ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

Curso Académico 2009/2010

Proyecto de Fin de Carrera

Sistema inalámbrico y multicanal para monitorización de señales biológicas en tiempo real

Autor: Javier Calle Plaza

Tutores: Jose Mª Cañas Plaza,
Roberto de la Prieta

"Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado"

Albert Szent-Györgyi

A Roberto de la Prieta, por poner ese interés en todos los aspectos del proyecto, por ayudarme a perfeccionar multitud de detalles que ahora considero fundamentales y por permitirme disfrutar de los debates tan interesantes que hemos tenido.

A Susana Borromeo, por sus valoradas correcciones y por proporcionarme los medios adecuados en los momentos oportunos.

A Javier López, por ser un gran compañero de laboratorio y tener ese arrojo con la fresadora.

A mi familia, no hay suficientes árboles en el mundo para poder proporcionar las hojas necesarias en las cuales escribir todo lo que les tengo que agradecer.

A mi niña, por ser la luz que me ilumina en los momentos de oscuridad.

Resumen del proyecto

Desde hace algunos años, la Telemedicina está siendo incorporada de manera progresiva en el ámbito de la medicina, tanto en aplicaciones clínicas como en investigación. El desarrollo de las tecnologías de la información y de las comunicaciones posibilita hoy en día que la atención sanitaria se pueda beneficiar de sistemas que permiten realizar consultas interactivas a distancia, monitorización remota o telediagnosis, entre otros.

En el presente proyecto se ha diseñado e implementado un sistema para la adquisición de varias señales biológicas y su envío de forma inalámbrica, así como para su recepción, visualización y almacenamiento.

El sistema consta de un módulo hardware y de una aplicación software. El dispositivo hardware permite la captura de varias señales y su transmisión simultánea de forma inalámbrica. La aplicación software se encarga de recibir los datos proporcionados por el módulo hardware, representarlos en pantalla y almacenarlos, bien en un fichero, o bien en una base de datos embebida en la propia aplicación.

Por otro lado, la aplicación software es capaz de actuar como servidor de señales, retransmitiendo a través de la red los datos que recibe del hardware. Además, el mismo software puede actuar como cliente, conectándose a otras instancias de la aplicación que estén configuradas como servidores de señal.

El sistema hardware ha sido diseñado para que sea portable y de bajo coste. Asimismo se ha conseguido dotar al dispositivo de una gran flexibilidad en cuanto al número de señales que puede manejar simultáneamente y a la frecuencia de las mismas. Este mismo criterio de flexibilidad se ha tenido en cuenta a la hora de abordar el diseño relativo a la parte de comunicaciones inalámbricas. Aunque el módulo de comunicación del actual prototipo utiliza Bluetooth, el mismo diseño hardware permite su sustitución por un módulo Wi-Fi.

El sistema propuesto se podría emplear para recibir una serie de señales biológicas procedentes de un sujeto y enviarlas por medio de un vinculo inalámbrico a un PC, en el cual se visualizarían. Adicionalmente, y dado que cualquier instancia de la aplicación puede funcionar indistintamente como cliente o servidor, las mismas señales podrían monitorizarse remotamente en otros PCs.

Índice de contenidos

Índice de figuras

Índice de tablas

1.- Introducción

1.1.- Motivación

En España hay multitud de pueblos y aldeas que, debido a su baja densidad de población o a una situación geográfica difícilmente accesible, no disponen de servicios médicos adecuados en las proximidades y los pacientes se ven obligados a acudir a los centros hospitalarios situados en el núcleo urbano más cercano, que en ocasiones está a varias decenas de kilómetros de distancia. Esto implica que el paciente debe disponer de un medio de transporte y de la salud adecuada para realizar el viaje, condiciones que muchas veces no se cumplen, surgiendo la necesidad de llevar la asistencia sanitaria a lugares que, por motivos de ordenación del territorio, no se dispone de suficiente cercanía a un centro hospitalario.

Otra de las necesidades que se pretende cubrir con este proyecto es el diseño de dispositivos portables y de bajo coste que permitan realizar lecturas de constantes vitales de manera fiable. Esto se puede cumplir en el caso en el que el hardware necesario no sea excesivamente complejo, como ocurre en el caso del diagnóstico por imagen (TAC, Resonancia Magnética, etc ...). En el caso de medición de señales tales como el EMG, el ECG, el EEG, la Oximetría de pulso o la temperatura, se puede afrontar la realización de dispositivos capaces de cumplir los requisitos anteriores.

Por otro lado, en algunas ocasiones, es necesario mejorar la comodidad del paciente cuando se realizan determinadas pruebas de monitorización de sus constantes biológicas, lo que puede conseguirse aumentando la portabilidad del sistema de medición, reduciendo sus dimensiones y sustituyendo, en la medida de lo posible, los cables por comunicaciones inalámbricas. Un ejemplo claro que ilustra esta mejora de la comodidad es el de una prueba de esfuerzo cardíaco, en la cual, el sujeto bajo estudio debe correr o andar sobre una cinta mientras se le están monitorizando las señales de ECG. En este caso la persona podrá moverse con mayor libertad si conseguimos eliminar los cables que le unen con el monitor de señales empleado por el personal médico.

Existe un sistema propuesto con anterioridad[CALLE], el cual consiste en un medidor de señales de EMG usando Bluetooth como tecnología inalámbrica. Dicho sistema es un prototipo con varias limitaciones, de las cuales detallamos algunas a continuación:

- Limitaciones Hardware:
  - → Limitación del número de señales simultáneas: Máximo de 5.

- → Frecuencia de muestreo por cada canal fija.
- → Uso de un modelo de microcontrolador con poca capacidad de procesado: El PIC16F876 de Microchip [MCHP1].
- → Código poco reutilizable: Escrito en ensamblador.

➢ Limitaciones Software:

- → Imposibilidad de visualización simultánea de las señales.
- → Imposibilidad de visualización de las señales procedentes de una captura a disco.
- → Una sola fuente de datos: Solo aceptaba señales del dispositivo hardware por Bluetooth™.
- → Imposibilidad de redistribución de las señales a otros nodos en tiempo real.

En cuanto a las limitaciones hardware, aunque PIC16F876 es capaz de ofrecer, como máximo, una frecuencia de trabajo de 5 MIPS usando un oscilador externo de 20 MHz, si usamos dicha frecuencia, el puerto serie que lleva implementado y que comunica el microcontrolador con el módulo wireless, a 115200 baudios presenta un porcentaje de error en las recepciones y envíos demasiado elevado para la correcta comunicación de ambos módulos.

Para disponer de una tasa de errores cercana al 0% se tenía que usar el PIC16F876 conectado a un oscilador de 3.6864 MHz, lo que limita la frecuencia de trabajo a 0.9216 MIPS. Sin embargo la velocidad de transmisión/recepción máxima que se puede conseguir con dicha configuración es de 57600 baudios, y no de 115200 baudios, que es el mínimo necesario (*ver Anexo IV*).

1.2.- Conceptos sobre telemedicina / telediagnóstico

Etimológicamente el término ***telemedicina*** viene de la composición del prefijo griego ***tele*** (*τελε*), que significa “lejos”, y de la palabra ***medicina*** que proviene del latín y significa “materia de curación”[1]. En lineas generales la telemedicina se puede definir como la utilización de las tecnologías de la telecomunicación para la provisión de información y servicios médicos [MED1] [MED2].

Por otro lado el término ***telediagnóstico*** proviene de la composición del mismo prefijo griego ***tele***, de la palabra griega ***diagnosis*** (*διαγνωσιζ*), que significa “capacidad de reconocer”, y del sufijo ***tico*** (*τικοζ*), que significa “relativo a”. Por lo que se puede traducir como la capacidad de reconocer, en este caso problemas médicos, a distancia.

1 Derivación de ***mederi*** = curar y el sufijo -***ina*** que significa “materia de”

En el término telemedicina se engloba el diagnóstico y el tratamiento practicados a distancia, por lo tanto, se podría decir que el telediagnóstico es una parte de la telemedicina. El presente proyecto se centra, principalmente, en el campo del telediagnóstico, dado que consiste en llevar los datos médicos de un paciente al personal sanitario, los cuales les servirán para poder realizar un diagnóstico adecuado.

Algunos ejemplos que podemos encontrar de telemedicina son el envío de radiografías o señales biomédicas por internet para su observación, la realización de consultas por videoconferencia o la capacidad de operar a distancia por medio de equipos robotizados [DAV].

1.3.- Conceptos sobre señales biológicas

Nos referiremos como **señal biológica** o **señal fisiológica** a la representación gráfica bidimensional del registro de la actividad eléctrica de ciertos órganos o partes del cuerpo humano en función del tiempo.

Generalmente, el registro de la actividad eléctrica de una parte del cuerpo humano se lleva a cabo a través de un **electrodo**, el cual es un elemento conductor, normalmente metálico, que recibe o transfiere corriente eléctrica de/a un cuerpo al entrar en contacto con el mismo.

Cabe destacar, que si bien no todas las señales fisiológicas son de tipo eléctrico, como puede ser la presión arterial o la temperatura, en muchas ocasiones dichas magnitudes pueden ser traducidas al dominio eléctrico mediante determinados dispositivos, comúnmente denominados **sensores** o **transductores eléctricos**. Por ejemplo, en el caso de la temperatura, se puede usar un sensor de temperatura, el cual produce una corriente o tensión eléctrica proporcional a la temperatura.

Normalmente, las señales biológicas medidas a través de un sensor o un electrodo, tienen un nivel de tensión, también llamado **amplitud de la señal**, muy bajo, en ocasiones del orden de milivoltios. Además la adquisición de estas señales presentan problemas de ruido, es decir, variaciones de la propia señal debidas a interferencias del ambiente, siendo una fuente de ruido habitual el tendido eléctrico de los hogares que está en el conjunto de frecuencias en torno a 50 Hz o 60 Hz [IEC1]. Debido a estos factores, la señal biológica que proviene de un sensor o un electrodo debe ser tratada por una etapa de **acondicionamiento de la señal**, la cual se encarga de, por un lado amplificar la señal a unos niveles de tensión manejables , y por otro lado, de filtrar la señal para que se mantenga en los límites de frecuencia que debe.

En la siguiente tabla (*Tabla 1)* se resumen las características, tanto en frecuencia como en amplitud, de las principales señales biológicas [CARRE][FERR].

Señal	Margen de tensiones (µV)	Margen de frecuencias (Hz)
ECG (electrocardiograma)	De 500 a 4000	De 0,01 a 250
EEG (electroencefalograma)	De 5 a 300	De continua a 150
EGG (electrogastrograma)	De 10 a 1000	De continua a 1
EMG (electromiograma)	De 100 a 5000	De continua a 10.000
EOG (electrooculograma)	De 50 a 3500	De continua a 50
ERG (electroretinograma)	De 0 a 900	De continua a 50

Tabla 1: Resumen de las características de las señales biológicas más comunes.

1.3.1.- Señales de ECG

ECG es la abreviatura de "electrocardiograma", que es la representación de las señales eléctricas del corazón. En un ECG se suelen tomar hasta un total de 12 señales provenientes de distintas derivaciones de diferentes músculos del corazón.

Las características de esta señal son las siguientes:

- Margen de tensiones: De 0,5 mV a 4 mV.
- Margen de frecuencias: De 0,01 Hz a 250 Hz.

La forma típica de una señal ECG se muestra en la siguiente figura (*Figura 1*).

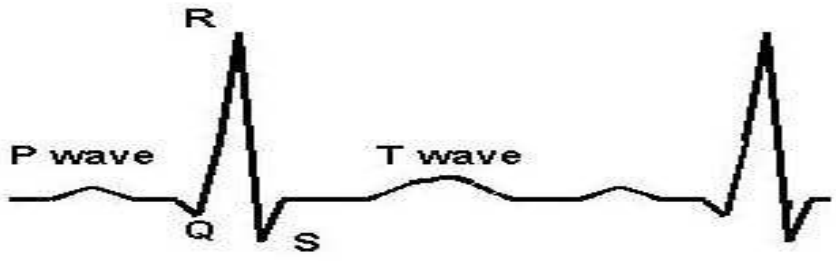

Figura 1: Ejemplo de una señal ECG.

1.3.2.- Señales de EMG

EMG significa "electromiograma", que es la representación de las señales eléctricas que provienen de los músculos. Existen básicamente dos tipos de EMG, por un lado está el EMG de superficie, en el cual los electrodos para capturar la señal se colocan en contacto con la piel, y por otro lado está el EMG interno, en el cual los electrodos son agujas concéntricas (*Figura 2*) que se insertan en las fibras musculares.

Figura 2: Agujas concéntricas.

Dependiendo del tamaño del músculo a monitorizar y del tipo de EMG, las características de la señal capturada varían, sobre todo en lo que al margen de tensiones se refiere, pero unos valores típicos son los siguientes:

- Margen de tensiones: de 0,1 mV a 5 mV.
- Margen de frecuencias: De 0 Hz a 10KHz.

Como vemos en este caso, la escala de frecuencias que encontramos es muy amplia, sin embargo a efectos de monitorización, se toma un margen de frecuencias más angosto, aproximadamente de 20 Hz a 500 Hz. En la siguiente figura (*Figura 3*) se muestra un ejemplo de señal típica obtenida por la contracción de un musculo.

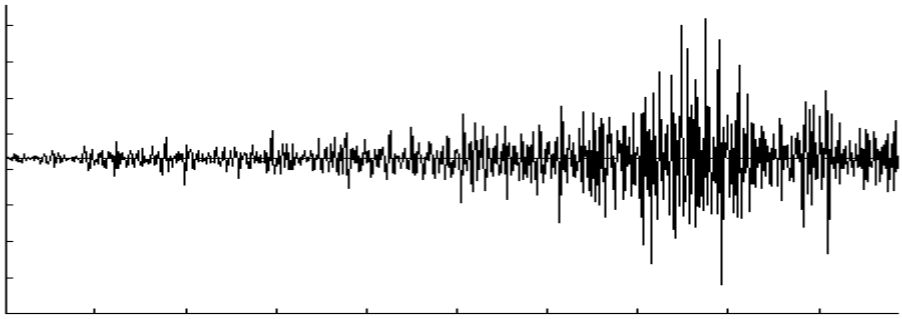

Figura 3: Ejemplo de una señal EMG.

1.3.3.- Señales de EEG.

EEG es la abreviatura de “electroencefalograma” y es la representación de las señales eléctricas producidas por el cerebro. Las neuronas de nuestro cerebro se comunican entre sí mediante estímulos eléctricos y, a través de un EEG, esta tenue actividad eléctrica puede ser medida. En un EEG típico se colocan entre 16 y 25 electrodos en la superficie del cuero cabelludo, por lo tanto, se pueden obtener entre 16 y 25 señales eléctricas diferentes.

Las características de esta señal son las siguientes:

- Margen de tensiones: De 5 µV a 300 µV.

- Margen de frecuencias: De 0 Hz a 150 Hz.

Un ejemplo típico de señales EEG son como muestra la siguiente figura *(Figura 4).*

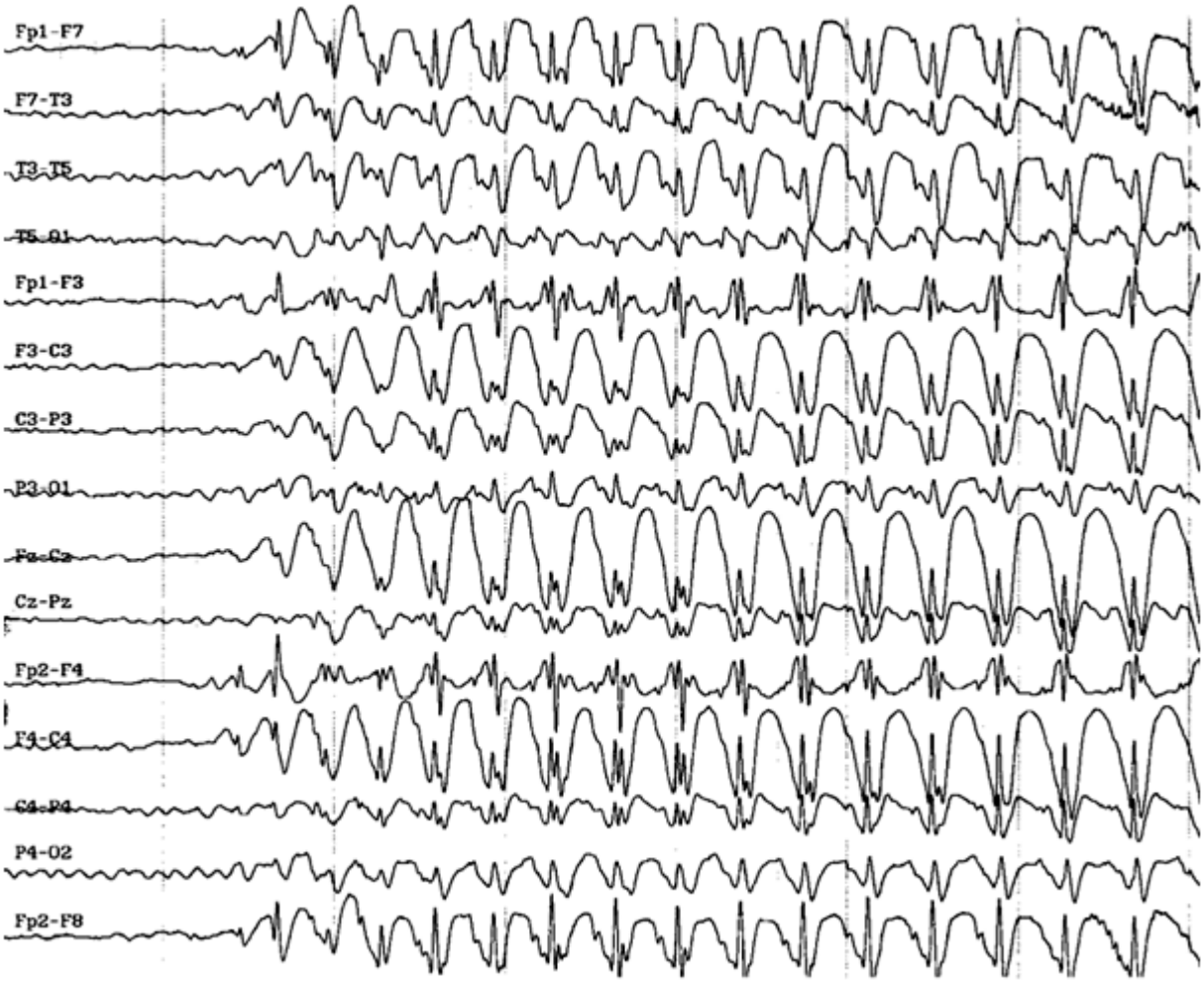

Figura 4: Ejemplo de un EEG de 15 canales

1.4.- Conceptos sobre digitalización de señales

Hasta ahora el concepto que hemos estado tratando de señal se refiere a la variación analógica en el tiempo de una determinada tensión o corriente, sin embargo, para que las señales puedan ser tratadas en el ámbito de la computación, éstas deben ser digitalizadas.

Una señal analógica es aquella cuyas dimensiones (tiempo y amplitud) pueden tomar cualquier valor real, es decir, que entre dos medidas siempre existen valores intermedios, por lo tanto el conjunto de valores que puede tomar una medida es continuo y, en consecuencia, infinito.

En una señal digital, el conjunto de valores que puede tomar una medida está acotado y es discreto. Para discretizar señales analógicas y convertirlas señales digitales se usa un **conversor analógico-digital**.

1.4.1.- El conversor analógico-digital

El conversor analógico digital (ADC, por sus siglas en inglés) es un dispositivo electrónico capaz de convertir **tensiones de entrada** en dígitos binarios, es decir, que toma a la entrada un valor de tensión y a la salida saca un total de N bits, a la que llamaremos **codificación de salida**. Genéricamente a un ADC con N bits de salida se le denomina **ADC de N bits**. A continuación se puede ver esquema básico *(Figura 5).*

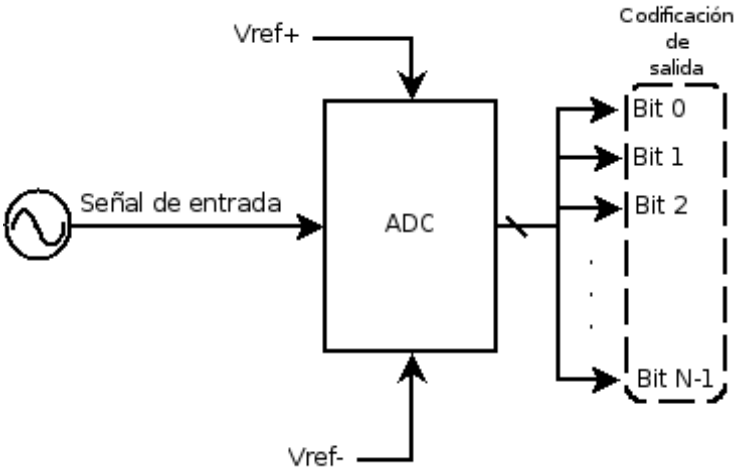

Figura 5: Esquema de un conversor analógico-digital.

Las **tensiones de referencia** V_{REF+} y V_{REF-} indican las cotas máxima y mínima que la señal de entrada debería tomar, cuando la señal de entrada toma un valor igual o superior a V_{REF+} la codificación de salida tomará el valor máximo en binario, es decir $\mathbf{2^N-1}$ (todos los bits a '1'), por el contrario, cuando la señal de entrada toma un valor igual o inferior a V_{REF-} la codificación de salida será el valor mínimo binario,es decir **0** (todos los bits a '0').

El **fondo de escala** (*FSR*) de un ADC es la diferencia de potencial, en valor absoluto, que dicho ADC es capaz de cuantificar. Dicha diferencia de potencial está limitada por las tensiones de referencia V_{REF+} y V_{REF-}, que son el valor máximo y mínimo de la tensión medida, respectivamente. Entonces la fórmula que representa el FSR es la siguiente:

$$FSR = \left| V_{\mathrm{REF+}} - V_{\mathrm{REF-}} \right|$$

Para calcular la codificación de salida (ADC_{OUT}) de un ADC de N bits en función de la tensión de entrada (V_{IN}) se hace aplicando la siguiente fórmula[2]:

$$Si\ V_{REF-} \le V_{IN} \le V_{REF+} \rightarrow ADC_{OUT} = round\left(\frac{(V_{IN} - V_{REF-}) \cdot (2^N - 1)}{FSR}\right)$$
$$Si\ V_{IN} < V_{REF-} \rightarrow ADC_{OUT} = 0$$
$$Si\ V_{IN} > V_{REF+} \rightarrow ADC_{OUT} = 2^N - 1$$

Por el contrario, llamaremos **valor de tensión interpolado** (V_{OUT}) a, dada una codificación de salida ADC_{OUT}, en un ADC de N bits, al valor de tensión que representa. El valor de tensión interpolado se calcula:

$$V_{OUT} = \frac{ADC_{OUT} \cdot FSR}{2^N - 1} + V_{REF-}$$

Por otro lado, la **resolución** (ADC_{RES}) es la variación de tensión mínima necesaria para que la codificación de salida del ADC sufra un cambio. Se calcula como el cociente entre el *FSR* y el número de intervalos que se pueden representar a la salida del ADC. De otra manera, suponiendo que el ADC tiene N bits de salida, el abanico de datos será [0 .. 2^N-1], por tanto, el número de intervalos a representar es $\mathbf{2^N-1}$, por tanto la resolución de un ADC de N bits será:

$$ADC_{RES} = \frac{FSR}{2^N - 1}$$

Por último, el **error máximo** de un ADC (ADC_{ERR}) es la desviación máxima que supone el valor de tensión interpolado (V_{OUT}) con respecto al valor real de tensión tomado a la entrada del ADC (V_{IN}). Se calcula como:

$$ADC_{ERR} = \frac{ADC_{RES}}{2}$$

Dicho de otra manera, si tomamos a la entrada un valor de tensión V_{REAL} en el margen [V_{REF-} .. V_{REF+}] y obtenemos una salida cuyo valor de tensión interpolado es V_{OUT}, entonces se cumple que:

$$(V_{OUT} - ADC_{ERR}) \le V_{REAL} \le (V_{OUT} + ADC_{ERR})$$

2 La operación "*round*" devuelve el valor entero más cercano. Aunque en determinados conversores se justifica al entero inmediatamente inferior ("*floor*").

1.4.2.- La frecuencia de muestreo

Hasta ahora hemos hablado de la digitalización de señales en una sola dimensión, la amplitud de la señal, pero aún nos queda hablar de como se discretiza otra de las dimensiones de una señal, el tiempo.

Debido a que el tiempo en una señal analógica es una dimensión continua, para poder digitalizar una señal, hay que discretizarla, lo que se logra tomando muestras de la señal a intervalos regulares. El intervalo de tiempo que transcurre desde que se toma una muestra de la señal hasta que se toma la siguiente muestra recibe el nombre de **periodo de muestreo** (T_M). Al numero de muestras que se toman por unidad de tiempo se le denomina de **frecuencia de muestreo** (F_M). Ambas magnitudes están relacionadas mediante la siguiente fórmula:

$$F_M = \frac{1}{T_M}$$

Si el periodo de muestreo se mide en segundos, entonces la frecuencia de muestreo está expresada en Hertzios.

Cuanto más alta sea la frecuencia de muestreo, más información tendremos de la señal, puesto que las muestras se tomarán a intervalos de tiempo más cortos, sin embargo, eso supone una mayor cantidad de información que quizá no sea necesaria. Por ejemplo, para una señal de temperatura del cuerpo humano, la cual es prácticamente constante a 36.5º Celsius, no es necesario tener un periodo de muestreo del orden de microsegundos puesto que, probablemente, las muestras tomadas tendrán el mismo valor, en este caso, se podrían tomar muestras cada segundo o incluso cada minuto.

Por otro lado, si tenemos una señal que varía mucho en el tiempo, dicho de otra manera, tiene una frecuencia elevada, con una frecuencia de muestreo demasiado baja estaríamos perdiendo información de la señal, puesto que no se registrarían las variaciones de la señal que estuvieran ubicadas entre dos muestras consecutivas.

Para poder ajustar la frecuencia de muestreo de una señal se utiliza el **teorema de Nyquist-Shannon**, el cual se explica a continuación.

1.4.2.1.- El teorema de Nyquist-Shannon

El teorema de Nyquist-Shannon afirma que, para una señal limitada en banda, la frecuencia de muestreo (F_{MS}) debe ser mayor que el doble de la frecuencia máxima de la señal (F_{MAXS}) para que ésta pueda ser reconstruida sin errores. De este modo:

$$F_{MS} > 2 \cdot F_{MAXS}$$

Esto es debido a que, si se muestrea por debajo del límite de Nyquist-Shannon, aparte de la consecuente pérdida de información de la señal muestreada, se puede producir un efecto conocido como **aliasing**, que básicamente consiste en que las señales analógicas, al ser digitalizarlas con una frecuencia de muestreo insuficiente, aparenten tener una frecuencia mucho menor que la que realmente tienen. En la siguiente figura (*Figura 6*) se puede apreciar el mencionado efecto aliasing.

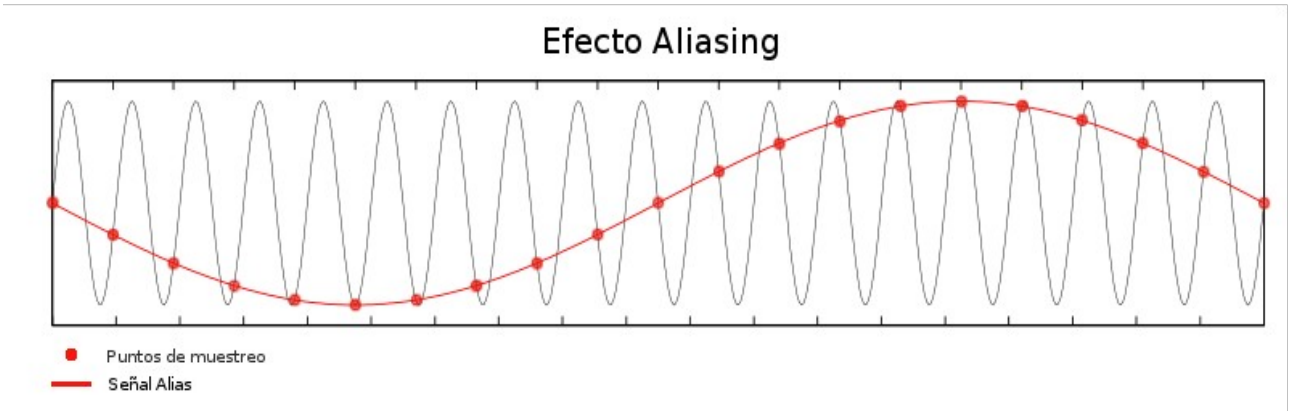

Figura 6: Ejemplo de efecto aliasing debido a una baja frecuencia de muestreo.

Entonces, según el teorema de Nyquist-Shannon, la frecuencia de muestreo para cada una de las señales biológicas enumeradas con antelación (*ver 1.3*) se puede ver reflejada en la siguiente tabla *(Tabla 2)*.

Tipo de señal	Frecuencia Máxima (Hz)	Frecuencia de muestreo mínima (Hz)
ECG (electrocardiograma)	250	500
EMG (electromiograma)	500[3]	1000
EEG (electroencefalograma)	150	300
EGG (electrogastrograma)	1	2
EOG (electrooculograma)	50	100
ERG (electroretinograma)	50	100

Tabla 2: Resumen con las frecuencias de muestreo para determinadas señales.

3 Frecuencia máxima a efectos de monitorización.

1.5.- Conceptos sobre comunicación inalámbrica

Utilizaremos el término **comunicación inalámbrica** o **wireless** para referirnos a la capacidad de realizar transferencias de datos entre dos o más dispositivos sin la necesidad de usar cables como medio de interconexión.

Hoy en día existen multitud de dispositivos que utilizan tecnologías inalámbricas como medio de comunicación, por ejemplo los teléfonos móviles, PDAs, ordenadores portátiles, aparatos de GPS e incluso el mando a distancia del televisor o del equipo de audio.

Podemos clasificar las tecnologías inalámbricas, en función de la banda de frecuencia del espectro electromagnético sobre la que operan, en dos tipos:

- Tecnologías inalámbricas basadas en medios ópticos.
- Tecnologías inalámbricas basadas en radiofrecuencia.

Las **tecnologías inalámbricas basadas en medios ópticos**, son aquellas que aprovechan las propiedades físicas de la luz. En concreto usan zonas de alta frecuencia del espectro electromagnético para modular las señales de datos que desean transmitir, habitualmente en torno a la banda visible del espectro. En este marco se engloban tecnologías tales como la infrarroja, o la tecnología láser. Aunque éstas presentan como ventaja una velocidad de transmisión potencialmente alta, también presentan el inconveniente de que entre el emisor y el receptor tiene que haber una línea de visibilidad, requisito que muchas veces no es alcanzable ni mediante el uso de espejos.

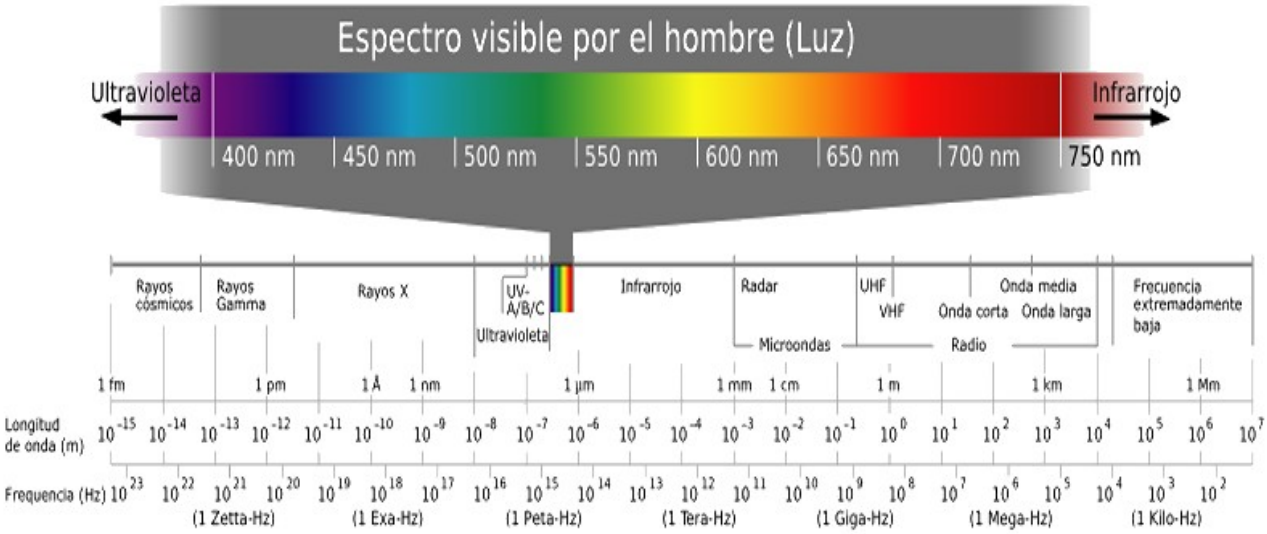

Figura 7: El espectro electromagnético.

Las **tecnologías inalámbricas basadas en radiofrecuencia**, por el contrario, utilizan una zona de menor frecuencia del espectro electromagnético (*Figura 7*), la banda de radiofrecuencia, para modular las señales de datos a transmitir. Son las más utilizadas hoy en día para realizar comunicaciones inalámbricas y poseen como ventaja que no tienen porqué tener visibilidad directa entre emisor y receptor, aunque hay determinados materiales que pueden hacer que la señal se pierda o atenúe de manera drástica.

Existen multitud de tecnologías basadas en radio frecuencia, como por ejemplo:

- Bluetooth™
- Wi-Fi
- ZigBee
- Wireless USB

En las siguientes secciones profundizaremos en las tecnologías inalámbricas basadas en radiofrecuencia de mayor relevancia para el presente proyecto.

1.5.1.- Bluetooth™

Bluetooth™ es el nombre común de la especificación industrial IEEE 802.15.1 que define un estándar global de comunicación inalámbrica segura que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia.

Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre nuestros equipos personales.

La tecnología Bluetooth™ comprende hardware, software y requerimientos de interoperabilidad, por lo que para su desarrollo ha sido necesaria la participación de los principales fabricantes de los sectores de las telecomunicaciones y la informática, tales como Ericsson, Nokia, Motorola, Toshiba, IBM e Intel, entre otros.

1.5.1.1.- Historia de Bluetooth™

El nombre procede del rey escandinavo Harald Blatand (Harold Bluetooth , en inglés) conocido por unificar las tribus noruegas, suecas y danesas. Haciendo un paralelismo, Bluetooth intenta unir diferentes industrias, como la de los ordenadores, teléfonos móviles, la automoción y la de multitud de periféricos.

De hecho el logotipo de Bluetooth es la unión de las runas nórdicas *"Hagalaz"* (la letra 'H') y *"Berkana"* (la letra 'B'), que coinciden con las siglas del mencionado rey *(Figura 8).*

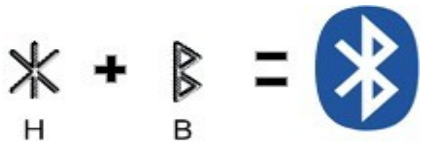

Figura 8: Origen del logotipo Bluetooth™.

En 1994, Ericsson inició un estudio para investigar la interconexión de dispositivos vía radio con la finalidad de no tener que depender de un medio cableado. El estudio partía de un largo proyecto que investigaba unos multicomunicadores conectados a una red celular, hasta que se llegó a un enlace de radio de corto alcance, llamado MC link.

Después de esto, numerosas compañías como Nokia, Toshiba, Intel e IBM se agruparon junto con Ericsson en lo que hoy se conoce como SIG[SIG1] y se dedicaron a desarrollar esta tecnología de manera que fue evolucionando hasta llegar a lo que hoy en día conocemos por Bluetooth™.

1.5.1.2.- Especificaciones técnicas de Bluetooth™

La frecuencia de radio con la que trabaja se sitúa en la escala de 2.4 a 2.48 GHz de la banda ISM, disponible a nivel mundial y que no requiere licencia de operador, lo que garantiza una compatibilidad universal entre dispositivos Bluetooth. Con el fin de evitar interferencias con otros protocolos que operen en la misma banda de frecuencias (*ver 1.5.2*), Bluetooth emplea la técnica de salto de frecuencias, conocida como FHSS, que consiste en dividir la banda en 79 canales (23 en España, Francia y Japón) de longitud 1 MHz y realizar 1600 saltos por segundo.

A partir de su primera versión, la 1.0, que se ratificó en julio de 1999, se han publicado sucesivas versiones:

- **Versión 1.1**:
  - → Soluciona erratas de la especificación 1.0.

- → Añade el Indicador de Calidad de Señal Recibida o RSSI.

- **Versión 1.2**:
  - → Implementa la técnica de salto en frecuencia AFH (una variante de FHSS) para mejorar la resistencia a interferencias.
  - → Introduce el tipo de enlace para aplicaciones de audio eSCO, que mejora la calidad de voz.
  - → Mejoras en el HCI, para una sincronización más rápida de las comunicaciones.

- **Versión 2.0**:
  - → Nueva versión compatible con las anteriores 1.x.
  - → Incorpora la tecnología EDR, que incrementa la velocidad de transmisión hasta 3 Mbps.
  - → Reducción del consumo de energía a pesar del incremento de velocidad.

La velocidad de transmisión varía según versiones del núcleo:

- **Versión 1.1**: Hasta 723.1 Kbps.
- **Versión 1.2**: Hasta 1 Mbps.
- **Versión 2.0 + EDR**: Desde 2.1 - 3 Mbps.

Atendiendo a la potencia de transmisión, los dispositivos Bluetooth se clasifican en 3 tipos:

- **Clase 1**: 100 mW / 20 dBm, con un alcance de aproximadamente 100 metros.
- **Clase 2**: 2.5 mW / 4 dBm, con un alcance de aproximadamente 10 metros.
- **Clase 3**: 1 mW / 0 dBm, con un alcance de aproximadamente 1 metros.

Se definen dos tipos de enlaces para soportar aplicaciones de voz y datos:

- Enlace asíncrono sin conexión (ACL):
  - → Conexiones simétricas o asimétricas punto-multipunto entre maestro y esclavo.
  - → Conexión utilizada para tráfico de datos.
  - → Sin garantía de entrega, se retransmiten paquetes.
  - → La máxima velocidad de envío es de 721 Kbps en una dirección y 57.6 Kbps en la otra.

- Enlace síncrono orientado a conexión (SCO):
  - → Conexiones simétricas punto a punto entre maestro y esclavo.
  - → Conexión capaz de soportar voz en tiempo real y tráfico multimedia.
  - → Velocidad de transmisión de 64 KBps

1.5.1.3.- La pila de protocolos Bluetooth™

La pila de protocolos Bluetooth™[SIG2] es la especificación desarrollada por el SIG, que define como funciona la tecnología bluetooth, proporcionando un estándar para que diversas implementaciones del protocolo puedan interoperar.

Como se muestra en la siguiente figura (*Figura 9*), se encuentra constituida por varios niveles siguiendo un modelo similar al definido por OSI (*Figura 10*).

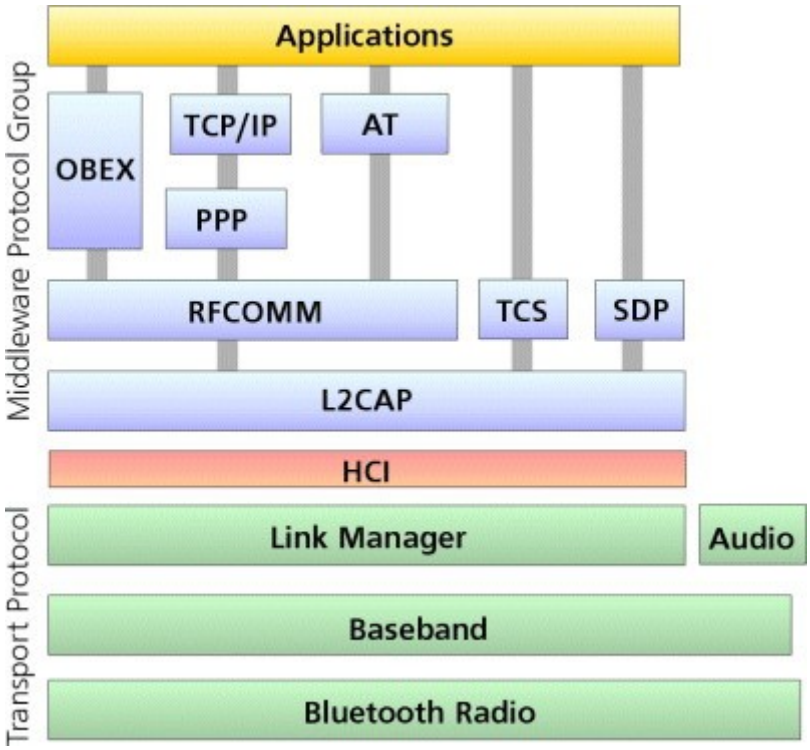

Figura 9: La pila de protocolos Bluetooth™

Algunas de las capas de la pila Bluetooth™ se comentan a continuación:

- **Baseband (Banda base) y Bluetooth Radio**: Su función principal es permitir el enlace físico por medio del uso de radiofrecuencia entre unidades Bluetooth, realizando las modulaciones y demodulaciones de los datos para que éstos puedan ser transmitidos/recibidos.
- **Link Manager (Gestor del enlace)**: Es la capa responsable de la configuración y control del enlace entre dispositivos Bluetooth, incluyendo el control y negociación del tamaño de los paquetes a manejar por la banda base.
- **HCI**: Es la capa encargada de abstraer el hardware de capas inferiores para que pueda ser manejado por el software de las capas superiores.

- **L2CAP**: Ofrece una capa homogénea para las capas superiores, además se encarga de la segmentación de paquetes de datos demasiado grandes para la capa de banda base.
- **SDP**: Proporciona un protocolo para realizar la búsqueda de servicios ofrecidos por un dispositivo Bluetooth™.
- **RFCOMM**: Proporciona la emulación de cable serie RS-232.
- **AT**: Protocolo basado comandos formados por cadenas de caracteres, tradicionalmente usado para el control y configuración de módems. El nombre viene de que los comandos suelen empezar por la cadena “AT”.
- **OBEX**: Protocolo que facilita el intercambio de objetos binarios entre dispositivos.

Además, la especificación Bluetooth™ ofrece una gran flexibilidad a la hora de añadir nuevas capas de protocolos a las ya existentes.

1.5.2.- Wi-Fi

Los protocolos de la rama 802.x de la IEEE son los que definen la tecnología referente a las redes de área local y metropolitana, siendo el estándar definido por la norma IEEE 802.11 al que conocemos comúnmente por el nombre de Wi-Fi.

El homónimo cableado de dicho estándar seria CSMA/CD, más conocido como **Ethernet**, el cual está regido por la norma IEEE 802.3. En ambos casos se define como trabajan las dos capas inferiores del modelo OSI *(Figura 10)*, es decir, las relativas al nivel físico y al nivel de enlace.

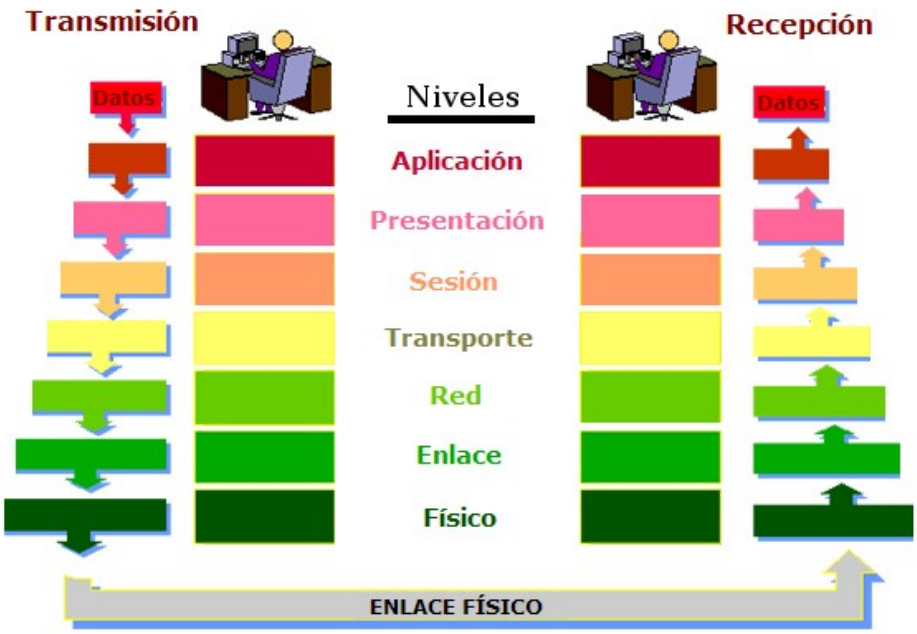

Figura 10: Modelo OSI de comunicación.

Dentro del estándar IEEE 802.11 se han hecho multitud de revisiones, las cuales se codifican mediante una letra a continuación del estándar. Las más comunes actualmente, en territorio europeo, son 802.11a, 802.11b, 802.11g y 802.11n las cuales se diferencian, fundamentalmente, en la banda base en la que operan y en su velocidad máxima de transmisión. Dichas diferencias se pueden ver reflejadas en la siguiente tabla:

Estándar	Frecuencia de banda base (GHz)	Velocidad máxima (Mbps)
802.11a	5	54
802.11b	2.4	11
802.11g	2.4	54
802.11n	2.4 y 5	600

Tabla 3: Comparativa estándares 802.11 más comunes.

1.6.- Conceptos sobre Java

Java es un lenguaje de programación orientado a objetos desarrollado por la compañía SUN Microsystems a principios del año 1990. Su sintaxis es muy similar a la del lenguaje C++, también orientado a objetos, sin embargo Java elimina la utilización explícita de punteros y la manipulación directa de memoria, debido a que suele inducir a muchos errores de ejecución.

Una de las características que lo hace más atractivo es su filosofía multiplataforma, basándose en el concepto de máquina virtual, del cual hablaremos más adelante. Es fácil de usar y existen magníficos IDEs para programar en este lenguaje, entre los que cabe destacar NetBeans [NETB], desarrollado por Sun, y su alternativa, Eclipse [ECLI], además ambos proveen numerosas herramientas y plug-ins para poder ser utilizados con otros lenguajes.

1.6.1.- Historia de Java

Java comenzó en 1991, en el ámbito de un proyecto llamado "*Green*" cuyo objetivo inicial era el de crear un nuevo lenguaje para construir aplicaciones que fuesen capaces de ejecutarse en electrodomésticos dotados de microprocesadores, sin embargo el equipo de desarrolladores se dio cuenta de que este tipo de tecnología estaba aún muy lejos de poder existir.

El proyecto dio como resultado un lenguaje de programación al que se llamó inicialmente "*Oak*" en honor al viejo roble que se encontraba situado en el exterior de la oficina de James Gosling, director del equipo del proyecto. Sin embargo dicho nombre no pudo establecerse como definitivo puesto que descubrieron que "*Oak*" era ya una marca registrada, por lo que finalmente acabó por llamarse "Java"[4].

Figura 11: Logotipo de Java™

La primera implementación de Java fue publicada en 1996. Una de sus máximas era "*Write once, run anywhere*", es decir "Escribe una vez, ejecuta en cualquier lugar", lo que teóricamente hacía que no supusiera ningún costo adicional migrar una misma aplicación entre distintas plataformas. Su popularidad creció rápidamente debido a que los principales navegadores incluyeron el soporte para ejecutar applets Java incrustados en las páginas web.

Con la llegada de "Java 2", surgieron nuevas versiones pensadas para diferentes tipos de plataformas, de ahí que surjan las siguientes versiones:

- J2EE: Para aplicaciones empresariales.
- J2ME: Para aplicaciones embebidas en dispositivos dotados con pocos recursos, como móviles.
- J2SE: La designación para la versión estándar.

1.6.2.- La máquina virtual de Java

Como ya se ha mencionado con anterioridad, una de las características de Java es el hecho de que es multiplataforma, es decir, que puede ser ejecutado con independencia del sistema operativo y de la arquitectura hardware subyacente.

Para ello, cuando se compila el código de una aplicación escrita en lenguaje Java, éste genera un código de nivel inferior denominado "*Bytecode*", que son instrucciones que pueden ser entendidas por un software especial llamado "máquina virtual de Java", o por sus siglas, **JVM**. Es la JVM, la

4 Ciertas teorías sostienen que el nombre viene dado por la siglas de los creadores: **J**ames Gosling, **A**rthur **V**an Hoff y **A**ndy Bechtoslheim, otras teorías apuntan a que puede ser el acrónimo de "***J**ust **A**nother **V**ague **A**cronym*" (sólo otro acrónimo ambiguo, en ingles). La teoría que más fuerza tiene es la de que viene del nombre de una cafetería cercana en la que se reunían durante horas.

encargada de interpretar las instrucciones en "*Bytecode*" y traducirlas al código máquina específico para la arquitectura hardware sobre la que se ejecute la aplicación.

Por tanto, para poder ejecutar aplicaciones Java en un sistema, previamente se debe haber instalado una JVM, a través de la instalación de un "*Java Runtime Environment*" (JRE)[JAVA], del cual existen diferentes versiones en función del sistema operativo y de la plataforma sobre la que se desea instalar, si bien es cierto que no existen versiones de JRE para la totalidad de sistemas operativos existentes, si es cierto que existen versiones disponibles para la mayoría de los sistemas operativos más usuales.

Además se dispone de numerosas librerías para facilitar el acceso a los servicios más comunes de un sistema, como puede ser la ejecución mediante "*threads*", el acceso a los interfaces de red o el acceso al sistema gráfico, y todo esto de manera unificada.

1.6.3.- El recolector de basura de Java

Otra de las características de Java es que utiliza el denominado "recolector de basura", también conocido por su voz anglosajona como "*garbage collector*".

En un programa escrito en otros lenguajes orientados a objetos como C++, cuando se crea un objeto, en tiempo de ejecución se reserva un espacio en memoria para él, cuando dicho objeto ya no es útil debe ser explícitamente eliminado para liberar la porción de memoria que está usando y pueda ser utilizada para otros fines. En Java esto no es necesario puesto que es el entorno de ejecución de Java es el que decide si un objeto ya no es útil y lo elimina.

La manera en la que el recolector de basura decide si un objeto ya no es útil es realizando un seguimiento de las referencias que existen a los objetos creados, si un objeto se queda sin referencias, es decir, ya nadie está apuntando a dicho objeto, el recolector de basura automáticamente lo borra, liberando los recursos de memoria asociados a dicho objeto.

De esta manera se minimiza el efecto de "fugas de memoria" producido por despistes del programador, aunque no lo elimina completamente, puesto que puede ser que el programador guarde inconscientemente referencias a objetos que ya no son útiles o puede haber casos más complejos de referencias recíprocas o cíclicas, es decir, que dos o más objetos, que ya no son útiles, estén apuntándose mutuamente.

Sin embargo, la técnica de usar un recolector de basura tiene algunas desventajas, como son el hecho de que la memoria puede estar retenida más tiempo del estrictamente necesario o una pérdida de eficiencia debido a que el recolector de basura tarda un determinado tiempo en hacer su trabajo. Además es complicado decidir cuando debe ser ejecutado el recolector de basura, aunque de todas maneras, Java también permite que el programador dicte cuando debe ser ejecutado.

2.- Objetivos

El objetivo del proyecto es el diseño de un sistema inalámbrico capaz de realizar el envío simultáneo, la monitorización y la gestión de varias señales analógicas, en particular de varias señales biológicas, que no tengan que ser necesariamente del mismo tipo.

Por otro lado se requiere que el sistema sea capaz de retransmitir las señales a otros nodos en la red, permitiendo la monitorización remota de señales sin la necesidad de implementar aplicaciones adicionales para dicho propósito.

El sistema se puede dividir en dos bloques *(Figura 12)*, el subsistema hardware y el subsistema software.

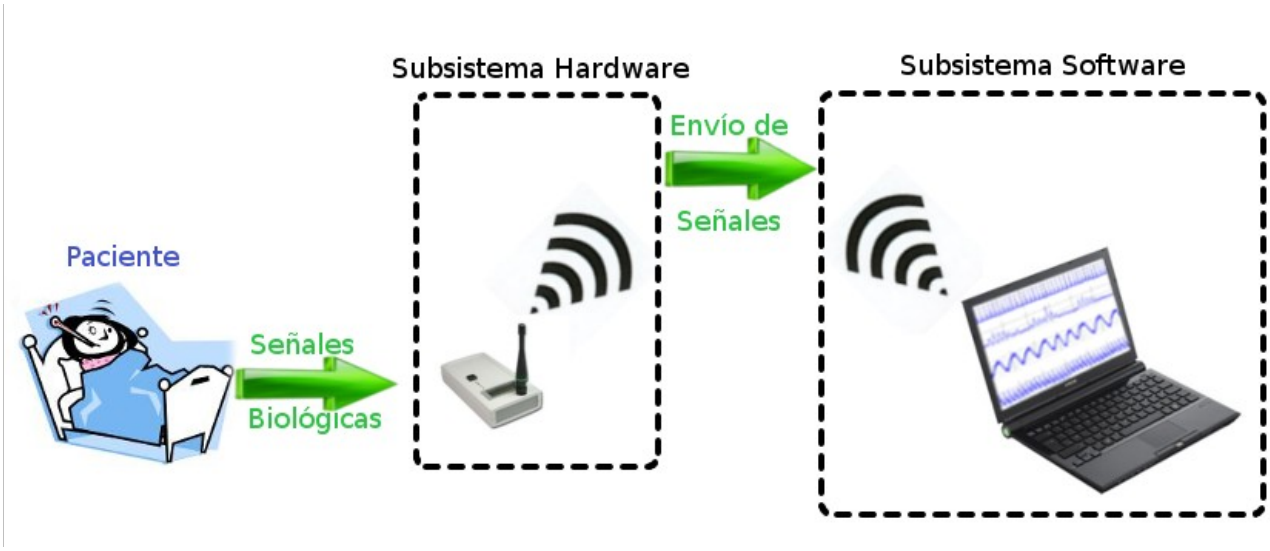

Figura 12: Esquema general del sistema completo.

A continuación describiremos en detalle los objetivos perseguidos para cada uno de los subsistemas involucrados en el presente proyecto.

2.1.- Objetivos del subsistema hardware

Antes de establecer los objetivos del subsistema hardware se va a definir la terminología empleada para el presente apartado.

Denominaremos **canal de señal** al registro o adquisición de una señal biológica de forma independiente, es decir, que puede ser realizado en paralelo con otros registros o adquisiciones de señal.

Hablaremos de **canales de señal heterogéneos** para hacer hincapié en el hecho de que cada una de las señales biológicas que componen el conjunto de canales, puede ser de diferente naturaleza con respecto del resto de señales.

Definiremos como **dispositivo hardware** a la implementación del subsistema hardware.

El primero de los objetivos, del subsistema hardware, consiste en diseñar y fabricar un dispositivo hardware capaz de realizar el envío de forma inalámbrica de, como mínimo, 5 canales de señal heterogéneos. Además se debe permitir cierta flexibilidad en cuanto al número de canales monitorizados. Para ello, se debe establecer algún método que permita variar tanto el número de canales, como la frecuencia de muestreo (*ver 1.4.2*) del conjunto de canales de señal.

El subsistema hardware no debe incluir en su diseño ninguna etapa de acondicionamiento de señal biológica, de manera que dichas etapas puedan ser independientes y, mediante algún mecanismo, puedan ser conectadas a este subsistema, aumentando la flexibilidad del mismo en cuanto a heterogeneidad de señales se refiere. Es decir, al subsistema hardware se le deben conectar etapas de acondicionamiento de la señales, no las señales que provienen directamente de los electrodos o sensores que capturan la señal biológica.

El siguiente objetivo requiere que, en la medida de lo posible, el envío de forma inalámbrica no sea cerrado, es decir, que pueda permitir el uso de diferentes tecnologías inalámbricas.

El dispositivo debe ser portátil, de reducidas dimensiones y bajo peso, de bajo coste y, a ser posible, con un consumo energético bajo.

Por último se requiere que el uso del dispositivo sea sencillo, a fin de que su manejo y configuración no demande personal técnicamente cualificado.

2.2.- Objetivos del subsistema software

Al igual que en el anterior apartado, se van a realizar las definiciones previas necesarias para el desarrollo de los objetivos del sistema software.

Definiremos el término **monitorización de señal** como la recepción y representación gráfica de una serie de datos de determinada señal.

Definiremos como **vínculo inalámbrico** a la conexión o enlace inalámbrico entre dos dispositivos. Un vínculo inalámbrico es el que permite la transferencia de datos de manera inalámbrica entre los extremos que lo comparten.

Denominaremos **servidor** a aquel que está en espera de peticiones de establecimiento de conexión y que, tras su establecimiento, envía datos por dicha conexión. Por otro lado, definiremos como **cliente** a aquel que inicia la petición de establecimiento de una conexión a un servidor y que además se encarga de recibir los datos enviados por éste.

Utilizaremos el término **memoria no volátil** para referirnos al tipo de memoria de un sistema que persiste aunque dicho sistema sea apagado, como por ejemplo un disco duro o una memoria Flash.

Definiremos como **nodo** al sistema que está ejecutando la implementación del subsistema software.

El primero de los objetivos del subsistema software consiste en diseñar e implementar una aplicación para PC que sea capaz de monitorizar un número variable de canales de señal heterogéneos, proporcionados por un dispositivo hardware. Para ello, la aplicación debe proporcionar mecanismos para la búsqueda de los dispositivo hardware disponibles, así como para la creación y clausura de un vínculo inalámbrico con dicho dispositivo. De este modo el dispositivo hardware actuará como servidor de señales.

En el caso de que el dispositivo hardware envíe más de una señal de forma simultánea, la aplicación debe ser capaz de monitorizarlas de forma paralela, es decir, que todas las señales puedan ser visualizadas en pantalla al mismo tiempo.

Es indispensable que la aplicación permita la captura de todas las señales que están siendo monitorizadas a memoria no volátil, debiendo proporcionar mecanismos para el arranque y la parada de la captura de los datos por parte del usuario.

Para complementar el objetivo anterior se desea que la aplicación sea capaz de obtener los datos de señal de una captura previamente guardada y mostrar su representación en pantalla, de forma similar a como se hace cuando se realiza una monitorización.

Otro de los objetivos del subsistema software consiste en que la aplicación permita la retransmisión de los datos que están siendo monitorizados usando internet[5] como medio de comunicación, permitiendo la recepción de peticiones de establecimiento de conexión procedentes de otras instancias de la misma aplicación, con el fin de poder distribuir la información de señal entre distintos nodos. Dicho de otra manera, la aplicación diseñada debe brindar la posibilidad de actuar como cliente y a su vez como servidor. A este comportamiento lo hemos denominado ***"peer to peer"*** (P2P).

Del objetivo anterior se deriva que la aplicación debe brindar la posibilidad de realizar peticiones de conexión a otros nodos de la red.

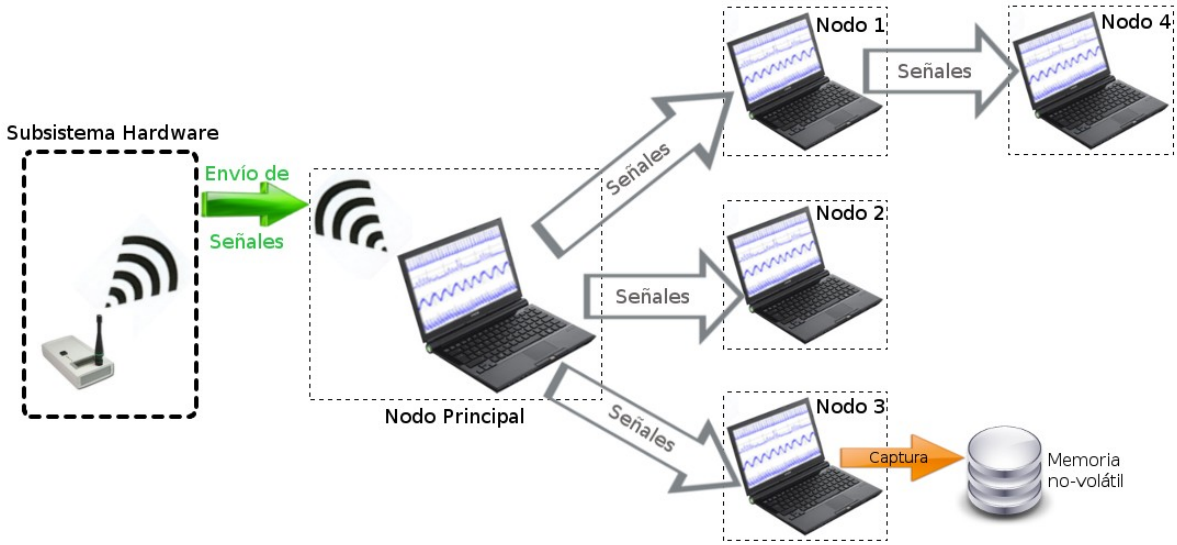

Figura 13: Ejemplo de escenario complejo del sistema.

En la anterior figura *(Figura 13)* podemos observar un ejemplo de escenario que ilustra las posibilidades del sistema, centrándose en el comportamiento P2P mencionado previamente.

En este ejemplo, el "Nodo Principal" está obteniendo los datos servidos por el dispositivo hardware mediante un vínculo inalámbrico, mientras que, a su vez, está retransmitiendo los datos a los nodos 1, 2 y 3 a través de una conexión de red. Por otro lado el "Nodo 1" está retransmitiendo los datos que le llegan del "Nodo Principal" al "Nodo 4", mientras que el "Nodo 3" se está encargando de guardar los datos de señal en un soporte de memoria no-volátil. Todos los nodos muestran la representación gráfica de las señales recibidas en sus respectivas pantallas.

5 Puede ser a una red de área local o de área extensa, dependiendo de la configuración del red del equipo en el que se ejecuta la aplicación.

Otro de los requisitos es que la aplicación sea multiplataforma, es decir, que pueda ser ejecutada en distintos entornos sin que suponga realizar una reprogramación o recompilación del código fuente de la misma.

Por último, otra de las características que debe cumplir la aplicación software es que debe tener una interfaz de usuario amigable, atractiva e intuitiva.

3.- Descripción de alternativas

Al igual que en la sección referente a los objetivos del proyecto vamos a dividir el estudio de alternativas en dos apartados, por un lado las referentes al subsistema hardware y por otro las referentes al subsistema software.

3.1.- Subsistema hardware: estudio de alternativas.

En líneas generales, el subsistema hardware puede dividirse, a nivel lógico, en las siguientes unidades *(Figura 14):*

- Unidad de Control.
- Unidad Inalámbrica.
- Interfaz Analógica.
- Interfaz de Administración.

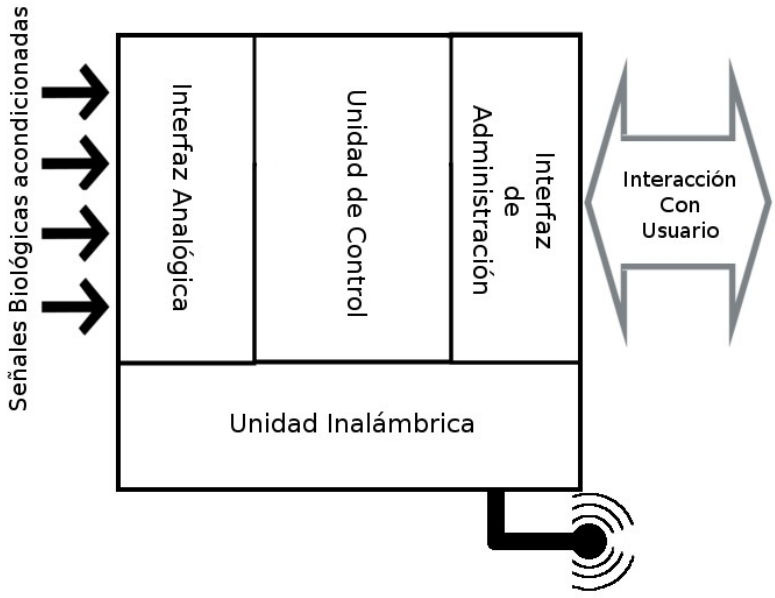

Figura 14: Unidades lógicas del subsistema hardware

3.1.1.- La Unidad de Control

La Unidad de Control es el núcleo de procesamiento del dispositivo desarrollado. Se encarga de digitalizar (*ver 1.4*) las señales que provienen de la Interfaz Analógica con determinada frecuencia, realizar el envío de los datos digitalizados al Módulo Inalámbrico y atender las peticiones de la Interfaz de Administración. En resumen, es la encargada de gestionar todos los recursos del dispositivo.

Dentro de la variedad de circuitos integrados existentes, la solución óptima para la realización de sistemas embebidos es mediante el uso de **microcontroladores**, debido a su bajo coste y a la alta capacidad de integración con otros periféricos.

A su vez, podemos encontrar una amplia oferta de fabricantes de microcontroladores como por ejemplo Atmel, Freescale , Intel o Microchip. Además, cada fabricante agrupa sus productos en **familias**, que es una agrupación de diferentes **modelos** que tienen características y prestaciones similares.

En la elección del modelo de microcontrolador se tuvieron en cuenta los siguientes criterios de selección:

- Experiencia previa con otros microcontroladores.
- Máxima frecuencia de trabajo.
- Encapsulado PDIP de 28 pines: para que se facilite la fabricación del hardware con lo medios disponibles.
- Número de UART: se requieren, al menos, 2 UART para que el microcontrolador se comunique con la unidad inalámbrica y con la interfaz de administración.
- ADC integrado, con el máximo número de entradas analógicas.
- Máxima tasa de muestreo del ADC: para poder maximizar la frecuencia de las señales analógicas de entrada (*ver 1.4.2*).

Debido a la experiencia previa con otros dispositivos de la marca Microchip , se decidió realizar una búsqueda de un dispositivo de dicha marca [MCHP2] que satisfaciese los anteriores criterios de selección. Dentro de los resultados obtenidos, se optó por seleccionar el modelo **dsPic30F3013**, perteneciente a la familia de procesadores digitales de señal de Microchip, el cual cuenta con las siguientes características:

- Hasta 30 MIPS de frecuencia de trabajo.
- Arquitectura de 16 bits.
- Reducidas dimensiones (35.18 mm x 7.5 mm), en su formato PDIP de 28 pines.
- 2 UART.
- ADC de hasta 10 entradas analógicas, con una tasa de muestreo máxima de 200 ksps.

Por otro lado, otra de las decisiones a tomar en cuanto a la Unidad de Control, es la manera en la que se programaría el microcontrolador elegido. Existen varias alternativas en cuanto a lenguajes de programación para microcontroladores de Microchip se refiere. En primer lugar se puede programar

el chip usando código escrito en **ensamblador**, también llamado **ASM**, el cual es un lenguaje de bajo nivel muy dependiente del modelo de microcontrolador elegido. En segundo lugar podemos usar lenguajes de alto nivel para la programación del chip, en este caso se debe encontrar un compilador específico del lenguaje seleccionado para la familia de microcontrolador en cuestión. Este compilador se encargará de traducir la sintaxis de alto nivel, utilizada en el código fuente del programa, a código máquina que pueda ser ejecutado por el microcontrolador.

Ambas opciones presentan ventajas e inconvenientes. Por un lado, el código en ASM es muy eficiente, esto quiere decir que el programador, debido al control a bajo nivel que ofrece el lenguaje, puede optimizar el tiempo de ejecución de los bloques de código y los recursos involucrados en la ejecución del programa. Sin embargo la programación en ASM es una labor muy tediosa y propensa a errores, con lo que el tiempo de desarrollo se ve incrementado exponencialmente para realizar implementaciones que requieran procesamientos complejos. Por otro lado, la programación en un lenguaje de alto nivel presenta como virtud que el tiempo de desarrollo disminuye y se puede llegar a desarrollar programas complejos en un tiempo razonable, debido a que se facilita la codificación del código fuente y se minimizan los errores en tiempo de ejecución. Además, usando un lenguaje de alto nivel, se aumenta la portabilidad del programa a otros modelos de microcontrolador. En contraposición se pierde control con respecto a la arquitectura de la máquina y posiblemente eficiencia, sin embargo los compiladores actuales para este tipo de situaciones suelen ser excelentes en el ámbito de la optimización del código generado.

Para este proyecto, se ha elegido utilizar una programación en un lenguaje de alto nivel, debido a que se espera que el programa cargado en el dsPIC30F3013 tenga una complejidad alta, además, gracias a la elevada frecuencia de trabajo que puede llegar a alcanzar este dispositivo, se ha decidido que merece la pena la posible pérdida de eficiencia frente al aumento de la portabilidad del código fuente generado y la facilidad de programación, lo que implica una reducción del tiempo de desarrollo.

El lenguaje de programación usado es **C**, un lenguaje de alto nivel ampliamente extendido en el ámbito de la programación. En este caso, se usa un compilador llamado "*CCS C*" [CCS], que usa la sintaxis de C ligeramente modificada y proporciona algunas librerías y funciones de gran utilidad, sobre todo para la gestión de la entrada/salida y el manejo de cadenas de caracteres. Además se puede integrar con MPLAB, que es el IDE que facilita Microchip para la programación de sus microcontroladores [MCHP3].

3.1.2.- La Unidad Inalámbrica

La Unidad Inalámbrica es la encargada de implementar la tecnología de comunicación wireless del subsistema hardware. Proporciona el nivel de abstracción necesario para simplificar el establecimiento de vínculos inalámbricos con otros dispositivos que soporten la misma tecnología, así como para el envío de datos sobre dichos vínculos.

La tecnología inalámbrica elegida para el desarrollo del proyecto ha sido Bluetooth, debido a múltiples factores:

- Es una tecnología madura y ampliamente extendida en el mundo de las comunicaciones inalámbricas.
- Existen multitud de fabricantes hardware que dan soporte para dicha tecnología.
- Se pueden alcanzar márgenes de cobertura aceptables para la finalidad de este proyecto.
- Permite velocidades de transmisión suficientes para nuestros propósitos.
- Existen múltiples implementaciones de la pila de protocolos Bluetooth para diversos sistemas.
- Uso extensivo en terminales móviles y en dispositivos portátiles para PCs.

Dentro de los dispositivos Bluetooth existentes en el mercado se ha elegido el chip **BISM II Serial Module** [BISM] de la marca **Ezurio** [LAIRD1], el cual tiene las siguientes características.

- Clase de Bluetooth: Clase 1.
- Versión de Bluetooth: 2.0.
- Dimensiones reducidas: 22.8 mm x 33.8 mm x 7.6 mm.
- Interfaz de comunicación: RS-232 (UART).
- Protocolo de comunicación: Comandos AT.
- Especialmente indicado para aplicaciones médicas, dado que cumple con el estándar EN-60601-1-2 [IEC2].

Este dispositivo utiliza un zócalo SMD, de 40 pines del tipo HIROSE, que lo hace idóneo para cumplir el requisito del sistema referente a que la tecnología inalámbrica empleada sea lo menos cerrada posible, puesto que existe otro modelo de dispositivo de la misma marca, el **WISM** [WISM], que usa el mismo zócalo y tiene similares dimensiones (*Figura 15*).

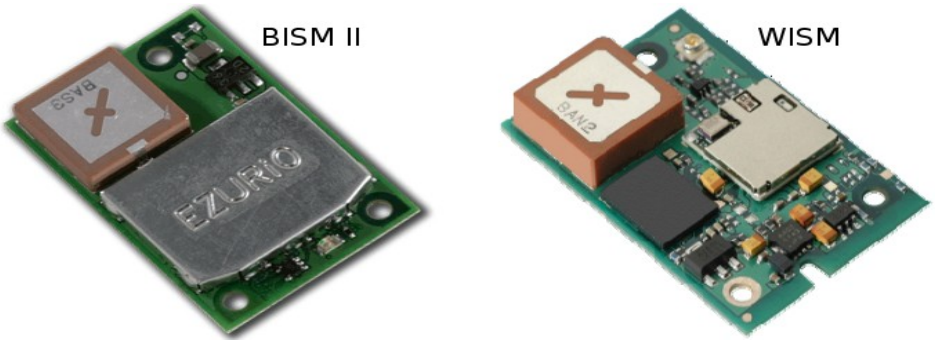

Figura 15: Los módulos inalámbricos BISM II y WISM

El módulo WISM es un dispositivo Wi-Fi que implementa los estándares IEEE 802.11b y 802.11g, con un **pinout** totalmente compatible con el del módulo BISM II. De esta manera sería factible reemplazar la Unidad Inalámbrica para que, en vez de ser un módulo Bluetooth el encargado de las comunicaciones inalámbricas en el subsistema hardware, lo sea un módulo Wi-Fi, debiendo modificar el programa cargado en la Unidad de Control. Por otro lado, ambos módulos (WISM y BISM II) han sido diseñados para poder coexistir en el mismo hardware, es decir, que se podría realizar un rediseño del subsistema hardware propuesto para que tuviese la capacidad de usar las dos tecnologías inalámbricas, Bluetooth y Wi-Fi, de forma simultánea.

3.1.3.- La Interfaz Analógica

La Interfaz Analógica es la encargada de definir la conexión entre las etapas de acondicionamiento de las diversas señales biológicas y nuestro subsistema hardware. Proporciona un modelo de patillaje de entrada/salida del dispositivo, con la finalidad de definir un estándar, el cual debe ser tenido en cuenta a la hora de diseñar las etapas de acondicionamiento de señal que quieran ser conectadas a nuestro dispositivo.

En la siguiente figura (*Figura 16*) se puede observar el diagrama de despliegue relativo a la conexión de diversas etapas de acondicionamiento con nuestro dispositivo hardware a través de su interfaz analógica.

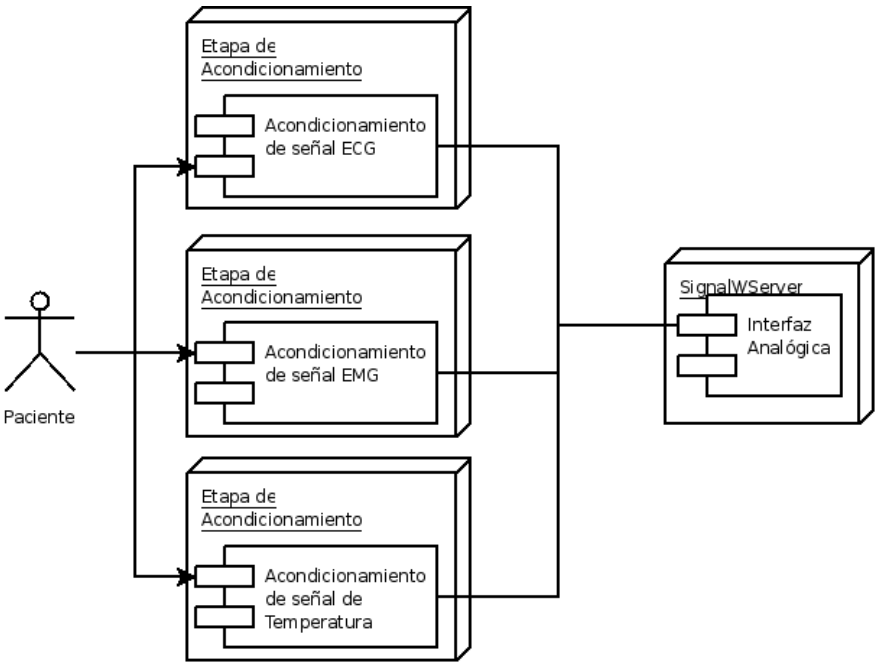

Figura 16: Diagrama de conexión de varias etapas de adquisición de señal.

Las alternativas disponibles para el pinout de la interfaz analógica son las siguientes:

- Basado en alimentación distribuida: cada etapa de adquisición y acondicionamiento de la señal lleva su propia fuente de alimentación.
- Basado en alimentación centralizada: cada etapa de adquisición y acondicionamiento tomará la alimentación del dispositivo hardware.

Se ha optado por usar un modelo basado en alimentación centralizada debido a que proporciona la ventaja de que facilita el diseño de las etapas de adquisición de señal y minimiza el tamaño de las mismas.

El pinout que se ha definido para la Interfaz Analógica se puede ver resumido en la siguiente tabla (Tabla 4).

Nombre del Pin	Descripción	Dirección[6]
AN0..AN9	Puertos de señal analógica acondicionada	Entrada
VCC_+9V	Tensión continua de +9 Voltios con respecto de masa	Salida
VCC_-9V	Tensión continua de -9 Voltios con respecto de masa	Salida
VCC_5V	Tensión continua de +5 Voltios con respecto de masa	Salida
VCC_3V3	Tensión continua de +3.3 Voltios con respecto de masa	Salida
GND	Masa	Salida

Tabla 4: Modelo de pinout de la Interfaz Analógica.

La elección de dicha configuración conlleva, además, la ventaja de que se puede usar un modelo de conexión basado en alimentación distribuida, siempre y cuando la señal resultante de una etapa de acondicionamiento sea referenciada con respecto de la masa de la interfaz analógica.

3.1.4.- La Interfaz de Administración

La Interfaz de Administración sirve para que un usuario pueda interactuar con el subsistema hardware a fin de consultar o modificar determinadas propiedades del dispositivo.

Para ello se ha seleccionado un interfaz de tipo RS-232, por las siguientes razones:

- La Unidad de Control seleccionada posee una UART implementada.
- Es factible introducirla en el hardware sin complicar su diseño.
- El manejo puede hacerse a través de un PC dotado de puerto RS-232 y usando cualquier terminal de puerto serie, como HyperTerminal, Eltima Software o Hercules.
- El manejo, desde el punto de vista del microcontrolador, es más sencillo que el de un USB, además los dispositivos que cumplen los requerimientos para la Unidad de Control que disponen de 1 UART y 1 USB en lugar de 2 UART, tienen una frecuencia máxima de trabajo de 12 MIPS, la cual es menor que los 30 MIPS que pueden conseguirse con el seleccionado actualmente.

3.2.- Subsistema software: estudio de alternativas

En cuanto al subsistema software, la decisión más relevante que hubo que tomar fue en relación al lenguaje de programación que iba a ser utilizado para desarrollar la aplicación software.

6 Dirección con respecto del dispositivo hardware diseñado.

Dentro de los lenguajes que cumplen el requisito de que la aplicación resultante sea multiplataforma, se pueden encontrar dos vertientes. Por un lado los lenguajes interpretados como “Perl”, “Ruby” o “Python”, por otro lado los lenguajes basados en entornos comunes de ejecución como “Java” o los pertenecientes a la plataforma “.Net”.

De todos estos casos se ha elegido Java por ser un lenguaje muy extendido y por la existencia de entornos de ejecución para una amplia variedad de sistemas.

Otra de las ventajas de Java es que ofrece librerías gráficas multiplataforma excelentes y muy potentes para el desarrollo de aplicaciones de escritorio (Swing y AWT). Además está altamente orientado a trabajo en red, el cual es un requisito importante de nuestro subsistema software.

En caso de querer trabajar con bases de datos, Java proporciona conectores JDBC para una gran cantidad de éstas, lo que le da una gran flexibilidad en cuanto a persistencia de datos se refiere.

Por último, otro de los motivos por el que se eligió este lenguaje es porque se dispone de muy buenas herramientas, como el IDE NetBeans, para facilitar el trabajo del programador, además se tiene una amplia experiencia en la implementación de aplicaciones con Java, lo que nos da como resultado un menor tiempo de desarrollo y una mayor calidad del software.

3.3.- Metodología de desarrollo

Para la realización del sistema se ha seguido una metodología de desarrollo en espiral *(Figura 17)*, el cual es un modelo de ciclo de vida en el que el producto se obtiene a partir de realizar múltiples iteraciones sobre un proceso de desarrollo. Cada iteración de la espiral representa una serie de actividades que hacen crecer el sistema, a raíz de nuevas funcionalidades y mejoras, y/o solucionan partes del sistema, provenientes de iteraciones anteriores cuyo resultado no ha sido satisfactorio.

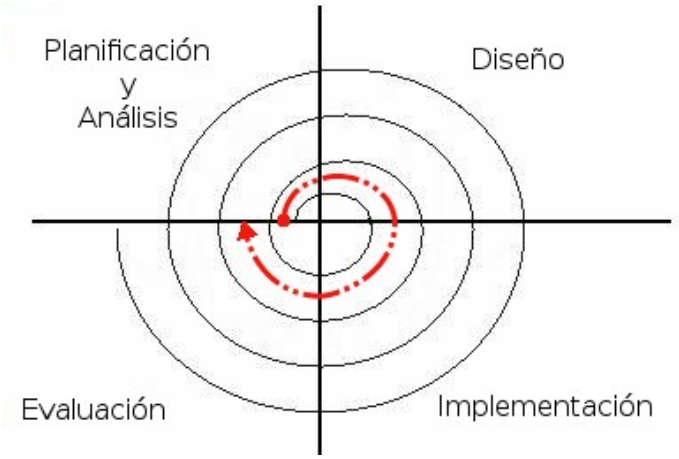

Figura 17: Modelo de desarrollo en espiral.

Las fases que componen dicho modelo de desarrollo son las siguientes:

- Planificación y análisis: Se obtienen los requisitos que se necesitan del sistema, bien mediante reuniones con los tutores del proyecto, o bien siendo definidos por el propio diseñador. Se establece cuáles de los requisitos obtenidos van a ser tenidos en cuenta para la presente iteración.
- Diseño: Para cada requisito seleccionado, se estudian las alternativas posibles y se evalúan las restricciones, así como su factibilidad. Se realiza un diseño de la solución que se adopta.
- Implementación: Se desarrolla la solución diseñada.
- Evaluación: Se prueba la solución implementada y se valoran/verifican los resultados.

En el *Anexo VIII* se pueden ver detalladas las tareas realizadas para el presente proyecto y la duración de las mismas, así como su diagrama de GANTT asociado.

4.- Descripción del Sistema

4.1.- Visión general

Se ha diseñado y realizado un sistema que sirve para monitorizar en tiempo real un máximo de 10 señales biológicas de distintos tipos. La monitorización se lleva a cabo a través de un dispositivo hardware, al que llamaremos “SignalWServer”, el cual digitaliza las señales y las envía por una conexión inalámbrica Bluetooth a una aplicación instalada en un PC. Dicha aplicación, a la que denominaremos “BioSignal”, se encarga de recibir los datos relativos a las señales que le envía el dispositivo SignalWServer y representarlas gráficamente en la interfaz de usuario.

El dispositivo SignalWServer está dotado de un puerto serie a través del cual se puede configurar la frecuencia de muestreo y el número de canales que el dispositivo va a transmitir (*ver 4.2.4*).

La aplicación BioSignal es capaz de actuar como servidor de señales, retransmitiendo a través de la red los datos que recibe de un dispositivo SignalWServer. Por otro lado, el mismo software BioSignal permite actuar de cliente en estas situaciones, conectándose a otra instancia de la aplicación BioSignal que esté configurada como servidor de señales.

Además, la aplicación BioSignal posibilita la capturar de las señales recibidas, bien a un fichero del disco duro, seleccionado por el usuario, o bien a una base de datos HSQLDB embebida en la aplicación, proporcionando métodos al usuario para iniciar o parar una captura en el momento que desee. Por otro lado la aplicación también permite la visualización de las señales capturadas en cualquiera de los dos medios y realizar labores de gestión de la base de datos embebida, así como importar y/o exportar los datos de señal de ésta.

En la siguiente figura *(Figura 18)* se puede observar el diagrama de despliegue que representa los nodos y componentes del sistema.

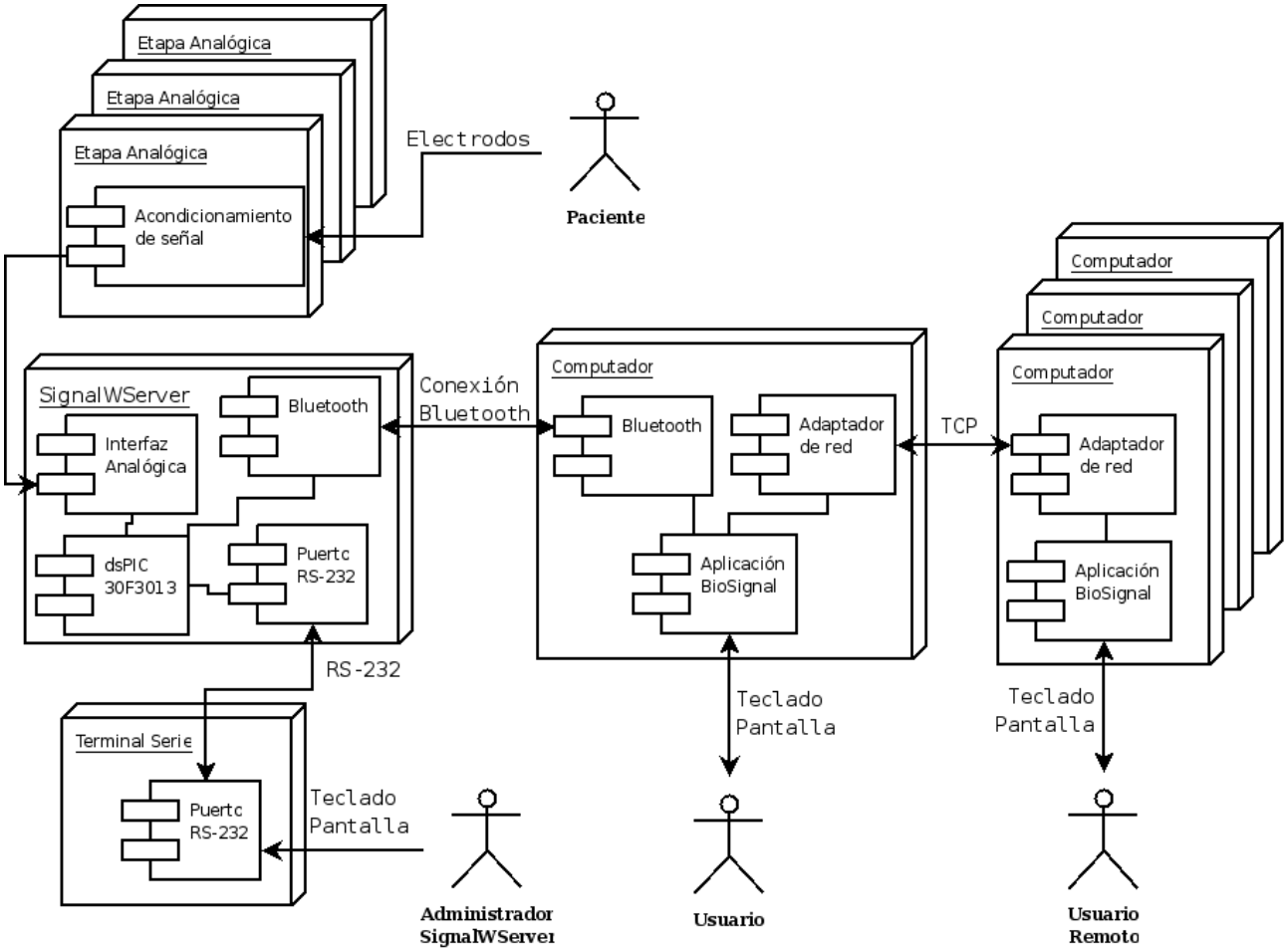

Figura 18: Diagrama de despliegue del sistema.

Para profundizar más en las características del sistema desarrollado vamos a dividir su descripción en las siguientes secciones:

- **El Subsistema Hardware:** donde se describe el dispositivo hardware SignalWServer.
- **El Subsistema Software:** donde se describe la aplicación BioSignal.
- **Comunicación entre Subsistemas:** donde se describen ciertos aspectos de la comunicación entre subsistemas.

4.2.- El Subsistema Hardware

Como ya se ha mencionado, al dispositivo que implementa el subsistema hardware lo hemos denominado "SignalWServer" *(Figura 19)*, que proviene de "*Signal Wireless Server*", es decir, servidor inalámbrico de señales.

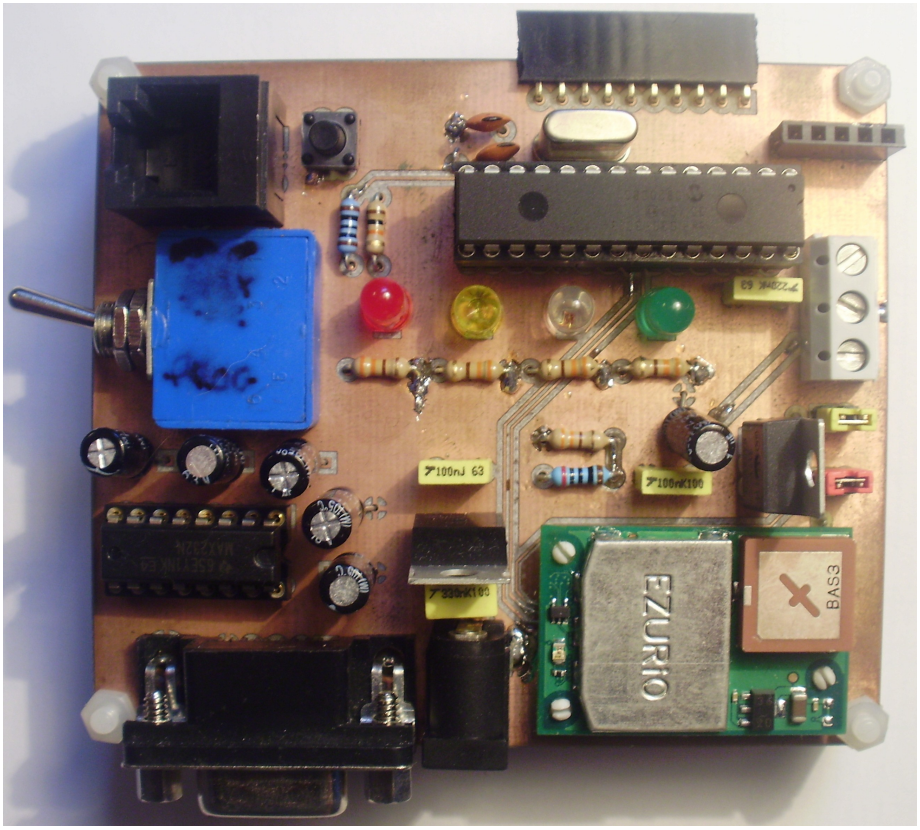

Figura 19: El dispositivo SignalWServer.

En lineas generales, el dispositivo SignalWServer se encarga de esperar conexiones de clientes BioSignal y, cuando una conexión ha sido establecida, de realizar el envío de los datos de señal procedentes de cada una de las entradas de su interfaz analógica. Por otro lado el dispositivo dispone de una consola de administración por puerto serie, la cual permite configurar la frecuencia de muestreo de los canales de señal y el número de canales a ser monitorizados.

En las siguientes figuras podemos observar los distintos bloques *(Figura 20)* que forman el SignalWServer y su diagrama de conexiones *(Figura 21)*.

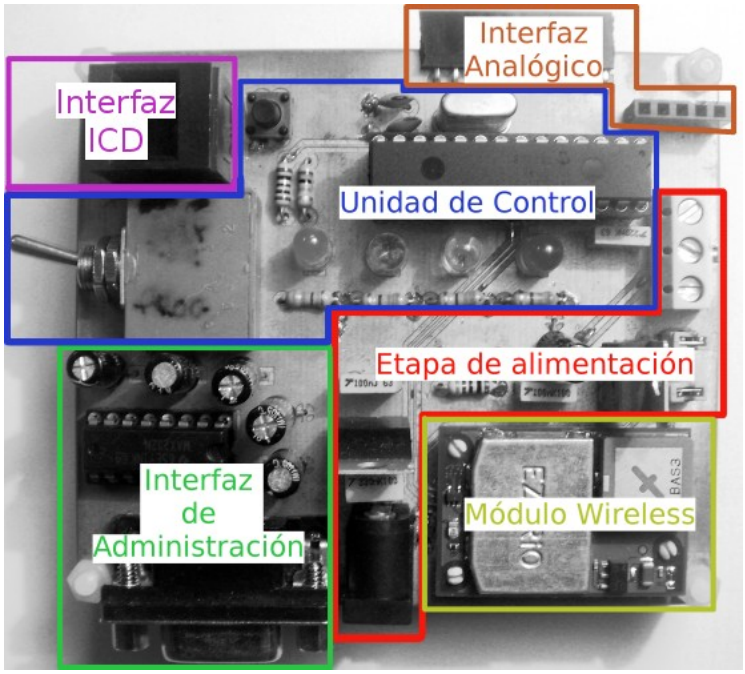

Figura 20: Bloques del dispositivo SignalWServer.

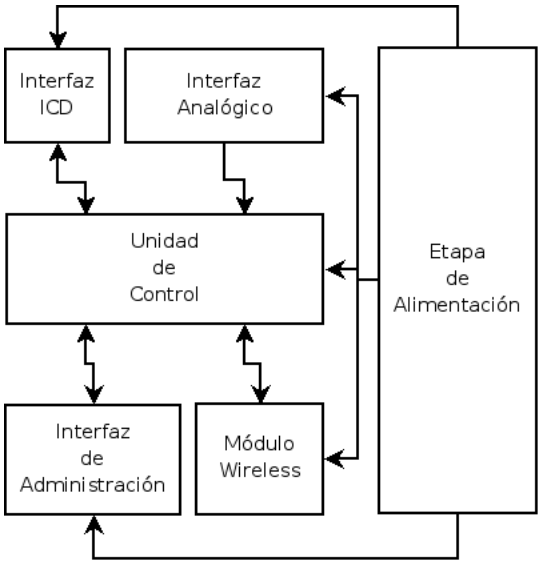

Figura 21: Diagrama de conexiones del dispositivo SignalWServer.

4.2.1.- La etapa de alimentación

El dispositivo dispone de dos entradas de alimentación que pueden ser utilizadas, la primera de ellas es una clema de tres vías *(Figura 23)*, la cual está preparada para que se le conecte una fuente de alimentación que le proporcione +9 Voltios, GND y -9 Voltios. Está orientada a que se pueda alimentar el dispositivo mediante baterías portátiles o pilas de 9V con la siguiente configuración *(Figura 22)*.

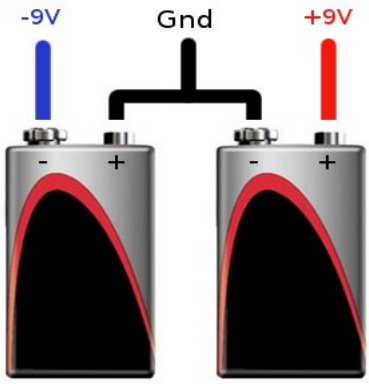

Figura 22: Configuración de la alimentación mediante pilas de 9V.

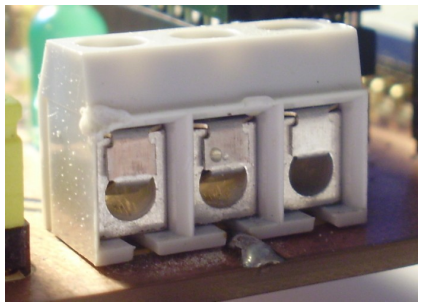

Figura 23: Clema para baterías del dispositivo.

Por otro lado se dispone también de una entrada de alimentación formada por un conector hembra de tipo cilíndrico *(Figura 25)*, también llamado "*hollowplug*", el cual está diseñado para que se le conecte un adaptador de corriente AC/DC que proporcione 9 Voltios de tensión continua. Ha sido diseñado para que se permita la configuración del dispositivo hardware, a través del interfaz de administración, sin necesidad de conectarle baterías. En este caso el exterior del conector macho del adaptador de corriente debe ser la masa del mismo mientras que el interior debe ser el polo de +9V *(Figura 24)*.

Figura 24: Conector macho hollowplug de adaptador AC/DC.

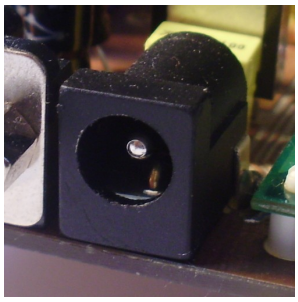

Figura 25: Conector hembra hollowplug del dispositivo.

Las tensiones con las que se deben alimentar los distintos bloques del dispositivo hardware se pueden ver reflejadas en la siguiente tabla *(Tabla 5)*.

Bloque	Tensiones de alimentación
Unidad de control	+3,3 V
Módulo wireless	+5 V; +3,3 V
Interfaz de administración	+5 V
Interfaz analógica	+9 V; -9 V[7]; +5 V; +3,3 V
Interfaz ICD	+5 V

Tabla 5: Tensiones de alimentación de los distintos módulos.

El conjunto de estas tensiones se consigue gracias a dos reguladores de tensión alimentados con 9 Voltios. Por un lado el LM317 [LM1] *(Figura 27)*, con el que se consigue una tensión de 3,3 V y por otro lado el LM7805 [LM2] *(Figura 28)*, gracias al que se consigue la tensión de 5 V. El resto de tensiones necesarias se consiguen directamente de la fuente de alimentación del circuito.

7 En caso de usar un adaptador de corriente para proporcionar alimentación al dispositivo se debe tener en cuenta que no se dispondrá de la salida de -9V en la interfaz analógica. (*ver 4.2.2*)

A continuación se muestran las partes del esquema del circuito referentes a la etapa de alimentación *(Figuras 26, 27 y 28).*

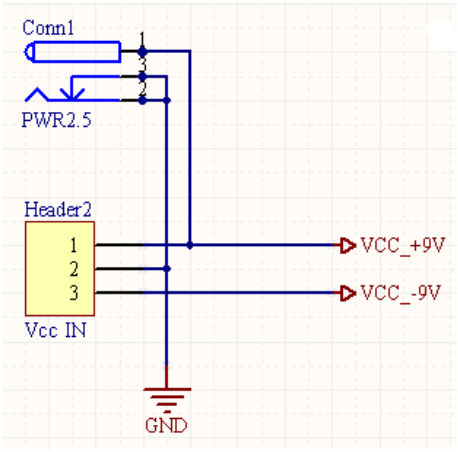

Figura 26: Esquema de las entradas de alimentación.

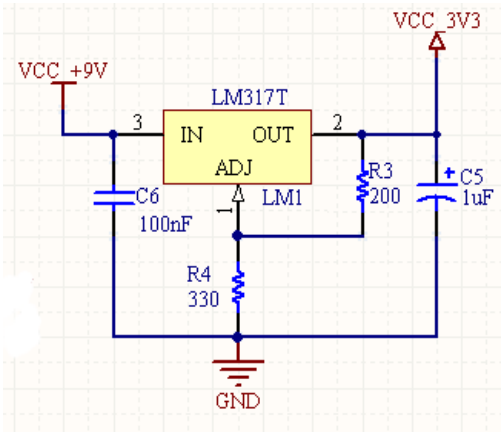

Figura 27: Esquema de conexión del regulador LM317.

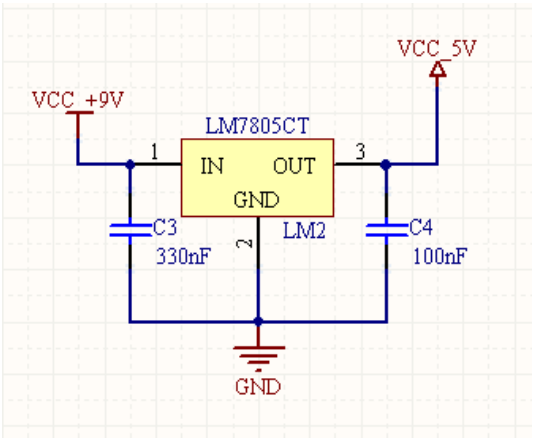

Figura 28: Esquema de conexión del regulador LM7805.

4.2.2.- La interfaz analógica

La interfaz analógica proporciona un pinout común que debe ser respetado por las diversas etapas de acondicionamiento de señal que quieran ser conectadas a nuestro dispositivo. Está formado por dos hileras planas de conectores hembra que proporcionan mayor flexibilidad a la hora de conectarle cualquier dispositivo. Aunque esta solución no es quizá óptima desde el punto de vista de la transmisión de señales analógicas, se ha considerado suficiente para este prototipo.

En la primera hilera, de 5 vías, se tiene las salidas de alimentación del circuito, es decir todas las tensiones que se manejan en el dispositivo SignalWServer pueden ser accedidas desde dicha hilera. La razón de su existencia es que las etapas de acondicionamiento de señal no estén obligadas a

disponer de alimentación propia y la puedan obtener del propio dispositivo. Por ello existe un conjunto de tensiones de ±9V, puesto que los amplificadores operacionales y de instrumentación que suelen manejar las etapas de acondicionamiento de señal se pueden alimentar con dicho conjunto de tensiones.

En la segunda hilera, de 10 vías, se encuentran las conexiones para las entradas analógicas que irán directamente al modulo de control para ser digitalizadas. Es aquí donde se adquieren las señales que provienen de los módulos de acondicionamiento.

En la siguiente imagen *(Figura 29)* se puede observar la interfaz analógica, la hilera de la izquierda es la de las salidas de alimentación y la de la derecha es la correspondiente a las entradas analógicas.

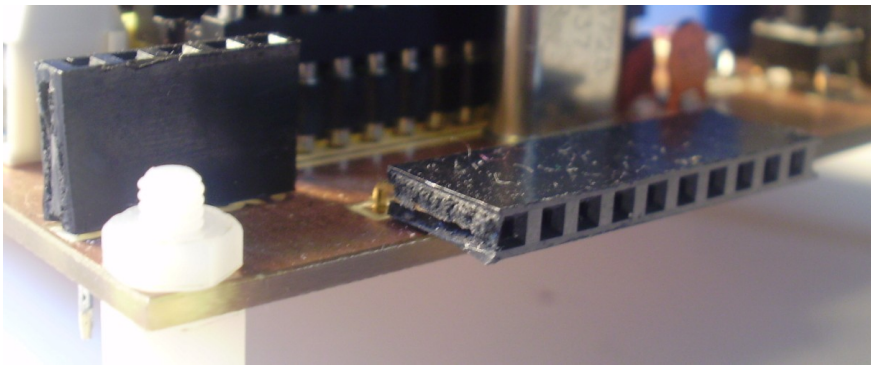

Figura 29: La interfaz analógica en SignalWServer

4.2.2.1.- Las etapas de acondicionamiento de señal

Aunque el presente proyecto no tiene como finalidad desarrollar ninguna etapa de acondicionamiento de señal, es necesario que definamos unos aspectos relevantes de la construcción de las mismas.

Debido a que existen diversas señales biológicas que deben ser acondicionadas de maneras diferentes, decidimos que nuestro dispositivo no debía integrar un módulo de acondicionamiento de señales debido a que le restaría flexibilidad, es decir, que no permitiría la monitorización nada más que de las señales para las que hubiera sido diseñado el mencionado módulo. De este modo se decidió que las etapas de acondicionamiento de señal fueran externas a nuestro dispositivo y éste debía proporcionar una interfaz adecuada para que cualquiera pudiese diseñar sus propios módulos de acondicionamiento.

Para diseñar un módulo de acondicionamiento de señal válido se deben cumplir dos restricciones: la primera es que **la tensión de salida** del módulo de acondicionamiento debe estar en la escala [0V .. 3,3V], dado que estos límites son, respectivamente, V_{REF-} y V_{REF+} del ADC (*ver 1.4*) de nuestra unidad de control. Tensiones fuera de dicha escala saturarían el ADC, dando una lectura no fiable y que incluso podrían dañar el dispositivo. La segunda es que la **impedancia máxima de salida** del módulo de acondicionamiento debe ser siempre inferior a 2.5 KΩ[MCHP4].

4.2.3.- La unidad de control

La unidad de control es el centro de procesamiento del dispositivo SignalWServer y está implementada a través de un dsPIC30F3013 (*ver 3.1.1*). Su funcionamiento básico consiste en esperar peticiones de establecimiento de vínculos inalámbricos por parte de clientes BioSignal y, una vez establecida una conexión, enviar datos relativos a las señales que muestrea de las entradas de señal que provienen del interfaz analógico.

4.2.3.1.- El hardware de la unidad de control

En la siguiente figura *(Figura 30)* se puede observar la parte del esquemático *(ver Anexo VI)* referente a la unidad de control.

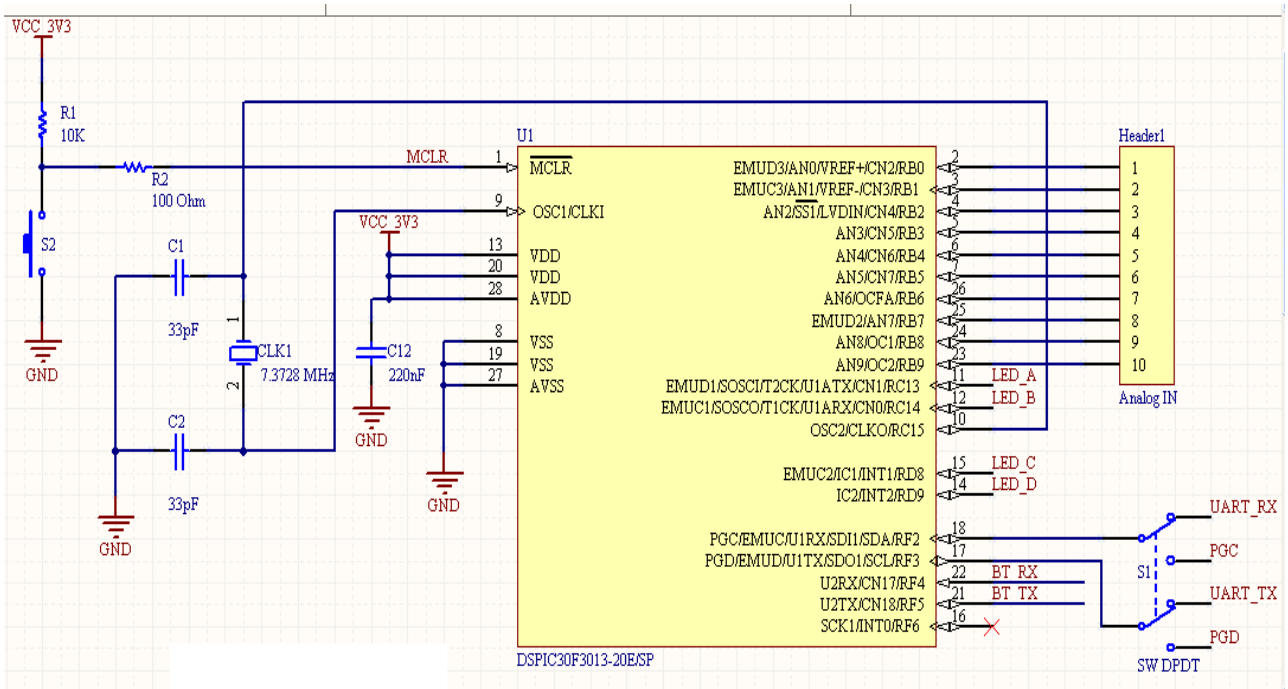

Figura 30: Esquema de conexión de la unidad de control.

Como podemos observar, el microcontrolador dsPIC30F3013 está conectado a un reloj externo de 7.3728 MHz. Aunque este modelo de microcontrolador dispone de un oscilador interno (FRC) el

cual trabaja a una frecuencia de 7.37 MHz, muy similar a la elegida, dicho FRC tiene un error del ±2%, por lo que se decidió conectar un reloj más exacto puesto que necesitamos una fiabilidad muy buena en el puerto serie que conecta la unidad de control con la unidad inalámbrica, debido a que la UART de ésta es muy sensible a la tasa de error de la velocidad de transcepción. Por otro lado este dispositivo dispone de un multiplicador de frecuencia de reloj (PLL) interno, el cual se puede configurar para que aumente la frecuencia del oscilador externo en un factor x4, x8 o x16.

En este caso, al usar un reloj externo de 7.3728 MHz (F_{OSC}) y configurar el PLL para que aumente dicha frecuencia en un factor x16, la frecuencia real de oscilador (F_R) es:

$$F_R = F_{OSC} \cdot PLL = 7.3728 \cdot 16 = \mathbf{117.9648\ MHz}$$

Dado que una instrucción tarda en ejecutarse 4 ciclos de reloj, podemos calcular que la frecuencia de trabajo (F_{CY}) es:

$$F_{CY} = \frac{F_R}{4} = \frac{117.9648}{4} = \mathbf{29.4912\ MIPS}$$

Que es una frecuencia de trabajo muy cercana a la máxima de 30 MIPS que se puede llegar a conseguir para este microcontrolador (*ver 3.1.1*).

Como podemos ver en el esquema anterior *(Figura 30)*, el dispositivo está dotado de dos UART. La primera de ellas ha sido conectada a un switch doble (“S1” en el esquemático) que permite la multiplexación manual entre la conexión hacia la interfaz ICD y la conexión hacia la interfaz de administración. La segunda, sin embargo, está directamente conectada con el módulo wireless y será la que permita la comunicación con el mismo.

Por otro lado el circuito también consta de la conexión a las entradas de señal de la interfaz analógica (“Header1”, en el esquema), un circuito de reset implementado mediante el pulsador “S2” y cuatro conexiones a sendos diodos LED (“LED_A”, “LED_B”, “LED_C” y “LED_D”, en el esquema) que permitirán comunicar al usuario el estado del dispositivo mediante señales luminosas. La localización de estos componentes en el prototipo SignalWServer se puede ver en la siguiente imagen *(Figura 31).*

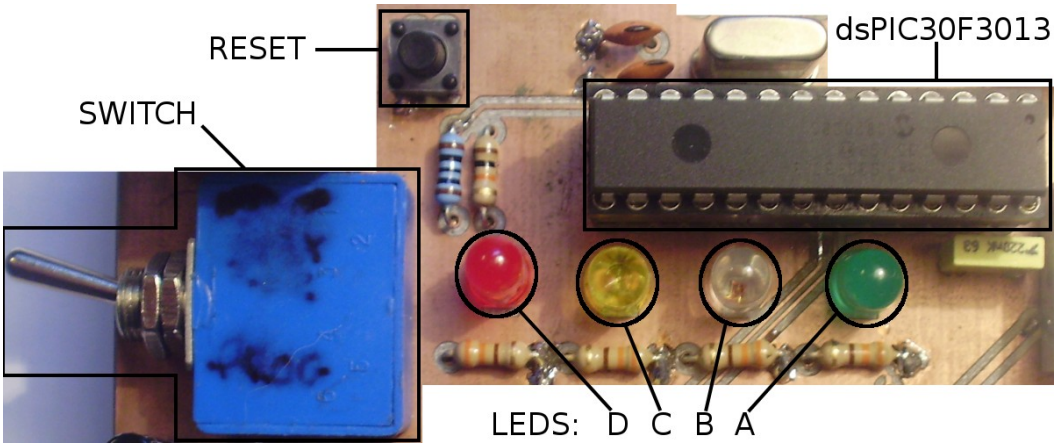

Figura 31: La unidad de control en el dispositivo SignalWServer.

4.2.3.2.- El firmware de la unidad de control

Como hemos mencionado anteriormente, para que la unidad de control funcione, el microcontrolador de su núcleo, el dsPIC30F3013, debe tener cargado un firmware el cual se encarga de las siguientes funciones:

- Configurar el modulo wireless.
- Aceptar peticiones de conexión/desconexión.
- Digitalización de los datos de señales provistos por la interfaz analógica.
- Envío de los datos de señal digitalizados a través de una conexión establecida.
- Permitir la configuración del dispositivo mediante el interfaz de administración.

El lenguaje que ha sido utilizado para programar el dsPIC30F3013 ha sido C, y para poder traducir la sintaxis de dicho lenguaje a código máquina, entendible por el microcontrolador, hemos usado el compilador CCS C a través del IDE MPLAB de microchip.

Para aprovechar al máximo las posibilidades del microcontrolador de hace un uso intensivo de las interrupciones. Una **interrupción** es un evento o un tipo de situación que provoca que el microcontrolador corte el flujo normal de ejecución del programa principal. Sin entrar en demasiado detalle, cuando se produce una interrupción, el microcontrolador guarda el estado de ejecución en una pila y salta a una dirección de la memoria de programa en la cual, a su vez, se salta a una **rutina de tratamiento de interrupción** (RTI). Una RTI es una función o bloque de código que cuando ha terminado de ejecutarse hace que se restaure el estado de ejecución que se tenía guardado en la pila, de manera que el flujo del programa principal sigue con normalidad *(Figura 32).*

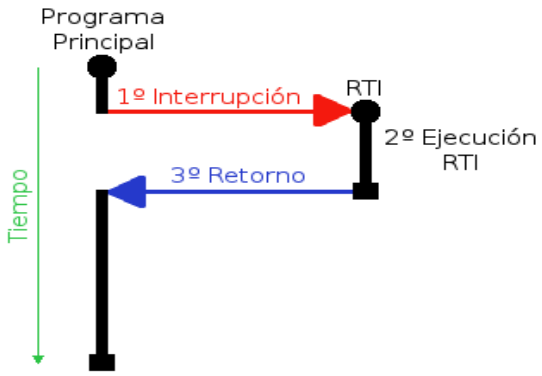

Figura 32: Modo de actuación de las interrupciones.

El dsPIC30F3013 dispone de hasta 21 tipos de interrupciones diferentes, de las cuales han sido utilizadas las que se enumeran en la siguiente tabla (Tabla 6).

Nombre	Evento
U1RX	Recepción de dato en la UART 1
U1TX	UART 1 lista para transmitir
U2RX	Recepción de dato en la UART 2
U2TX	UART 2 lista para transmitir
T1	Desbordamiento del TIMER1
ADC	Conversión lista en el ADC

Tabla 6: Interrupciones del dsPIC30F3013 utilizadas.

A continuación mostramos los diagramas de flujo que describen el comportamiento del firmware. La leyenda utilizada en éstos es la siguiente *(Figura 33).*

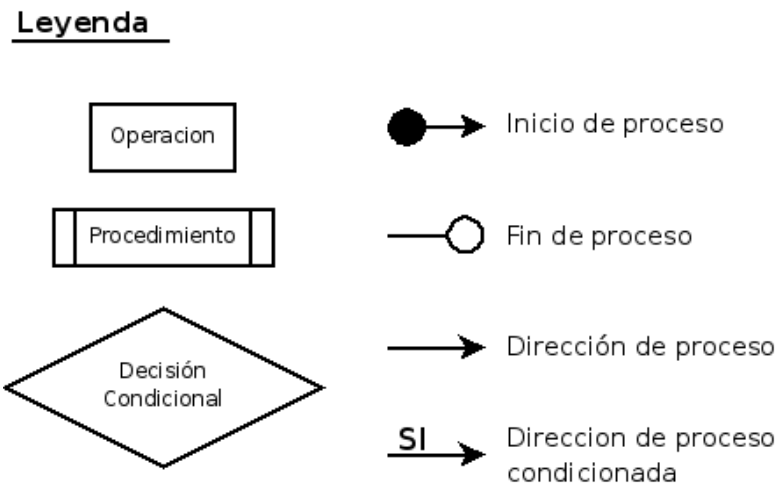

Figura 33: Leyenda de los diagramas de flujo.

El diagrama de flujo correspondiente al programa principal es el siguiente *(Figura 34)*.

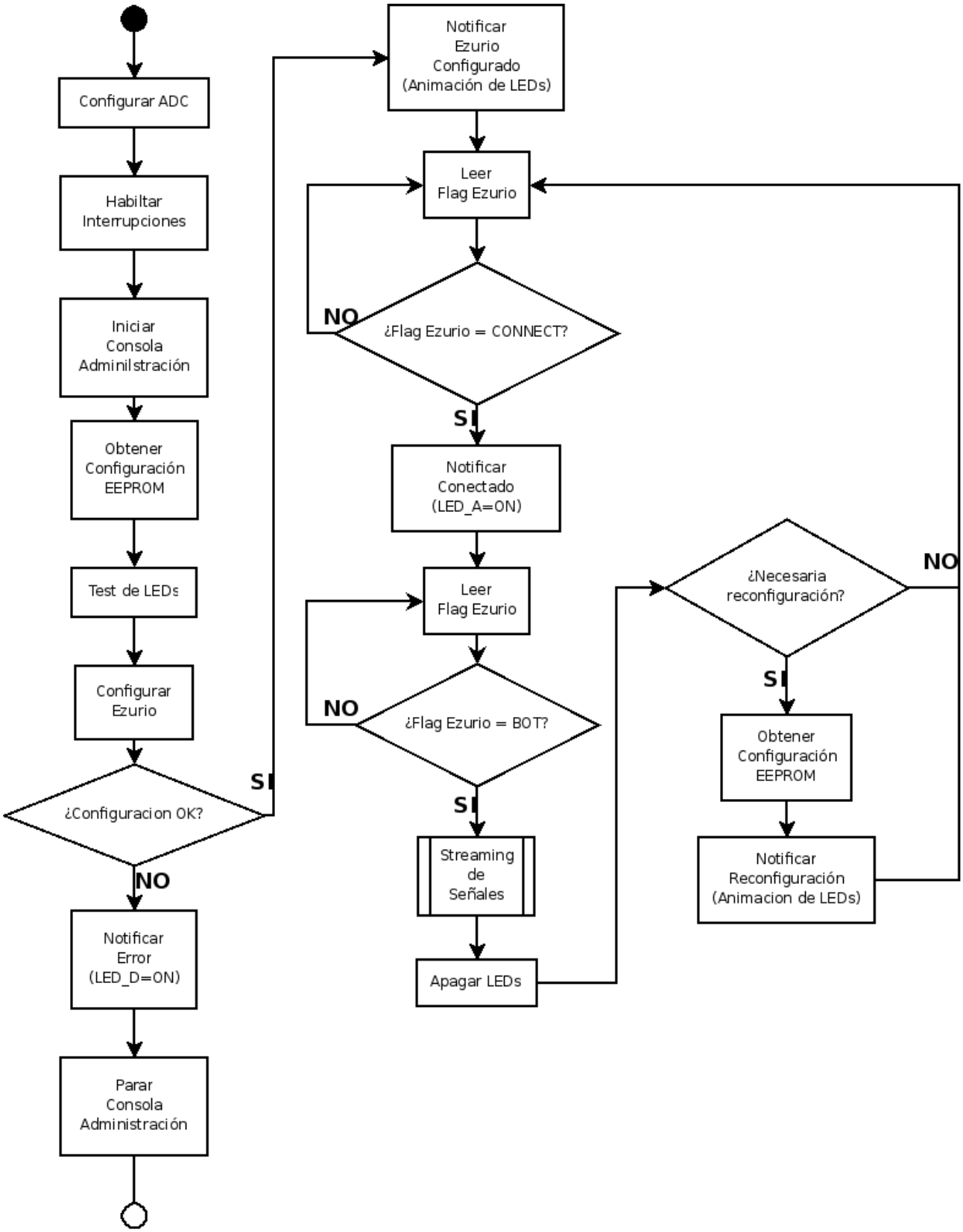

Figura 34: Diagrama de flujo del programa principal del firmware.

El diagrama de flujo del procedimiento “Streaming de Señales” es el siguiente *(Figura 35).*

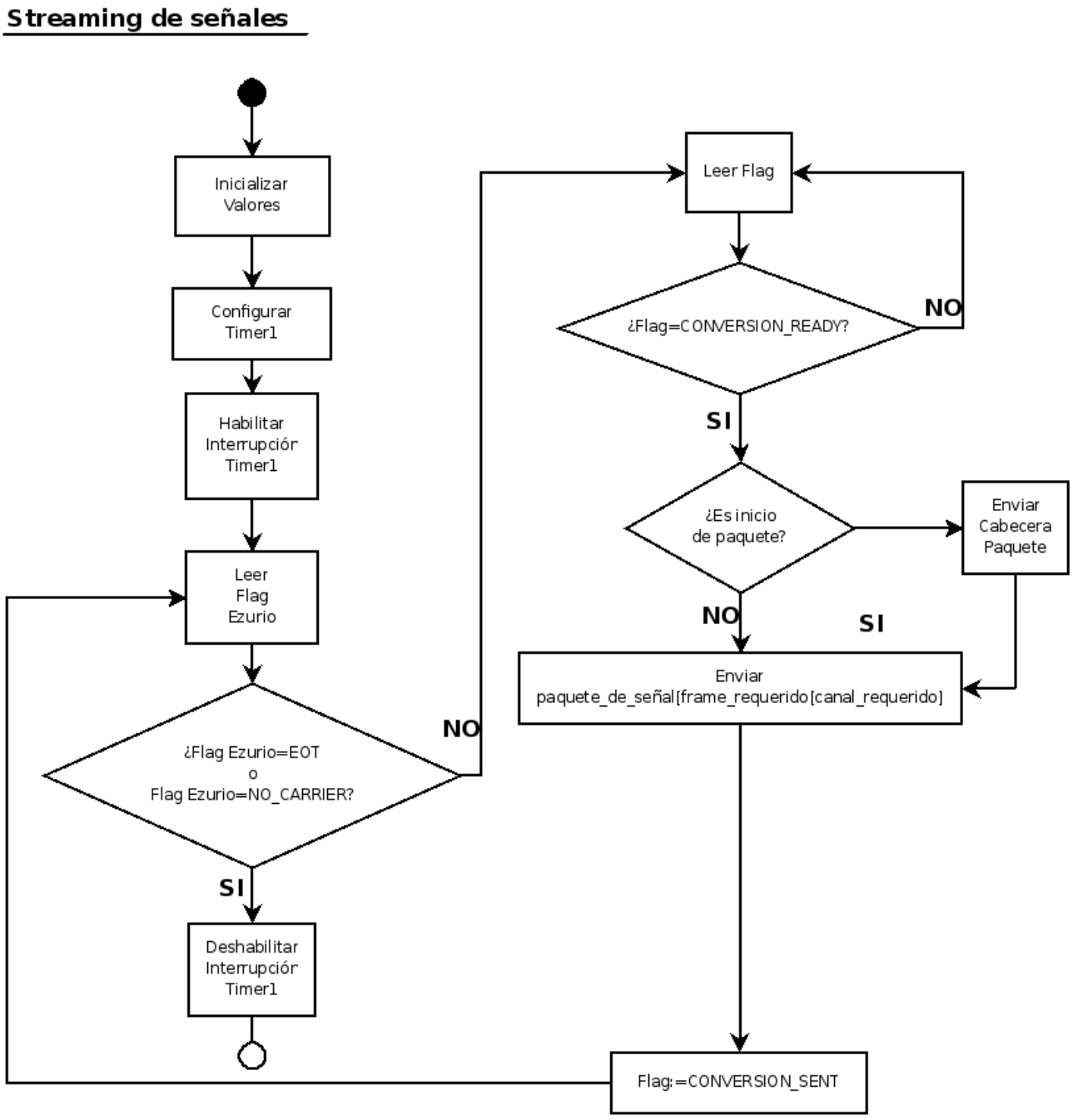

Figura 35: Diagrama de flujo del procedimiento “Streaming de Señales”.

El diagrama de flujo de la RTI llamada por la interrupción T1 se muestra a continuación *(Figura 36).*

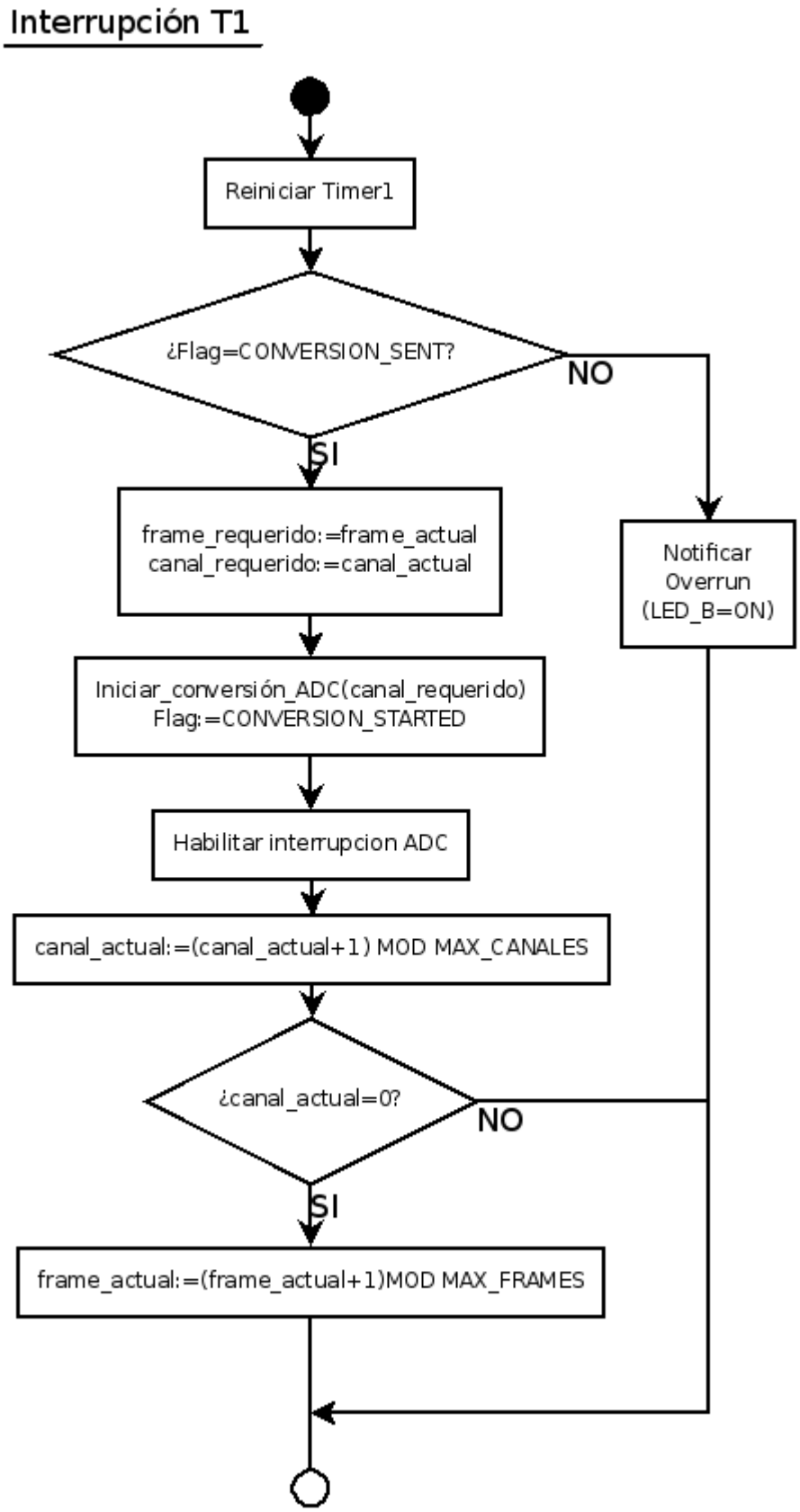

Figura 36: Diagrama de flujo de la RTI de la interrupción T1.

El diagrama de flujo de la RTI llamada por la interrupción ADC es la siguiente *(Figura 37)*.

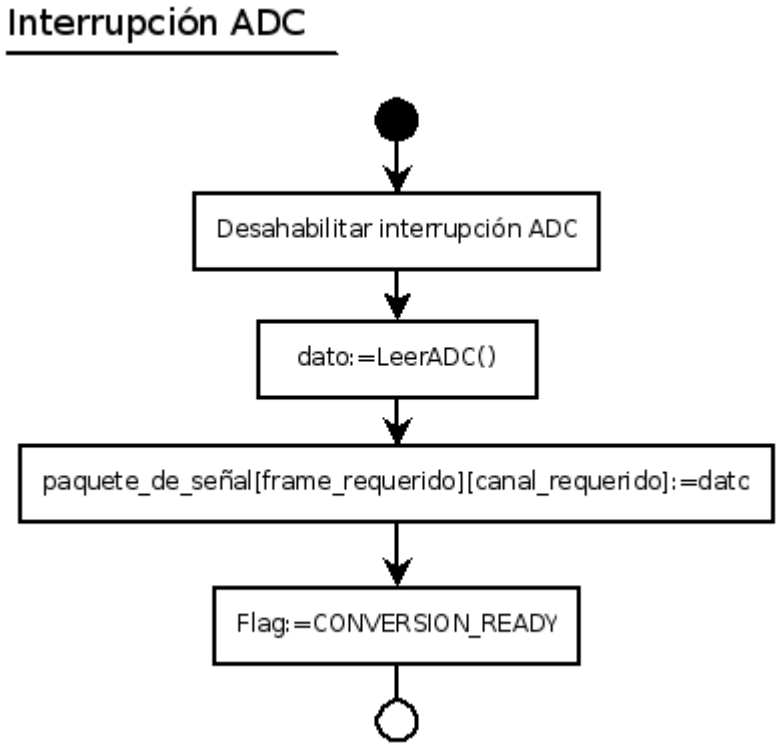

Figura 37: Diagrama de flujo de la RTI de la interrupción ADC.

A continuación se muestra el diagrama de flujo de la RTI llamada por la interrupción U1RX *(Figura 38).*

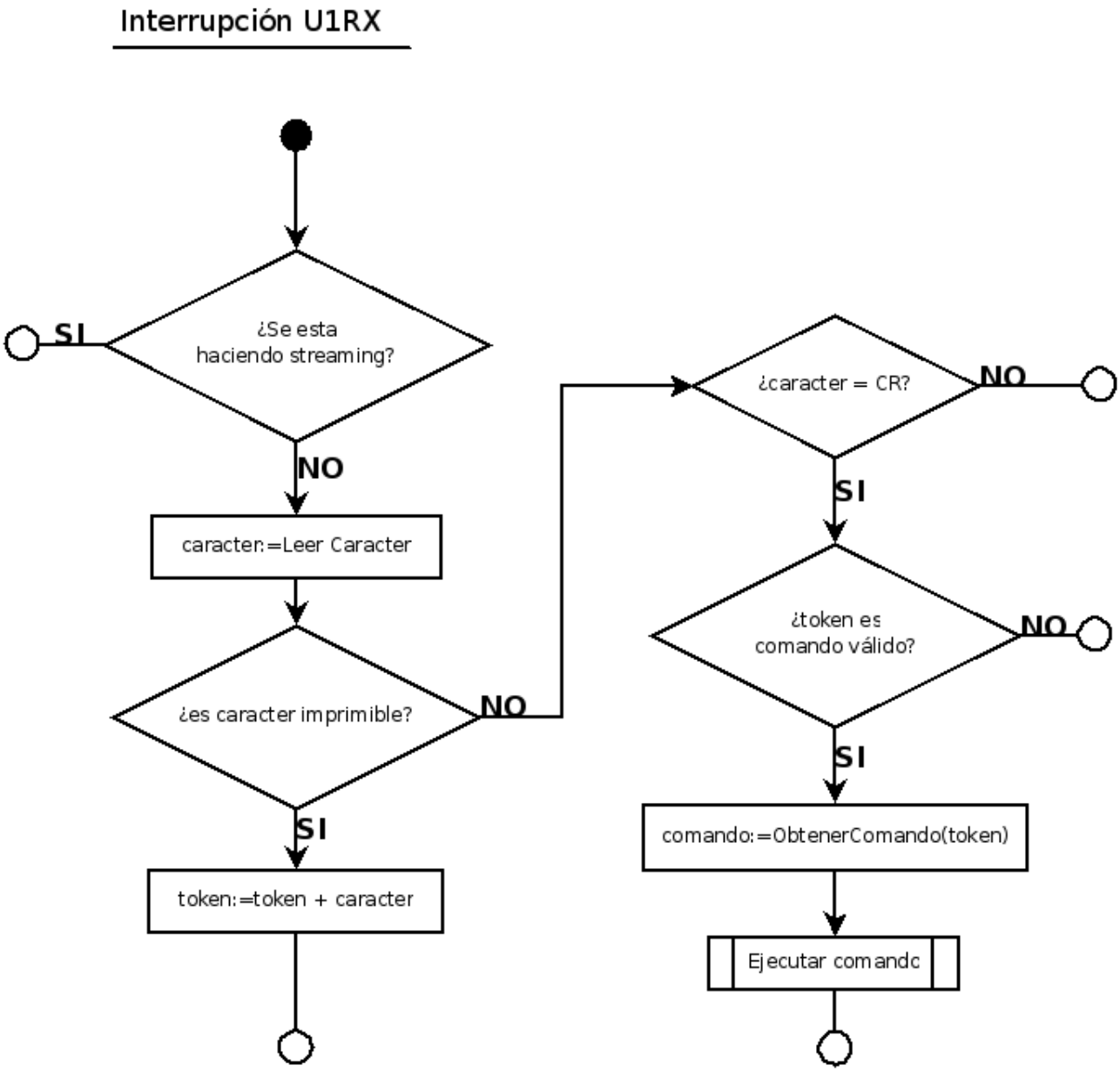

Figura 38: Diagrama de flujo de la RTI de la interrupción U1RX.

En este caso, el procedimiento “Ejecutar comando” selecciona un comando válido de una lista de comandos disponibles y ejecuta las acciones pertinentes (*ver 4.2.4*). En caso de haber sido introducido un comando inválido se muestra un error por la interfaz de administración.

El diagrama de flujo de la RTI llamada por la interrupción U2RX es el siguiente (*Figura 39*).

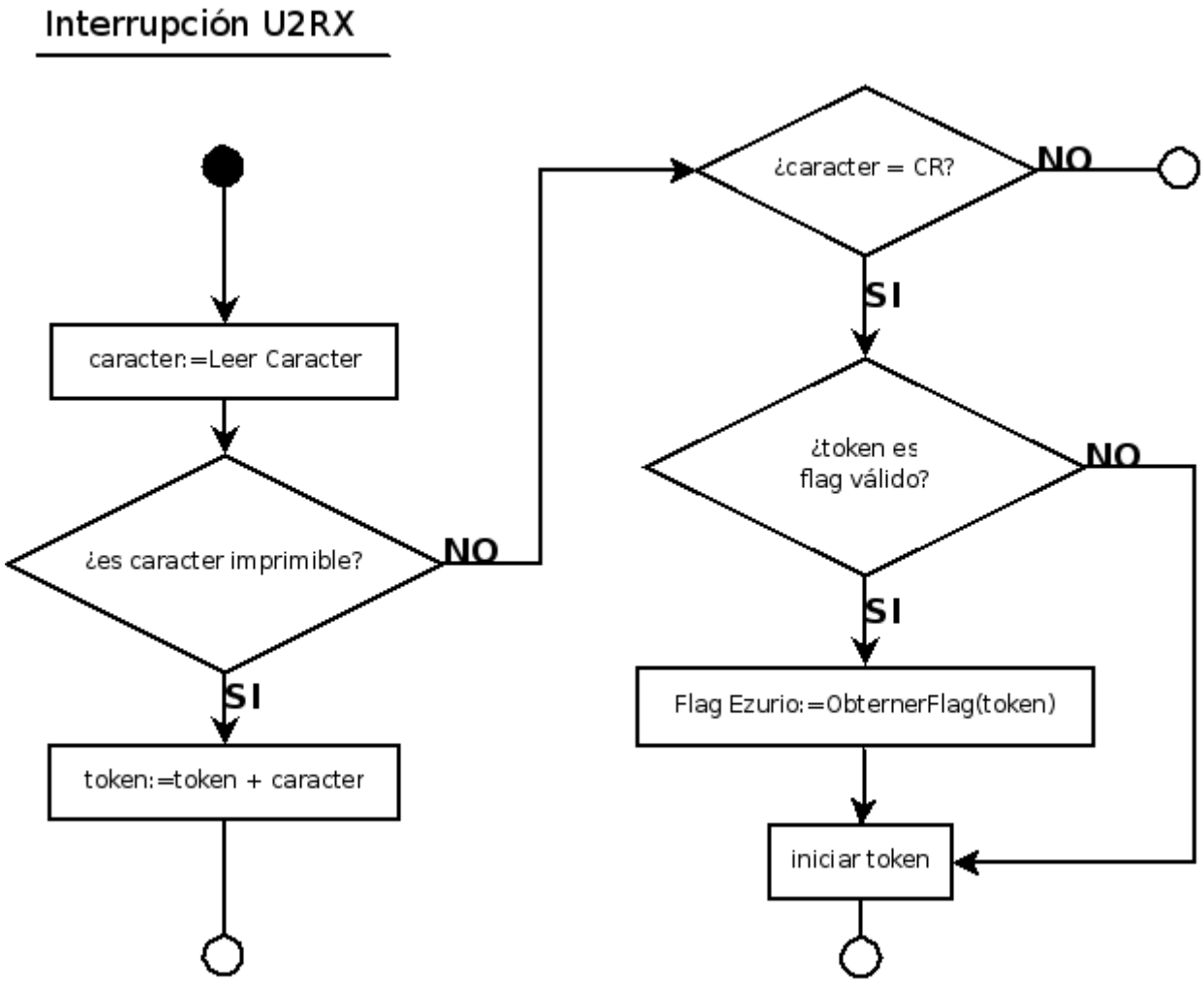

Figura 39: Diagrama de flujo de la RTI de la interrupción U2RX.

Por último, las RTI de las interrupciones U1TX y U2TX, han sido utilizadas para realizar escrituras no bloqueantes a través de búfferes de salida. Cuando el programa desea escribir en una de las dos UART, simplemente introduce el dato en el buffer de salida pertinente y, por medio de interrupciones, cuando la UART involucrada esté lista para la escritura, leerá los datos del buffer y los escribirá en la UART. En el caso de que el buffer de salida estuviera lleno al realizarse una escritura el programa se bloqueará y avisará al usuario de dicha circunstancia mediante un LED (en concreto con el LED_C). Por otro lado si el buffer de salida está vacío cuando la UART está preparada, simplemente se desactivará la interrupción correspondiente. Este proceso descrito se puede ver reflejado en el siguiente diagrama de flujo *(Figura 40).*

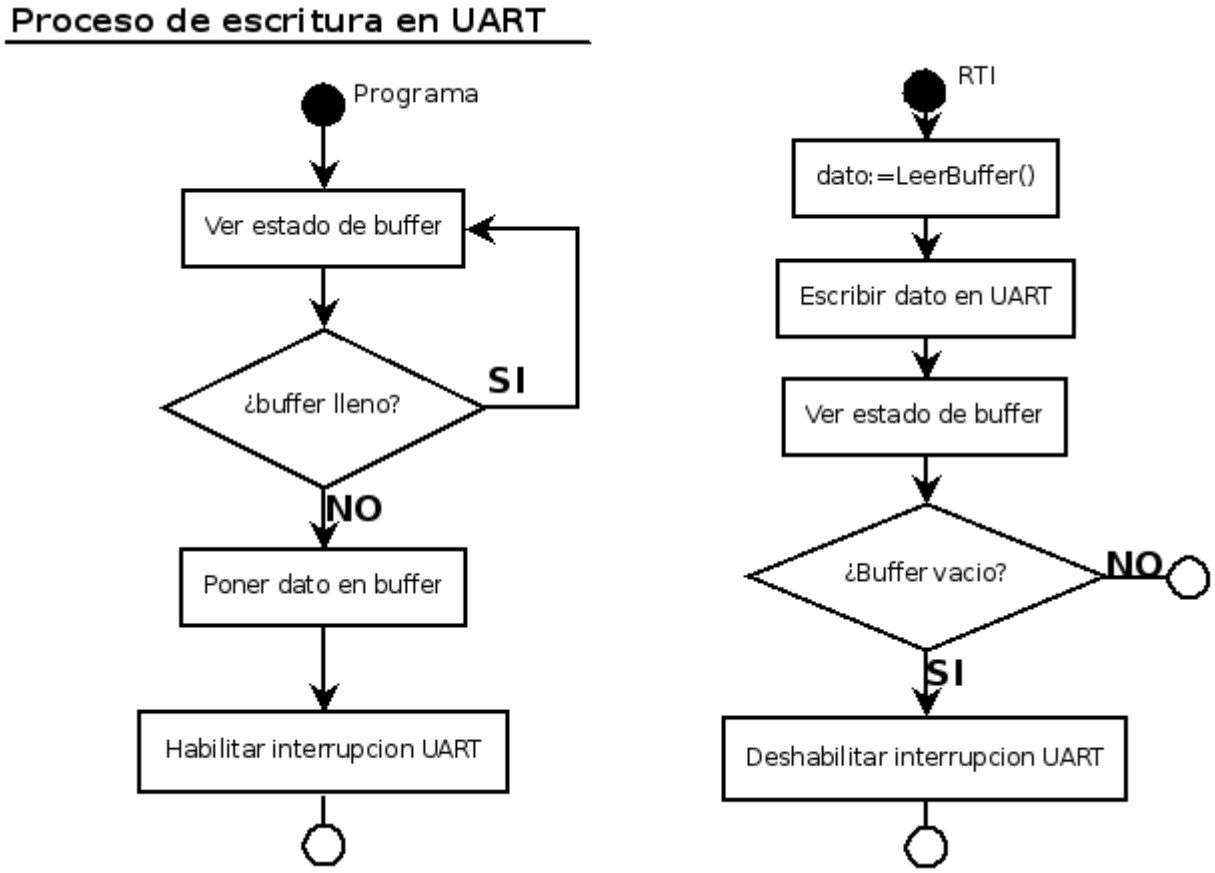

Figura 40: Proceso de escritura en UART mediante las interrupciones U1TX y U2TX.

4.2.4.- La interfaz de administración

La interfaz de administración permite la configuración de determinados parámetros del dispositivo SignalWServer, tales como la frecuencia de muestreo o el número de canales de la interfaz analógica que serán digitalizados.

Se trata de un conector DB9 *(Figura 41)* cuya finalidad es la de conectarse a la interfaz RS232 de un PC a través de un cable serie.

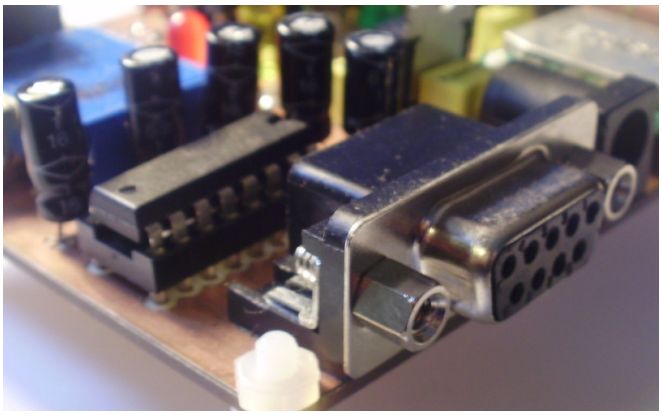

Figura 41: Conector DB9 hembra de la interfaz de administración.

Para poder hacer uso de la interfaz de administración se utilizará un programa de tipo "**consola de puerto serie**", configurada a 115200 baudios en modo 8n1[8], que sirve para poder interactuar con un puerto serie del PC a través de su teclado y pantalla. Existen multitud de consolas de puerto serie, como por ejemplo "HyperTerminal" o "Hércules", para sistemas operativos Windows, o "Minicom" para sistemas operativos Linux.

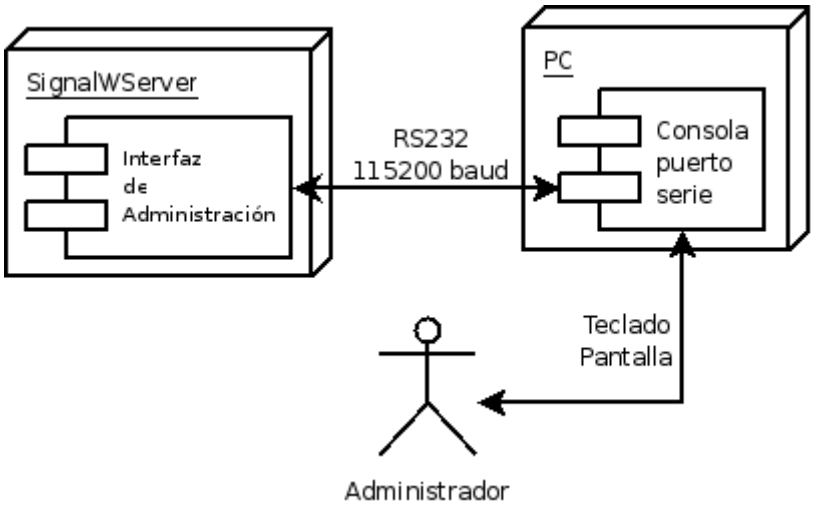

Figura 42: Conexión al interfaz de administración.

8 Con un bit de inicio, 8 bits de datos, un bit de parada y sin paridad.

De esta manera podremos administrar nuestro dispositivo a través de una especie de **shell**, en la cual se pueden escribir comandos, los cuales ejecutarán acciones en SignalWServer.

A continuación *(Figura 43)* se muestra el esquema de conexiones de la interfaz de administración, cuyo núcleo es el integrado MAX232 [MAX]. Dicho integrado sirve para adaptar los niveles TTL, provistos por la unidad de control, a los del estándar definido por RS-232.

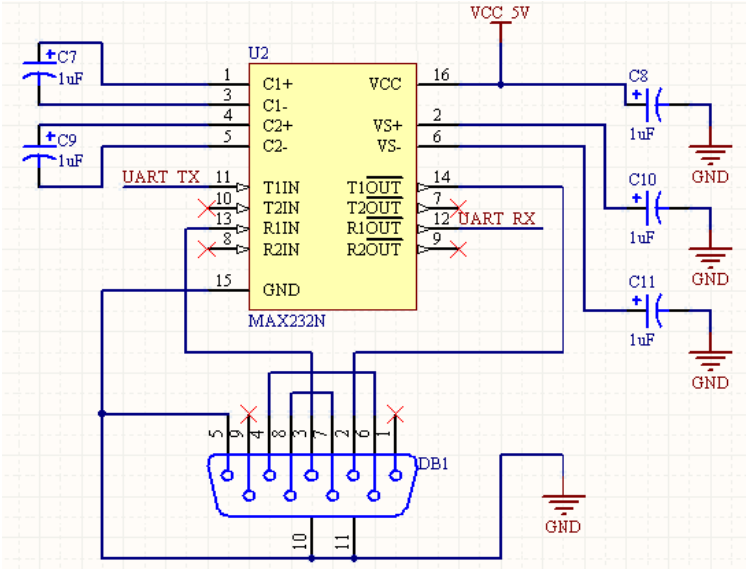

Figura 43: Esquema de conexión de la interfaz de administración

Los comandos que se pueden ejecutar son los siguiente:

- **HELP** : Muestra información de los comandos disponibles y su descripción.
- **GET PREESCALER**: Obtiene el valor de preescaler actualmente.
- **GET INI TIME**: Obtiene el valor de ini time actual.
- **GET CHANNELS**: Obtiene el numero de canales a digitalizar.
- **GET ALL**: Obtiene los valores de preescaler, ini time y numero de canales.
- **SET PREESCALER**: Actualiza el valor de preescaler.
- **SET INI TIME**: Actualiza el valor de ini time.
- **SET CHANNELS**: Actualiza el valor de numero de canales.
- **CALC INI TIME**: Calcula el valor de ini time en función de la frecuencia deseada y los valores actuales de preescaler y numero de canales.
- **CALC CHANNEL FREQ**: Calcula el valor de frecuencia por canal que se obtendría con los valores actuales.
- **LOAD DEFAULTS**: Salva los valores por defecto de todos los parámetros en memoria no volátil y los aplica.
- **SAVE**: Salva las modificaciones en memoria y aplica los cambios.

Todos los comandos son "*case insensitive*", es decir que no distinguen entre mayúsculas o minúsculas. En la consola los valores que pueden tomar PREESCALER, INI TIME y CHANNELS, pueden ser de dos tipos "*stored*" (salvado) o "*current*" (actual), donde el tipo "*stored*" se refiere a que el valor correspondiente está guardado en memoria no volátil[9] y es el que el dispositivo está aplicando, por el contrario, el tipo "*current*" indica que el valor todavía no ha sido guardado en memoria no volátil y no se hará hasta que se ejecute el comando SAVE. De este modo, los comandos "SET xxx" actualizan el valor "*current*" de "xxx" pero no lo guardan en memoria no volátil hasta ejecutar un SAVE. Por otro lado, los comandos de tipo "GET xxx" muestra en pantalla el valor "*stored*" de "xxx" y, en el caso de haber realizado un "SET xxx" previamente, también muestra el valor "*current*" de "xxx"[10].

Los valores PREESCALER e INI TIME sirven para configurar la **frecuencia de muestreo global** del sistema, siendo los que indican la frecuencia de llamada a la RTI de la interrupción T1. INI TIME indica con que valor se iniciará el Timer1 la primera vez y cada vez que se ejecute la RTI de la interrupción T1, por otro lado PREESCALER sirve para aumentar o decrementar en determinado factor, cada cuanto se incrementa el valor del Timer1 en función de la frecuencia de trabajo del microcontrolador.

Por otro lado el valor CHANNELS especifica la cantidad de canales que van a ser muestreados y debe estar en el margen [1 .. 10]. Si dividimos el valor de frecuencia de muestreo global por el numero de canales definido por CHANNELS obtendríamos el valor de **frecuencia de muestreo por canal**.

Dado que traducir un valor de frecuencia por canal en valores de PREESCALER, INI TIME y CHANNELS es un cálculo complejo, se proporcionan las herramientas CALC INI TIME y CALC CHANNEL FREQ.

CALC INI TIME pide al usuario un valor de frecuencia de muestreo por canal y calcula el valor de INI TIME que el usuario debe introducir mediante el comando SET INI TIME para lograr dicha frecuencia. El valor de INI TIME es calculado en función de los valores "*current*" de PREESCALER y CHANNEL.

9 La EEPROM del dsPIC30F3013.

10 En el caso de GET ALL, es análogo a ejecutar GET PREESCALER, GET INI TIME y GET CHANNELS de una vez.

CALC CHANNEL FREQ toma los valores "*current*" de PREESCALER, INI TIME y CHANNELS y muestra por pantalla la frecuencia de muestreo por canal que se obtiene[11].

De este modo, el proceso básico que debería seguir un administrador para la configuración del sistema mediante el interfaz de administración se muestra en el siguiente diagrama de secuencia *(Figura 44).*

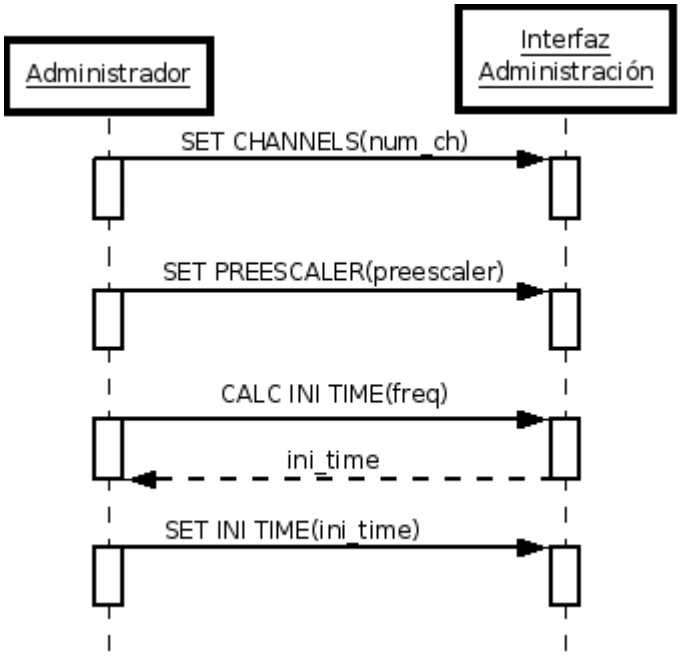

Figura 44: Diagrama de secuencia para la configuración de la frecuencia de muestreo.

4.2.5.- El módulo wireless

Como ya se ha mencionado anteriormente, el módulo utilizado para la comunicación mediante Bluetooth es el BISM II, de la marca Ezurio. Dicho módulo se conecta a la unidad de control a través de las UART que ambos tienen implementadas.

A continuación se puede ver el módulo Bluetooth utilizado (*Figura 46*) y su esquema de conexiones (*Figura 45*).

11 Para más información acerca de como se calcula la frecuencias de muestreo en base a los valores de PREESCALER, INI TIME y CHANNELS véase el *Anexo II*.

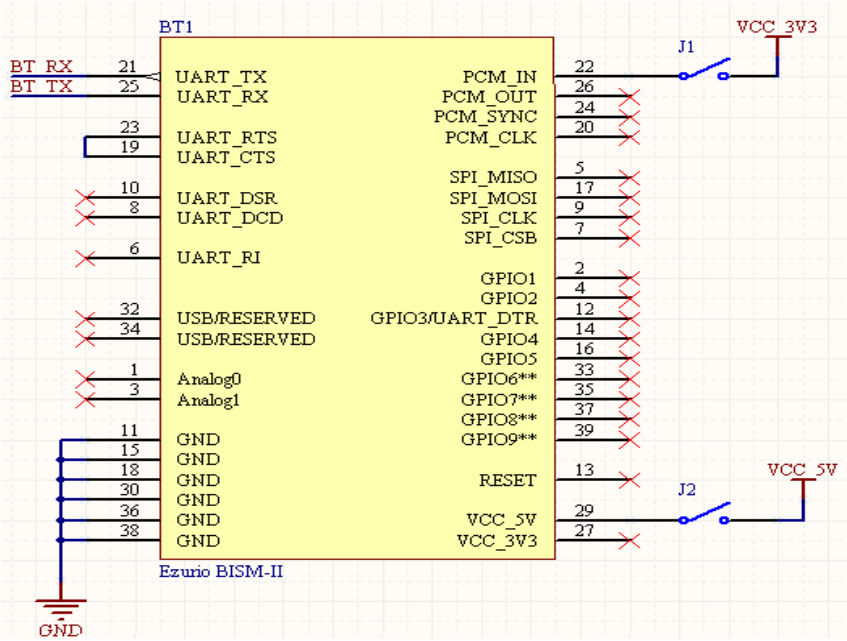

Figura 45: Esquema de conexión del módulo wireless

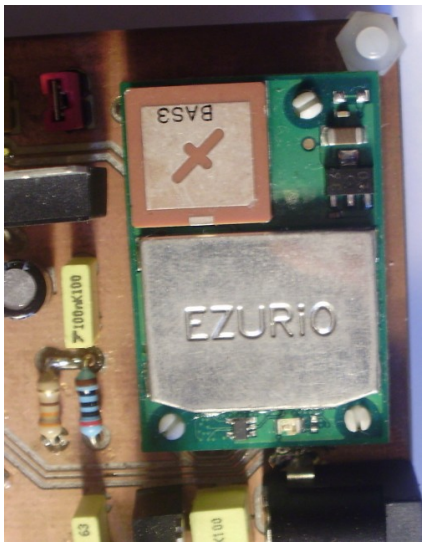

Figura 46: El módulo wireless en SignalWServer

El modo en el que se comunican la unidad de control y el módulo wireless es a través de la capa AT de la pila de protocolos Bluetooth, usando para ello comandos y respuestas AT. Los comandos AT consisten en cadenas de caracteres ASCII finalizadas por un carácter de control CR (retorno de carro) y generalmente encabezadas por los caracteres 'AT'. Cuando un comando AT es enviado al módulo wireless, éste contesta con una respuesta AT, que también consiste en una cadena de caracteres, delimitada a ambos lados por los caracteres de control CR y LF (nueva línea), la cual por norma general suele ser 'OK' , en caso de tener éxito en la ejecución del comando, o 'ERROR' seguido de un código de error, para aquellos comandos que por algún motivo no pudieron ser completados satisfactoriamente.

También existen respuestas AT no solicitadas, es decir, aquellas que el módulo wireless puede enviar pero que no son el resultado de la ejecución de un comando AT, si no que son enviadas por el módulo wireless cuando se produce algún evento de especial importancia. En particular se pueden encontrar respuestas no solicitadas de los siguientes tipos:

- **RING**: Es enviada cuando un dispositivo remoto está solicitando la creación de un vínculo inalámbrico con nuestro dispositivo local.

- **CONNECT**: Es enviada cuando se acepta una solicitud de vínculo inalámbrico y se establece una conexión inalámbrica con un dispositivo remoto. A partir de la recepción de este tipo de respuesta, el módulo wireless abandona el modo de comandos y se coloca en modo de datos, es decir, que todo lo que la unidad de control envía al módulo wireless será remitido al dispositivo remoto conectado.
- **NO CARRIER**: Es enviada cuando se ha abandonado una conexión previamente establecida, por lo que el módulo wireless abandona el modo de datos y vuelve a estar disponible para recibir comandos AT.

4.2.5.1.- Configuraciones previas del módulo wireless

Debido a que la comunicación entre la unidad de control y el módulo wireless se hace a través de sus respectivas UART, la velocidad de transcepción de ambas debe ser igual, dado que de no ser así ambos módulos no podrían comunicarse correctamente. Por ello es necesario que, previamente, la UART del módulo wireless sea configurada a 115200 baudios, 1 bit de inicio, 8 de datos y 1 bit de parada, sin paridad.

Para la configuración del módulo wireless se necesitará un circuito que posibilite la conexión entre un PC y el módulo wireless, permitiendo acceder a la UART del mismo. Para este propósito se puede utilizar un entrenador de la marca Ezurio [LAIRD2], el cual es un dispositivo al que se le puede conectar el módulo BISM II y proporciona una interfaz RS232 para que manipule el mismo, mediante comandos AT, a través de una consola de puerto serie[12] instalada en un PC.

El BISM II viene configurado de fábrica con la UART a 9600 baudio, con 1 bit de inicio, 8 de datos y 1 bit de parada, sin paridad, por lo que, para poder cambiar dicha configuración a la deseada habrá que introducirle los siguientes comandos AT[BISM2]:

$'\boldsymbol{ATS520=115200}' \rightarrow$ Velocidad: 115200 baudios
$'\boldsymbol{ATS523=1}' \rightarrow$ Bits de parada: 1
$'\boldsymbol{ATS524=0}' \rightarrow$ Sin paridad

4.2.5.2.- Configuraciones que se realizan del módulo wireless

Una de las responsabilidades del firmware embebido en la unidad de control es la de configurar el módulo wireless al inicio del proceso de ejecución. A continuación *(Tabla 7)* se detallan los comandos AT que se envían al módulo wireless para dicha configuración.

12 En la propia web de Ezurio se puede encontrar una consola para puerto serie de manera gratuita.

Comando AT	Descripción
AT&F	Pone el valor por defecto en los registros, a excepción de la configuración de velocidad de la UART.
ATE0	Deshabilita el eco en el dispositivo.
ATS0=1	Habilita el modo de conexión automático al recibir 1 petición de establecimiento de conexión (RING).
ATS512=4	Establece el modo del dispositivo como “descubrible” y “conectable”.
ATS102=1	Establece el perfil del dispositivo a SPP.
ATS502=0	Deshabilita la autenticación.
ATS12=100	Establece el tiempo (100 mseg) que tiene que transcurrir como mínimo entre caracteres '^' cuando se está en modo de datos para que se interprete como una petición de pasar a modo de comandos[13].
AT+BTS=”Biosignal”	Actualiza el nombre del servicio Bluetooth.
AT+BTN=”SignalWServer”	Actualiza el nombre del dispositivo Bluetooth.
AT&W	Salva el valor de los registro en memoria no-volátil.
ATZ	Reinicia el módulo con las configuraciones aplicadas.

Tabla 7: Comandos AT de configuración del módulo wireless.

4.2.6.- La interfaz ICD

La interfaz ICD del dispositivo SignalWServer consiste en un conector RJ-11 hembra (*Figura 47*) que permite conectar un programador/depurador para microcontroladores de la marca Microchip. Proporciona la ventaja de poder programar el microcontrolador “*on board*”, es decir, en la propia placa del circuito impreso, sin la necesidad de desconectarlo de la placa e introducirlo en un programador dedicado, como por ejemplo el PICSTART Plus [MCHP5]. Además tiene la ventaja de que, usando un hardware adecuado, se puede hacer uso del modo de depuración del código, permitiendo establecer puntos de ruptura (*breakpoints*) o realizar una depuración instrucción a instrucción, pudiendo ver en cada momento el estado de los registros y variables en tiempo de ejecución.

El interfaz se ha probado conectando el programador/depurador ICD3 [ICD] de la marca Microchip (*Figura 48*), el cual , a su vez, se conecta mediante un puerto USB a un PC y se integra con el IDE MPLAB, permitiendo la funcionalidad antes descrita.

13 Estando en modo de datos el dispositivo permite pasar a modo comandos enviando tres '^' separados por determinado tiempo.

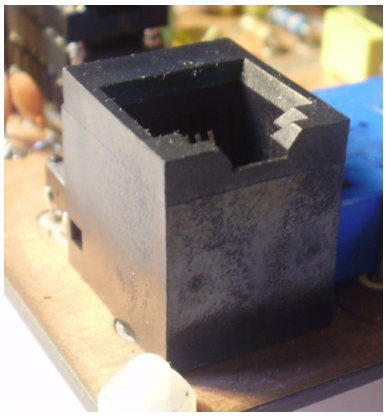

Figura 47: La interfaz ICD en SignalWServer

Figura 48: El programador/depurador ICD3.

Señalar que, debido a que la unidad de control se comunica con el interfaz ICD a través de la misma UART que con el interfaz de administración, para poder hacer uso de la programación del dispositivo se debe colocar de forma correcta el switch que multiplexa dicha UART (*ver 4.2.3.1*).

4.3.- El Subsistema Software

Como ya hemos mencionado en anteriores ocasiones, a la aplicación que implementa el subsistema software la hemos llamado “BioSignal”, que claramente hace referencia al término señal biológica. Es una aplicación escrita en Java, lo que permite que la aplicación sea portable entre sistemas sin realizar ningún esfuerzo añadido.

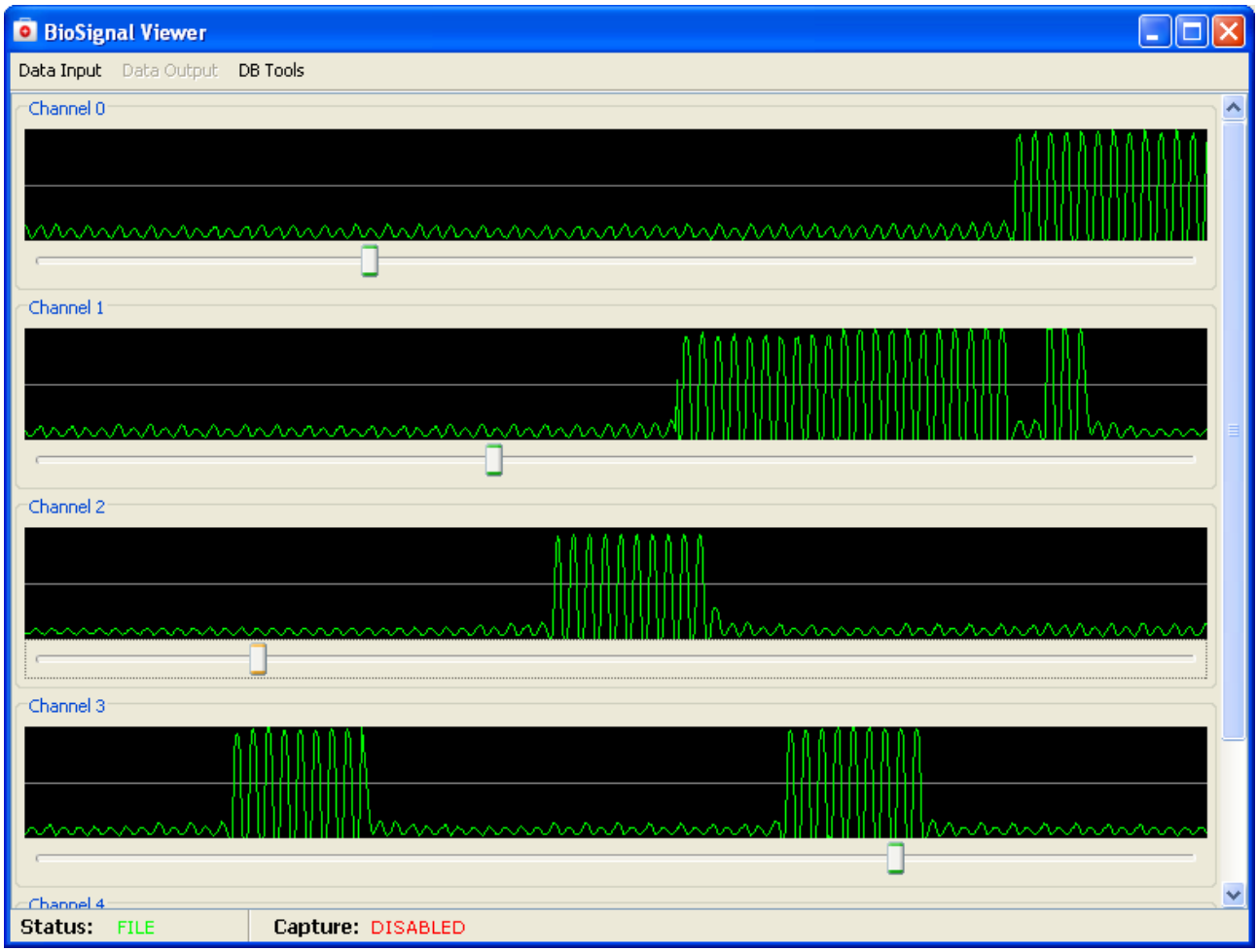

Figura 49: Ventana de la aplicación BioSignal.

La aplicación BioSignal es la responsable, principalmente, de obtener los datos de señal que captura un dispositivo SignalWServer de su interfaz analógica y mostrarlos de manera gráfica por pantalla. Además proporciona mecanismos para poder capturar señales a memoria no volátil, visualizar los datos de señal recuperándolos de capturas anteriores o actuar como servidor de señales, entre otros.

La funcionalidad de la aplicación se puede resumir a través del diagrama de casos de uso que se muestra en la siguiente figura *(Figura 50).*

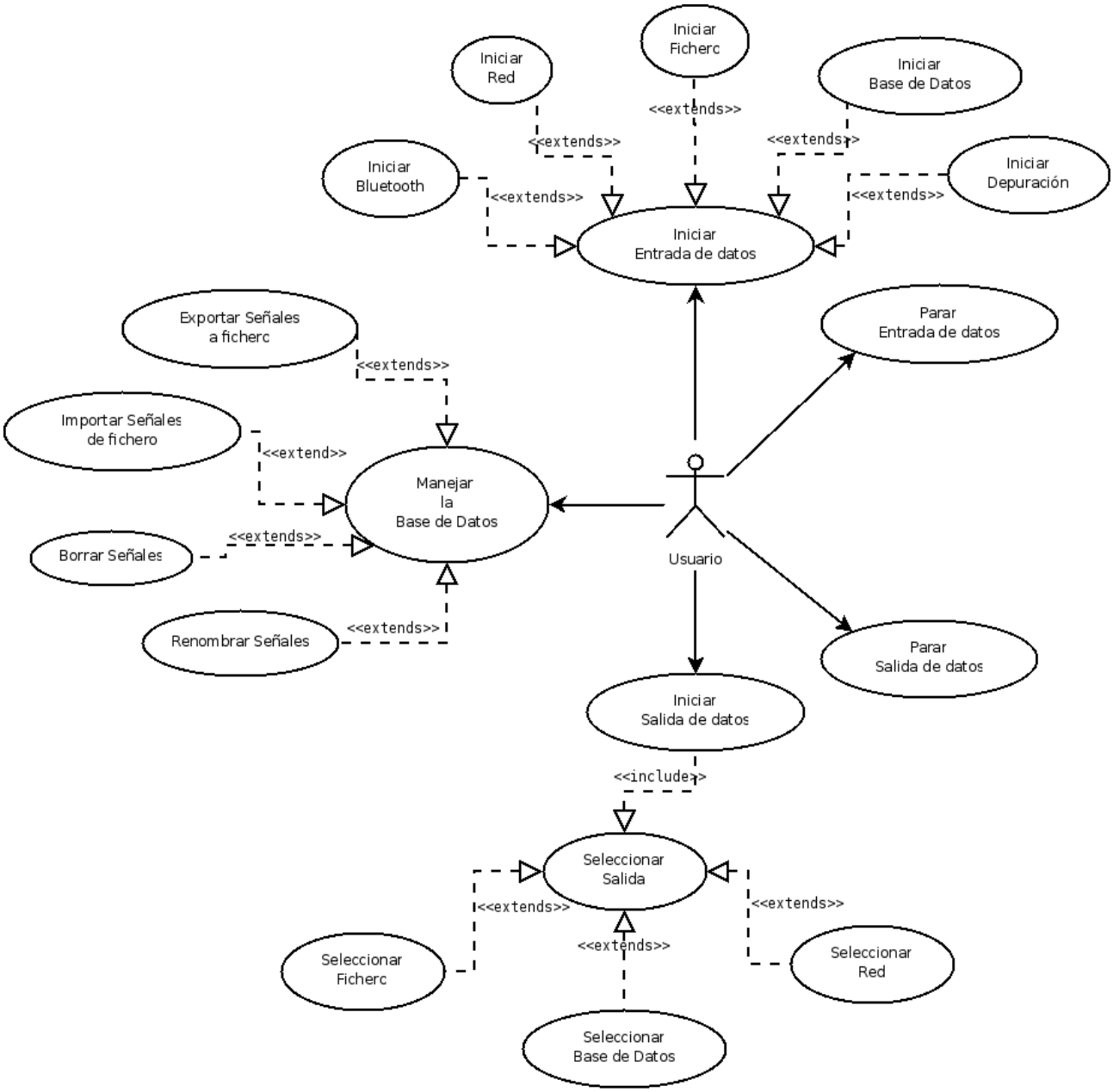

Figura 50: Diagrama de casos de uso de BioSignal.

4.3.1.- Iniciar entrada de datos

Para comenzar la recepción de datos por parte de la aplicación y representar la señales en pantalla se requiere que el usuario elija una entre una serie de fuentes de datos disponibles. Las fuentes de datos las podemos dividir en dos categorías en base a su naturaleza:

- Fuente de datos en "*streaming*": aquellas fuentes de datos que se comportan como un *stream*, es decir que no se conoce el momento de su finalización. Esto implica que en la representación gráfica de la señal no podremos navegar por los datos, si no que simplemente se mostrará la representación de la señal en el área de visualización a medida que la fuente vaya proporcionando datos de señal.

- Fuente de datos estática: son aquellas fuentes de las que se conoce la totalidad de los datos que son capaces de proporcionar y que, por tanto, pueden ser representadas gráficamente de forma íntegra.

Al primer grupo pertenecen las fuentes de datos procedentes de una conexión Bluetooth (tipo "*Bluetooth*"), una conexión de Red (tipo "*Network*") o de la entrada de Depuración (tipo "*Debug*"). Por otro lado, al grupo de las fuentes de datos estáticas pertenecen los ficheros almacenados en un soporte no volátil (tipo "*File*") y las señales almacenadas en la base de datos HSQLDB embebida (tipo "*Database*").

El modo en que el usuario puede seleccionar la fuente de datos deseada es a través del menú "Data Input" de la aplicación *(Figura 51).*

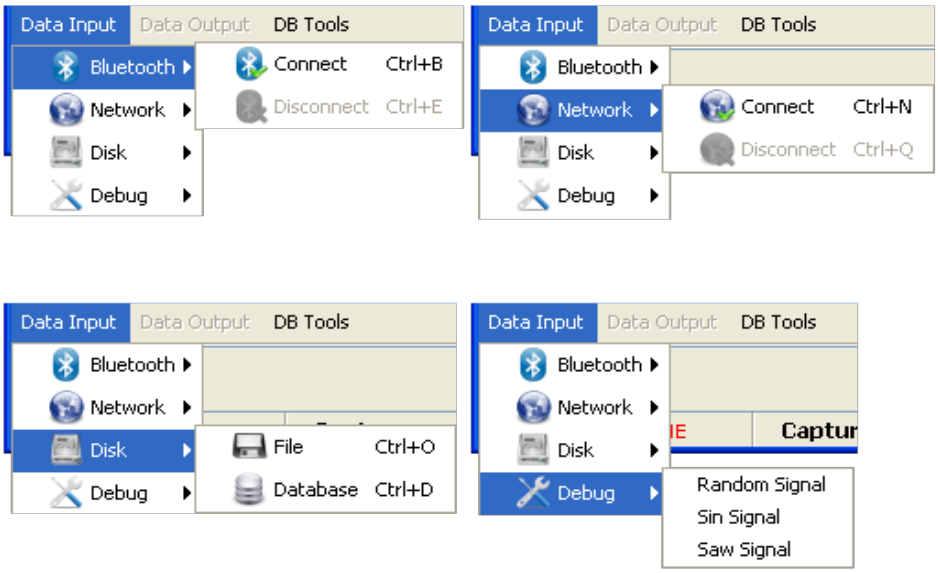

Figura 51: Menú de selección de fuente de datos.

Dependiendo del tipo de fuente de datos seleccionada la aplicación pedirá al usuario los datos relativos a la fuente de datos concretas a través de diversas ventanas de diálogo. En el caso de las fuentes de datos de tipo "*Bluetooth*" se ha implementado un gestor de búsqueda de dispositivos y servicios Bluetooth *(Figura 52)* que guía al usuario a través del proceso de conexión a un dispositivo SignalWServer.

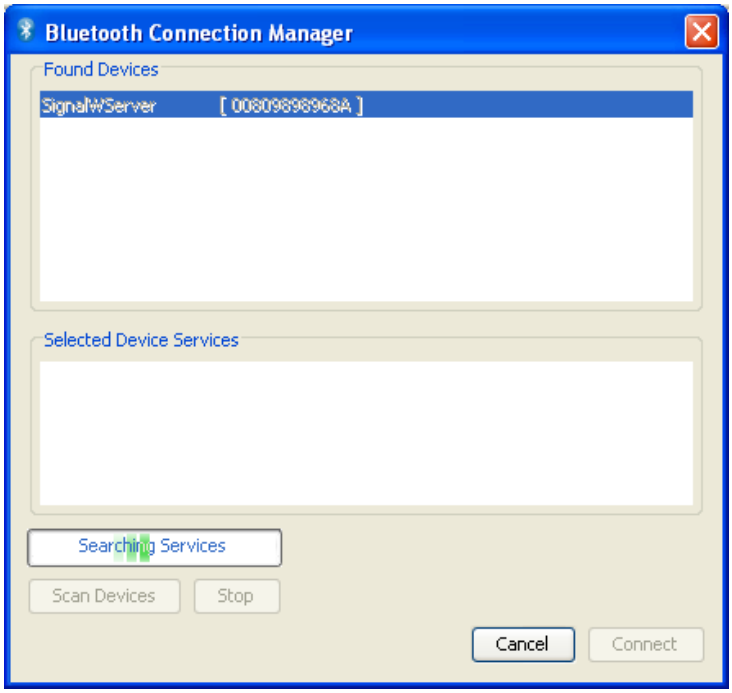

Figura 52: Diálogo de búsqueda de dispositivos y servicios Bluetooth.

En el caso de las fuentes de datos de tipo "*Network*", se piden a usuario los datos de un nodo BioSignal, es decir un nodo en el cual está siendo ejecutada otra instancia de la aplicación BioSignal en modo servidor (*ver 4.3.3*).

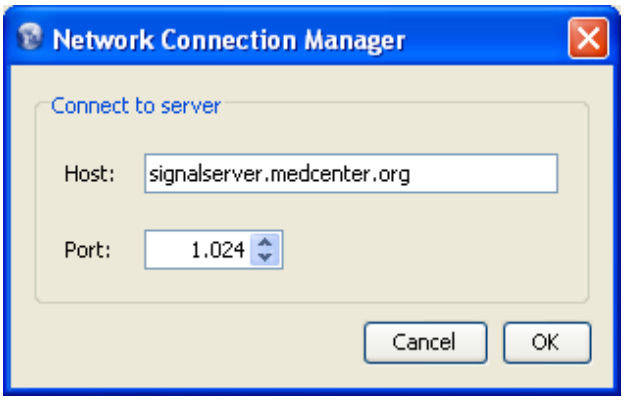

Figura 53: Diálogo de conexión a otro nodo BioSignal.

En el caso de la fuentes de datos de tipo "*File*", la aplicación muestra un diálogo estándar de selección de ficheros, aplicando por defecto un filtro de visualización de ficheros con extensión ".spf". Por otro lado, en el caso de fuentes de datos de tipo "*Database*" se muestra una ventana de diálogo *(Figura 54)* que permite al usuario elegir una entre todas las señales almacenadas en la base de datos local.

Figura 54: Diálogo de selección de señal de la base de datos.

4.3.2.- Parar entrada de datos

Para finalizar con la visualización de cualquier tipo de señal se ha dispuesto en el menú "*Data Input*" un ítem llamado "*Stop All*" (*Figura 55*), que sólo está activo cuando la aplicación está mostrando datos de señal, que permite la parada de cualquier entrada de datos en curso, lo que conlleva la desaparición de la representación gráfica de los datos y el reinicio del estado de la ventana.

Por otro lado cabe destacar que en el caso de las fuentes de datos de tipo "*Bluetooth*" y "*Network*" existe un ítem "*Disconnect*" *(Figura 55)* dentro de cada submenú, que sólo está activo cuando se selecciona la fuente de datos pertinente. Ambos ítems se comportan de manera exactamente igual que el mencionado "Stop All", sin embargo se han mantenido puesto que se ha estimado que son sumamente intuitivos y pueden mejorar la usabilidad de la aplicación.

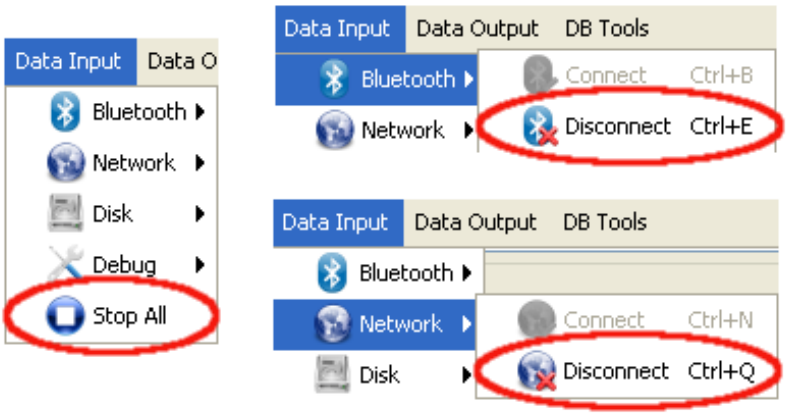

Figura 55: Ítems de parada de entrada de datos.

Al realizar una parada de entrada de datos de una fuente de tipo "*streaming*" la aplicación se desconectará de la fuente concreta y parará cualquier captura en curso (*ver 4.3.3*), incluyendo las captura de datos a red. Esto es debido a que si no existe ningún flujo de datos entrante no tiene sentido seguir capturando datos, puesto que éstos no van a seguir recibiéndose.

4.3.3.- Iniciar salida de datos

Denominaremos como **captura** a la redirección de los datos que provienen de una fuente de entrada hacia un sumidero de datos de salida.

El inicio del mecanismo de captura se hace a través del menú "*Data Output*", que únicamente está activo cuando hay una fuente de datos en "*streaming*" enviando datos a la aplicación BioSignal. No es posible la realización de la captura de los datos que provengan de fuentes de datos estáticas debido a que esto es equivalente a realizar una copia de la fuente de datos y existen mejores mecanismos para realizarla[14].

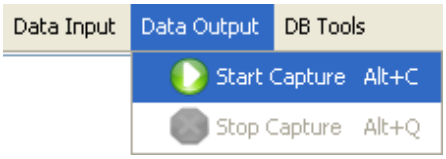

Figura 56: Menú de inicio de captura.

14 Por ejemplo, en el caso de ficheros, el mecanismo de copiado del sistema operativo en el que se ejecute la aplicación.

Dentro del menú "*Data Output*" existe el ítem "*Start Capture*" *(Figura 56)* el cual, al ser seleccionado, muestra un diálogo al usuario para que escoja el modo de captura que desee. Existen varios modos de captura de datos disponibles, los cuales, sin embargo, no pueden ser habilitados de forma simultánea:

- None: desactiva la captura de datos, es la opción por defecto.
- File: captura a un fichero en el formato propio SPF.
- Database: captura a la base de datos local HSQLDB.
- Network: habilita el modo servidor de la aplicación.

En el caso del modo de captura "*File*", se insta al usuario bien a seleccionar un nuevo nombre de fichero y una ruta en la cual se almacenará un nuevo fichero o bien a que elija un fichero existente del sistema operativo, con la pertinente advertencia por parte de la aplicación. Esto se hace a través de un diálogo estándar de salvaguarda de ficheros.

En el caso del modo de captura "*Database*", el usuario podrá introducir un nuevo nombre de señal o seleccionar uno existente a través del diálogo de selección de señal que hemos introducido anteriormente *(Figura 54)*. Al igual que en el modo anterior si el nombre de señal existe se avisará al usuario de esta situación.

Por último el modo "*Network*" es el que brinda la posibilidad de que la aplicación actúe como servidor de señales, permitiendo, como se ha mencionado anteriormente, que diferentes instancias de BioSignal interactúen formando una red, de topología arbórea, para la difusión de datos de señal. En este caso, el usuario podrá introducir un número de puerto TCP en el que la aplicación escuchará peticiones de conexión de otras instancias de la aplicación BioSignal y podrá también configurar si se desea activar o desactivar el algoritmo de Nagle (*ver 4.4.2*) para la transmisión de los datos de señal (por defecto está activado).

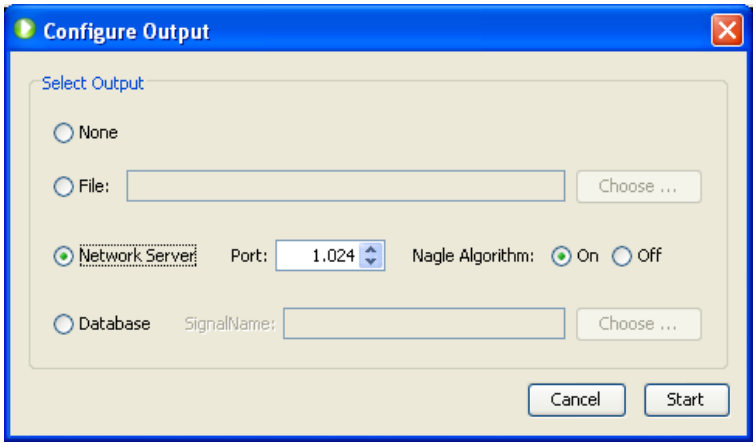

Figura 57: Diálogo de selección del modo de captura.

4.3.4.- Parar salida de datos

El usuario puede parar la captura de los datos en el momento que desee a través del ítem "*Stop Capture*" del menú "*Data Output*", lo que hace que se deje de atender peticiones de conexión y se cierren las conexiones existentes, en el modo de captura "*Network*", y en el caso de capturas en modo "*File*" o "*Database*", se deje de escribir datos en los sumideros pertinentes.

4.3.5.- Manejar la base de datos

Al introducir la posibilidad de almacenar datos en una base de datos local es prácticamente de obligado cumplimiento que se provea de mecanismos para la gestión de la misma. En nuestro caso los mecanismos de gestión que proporciona la aplicación BioSignal son los siguientes:

- Borrado de señales.
- Renombrado de señales.
- Exportación de señales a fichero SPF.
- Importación de señales desde fichero SPF.

Los dos primeros tipos de gestión se pueden realizar a través del ítem "*Database Manager*", mientras que el resto se realizan a través del ítem "*Signal Manager*". Ambos están ubicados en el menú "*DB Tools*" *(Figura 58).*

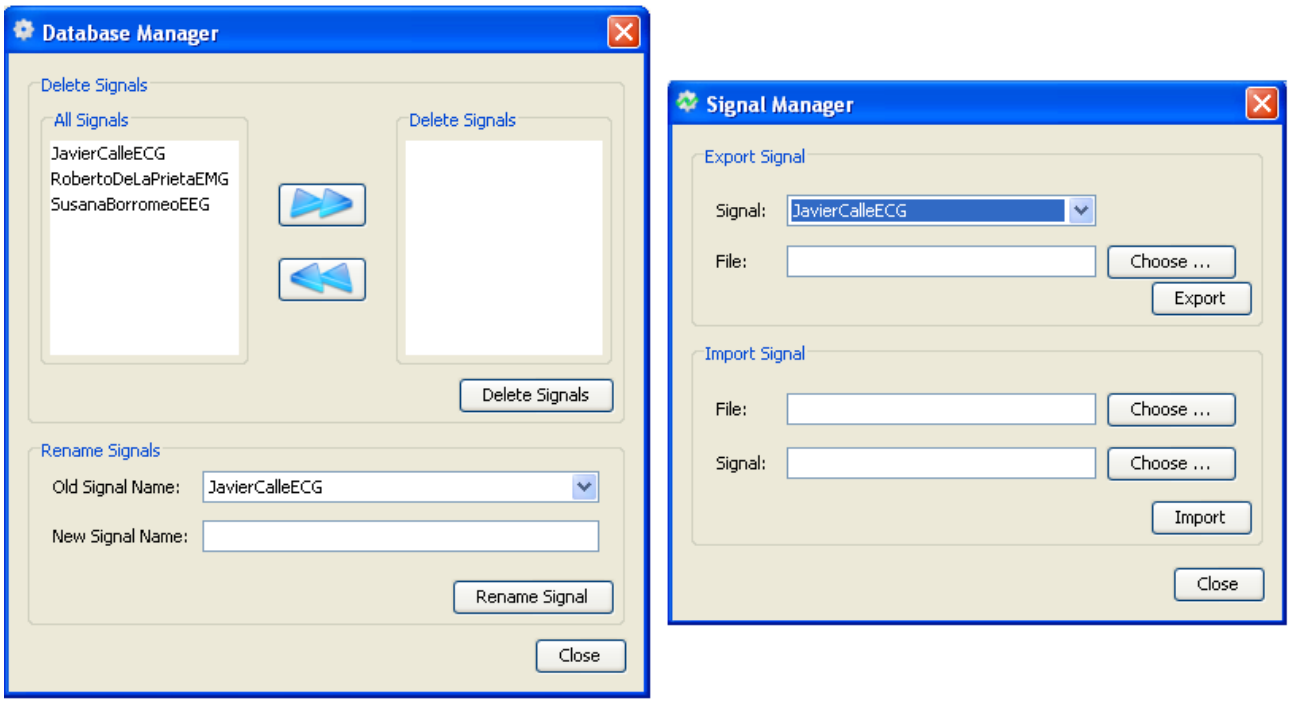

Figura 58: Diálogos para la gestión de la base de datos.

4.3.6.- Consideraciones adicionales

Cuando se realiza un muestreo de varios canales de señal de forma paralela, se define como **frame** al conjunto de datos formado por una muestra de cada canal, las cuales han sido tomadas en el mismo instante de tiempo.

La aplicación tiene específicamente implementadas partes de la interfaz que sirven para notificar información al usuario que le puede ser de utilidad. Toda esta información se encuentra localizada en la parte inferior de la ventana de visualización *(Figura 59)* y se compone de:

- Estado de la entrada (*Status*): notifica al usuario el modo de entrada de datos que está siendo usado.
- Estado de la salida (*Capture*): notifica al usuario el modo de captura que está siendo usado.
- Velocidad (*Speed*): para las entradas de datos de tipo "*streaming*", notifica al usuario la velocidad de recepción en frames/seg.

Figura 59: Barra inferior de BioSignal

Se ha invertido un gran esfuerzo en la parte gráfica de la aplicación, debido a que requiere de conocimientos avanzados de las librerías *Swing* de Java. Especialmente se ha dedicado mucho trabajo para que la visualización de los datos de señal sea flexible y de fácil manejo. Para ello se ha establecido que el área de ventana, durante una monitorización, por defecto se ajuste al área de pantalla. Por otro lado, si se deben monitorizar un número de señales mayor del que se permite visualizar, se proporciona automáticamente un *scrollbar* vertical para poder desplazarse hasta aquellas señales que estén fuera del ámbito visual.

Una de las labores de mayor complejidad fue la de integrar la posibilidad de navegar horizontalmente por aquellas señales que provenían de fuentes de datos estáticas, en este caso se optó por introducir un "*slider*" independiente para cada una de las señales de manera que se pudiera controlar el área de visualización de cada señal de forma separada Sin embargo, el uso de esta técnica, supone una limitación en cuanto al tamaño máximo de fichero que se va a poder visualizar, debido a que la implementación del *slider* en Java (*JSlider*) está basada en el tipo entero (*Integer*).

El máximo número de frames (*MaxFrames*) que van a poder ser visualizados mediante la aplicación BioSignal es del orden del máximo entero en Java (*MaxJavaEnt*). Dado que los enteros con signo de Java son de 32 bits, esto implica que:

$$MaxJavaEnt = 2^{31} - 1$$
$$MaxFrames = MaxJavaEnt = \mathbf{2147483648}\ \boldsymbol{frames}$$

Si definimos *Ch* como el numero de canales de los cuales alberga información un fichero y tenemos en cuenta que el tamaño de cada dato es de un byte y que además cada 4 Frames se envían 3 bytes de meta-información (*ver 4.4.1*) entonces el tamaño máximo (*Tam*) de un fichero es del orden de:

$$Tam = [(Ch \cdot MaxFrames) + (\frac{3}{4} \cdot MaxFrames)]\ Bytes = [(Ch + \frac{3}{4}) \cdot MaxFrames]\ Bytes$$

Por ejemplo para un fichero que deba albergar 10 canales de señal, el tamaño máximo de un fichero de captura debería ser:

$$Tam = [10.75 \cdot MaxFrames]\ B \simeq \mathbf{21.5}\ \boldsymbol{GB}$$

Por otro lado, si suponemos una tasa de transferencia de 1000 frames/seg, que es la necesaria para el muestreo de una señal de EMG, podemos obtener que el tiempo máximo de captura (*T*) es:

$$T = \frac{MaxFrames}{1000}\ seg \simeq \mathbf{596.5}\ \boldsymbol{horas} \simeq \mathbf{20.5}\ \boldsymbol{días}$$

El cual parece un tiempo máximo de captura muy razonable.

Por último cabe destacar que una de las preocupaciones que se tuvieron en cuenta, a la hora de diseñar la aplicación BioSignal, fue que su programa fuese flexible a la hora de poder introducir nuevos tipos de fuentes y sumideros de datos. Para facilitar la labor en una futura revisión del software se crearon dos interfaces, los cuales proporcionan una norma que deben cumplir las nuevas fuentes y sumideros de datos para ser compatibles con la aplicación. La primera de las interfaces, llamada "*SignalPackDataStream*", sirve para crear fuentes y sumideros de tipo *stream*, la segunda es la interfaz "*RandomDiskAccess*" y sirve para la creación de fuentes y sumideros de tipo estático.

4.4.- Comunicación entre Subsistemas

4.4.1.- Comunicación entre SignalWServer y un nodo BioSignal

Aunque ya se ha hablado en secciones anteriores del protocolo basado en comandos AT usado entre la unidad de control y el módulo wireless, aún no se ha comentado qué ocurre cuando se establece una conexión entre un dispositivo SignalWServer y un nodo BioSignal. Una conexión se considera establecida cuando a la unidad de control del dispositivo SignalWServer le llega una respuesta no solicitada, proveniente del módulo wireless, del tipo CONNECT, a partir de ese momento todo lo que envíe la unidad de control hacia el módulo wireless le será enviado por la conexión inalámbrica al nodo BioSignal remoto.

Sobre el establecimiento de conexión inalámbrica, se ha diseñado un pequeño protocolo que permita realizar un streaming de datos de señal de manera sincronizada, el proceso básico se puede ver en la siguiente figura *(Figura 60).*

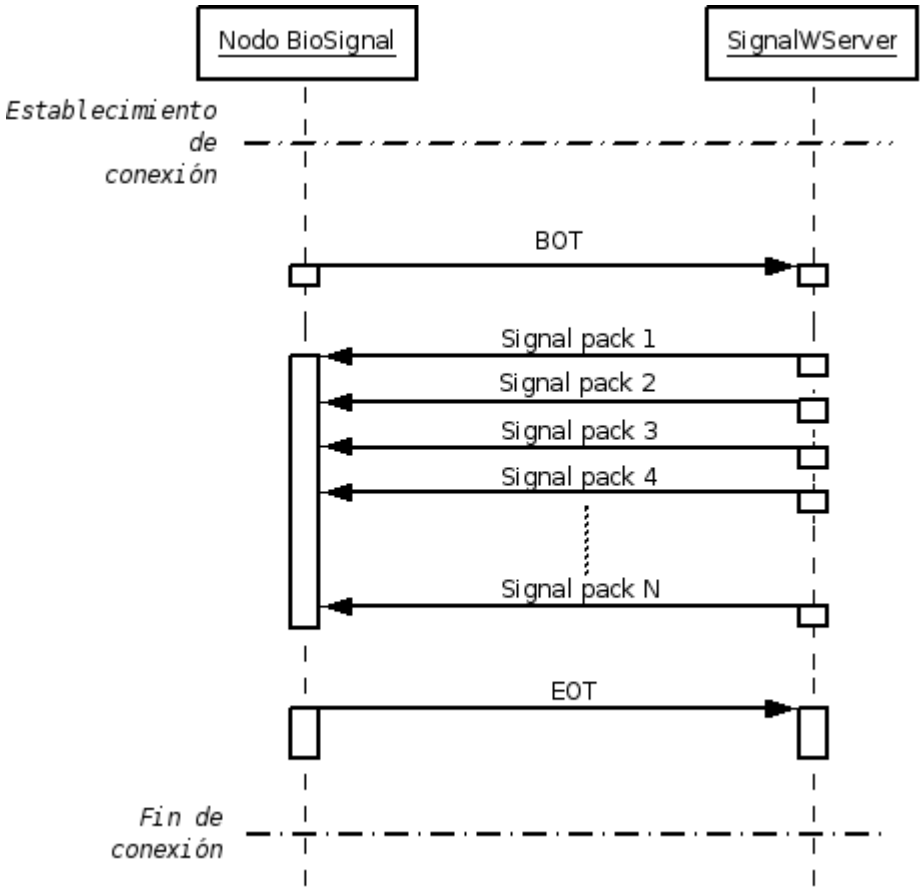

Figura 60: Diagrama de secuencia del protocolo SignalWServer – BioSignal.

Como puede verse, tras el establecimiento de la conexión, el nodo Biosignal debe enviar un BOT (*Begin Of Transmission*) para notificar al dispositivo SignalWServer de que está preparado para recibir los datos de las señales digitalizadas. Dicho BOT consiste en la cadena de caracteres 'BOT' seguida de un retorno de carro (CR). A partir de que el dispositivo SignalWServer recibe dicho BOT empieza a retransmitir “paquetes de señal” hasta que el nodo BioSignal envíe un EOT (*End Of Transmission*), que consiste en la cadena de caracteres 'EOT' seguido de un CR. Una vez se ha mandado un EOT por parte de la aplicación BioSignal, ésta simplemente abandona la conexión, con la consecuente recepción por parte de la unidad de control del dispositivo SignalWServer de una respuesta no solicitada de tipo NO CARRIER. Por otro lado, se tiene en cuenta, por parte del dispositivo SignalWServer, la posibilidad de desconexión fortuita del nodo BioSignal, lo que implicaría que no es enviado ningún EOT de finalización.

En el diagrama de secuencia anterior *(Figura 60)* se puede observar que, al realizar el envío de “paquetes de señal”, no se espera confirmación alguna por parte del receptor de que el paquete ha llegado a su destino, a esto se le denomina **protocolo de comunicación asíncrono**. Por otro lado en un **protocolo de comunicación síncrono**, cuando un emisor envía un conjunto de datos, el receptor, al recibirlos, devuelve una respuesta para confirmar que los datos han llegado correctamente. Dicha respuesta se conoce comúnmente con el nombre de **ACK**.

Como hemos mencionado, el dispositivo SignalWServer envía los datos referentes a las señales digitalizadas por medio de lo que hemos llamado “paquetes de señal”. Dichos paquetes de señal son ristras de bytes con un formato que permite cierta sincronización entre el nodo BioSignal y el dispositivo SignalWServer sin que se tenga que recurrir a un protocolo síncrono basado en ACKs.

Un “*signal pack*” o paquete de señal está compuesto de dos partes:

- Metainformación: Es la primera parte del paquete y ocupa 3 bytes fijos. El primer byte es una cabecera (el número 255), y los dos restantes representan, el número de frames y canales por frame que componen el paquete.
- Datos: Se envía a continuación de la metainformación.

La parte de datos está formada por las muestras procedentes de la digitalización de señales. Su tamaño varía en función del numero de frames y canales que deban componerlo. Para un paquete con un numero de frames *NF* y un número de canales por frame *NC*, el tamaño de la parte de datos del paquete (TAM_{DATOS}) es:

$$TAM_{DATOS} = NF \cdot NC\ Bytes$$

En la siguiente figura *(Figura 61)* puede verse el formato de los paquetes de señal.

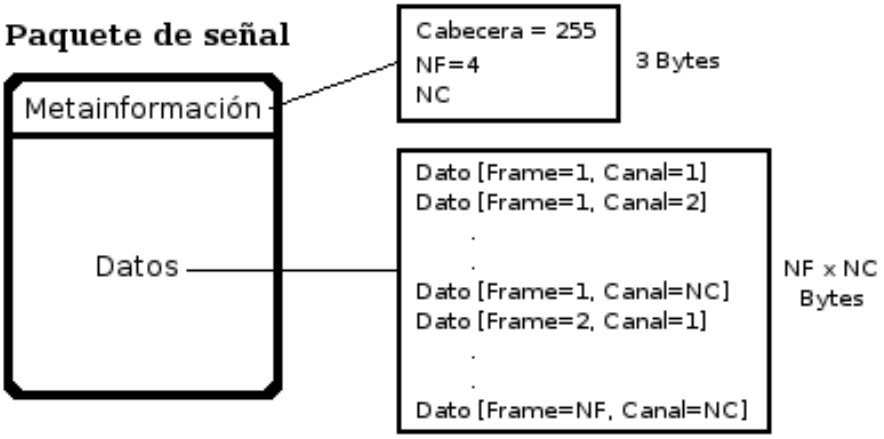

Figura 61: Formato de los paquetes de señal.

El valor de cabecera de la parte de metainformación de cada paquete sirve para que exista un sincronismo entre emisor y receptor, puesto que el emisor no espera confirmaciones de recepción correcta, lo que se hace es marcar con un 255 el inicio de cada paquete, de manera que el receptor puede notificar un fallo al usuario de la aplicación si esperaba una cabecera 255 y obtuvo otro valor. Además, esto garantiza que si existe desorden o pérdida de bytes en la transmisión, el receptor abortaría la conexión debido a que no están llegando datos de señal correctos.

Por otro lado el hecho de que cada paquete esté autocontenido, es decir, que posea toda la información necesaria, hace que en la aplicación BioSignal se pueda detectar el número de canales que se están enviando de forma dinámica, pudiendo representar gráficamente sólo los canales necesarios.

Con respecto al desaprovechamiento del ancho de banda que supone el envío de la metainformación cada vez que se transmite un paquete de datos, si tomamos un caso típico en el que se usen los 10 canales del dispositivo (*NC=10*), el porcentaje que supone el tamaño de la metainformación (TAM_{META}) frente al tamaño del paquete (TAM_{PACK}) es:

$$TAM_{META} = 3$$
$$TAM_{PACK} = TAM_{DATOS} + TAM_{META} = NF \cdot NC + 3 = 4 \cdot 10 + 3 = 43$$
$$Desperdicio = \frac{3 \cdot 100}{43} \simeq \mathbf{7\ \%}$$

Es decir, que el desaprovechamiento del ancho de banda es del orden del 7%, valor que no es excesivamente elevado. Aunque este porcentaje aumenta hasta cerca del 43% en el caso más desfavorable, cuando se monitoriza un sólo canal de señal, se ha decidido no variar el numero fijo de frames que se envían debido a que, en caso de aumentarlo, se observaría una tasa de refresco menor de la señal en la aplicación BioSignal y, por otro lado, se tendría la necesidad de reservar una cantidad de memoria mayor en la unidad de control.

4.4.2.- Comunicación entre nodos BioSignal

Una parte del software BioSignal consiste en la posibilidad de leer las señales biológicas usando como fuente de datos otro nodo BioSignal disponible en internet, así como tener la capacidad de redirigir las señales que se estén recogiendo poniéndolas a disposición de otros nodos BioSignal en el ámbito de internet. Por ello, la aplicación BioSignal debe tener la posibilidad de actuar de ambos modos, por un lado de cliente y por otro lado de servidor de paquetes de señal, siendo este comportamiento al que se le ha estado denominando, durante el transcurso del presente documento, como P2P.

El software BioSignal es capaz de leer paquetes de señal de una fuente de datos de tipo streaming y representarlos en pantalla. Opcionalmente, puede actuar de manera simultánea como servidor de señales, "volcando" paquetes de señal a internet. Este comportamiento de servidor de señales permite, además, que un mismo nodo BioSignal sirva paquetes de señal a varios clientes BioSignal, haciendo una especie de "*multicast*" de señales, pudiendo conseguir topologías de red arborescentes como la de la siguiente figura *(Figura 62)*

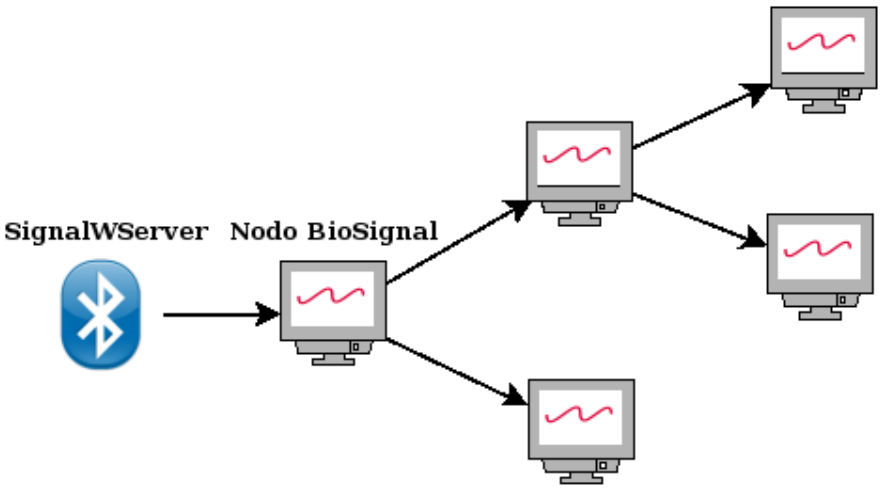

Figura 62: Topología en árbol de una red BioSignal.

4.4.2.1.- Alternativas de protocolos

Para la elección del protocolo de conexión entre los distintos nodos se han planteado las siguientes alternativas:

- TCP
- UDP

TCP posee las ventajas de que es un protocolo fiable. Esto quiere decir que está diseñado para que los paquetes que se envían desde un nodo emisor lleguen a un nodo receptor sin que existan pérdidas de paquetes ni se produzca un desorden de los mismos. Además el comportamiento de los sockets TCP en Java se basa en el de los streams usados para otras fuentes y sumideros de datos, de hecho, una vez abierta una conexión, se obtienen los streams de entrada y de salida de la conexión y se puede leer/escribir de/en la conexión fácilmente, permitiendo una gran flexibilidad y comodidad.

UDP es un protocolo que no es fiable, con lo que los paquetes que se envíen por la red pueden perderse antes de llegar a su destino o desordenarse, por lo que sería tarea del programador cerciorarse de que los paquetes perdidos se retransmitan en el origen o los desordenados se ordenen en el destino. En realidad es ideal para su uso en redes de área local, donde el índice de pérdidas es muy bajo y la presencia de pocos *switches* hace que haya poco desorden de paquetes. Dado que es un protocolo que permite mayor velocidad que TCP por ser más ligero, en comunicaciones que requieran tiempo real es lo que se debería usar puesto que en estas circunstancias un paquete que se haya perdido o no llegue en un tiempo razonable debería ser descartado.

Como se puede observar, ambos protocolos tiene ventajas y desventajas, sin embargo, dependiendo del uso de la aplicación, será más idóneo el uso de uno u otro, por ello se han tomado en consideración los siguientes factores:

- LAN frente a WAN: en nuestro caso hemos decidido que el entrono de operación de nuestra aplicación no sólo se debe limitar al de las redes de área local (LAN) si no que un nodo BioSignal puede poner sus datos a disposición de toda la red (WAN).
- Tiempo Real frente a Fiabilidad: en este caso, nuestra aplicación si que tiene requisitos de tiempo real, para los cuales aporta mayores ventajas el uso de UDP, sin embargo por otro lado nos resulta más importante que la señal que llegue a un nodo BioSignal sea fiable, aunque llegue con una mayor latencia, para lo que aporta ventajas el protocolo TCP.

- Flexibilidad: el código fuente de BioSignal está estructurado para que sea simple introducir un nuevo tipo de fuente/sumidero de datos, siempre y cuando éste se comporte de la manera más similar posible a un stream de entrada/salida, característica que nos ofrece la implementación del protocolo TCP en Java.

De este modo, el protocolo de comunicación que hemos usado entre nodos BioSignal es TCP, el cual cumple las expectativas en cuanto al uso de la aplicación en el ámbito de la WAN, fiabilidad y flexibilidad, aunque, por contra, se vean mermados los requisitos de tiempo real en la transmisión de las señales.

4.4.2.2.- El algoritmo de Nagle

Para tratar de paliar los efectos de la pérdida de tiempo real de la aplicación como consecuencia del uso del protocolo TCP, se ha barajado la posibilidad de activar o desactivar el algoritmo de Nagle para la transmisión de señales.

El algoritmo de Nagle es un método que permite optimizar el ancho de banda en las comunicaciones mediante el protocolo TCP. Consiste en tratar de no enviar segmentos TCP con un campo de datos excesivamente pequeño, puesto que de ser así, cada vez que se quiera enviar una porción de datos muy pequeña se estaría desperdiciando ancho de banda por el hecho de tener que enviar, junto con los datos, las cabeceras relativas al protocolo TCP y a los protocolos de niveles inferiores en la pila OSI, cuyo tamaño puede llegar a superar con creces el de los propios datos. En líneas generales, el algoritmo de Nagle actúa esperándose a que se tengan suficientes datos que quieran ser enviados y agrupándolos para poder enviarlos en un sólo paquete con un tamaño de datos suficientemente grande para que las cabeceras antes mencionadas no supongan un gran desperdicio de ancho de banda.

Por otro lado, el algoritmo de Nagle hace que aumente la latencia en las comunicaciones, puesto que un dato no es enviado hasta que el algoritmo decide que se debe enviar y esto es algo que no beneficia a las características de tiempo real que debería tener nuestra aplicación.

La activación/desactivación del algoritmo de Nagle es un aspecto delicado, puesto que se puede llegar a hacer un uso demasiado ineficiente del ancho de banda de una red, por lo tanto, se ha decidido que la aplicación BioSignal permita al usuario la configuración del estado de dicho algoritmo al activar el modo servidor en la aplicación BioSignal *(Figura 63)*.

Figura 63: Configuración del estado del algoritmo de Nagle.

4.4.2.3.- Transmisión de datos

Al igual que en anteriores secciones, se planteó la posibilidad de realizar el envío de paquetes de señal transmitiéndolos byte a byte. Sin embargo, en caso de inhabilitar algoritmo de Nagle si que puede ser un desperdicio demasiado elevado de ancho de banda, además de ser ineficiente desde el punto de vista de la programación. Por ello se decidió usar la capacidad de serialización de objetos que brinda Java.

La **serialización**, también llamado "*marshalling*" o aplanado, de objetos es un proceso por el cual se toma un objeto y se representa como una serie de bytes, los cuales pueden ser enviados por red o almacenados en un soporte no volátil. La gran ventaja que ofrece es que dicha serie de bytes está en un formato común que puede ser entendible en otras plataformas.

En nuestro caso, cuando leemos datos que provienen de un dispositivo SignalWServer, usamos un objeto de la clase "*SignalPack*" para almacenar los datos de paquete que leemos, permitiéndonos operar con dichos datos de una manera cómoda. Dicha clase está marcada como *Serializable*, de manera que los objetos de dicha clase pueden ser aplanados para posteriormente enviarlos por red.

Cuando un cliente y un servidor BioSignal se conectan, el servidor obtiene de la conexión el *stream* de salida, el cual enlaza con un "*ObjectOutputStream*" y el cliente, por su lado, obtiene el *stream* de entrada de la conexión, el cual enlaza con un "*ObjectInputStream*". De esta manera el servidor sólo tiene que escribir objetos de la clase "*SignalPack*" en su "*ObjectOutputStream*" mediante el método *write* del mismo mientras que, en el otro extremo de la conexión, el cliente podrá leerlos mediante la llamada al método *read* de su "*ObjectInputStream*".

5.- Resultados

Se ha diseñado un prototipo hardware que permite la digitalización y envío por Bluetooth de un máximo de 10 canales de señal diferentes con unas dimensiones reducidas (7.5 cm x 9 cm), el cual puede ser alimentado mediante baterías, permitiendo su portabilidad.

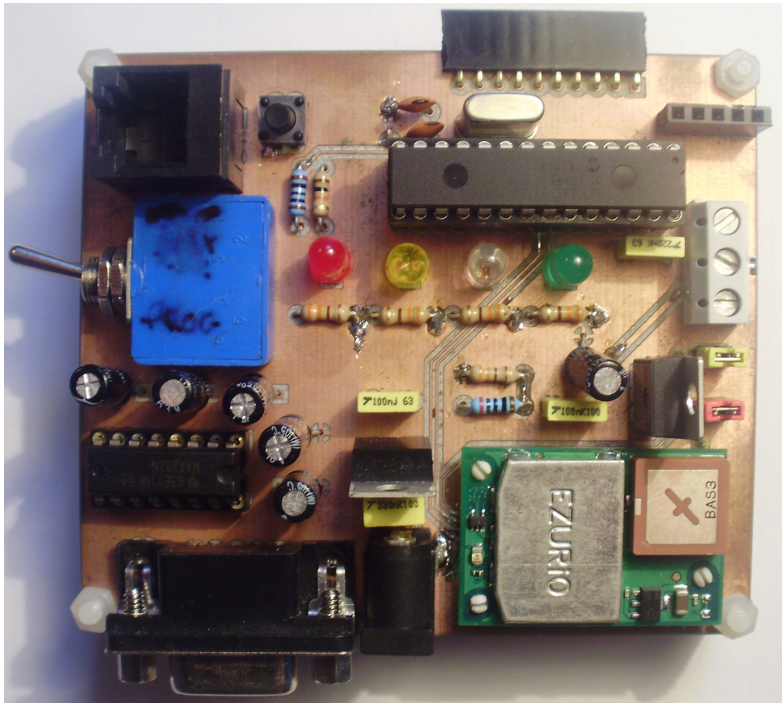

Figura 64: Dispositivo hardware desarrollado.

Se ha conseguido una tasa de transmisión de 1070 frames/seg para el límite de 10 canales de señal permitidos, lo que posibilita el envío de señales de frecuencia máxima del orden de 500 Hz. Sin embargo se puede aumentar dicha frecuencia disminuyendo el número de canales.

Se ha dotado al dispositivo hardware de una consola de administración accesible mediante el puerto RS232 que lleva implementado, pudiendo acceder a ella través de una sencilla consola de puerto serie, configurada a 115200 baudios en modo 8n1. De esta manera se pueden realizar modificaciones en el dispositivo relativas al número de canales a enviar y a la frecuencia de muestreo de cada uno de ellos (*ver 4.2.2*). En la siguiente figura (*Figura 65)* se puede observar una captura de pantalla de la configuración del dispositivo usando el programa "*Eltima Advanced Serial Port Terminal*".

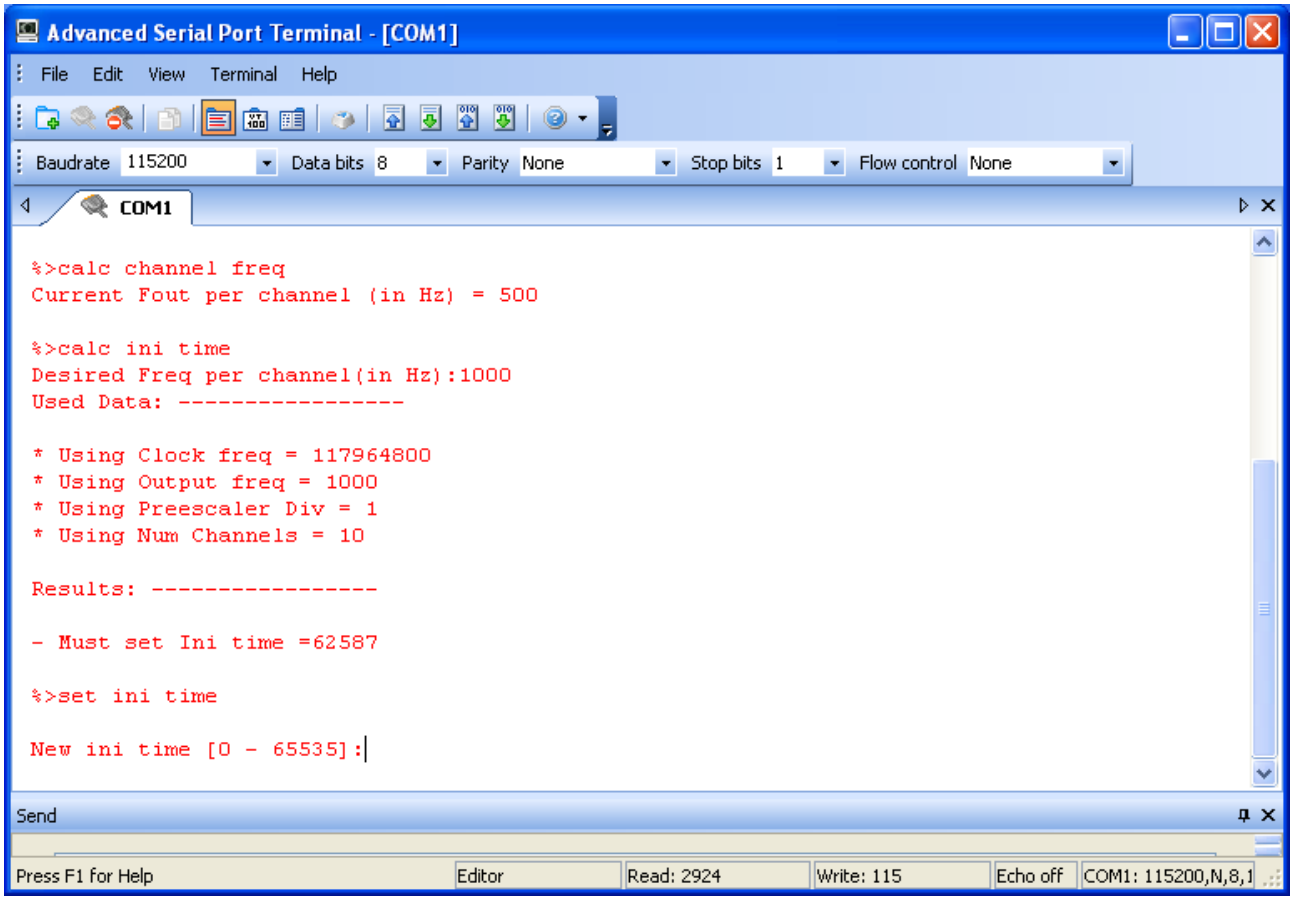

Figura 65: Configuración del dispositivo usando una consola para puerto serie.

Se ha medido el consumo eléctrico máximo del dispositivo tomando los valores tensión y corriente de entrada (V_{IN} e I_{IN}, respectivamente) cuando éste se encuentra retransmitiendo pero sin ninguna etapa de adquisición conectada, obteniendo los siguientes valores:

$$V_{IN} = 8.72\,V$$
$$I_{IN} = 110\,mA$$

Esto implica una potencia (P_{IN}):

$$P_{IN} = V_{IN} \cdot I_{IN} = 8.72\,V \cdot 110 \cdot 10^{-3}\,A = \mathbf{0.96\,W}$$

Por otro lado se ha conseguido desarrollar una aplicación software, con una interfaz de usuario amigable, que permite la recepción de un número variable de canales de señal y, consecuentemente, representa gráficamente en pantalla dichas señales de forma simultánea. Además, se ha logrado otorgar al software la capacidad de retransmitir la información relativa a las señales que están

siendo recibidas hacia otros equipos que ejecuten la misma aplicación. De esta manera, los nodos que ejecuten dicha aplicación pueden interactuar entre ellos formando una red de tipo P2P.

Se han proporcionado mecanismos para que el usuario pueda capturar las señales a memoria no volátil, pudiendo elegir actualmente entre una captura a fichero o a una base de datos embebida en la aplicación. Por otro lado se permite la visualización de señales previamente guardadas, pudiendo realizar operaciones más avanzadas sobre las mismas, tales como hacer zoom a la señal, ampliar el área vertical de visualización o definir "*trackers*", los cuales proporcionan un método para poder medir los datos de señal obtenidos *(Figura 66)*.

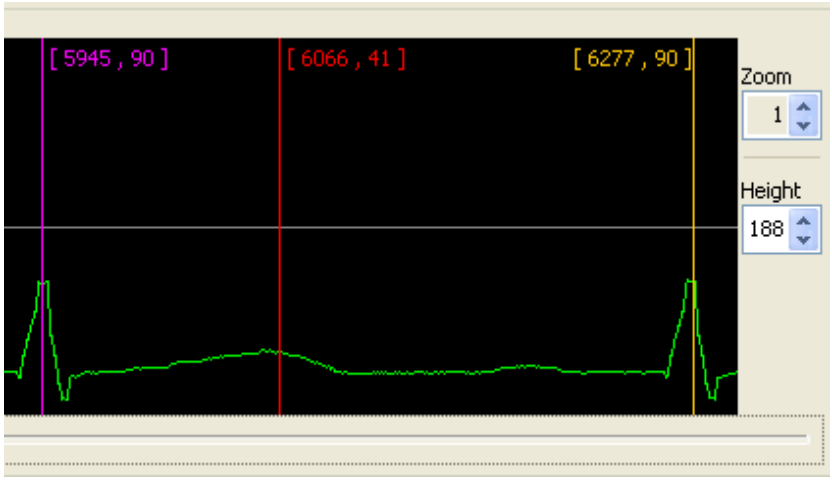

Figura 66: Canal de señal en la interfaz de visualización.

Para poder comprobar la fiabilidad del sistema se ha utilizado un dispositivo físico que reproduce las señales de ECG [ST] junto con un módulo hardware de acondicionamiento de las mismas y se han guardado en disco las señales resultantes usando un osciloscopio digital[15], el cual permite la captura de datos de señal a una memoria USB. Mediante el software Octave[OCT], se han exportado los datos de señal , obteniendo el siguiente gráfico *(Figura 67)*.

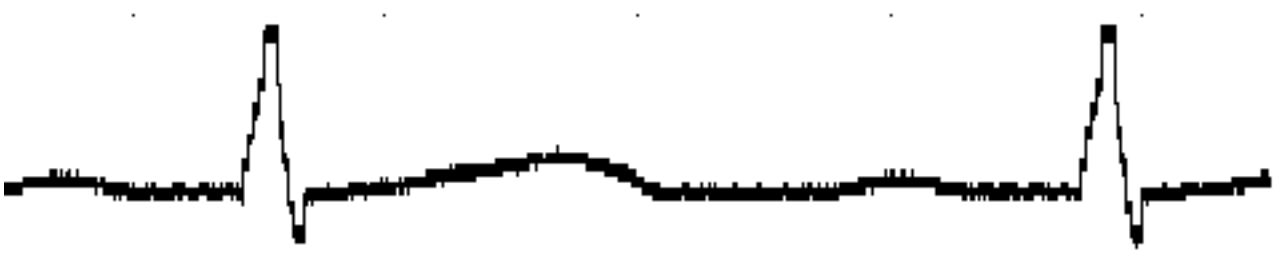

Figura 67: Señal de ECG medida mediante osciloscopio.

15 LeCroy Wavesurfer 422. 200MHz, 2 Gs/seg.

Por otro lado se realizó el mismo experimento usando el sistema propuesto y se obtuvo la siguiente señal resultante (*Figura 68)*, muy similar a la anterior.

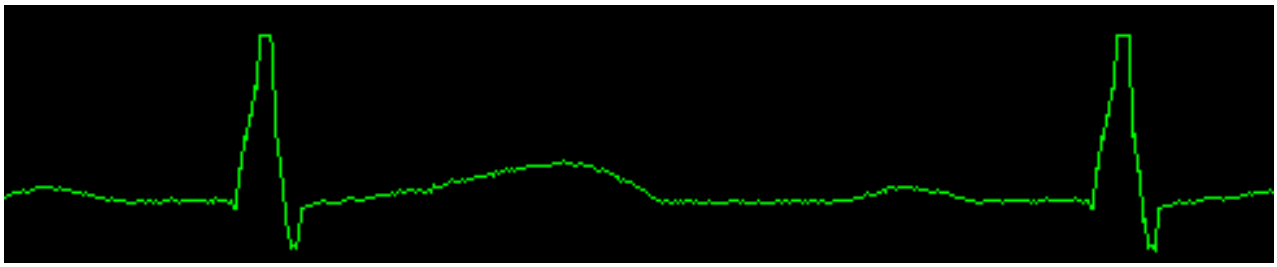

Figura 68: Señal de ECG medida mediante el sistema propuesto.

Además se ha diseñado una sencilla etapa de acondicionamiento de señal, la cual permite la lectura de temperatura del cuerpo humano. Dicha etapa ha sido desarrollada usando el sensor de temperatura LM35 y ha sido calibrada para que el sistema pueda muestrear temperaturas en la escala de 50 ºC a 23 ºC, de manera que en el punto medio de la amplitud de la señal se encuentre representada la temperatura normal del cuerpo humano, 36.5 ºC. A continuación se puede observar una imagen de la etapa de acondicionamiento desarrollada *(Figura 69)*, así como su esquema de conexiones*(Figura 70)*.

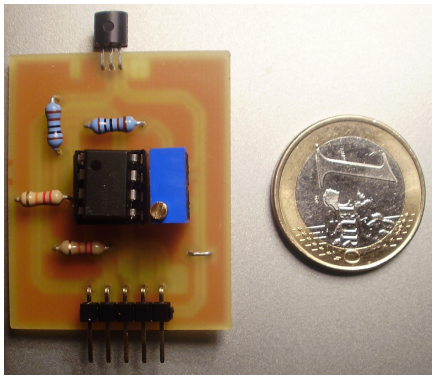

Figura 69: Sensor de temperatura desarrollado.

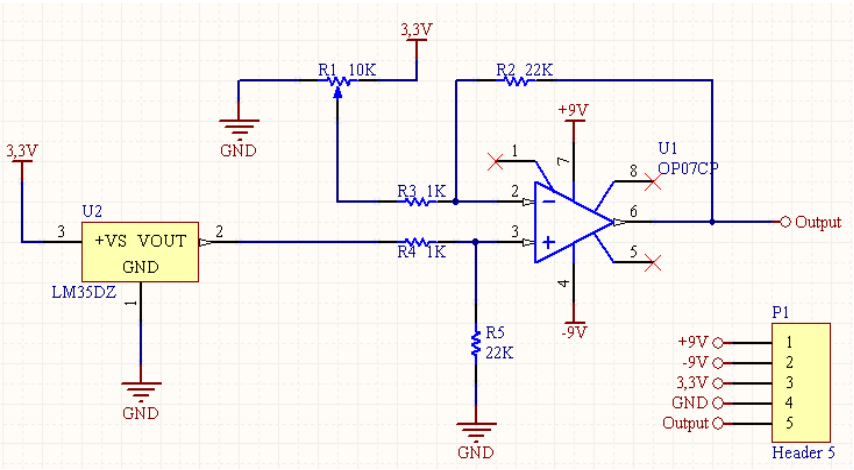

Figura 70: Esquema del sensor de temperatura desarrollado.

En otro orden de cosas, el desarrollo del sistema consta de 6989 líneas de código estructuradas como se muestra a continuación:

- Aplicación software BioSignal: 5904 líneas de código Java.
- Firmware de SignalWServer: 1085 líneas de código C.

6.- Conclusiones y futuras mejoras

6.1.- Conclusiones

Como resultado del trabajo realizado en este PFC se ha conseguido diseñar e implementar un sistema original y completo capaz de realizar la monitorización y transmisión de un número configurable de señales diferentes de manera simultánea.

El sistema está orientado para que las señales monitorizadas sean de naturaleza fisiológica, sin embargo dichas señales basta con que verifiquen unos requisitos de amplitud y frecuencia máxima para su correcta adquisición por parte del sistema. Por un lado la amplitud de la señal debe encontrarse en la escala de 0 a 3,3 V. Por otro lado, su frecuencia máxima permitida es inversamente proporcional al número de canales de señal a monitorizar, sin embargo, para el máximo número de canales establecido, la frecuencia máxima por canal de señal no debe superar los 500 Hz.

Debido a la limitación en el número de entradas analógicas disponibles en el microcontrolador, que gobierna el módulo hardware, el número máximo de canales disponibles es 10. No obstante, la aplicación software es capaz de manejar muchos más.

El sistema tiene la ventaja de que es completamente modular, por lo que sería factible reemplazar diversas partes del mismo o incorporar nuevas funcionalidades, sin que ello suponga un esfuerzo significativo. Con respecto al subsistema hardware, se ha procurado adaptar la arquitectura del circuito para que la tecnología wireless utilizada pueda ser reemplazada por otra, mencionando incluso alternativas viables. Con respecto al subsistema software, éste podría ser usado con otro tipo de dispositivos hardware siempre y cuando se cumpla el protocolo de comunicación que hemos especificado (*ver 4.4.1*) Adicionalmente, se ha logrado que el subsistema hardware sea configurable a través de una sencilla consola para puerto serie, de muy fácil manejo.

Una de las características más destacables que tiene el sistema es la posibilidad de distribuir la información a través de internet. De este modo, cualquiera de las instancias de la aplicación que implementa el visor de señales, puede actuar como servidor de datos de señal permitiendo conexiones vía red de otras instancias del mismo software, las cuales actuarán como clientes. Así

las instancias servidoras difundirán los datos de las señales adquiridas a los distintos clientes conectados.

Finalmente cabe destacar que, al haber sido desarrollado en lenguaje Java, el software de este proyecto es multiplataforma. Aunque, por el momento sólo ha sido probado en los sistemas operativos Windows XP y Linux[16], la adaptación a otros sistemas requeriría únicamente la instalación de un entorno de ejecución Java, el cual está disponible de forma gratuita para diferentes arquitecturas.

A continuación se muestra una imagen (*Figura 71*) de la parte hardware del sistema, incluyendo el dispositivo SignalWServer, el sensor de temperatura desarrollado y una etapa de acondicionamiento de señales de ECG.

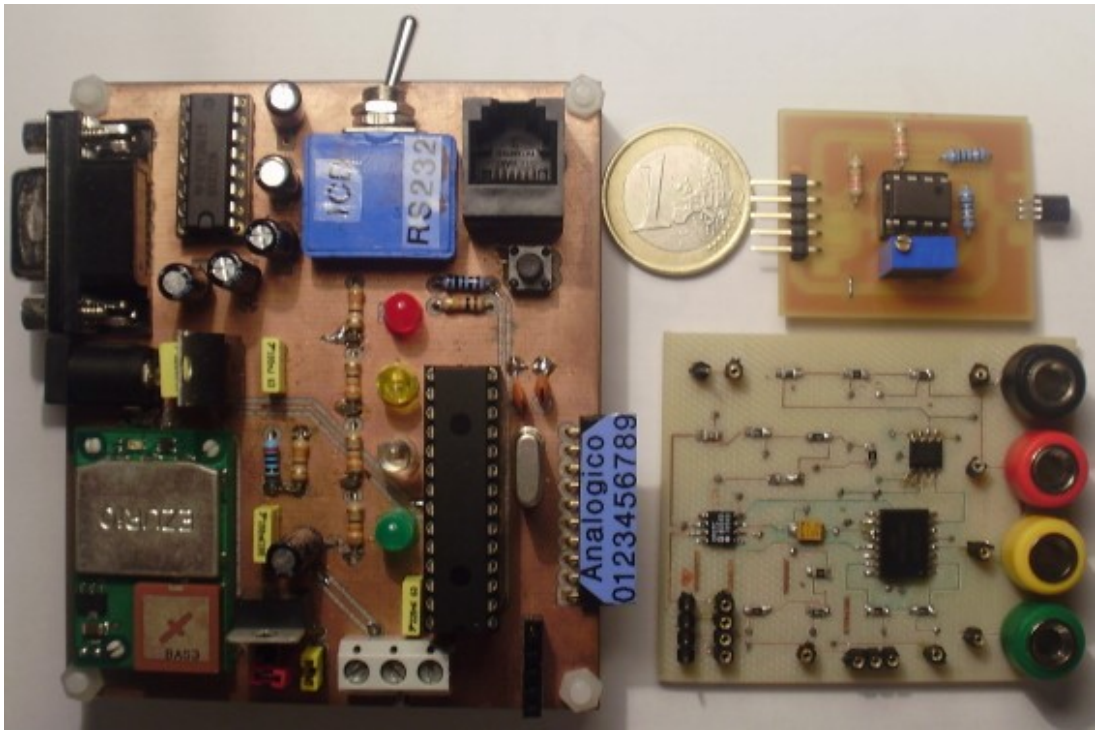

Figura 71: Sistema hardware completo.

6.2.- Futuras mejoras

Una primera mejora en lo que respecta a la parte hardware del sistema consistiría en modificar el firmware de la unidad de control para que permitiera conectar indistintamente un módulo Bluetooth (como el actualmente utilizado BISM II [BISM]) o un módulo Wi-Fi (en concreto el modelo WISMC03BI [WISM]). Es más, sería factible y relativamente sencillo modificar dispositivo

16 Concretamente Debian 5.04 “Lenny”.

hardware del sistema para que ambos módulos puedan coexistir y ser utilizados de forma simultánea.

También se podría mejorar el hardware del sistema permitiendo la transmisión de paquetes de señal mediante una tecnología cableada de alta velocidad, como puede ser USB o Firewire.

Otra mejora interesante del sistema sería la ampliación del firmware de la unidad de control para llevar a cabo un procesamiento digital adicional de las señales adquiridas en el propio subsistema hardware. Dicho procesamiento se podría implementar a través de un sistema operativo específico para dsPIC (tipo RTOS o similares) o bien, de manera más inmediata, mediante la programación en C de los algoritmos de procesamiento. De hecho, está última opción se ha empezado a explorar pero no se han incluido resultados en este trabajo por razones de espacio y de tiempo.

En otro orden de cosas, también se podrían añadir nuevas funcionalidades a la consola de administración para que permita configurar diversas características del módulo inalámbrico conectado, tales como el nombre del dispositivo o el nivel de seguridad en las comunicaciones.

Aunque las dimensiones del prototipo diseñado para el subsistema hardware son apropiadas para satisfacer los objetivos propuestos, se podrían reducir aún más utilizando componentes de montaje superficial (SMD) y eliminando la sección relativa al interfaz ICD, junto con el interruptor de multiplexación manual entre dicha interfaz y la interfaz de administración.

Finalmente, con respecto a la aplicación software, cabe destacar que hemos comprobado que la base de datos del sistema (HSQLDB) presenta tiempos de acceso excesivamente largos para nuestros propósitos. Afortunadamente, dado que su manejo ha sido implementado a través de conectores JDBC, sería relativamente sencillo sustituir HSQLDB por otro sistema gestor de base de datos. De hecho, sería interesante introducir alguno que permita tener un servidor de bases de datos independiente ejecutándose en otra máquina (mySql, Oracle) lo que mejoraría aún más las capacidades de distribución de la información que actualmente ofrece el sistema.

7.- Referencias Bibliográficas

[BISM]: Laird Technologies. *BISM II Bluetooth™ Version 2.0 Serial Module*. Disponible en <http://www.tdksys.com/dl/open/?id=162>.

[BISM2]: Laird Technologies. *AT Command Set*. Disponible en <http://www.tdksys.com/dl/open/?id=103>.

[CALLE]: CALLE PLAZA, Javier. *Electromiógrafo digital multicanal con transmisión inalámbrica vía Bluetooth™*. Tutores: Juan A. Hernández Tamames,Mª Cristina Rodríguez Sánchez. Universidad Rey Juan Carlos, Departamento de Tecnología Electrónica. Madrid, 2007.

[CARRE]: CARRETIÉ ARANGÜENA, L. (2001). *Psicofisiología*. Madrid: Ediciones Pirámide, ISBN: 84-368-1618-8.

[CCS]: Custom Computer Services Inc. *C Compiler Reference Manual*. Disponible en <http://www.ccsinfo.com/downloads/ccs_c_manual.pdf>.

[DAV]: DaVinci Surgery. *Página Web Principal*. Disponible en <http://www.davincisurgery.com/>.

[ECLI]: Eclipse. *Web Principal*. Disponible en <http://www.eclipse.org/>.

[FERR]: FERRERO CORRAL, J. M. (1993). *Bioelectrónica. Señales bioeléctricas*. Valencia: Publicaciones UPV, ISBN: 84-772-1250-3.

[ICD]: Microchip Tecnology Inc. *MPLAB® ICD 3 In-Circuit Debugger User's Guide*. Disponible en <http://www.microchip.com/Microchip.WWW.SecureSoftwareList/secsoftwaredownload.aspx?device=en537580&lang=en&ReturnURL=http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en537580#>.

[IEC1]: International Electrotechnical Commission. *Plugs and Sockets*. Disponible en <http://www.iec.ch/zone/plugsocket/>.

[IEC2]: International Electrotechnical Commission (2007). *Medical electrical equipment - Part 1-2: General requirements for basic safety and essential performance - Collateral standard: Electromagnetic compatibility - Requirements and tests*. ISBN: 2-8318-9050-0.

[JAVA]: Java. *Página Web Principal*. Disponible en <http://www.java.com/>.

[LAIRD1]: Laird Technologies. *Página Web Principal*. Disponible en <http://www.lairdtech.com/>.

[LAIRD2]: Laird Technologies. *BISDK02BI-03 Development Kit*. Disponible en <http://lairdtech.thomasnet.com/item/wifi-radio-modules/development-kit/bisdk02bi-03?&seo=110&bc=100|3001625|3001699|3001763>.

[LM1]: On Semiconductor. *LM317, NCV317 1.5 A Adjustable Output, Positive Voltage Regulator*. Disponible en <http://www.onsemi.com/pub_link/Collateral/LM317-D.PDF>.

[LM2]: Fairchild Semiconductor. *LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator*. Disponible en <http://www.fairchildsemi.com/ds/LM/LM7805.pdf>.

[MAX]: Texas Instruments. *MAX232, MAX232I. DUAL EIA-232 DRIVERS/RECEIVERS*. Disponible en <http://focus.ti.com/lit/ds/symlink/max232.pdf>.

[MCHP1]: Microchip Technology Inc. *PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers*. Disponible en <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>.

[MCHP2]: Microchip Technology Inc. *Microchip Advanced Part Selector*. Disponible en <http://www.microchip.com/maps/microcontroller.aspx>.

[MCHP3]: Microchip Technology Inc. *MPLAB® IDE User's Guide with MPLAB Editor and MPLAB SIM Simulator*. Disponible en <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002>.

[MCHP4]: Microchip Technology Inc. *dsPIC30F2011/2012/3012/3013 Data Sheet . High-Performance, 16-bit Digital Signal Controllers*. Disponible en <http://ww1.microchip.com/downloads/en/DeviceDoc/70139F.pdf>.

[MCHP5]: Microchip Technology Inc. *PICSTART Plus*. Disponible en <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010020>.

[MED1]: WOOTTON, R., CRAIG, J. and PATTERSON, V. (2006). *Introduction to telemedicine. Second edition*. London: The Royal Society of Medicine Press Ltd, ISBN: 18-531-5677-9.

[MED2]: PEREDNIA, D.A. and ALLEN, A. (1995). Telemedicine technology and clinical applications. *JAMA*, 273(6), 483-488.

[NETB]: Sun Microsystems. *NetBeans, Web Principal*. Disponible en <http://netbeans.org/>.

[OCT]: GNU Octave. *Página web principal*. Disponible en <http://www.gnu.org/software/octave/>.

[SIG1]: Bluetooth SIG. *About the Bluetooth SIG*. Disponible en <http://www.bluetooth.com/ENGLISH/SIG/Pages/default.aspx>.

[SIG2]: Bluetooth SIG. *How Bluetooth Technology Works*. Disponible en <http://www.bluetooth.com/English/Technology/Works/Pages/default.aspx>.

[ST]:ST-Electromedicina, S.A.. *Simulador / Calibrador ECG St-16*. Disponible en <http://www.stelec.com/pdf/st/Folleto%20ST-16%20v.0709.pdf>.

[WISM]: Laird Technologies. *802.11b/g Wireless LAN Module / Device Server*. Disponible en <http://www.lairdtech.com/WorkArea/linkit.aspx?LinkIdentifier=id&ItemID=2538>.