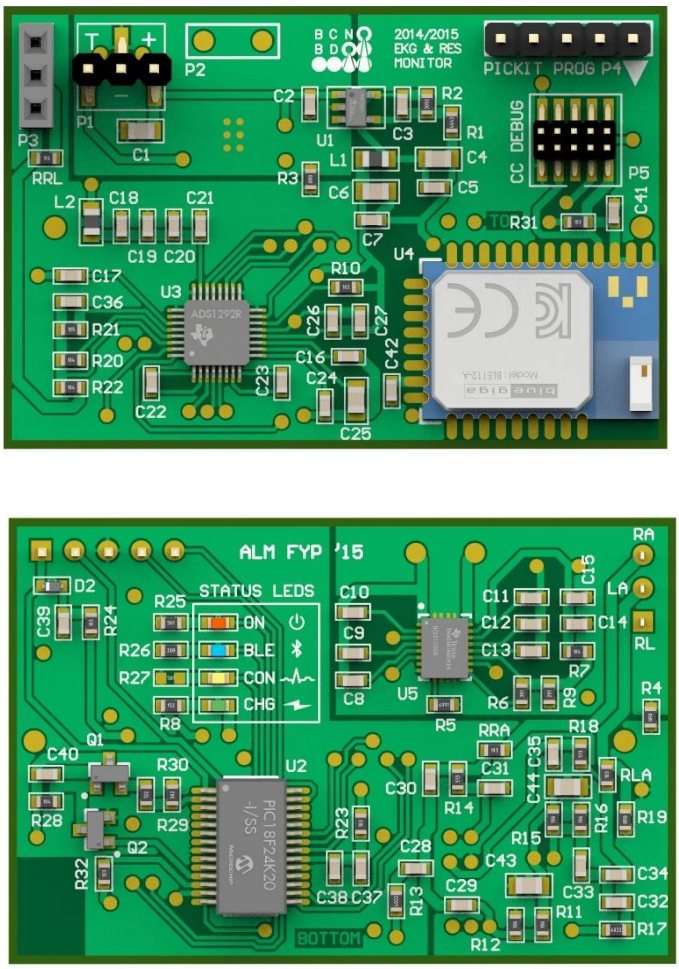


Figura 3-23. Modelo 3D del circuito impreso con componentes soldados

Tras una semana desde que se pidió se ha recibido la placa del proveedor, la cual se muestra en la Figura 3-24. La serigrafía de los límites de los componentes pasivos fue eliminada por el fabricante debido a una restricción adicional que no se tuvo en cuenta antes de encargar la placa. Aún así no debe afectar a la hora de soldar los componentes.

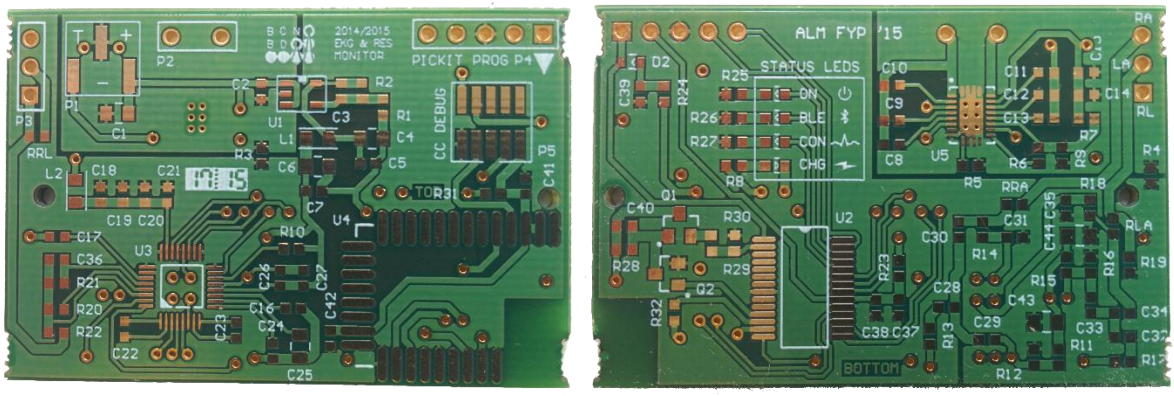


Figura 3-24. Placa de circuito impreso

A partir del los esquemáticos y usando como referencia la serigrafía de los códigos se deben ir soldando los componentes en la placa. Dado que los componentes pasivos son de pequeño tamaño (2x1.2 mm) se ha preferido usar el microcoscopio para soldar componentes más rápido, tal y como se aprecia en la Figura 3-25.

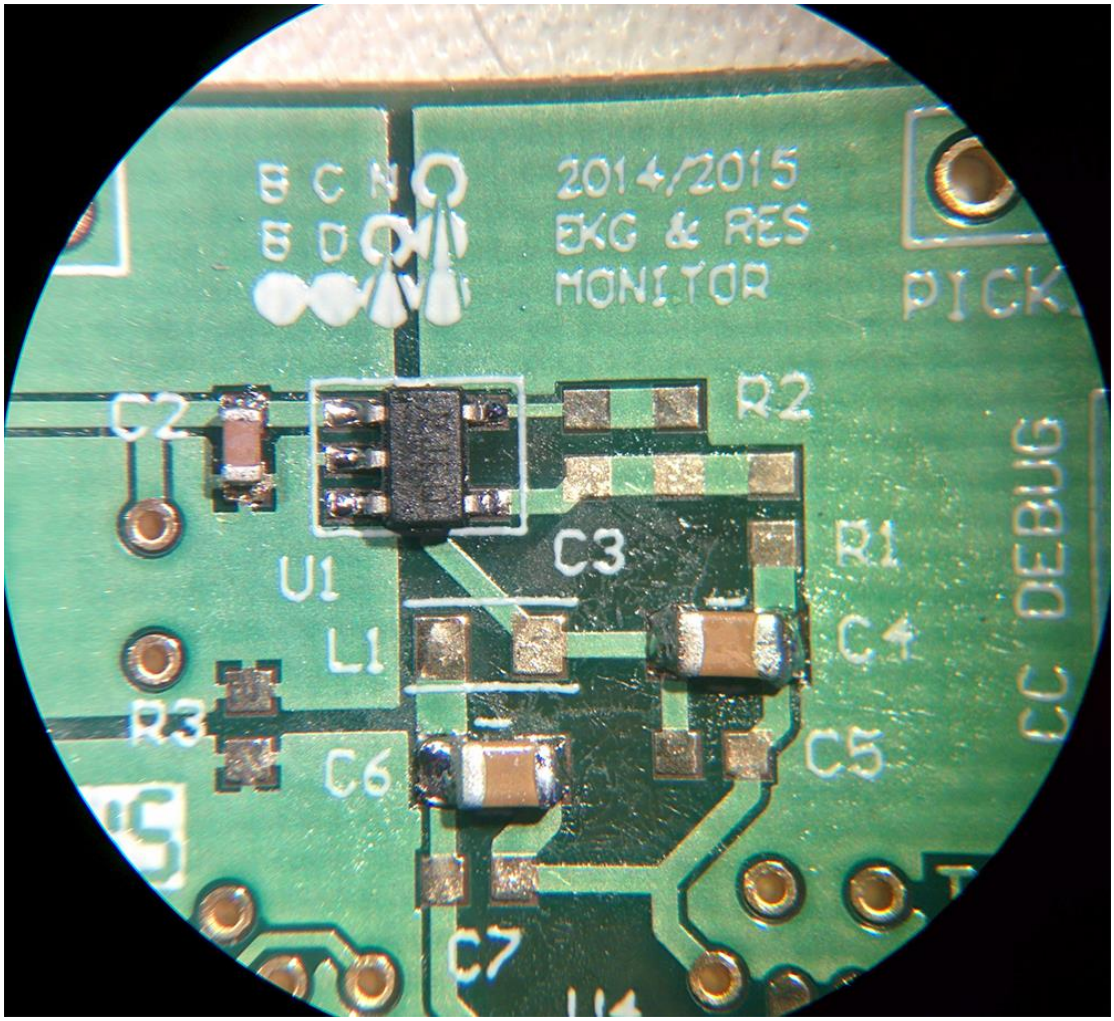


Figura 3-25. Vista al microscopio de la placa de circuito impreso

Este proceso ha durado en torno a 2 semanas dado que no se tenía experiencia previa con componentes tan pequeños. Se han ido probando cada uno de los bloques por separado, además de realizar correciones en el diseño descubiertas después de realizar estas comprobaciones.

3.2.3 Resultado final

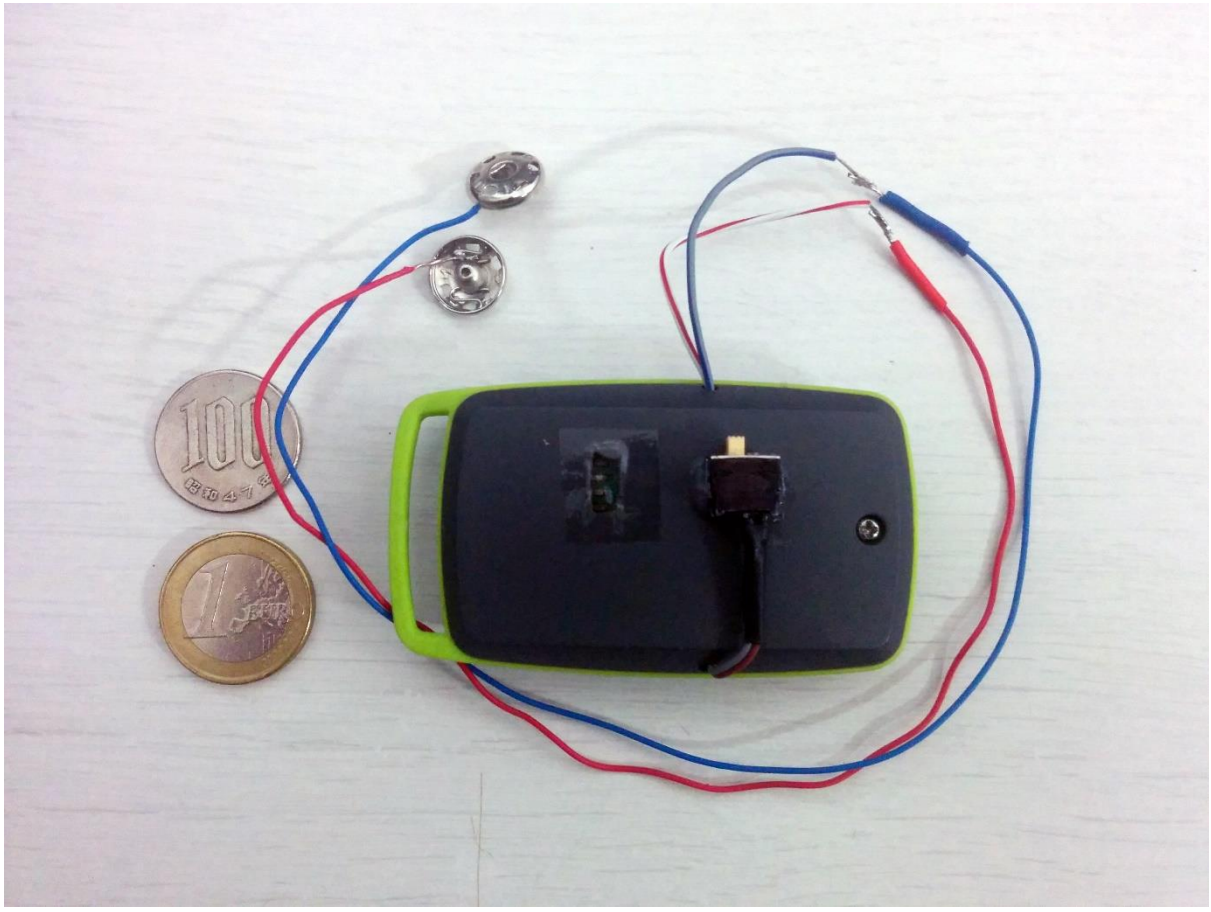


Figura 3-26. Segundo prototipo funcional

El último prototipo ha quedado tal y como aparece en la Figura 3-26. Se han instalado dentro de la caja: la placa de circuito impreso con todos los componentes soldados, la batería conectada junto con un interruptor situado en el exterior y la bobina de recepción del módulo de carga.

Una vez han sido programados tanto el PIC18F24K20 como el módulo BLE112, se han retirado los pines para que no molestasen al cerrar la caja. En el siguiente capítulo se analizará el correcto funcionamiento de esta segunda versión.

4 CARACTERIZACIÓN DEL SISTEMA

El segundo sistema ha sido implementado con éxito en base a los esquemáticos diseñados. A continuación se presentan una serie de pruebas para caracterizar el sistema. Por una parte se comprobará que las muestras recogidas por el ADS1292R se corresponden con las recibidas por el *Monitor*, que el ***enlace*** Bluetooth es ***fiable*** y no se producen pérdidas notables, y que ***sistema*** se mantiene ***estable*** y sin bloquearse durante las 24 horas fijadas como objetivo.

Tanto este capítulo de Caracterización del sistema como el siguiente de Resultados ha necesitado en todo momento el uso de la aplicación ***Monitor*** realizada en paralelo con este proyecto final de carrera.

4.1 Adquisición de la señal

El integrado ADS1292R, tal y como se indicó con anterioridad, ha pasado las certificaciones [13] que lo hacen apto para su utilización en dispostivos médicos como el implementado aquí. No obstante, se realizará un análisis de las señales mostrada en el Monitor para distintas entradas y frecuencias en el dispositivo.

Éste se ha configurado para aplicar ***ganancia unitaria*** y frecuencia de ***500 muestras por segundo***. El resto de parámetros no deben afectar al análisis que se va a realizar. Según la documentación del integrado, el filtrado digital tiene una frecuencia de corte a 3 dB que equivale a $0.262 \cdot f_{DR} = 131Hz$. En la siguiente figura se muestran los valores de V_{pp} obtenidos para una entrada de 1 mV de un generador de señales conectado a la entrada del chip.

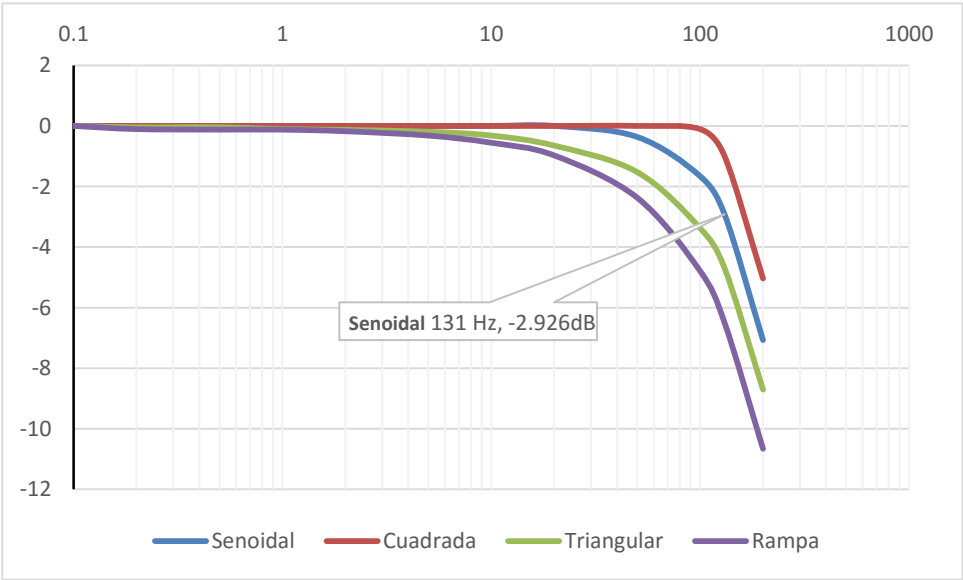


Figura 4-1. Diagrama de magnitud para distintas entradas

4.2 Transmisión Bluetooth

El análisis del enlace entre dispositivo y Monitor se ha realizado teniendo en cuenta dos parámetros:

- Potencia recibida en el Monitor o ***RSSI*** (*Received Signal Strength Indicator*) en dBm.
- **Número de secuencia** o campo *ID/Contacto* de la característica *Multisample Measurement* implementada en el perfil Bluetooth de la aplicación.

El gráfico mostrado en la Figura 4-2 se ha realizado bajo las siguientes circunstancias:

- Entorno sin **ningún obstáculo visible** a menos de 20 metros.
- Dispositivo colocado delante del tórax del paciente.
- Monitor de frente, también a la altura del tórax.
- Potencia máxima configurada para la transmisión Bluetooth (0 dBm).
- Potencia recibida media en intervalos de 2 milisegundos durante un total de 15 segundos.

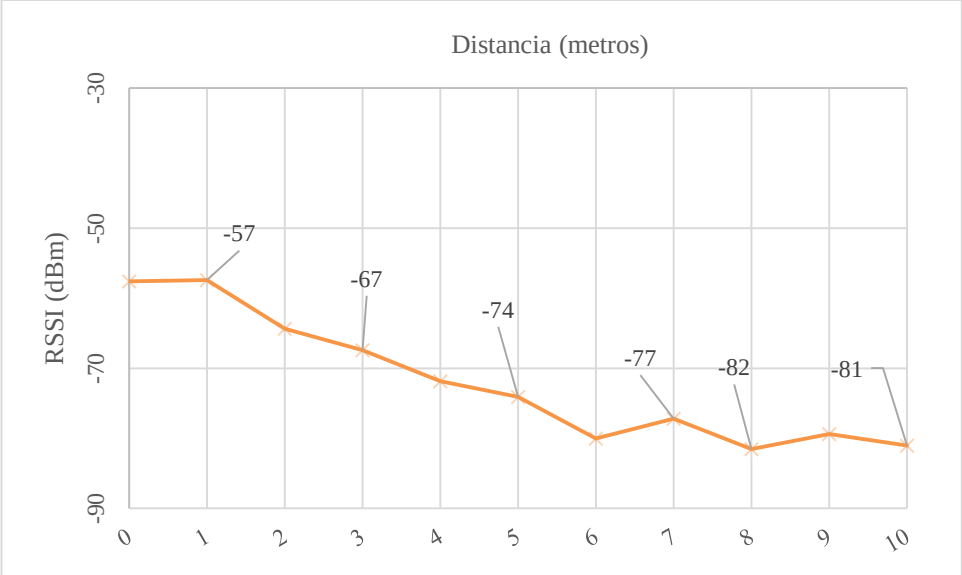


Figura 4-2. RSSI en el Monitor, función de la distancia

No es novedad que la potencia recibida se reduce con la distancia, aunque cabe destacar que la variabilidad obtenida del RSSI hace que no sea tan lineal como debiera. Se considera una distancia de 0 metros al caso del paciente sujetando el *Monitor* en la mano a la misma altura que el dispositivo.

Por otro lado, en las distancias representadas (0 a 10 metros) no se han producido pérdidas en el envío de los datos al Monitor. Sólo a partir de una distancia de 11 metros, el enlace se ha vuelto inestable y las pérdidas han alcanzado hasta el 50% en algún momento de la prueba.

4.3 Autonomía y estabilidad

La batería para el dispositivo se adquirió teniendo en cuenta que se requería una autonomía de 24 horas, suponiendo que se refería funcionando en el estado de envío de la información al Monitor. Para realizar los cálculos de consumo del sistema se ha utilizado el multímetro digital **PREMA 5017** [26]. Este incluye opciones de filtrado que realiza la media de las últimas medidas tomadas directamente desde el propio multímetro.

En la Tabla 4–1 se muestran los distintos estados del dispositivo con el consumo asociado a cada uno de ellos. Dado que el único parámetro de configuración que afecta al consumo es la frecuencia de muestreo, el estado *Streaming* se ha desglosado para cada una de ellas. Por otro lado, ya que no se tuvo en cuenta al hacer el cálculo de la batería, se ha incluido el consumo total sin tener en cuenta el de los LED de la placa.

Cada uno de los LED está limitado en corriente por una resistencia. En el caso del *rojo* se produce un consumo medio de 2 mA, mientras que para el *amarillo* y *azul*, el mismo es de 2 y 5 mA, respectivamente. El LED *verde* no se ha tenido en cuenta, ya que sólo se enciende en el proceso de carga de la batería.

A su vez, cada estado tiene una combinación distinta de LED encendidos:

- ***Advertising (Fast Connection y Power Saving)***: El *rojo* está siempre encendido y el *azul* sólo la mitad del tiempo, un total de 4.5 mA.
- ***Linked***: El *rojo* siempre encendido y el *amarillo* sólo la mitad del tiempo, un total de 3 mA.
- ***Streaming***: Tanto el *rojo* como el *amarillo* siempre encendidos, un total de 4 mA.

Tabla 4–1 Consumos medios para cada uno de los estados

	Consumo medio	**Consumo sin LED**
Advertising (Fast Connection)	16.86 mA	12.36 mA
Advertising (Power Saving)	14.98 mA	10.48 mA
Linked	15.60 mA	12.60 mA
Streaming (125 muestras/s)	26.82 mA	22.86 mA
Streaming (250 muestras/s)	27.79 mA	23.79 mA
Streaming (500 muestras/s)	28.58 mA	24.58 mA
Streaming (1000 muestras/s)	30.71 mA	26.71 mA

El objetivo del proyecto fijaba una autonomía de 24 horas de manera que para el estado *Streaming* a 500 muestras por segundo el consumo no debe superar la capacidad total de la batería.

$$Consumo\ total = (28.58\ mA) \cdot 24h \simeq 685.92\ mAh \tag{4–1}$$

Como puede verse, este consumo es inferior a la capacidad total de la batería (710 mAh), por lo que es prueba suficiente de que se ha cumplido la especificación de autonomía.

La estabilidad se ha comprobado realizando las siguientes pruebas:

- Desde el *Monitor*, provocando cambios de estado en el dispositivo, además de comprobar que el sistema sigue funcionando tras solicitar todas las combinaciones posibles de configuración.
- Dejando el sistema funcionar en el estado *Streaming* durante un periodo de 6 horas. Es un tiempo más que suficiente para suponer que ningún error en la programación del dispositivo pudiera provocar su inestabilidad.

En ambos casos, el sistema se ha mantenido estable, si bien es cierto que en el momento de establecer la conexión con el Cliente ha habido ocasiones que han necesitado de más de un intento. Aun así, el dispositivo no se ha bloqueado durante el transcurso de la prueba.

5 RESULTADOS

Una vez caracterizado nuestro dispositivo, sólo queda comprobar que las muestras recibidas en el ***Monitor*** se corresponden con la señal capturada a la entrada. Por una parte, se utilizará tanto el Simulador de paciente que genera una señal estable y conocida, para más adelante conectarlo a un paciente en reposo y así comprobar que funciona también en casos reales.

Las señales mostradas a continuación han sido reconstruidas a partir de los números correspondientes a las muestras recibidas en el *Monitor*, sin realizar ningún tipo de procesamiento.

5.1 Pruebas realizadas

5.1.1 Simulador de paciente

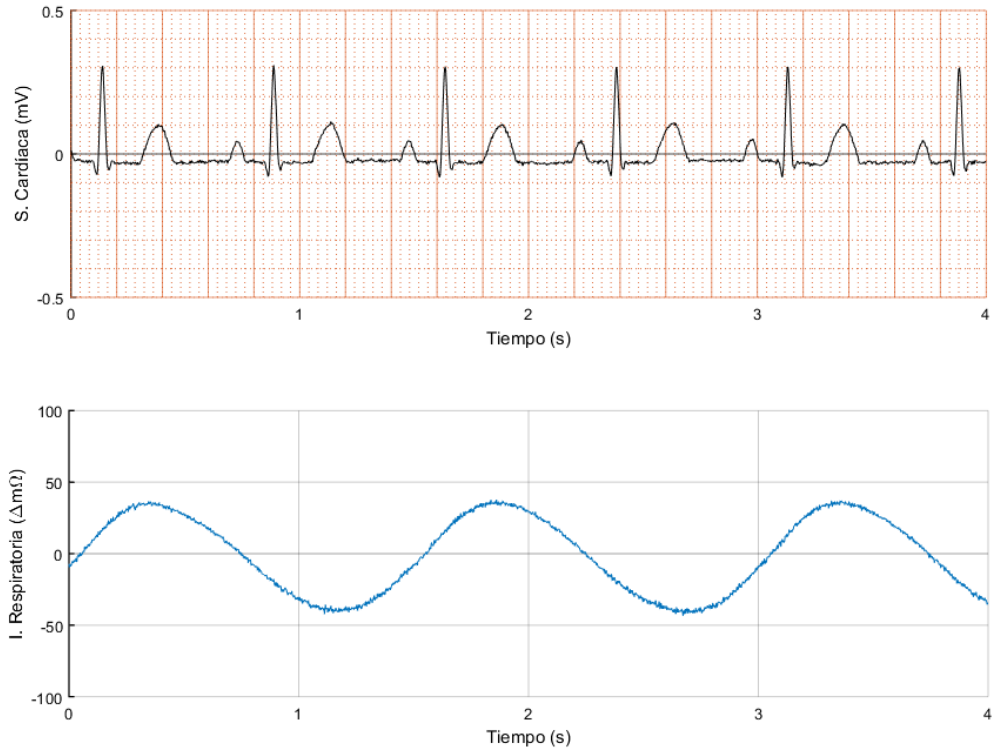


Figura 5-1. Señales recibidas en el *Monitor* del Simulador de paciente

Como podemos ver en la Figura 5-1, el electrocardiograma muestra todas las componentes que hemos definido en una señal cardíaca: la ***despolarización P***, el ***complejo QRS*** y la ***repolarización T***. Como podemos ver en la imagen, la señal cardiaca apenas alcanza los $0.4\ mV_{PP}$, a pesar de que el *Simulador de Paciente* ha sido configurado para que generase una señal de amplitud $1\ mV_{PP}$.

5.1.2 Pruebas en reposo

Las pruebas con un paciente real se han realizado en las siguientes condiciones:

- Paciente en reposo sentado respirando profundamente.
- Electrodos situados en el tórax, a la altura del pecho y cerca de la axila [3].

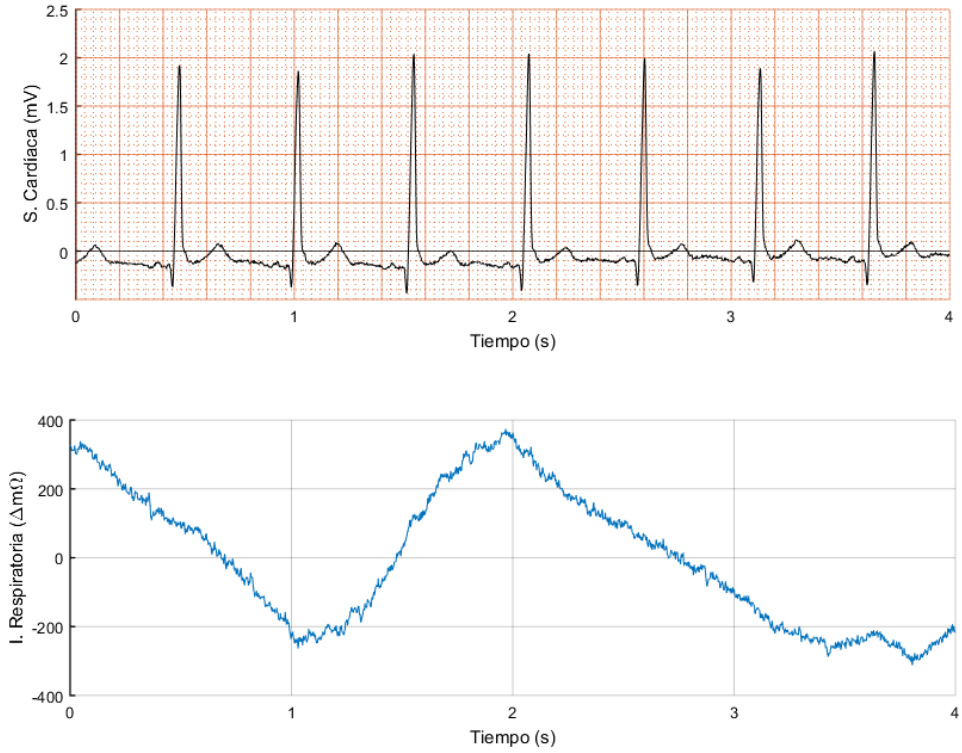


Figura 5-2. Señales recibidas en el *Monitor* de un paciente real

Tal y como se aprecia en la Figura 5-2, la señales capturadas en esta situación presentan mayor irregularidad, algo esperable por otra parte, teniendo en cuenta que presentan ruido de diversas fuentes no presente en el caso anterior.

En la señal cardíaca se aprecian las tres componentes principales:

- La ***despolarización P*** se recibe más débil, apenas distinguible con el ruido.
- El ***complejo QRS*** pronunciado, los 3 puntos se reciben con claridad siendo el ***S*** el que menos.
- La ***repolarización T*** con cierta atenuación, pero a diferencia de la *P* se distingue con claridad.

En cuanto a la respiración, la amplitud es mayor debido a que el paciente ha realizado exhalaciones e inhalaciones profundas. En la Figura 5-2, pueden observarse claramente una inhalación entre los segundos 1 y 2, continuando con una exhalación entre los segundos 2 y 3. Por otra parte, el nivel de ruido es elevado porque la variación de impedancia se ve afectada por cualquier otro movimiento del cuerpo.

6 ESTUDIO ECONÓMICO

Como último capítulo antes de llegar a las conclusiones se presenta un estudio económico de los diferentes gastos que se han producido durante la realización de este proyecto final de carrera. Los costes derivados de este proyecto se han calculado teniendo en cuenta que las tareas son realizadas por un Ingenierio de Telecomunicación recién titulado con un sueldo de 20 €/hora.

El tiempo empleado ha sido de 9 meses a razón de 5 horas al día durante un total de 184 días, lo cual no deja de ser una estimación del tiempo y esfuerzo totales invertidos. Todos los costes aquí reflejados se encuentran desarrollados en el **Anexo E**.

6.1 Costes de diseño y desarrollo

Además de la mano de obra, el proyecto ha requerido del equipamiento electrónico y todos los componentes utilizados en la fabricación del dispositivo.

Los costes de los componentes pueden variar según el momento de compra, fabricante, tecnología y distribuidor, por lo que el coste total derivado en este proyecto debe considerarse como un gasto aproximado ya que depende de varios factores variables en el tiempo.

También se incluye la amortización del equipamiento usado en el laboratorio, tanto el ordenador como las herramientas que ya se encontraban en el mismo y han sido utilizadas.

Tabla 6–1 Costes de diseño y desarrollo

Concepto	**Coste (€)**
Mano de obra (Ingeniero titulado)	18300
Kit de desarrollo bq51050BEVM-764	183,29
Transmisor Qi Aukey	12,99
Amortización del equipamiento del laboratorio	250
Costes del segundo prototipo	974,96
Total	**19721,24**

Los costes del segundo prototipo incluyen, salvo batería y caja, material para fabricar hasta 4 prototipos distintos. Esto es debido a que los proveedores exigen una cantidad mínima en la compra de componentes y la opción elegida en la fabricación de la placa de circuito impreso era de un paquete de 4 a 6 unidades a un precio de 268€.

6.2 Costes del producto final

Los costes de componentes de un dispositivo tal y como se encargaron para este proyecto ascienden a **150.50€** por dispositivo. Este coste no incluye las cantidades mínimas exigidas al comprar dichos componentes a los proveedores (RS-Components y Farnell).

El coste de componentes de un total de 100000 dispositivos ascenderían a **49.49€** por dispositivo. Este número es debido a utilizar un hipotético proveedor con mejores precios para grandes volúmenes (Digikey) y un fabricante de placas de circuito impreso a gran escala en China (PCBWay). Sin embargo, habría que añadir los costes de ensamblaje de componentes en las placas, lo cual podría aumentar ligeramente el coste del dispositivo.

7 CONCLUSIONES

El objetivo de este proyecto era desarrollar un dispositivo capaz de capturar señales biomédicas como la cardíaca y respiratoria, para ser enviadas, procesadas y mostradas en tiempo real. Dichas señales han sido analizadas y su caracterización ha sido clave en el diseño y las pruebas del mismo. Además como se desprende de los resultados, el sistema funciona correctamente.

Cada una de las partes se ha desarrollado en primer lugar de manera aislada, invirtiendo la mayor parte del tiempo en hacer funcionar todo el sistema en su conjunto. Formar parte de un proyecto más grande junto al desarrollo en paralelo de otro proyecto encargado del *Monitor* también ha sido un elemento clave en el tiempo empleado, tentiendo que acompasar el desarrollo ambos.

7.1 Consecución de los objetivos

A partir de unas especificaciones básicas y en base a un diagrama conceptual, se ha diseñado cada uno de los bloques funcionales que componen el sistema. En primer lugar por separado, para poco a poco añadir más bloques hasta conseguir completarlo.

Se han estudiado los diferentes caminos posibles y se ha tomado aquel que ha resultado más conveniente para este proyecto en particular. Los requisitos fijados han sido superados, salvo por el tamaño, que dependía de factores que no se podían controlar a priori. Aún así, con una resolución de 24 bits, frecuencia de muestreo hasta 1kHz y un tamaño reducido, el producto final cumple en su totalidad con el objetivo fijado en este proyecto.

Hay múltiples aplicaciones para las que este dispositivo puede resultar de utilidad. Fue concebido para obtener las señales cardíaca y respiratoria de un paciente con una resolución elevada y en tiempo real. Diversos grupos de investigación centrados en el análisis y detección de afecciones a través del análisis de las mismas podrían hacer uso del mismo, sobretodo en el laboratorio donde ha sido diseñado.

7.2 Mejora y línea futura de desarrollo

Finalmente, como para cualquier dispositivo de este tipo, es necesario mejorarlo tanto en el aspecto de su las funcionalidades que ya posee como otras que podría ofrecer. El chip *ADS1292R* es capaz de utilizar uno de los canales para obtener la temperatura, otra señal biométrica de gran interés. Además el código correspondiente al bloque de control puede ser mejorado en aspectos como el consumo o el protocolo de comunicación.

La fabricación de este dispositivo no hace si no seguir un camino que ya se abrió en anteriores proyectos: El desarrollo de aplicaciones y dispositivos que permitan obtener señales biométricas para sacar conclusiones que ayuden a mejorar la calidad de vida y la salud de los usuarios.

Anexo A – Servicio y Perfil BLE

BLUETOOTH® DOC	Date / Year-Month-Day	Status	Revision	Document No
	2015-06-25	Released	V10r00	ERS_SPEC
Prepared By Antonio López Marín	E-mail Address lopmar.antonio@gmail.com			N.B.

ELECTROCARDIOGRAM & RESPIRATION IMPEDANCE SERVICE

Abstract:

This service exposes Electrocardiogram and Respiration Impedance signals and other related data from a vendor-specific sensor device intended for fitness and health care applications.

Revision History

Revision	Date	Comments
D01r00	2014-10-10	Initial Draft
D01r01	2014-11-14	First definition
D01r02	2015-01-13	First functional version
D01r03	2015-02-03	Changes to detect the sensor contact through a new Characteristic.
D01r04	2015-03-11	Changes to allow a higher data rate between Sensor and Collector.
D01r05	2015-03-17	Added writable configuration Characteristic to allow Collector to decide technical parameters.
D01r06	2015-03-17	Added writable Characteristic to indicate start and stop from the Collector instead of automatic retrieval of data just after the connection is established.
D01r07	2015-06-16	Merged some Characteristics to reduce complexity of the Service. Increased error rate calculation precision.
V10r00	2015-06-25	The service has been implemented and properly tested in the final device.

Contributors

Name
Antonio López Marín
Francisco Ruiz Villar

Document Terminology

The Bluetooth SIG has adopted Section 13.1 of the IEEE Standards Style Manual, which dictates use of the words ‘‘shall’’, ‘‘should’’, ‘‘may’’, and ‘‘can’’ in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Table of Contents

1 Introduction

The Electrocardiogram & Respiration Impedance Service exposes electrocardiogram and Respiration Impedance signals and other data from a vendor-specific EKG & RES Sensor intended for fitness and health care applications.

1.1 Service Dependencies

This service is not dependent upon any other services.

1.2 Conformance

If conformance to this profile is claimed, all capabilities indicated as mandatory for this profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the *Bluetooth* qualification program.

1.3 Bluetooth Specification Release Compatibility

This specification is compatible with any *Bluetooth* Core Specification [1] that includes the Generic Attribute Profile (GATT) specification and the Bluetooth Low Energy Controller specification.

1.4 GATT Sub-Procedure Requirements

Requirements in this section represent a minimum set of requirements for an EKG & RI Sensor (Server). Other GATT sub-procedures may be used if supported by both Client and Server.

Table 1.1 summarizes additional GATT sub-procedure requirements beyond those required by all GATT Servers.

GATT Sub-Procedure	Requirements
Write Characteristic Value	M
Notifications	M
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 1.1: GATT Sub-procedure Requirements

1.5 Transport Dependencies

This service shall operate over an LE transport.

1.6 Error Codes

This service does not define any error code.

1.7 Byte Transmission Order

All characteristics used with this service shall be transmitted with the least significant octet first (i.e., little endian). The least significant octet is identified in the every characteristic definition in this document.

2 Service Declaration

The Electrocardiogram & Respiration Impedance Service shall be instantiated as a «Primary Service».

The service UUID shall be set to the value assigned to «E&R Service» defined in Table 2.1.

Characteristic Name	**UUID**
Electrocardiogram & R. Impedance Service	C8634AB0-6510-11E4-9803-0800200C9A66
Multisample Measurement	C8634AB1-6510-11E4-9803-0800200C9A66
Status / Configuration Characteristic	C8634AB2-6510-11E4-9803-0800200C9A66

Table 2.1: E&R Service and characteristic identifiers.

3 Service Characteristics

The following characteristics are exposed in the E&R Service. Unless otherwise specified, only one instance of each characteristic is permitted within this service.

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Multisample Measurement	M	Read, Notify		None.
Multisample Measurement Characteristic Configuration descriptor	M	Read, Write		None.
Status/Configuration Characteristic	M	Write		None.

Table 3.1: E&R Service characteristics

- Security Permissions of “None” means that this service does not impose any requirements.
- Properties not listed as Mandatory or Optional are Excluded.

3.1 Multisample Measurement

The Multisample Measurement characteristic is used to send both cardiac signal and Respiration Impedance measurements. Included in the characteristic, there are: a Packet ID field (so the client may notice if there are transmission losses between measurements), an electrocardiogram measurement value field and a Respiration Impedance value field. Each value contains several samples from the above-mentioned signals (see Figure 3.1).

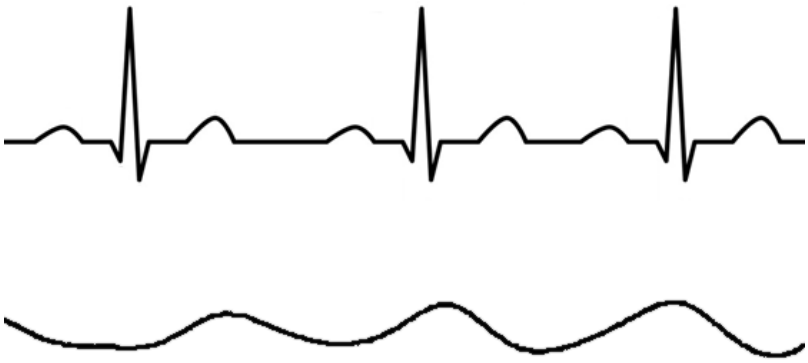


Figure 3.1: EKG waveform and Respiration Impedance signal

3.1.1 Characteristic Behavior

When this characteristic has been subscribed and the *Streaming flag* in the *Status / Configuration Characteristic* is set, it shall be notified while in a connection.

The following diagram presents the memory organization of the different fields and it shall be organized in this way. Each color represents a different field that will be explained later on.

Figure 3.2: Multisample Measurement Characteristic structure

The Multisample Measurement characteristic contains time-sensitive data, thus the requirements for time-sensitive data and data storage defined in section 3.3 apply.

3.1.1.1 ID/Sensor Contact Field

The ID/Sensor Contact field shall be included in the Multisample Measurement Characteristic. This field is fixed length and shall contain 2 bytes, an unsigned integer value between 0 and 65535. The internal structure of this field is as follows:

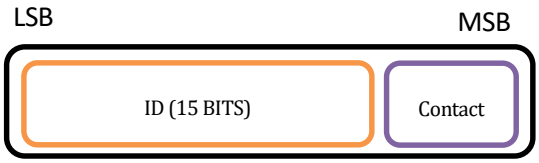


Figure 3.3: ID / Sensor Contact Field structure

Notice the endianness of this field. The client shall receive the most significant octet first though the byte transmission order is little endian, as explained in section 1.8. Moreover, it is divided in two subfields:

- *Sensor Contact* subfield that corresponds to the MSB of the field.
- *ID* subfield, this corresponds to the remaining 15 bits [0, 32767]

Regarding the *Sensor Contact* bit, it shall be set when the contact has not been detected during the sampling of any sample contained in the whole packet. In other case, it shall be cleared.

Two measurements in a row shall give consecutive values in this field, except when passing from 32767 to 0. This values can be used to notice transmission errors and deduce an error rate for the link.

There are several ways to use this information to calculate the loss factor, such as using another counter at the client side. Notice also that **one sequence number** in this field corresponds to **three samples**, as explained in the following sections.

The value of the ID / Sensor Contact field may change during a connection.

3.1.1.2 Electrocardiogram Measurement Value Field

The Electrocardiogram Measurement Value field shall be included in the Multisample Measurement Characteristic.

While a single cardiac signal sample require 24 bits, this field contains 72 bits, i.e. 3 consecutive samples of 3 bytes each. Notice the endianness of the samples. The client shall receive the most significant octet first though the byte transmission order is little endian, as explained in section 1.8. Thereby the client side shall be aware of this issue when storing the samples.

A single 24-bit sample is encoded as an Offset Binary number and shall be interpreted as follows. The possible values are contained in the [0, $2^{24} – 1$] interval and corresponds to a dynamic range of [–2.42, 2.42] volts. This means that the step between consecutive values should be around 0.575 µV.

The value of the Electrocardiogram Measurement Value field may change during a connection.

3.1.1.3 Respiration Impedance Value Field

The Respiration Impedance Value field shall be included in the Multisample Measurement characteristic.

While a single RI sample require 24 bits, this field contains 72 bits, i.e. 3 consecutive samples of 3 bytes each. Notice the endianness of the samples. The client shall receive the most significant octet first though the byte transmission order is little endian, as explained in section 1.8. Thereby the client side shall be aware of this issue when storing the samples.

A single 24-bit sample is encoded as an Offset Binary number and shall be interpreted as follows. The possible values are contained in the [0, $2^{24} – 1$] interval and corresponds to a dynamic range of [0, 10000] ohms. This means that the step between consecutive values should be around 1.2 milliohms.

The value of the Respiration Impedance Value field may change during a connection.

3.1.1.4 Transmission Interval

In typical applications, the Multisample Measurement characteristic is notified approximately 1 time per every 6 milliseconds and includes all the above-mentioned fields. These intervals may vary and are configurable by the Client through the *Status/ Configuration Characteristic*.

3.1.2 Characteristic Descriptors

3.1.2.1 Client Characteristic Configuration Descriptor

The *Client Characteristic Configuration* descriptor shall be included in the Multisample Measurement characteristic.

3.2 Status / Configuration

The Status / Configuration Characteristic shall allow the Client to configure several parameters of the Analog-to-digital conversion in the Server side, e.g. sampling frequency and gain, and to switch the streaming state.

This is a 1-octet fixed-length Characteristic. Support for this characteristic is mandatory though the Server shall operate at a default state whether or not this characteristic is written. The internal structure of this field is as follows:

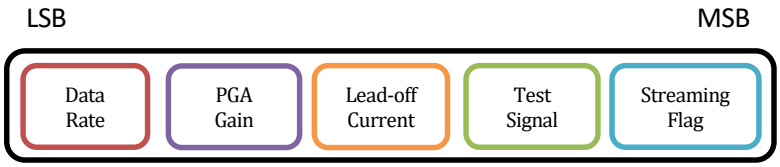


Figure 3.4: Status / Configuration Characteristic structure

3.2.1 Characteristic Behavior

The Status / Configuration Characteristic shall configure the Analog-to-digital conversion of the samples from the Multisample Measurement characteristic when written. The different fields are shown from least to most significance in the bit order.

3.2.1.1 Data Rate bits

This 2-bit field allows the client to set the Sampling Frequency for both EKG and RI channels and the possible values are defined in the following table along with the Notification intervals associated with the *Multisample Measurement characteristic.*

Value	Sampling Frequency	Notification interval
0b00	125	24
0b01	250	12
0b10	500 *(default)*	6
0b11	1000	3

Table 3.1: Data Rate configuration

Notice that the notification interval value may not be a strictly fixed value to avoid congestion in the Server side.

3.2.1.2 Gain bits

This 3-bit field shall allow the Client to set a Gain parameter for both EKG and RI channels of the *Multisample Measurement characteristic* and the possible values are defined in the following table.

Value	PGA Gain
0b000	6
0b001	1
0b010	2
0b011	3
0b100	4
0b101	8
0b110	12
0b111	Discarded or '6'

Table 3.2: Gain configuration

3.2.1.3 Lead-off Current bits

This bit shall allow the Client to set a current used for skin detection through the *ID / Sensor Contact Field* within the *Multisample Measurement Characteristic* and the possible values are defined in the Table 3.3. It is intended that the Client side configure this setting depending on the analysis of the received signals.

Value	Lead-off detection
0	6 nA (default)
1	22 nA

Table 3.3: Lead-off detection current configuration

3.2.1.4 Test Signal bit

When this bit is set, the Server side shall send a well-known signal such as a square wave in both channels instead of the sampled signals, so the Client side can check the adequate operation of the system.

If the value of the *Test Signal bit* is 0, the Server shall send both channel signals as usually.

3.2.1.5 Streaming Flag bit

Using this bit, the Server side shall start or stop the transmission of the channels through the *Multisample Measurement* Characteristics. The client shall be subscribed to this characteristic in order to receive the samples.

If the *Streaming Flag bit* is set, the Server shall start the transmission. On the contrary, the streaming transmission shall stop if the bit is cleared.

3.5 Requirements for Time-Sensitive Data

The Multisample Measurement characteristic contains time sensitive data and is considered a time-sensitive characteristic, thus the following requirements apply:

Since this service does not provide for a time stamp to identify the measurement time (and age) of the data, the value of the Multisample Measurement characteristic shall be discarded if either the connection does not get established or if the notification is not successfully sent to the Client (e.g., link loss).

It is up to the Client to calculate the corresponding timestamp based on the chosen Data rate. Moreover, when using the highest data rate option when streaming some packets may be lost due to congestion at the Server side. Therefore, both encryption and indication are not used for this Service and shall be discouraged.

4 Acronyms and Abbreviations

Acronyms and Abbreviations	Meaning
E&R	Electrocardiogram and Respiration Impedance
EKG	Electrocardiogram
RI	Respiration Impedance
GATT	Generic Attribute Profile
LE	Low Energy
PGA	Power-gain Amplifier
UUID	Universally Unique Identifier

Table 7.1: Acronyms and Abbreviations

5 References

[1] Bluetooth Core Specification v4.0

BLUETOOTH® DOC	Date / Year-Month-Day	Status	Revision	Document No
	2015-06-16	Released	V10r00	ERP_SPEC
Prepared By Antonio López Marín	E-mail Address lopmar.antonio@gmail.com			N.B.

ELECTROCARDIOGRAM & RESPIRATION IMPEDANCE PROFILE

Abstract:

This profile enables a Collector device to connect and interact with an Electrocardiogram and Thoracic impedance sensor for use in fitness applications.

Revision History

Revision	Date	Comments
D01r00	2014-10-10	Initial Draft
D01r01	2014-11-14	First definition
D01r02	2015-01-13	First functional version
D01r03	2015-02-03	Added Appearance Characteristic UUID from GAP to match *E&R Sensor: Heart Rate Belt*
D01r04	2015-03-11	Introduced Battery Service and Battery Level characteristic to the profile.
D01r05	2015-03-17	Added Device Information Service along with Manufacturer Name, Model Number and Firmware Revision string characteristics.
V10r00	2015-06-25	The profile has been implemented and properly tested in the final device.

Contributors

Name
Antonio López Marín
Francisco Ruiz Villar

Document Terminology

The Bluetooth SIG has adopted Section 13.1 of the IEEE Standards Style Manual, which dictates use of the words ``shall'', ``should'', ``may'', and ``can'' in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Table of Contents

1 Introduction

The Electrocardiogram & Respiration Impedance Profile is used to enable a data collection device to obtain data from an E&R Sensor that exposes the E&R Service [1].

1.1 Profile Dependencies

This profile requires the Generic Attribute Profile (GATT).

1.2 Conformance

If conformance to this profile is claimed, all capabilities indicated as mandatory for this profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the *Bluetooth* qualification program.

1.3 Bluetooth Specification Release Compatibility

This specification is compatible with any *Bluetooth* Core Specification [2] that includes the Generic Attribute Profile (GATT) specification and the Bluetooth Low Energy Controller specification.

2 Configuration

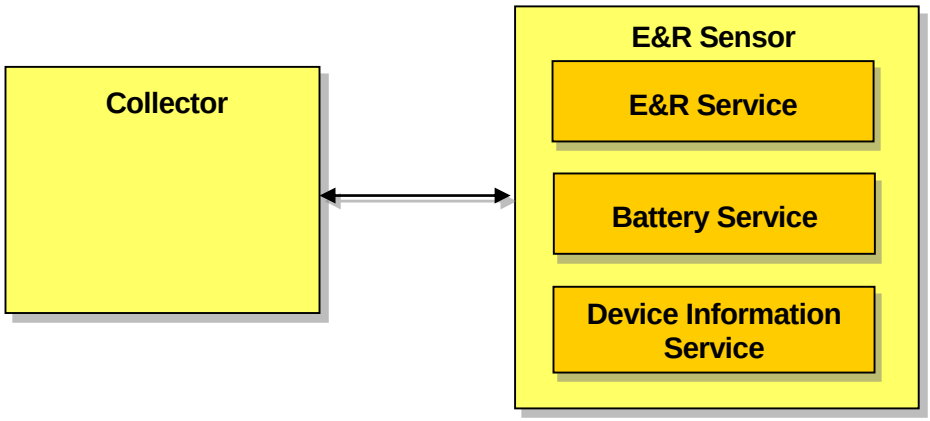
2.1 Roles

The profile defines two roles: E&R Sensor and Collector. The E&R Sensor is the device that measures both electrocardiogram and respiration impedance signals and other information and the Collector is the device that receives the measurement and other data from an E&R Sensor.

- The E&R Sensor shall be a GATT Server.
- The Collector shall be a GATT Client.

2.2 Role/Service Relationships

The following diagram shows the relationships between services and the two profile roles.



Note: Profile roles are represented by yellow boxes and services are represented by orange boxes.

An E&R Sensor instantiates one and only one E&R Service [1] and Battery Service [2] and instantiates one Device Information Service [3].

2.3 Concurrency Limitations and Restrictions

There are no concurrency limitations or restrictions for the Collector and E&R Sensor roles imposed by this profile.

2.4 Topology Limitations and Restrictions

The E&R Sensor shall use the GAP Peripheral role.

The Collector shall use the GAP Central role.

2.5 Transport Dependencies

This profile shall operate over an LE transport.

3 E&R Sensor Role Requirements

The E&R Sensor shall instantiate one and only one E&R Service [1].

The E&R Sensor shall instantiate one and only one Battery Service (BAS) [2].

The E&R Sensor shall instantiate the Device Information Service [3].

Service	E&R Sensor
E&R Service	M
Battery Service (BAS)	M
Device Information Service	M

Table 3.1: E&R Sensor Service Requirements

See Sections 5.1 and 6.1 for additional requirements for the E&R Sensor role.

3.1 Incremental Device Information Service Requirements

The table below shows additional requirements beyond those defined in the Device Information Service.

Device Information Service Characteristic	Requirement
Manufacturer Name String	M
Model Number String	M
Firmware Revision String	M

Table 3.2: Device Information Service Requirements

Some characteristics in this service may be transcoded for use in an ISO/IEEE 11073 ecosystem. See the Personal Health Devices Transcoding White Paper [6] for more information. Since strings in this service are encoded as UTF-8, and IEEE 11073-20601 [5] specifies that strings are encoded as ASCII printable characters (a subset of UTF-8), string characteristics that are to be transcoded for use in an ISO/IEEE 11073 ecosystem must restrict their values to the printable ASCII character set.

4 Collector Role Requirements

The Collector shall support the E&R Service [1].

The Collector may support the Battery Service (BAS) [4].

The Collector may support the Device Information Service [3].

Service	Collector
E&R Service	M
Battery Service (BAS)	O
Device Information Service	O

Table 4.1: Collector Service Requirements

This section describes the profile procedure requirements for a Collector.

Profile Requirement	Section	Support in Collector
Service Discovery	4.2	M
E&R Service Discovery	4.2.1	M
Battery Service (BAS) Discovery	4.2.2	O
Device Information Service Discovery	4.2.3	O
Characteristic Discovery	4.3	M
E&R Service Characteristic Discovery	4.3.1	M
Battery Service (BAS) Characteristic Discovery	4.3.2	O
Device Information Service Characteristic Discovery	4.3.3	O

Table 4.2: Collector Requirements

4.1 GATT Sub-Procedure Requirements

Requirements in this section represent a minimum set of requirements for a Collector (Client). Other GATT sub-procedures may be used if supported by both Client and Server.

The table below summarizes *additional* GATT sub-procedure requirements beyond those required by all GATT Clients.

GATT Sub-Procedure	Collector (Client)
Discover All Primary Services	C.1
Discover Primary Services by Service	C.1
Discover All Characteristics of a Service	C.2
Discover Characteristics by UUID	C.2
Discover All Characteristic Descriptors	M
Read Characteristic Value	M
Write Characteristic Value	M
Notification	M
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 4.3: Additional GATT Sub-Procedure Requirements

C.1: Mandatory to support at least one of these sub-procedures. C.2: Mandatory to support at least one of these sub-procedures.

4.2 Service Discovery

The Collector shall perform primary service discovery using either the GATT *Discover All Primary Services* sub-procedure or the GATT *Discover Primary Services by Service UUID* sub-procedure. Recommended fast connection parameters and procedures for connection establishment are defined in Section 5.2.4.

4.2.1 E&R Service Discovery

The Collector shall perform primary service discovery to discover the E&R Service.

4.2.2 Battery Service (BAS) Discovery

The Collector may perform primary service discovery to discover the Battery Service.

4.2.3 Device Information Service Discovery

The Collector may perform primary service discovery to discover the Device Information Service.

4.3 Characteristic Discovery

As required by GATT, the Collector must be tolerant of additional optional characteristics in the service records of services used with this profile.

4.3.1 E&R Service Characteristic Discovery

The Collector shall use either the GATT *Discover All Characteristics of a Service* sub-procedure or the GATT *Discover Characteristics by UUID* sub-procedure to discover the characteristics of the service.

The Collector shall use the GATT *Discover All Characteristic Descriptors* sub-procedure to discover the characteristic descriptors described in the following sections.

4.3.1.1 E&R Measurement Characteristic

The Collector shall discover the Multisample Measurement characteristic.

The Collector shall discover the *Client Characteristic Configuration* descriptor of the Multisample Measurement characteristic.

4.3.1.2 Status / Configuration Characteristic

The Collector shall discover the Status / Configuration characteristic.

4.3.2 Battery Service Characteristic Discovery

The Collector may discover the characteristics of the Battery Service.

In order for the Collector to discover the characteristics of the Device Information

Service, it shall use either the GATT *Discover All Characteristics of a Service* sub-procedure or the GATT *Discover Characteristics by UUID* sub-procedure.

4.3.2.1 Battery Level Characteristic

The Collector may discover the Battery Level characteristic.

The Collector may discover the *Client Characteristic Configuration* descriptor of the Battery Level characteristic.

4.3.3 Device Information Service Characteristic Discovery

The Collector may discover the characteristics of the Device Information Service. In order

for the Collector to discover the characteristics of the Device Information

Service, it shall use either the GATT *Discover All Characteristics of a Service* sub-procedure or the GATT *Discover Characteristics by UUID* sub-procedure.

4.4 E&R Measurement

The Collector shall control the configuration of notifications (i.e., via the Client Characteristic Configuration descriptor) of the Multisample Measurement characteristic.

The Collector shall be able to receive multiple notifications of the Multisample Measurement characteristic from the E&R Sensor.

When a Collector requires a connection to an E&R Sensor to receive measurements it shall follow the connection procedures described in Section 5.

The Collector shall know the kind of contents of the Multisample Measurement characteristic based on the value written on the Status / Configuration characteristic.

The update rate of the Multisample characteristic depends upon the data rate set in the Status / Configuration characteristic. The higher the data rate, the larger the number of Multisample events per second and the larger the number of RR-Interval sub-fields in the characteristic.

4.5 Battery Service Characteristics

The Collector may read the value of Device Information Service characteristics.

4.6 Device Information Service Characteristics

The Collector may read the value of Device Information Service characteristics.

5 Connection Establishment Procedures

This section describes the connection establishment and connection termination procedures used by an E&R Sensor and Collector in certain scenarios.

The following scenario description is informative:

An E&R Sensor will typically remain powered on between uses and will advertise and allow a Collector to connect the whole time. In this scenario, the E&R Sensor will remain in a GAP Connectable Mode. The Collector will typically execute a GAP connection establishment procedure such that it is scanning for the E&R Sensor. When a connection is established and E&R Sensor is configured for the notifications by the Collector, the E&R Sensor shall send notifications to the Collector at regular intervals. When the training or diagnostic session is ended on the Collector, the Collector typically terminates the connection.

5.1 E&R Sensor Connection Establishment

This section describes connection procedures to address the following scenarios:

- Section 5.1.1 describes the connection procedure when the E&R Sensor does not support bonding.
- Section 5.1.2 is used when the established connection is broken after a link loss.

5.1.1 Connection Procedure for Unbonded Devices

This procedure is used for connection establishment when the E&R Sensor is not bonded with any Collectors and ready for connection.

The E&R Sensor should use the GAP General Discoverable Mode with connectable undirected advertising events when establishing a connection.

It is recommended that the E&R Sensor advertises using the parameters in Table 5.1. The interval values in the first row are designed to attempt fast connection to devices during the first 30 seconds; however, if a connection is not established within that time, the interval values in the second row are designed to reduce power consumption for devices that continue to advertise.

Advertising Duration	Parameter	Value
First 30 seconds (fast connection)	Advertising Interval	20 ms to 30 ms
After 30 seconds (reduced power)	Advertising Interval	1 s to 2.5 s

Table 5.1: Recommended Advertising Interval Values

The advertising interval and time to perform advertising should be configured with consideration for user expectations of connection establishment time.

The E&R Sensor shall accept any valid values for connection interval and connection latency set by the Collector until service discovery, bonding and/or encryption (if required) is complete. Only after that should the E&R Sensor request to change to the preferred connection parameters that best suit its use case.

When the E&R Sensor is disconnected by the Collector and it is ready for connection (e.g., when in contact with the body or when commanded by the user), the E&R Sensor should reinitiate the connection procedure. This will enable reconnection and/or connection with other Collectors.

5.1.2 Link Loss Reconnection Procedure

When a connection is terminated due to link loss, an E&R Sensor should attempt to reconnect to the Collector by entering a GAP Connectable Mode using the recommended advertising interval values shown in Table 5.1.

5.2 Collector Connection Establishment

This section describes connection procedures to address the following scenarios:

- Section 5.2.1 describes the connection procedure if the Collector does not support bonding.
- Section 5.2.2 is used when the established connection is broken after a link loss.

A Collector used in a public environment (e.g., a public fitness machine) is likely to connect to a large number of E&R Sensors on a daily basis and is not expected to store connection information. A Collector used in this scenario will typically not bond with an E&R Sensor.

A Collector used in a personal environment (e.g., a personal fitness machine, mobile phone or watch) is likely to connect frequently to a user's E&R Sensor and is expected to store connection information. A Collector used in this scenario will typically bond with an E&R Sensor during the initial connection.

As an early implementation of this profile, only the first case is taken into account for either public or personal environments.

5.2.1 Connection Procedure for Unbonded Devices

This procedure is used for connection establishment when the Collector connects to an E&R Sensor to which it is not bonded. A Collector will typically execute a connection establishment procedure at the start of a training session such that it scans for a connectable E&R Sensor in the background or when commanded by the user.

The Collector should use the GAP *General Discovery* procedure to discover a E&R Sensor.

A Collector may use one of the following GAP connection procedures based on its connectivity requirements:

- *General Connection Establishment* procedure. The Collector may use this procedure when it requires measurements from one or more E&R Sensors. This procedure allows a Collector to connect to an E&R Sensor discovered during a scan without using the White List.
- *Direct Connection Establishment* procedure. The Collector may use this procedure when it requires measurements from a single E&R Sensor.

A Collector should use the recommended scan interval values shown in Table 5.2. For the first 30 seconds (or optionally continuously for mains powered devices), the Collector should use the first scan window / scan interval pair to attempt fast connection. However, if a connection is not established within that time, the Collector should switch to one of the other scan window / scan interval options as defined below to reduce power consumption.

Scan Duration	Parameter	Value
First 30 seconds (fast connection)	Scan Interval	30 ms to 60 ms*
	Scan Window	30 ms
After 30 seconds (reduced power) - Option 1	Scan Interval	1.28 s
	Scan Window	11.25 ms
After 30 seconds (reduced power) - Option 2	Scan Interval	2.56 s
	Scan Window	11.25 ms

Table 5.2: Recommended Scan Interval and Scan Window Values

* A scan interval of 60 ms is recommended when the Collector is supporting other operations to provide a 50% scan duty cycle versus 100% scan duty cycle.

Option 1 in Table 5.2 uses the same background scanning interval used in BR/EDR so the power consumption for LE will be similar to the power consumption used for background scanning on BR/EDR. Option 2 uses a larger background scanning interval (e.g. twice as long) than used in BR/EDR so the power consumption for LE will be less than the power consumption used for background scanning on BR/EDR. Connection times during background scanning will be longer with Option 2.

The Collector should use a scan window and scan interval suitable to its power and connection time requirements. Increasing the scan window increases the power consumption, but decreases the connection time.

The scan interval and scan window should be configured with consideration for user expectations of connection establishment time.

When the connection is established, the Collector should configure the Client Characteristic Configuration descriptor of the E&R Sensor to enable notifications.

The Collector should terminate the connection when the measurement session is terminated at the Collector by the user.

When the Collector is disconnected, the Collector may reinitiate the connection procedure. This will enable reconnection and/or connection with other E&R Sensors.

5.2.2 Link Loss Reconnection Procedure

When a connection is terminated due to link loss, a Collector should attempt to reconnect to the E&R Sensor using any of the GAP connection procedures with the parameters in Table 5.2.

5.2.3 Fast Connection Interval

To avoid very long service discovery times, the Collector should use the connection intervals defined in Table 5.3 in the connection request.

Parameter	Value
Minimum Connection Interval	50 ms
Maximum Connection Interval	70 ms

Table 5.3: Recommended Fast Connection Interval Values

At any time a lower latency is required, this should be preceded with a connection parameter update to the minimum and maximum connection interval values defined in Table 5.3 and a connection latency of zero. This fast connection interval should be maintained as long as low latency is required. After that, it should switch to the preferred connection parameters as decided by the E&R Sensor using the GAP *Connection Parameter Update* procedure.

6 Acronyms and Abbreviations

Acronyms and Abbreviations	Meaning
AD	Advertising Data
BR/EDR	Basic Rate / Enhanced Data Rate
E&R	Electrocardiogram and Respiration Impedance
EKG	Electrocardiogram
RI	Respiration Impedance
GAP	Generic Access Profile
GATT	Generic Attribute Profile
LE	Low Energy
UUID	Universally Unique Identifier

Table 7.1: Acronyms and Abbreviations

7 References

[1] Electrocardiogram and Respiration Impedance Service

[2] Bluetooth Core Specification v4.0

[3] Device Information Service

[4] Battery Service (BAS)

[5] IEEE Std 11073-20601™- 2008 Health Informatics - Personal Health Device Communication Application Profile - Optimized Exchange Protocol - version 1.0 or later

[6] Personal Health Devices Transcoding White Paper v1.0 or later

[7] Characteristic and Descriptor descriptions are accessible via the Bluetooth SIG Assigned Numbers.

Anexo B – Ficheros BLE112-A

Código B.1 - config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<config>
    <connections value="1" />
    <script_timeout value="0" />
    <throughput optimize="performance" />
</config>
```

Código B.2 - hardware.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <sleeposc enable="true" ppm="30" />
    <sleep enable="true" max_mode="3" />
    <usb enable="false" endpoint="none" />
    <txpower power="15" bias="5" />
    <usart channel="1" alternate="1" baud="1000000" flow="false"
endpoint="api" mode="packet" />
    <wakeup_pin enable="true" port="0" pin="0" />
    <port index="0" tristatemask="0" pull="down" />
    <pmux regulator_pin="7" />
</hardware>
```

Código B.3 - gatt.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
    <!-- Generic Access Profile Service -->
    <service uuid="1800">
        <description>Generic Access Profile</description>
        <characteristic uuid="2A00">
            <description>Device name</description>
            <properties read="true" const="true" />
            <value>WiER monitor</value>
        </characteristic>
        <characteristic uuid="2A01" >
            <description>Appearance</description>
            <properties read="true" const="true" />
            <value type="hex" >4103</value>
        </characteristic>
    </service>
    <!-- Device Information Service -->
    <service uuid="180A" >
      <description>Device Information</description>
        <characteristic uuid="2A29">
            <description>Manufacturer Name String</description>
            <properties read="true" const="true" />
            <value>BCNdevices</value>
        </characteristic>
         <characteristic uuid="2A24">
            <description>Model Number String</description>
            <properties read="true" const="true" />
            <value>WiER monitor</value>
        </characteristic>
        <characteristic uuid="2A26" id="firmware_revision">
            <description>Firmware Revision String</description>
            <properties read="true" />
            <value length="8" />
        </characteristic>
    </service>
    <!-- Battery Service -->
    <service uuid="180F" advertise="true">
        <description>Battery Service</description>
        <characteristic uuid="2A19" id="battery_level">
            <description>Battery Level</description>
            <properties read="true" notify="true" />
            <value length="1" />
        </characteristic>
    </service>
    <!-- ECG / Respiration Impedance Service -->
    <service uuid="C8634AB0-6510-11E4-9803-0800200C9A66" advertise="true">
        <description>ECG/RI Monitor</description>
        <characteristic uuid="C8634AB1-6510-11E4-9803-0800200C9A66">
            <description>Multisample</description>
            <properties read="true" notify="true"/>
            <value length="20"/>
        </characteristic>
        <characteristic uuid="C8634AB2-6510-11E4-9803-0800200C9A66">
            <description>Configuration</description>
            <properties write="true"/>
            <value length="1"/>
        </characteristic>
    </service>
</configuration>
```

Anexo C – Código PIC18F24K20

Código C.1 - main.h

```
#ifndef MAIN_H
#define MAIN_H

/*! Type definitions for integer variables */
typedef unsigned char           uint8_t;    /*!< 8-bit unsigned integer */
typedef signed char             int8_t;     /*!< 8-bit signed integer */
typedef unsigned int            uint16_t;   /*!< 16-bit unsigned integer */
typedef signed int              int16_t;    /*!< 16-bit signed integer */
typedef unsigned long           uint32_t;   /*!< 32-bit unsigned integer */
typedef signed long             int32_t;    /*!< 32-bit signed integer */

#define FIRMWARE_REV            "220615-C"  /*!< Firmware Revision string */
#define FSTR_SIZE               8       /*!< Firmware Revision str size */

#define BIT0                    0x01    /*!< Bitmask for the bit 0 */
#define BIT1                    0x02    /*!< Bitmask for the bit 1 */
#define BIT2                    0x04    /*!< Bitmask for the bit 2 */
#define BIT3                    0x08    /*!< Bitmask for the bit 3 */
#define BIT4                    0x10    /*!< Bitmask for the bit 4 */
#define BIT5                    0x20    /*!< Bitmask for the bit 5 */
#define BIT6                    0x40    /*!< Bitmask for the bit 6 */
#define BIT7                    0x80    /*!< Bitmask for the bit 7 */

/* PORT A definitions */
#define BAT_ADC                 BIT0    /*!< PORT[A] - Battery ADC input */

/* PORT B definitions */
#define ADS_RST                 BIT0    /*!< PORT[B] - ADS Reset output */
#define ADS_RDY                 BIT1    /*!< PORT[B] - ADS Data ready */
#define ADS_CS                  BIT2    /*!< PORT[B] - ADS CS output */
#define LED_YLW                 BIT3    /*!< PORT[B] - LED (ECG) output */
#define LED_BLU                 BIT4    /*!< PORT[B] - LED (BLE) output */
#define LED_RED                 BIT5    /*!< PORT[B] - LED (ON) output */

/* PORT C definitions */
#define BAT_EN                  BIT0    /*!< PORT[C] - Bat MEnable output */
#define BLE_RST                 BIT1    /*!< PORT[C] - BLE112 RST output */
#define BLE_WUP                 BIT2    /*!< PORT[C] - BLE112 WUP output */
#define SPI_CLK                 BIT3    /*!< PORT[C] - SPI CLK output */
#define SPI_IN                  BIT4    /*!< PORT[C] - SPI ata input */
#define SPI_OUT                 BIT5    /*!< PORT[C] - SPI ata output */
#define UART_TX                 BIT6    /*!< PORT[C] - UART data input */
#define UART_RX                 BIT7    /*!< PORT[C] - UART data output */

/*! Mulisample Characteristic size in bytes for msg_payload_t */
#define BLE_PAYLOAD_SIZE        20

/*! Number of sampling times before writing the GH_SAMPLES Characteristic */
#define BLE_SAMP_TIMES          3
```

```
/*! States used in the finite state machine WIERmonitor_t */
typedef enum {
    /** Bluetooth Advertisement state - Fast-connection mode */
    ADV_FAST_CONNECTION,
    /** Bluetooth Advertisement state - Power-save mode */
    ADV_POWER_SAVE,
    /** New connection state (-> Linked) */
    CONNECTION,
    /** Client disconnection state (-> Advertisement) */
    DISCONNECTION,
    /** Linked state - Client is connected */
    LINKED,
    /** Streaming state - Client is connected and receiving data */
    STREAMING,
    /** Configuration state - Client is connected and receiving data */
    CONFIGURATION,
} fsm_state_t;

/*! FSM 1-byte status union to manage changes between states. */
typedef union {
    uint8_t value;                  /*!< Byte variable for direct access */

    struct {
        unsigned CONNECTED : 1;     /*!< Client connected flag */
        unsigned STREAMING : 1;     /*!< Streaming turned on flag */
        unsigned BLECLLBCK : 1;     /*!< Awaited reply from BLE112 flag */
        unsigned ADSNEWDRY : 1;     /*!< New ADS1292R sample ready flag */
        unsigned ADSPSDSMP : 2;     /*!< Processed samples to be sent */
        unsigned LOFFSTAT :  1;     /*!< Sensor Contact NOT detected flag */
        unsigned BUSYBLE :   1;     /*!< BLE112 busy flag */
    };
} status_byte_t;

/*! Configuration union to understand the new configuration requested by
 *  the Client for sampling. */
typedef union {
    uint8_t byte;                   /*!< Byte variable for direct access */

    struct {
        unsigned DR :   2;          /*!< Data Rate (4 possible values) */
        unsigned GAIN : 3;          /*!< PGA Gain (8 possible values) */
        unsigned LOFF : 1;          /*!< Lead-off current (2 possible
values) */
        unsigned TEST : 1;          /*!< TEST signal set up (boolean) */
        unsigned SEND : 1;          /*!< STREAMING start signal (boolean) */
    };
} ads_config_t;

/*! Structure to manage the connection between MCU and BLE112 */
typedef struct {
    uint8_t   handle;             /*!< Actual Connection Handler. */
    uint16_t  interval_min;       /*!< Min BLE Conn Interval allowed. */
    uint16_t  interval_max;       /*!< Max BLE Conn Interval allowed. */
    uint16_t  timeout;            /*!< Connection Supervision Timeout. */
    uint16_t  latency;            /*!< Slave Latency. */
} connect_param_t;
```

```
/*! ADS1292R Channel 24 bits union to process after reception. */
typedef union {
    uint8_t bytearray[3];       /*!< 3-byte array id for reception. */

    struct {
        uint8_t hi;             /*!< 24 bits variable High byte. */
        uint8_t mi;             /*!< 24 bits variable Middle byte. */
        uint8_t lo;             /*!< 24 bits variable Low byte. */
    };

    struct {
        unsigned :      7;
        unsigned MSB :  1;      /*!< Most Significant Bit. */
        unsigned :      8;
        unsigned LSB :  1;      /*!< Least Significant Bit. */
        unsigned :      7;
    };
} ads_sample_t;

/*! BLE112 Multisample 20 bytes union to be sent to the connected */
typedef union {
    char bytearray[BLE_PAYLOAD_SIZE];       /*!< Array id for the TX. */

    struct {
        ads_sample_t    ECG_N1;             /*!< ECG Channel 1st sample. */
        ads_sample_t    ECG_N2;             /*!< ECG Channel 2nd sample. */
        ads_sample_t    ECG_N3;             /*!< ECG Channel 3rd sample. */
        ads_sample_t    RES_N1;             /*!< RES Channel 1st sample. */
        ads_sample_t    RES_N2;             /*!< RES Channel 2nd sample. */
        ads_sample_t    RES_N3;             /*!< RES Channel 3rd sample. */
        uint16_t        sts_ctr;            /*!< Sensor Contact/Counter. */
    };
} msg_payload_t;

/*! FSM structure to be accessed from any part of the code. */
typedef struct {
    fsm_state_t         state;          /*!< Current state of the FSM. */
    status_byte_t       status;         /*!< Several status flags. */
    ads_config_t        config;         /*!< ADS1292R configuration word. */
    connect_param_t     cparam;         /*!< BLE112 conn. parameters. */
    uint8_t             battery;        /*!< Battery state value (0-100). */
    uint8_t             timer;          /*!< Generic timer to be used. */
    msg_payload_t       payload;        /*!< MSG payload to be sent. */
} WIERmonitor_t;
```

Código C.2 - main.c

```
/*! PIC Library directive to use OR bitmasks when writing registers */
#define USE_OR_MASKS

#pragma config FOSC = INTIO67, FCMEN = ON, IESO = OFF       /* CONFIG1H */
#pragma config PWRT = OFF, BOREN = OFF, BORV = 30           /* CONFIG2L */
#pragma config WDTEN = OFF, WDTPS = 32768                   /* CONFIG2H */
#pragma config MCLRE = ON, HFOFST = ON, LPT1OSC = OFF       /* CONFIG3H */
#pragma config PBADEN = ON, CCP2MX = PORTC                  /* CONFIG3H */
#pragma config STVREN = ON, LVP = OFF, XINST = OFF          /* CONFIG4L */
#pragma config CP0 = OFF, CP1 = OFF                         /* CONFIG5L */
```

```
#pragma config CPB = OFF, CPD = OFF                                 /* CONFIG5H */
#pragma config WRT0 = OFF, WRT1 = OFF                               /* CONFIG6L */
#pragma config WRTB = OFF, WRTC = OFF, WRTD = OFF                   /* CONFIG6H */
#pragma config EBTR0 = OFF, EBTR1 = OFF                             /* CONFIG7L */
#pragma config EBTRB = OFF                                          /* CONFIG7H */

#include "main.h"
#include "mcu.h"
#include "ads1292.h"
#include "ble112.h"

extern WIERmonitor_t WIERmonitor;       /*!< FSM control variable */

/* Variables to manage the data processing */
/*! Lead-off bits variable to control Sensor Contact */
LOFFbits_t LOFFbits;
/*! Pointer to store the next Current ECG sample */
ads_sample_t *Current_ECG = &WIERmonitor.payload.ECG_N1;
/*! Pointer to store the next Current RES sample */
ads_sample_t *Current_RES = &WIERmonitor.payload.RES_N1;
/*! Counter of sent packets for Error checking in Client side */
uint16_t sent_counter;

/*! This function configures the PIC device and its peripherals. */
void pic18_configuration(void);

/*! High-priority Interrupt Service Routine.
 *  It manages the UART RX interruption from the BLE112 module. */
void interrupt pic18_isr(void);

/*! Low-priority Interrupt Service Routine.
 *  It manages the PORTB.1 interruption from the ADS module
 *  and the Timer 0 overflow interruption. */
void interrupt low_priority pic18_low_isr(void);

/*! This function checks if the Battery voltage has changed.
 *  If that is the case, it is updated in the BLE112 module. */
void pic18_update_battery(void);

void main(void)
{
    pic18_configuration();

    ads_hw_reset_pulse();
    ble_hw_reset_pulse();

    ads_sw_default_config();
    ble_sw_default_config();

    pic18_led_turn_on(LED_RED);

    while (1)
    {
        switch (WIERmonitor.state)
        {
        default:
        case ADV_FAST_CONNECTION:
        {
            if (WIERmonitor.timer >= ADVTIME_CHANGEMODE)
            {
                ble_advert_configuration(POWERSAVE);
```

```
            WIERmonitor.state = ADV_POWER_SAVE;
        }
    }
    break;

    case ADV_POWER_SAVE:
    {

    }
    break;

    case CONNECTION:
    {
        pic18_led_turn_off(LED_BLU);

        /* Connection parameters MUST be renegociated from server */
        ble_cmd_config_connection_interval();
        ble_wait_for_msg();

        WIERmonitor.state = LINKED;
    }
    break;

    case DISCONNECTION:
    {
        pic18_led_turn_off(LED_YLW);

        if (WIERmonitor.status.STREAMING)
        {
            WIERmonitor.status.STREAMING = 0;
            ads_stop_conversion();       /* Stop further conversions */
            ble_hw_system_sleepm();      /* Allow BLE to go to sleep */
        }

        /* Advertisement interval (Fast Connection) */
        ble_advert_configuration(FASTCONN);
        WIERmonitor.state = ADV_FAST_CONNECTION;
    }
    break;

    case LINKED:
    {
        if (WIERmonitor.status.STREAMING)
        {
            pic18_led_turn_on(LED_YLW);

            /* Disable sleepmode on BLE during STREAMING */
            ble_hw_system_wakeup();
            ble_wait_for_msg();

            ads_start_conversion();
            sent_counter = 0;
            WIERmonitor.state = STREAMING;
        }
    }
    break;
```

```
case STREAMING:
{
    if (WIERmonitor.status.STREAMING)
    {
        /* Wait while new data arrive and client is connected */
        while (!WIERmonitor.status.ADSNEWDRY &&
                WIERmonitor.status.CONNECTED);


        if (WIERmonitor.status.CONNECTED)
        {
            /* Transf. samples Two's Complement to Offset Binary */
            Current_RES->MSB ^= 1;
            Current_ECG->MSB ^= 1;

            /* Checking Sensor Contact status from the sample */
            WIERmonitor.status.LOFFSTAT |=
                    LOFFbits.IN1P | (LOFFbits.IN2N & LOFFbits.IN1N);

            WIERmonitor.status.ADSPSDSMP++;

            if (WIERmonitor.status.ADSPSDSMP == BLE_SAMP_TIMES)
            {
                /* Changing Counter endianness */
                WIERmonitor.payload.sts_ctr = ((sent_counter >> 8)
                          &0x7F) | (sent_counter << 8);

                /* Adding Sensor Contact info to the variable */
                if (WIERmonitor.status.LOFFSTAT)
                    WIERmonitor.payload.sts_ctr |= (1 << 7);

                /* Send the packet with all the samples */
                if (WIERmonitor.status.CONNECTED)
                ble_cmd_samples_send(WIERmonitor.payload.bytearray);

                /* Preparing variables for next packet */
                sent_counter++;

                WIERmonitor.payload.sts_ctr = 0;
                WIERmonitor.status.LOFFSTAT = 0;
                LOFFbits.loff[0] = 0; LOFFbits.loff[1] = 0;

                Current_RES = &WIERmonitor.payload.RES_N1;
                Current_ECG = &WIERmonitor.payload.ECG_N1;
                WIERmonitor.status.ADSPSDSMP = 0;
            }
            else
            {
                Current_RES++;
                Current_ECG++;
            }
            /* Clear new data flag */
            WIERmonitor.status.ADSNEWDRY = 0;
        }
    }
}
break;

case CONFIGURATION:
{
    ads_stop_conversion();
```

```
            ble_hw_system_sleepm();
            ads_sw_new_configuration(WIERmonitor.config);

            WIERmonitor.state = LINKED;
        }
        break;
        }
    }
}

void pic18_configuration(void)
{
    bglib_output = ble_write_msg;
    WIERmonitor.status.CONNECTED = 0;
    WIERmonitor.status.STREAMING = 0;

    /* LATx register set pins values to high state (3V)
     * TRISx register set pins as inputs. */

    LATA = 0;
    TRISA = BAT_ADC;

    LATB = ADS_RST + ADS_CS;
    TRISB = ADS_RDY;

    LATC = BLE_RST;
    TRISC = SPI_IN + UART_RX;

    ANSEL = ADCPORTCFG;
    ANSELH = ADCPORTCFG >> 8;

    /* Oscillator 8 x 4(PLL) = 32 MHz */

    OSCCON = 0b01100000; OSCTUNE = 0b11000000;

    /* ADC (10 bits) module configuration */

    ADCON0bits.CHS = 0;             /* Channel 0 */
    ADCON1bits.VCFG = 0;            /* Vdd-Vss reference */
    ADCON2bits.ADFM = 1;            /* Right-justified */
    ADCON2bits.ADCS = 0b010;        /* Fosc_32 = 1MHz -> TAD = 1us */
    ADCON2bits.ACQT = 0b101;        /* 12 TAD = 12 us > TACQ = 7.45 us */
    ADCON0bits.ADON = 1;            /* ADC Enable */

    /* Timer 0 module configuration */

    TMR0H = TMR0_PRELOAD >> 8;
    TMR0L = TMR0_PRELOAD;
    T0CON = TMR0_CONFIG;

    /* EUSART module configuration */

    TXSTA = 0;
    RCSTA = 0;                      /* Reset USART registers to POR state */
    RCSTAbits.CREN = 1;
    TXSTAbits.BRGH = 1;

    BAUDCON = BAUD_CONFIG;
    SPBRG = SPBRG_UART;
```

```
    if (BAUD_CONFIG & BAUD_16_BIT_RATE)
         SPBRGH = SPBRG_UART >> 8;
     TXSTAbits.TXEN = 1;             /* Enable transmitter */
     RCSTAbits.SPEN = 1;             /* Enable receiver */

     /* MSSP module configuration */

     SSPSTATbits.SMP = 1;            /* Data sampled at the end */
     SSPCON1bits.SSPM = SPI_FOSC_64; /* SCLK = 512KHz */
     SSPCON1bits.SSPEN = 1;          /* SPI enabled */

     /* Hardware Interrupts configuration */

     INTCON2bits.TMR0IP = 0;         /* Timer0 is low priority interrupt */
     INTCONbits.TMR0IF = 0;          /* Clear Timer0 overflow flag */
     INTCONbits.TMR0IE = 1;          /* Enable Timer0 overflow interrupt */
     T0CONbits.TMR0ON = 1;           /* Enable Timer0 */

     PIR1bits.TXIF = 0;              /* Clear UART TX flag */
     PIR1bits.RCIF = 0;              /* Clear UART RX flag */
     IPR1bits.RCIP = 1;              /* High priority UART RX interrupt */
     PIE1bits.RCIE = 1;              /* Enable UART RX interrupt */

     INTCON2bits.INTEDG1 = 0;        /* Interrupt on falling edge of INT1 */
     INTCON2bits.RBPU = 0;           /* Enable pull ups on PORTB */
     INTCON3bits.INT1IF = 0;         /* Ensure flag is cleared */
     INTCON3bits.INT1IP = 0;         /* Low priority INT1 interrupt */
     INTCON3bits.INT1E = 1;          /* Keep INT1 interrupt enabled */


     RCONbits.IPEN = 1;              /* Enable priority levels on ints */
     INTCONbits.GIEL = 1;            /* Low priority interrupts allowed */
     INTCONbits.GIEH = 1;            /* Interrupts allowed */
}

void interrupt pic18_isr(void)
{
     if (PIC18_UART_FLAG)
         ble_read_msg();
}

void interrupt low_priority pic18_low_isr(void)
{
     if (PIC18_INT1_FLAG)
     {
         PIC18_INT1_FLAG = 0;
         LATB &= ~ADS_CS;

         /* Get the 72 bits from ADS (24 L-Off + 2x(24 per channel)) */
         pic18_spi_gets(LOFFbits.loff, ADS_SAMPLE_LENGTH);
         pic18_spi_gets(Current_RES->bytearray, ADS_SAMPLE_LENGTH);
         pic18_spi_gets(Current_ECG->bytearray, ADS_SAMPLE_LENGTH);

         LATB |= ADS_CS;
         WIERmonitor.status.ADSNEWDRY = 1;           /* Set New Data flag */
     }
     if (PIC18_TMR0_FLAG)
     {
         PIC18_TMR0_FLAG = 0;
         TMR0H = TMR0_PRELOAD >> 8;
         TMR0L = TMR0_PRELOAD;
```

```
        WIERmonitor.timer++;
        if(WIERmonitor.status.CONNECTED & !WIERmonitor.status.STREAMING)
            pic18_led_switch(LED_YLW);
        else if(!WIERmonitor.status.CONNECTED)
            pic18_led_switch(LED_BLU);
    }
}

void pic18_update_battery(void)
{
    static uint8_t bat_prev = 0;
    if (!(WIERmonitor.timer % 180))
    {
        /* Calculate Battery % from ADC */
        WIERmonitor.battery = pic18_read_battery();

        if (!WIERmonitor.status.BUSYBLE &&
            (WIERmonitor.battery != bat_prev))
        {
            ble_cmd_attributes_write(GH_BATTERY, 0, 1,
                                     &WIERmonitor.battery);
            bat_prev = WIERmonitor.battery;
        }
    }
}
```

Código C.3 - mcu.h

```
#ifndef MCU_H
#define MCU_H

#include <xc.h>
#include "main.h"

/*! Oscillator Frequency definition for delay routines */
#define _XTAL_FREQ      32000000
/*! Baudrate configuration word for the UART module */
#define BAUD_CONFIG     BAUD_IDLE_CLK_LOW | BAUD_8_BIT_RATE |
BAUD_WAKEUP_OFF | BAUD_AUTO_OFF

/*! Baudrate SPBRG number corresponding to 1 Mbps @ f_OSC = 32MHz */
#define SPBRG_UART      1
/*! Baudrate used for comm (Not actually used, just as reference) */
#define BAUD_RATE       1000000
/*! Analog to Digital Converter Port (Port A0 activated) */
#define ADCPORTCFG      ADC_1ANA
/*! Timer 0 preload (Corresponds to 1 second) */
#define TMR0_PRELOAD    0x0BDC

/*! Timer 0 configuration word */
#define TMR0_CONFIG     T0_16BIT | T0_SOURCE_INT | T0_EDGE_RISE |
T0_PS_1_128
/*! PIC18F24K20 UART Interrupt flag bit */
#define PIC18_UART_FLAG             PIR1bits.RCIF
/*! PIC18F24K20 PORTB-RB1 Interrupt flag bit */
#define PIC18_INT1_FLAG             INTCON3bits.INT1IF
/*! PIC18F24K20 TIMER0 Interrupt flag bit */
#define PIC18_TMR0_FLAG             INTCONbits.TMR0IF
```

```
/* Experimental ADC values - Change depending on the actual battery state */
/*! Battery Timer mask to use with Timer 0 counter */
#define BATINTERVAL     0x3
/*! ADC (10bits) that corresponds to the 80% of the battery capacity */
#define BAT_80          939
/*! ADC (10bits) that corresponds to an empty battery */
#define BAT_EMPTY       731
/*! ADC (10bits) result that corresponds to a fully-charged battery */
#define BAT_FULL        997

/*! List the different speeds at which SPI protocol runs */
typedef enum {
    SPI_SLOW,
    SPI_FAST
} spi_rate_t;

/*! This function will write a single byte data via UART */
void pic18_uart_putc(uint8_t data);

/*! This function will write an array of bytes via UART */
void pic18_uart_puts(uint8_t *data, uint16_t length);

/*! This function will read a single byte from the UART */
uint8_t pic18_uart_getc(void);

/*! This function will read an array of bytes via UART and will write it to
 *  the pointed buffer. */
void pic18_uart_gets(uint8_t *data, uint16_t length);

/*! This function will write a single byte data via SPI */
void pic18_spi_putc(uint8_t data);

/*! This function will write an array of bytes via SPI */
void pic18_spi_puts(uint8_t *data, uint16_t length);

/*! This function will read a single byte from the UART */
char pic18_spi_getc(void);

/*! This function will read an array of bytes via SPI and will write it to
 *  the pointed buffer. */
void pic18_spi_gets(uint8_t *data, uint16_t length);

/*! This function will setup the SPI protocol baudrate and will reset the
 *  peripheral before returning. */
void pic18_spi_rate(spi_rate_t rate);

/*! This function will read the ADC value (10 bits) and will return the
 *  estimated battery percentage through an 8-bit variable. */
uint8_t pic18_read_battery(void);

/*! Define-function to turn a specific LED on */
#define pic18_led_turn_on(led)      (LATB|= led)
/*! Define-function to turn a specific LED off */
#define pic18_led_turn_off(led)     (LATB&= ~led)
/*! Define-function to toggle a specific LED */
#define pic18_led_switch(led)       (LATB^= led)

#endif  /* MCU_H */
```

Código C.4 - mcu.c

```
#include "mcu.h"

void pic18_uart_putc(uint8_t data)
{
    while(!TXSTAbits.TRMT);
    TXREG = data;
}

void pic18_uart_puts(uint8_t *data, uint16_t length)
{
    for (uint16_t i = 0; i < length; i++)
        pic18_uart_putc(data[i]);
}

char pic18_uart_getc(void)
{
    return RCREG;
}

void pic18_uart_gets(uint8_t *data, uint16_t length)
{
    for (uint16_t i = 0; i < length; i++)
    {
        while(!PIR1bits.RCIF);
        data[i] = RCREG;
    }
}

void pic18_spi_putc(uint8_t data)
{
    PIR1bits.SSPIF = 0;
    SSPBUF = data;
    while (!PIR1bits.SSPIF);
}

void pic18_spi_puts(uint8_t *data, uint16_t length)
{
    for (uint16_t i = 0; i < length; i++)
        pic18_spi_putc(data[i]);
}

uint8_t pic18_spi_getc(void)
{
  PIR1bits.SSPIF = 0;
  SSPBUF = 0x00;
  while(!PIR1bits.SSPIF);

  return  SSPBUF;
}

void pic18_spi_gets(uint8_t *data, uint16_t length)
{
    for (uint16_t i = 0; i < length; i++)
        data[i] = pic18_spi_getc();
}

void pic18_spi_rate(spi_rate_t rate)
{
    LATC &= ~SPI_OUT;
```

```
    SSPCON1bits.SSPEN = 0;

    switch(rate)
    {
    default:
    case SPI_SLOW:
        SSPCON1bits.SSPM = SPI_FOSC_64;
    case SPI_FAST:
        SSPCON1bits.SSPM = SPI_FOSC_16;
    }

    SSPCON1bits.SSPEN = 1;
}

uint8_t pic18_read_battery(void)
{
    uint32_t bat_adc = 0;
    uint8_t bat_x100 = 0;

    LATC |= BAT_EN;
    ADCON0bits.GO = 1;
    while (ADCON0bits.NOT_DONE);
    LATC &= ~BAT_EN;

    bat_adc = ((uint16_t)ADRESH) << 8 | ADRESL;

    bat_x100 = (bat_adc - BAT_EMPTY)* 100 / 256;

    return bat_x100;
}
```

Código C.5 - ads1292.h

```
#ifndef ADS1292_H
#define ADS1292_H

#include "mcu.h"
#include "main.h"

/* ADS MANUFACTURER DEFINES */
/*! ADS Sample length (3 bytes) */
#define ADS_SAMPLE_LENGTH       3
/*! ADS Data length (2 ch + 1 loff) */
#define ADS_DATA_LENGTH         9

/* ADS1292 Command Definitions */
/* Wait 7.8us prior to send any of these codes */
/*! Wake-up from standby mode Opcode */
#define ADS_CMD_WAKEUP      0x02
/*! Enter standby mode Opcode */
#define ADS_CMD_STANDBY     0x04
/*! Device registers Reset Opcode */
#define ADS_CMD_RESET       0x06

/*! Start/restart (synchronize) conversions Opcode */
#define ADS_CMD_START       0x08
/*! Stop conversions Opcode */
#define ADS_CMD_STOP        0x0A
```

```
/*! Channel offset calibration Opcode */
#define ADS_CMD_OFFSETCAL   0x1A
/*! Enable Read Data Continuous mode. */
#define ADS_CMD_RDATAC      0x10
/*! Stop Read Data Continuously mode. */
#define ADS_CMD_SDATAC      0x11
/*! Read data by command. */
#define ADS_CMD_RDATA       0x12
/*! Read Configuration Registers. */
#define ADS_CMD_RREG        0x20
/*! Write Configuration Registers. */
#define ADS_CMD_WREG        0x40

/*! List the different addresses of the ADS1292R registers */
typedef enum
{
    ADSREG_ID = 0,          /*!< (0) ID Control Register */
    ADSREG_CONFIG1 = 1,     /*!< (1) Configuration Register 1 */
    ADSREG_CONFIG2 = 2,     /*!< (2) Configuration Register 2 */
    ADSREG_LOFF = 3,        /*!< (3) Lead-Off Control Register */
    ADSREG_CH1SET = 4,      /*!< (4) Channel 1 Settings Register */
    ADSREG_CH2SET = 5,      /*!< (5) ID Control Register */
    ADSREG_RLD_SENS = 6,    /*!< (6) Right Leg Drive Sense Selec Register */
    ADSREG_LOFF_SENS = 7,   /*!< (7) Lead-Off Sense Selection Register */
    ADSREG_LOFF_STAT = 8,   /*!< (8) Lead-Off Status Register */
    ADSREG_RESP1 = 9,       /*!< (9) Respiration Control Register 1 */
    ADSREG_RESP2 = 10       /*!< (10) Respiration Control Register 2 */
} ads_registers;

/* (0) ID Control Register - Address: 0x00 */
#define ADS1292R            0b01110011  /*!< Revision ID for ADS1292R */

/* (1) Configuration Register 1 - Address: 0x01 */
#define S_SHOT              0b10000000  /*!< Single-shot conversion mode */
#define C_CONV              0b00000000  /*!< Continuous conversion mode */
#define DR_125              0b00000000  /*!< CH Oversampling : 125 SPS */
#define DR_250              0b00000001  /*!< CH Oversampling : 250 SPS */
#define DR_500              0b00000010  /*!< CH Oversampling : 500 SPS */
#define DR_1k               0b00000011  /*!< CH Oversampling : 1000 SPS */
#define DR_2k               0b00000100  /*!< CH Oversampling : 2000 SPS */
#define DR_4k               0b00000101  /*!< CH Oversampling : 4000 SPS */
#define DR_8k               0b00000110  /*!< CH Oversampling : 8000 SPS */

/* (2) Configuration Register 2 - Address: 0x02 */
#define PDB_LOFF_COMP       0b11000000  /*!< Lead-off comparator enable */
#define PDB_REFBUF          0b10100000  /*!< Reference buffer enable */
#define VREF_4V             0b10010000  /*!< 4-V reference */
#define CLK_EN              0b10001000  /*!< CLK connection */
#define INT_TEST            0b10000010  /*!< Test signal selection */
#define TEST_FREQ           0b10000001  /*!< Test signal freq (1-Hz sq) */

/* (3) Lead-Off Control Register - Address: 0x03 */
#define COMP_TH_95          0b00010000  /*!< Comparator threshold 95% */
#define COMP_TH_925         0b00110000  /*!< Comparator threshold 92.5% */
#define COMP_TH_90          0b01010000  /*!< Comparator threshold 90% */
#define COMP_TH_875         0b01110000  /*!< Comparator threshold 87.5% */
#define COMP_TH_85          0b10010000  /*!< Comparator threshold 85% */
#define COMP_TH_80          0b10110000  /*!< Comparator threshold 80% */
#define COMP_TH_75          0b11010000  /*!< Comparator threshold 75% */
#define COMP_TH_70          0b11110000  /*!< Comparator threshold 70% */
```

```
#define ILEAD_OFF_6N        0b00010000  /*!< Lead-off current 6nA */
#define ILEAD_OFF_22N       0b00010100  /*!< Lead-off current 22nA */
#define ILEAD_OFF_6U        0b00011000  /*!< Lead-off current 6uA */
#define ILEAD_OFF_22U       0b00011100  /*!< Lead-off current 22uA */
#define FLEAD_OFF_DC        0b00010000  /*!< Lead-off DC mode */
#define FLEAD_OFF_AC        0b00010001  /*!< Lead-off AC mode */

/* (4,5) Channel 1,2 Settings - Address: 0x04, 0x05 */
/*! Channel Power-down */
#define POWER_DOWN          0b10000001
#define GAIN_6              0b00000000  /*!< Channel PGA gain 6 */
#define GAIN_1              0b00010000  /*!< Channel PGA gain 1 */
#define GAIN_2              0b00100000  /*!< Channel PGA gain 2 */
#define GAIN_3              0b00110000  /*!< Channel PGA gain 3 */
#define GAIN_4              0b01000000  /*!< Channel PGA gain 4 */
#define GAIN_8              0b01010000  /*!< Channel PGA gain 8 */
#define GAIN_12             0b01100000  /*!< Channel PGA gain 12 */
#define ELECTRODE           0b00000000  /*!< Channel Input: Electrode */
#define INPUT_SHORTED       0b00000001  /*!< Channel Input: Input shorted */
#define RLD_MEASURE         0b00000010  /*!< Channel Input: RLD Measure */
/*! Channel Input: Supply measurement */
#define MVDD                0b00000011
#define TEMP_SENSOR         0b00000100  /*!< Channel Input: Temp Sensor */
#define TEST_SIGNAL         0b00000101  /*!< Channel Input: Test signal */
#define RLD_DRP             0b00000110  /*!< Channel Input: RLDIN + */
#define RLD_DRM             0b00000111  /*!< Channel Input: RLDIN - */
#define RLD_DRPM            0b00001000  /*!< Channel Input: RLDIN +/- */
#define IN3_P_N             0b00001001  /*!< Channel Input: IN3 +/- */

/* (6) Right Leg Drive Sense Selection - Address: 0x06 */
/* PGA Chop frequency */
#define CHOP_DIV16          0b00000000
#define CHOP_DIV2           0b10000000
#define CHOP_DIV4           0b11000000
#define PDB_RLD             0b00100000  /*!< RLD buffer enable */
#define RLD_LOFF_SENSE      0b00010000  /*!< RLD lead-off sense function */
#define RLD2N               0b00001000  /*!< RLD connected to IN2N */
#define RLD2P               0b00000100  /*!< RLD connected to IN2P */
#define RLD1N               0b00000010  /*!< RLD connected to IN1N */
#define RLD1P               0b00000001  /*!< RLD connected to IN1P */

/* (7) Lead-Off Sense Selection - Address: 0x07 */
#define NO_LOFFS            0b00000000  /*!< No Lead-off sense selected */
#define FLIP2               0b00100000  /*!< CH2 Current direction */
#define FLIP1               0b00010000  /*!< CH1 Current direction */
#define LOFF2N              0b00001000  /*!< Input(-) for CH2 detection */
#define LOFF2P              0b00000100  /*!< Input(+) for CH2 detection */
#define LOFF1N              0b00000010  /*!< Input(-) for CH1 detection */
#define LOFF1P              0b00000001  /*!< Input(+) for CH1 detection */

/* (8) Lead-Off Status - Address: 0x08 */
#define CLK_DIV_4       0b00000000  /*!< \f$f_{MOD}=\frac{f_{CLK}}{4}\f$ */
#define CLK_DIV_16      0b01000000  /*!< \f$f_{MOD}=\frac{f_{CLK}}{16}\f$ */
#define RLD_STAT_NC         0b00010000  /*!< RLD Not Connected */
#define IN2N_OFF            0b00001000  /*!< CH2 Electrode(-) not conn */
#define IN2P_OFF            0b00000100  /*!< CH2 Electrode(+) not conn */
#define IN1N_OFF            0b00000010  /*!< CH1 Electrode(-) not conn */
#define IN1P_OFF            0b00000001  /*!< CH1 Electrode(+) not conn */
```

```
/* (9) Respiration Control Register 1 - Address: 0x09 */
/*! Respiration <B>demodulation</B> circuitry enable */
#define RESP_DEMOD_EN1      0b10000010
/*! Respiration <B>modulation</B> circuitry enable */
#define RESP_MOD_EN         0b01000010
/*! Respiration phase */
#define RESP_PH_00              0b00000010
#define RESP PH 01              0b00000110
#define RESP_PH_02              0b00001010
#define RESP_PH_03              0b00001110
#define RESP_PH_04              0b00010010
#define RESP_PH_05              0b00010110
#define RESP_PH_06              0b00011010
#define RESP_PH_07              0b00011110
#define RESP_PH_08              0b00100010
#define RESP_PH_09              0b00100110
#define RESP_PH_10              0b00101010
#define RESP_PH_11              0b00101110
#define RESP_PH_12              0b00110010
#define RESP_PH_13              0b00110110
#define RESP_PH_14              0b00111010
#define RESP_PH_15              0b00111110
/*! Internal Respiration with External clock */
#define RESP_EXT_CLK            0b00000011

/* (10) Respiration Control Register 2 - Address: 0x0A */
#define CALIB_ON            0b10000001  /*!< Offset calibration enable */
#define RESP_FREQ_32K       0b00000001  /*!< Respiration frequency 32 kHz */
#define RESP_FREQ_64K       0b00000101  /*!< Respiration frequency 64 kHz */
#define RLDREF_INT          0b00000011  /*!< Internal RLD reference */

/*! Lead-off Status 24 bits union to properly read after reception. */
typedef union {
    uint8_t loff[3];        /*!< 3-byte array identifier for reception */

    struct {
        /* 1st byte */
        unsigned IN1N : 1;      /*!< Ch 1 - Electrode Disconnection */
        unsigned IN2P : 1;      /*!< Ch 2 + Electrode Disconnection */
        unsigned IN2N : 1;      /*!< Ch 2 - Electrode Disconnection */
        unsigned RLD_STAT : 1;  /*!< Right-leg drive Lead-off Status */
        unsigned : 4;

        /* 2nd byte */
        unsigned : 5;
        unsigned GPIOD1 : 1;    /*!< GPIO 1 Port current value */
        unsigned GPIOD2 : 1;    /*!< GPIO 2 Port current value */
        unsigned IN1P : 1;      /*!< Ch 1 + Electrode Disconnection */

        /* 3rd byte (discarded) */
        unsigned : 8;
    };
} LOFFbits_t;

/*! ADS1292R 72 bits union to allocate all the data in a row. */
typedef union {
    uint8_t buffer[ADS_DATA_LENGTH]; /*!< 9-byte array identifier for RX */

    struct {
        LOFFbits_t LOFFbits;         /*!< 3-byte Lead-off bits structure */
```

```
        ads_sample_t CH1;            /*!< 3-byte Channel 1 structure */
        ads_sample_t CH2;            /*!< 3-byte Channel 2 structure */
    };
} ads_data_t;

 /*! This function sends a single byte <tt>data</tt> to the ADS. */
void ads_write_byte(uint8_t byte);

/*! This function writes a configuration register in the ADS chip. */
void ads_write_register(uint8_t reg, uint8_t config);

/*! Configures the ADS1292R device to its default state. */
void ads_sw_default_config(void);

/*! Keeps RST line low so the ADS will be on after returning. */
void ads_hw_power_on(void);

/*! Keeps RST line high so the ADS will be off after returning. */
void ads_hw_power_off(void);

/*! Send a Reset pulse through the RST line to the ADS. */
void ads_hw_reset_pulse(void);

 /*! This function sends a command to the ADS. */
void ads_cmd(uint8_t command);

 /*! This function sends the proper commands to start the ADS continuous
  *  conversion mode. */
void ads_start_conversion(void);

 /*! This function sends the proper commands to stop the ADS continuous
 *   conversion mode. */
void ads_stop_conversion(void);

/*! Configures the ADS1292R device to the state defined by the externally
 *  defined ads_config_t word (8 bits). */
void ads_sw_new_configuration(ads_config_t config_byte);

#endif  /* ADS1292_H_ */
```

Código C.6 - ads1292.c

```
#include "ads1292.h"

extern WIERmonitor_t WIERmonitor;         /*!< FSM control variable */

void ads_write_byte(unsigned char byte)
{
    pic18_spi_putc(byte);
    __delay_us(8); /* Wait 7.8us (4*Tclk)*/
}

void ads_sw_default_config()
{
    pic18_spi_rate(SPI_SLOW);
    LATB &= ~ADS_CS;
    __delay_us(8);

    ads_write_byte(ADS_CMD_WREG);
```

```
    ads_write_byte(10);
    ads_write_byte(ADS1292R);                   /* ID Control */
    ads_write_byte(C_CONV | DR_500);            /* Configuration 1 */
    ads_write_byte(PDB_LOFF_COMP | PDB_REFBUF); /* Configuration 2 */
                                                /* Lead-off Control */
    ads_write_byte(COMP_TH_95 | ILEAD_OFF_6N | FLEAD_OFF_DC);
    ads_write_byte(GAIN_6 | ELECTRODE);         /* Channel 1 Settings */
    ads_write_byte(GAIN_6 | ELECTRODE);         /* Channel 2 Settings */
                                                /* RLD Sens Selection */
    ads_write_byte(CHOP_DIV16 | PDB_RLD | RLD_LOFF_SENSE);
                                                /* Lead-off Sens Sel. */
    ads_write_byte(LOFF1N | LOFF1P | LOFF2N | LOFF2P);
    ads_write_byte(CLK_DIV_4);                  /* Lead-off Status */
                                                /* Respiration Ctl 1 */
    ads_write_byte(RESP_DEMOD_EN1 | RESP_MOD_EN | RESP_PH_00);
                                                /* Respiration Ctl 2 */
    ads_write_byte(CALIB_ON | RESP_FREQ_32K | RLDREF_INT);

    __delay_us(8);
    LATB |= ADS_CS;
    pic18_spi_rate(SPI_FAST);
}

void ads_write_register(uint8_t reg, uint8_t config)
{
    LATB &= ~ADS_CS;
    __delay_us(8);

    ads_write_byte(ADS_CMD_WREG | reg);
    ads_write_byte(0x00);
    ads_write_byte(config);

    LATB |= ADS_CS;
}

void ads_hw_power_on(void)
{
    LATB |= ADS_RST;
    __delay_ms(10);
}

void ads_hw_power_off(void)
{
    LATB &= ~ADS_RST;
}

void ads_hw_reset_pulse(void)
{
    LATB |= ADS_RST;
    __delay_ms(16);             /* t_{POR} = 32ms */
    __delay_ms(16);
    LATB &= ~ADS_RST;
    __delay_us(8);              /* t_{RST} = 8us */
    LATB |= ADS_RST;
    __delay_us(36);             /* 18 t_{CLK} = 35.16us */
}
```

```
void ads_cmd(uint8_t command)
{
    __delay_us(8);              /* Required to bypass 4 Tclk */
    ads_write_byte(command);
    if (command == ADS_CMD_RESET)
        __delay_us(62);
}

void ads_start_conversion(void)
{
    LATB &= ~ADS_CS;
    ads_cmd(ADS_CMD_OFFSETCAL);
    ads_cmd(ADS_CMD_RDATAC);
    ads_cmd(ADS_CMD_START);
    LATB |= ADS_CS;

    WIERmonitor.status.ADSNEWDRY = 0;
    WIERmonitor.status.ADSPSDSMP = 0;
}

void ads_stop_conversion(void)
{
    LATB &= ~ADS_CS;
    ads_cmd(ADS_CMD_SDATAC);
    ads_cmd(ADS_CMD_STOP);
    LATB |= ADS_CS;

    WIERmonitor.status.ADSNEWDRY = 0;
    WIERmonitor.status.ADSPSDSMP = 0;
}

void ads_sw_new_configuration(ads_config_t config_byte)
{
    uint8_t rconfig1 = C_CONV | config_byte.DR;
    uint8_t rconfig2 = PDB_LOFF_COMP | PDB_REFBUF;
    uint8_t rloff = COMP_TH_95;
    uint8_t rchset = ELECTRODE | (config_byte.GAIN << 4);

    pic18_spi_rate(SPI_SLOW);

    if (config_byte.TEST)
    {
        rconfig2 |= INT_TEST | TEST_FREQ;
        rchset |= TEST_SIGNAL;
    }


    WIERmonitor.status.STREAMING = config_byte.SEND;

    ads_write_register(ADSREG_CONFIG1, rconfig1);
    ads_write_register(ADSREG_CONFIG2, rconfig2);
    ads_write_register(ADSREG_LOFF, rloff);
    ads_write_register(ADSREG_CH1SET, rchset);
    ads_write_register(ADSREG_CH2SET, rchset);

    pic18_spi_rate(SPI_FAST);
}
```

Código C.7 - ble112.h

```
#ifndef BLE112_H
#define BLE112_H

#include "cmd_def.h"
#include "mcu.h"
#include "main.h"

/*! BGAPI protocol message header size (permanent) */
#define BLE_HEADER_SIZE          4

/**  BLE112_advert BLE112 Advertisement Times */
/*!  List of different advertisement modes. */
typedef enum {
    FASTCONN = 0,   /*!< Fast-connection mode identifier */
    POWERSAVE = 1   /*!< Power-save mode identifier */
} adv_state_t;

#define ADVTIME_CHANGEMODE  30  /*!< Time in seconds to change mode. */
#define ADVTIME_FAST_MIN    20  /*!< Fast-connection mode - Lower limit */
#define ADVTIME_FAST_MAX    30  /*!< Fast-connection mode - Upper limit */
#define ADVTIME_PSAVE_MIN   1000    /*!< Power-save mode - Lower limit */
#define ADVTIME_PSAVE_MAX   2500    /*!< Power-save mode - Upper limit */

/*! BLE112 GATT 16-bit handles */
#define GH_FIRMWARE     0x0010  /*!< Firmware Revision (8 Bytes) */
#define GH_BATTERY      0x0014  /*!< Battery % level (1 Byte) */
#define GH_SAMPLES      0x0019  /*!< Multisample (20 Bytes) */
#define GH_LEADOFF      0x001D  /*!< \deprecated Lead-off flag (1 Byte) */
#define GH_CONFIG       0x0021  /*!< Configuration (1 Byte) */
#define GH_STREAM       0x0024  /*!< \deprecated Streaming switch (1 Byte)
*/

/*! Sends the message pointed by data1 and data2 to the BLE112 module via
 *  UART protocol. */
void ble_write_msg(uint8_t len1, uint8_t *data1, uint16_t len2, uint8_t
*data2);

/*! Reads the message sent from the BLE112 module and call the pertinent
 *  function handler using the BGlib API. */
void ble_read_msg(void);

/*! Spinlock-based busy-waiting routine for expected Event and Response
    Messages from the BLE112 module. */
void ble_wait_for_msg(void);

/*! Configuration function to call just after the module becomes active. */
void ble_sw_default_config(void);

/*! Send a Reset pulse through the RST line to the BLE112. */
void ble_hw_reset_pulse(void);

/*! This function keeps the Wake-up pin asserted so that the module
 *  is awake and able to receive any message using ble_write_msg.
 */
void ble_hw_system_wakeup(void);
```

```
/*! This function keeps the Wake-up pin de asserted so that the module can
 *  enter any configured Power Sleep mode and it shall not be able to
 *  receive any message using ble_write_msg. */
void ble_hw_system_sleepm(void);

/*! This function sets the Advertisement mode selected with the state
 *  parameter (Fast-connection or Power-save mode). */
void ble_advert_configuration(adv_state_t state);

/*! This function actually calls the ble_cmd_attributes_send one from BGlib
 *  to send the Multi-sample characteristic new value to the connected
 *  client. */
void ble_cmd_samples_send(uint8_t *bytearray);

/*! This function actually calls the ble_cmd_connection_update one from
 *  BGlib to set the connection parameters defined in the structure
 *  connect_param_t. */
void ble_cmd_config_connection_interval(void);

#endif  /* BLE112_H_ */
```

Código C.8 - ble112.c

```
#include "ble112.h"

extern WIERmonitor_t WIERmonitor;       /*!< FSM control variable */

const struct ble_msg *BTmsg;            /*!< BLE112 message */
struct ble_header BThdr;                /*!< Header of the message */
uint8_t data[256] = {0};                /*!< Payload of the message */

void ble_write_msg(uint8_t len1, uint8_t *data1,
                   uint16_t len2, uint8_t *data2)
{
    if (!WIERmonitor.status.STREAMING)
    {
        ble_hw_system_wakeup();         /* Keep BLE awake during TX */
        ble_wait_for_msg();
    }

    pic18_uart_putc(len1 + len2);       /* Send the packet length */
    pic18_uart_puts(data1, len1);       /* Send the header */
    pic18_uart_puts(data2, len2);       /* Send the payload */

    if (!WIERmonitor.status.STREAMING)
        ble_hw_system_sleepm();         /* Put BLE to bed */
}

void ble_read_msg(void)
{
    /* Read the BLE message header (4 bytes) */
    pic18_uart_gets((uint8_t*) & BThdr, BLE_HEADER_SIZE);

    /* Read the BLE message payload */
    pic18_uart_gets(data, BThdr.lolen);

    /* Find the appropiate message based on the header */
    BTmsg = ble_get_msg_hdr(BThdr);
```

```
    /* Call the appropiate handler */
    if (BTmsg)
        BTmsg->handler(data);
}

void ble_hw_reset_pulse(void)
{
    LATC |= BLE_RST;
    __delay_ms(20);
    LATC &= ~BLE_RST;
    __delay_ms(20);
    LATC |= BLE_RST;

    /* Wait for BLE112 System boot EVENT or Reset RESPONSE */
    ble_wait_for_msg();
}

void ble_sw_default_config(void)
{
    WIERmonitor.status.BLECLLBCK = 1;
    WIERmonitor.cparam.timeout = 100;
    WIERmonitor.cparam.latency = 100;
    WIERmonitor.cparam.interval_min = 6;   /* 7.5 ms (Android min value) */
    WIERmonitor.cparam.interval_max = 6;   /* 7.5 ms (Android min value) */

    /* Firmware Revision String Attribute writing */
    ble_cmd_attributes_write(GH_FIRMWARE, 0,
                             FSTR_SIZE, (uint8*) FIRMWARE_REV);
    ble_wait_for_msg();

    ble_advert_configuration(FASTCONN);
}

void ble_advert_configuration(adv_state_t state)
{
    switch (state)
    {
    default:
    case FASTCONN:
    {
        /* Advertisement interval - All advertisement channels */
        /* Min/Max parameters are in units of 625 us */
        ble_cmd_gap_set_adv_parameters(ADVTIME_FAST_MIN * 8 / 5,
                                       ADVTIME_FAST_MAX * 8 / 5, 7);
        ble_wait_for_msg();
    }
    break;

    case POWERSAVE:
    {
        /* Advertisement interval (Reduced power) */
        ble_cmd_gap_set_adv_parameters(ADVTIME_PSAVE_MIN * 8 / 5,
                                       ADVTIME_PSAVE_MAX * 8 / 5, 7);
        ble_wait_for_msg();
    }
    break;
    }
```

```
    /* Start Advertising (General Discoverable & Undirected Connectable) */
    ble_cmd_gap_set_mode(GAP_AD_FLAG_GENERAL_DISCOVERABLE,
                         GAP_SCAN_HEADER_ADV_NONCONN_IND);
    ble_wait_for_msg();

    pic18_led_turn_on(LED_BLU);
    WIERmonitor.timer = 0;
}

void ble_cmd_samples_send(uint8_t *bytearray)
{
    WIERmonitor.status.BUSYBLE = 1;     /* BLE112 is busy until response */

    ble_cmd_attributes_send(WIERmonitor.cparam.handle, GH_SAMPLES,
                            BLE_PAYLOAD_SIZE, bytearray);
}

void ble_cmd_config_connection_interval(void)
{
    ble_cmd_connection_update(WIERmonitor.cparam.handle,
                              WIERmonitor.cparam.interval_min,
                              WIERmonitor.cparam.interval_max,
                              WIERmonitor.cparam.latency,
                              WIERmonitor.cparam.timeout);
}

void ble_wait_for_msg(void)
{
    WIERmonitor.status.BLECLLBCK = 0;
    while (!WIERmonitor.status.BLECLLBCK);
}

void ble_hw_system_wakeup(void)
{
    LATC |= BLE_WUP;
    __delay_us(4);
}

void ble_hw_system_sleepm(void)
{
    while (!TXSTAbits.TRMT);
    LATC &= ~BLE_WUP;
}

/* Used BGlib function handlers */
/*! EVENT: Device booted up, and is ready to receive commands */
void ble_evt_system_boot(const struct ble_msg_system_boot_evt_t *msg)
{
    WIERmonitor.status.BLECLLBCK = 1;       /* Set message as processed */
}

/*! EVENT: I/O-port state changed (Used when Wake-up pin status changes) */
void ble_evt_hardware_io_port_status(const struct
                         ble_msg_hardware_io_port_status_evt_t *msg)
{
    /* If Wakeup Pin (0,0) has changed its value */
    if (msg->port == 0)
        WIERmonitor.status.BLECLLBCK = 1;   /* Set message as processed */
}
```

```
/*! EVENT: Link Disconnected */
void ble_evt_connection_disconnected(const struct
                          ble_msg_connection_disconnected_evt_t *msg)
{
    /* Client has DISCONNECTED */
    WIERmonitor.status.CONNECTED = 0;
    WIERmonitor.state = DISCONNECTION;
}

/*! EVENT: Connection opened */
void ble_evt_connection_status(const struct
                          ble_msg_connection_status_evt_t *msg)
{
    if (msg->flags == (connection_connected | connection_completed))
    {
        WIERmonitor.cparam.handle = msg->connection;
        WIERmonitor.status.CONNECTED = 1;
    }
}

/*! EVENT: Remote version information */
void ble_evt_connection_version_ind(const struct
ble_msg_connection_version_ind_evt_t *msg)
{
    /* DO NOTHING */
}

/*! EVENT: Remote feature information */
void ble_evt_connection_feature_ind(const struct
                           ble_msg_connection_feature_ind_evt_t *msg)
{
    if (WIERmonitor.status.CONNECTED)
        WIERmonitor.state = CONNECTION;
}

/*! EVENT: Attribute value has changed */
void ble_evt_attributes_value(const struct
                           ble_msg_attributes_value_evt_t *msg)
{
    if (msg->handle == GH_CONFIG)
    {
        WIERmonitor.config.byte = msg->value.data[0];
        WIERmonitor.state = CONFIGURATION;
    }
}

/*! RESPONSE: Update connection parameters */
void ble_rsp_connection_update(const struct
                             ble_msg_connection_update_rsp_t *msg)
{
    WIERmonitor.status.BLECLLBCK = 1;       /* Set message as processed */
}

/*! RESPONSE: Set advertising parameters */
void ble_rsp_gap_set_adv_parameters(const struct
                             ble_msg_gap_set_adv_parameters_rsp_t *msg)
{
    WIERmonitor.status.BLECLLBCK = 1;       /* Set message as processed */
}
```

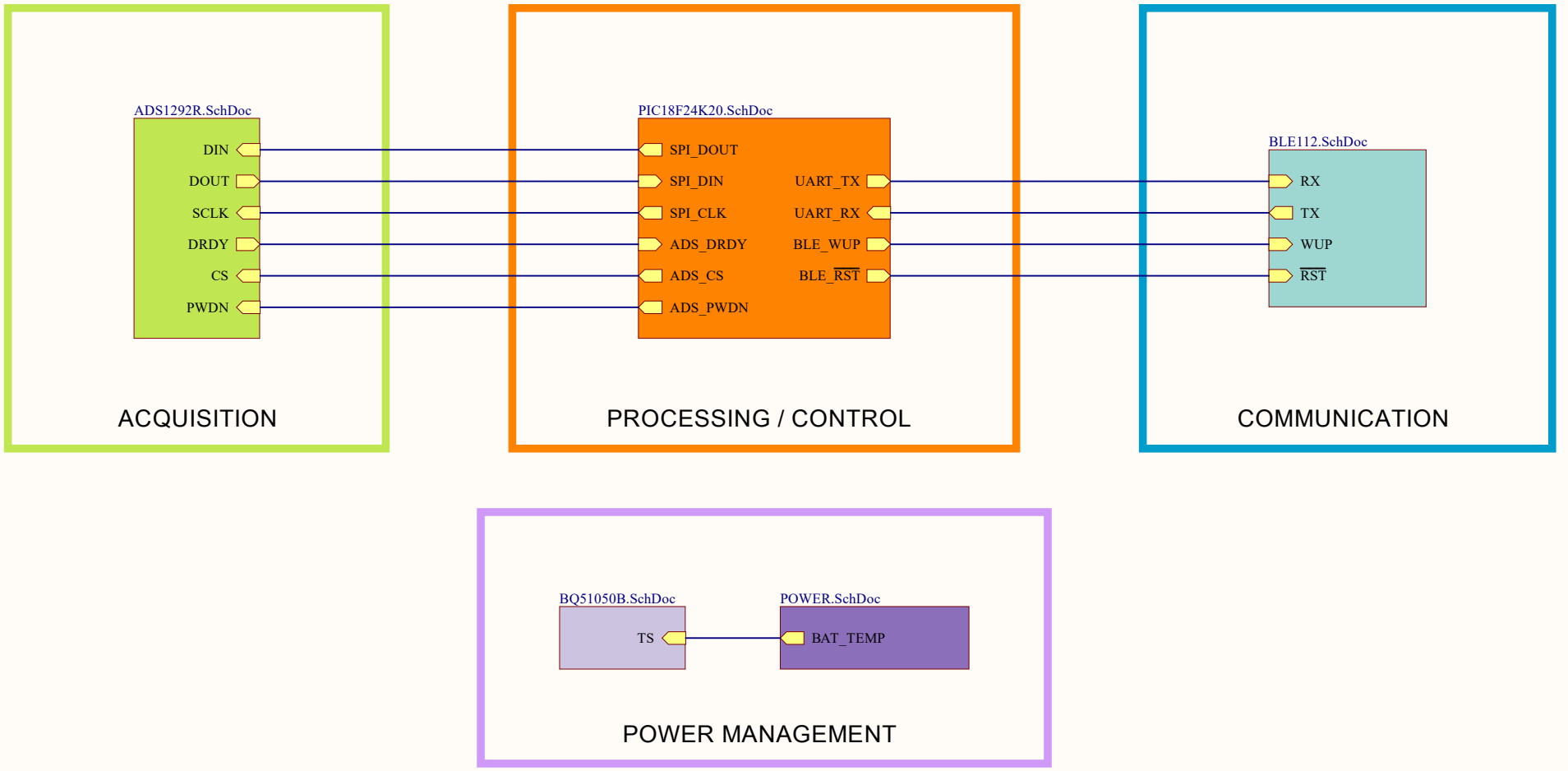
```
/*! RESPONSE: Set discoverable and connectable mode */
void ble_rsp_gap_set_mode(const struct ble_msg_gap_set_mode_rsp_t *msg)
{
    WIERmonitor.status.BLECLLBCK = 1;       /* Set message as processed */
}

/*! RESPONSE: Write attribute to database */
void ble_rsp_attributes_write(const struct
                              ble_msg_attributes_write_rsp_t *msg)
{
    WIERmonitor.status.BLECLLBCK = 1;       /* Set message as processed */
}

/*! RESPONSE: Send notification or indication to remote device. */
void ble_rsp_attributes_send(const struct
                              ble_msg_attributes_send_rsp_t *msg)
{
    WIERmonitor.status.BUSYBLE = 0;         /* BLE112 is free to listen */
}
```

Anexo D – Esquemáticos del circuito

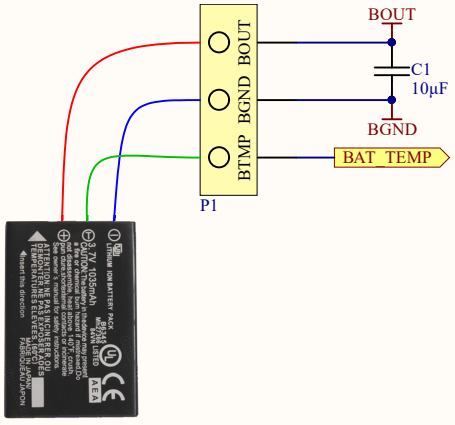
En las siguientes páginas se pueden encontrar los esquemáticos:

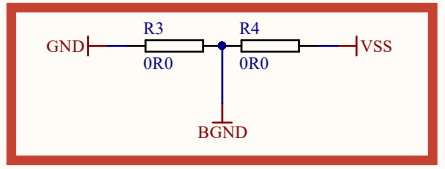
- Esquema general del sistema
- Bloques de alimentación y carga inalámbrica
- Bloque de adquisición
- Bloque de procesamiento
- Bloque de transmisión

Además de un documento con las capas reveladas en el circuito impreso del segundo prototipo.



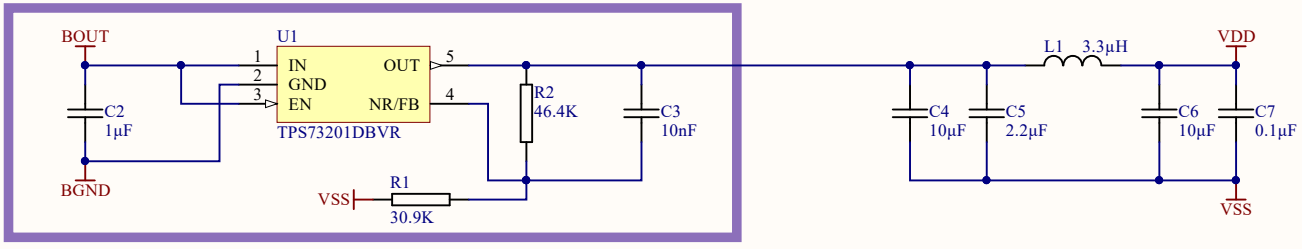
Title	**System Main Scheme**	
Size: A4	Number	Revision
Date: 15/09/2015		Sheet 1 of 6
File: D:\Dropbox\..\MAIN.SchDoc		Drawn By: Antonio López Marín





Zero-ohm links used to connect the three GROUND PLANES

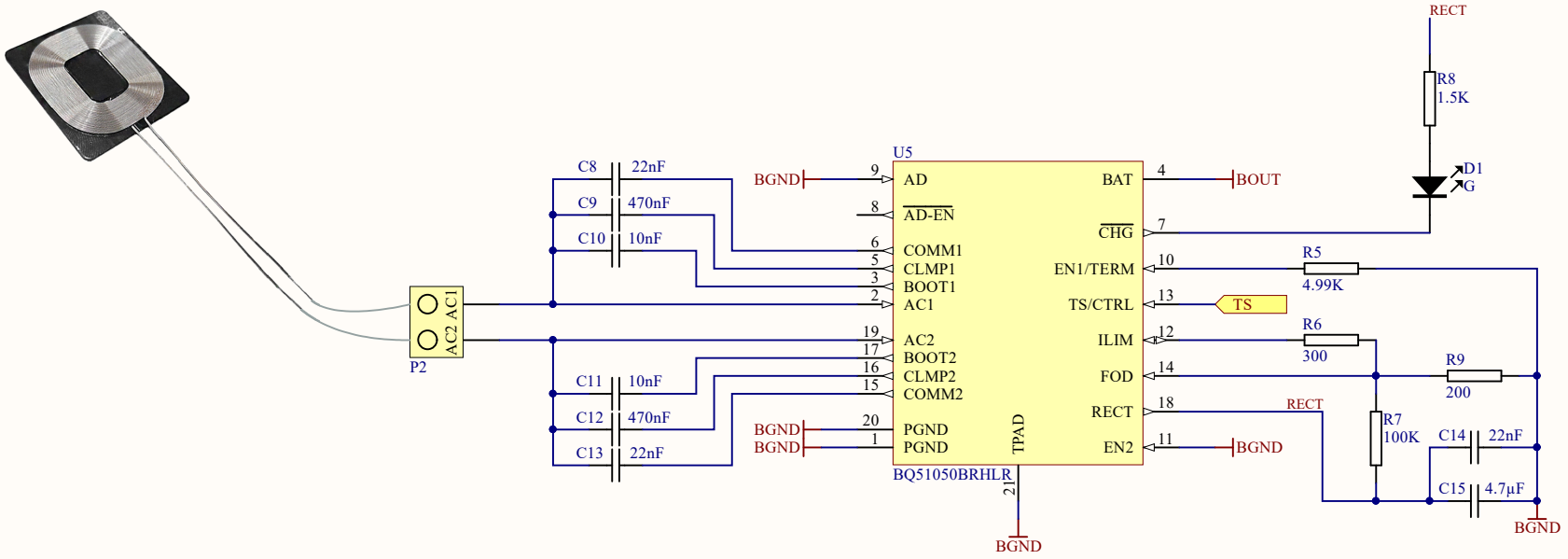
BATTERY ANALOG DIGITAL



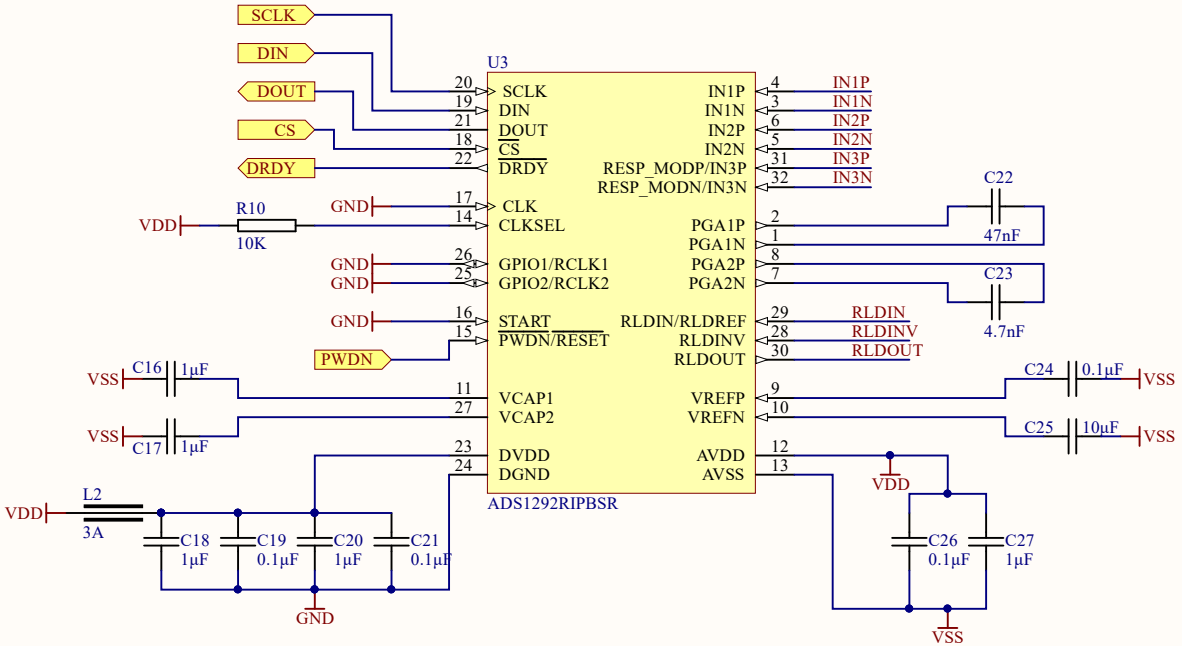
Step-down regulator from 4.2V (BAT) to 3V using a LDO one for ANALOG & DIGITAL PART

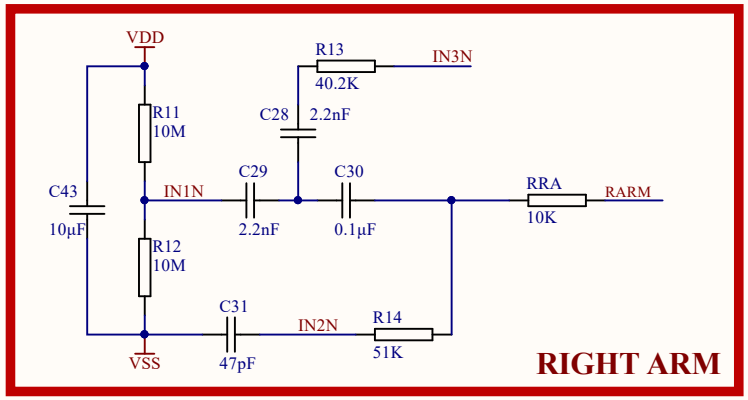
ADS1292R PIC18F45K20 BLE113

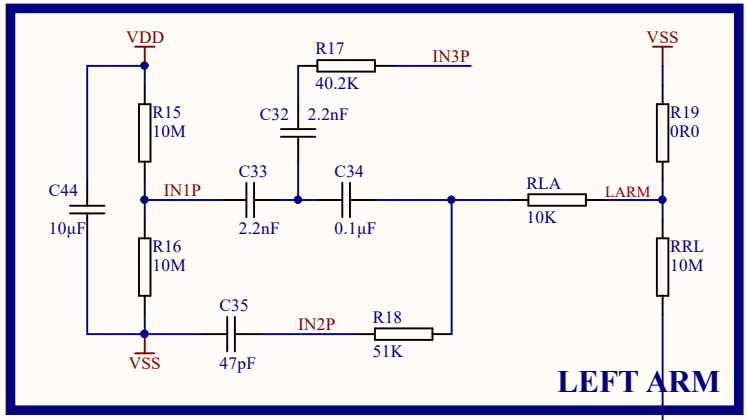
Title	TPS73201 - Low-noise LDO voltage regulator	
Size: A4	Number:	Revision:
Date: 15/09/2015		Sheet 2 of 6
File: D:\Dropbox\..\POWER.SchDoc		Drawn By: Antonio López Marín

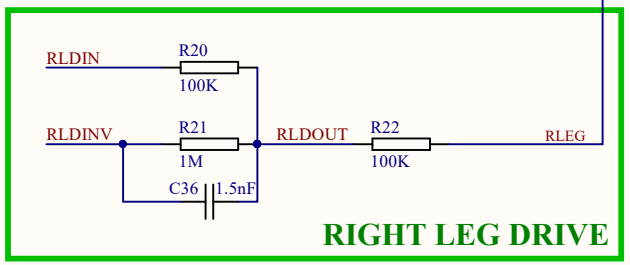


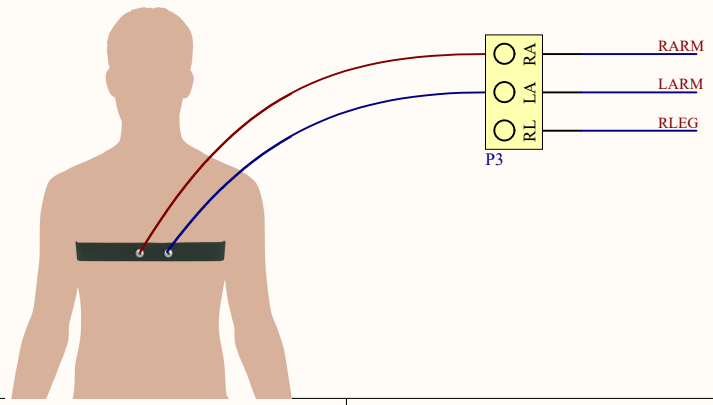
Title	**BQ51050B - Wireless Li-ion battery charger**	
Size: A4	Number	Revision
Date: 15/09/2015		Sheet 3 of 6
File: D:\Dropbox\..\BQ51050B.SchDoc		Drawn By: Antonio López Marín



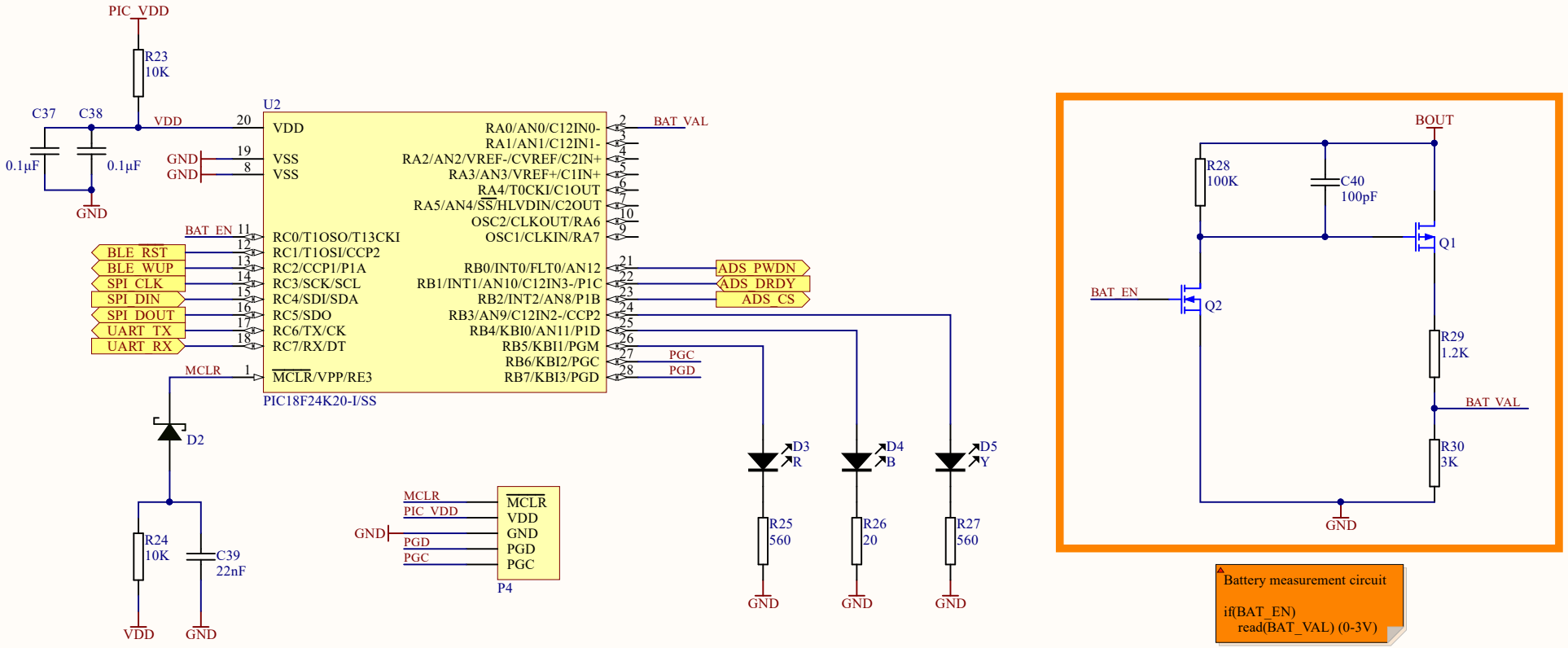




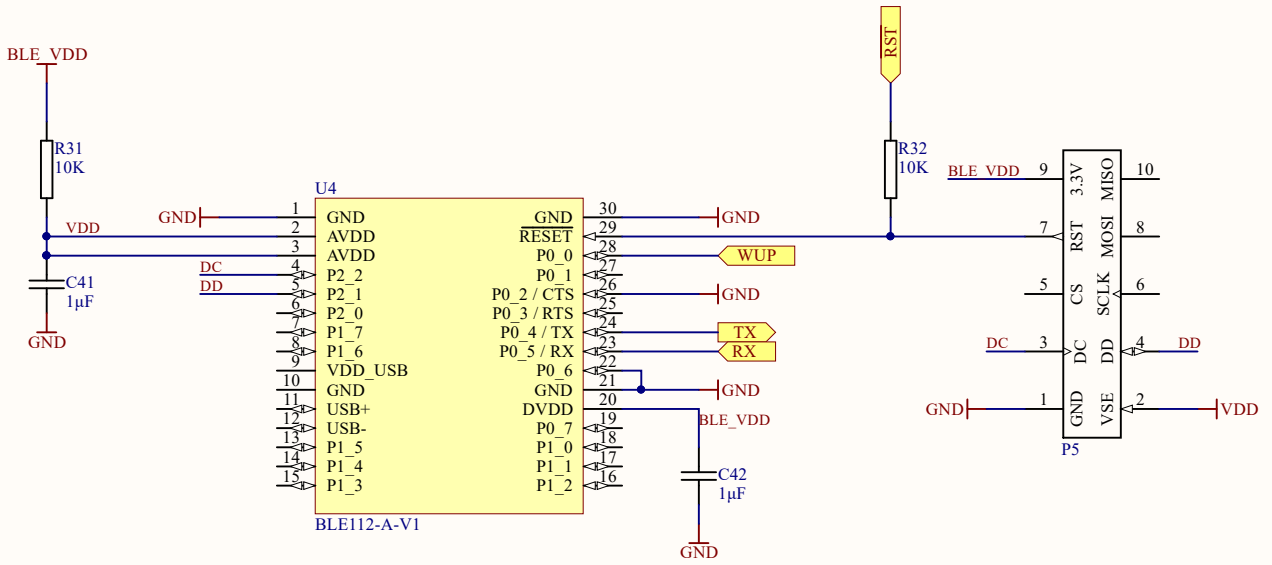




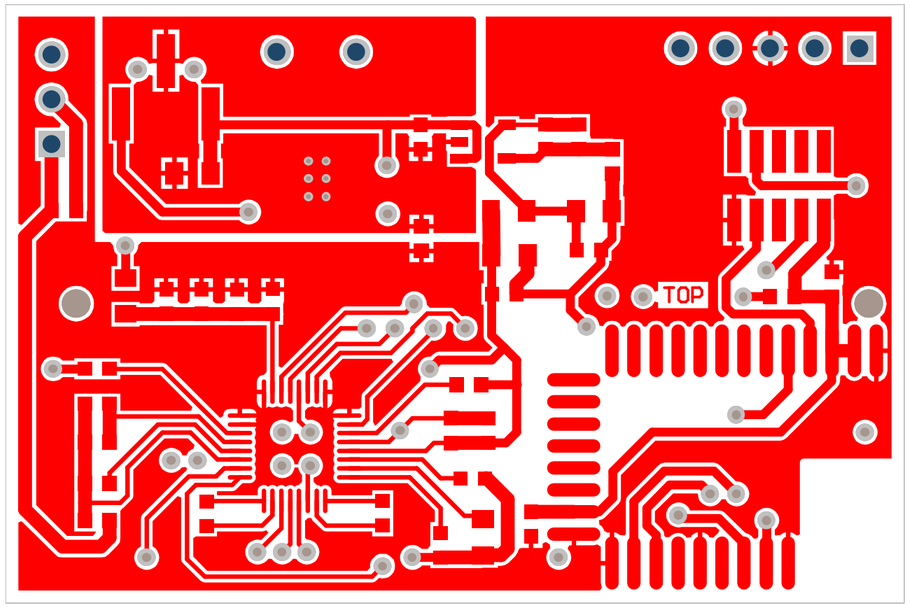
Title	ADS1292R - 24bit ADC w/ EKG & RES front-end	
Size: A4	Number	Revision
Date: 15/09/2015		Sheet 4 of 6
File: D:\Dropbox\..\ADS1292R.SchDoc		Drawn By: Antonio López Marín

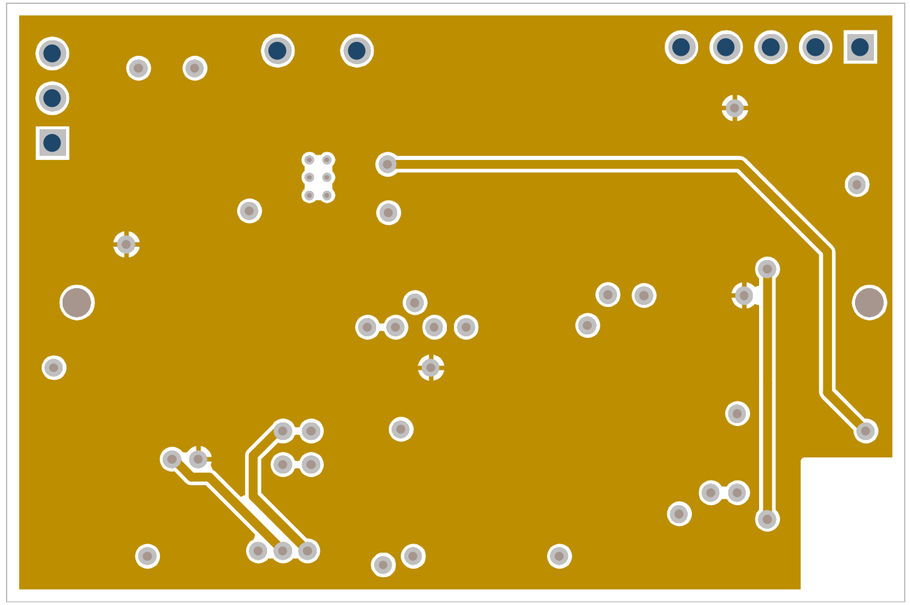


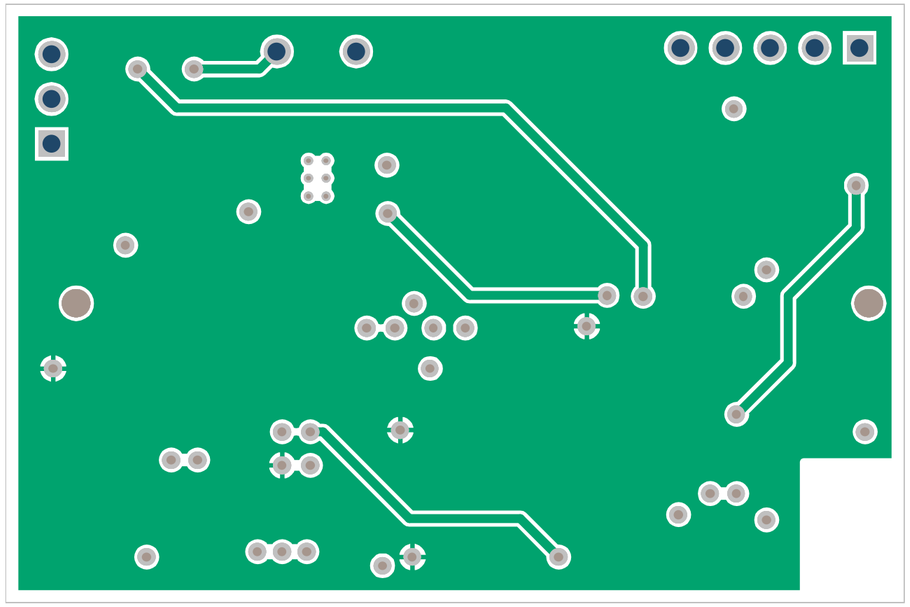
Title	**PIC 18F24K20 - Low-power MCU w/ internal oscillator**	
Size: A4	Number:	Revision:
Date: 15/09/2015		Sheet 5 of 6
File: D:\Dropbox\...\PIC18F24K20.SchDoc		Drawn By: Antonio López Marín

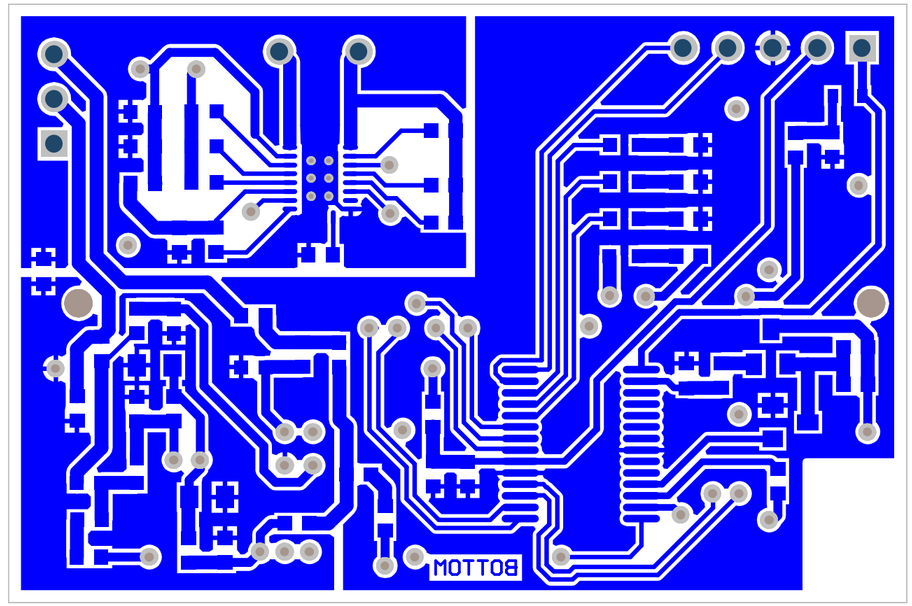


Title	BLE112 - Bluetooth Low Energy TX/RX module	
Size: A4	Number:	Revision:
Date: 15/09/2015		Sheet 6 of 6
File: D:\Dropbox\..\BLE112.SchDoc		Drawn By: Antonio López Marín









Layer	Name	Material	Thickness	Constant	Board Layer Stack
1	Top Overlay				
2	Top Solder	Solder Resist	0.010mm	3.5	
3	Top Layer	Copper	0.035mm		
4	Prepreg 7628	FR-4	0.125mm	4.2	
5	Power	Copper	0.035mm		
6	Core FR-4	FR-4	1.200mm	4.2	
7	Ground	Copper	0.035mm		
8	Prepreg 7628	FR-4	0.125mm	4.2	
9	Bottom Layer	Copper	0.035mm		
10	Bottom Solder	Solder Resist	0.010mm	3.5	
11	Bottom Overlay				

Altium Limited
12a Rodborough Rd
Frenchs Forest
NSW 2086

ENGINEER: ANTONIO LOPEZ MARIN	TITLE: WiER Monitor layout Signal Layers (Top, Power, Ground, Bottom)	
PCB DESIGNER: ANTONIO LOPEZ MARIN		
DATE: 15/09/2015	PART NO.:	REV:
FILE NAME: WiER_PCB_34x51S.PcbDoc	DWG NO:	SCALE: 2:1

Anexo E – Lista de materiales

Desglose para un dispositivo

Componente	Nombre	Cantidad	Min. compra	Precio (unidad)	Precio (dispositivo)	Total pagado
CAP CER 10UF 25V 10% X5R 0805	C1, C4, C6, C25, C43, C44	6	50	0.08 €	0.50 €	4.20 €
CAP CER 1µF 50V 10% X5R 0603	C2, C16, C17, C18, C20, C27, C41, C42	8	50	0.04 €	0.29 €	1.80 €
CAP CER 10000pF 50V 10% X5R 0603	C3, C10, C11	3	50	0.04 €	0.13 €	2.10 €
CAP CER 2.2µF 10V 10% X5R 0603	C5	1	50	0.03 €	0.03 €	1.70 €
CAP CER 0.1µF 50V 10% X7R 0603	C7, C19, C21, C24, C26, C30, C34, C37, C38	9	100	0.03 €	0.26 €	2.90 €
CAP CER 2200pF 50V 10% X7R 0603	C8, C28, C29, C32, C33	5	25	0.02 €	0.09 €	0.45 €
CAP CER 0.47µF 50V 10% X5R 0603	C9, C12	2	50	0.04 €	0.07 €	1.80 €
CAP CER 0.022µF 50V 10% X7R 0603	C13, C14, C39	3	200	0.01 €	0.02 €	1.40 €
CAP CER 4.7UF 25V 10% X5R 0603	C15	1	25	0.17 €	0.17 €	4.33 €
CAP CER 0.047µF 50V 5% X7R 0603	C22	1	200	0.03 €	0.03 €	5.00 €
CAP CER 4700PF 50V 5% X7R 0603	C23	1	50	0.08 €	0.08 €	4.10 €
CAP CER 47pF 50V 5% NP0 0603	C31, C35	2	25	0.02 €	0.04 €	0.45 €
CAP CER 1500pF 100V 10% X7R 0603	C36	1	100	0.03 €	0.03 €	3.30 €
CAP CER 100pF 50V 5% NP0 0603	C40	1	25	0.02 €	0.02 €	0.45 €
LED 0603 PURE GREEN 15MCD	D1	1	50	0.16 €	0.16 €	7.85 €
DIODE SCHOTTKY 30V 200MA SOD523	D2	1	100	0.06 €	0.06 €	5.80 €
LED 630NM RED WTR CLR 0603 SMD	D3	1	10	0.26 €	0.26 €	2.59 €
LED 470NM BLUE CLEAR 0603 SMD	D4	1	5	0.39 €	0.39 €	1.96 €
LED 590NM YLW WTR CLR 0603 SMD	D5	1	10	0.26 €	0.26 €	2.59 €
FIXED IND 3.3µH 450MA 340 MOHM	L1	1	10	0.24 €	0.24 €	2.42 €
FERRITE CHIP 120 OHM 0805	L2	1	25	0.02 €	0.02 €	0.45 €
RECEIVER 1 COIL 1 LAYER (2THVIAS)	P2	1	10	9.20 €	9.20 €	92.00 €
Minitek 127™ 20021121	P5	1	5	0.66 €	0.66 €	3.30 €
MOSFET P-CH 20V 400MA SOT-23	Q1	1	50	0.27 €	0.27 €	13.45 €
MOSFET N-CH 30V TO-236AB	Q2	1	150	0.04 €	0.04 €	5.70 €
RES SMD 46.4K OHM 1% 1/10W 0603	R1	1	5	0.24 €	0.24 €	1.22 €
RES SMD 30.9K OHM 1% 1/5W 0603	R2	1	5	0.34 €	0.34 €	1.72 €
RES SMD 0 OHM 1% 1/10W 0603	R3, R4, R19	3	50	0.01 €	0.03 €	0.45 €
RES SMD 4.99K OHM 1% 1/10W 0603	R5	1	50	0.06 €	0.06 €	3.20 €
RES SMD 300 OHM 1% 1/10W 0603	R6	1	50	0.01 €	0.01 €	0.65 €

RES SMD 100K OHM 1% 1/5W 0603	R7, R20, R22, R28	4	50	0.01 €	0.05 €	0.65 €
RES SMD 1.5K OHM 1% 1/5W 0603	R8	1	100	0.64 €	0.64 €	64.00 €
RES SMD 200 OHM 1% 1/16W 0603	R9	1	100	0.02 €	0.02 €	1.80 €
RES SMD 10K OHM 5% 1/5W 0603	R10, R23, R24, R31, R32, RLA, RRA	7	100	0.51 €	3.57 €	51.00 €
RES SMD 10M OHM 1% 1/10W 0603	R11, R12, R15, R16, RRL	5	50	0.02 €	0.10 €	1.00 €
RES SMD 40.2K OHM 0.1% 1/6W 0603	R13, R17	2	10	0.32 €	0.64 €	3.22 €
RES SMD 51K OHM 1% 1/10W 0603	R14, R18	2	10	0.10 €	0.21 €	1.04 €
RES SMD 1M OHM 1% 1/5W 0603	R21	1	100	0.64 €	0.64 €	64.00 €
RES SMD 560 OHM 1% 1/5W 0603	R25, R27	2	100	0.51 €	1.02 €	51.00 €
RES SMD 20 OHM 1% 1/8W 0603	R26	1	50	0.02 €	0.02 €	0.95 €
RES SMD 1.2K OHM 1% 1/5W 0603	R29	1	50	0.01 €	0.01 €	0.70 €
RES SMD 3K OHM 1% 1/4W 0603	R30	1	5	0.26 €	0.26 €	1.31 €
IC REG LDO ADJ 0.25A SOT23-5	U1	1	5	1.50 €	1.50 €	7.48 €
PIC18F24K20-I/SS	U2	1	5	1.95 €	1.95 €	9.75 €
ADS1292RIPBSR	U3	1	4	9.34 €	9.34 €	37.36 €
BQ51050BRHLR	U5	1	4	4.91 €	4.91 €	19.62 €
BLE112-A-V1	U4	1	4	11.56 €	11.56 €	46.24 €
Energizer OL-40B	XX	1	1	18.35 €	18.35 €	18.35 €
OKW Minitec B9004858	XX	1	4	37.04 €	37.04 €	148.16 €
Fabricación de las placas	XX	1	6	44.67 €	44.67 €	268.00 €
Total					**150.50 €**	**974.96 €**

Desglose para fabricación a gran escala

Componente	Nombre	Cantidad (x100000)	Precio (unidad) (x100000)	Precio (dispositivo) (x100000)
CAP CER 10UF 25V 10% X5R 0805	C1, C4, C6, C25, C43, C44	600000	0.034 €	20,334.00 €
CAP CER 1µF 50V 10% X5R 0603	C2, C16, C17, C18, C20, C27, C41, C42	800000	0.020 €	15,872.00 €
CAP CER 10000pF 50V 10% X5R 0603	C3, C10, C11	300000	0.024 €	7,200.00 €
CAP CER 2.2µF 10V 10% X5R 0603	C5	100000	0.003 €	269.00 €
CAP CER 0.1µF 50V 10% X7R 0603	C7, C19, C21, C24, C26, C30, C34, C37, C38	900000	0.010 €	9,009.00 €
CAP CER 2200pF 50V 10% X7R 0603	C8, C28, C29, C32, C33	500000	0.003 €	1,615.00 €
CAP CER 0.47µF 50V 10% X5R 0603	C9, C12	200000	0.020 €	3,968.00 €
CAP CER 0.022µF 50V 10% X7R 0603	C13, C14, C39	300000	0.007 €	2,061.00 €
CAP CER 4.7UF 25V 10% X5R 0603	C15	100000	0.057 €	5,718.00 €
CAP CER 0.047µF 50V 5% X7R 0603	C22	100000	0.009 €	929.00 €
CAP CER 4700PF 50V 5% X7R 0603	C23	100000	0.042 €	4,219.00 €
CAP CER 47pF 50V 5% NP0 0603	C31, C35	200000	0.004 €	754.00 €
CAP CER 1500pF 100V 10% X7R 0603	C36	100000	0.004 €	372.00 €
CAP CER 100pF 50V 5% NP0 0603	C40	100000	0.003 €	323.00 €
LED 0603 PURE GREEN 15MCD	D1	100000	0.105 €	10,456.00 €
DIODE SCHOTTKY 30V 200MA SOD523	D2	100000	0.021 €	2,085.00 €
LED 630NM RED WTR CLR 0603 SMD	D3	100000	0.056 €	5,555.00 €
LED 470NM BLUE CLEAR 0603 SMD	D4	100000	0.092 €	9,170.00 €
LED 590NM YLW WTR CLR 0603 SMD	D5	100000	0.056 €	5,555.00 €
FIXED IND 3.3µH 450MA 340 MOHM	L1	100000	0.127 €	12,724.00 €
FERRITE CHIP 120 OHM 0805	L2	100000	0.018 €	1,800.00 €
RECEIVER 1 COIL 1 LAYER (2THVIAS)	P2	100000	5.732 €	573,217.00 €
Minitek 127™ 20021121	P5	0	0.000 €	0.00 €
MOSFET P-CH 20V 400MA SOT-23	Q1	100000	0.067 €	6,737.00 €
MOSFET N-CH 30V TO-236AB	Q2	100000	0.034 €	3,380.00 €
RES SMD 46.4K OHM 1% 1/10W 0603	R1	100000	0.036 €	3,638.00 €
RES SMD 30.9K OHM 1% 1/5W 0603	R2	100000	0.036 €	3,638.00 €
RES SMD 0 OHM 1% 1/10W 0603	R3, R4, R19	300000	0.001 €	249.00 €
RES SMD 4.99K OHM 1% 1/10W 0603	R5	100000	0.003 €	309.00 €
RES SMD 300 OHM 1% 1/10W 0603	R6	100000	0.006 €	596.00 €
RES SMD 100K OHM 1% 1/5W 0603	R7, R20, R22, R28	400000	0.006 €	2,384.00 €
RES SMD 1.5K OHM 1% 1/5W 0603	R8	100000	0.004 €	445.00 €
RES SMD 200 OHM 1% 1/16W 0603	R9	100000	0.009 €	893.00 €
RES SMD 10K OHM 5% 1/5W 0603	R10, R23, R24, R31, R32, RLA, RRA	700000	0.005 €	3,430.00 €
RES SMD 10M OHM 1% 1/10W 0603	R11, R12, R15, R16, RRL	500000	0.003 €	1,545.00 €
RES SMD 40.2K OHM 0.1% 1/6W 0603	R13, R17	200000	0.036 €	7,276.00 €
RES SMD 51K OHM 1% 1/10W 0603	R14, R18	200000	0.001 €	230.00 €
RES SMD 1M OHM 1% 1/5W 0603	R21	100000	0.007 €	685.00 €

RES SMD 560 OHM 1% 1/5W 0603	R25, R27	200000	0.005 €	980.00 €
RES SMD 20 OHM 1% 1/8W 0603	R26	100000	0.003 €	309.00 €
RES SMD 1.2K OHM 1% 1/5W 0603	R29	100000	0.006 €	596.00 €
RES SMD 3K OHM 1% 1/4W 0603	R30	100000	0.036 €	3,638.00 €
IC REG LDO ADJ 0.25A SOT23-5	U1	100000	0.571 €	57,114.00 €
PIC18F24K20-I/SS	U2	100000	1.460 €	145,970.00 €
ADS1292RIPBSR	U3	100000	5.345 €	534,527.00 €
BQ51050BRHLR	U5	100000	2.414 €	241,364.00 €
BLE112-A-V1	U4	100000	8.336 €	833,617.00 €
Energizer OL-40B	XX	100000	16.720 €	1,672,000.00 €
OKW Minitec B9004858	XX	100000	7.060 €	706,000.00 €
Fabricación de las placas	XX	100000	0.246 €	24,580.69 €
Total				**49.49 €**

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

REFERENCIAS

[1] K. Ioannou, M. Ignaszewski y I. Macdonald, «Ambulatory electrocardiography: The contribution of Norman Jefferis Holter», *British Columbia Medical Journal,* vol. 56, pp. 86-89, 2014.

[2] S. Chatterjee y A. Miller, Biomedical Instrumentation Systems, 1ª ed., Delmar, 2010.

[3] J. D. Bronzino, Medical Devices and Systems – The Biomedical Engineering Handbook, 3ª ed., Taylor & Francis, 2006.

[4] T. Kanti Bera, «Bioelectrical Impedance Methods for Noninvasive Health Monitoring: A Review», *Journal of Medical Engineering,* vol. 2014, nº 381251, 2014.

[5] V. Clearbridge, *CardioLeaf ULTRA,* http://www.clearbridgevitalsigns.com/

[6] Zephyr, *BioPatch,* http://www.zephyr-technology.nl/en/

[7] Biodevices, *VitalJacket,* http://www.vitaljacket.com/

[8] F. Touati, R. Tabish y A. B. Mnaouer, «A Real-time BLE Enabled ECG System for Remote Monitoring» de *The 3rd International Conference on Biomedical Engineering and Technology*, Beijing, China, 2013.

[9] R. Macías, M. A. García, J. Ramos, R. Bragós y M. Fernández, «Contactless electrical bioimpedance system for monitoring ventilation. A biodevice for vehicle environment», de *Biodevices 2013*, Barcelona, Spain, 2013.

[10] Texas Instruments, *ADS1292R, ADS1292ECG-FE, MSP430, TPS73201, TIDA-00329, BQ51050B, BQ51050BEVM-764,* http://www.ti.com/

[11] Analog Devices, *ADAS1000,* http://www.analog.com/

[12] Metron, *PS-420 (Fluke Biomedical),* http://www.flukebiomedical.com/

[13] International Electrotechnical Commission, IEC 60601-2-25, 2ª ed., 2011.

[14] Microchip, *PIC18F, PICkit, MPLAB, XC8, MCP1702,* http://www.microchip.com/

[15] Bluetooth SIG, «Bluetooth Low Energy (BLE) Specification»,
http://www.bluetooth.com/Pages/low-energy.aspx

[16] Bluegiga (Silicon Labs), *BLE112-A, BLE113-A, DKBLE113, bleGUI, BLE Software and SDK,* http://www.bluegiga.com/home

[17] Digi, *Xbee ZigBee,* http://www.digi.com/

[18] Bluegiga (Silicon Labs), *Bluegiga Knowledgebase,* https://bluegiga.zendesk.com/

[19] intersil, «AN9657.1 - Data Conversion Binary Code Formats», http://www.intersil.com/content/dam/Intersil/documents/an96/an9657.pdf, 1997.

[20] Nordic Semiconductor, *nRF Master Control Panel,* https://www.nordicsemi.com/

[21] Altium, *Altium Designer 15, Altium Tech Docs,* http://www.altium.com/

[22] SparkFun Electronics, «AN - Design Considerations for Mixed-Signal Design», https://www.sparkfun.com/, 2009.

[23] *FreeCAD,* http://www.freecadweb.org/

[24] *Inkscape,* https://inkscape.org

[25] *3D Content Central,* http://www.3dcontentcentral.com/

[26] Prema, *Prema 5017,* http://www.prema.com/