PROYECTO FIN DE CARRERA

TÍTULO: Diseño e implementación de un servicio de acceso inalámbrico a dispositivos médicos mediante H2H

AUTOR**:** Tomás Valencia Noheda

TUTOR**:** Álvaro Araujo Pinto

DEPARTAMENTO**:** Ingeniería electrónica

TRIBUNAL:

Presidente**:** Da. Ma Elena Hernando Pérez

Vocal: D. Alvaro Araújo Pinto

Secretario**:** D. Fernando Fernández Martínez

VocalSuplente**:** D. Octavio Nieto-Taladriz García

FECHA DE LECTURA**:** Madrid, de de 2016

CALIFICACION**:**

Resumen

En este proyecto de fin de carrera se ha desarrollado el software para el servicio de autentificación *Heart To Heart*(H2H). El objetivo es conseguir un método de acceso seguro a dispositivos médicos implantables que utilice claves generadas dinámicamente a partir de la señal biométrica del corazón. El sistema estará compuesto por un dispositivo de acceso que será utilizado por el usuario que quiera conseguir acceso al dispositivo medico implantable, y un dispositivo que simulará dicho dispositivo médico. Ambas plataformas hardware han sido diseñadas e implementadas en el proyecto “Diseño e implementación de una plataforma hardware para un sistema de acceso inalámbrico a dispositivos médicos mediante *Heart-To-Heart*”.

Para realizar este proyecto se ha hecho un estudio del protocolo de autentificación H2H, que se caracteriza principalmente por eliminar las claves estáticas y utilizar características físicas dinámicas, en nuestro caso la señal del corazón, para generar claves de un solo uso, de manera que no solo se incremente la seguridad si no que se evite el uso de cualquier autoridad de certificación.

A continuación, se ha realizado un proceso de diseño de los distintos módulos software que realizaran las funcionalidades necesarias para conseguir la implementación del servicio *Heart To Heart* tanto en el lado del dispositivo acceso como en el simulador de dispositivo médico.

Por último, se han realizado una serie de pruebas para comprobar el correcto funcionamiento del sistema y su resistencia a ataques maliciosos, además de su posible utilidad como herramienta de ayuda al diagnóstico. De estas pruebas realizadas se han extraído una serie de conclusiones y se han establecido varias líneas de trabajo futuro para el proyecto.

Palabras clave

Dispositivo medico implantable, protocolo H2H, seguridad biométrica (…)

Agradecimientos

Índice

Índice de figuras

Índice de tablas

Acrónimos

H2H *Heart to Heart*

ECG Electrocardiograma

IMD Implantable Medical Device

# Introducción

## Motivación

El uso de los dispositivos médicos implantables está ampliamente extendido debido a las facilidades que ofrecen a la hora de monitorizar constantemente al paciente y proporcionarle tratamiento que de otra manera seria más complejos o agresivos de aplicar. Los dispositivos médicos activos (marcapasos, bombas de insulina, etc.) normalmente disponen de módulos radio que permiten tanto la extracción de datos del paciente como la reprogramación/reajuste de dichos dispositivos de una manera no invasiva.

En la actualidad, los protocolos de comunicación son poco seguros ya que se prioriza el que el acceso sea rápido y sencillo en casos de riesgo crítico para la salud. Sin embargo, diversos estudios como el realizado en [3] han demostrado que dicho planteamiento es altamente inseguro, ya que con una formación y unos recursos “básicos” se pueden realizar ataques que permitan el acceso a los datos monitorizados e incluso desactivar o alterar el funcionamiento del dispositivo médico implantables (IMD).

Para solucionar esta deficiencia de seguridad, se plantean diversos enfoques entre los que destacan el uso de métodos criptográficos para cifrar las comunicaciones inalábrambricas y métodos para el control de acceso a los dispositivos. Dentro de este segundo grupo, se encuentran las políticas de acceso que emplean alguna señal biométrica para conseguir el acceso al IMD, de manera que el paciente debe dar su consentimiento explícito para que dicha señal se monitorizada.

Una de estas políticas de acceso, y cuya implementación será el objetivo de este proyecto, es la llamada *Heart-To-Heart*(H2H). En esta política de acceso, se emplea un dispositivo de acceso externo que monitoriza la señal del corazón durante unos segundos para extraer la información necesaria que permita la validación de este dispositivo. Una vez cumplido el periodo de monitorización, el dispositivo de acceso envía la medida al IMD para que este la compare con su propia medida y decida si el intento de acceso es legítimo.

Utilizando este método de acceso conseguimos que el paciente sea consciente de los intentos de acceso a su dispositivo y eliminamos inconvenientes de otros métodos de control de acceso como el uso de certificados, que requiere infraestructura adicional para la gestión de dichos certificados.

Por último, para evitar las situaciones de riesgo en las que sea impracticable dicha monitorización, ya sea porque el paciente este sufriendo convulsiones o porque no se pueda perder ese tiempo, el protocolo propone un modo promiscuo de acceso en el que se prima la facilidad de acceso ante la seguridad.

## Objetivos

El objetivo principal del proyecto conjunto es desarrollar e implementar un sistema electrónico portátil que permita acceder a la configuración y registros de un marcapasos simulado utilizando una política de acceso “*touch-to-access*” a través de una conexión inalámbrica.

El hardware de dicho dispositivo ha sido diseñado en el proyecto “Diseño e implementación de una plataforma hardware para un sistema de acceso inalámbrico a dispositivos médicos mediante *Heart-To-Heart*” realizado por Samuel López Asunción.

Por lo tanto, el objetivo de este proyecto es el de diseñar e implementar un servicio de acceso inalámbrico a dispositivos médicos mediante H2H. El sistema se encarga de proporcionar un método de acceso sencillo y seguro a dispositivos médicos implantables.

Dentro de este objetivo principal, pueden distinguirse los siguientes subobjetivos:

* Desarrollo de un módulo software que se encargará del tratamiento de la señal recogida por los electrodos y la extracción de la información necesaria para generar la clave.
* Desarrollo de un módulo software que controlará el acceso al dispositivo médico, bien validando la contraseña recibida o bien entrando en modo promiscuo en caso de que situación de riesgo para la salud.
* Desarrollo de un módulo software que gestionará la comunicación inalámbrica, tanto el proceso de descubrimiento de dispositivos como el de emparejamiento de los mismos
* Desarrollo de un módulo software de interfaz de usuario, lo cual permitirá al usuario comenzar el proceso de acceso de una manera sencilla e intuitiva, así como la presentación de otros datos de interés.
* Integración y pruebas de cada uno de los módulos en el sistema final.

## Planificación del desarrollo

El desarrollo del proyecto ha seguido las siguientes fases:

Análisis y definición del sistema

En esta fase se ha realizado el estudio de los trabajos en las que se basa este proyecto y la plataforma hardware en la que se va a desarrollar con el objetivo de poder definir los requisitos que tiene que cumplir la solución propuesta.

Diseño de la solución propuesta

Una vez definidos los requisitos que debe cumplir nuestro sistema, se hizo el diseño de cada uno de los módulos principales que integran el proyecto, estableciendo para cada uno de ellos los submódulos de los que estará formado y detallando las decisiones que se han tomado para conseguir que se cumplan los requisitos definidos antes.

Implementación del software del sistema

El siguiente paso en el desarrollo fue la implementación del software diseñado, lo cual implica un análisis de las distintas estrategias posibles para abordar el problema, la decisión de cual seguir y la programación de software necesario para dicha solución.

Integración y pruebas

La última fase fue la integración de cada uno de los módulos individuales y la realización una serie de pruebas para comprobar el correcto funcionamiento de todos del sistema conjunto, tanto del hardware como del software desarrollado, y su conformidad con los requisitos planteados. Por último, se han extraído una serie de conclusiones sobre la solución final implementada y se sugieren varias líneas futuras a partir del trabajo realizado.

## Estructura de la memoria

En primer lugar, se expone en esta memoria un análisis de requisitos en el que se especifican los requisitos del software de la plataforma que se deben cumplir en el sistema final.

A continuación, se describe el proceso de diseño que se ha seguido para definir los distintos módulos software que integran el proyecto y justificando las decisiones que se han tomado. Dicho software está dividido en dos prototipos, ya que durante la implementación del software del primero se detectaron problemas insalvables que obligaron a abandonar dicho prototipo.

En tercer lugar, se detalla el proceso de implementación del software de la plataforma, lo cual incluye los problemas encontrados durante la implementación del primer prototipo y los motivos por los cuales fue descartado el primer prototipo hardware.

Seguidamente se describen las pruebas realizas al sistema para determinar el correcto funcionamiento del mismo, y se añaden las conclusiones que se han extraído de dichas pruebas, así como las posibles líneas de trabajo futuro.

En la última parte de la memoria se puede encontrar, además, un apéndice con una lista de las herramientas empleadas y una lista de referencias utilizadas a lo largo de este proyecto.

# Análisis de requisitos

## Introducción

En este capítulo se analizarán los requisitos que deben cumplir los distintos módulos software. En el desarrollo del servicio de acceso H2H se deben diseñar diversos módulos que gestionen tanto la lógica del algoritmo de validación como la comunicación inalámbrica de ambos dispositivos y, en el caso del dispositivo de acceso, la interfaz de usuario.

Este proyecto está basado en el contenido del trabajo [1], por lo que los requisitos han sido definidos acorde con lo necesario para implementar su propuesta. Los motivos de la elección de este método de control de acceso frente a otros métodos son el hecho de que no se necesite ningún tipo de infraestructuras/servicios adicionales para su funcionamiento y el tipo de señal biométrica que utiliza. El primer motivo es una ventaja inmediata si comparamos este protocolo con el uso de certificados, en los que es necesaria la intervención de una entidad certificadora cada vez que se establece una comunicación. El segundo motivo se debe a la variabilidad que tiene la señal cardiaca que permite generar claves de un solo uso, lo cual supone una ventaja frente a otras características fisiológicas estáticas (huella dactilar).

A pesar de que el sistema cuenta con dos dispositivos distintos (programador y simulador de marcapasos), dado sus similitudes en el hardware se ha decidido hacer un único software y diseñarlo de manera que pueda activar/desactivar las funcionalidades que sean necesarias en cada uno de los dispositivos.

## Requisitos generales

El sistema que vamos a diseñar e implementar basa su funcionamiento en el muestreo periódico de una señal externa, la cual será procesada tanto para su uso en el algoritmo de autentificación como, en el caso del dispositivo de acceso, ser mostrada a través de la interfaz de usuario. Dicha interfaz de usuario además proporcionará una interactividad con el dispositivo, lo cual permitirá desde comenzar el procedimiento de acceso hasta la configuración de ciertos parámetros tanto del propio dispositivo como del simulador de marcapasos.

Estas características hacen que el diseño de este sistema deba hacerse teniendo en cuenta las restricciones de un sistema de tiempo real estricto, ya que un *jitter* en el muestreo de la señal puede provocar errores en la detección del ritmo cardiaco y, por lo tanto, en el algoritmo de autentificación. Por otro lado, la interfaz de usuario está sujeta a las restricciones típicas de responsividad y disponibilidad necesarias para que la experiencia de usuario sea positiva.

Por lo tanto, en el diseño de cada uno de los módulos deberá cumplir con unos requisitos de tiempo de ejecución estrictos que permitan cumplir los plazos de todas las tareas necesarias para su funcionamiento.

Adicionalmente, dado que se trata de un sistema alimentado por batería, se deberá tener en cuenta el consumo desde el punto de vista software de manera que en cada instante solo estén en funcionamiento aquellos módulos, tanto hardware como software, que sean estrictamente necesarios en ese momento.

## Servicio

### Tratamiento de la señal cardiaca

Este módulo deberá acondicionar la señal adquirida con el objetivo de extraer la información de la manera más precisa posible. Dado que el sistema hardware no cuenta con un módulo especifico de tratamiento de señal, el diseño de esta parte deberá tener especial cuidado con el tiempo de ejecución, con lo que habrá que llegar a un compromiso entre la exactitud y el tiempo de ejecución.

Dado que la señal con la que vamos a trabajar presenta unos rasgos distintivos muy marcados, es conveniente hacer una pequeña caracterización de dicha señal para entender los requisitos que vamos a imponer a este módulo.

Dejando de la caracterización fisiología de la señal, la cual esta descrita con más detalle en la memoria del proyecto (conjunto), si realizamos el análisis desde un punto de vista eléctrico, debemos tener en cuenta dos cosas: los tipos de ruido que afectan a la señal y el ancho de banda donde se sitúa la información de interés.

La mayor contribución de ruido en ese tipo de señal proviene del ruido mioelectrico (muscular) y de las imperfecciones en los electrodos, si bien como cualquier sistema electrónico está expuesto a interferencias electromagnéticas, estas a menudo están corregidas en el propio hardware (por ejemplo, el Analog Front End que utilizamos filtra la interferencia de red).

Dicho ruido se sitúa en frecuencias bajas que solapan con la propia señal del corazón, con lo cual es muy difícil de eliminar, pero hay un fenómeno que afecta especialmente y que si se puede eliminar con cierta facilidad. Este fenómeno se denomina *baseline wander*, y consiste en una variación brusca del nivel de continua debido generalmente a malos contactos de los electrodos o a espasmos musculares del paciente. Esta componente de ruido tiene concentrada su energía entre 0 y 1 Hz, con lo que se puede eliminar a través de filtrado.

Respecto al ancho de banda de información, la bibliografía existente al respecto distingue entre adultos e infantes: para el primer caso el ancho de banda está entre 1 y 150 Hz y para el segundo caso, entre 1 y 250 Hz. Esta diferencia es debido a características fisiológicas del sistema cardiaco de los infantes en las que no entraremos en detalles, pero que desde el punto de vista de nuestro análisis presentan un problema a la hora de definir nuestro ancho de banda de trabajo y también el diseño del modo promiscuo mencionado previamente.

Por último, dado que los algoritmos de detección del ritmo cardiaco diseñados para tiempo real basan su funcionamiento en la amplitud de la señal, es conveniente una caracterización de la “forma” de la señal. En la figura X (ya veremos el numero) se puede ver una muestra de esta señal con la que vamos a trabajar:



Como se puede observar, se producen una gran cantidad de eventos durante el proceso de un latido los cuales pueden ser utilizados para la detección del ritmo cardiaco. En nuestro caso, utilizamos la transición abrupta de los complejos QRS (intervalo formador por las ondas Q, R y S) para calcular la distancia temporal R-R.

Vistas las características de la señal del corazón, podemos pasar a definir los requisitos que debe cumplir las etapas de tratamiento de señal.

#### Requisitos de resolución y fase de los filtros

El filtrado digital de este sistema se encargará de eliminar la máxima cantidad de ruido proveniente de la adquisición de la señal cardiaca conservando las componentes que contienen la información fundamental para el sistema y respetando las restricciones de tiempo de ejecución que tenemos.

Los filtros elegidos deben lo más sencillos posible, ya que un mayor orden del filtro implica un mayor número de operaciones y esto impacta negativamente en el tiempo que tarda en realizarse el filtrado. Además, su fase deberá ser lo más lineal posible, ya que un retardo de grupo no constante puede distorsionar el cálculo de la frecuencia cardiaca.

Por último, hay que tener en cuenta la cuantización que van a sufrir los coeficientes, en especial si no disponemos de unidad de coma flotante hardware y tenemos que trabajar en coma fija. Esta cuantización no debe afectar a la respuesta en frecuencia de nuestros filtros (o en caso de hacerlo, debe ser compensada). Dicha respuesta en frecuencia debe ser estable y presentar un rizado mínimo en la banda de interés para que no afecte a la detección del ritmo cardiaco.

#### Algoritmos de detección de ritmo cardiaco

EL algoritmo utilizado deberá ser capaz de extraer la información de la señal cardiaca en tiempo real, ya que en el hardware dispone de una cantidad limitada de memoria y no es viable el almacenamiento de las muestras para un procesado offline.

##### El algoritmo utilizado para este sistema deberá ser capaz de extraer la información pulso a pulso ya que en la generación de la contraseña se toma la distancia entre los máximos de la onda R del ECG. Adicionalmente, con el objetivo de no acaparar los recursos del microprocesador, se evitará el uso de algoritmos que almacenen gran cantidad de muestras.

### Algoritmo de autenticación

El algoritmo de autenticación consiste a grandes rasgos en tres etapas.

En la primera etapa, ambos dispositivos monitorizan la señal del corazón y extraen una clave concatenando los cuatro últimos bits de la distancia temporal R-R, los cuales son independientes e idénticamente distribuidos.

Una vez que tienen dicha clave, calculan el espacio de posibles claves validas, dado que la lectura de la señal del corazón está sujeta a ruido y es casi imposible obtener exactamente la misma clave. Dicho cálculo se hará un umbral fijo calculado previamente *offline*, en nuestro caso con Matlab.

Cuando ya se tiene toda la información, se inicia un proceso en el que el IMD envía primero su clave con el objetivo de que el dispositivo de acceso decida si efectivamente está intentando a un IMD o impostor que intenta conseguir su clave para luego usarla para acceder al IMD. Una vez que el dispositivo de acceso ha hecho la verificación, envía su clave para que el IMD la valide y proporcione o no acceso.

Todos los detalles sobre este algoritmo y las mejoras que seguridad que supone su uso, están detalladas en [1].

Visto el procedimiento de validación, este módulo será deberá ser capaz de hacer todos los cálculos necesarios para llevar a cabo la generación de la clave y la comprobación de la validez de esta, así como el de gestionar todo el proceso de intercambio de información.

### Comunicación inalámbrica

La comunicación inalámbrica es un punto fundamental de nuestro diseño. Dado que el acceso físico a los IMD requiere de intervenciones quirúrgicas muy agresivas para el paciente, las nuevas generaciones están siendo diseñadas con una interfaz inalámbrica como requisito básico para evitar dichas intervenciones.

En nuestro caso, la interfaz no solo debe proporcionar un medio para comunicarnos con el IMD, sino que además tiene también la función de proporcionar algún método para proteger dicha comunicación, ya que, si los datos que se van a intercambiar por esta fuera fácilmente interceptarles, carecería de sentido el implementar cualquier control de acceso.

En cuanto a la inicialización de la comunicación, este módulo debe ser capaz de ser capaz de realizar una serie de funciones para que esta pueda llegar producirse. Estas funciones se pueden clasificar de la siguiente forma:

* Descubrimiento de dispositivos: En el lado del dispositivo de acceso, dado que en principio carecemos de la información concreta de cada IMD, como podría ser una dirección física a la que conectarse, la interfaz deberá ser capaz de detectar los dispositivos cercanos para poder iniciar el vínculo. En el lado del simulador de marcapasos, debe poderse configurar como descubrible.
* Vinculación de dispositivos: Una vez que se dispone de la lista de los dispositivos a los que podemos conectarnos, se debe proporcionar un método en el que el dispositivo de acceso y el dispositivo elegido establezcan un vínculo. Esta vinculación incluye el establecimiento del canal físico y el intercambio de un identificador que permita que la comunicación solo se produzca entre los dos dispositivos vinculados.
* Conexión y transmisión de datos: una vez establecidos todos los parámetros necesarios para la conexión, este módulo debe ser capaz de enviar y recibir datos a través de ese canal físico para poder intercambiar información entre ambos dispositivos.

La tecnología radio utilizada vendrá impuesta por el hardware proporcionado, y la tarea será proveer de una pila de protocolos capaz de gestionarlo. Dicha pila deberá proporcionar además de los servicios básicos para poder establecer la comunicación, algún método de protección de los datos transmitidos en por el canal físico para evitar que estos sean interceptados.

Adicionalmente, se valoran protecciones adicionales ante ataques del tipo *man in the middle,* mediante alguna capa adicional a la pila de protocolos que implemente métodos de transmisión seguros, como puede ser encriptación de los datos.

### Interfaz de usuario

El dispositivo de acceso está pensado para sea usado por un usuario que intenta acceder al IMD con el objetivo de reconfigurar o consultar ciertos datos. Por este motivo, es necesario proveer de una interfaz que le permita manejar todas estas operaciones de una forma sencilla y presentar todos esos datos sin necesidad de ningún método externo.

Por lo tanto, este módulo debe gestionar todo lo relacionado con la interacción usuario-dispositivo y con la presentación de información útil. Dado que nuestra interfaz de usuario viene impuesta por el hardware del que disponemos (LCD + touchpad resistivo), estos requisitos pueden resumirse en lo siguiente:

* Controles básicos del sistema electrónico, que permitirán el inicio de adquisición de datos, apagado por software, etc. Adicionalmente se valora la posibilidad de añadir un input de datos más complejo, como puede ser un teclado en pantalla al estilo de los Smartphone.
* Se ofrecerá la posibilidad de mostrar por pantalla el ECG capturado con el objetivo de que este dispositivo pueda tener futuros usos de ayuda al diagnóstico.
* Para iniciar la comunicación con el simulador de marcapasos, la interfaz debe ofrecer la posibilidad de seleccionar el dispositivo al que nos queremos conectar y mostrar el estado del establecimiento de la comunicación.
* Manipulación de parámetros del simulador de marcapasos, con vistas a un futuro uso de reprogramación de los IMD.

# Diseño del sistema

En esta sección se detalla el diseño del software del sistema. En primer lugar, comenzamos con un análisis de la plataforma hardware proporcionada, poniendo especial atención en el alcance de la capa de abstracción hardware, para poder determinar la flexibilidad de la que disponemos a la hora de modificar algún parámetro de un módulo hardware. Tras esto, daremos una vista general de los bloques que componen el sistema y como se relacionan con los distintos módulos hardware previamente analizados. Por último, detallaremos la funcionalidad individual de cada módulo y las decisiones que se han tomado para el diseño.

Antes de entrar en más detalle, para contextualizar el hecho que se haga referencia a dos prototipos distintos hay que decir que durante la implementación del software del primero de los prototipos se encontraron varios problemas que obligaron a descartar dicho prototipo. Estos problemas fueron causados por el funcionamiento incorrecto del multiplicador hardware y por un incorrecto dimensionamiento de la memoria RAM necesaria para la pila de protocolos (en la sección de implementación se darán más detalles). Esto hizo que en el diseño del primero de ellos solo se hayan realizado los de tratamiento de la señal cardiaca y el algoritmo de autenticación.

## Análisis de la plataforma hardware

El primer paso para el diseño del sistema es el determinar los elementos hardware de los que disponemos y poner especial cuidado en aquellos aspectos que nos impongan algún tipo de restricción, ya sea en funcionalidad, resolución o temporización. Dado que también se ofrece una capa de abstracción hardware, en el análisis obviaremos los detalles de control y configuración que cubre dicha capa de abstracción hardware y al final haremos un breve análisis de dicha capa de abstracción.

### Hardware del primer prototipo

#### Microcontrolador

La plataforma cuenta con un MSP430FR5972 con frecuencia máxima de trabajo de 16MHz y una capacidad de memoria de 64Kb de FRAM para código y 2Kb de RAM, y es utilizado para las funciones de control, calculo y comunicación con otros elementos hardware. Dicho microcontrolador cuenta con multiplicador hardware de 32 bits y carece de unidad de punto flotante, lo cual nos impone el que las operaciones aritméticas se hagan en punto fija.

En cuanto a la comunicación con los demás elementos hardware, cada elemento cuenta con su propio puerto salvo *el Analog Front End* y el *Fuel Gauge* que están conectados al mismo puerto (aunque en distintos pines), por lo que habrá que tener cuidado de reconfigurar el *pinout* del puerto adecuadamente cada vez que accedamos a cada uno. Para las operaciones de acceso directo a memoria, tenemos tres canales de *Direct Memory Access*(DMA).

Con respecto a los requisitos de tiempo real, debido a la poca cantidad de memoria RAM de la que dispone este microcontrolador, nos obliga a usar una solución basada en ejecutivo cíclico, ya que no se han encontrado sistemas operativos para sistemas empotrados que puedan funcionar con una cantidad tan pequeña de RAM.

Por último, cabe comentar que dispone una unidad de encriptación hardware (AES256) que nos evitará tener que implementar una encriptación por software.

#### Adquisición de datos

Para la digitalización de la señal cardiaca, se utiliza un Analog Front End(AFE) especializado para aplicaciones de electrocardiograma (ADS1291), con una velocidad máxima de muestreo de 8000 muestras por segundo y 24 bits por muestras, lo cual es más que suficiente para los requisitos de resolución del algoritmo. Este periférico cuenta con una línea de interrupción para indicar cuando los datos están listos (lo cual nos evitar tener que tener que hacer *polling* de la señal de *data ready*) y transmite los datos por SPI.

#### Comunicación inalámbrica

El dispositivo utilizado para la comunicación inalámbrica es el CC2560B, que es un circuito integrado que soporta BT ® clásico, lo cual nos impone el uso de la pila de protocolos para esa tecnología y nos ofrece la posibilidad de establecer comunicaciones autenticadas para la transmisión de los datos necesarios entre el dispositivo de acceso y el simulador de marcapasos. Este módulo se comunica por un puerto UART con control hardware de flujo.

#### Interfaz de usuario

Como elementos de interacción con el usuario, cuenta con tres leds de color rojo, verde y azul, y un *buzzer* para emitir avisos sonoros y visuales básicos, un interruptor para cortar la alimentación. El PWM necesario para el buzzer está generado por uno de los temporizadores del microcontrolador y los leds se controlan con entradas/salidas de propósito general.

También disponemos *display* LCD que ofrece la posibilidad de mostrar todo tipo de información visual, desde mensajes de texto hasta representación de señales digitales. Dicho display además cuenta con una pantalla táctil resistiva, lo cual nos permite implementar un intérprete de comandos para incrementar la interacción que se tiene con el sistema. El envío de comandos al sistema se hará asociando cada zona de la pantalla a cierto comando, de manera que cuando se detecte que dicha zona ha sido pulsada, se enviara el comando asociado al interprete.

El LCD no tiene un puerto serie hardware específico para él, sino que se ha tenido que implementar un puerto paralelo software mediante varios GPIOs, lo cual limita la tasa de refresco. El panel táctil se comunica por SPI y dispone de una línea de interrupción para avisar cuando hay un cambio de presión en la pantalla.

#### Control de consumo

Para este objetivo contamos con dos elementos principalmente: el primero de ellos son los modos de bajo consumo del microcontrolador, y el segundo es la disposición de *LDO’s* independientes para cada módulo, lo cual nos permite apagar de forma independiente cada elemento hardware, pudiendo así reducir el consumo sin necesidad de apagar completamente el sistema.

### Capa de abstracción hardware del primer prototipo

Esta capa de abstracción hardware está basada en los registros definidos en el *header* de nuestro micro (msp430fr5972.h) que proporciona el entorno de desarrollo de Texas Instruments Code Composer Studio.

Esta capa cuenta con funciones que permiten la configuración y el uso de los distintos módulos hardware utilizados en el sistema hardware, lo cual permite construir todo el sistema alrededor de dichas funciones.

#### Funciones de configuración

Estas funciones de configuración nos permitirán configurar los periféricos del sistema con los valores necesarios para su correcto funcionamiento e interacciona con los módulos hardware, y también nos permitirán configurar los módulos hardware con los parámetros que nosotros deseemos. Dichas funciones agrupadas por bloques son las siguientes:

* Relojes del sistema: Se encarga de configurar los registros necesarios para poder trabajar a la frecuencia escogida (8 MHz). Adicionalmente, se encargará de seleccionar las fuentes de reloj para cada una de las salidas de reloj que utiliza el microcontrolador para proporcionar el reloj de los módulos hardware que lo necesitan (AFE y módulo de radiofrecuencia)
* *Analog Front End*: Configura el puerto SPI para que se adecue a las necesidades del AFE (velocidad, flanco de lectura, etc.) y también se encarga de enviar la secuencia de comandos necesaria para configurar los registros del Analog Front End. Esta secuencia de inicialización incluye la selección de velocidad de muestreo, calibración y configuración de modo de envío de datos (envió continuo por interrupciones).
* Interfaz inalámbrica: Realiza la configuración de la UART a través de la cual nos comunicaremos con el modulo radio.
* Buzzer: Esta función configura los registros del temporizador utilizado para generar la señal modulada en ancho de pulso (PWM) y también los del que se utiliza como base de tiempos para la duración del pitido.
* Pantalla táctil: Estas funciones engloban tanto la configuración del SPI para el touchpad como la secuencia de inicialización del display.

#### Funciones de uso de periféricos

Estas funciones encapsulan las instrucciones necesarias para la interacción del microcontrolador con los módulos hardware. Estas instrucciones van desde el control de las señales de “preparado” de algunos periféricos como el SPI, de manera que se bloquea el envío hasta que dichos periféricos estén listos, hasta la gestión de las señales de control, como por ejemplo el *chip select* del AFE. Estas funciones se agrupan de la siguiente forma

* Envío/recepción de datos: Estas funciones gestionan la recepción y transmisión de datos al periférico para el que están implementadas. Todas ellas están diseñadas de manera que bloquean la ejecución hasta que se completa la operación, lo cual debe ser tenido en cuenta a la hora de la temporización. Aunque el uso es el mismo, según el periférico al que se dirijan, deben gestionar además ciertas señales:
  + AFE y panel táctil: Gestiona la activación del *chip select*
  + LCD: Gestiona la generación y temporización de todas las señales necesarias (referencia al *datasheet*)
* Activación del buzzer: Permiten la emisión de un pitido con el buzzer de una frecuencia y duración determinadas.

### Hardware del segundo prototipo

Como se ha comentado antes, fue necesario diseñar un segundo prototipo hardware que nos permitiera seguir el proyecto. Dicho hardware tiene diversas mejoras, pero para debido a que se mantiene la capa de abstracción hardware, únicamente tendremos que tener en cuenta el cambio de microcontrolador y el cambio de display.

Desde el punto de vista de diseño, el cambio de microcontrolador supone reducir las restricciones de memoria, con lo que podemos permitirnos utilizar un sistema operativo de tiempo real para cumplir los requisitos de tiempo real.

Por otro lado, el cambio de LCD no supone demasiado cambio, ya que el controlador de la pantalla táctil es el mismo y la comunicación se hace también por un puerto paralelo de 16 bits de datos.

Dado que el segundo prototipo tiene en común varios elementos, solo detallaremos las diferencias, respecto al primer prototipo.

#### Microcontrolador

La plataforma cuenta con un STM32L162VD que ofrece mayores prestaciones en diversos aspectos, algunos de los cuales son especialmente interesantes de cara al servicio.

Primero, su frecuencia de trabajo máximo es mayor (32MHz) así como su memoria para código (384 Kb de flash) y RAM (64Kb).

Por otro lado, para el control del LCD disponemos de un puerto paralelo hardware, lo cual nos va a permitir alcanzar velocidades sensiblemente mayores. En cuanto al resto de periféricos, cada uno cuenta con su propio puerto serie y además disponemos de dos controladores de DMA con siete canales cada uno lo cual nos permite asociar un canal de DMA para cada periférico que utilizamos de manera individual y facilitar de esta manera la gestión de las transferencias.

#### Interfaz de usuario

Aunque el esquema es similar (*leds* + *buzzer* + LCD con touchpad resistivo) tenemos ciertas diferencias respecto el otro prototipo.

El trio de led ha sido sustituido por un único led *RGB*, y el interruptor se ha cambiado por un pulsador conectado a una línea del *wakeup* del microcontrolador, lo que nos permitirá implementar un “apagado” por software y alguna otra funcionalidad básica, como puede ser apagar la luz de retroiluminación de la pantalla, de una manera similar al pulsador que tienen los Smartphone.

### Capa de abstracción hardware del segundo prototipo

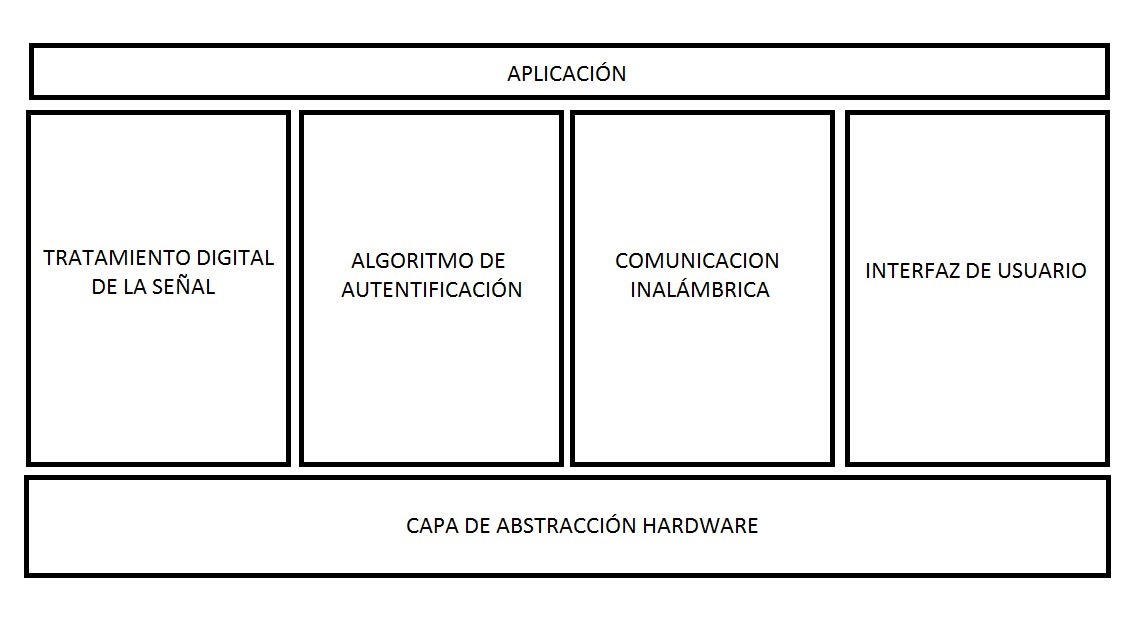
En este caso, esta capa está basada en el código de inicialización generado por el programa STM32CubeMX, el cual permite configurar todos los periféricos del microcontrolador y proporciona funciones para el uso de los puertos serie (entre otras muchas cosas).

La funcionalidad que ofrece esta nueva capa es idéntica a la del primer prototipo, salvo que en este caso no bloquean la ejecución del programa principal, ya que utilizan las funciones que ofrece el código de inicialización diseñadas para este propósito.

Este bloqueo se evita mediante el uso de la función de acceso directo a memoria disponible para los periféricos del sistema, de manera que la forma de utilizar las funciones de lectura y escritura es iniciar una transferencia y esperar la interrupción que produce el canal asociado a esta operación. Mientras se produce esta transferencia, el procesador puede seguir haciendo operaciones que no requieran el bus de datos.

## Visión general del sistema

En la siguiente figura se puede ver un esquema de la estructura del sistema software en que se detalla el nivel de separación entre la capa de abstracción hardware que proporciona el proyecto conjunto y el software que vamos a diseñar en este proyecto.

El funcionamiento de este servicio será diferente según se ejecute en el dispositivo de acceso o en el IMD, pero dado que ambas plataformas comparten la gran mayoría de la funcionalidad, se ha decidido diseñar un único sistema, el cual después incluirá o no ciertas partes según se trate del dispositivo de acceso o el simulador de marcapasos.

Debido a que este es un sistema de uso puntual, la mayor parte del tiempo se encontrara en estado de suspensión para reducir el consumo, lo cual es especialmente crítico en un hipotético IMD, donde la recarga de la batería es especialmente problemática.

Durante el acceso, el simulador de marcapasos entrará en modo escucha para poder ser detectado por el dispositivo de acceso, el cual establecerá una comunicación autenticada mediante un proceso que se detallará más adelante. Una vez el simulador de marcapasos nos haya concedido el acceso, el dispositivo ofrece una serie de opciones que simulan la manipulación de parámetros del marcapasos. Finalmente, el dispositivo cierra la conexión enviando un comando que “apaga” el modulo radio del simulador.

## Arquitectura Software del primer prototipo

La arquitectura de este sistema está basada en la capa de abstracción hardware que se nos proporciona. Nuestro objetivo será el de ampliar la funcionalidad de dicha capa para que después podamos implementar a nivel de aplicación nuestro protocolo. El software se divide en tres niveles que son los siguientes:

* *Hardware Abstraction Layer* (HAL): Implementada en el proyecto “Diseño e implementación de una plataforma hardware para un sistema de acceso inalámbrico a dispositivos médicos mediante “Heart-To-Heart” [1], consiste en una serie de archivos .c que permiten la configuración y gestión de los periféricos del microcontrolador, así como la configuración inicial del propio microcontrolador.
* Controladores: Son los módulos software expanden la funcionalidad de comunicación con los módulos hardware que se nos proporciona en la HAL. Esta ampliación será más detallada a continuación, pero consiste sobretodo en el tratamiento de los datos recibidos por parte de los periféricos que los que recibidos datos y el encapsulamiento de algunas instrucciones para una tarea concreta, como puede ser el pintar una línea en el LCD
* Aplicación: Esta capa se encarga de controlar las funciones de más alto nivel, como son el algoritmo de detección de ritmo cardiaco, la interfaz de usuario, la inicialización de la pila de protocolos, etc.

## Diseño del software del primer prototipo

### Adquisición y tratamiento de la señal

El diseño de este módulo incluye todo lo referente a la sincronización de la lectura de datos del AFE, la adaptación de dicha señal para se adecue al algoritmo de detección del ritmo cardiaco y el propio algoritmo de detección de ritmo cardiaco.

La parte de adquisición incluye una lectura de datos del AFE, el cual devuelve 3 bytes que representan el dato y los cuales son posteriormente formateados como un entero de 32 bits para poder trabajar con él. Una vez que tenemos dicho dato, se pasa a una etapa de tratado digital de la señal compuesto por una serie de filtros digitales con el objetivo que la etapa de detección de ritmo cardiaco pueda trabajar con ellos.

#### Adquisición

La lectura de los datos a través del puerto SPI se gestionará a través de la interrupción de la línea DRDY de la que dispone el AFE. Dicha línea indica en que instante están disponibles los datos leídos y está conectada a una entrada/salida de propósito general que puede configurarse como línea de interrupción externa.

Dicha interrupción se encargará leer los 24 bits de datos que envía el AFE, formatearlos los tres bytes recibidos en un único entero con signo para que sea manejable por el resto de funciones y los escribirá en un buffer circular para que sean accesibles a los otros consumidores de dichos datos. Estos consumidores serán la etapa de procesado digital y la interfaz de usuario, la cual los utilizara para la muestra del ECG en el display.

#### Procesado digital

El procesado digital tiene como objetivos eliminar la mayor cantidad de ruido que pueda afectar a señal, tanto fisiológico (respiración, ruido electromiografico, etc.) como electromagnético (interferencia de red) y también resaltar la información que facilite la detección del ritmo cardiaco. La estructura de este procesado se ha hecho atendiendo al paradigma que siguen las etapas de procesado de señal que tienen todos los algoritmos de detección de ritmo cardiaco que se han encontrado en la bibliografía.

Dicha estructura se basa en una primera etapa en la que se emplean filtros digitales para la eliminación de ruido. Esta etapa siempre empieza con un filtro paso alto para eliminar el *baseline wander* y le sigue un filtro paso bajo para eliminar el ruido de alta frecuencia que cae fuera del ancho de banda de nuestra señal. Algunos esquemas limitan el ancho de banda en el filtro paso bajo a aproximada 40 Hz, de manera que no es necesario tratamiento adicional para la eliminación de la interferencia de red, pero dado que nosotros disponemos de una etapa de filtrado en el AFE que dispone de dicha interferencia, respetaremos todo el ancho de banda. Esta señal con ruido eliminado será la que se utilice para mostrar por la interfaz de usuario.

Después de este filtrado, tenemos la etapa de resaltado de la onda R. Esta etapa sí que varía mucho entre esquema y esquema, desde alguno en que directamente no lo incluye y basa su detección en la amplitud tal cual, hasta otros que implementan incluso filtrado no lineal. Para nuestro diseño, vamos a utilizar un esquema intermedio en el que la esta etapa va a consistir en un filtro derivador.

La estructura en la que se organiza las distintas etapas de procesado se muestra en la siguiente figura:



* Filtro paso alto: Este bloque tiene dos funcionalidades: Por un lado, elimina la tensión continua necesaria para la polarización del amplificador en modo común y el fenómeno de *baseline wander* (tensión de baja frecuencia causa por un mal contacto de los electrodos que modula la señal). Dicho filtro filtrará las frecuencias entre 0 y 1 Hz, será de tipo FIR, dado estos filtros son de fase lineal, y su orden estará limitado a 200.
* Filtro paso bajo: Este bloque se encarga de eliminar de filtrar la señal por encima de los 145 Hz, ya que no contiene información útil a nuestro algoritmo y solo añade ruido. Sera de tipo FIR y el orden estará limitado a 200.
* Filtro diferenciador: Este bloque resalta las transiciones bruscas, como la del complejo QRS que es el que queremos detectar.

##### Selección del tipo de filtros

Como se ha comentado en los requisitos, la selección de filtros ha de hacerse con especial cuidado, ya que la fase de estos filtros puede distorsionar la señal y llevar a una incorrecta detección del ritmo cardiaco.

La elección más evidente para evitar esto es el uso de filtros de respuesta al impulso finita(FIR), los cuales presentan no solo la ventaja de ser de fase perfectamente lineales, sino que además son insensibles a la cuantización de los coeficientes, que era otro de los problemas que se nos planteaba. No obstante, el uso de este tipo de filtros es prácticamente inviable cuando el ancho de la transición entre la banda de paso y la eliminada en muy estrecha (como sería el caso del filtro paso alto) ya que el orden del filtro se dispara para conseguir atenuaciones moderadas. En este caso, se plantea el uso de filtro de respuesta al impulso infinita (IIR) al que se aplican técnicas de lineal izado de fase para mitigar el efecto de la distorsión de fase.

Teniendo esto en cuenta, para el filtro paso bajo y diferenciador, los cuales no tienen restricción de banda de transición estrecha, se ha seleccionado el diseño con un filtro tipo FIR con lo que conseguimos los siguientes resultados:



Figura X. Respuesta en amplitud y fase del filtro diferenciador



Figura X. Respuesta en amplitud y fase del filtro paso bajo

Como se puede observar, obtenemos la respuesta en amplitud que buscamos, tanto en banda como en rizado, el cual está limitado a 0.1 dB, y una fase perfectamente lineal. El orden del filtro paso bajo es de 140 y el del diferenciador es de 10, lo cual cumple nuestras restricciones de diseño

Para el filtro paso alto, se ha elegido usar un filtro IIR con un ecualizado de fase basado en un filtro paso todo de orden 4, el cual linealiza la fase en la banda de paso. El motivo por el cual el filtro ecualizador no se ha elegido de mayor orden ha sido porque con la perdida de precisión que supone la cuantificación de los coeficientes provoca que para un orden mayor la respuesta variara tanto que en vez de sintonizar, distorsionaba más a aun la fase. La respuesta conjunta de filtro paso alto y sintonizador es la siguiente:



En este caso, si bien la respuesta en amplitud es la buscada, la fase tiene una ligera distorsión. Esta distorsión podría ser corregida si el microcontrolador que usamos dispusiera de unidad de coma flotante hardware y por lo tanto pudiéramos diseñar un ecualizador de fase de mayor orden. No obstante, las pruebas realizadas más adelante muestran que esta pequeña distorsión de fase es aceptable para nuestra aplicación, y por lo tanto el diseño es válido.

#### Algoritmo de detección de ritmo cardiaco

Este módulo toma la señal que ha sido previamente acondicionada por la etapa de tratamiento digital para analizarla y extraer la información de ritmo cardiaco. Dado que trabaja con la información que ofrece la primera derivada de la señal cardiaca, este tipo de algoritmos se les conoce como “detección en primera derivada”.

Su principio de funcionamiento es muy sencillo: dado que la onda R es un “pico” mucho más abrupto que cualquier otra componente de la señal cardiaca, la derivada de dicha onda tendrá un valor mucho mayor que el resto, ya que su pendiente es más elevada. Con esto conseguimos que la detección sea muy sencilla, ya que el máximo de la onda R corresponde a un cero en la derivada. La complejidad por lo tanto se traslada a determinar correctamente cuando nos encontramos en un complejo QRS, para lo cual el algoritmo utiliza dos umbrales.

El primero de los umbrales se conoce como umbral alto y su objetivo es el de detectar que la señal comienza a tener una pendiente muy elevada (valor de la derivada muy alto), lo que indica que estamos entrando en una onda R. Este umbral puede ser estático o dinámico. Dado que el uso de un umbral estático requiere de una normalización de la señal que puede llegar a ser compleja de implementar, se ha decido utilizar el umbral dinámico.

El segundo umbral se conoce como umbral bajo y es simplemente una cota inferior del umbral alto. Dicha cota es necesaria para evitar que el umbral alto llegue a un nivel demasiado bajo y se identifique como complejo QRS otra transición de la señal cardiaca. Este umbral también se ha hecho dinámico por el mismo motivo que el anterior.

El cálculo de estos umbrales utiliza dos métodos diferentes durante el proceso de detección: uno de ellos se aplica en la región fuera del complejo QRS y el otro, cuando se detecta una onda R. En el primero, el umbral alto decrece geométricamente con razón igual a 0.99 hasta un valor límite del umbral bajo de manera si se produce una reducción de amplitud por cualquier motivo en un complejo QRS, el umbral se haya reducido lo suficiente como para que no se pase por alto dicho complejo y se produzca la detección. En el segundo método, se almacena el valor máximo de la derivada durante el complejo QRS y cuando se produce la detección del máximo de la onda R (paso por cero de la derivada) los valores de ambos umbrales se actualicen siguiendo las siguientes ecuaciones:

En la siguiente figura, se presenta de forma gráfica el funcionamiento. 

Finalmente, cuando se producen dos detecciones consecutivas de onda R, el modulo devuelve la distancia temporal en número de muestras para que pueda ser utilizado por el módulo de autenticación y se genere la clave a partir de esta distancia temporal.

Definición de situación de riesgo cardiaca

Como se ha comentado anteriormente, este protocolo debe incluir un modo promiscuo para las situaciones en los que la vida del paciente corra riesgo y por lo tanto sea primordial un acceso rápido por encima de un acceso seguro. Para ello, este es el modulo que puede determinar esta situación de riesgo cardiaco, ya que, al ser el encargado de detectar el ritmo cardiaco, puede determinar si hay alguna anomalía en este que pueda suponer un riesgo, como un paro cardiaco o una taquicardia.

La definición de las posibles situaciones de riesgo cardiaco detectadas a través del ECG es compleja, ya que requiere un estudio muy detallado de las posibles situaciones que no solo se refleja en una variación extrema del ritmo cardiaco, sino también por la forma de onda de la señal ECG. Debido a que dicho estudio cae fuera del alcance de este proyecto, se van a definir dos situaciones muy básicas que no requieres más que el promediado del ritmo cardiaco detectado y se dejara para líneas futuras de trabajo el definir todas las situaciones posibles de riesgo.

Estas dos situaciones son la de paro cardiaco y taquicardia. Estas son fácilmente reconocibles, ya que en el primero el corazón se para completamente y en el segundo se acelera a valores muy por encima del ritmo normal del corazón. Para detectar estas situaciones, se añadirá un promediado del ritmo cardiaco detectado por este módulo de manera que, si se detecta una situación como estas por un periodo de más de 10 segundos, el sistema pueda entrar en modo promiscuo.

### Algoritmo de autenticación

Este módulo de autenticación realiza dos funciones principalmente: la generación de la clave mediante la información de la señal cardiaca y la validación de una clave recibida.

La generación de la clave es un proceso sencillo en el cual se almacenan *n* lecturas del módulo de detección de ritmo cardiaco y se concatenan los 4 últimos bits de dichas lecturas en un numero de 4\**n* bits. El motivo de usar esto últimos cuatro bits es porque dichos bits son independientes e idénticamente distribuidos (i.i.d) y por lo tanto en caso de un ataque de fuerza bruta al protocolo, el atacante no tendría solo podría generar la clave de manera completamente aleatoria, con lo que se minimiza la probabilidad de éxito del atacante.

La validación de la clave intercambiada consiste en determinar si la clave recibida ha sido generada a partir de la misma lectura de ritmo cardiaco, se modela como un test de hipótesis cuya hipótesis. Este test se realiza aplicando el lema de Neyman-Parson, que establece que, para una probabilidad de falsa aceptación dada, la probabilidad de falso rechazo se minimiza aplicando el siguiente criterio:

(1)

En dicha ecuación, se denomina “puntuación” ya que mide como de posible es que la clave recibida sea legitima, *u* representa el vector error que contiene el conjunto de diferencias entre la clave propia y la recibida, y se calcula como un XOR entre ambas claves. P (∙) representa la probabilidad de que un impostor genere la clave que dé como resultado el vector *u* y Q (∙) la probabilidad de que un intento de acceso legitimo genere la clave que dé como resultado el vector *u*. Dado que los cuatro bits que utilizamos de cada medida <x4, x3, x2, x1> son i.i.d, la probabilidad de que *n* lecturas distintas tengan *m* fallos en el mismo bit se modela como una distribución de Bernoulli y por lo tanto dicha ecuación se puede calcular como:

(2)

La caracterización estadística de los cuatro bits que utilizamos se detallan en [1], y están definidas tanto para impostor como para acceso legítimo. Con estos valores, precalculamos el valor de y y los almacenamos en memoria, de manera que el proceso de validación se reduce a ocho sumas y una resta, lo cual es muy positivo para nuestras restricciones de tiempo real.

Para el cálculo del umbral que se usa en esta comprobación se utiliza un algoritmo que se describe en la bibliografía como “de fuerza bruta”. Dicha computación se realiza construyendo una tabla cuyas filas sean la tupla (P(u), Q(u), ) para todas las posibles combinaciones de *u* y una vez las tenemos calculadas, se ordenan de mayor a menor por valor de , de manera que la suma de los *m* primeros Q(u) sea la tasa de falso negativo y la suma de los *m* primeros P(u) sea la tasa de falso positivo para un umbral dado. El umbral entonces se selecciona como el valor de en la fila m, de manera que donde la tasa de falso rechazo se ha fijado previamente.

En resumen, el resultado del diseño de este apartado son los siguientes tres funcionalidades:

* Generación de clave de validación: Se encarga de concatenar los últimos cuatro bits n lecturas de ritmo cardiaco en una clave de 4\*n bits.
* Validación de contraseña: Verifica que la clave recibida es legitima realizando un contraste de hipótesis.
* Calculo de umbral de validación: Computa el umbral utilizado en el contraste de hipótesis mediante un método de fuerza bruta. Dicha computación no se realiza en tiempo de ejecución, sino que es un parámetro fijo precalculado.

Los detalles sobre la caracterización estadística de la información utilizada en los cálculos de este submodulo se encuentran en [1].

## Arquitectura Software del segundo prototipo

La arquitectura de este segundo prototipo es muy similar a la del primer prototipo, pero en este caso se disponía de la capacidad suficiente como para incluir un sistema operativo que simplificase la tarea de garantizar que se cumplían los requisitos de tiempo real que se definieron al principio.

El sistema operativo elegido fue el FREERTOS y los motivos por los que fue elegidos fueron dos fundamentalmente. El primero de ellos fue el hecho de que en la herramienta utiliza para generar la HAL (STM32Cubemx) se podían generar todos los archivos necesarios para usar dicho sistema operativo, con lo que se evitaban los posibles problemas de importar todos los archivos y configuraciones manualmente. El segundo fue que la pila de protocolos seleccionada, la cual se detallara en el apartado 3.6.1.1, tenía soporte para este sistema operativo y de nuevo usar este sistema operativo nos evitaba el tiempo de desarrollo que supondría adaptarlo a otro sistema operativo. Con este nuevo elemento, la arquitectura por niveles quedaría así:

* *Hardware Abstraction Layer* (HAL): Implementada en el proyecto “Diseño e implementación de una plataforma hardware para un sistema de acceso inalámbrico a dispositivos médicos mediante “Heart-To-Heart”, consiste en una serie de archivos .c que permiten la configuración y gestión de los periféricos del microcontrolador, así como la configuración inicial del propio microcontrolador.
* Sistema operativo: Este módulo engloba los controladores de los módulos hardware junto con el planificador de tareas de sistema operativo de tiempo real FREERTOS. Este sistema operativo permite la definición de tareas con espacio de memoria aislado, así como elementos de sincronización y envío de datos entre dichas tareas.
* Aplicación: Esta capa se encarga de controlar las funciones de más alto nivel, como son el algoritmo de detección de ritmo cardiaco, la interfaz de usuario, la inicialización de la pila de protocolos, etc.

## Diseño del software del segundo prototipo

En el diseño de esta parte se reutilizó el diseño del tratamiento digital de la señal y el algoritmo de autentificación, ya que estas son completamente independientes de la plataforma. Por lo tanto, el diseño para este segundo prototipo se centra en el sistema operativo, la pila de protocolos y la interfaz de usuario.

### Comunicación inalámbrica

Este módulo es el encargado de la gestión de comunicación inalámbrica. El diseño de este módulo comenzara con una descripción de la pila de protocolos utilizada, poniendo especial cuidado a aquello que afecte al cumplimiento de los requisitos que definimos previamente. Después de esto, se discutirán las opciones que de las que se dispone para la red de comunicaciones, así como que medidas de seguridad se utilizaran para proteger el canal físico ante intentos de interceptar la comunicación.

#### Pila de protocolos

Como ya se ha dicho en el análisis del hardware, la tecnología para la comunicación es BT ® clásico, en concreto la versión 4.1, para el cual Texas Instruments ofrece una pila de protocolos para Bluetooth (Bluetopia). Dicha pila dispone de implementaciones tanto para módulos de evaluación que comercializa Texas Instuments y como para familias enteras de microcontroladores entre las que se incluye los ARM-Cortex-M3, que es nuestro caso.

Esta pila está organizada en una serie de submodulos autocontenidos y de funcionamiento es completamente autónomo, esto es, que se relacionan entre ellos solamente en términos de entrada/salida y cada uno reserva su propia memoria para datos. Este diseño conlleva una potencial duplicidad de memoria y por lo tanto un mayor uso de RAM, lo cual puede ser un inconveniente importante a la hora de seleccionarla como opción para gestionar la comunicación inalámbrica. Sin embargo, se ha optado por esta opción dado que el circuito integrado que se utiliza es también de Texas Intrument (CC2560) y por lo tanto la pila está diseñada y probada para este chip, con lo que se evita que pueda surgir problemas relacionados con incompatibilidades del chip con otra pila.

La estructura de la pila es la siguiente:



Como se puede observar, la pila hace usos de diversos protocolos de Bluetooth para su funcionamiento y para cada uno de ellos ofrece una API que permite su manejo y configuración. Dado que la definición de cada protocolo está en la especificación de Bluetooth y no es el objetivo del proyecto un estudio tan profundo del mismo, lo que hemos hecho ha sido un estudio funcional de cada uno de ellos. Este es el resumen de cada uno de ellos:

* *Bluetooth Stack Controller*:Este módulo se ofrece una interfaz para la monitorización de los distintos módulos y encapsula ciertas funcionalidades, como el de inicialización de la pila. A pesar de que no está incluido en la especificación de Bluetooth, Bluetopia lo proporciona con el objetivo de simplificar el uso de la pila y ofrecer facilidades a la depuración del código.
* *Physical HCI Transport*: Se encarga de gestionar el nivel físico, es decir, el envío de datos entre el microcontrolador y el chip de Bluetooth. Esta capa necesita adaptarse para cada microcontrolador (*pinout* del puerto serie, símbolos por segundo, etc.)
* *Host Controller Interface (HCI)*: Este módulo no es un protocolo como tal, sino una capa de transporte. Establece una manera estándar de comunicación entre el chip de Bluetooth y microcontrolador, de manera que sea independiente de que microcontrolador o chip concreto se esté usando. Dicho estándar esta implementado para múltiples interfaces hardware (en nuestro caso, UART) y se utiliza para la transmisión de comandos, eventos y paquetes de datos.
* *Logicial link control and adaptation protocol (L2CAP):* Este protocolo se encarga del intercambio de paquete entre el HCI y las capas superiores. Entre sus funcionalidades, se incluye la multiplicación de múltiples aplicaciones en un solo link físico, segmentación y re ensamblado de paquetes y el control de calidad de servicio en la comunicación.
* *Radio frequency communication (RFCOMM)*: Emula un puerto serie RS-232, proporcionando canal para el envío de bits por parte de la aplicación.
* *Service discovery protocol (SDP)*: Este protocolo es el encargado de descubrir que servicios están soportados en un dispositivo remoto y que parámetros utilizar para realizar una conexión.
* *Synchronous Connection-Oriented (SCO)*: Este módulo controla el enlace con un dispositivo remoto. El tipo de enlace usado es el que indica el nombre del módulo “enlace síncrono orientado a conexión “.

La gestión de los eventos generados por cada uno de estos módulos se hace de forma asíncrona. Para ello, en momento que se inicializa la pila de protocolos, se registran una serie de funciones que a las que se llama cuando se produce un evento, llamadas *callback,* evitando así tener que implementar cualquier tipo de sincronización para el uso de pila como tal.

Además de los protocolos necesarios para la gestión de la comunicación, la pila soporta el uso de diversos perfiles. Para nuestra aplicación, haremos uso de dos de ellos, uno de ellos es el perfil de acceso genérico(GAP) y el otro es el perfil de puerto de serie (SPP).

El perfil GAP proporciona las bases para los demás perfiles y establece los pasos que deben seguirse para crear un enlace de banda base entre los dispositivos con tecnología Bluetooth. Se encarga de gestionar la detección y el establecimiento de la conexión entre dispositivos que no están conectados permitiendo que dos dispositivos con tecnología Bluetooth, independientemente de su fabricante y aplicaciones, puedan intercambiar información sobre los parámetros necesarios para establecer la comunicación.

El perfil SPP describe cómo configurar puertos de serie y conectar dos dispositivos con tecnología Bluetooth utilizando el protocolo RFCOMM para emular el puerto de serie. Reemplaza a los cables por un sustituto inalámbrico en las aplicaciones de comunicación y señales de control basadas en conexiones de serie RS-232.

#### Gestión de la red

Una vez descrita la pila y que perfiles vamos a utilizar para la comunicación, vamos a describir el proceso de establecimiento e intercambio de datos como tal.

El establecimiento de comunicación se hace a través del perfil GAP, el cual se encarga de todo el proceso que se describe en los requisitos (descubrimiento de dispositivos, vinculación y conexión). En primer lugar, es necesario configurar el simulador de marcapasos para que sea visible a otros dispositivos y que admita las peticiones de vinculación y conexión que provengan de otros dispositivos remotos.

Una vez está configurado el simulador, el dispositivo de acceso debe seguir una secuencia de acciones para que se cree el canal inalámbrico que después será utilizado por el perfil SPP para transmitir y recibir datos. Dicha secuencia es la siguiente:

* Escaneo de dispositivos visibles: el dispositivo de acceso escanea todos los canales de Bluetooth en busca de dispositivos que estén configurados como visibles y devuelve una lista con las direcciones físicas de todos aquellos dispositivos que han respondido.
* Creación de enlace: Una vez que hemos seleccionado a que dispositivo queremos conectarnos, establecemos un canal físico dedicado por el que se enviaran los datos a partir de ahora. Durante dicha creación, se intercambia una clave para identificar el enlace.
* Consulta de capacidades de entrada/salida del dispositivo: El protocolo de Bluetooth ofrece diversas maneras de autenticar una conexión según las capacidades de entrada/salida del dispositivo al que se conecte y el que inicie la conexión; por ejemplo, si ambos dispositivos disponen de un teclado, se podría utilizar un PIN de acceso (como es el caso de los teléfonos móviles).
* Confirmación de conexión: Debido a que el simulador de marcapasos no tiene ninguna capacidad de entrada salida, se establece una conexión no autenticada con cualquier dispositivo que intente conectarse a él. Este comportamiento es uno de los modos de seguridad de Bluetooth, en el que la conexión se establece sin ningún tipo de autenticación y se delega al nivel de aplicación la autenticación, que en nuestro caso se hará a través del protocolo H2H.

Una vez finalizada la conexión, comienza a funcionar el perfil SPP para establecer el puerto serie virtual por el que se intercambiaran datos. El establecer dicho puerto requiere igual que antes acciones por parte tanto del dispositivo de acceso como del simulador de marcapasos.

En primer lugar, el simulador de marcapasos debe crear un servidor SPP con un puerto serie disponible, de manera que las conexiones SPP entrantes puedan conectarse a ese servidor. Se ha elegido habilitar únicamente un puerto virtual ya que esta aplicación está pensada para que solo haya una comunicación simultanea entre el dispositivo de acceso y el IMD, por lo que habilitar más puertos solo daría la posibilidad de conexiones no deseadas.

Una vez abierto el puerto, el dispositivo de acceso debe hacer una petición a través del protocolo SDP al simulador para, entre otros parámetros, recibir el identificador del puerto serie virtual y así conectarse a él. Una vez que se haya finalizado este proceso, la aplicación puede enviar y recibir bytes a través del perfil SPP hasta que uno de los lados cierre la conexión.

#### Conexión segura

(Esto queda pendiente a si da tiempo de ver algo)

### Interfaz de usuario

En este módulo se engloba todo lo relacionado con los elementos de los que dispone el dispositivo de acceso para presentar información al usuario (leds, buzzer y pantalla LCD) como para recibir instrucciones del mismo (panel táctil). Dado que los leds y el buzzer son elementos muy simples que pueden manejarse directamente con las funciones de la HAL, el diseño se centrara en la representación de la señal cardiaca a través de la pantalla y la implementación de los menos para que el usuario pueda realizar diversas acciones como iniciar la conexión del dispositivo al marcapasos o configurar algún parámetro del dispositivo.

Se implementarán tres menús distintos para cubrir interactividad que se pretende de la interfaz de usuario:

* Menú principal: Se utiliza simplemente para acceder a los otros menús y como punto de retorno cuando se sale de alguno de los otros menús.
* Menú ECG: Se encarga de mostrar la señal de corazón y el ritmo cardiaco calculo, y adicional permite seleccionar cuál de las derivaciones cardiacas representar.
* Menú H2H: En este menú se puede iniciar el protocolo de acceso al simulador de marcapasos. A través de este menú, se realiza la búsqueda de dispositivos Bluetooth cercanos y se selecciona el dispositivo al cual conectarse para después comenzar el proceso de autenticación.
* Menú de configuración: Como su nombre indica, en dicho menú podremos modificar los parámetros de configuración del propio dispositivo.

La interacción con estos menús se hará a través del panel táctil. En cada menú se definirán ciertas áreas, a las cuales denominamos botones, asociadas a unas coordenadas del panel táctil, de manera que cuando se detecte una presión en dichas coordenadas, se dispara un evento de la máquina de estados, el cual provocara una transición de estado y una serie de acciones asociadas a dicha transición.

Por ultimo mencionar que los menús de ECG, H2H y configuración tienen en común que se puede retroceder al menú principal en cualquier momento.

#### Menú ECG

Como ya hemos comentado, en este menú se permitirá seleccionar la derivación que queramos ver y se mostrará por pantalla. La señal se dibujará en un área delimitada de manera que el eje X sea un eje de tiempos y el Y sea el eje de amplitud de la señal. En esa área se hará un barrido de izquierda a derecha en la que se ira dibujado el nivel de señal.

La señal cardiaca que vamos a mostrar al usuario proviene de la adquisición a través de AFE y su posterior tratado digital para la eliminación del ruido que puede haber presente en el momento de la adquisición. Dicha señal esta muestreada a una velocidad demasiado alta como para representar todas las muestras por pantalla, no porque no se puedan enviar los datos a la pantalla lo suficientemente rápido, sino porque el barrido se tendría que hacer demasiado rápido y no se podría ver nada de detalle de la señal. La señal por tanto debe ser submuestreada de manera que en la pantalla se puedan representar al menos cinco segundos de señal y también debe ser escalada en amplitud para ajustarse a la resolución de la zona delimitada

#### Menú H2H

A través de este menú, el usuario puede manejar todo aquello relacionado con la conexión a través de H2H con el simulador de marcapasos. En primer lugar, al acceder a este menú se inicializa la pila de protocolos de Bluetooth si es la primera vez que se accede para que se pueda usar la comunicación inalámbrica.

Una vez dentro del menú, se ofrece la posibilidad de realizar una búsqueda de dispositivos, los cuales se mostrarán por pantalla una vez para que se pueda seleccionar y se inicie el protocolo de acceso H2H, que finalizara cuando se reciba una respuesta, positiva o negativa, por parte del simulador de marcapasos.

#### Menú de configuración

Este menú está organizado en pestañas que dan acceso a la configuración de diferentes parámetros del dispositivo de acceso. Los parámetros que se contemplan en este momento son la configuración de fecha y hora del reloj de tiempo real (RTC), la calibración del panel táctil y el control de brillo de la pantalla, aunque a este esquema sería fácil añadir.

Para introducir los valores de configuración del RTC y el brillo de la pantalla se ha diseñado un método de entrada muy básico, el cual consiste en incrementar o decrementar el valor del parámetro de configuración a través de pulsaciones en unos botones que se definen para este caso en concreto y después confirmar el valor para dicho para enviarlo y que se actualice donde corresponda.

Por otro lado, la calibración del panel táctil consistirá en tocar en tres coordenadas concretas que se mostraran gráficamente por pantalla para que sea más sencillo. Una vez capturados esos puntos, la aplicación se encargará de llevar a cabo la rutina de calibración.

### Módulo de gestión de consumo

En nuestra aplicación, el control de consumo es un punto crítico especialmente en la parte de IMD, ya que en esto el cambio de batería en dichos dispositivos conlleva una cirugía en el paciente. Por ello, hemos diseñado un módulo especifico que se encarga de reducir el consumo siempre que sea posible para extender al máximo la vida de las baterías tanto en el dispositivo de acceso como en el simulador de marcapasos.

La primera medida de reducción de consumo que se plantea es lo que llamamos “apagado por software”. Dicha medida se basa en cortar la alimentación de todos los módulos hardware deshabilitando sus respectivos reguladores de tensión a través de su línea de *enable* y después poner al microcontrolador su modo de más bajo consumo(standby), de manera que el único consumo existente sea el del microcontrolador en este modo. Para entrar o salir de este modo se utiliza el pulsador del que se dispone en el hardware, el cual está conectado a una línea del microcontrolador con capacidad de despertarlo de este estado de bajo consumo, y proceso es pulsar dicho botón durante un tiempo de 2s, tras lo cual se produce el “apagado” del dispositivo.

La siguiente medida que se propone afecta solo al dispositivo de acceso y es el control de consumo de la pantalla LCD. Dado que la retroiluminación de la pantalla supone el mayor consumo de batería con diferencia, es necesario ofrecer alguna medida para reducir de alguna manera dicho consumo. La primera medida que se ofrece es el apagado de la retroiluminación de la pantalla a través del pulsador hardware solo que en este caso el tiempo de pulsación debe ser menor de dos segundos (si llegamos a los dos segundos, se activaría el apagado por software). Además de esta medida en la que se apaga por completo la pantalla, en el menú de configuración de se ofrece la posibilidad de reducir la intensidad de la retroiluminación, lo cual también reduce el consumo.

La última medida que ha diseñado es la reducción de consumo en RF. En este caso, el consumo no es tan elevado ni tan constante como el de la pantalla, pero sí que se puede dar el caso en que el dispositivo comience a recibir peticiones de otros dispositivos Bluetooth ajenos al sistema, las cuales hacen que para la respuesta la antema consuma innecesariamente batería (de hecho, esto es un tipo de ataques que pueden sufrir los IMD). Aunque la solución más evidente seria desconectar la alimentación del chip de Bluetooth y apagarlo por completo, esto presenta el problema de que durante la inicialización de la pila de protocolos se envían una serie de comandos al chip que lo configuran para su funcionamiento y si optásemos por esta solución, habría que reinicializar la pila cada vez o como gestionar de manera externa a la pila el envío de dichos comandos. Por ello, lo que se hace es enviar un comando al chip de Bluetooth que lo pone en modo bajo consumo, pero conserva la configuración.

### Sistema operativo

El sistema operativo utilizado es el FREERTOS, un sistema operativo de tiempo real para sistema empotrados. Este sistema operativo soporta distribuye bajo una licencia general pública(GLP) y soporta varias familias de microcontroladores, entre las que se encuentra la nuestra ARM-Cortex-M3. Entre las características que tiene este sistema operativo, las que nos afectan directamente al diseño son:

* Tamaño reducido del núcleo de entre 4 y 9Kbytes, rápida ejecución y añade una sobrecarga computacional mínima, lo cual encaja con la filosofía de sobrecargar los recursos del microcontrolador.
* Tiene definidos varios elementos para la sincronización entre tareas y entre una tarea y una interrupción, lo cual nos permite seguir usando interrupciones para la comunicación con los módulos hardware como el AFE. Además de sincronización, también tiene definidos elementos de comunicación entre tareas, los cuales garantizan exclusión mutua y son útiles para implementar un modelo de productor-consumidor.
* No tiene limitaciones en el número de tareas que se pueden definir ni la prioridad que se puede asignar a cada una de estas tareas.
* Existen un gran número de métodos y herramientas para monitorizar el funcionamiento del sistema operativo, lo cual permite una depuración rápida del código.

El sistema operativo reparte el tiempo de procesador entre las tareas que definimos para realizar todas las funciones necesarias en el funcionamiento del sistema. Esta distribución temporal en este sistema operativo se hace siguiendo la estrategia de planificación por prioridades fijas, de manera que siempre se ejecuta la tarea con mayor prioridad de las que están preparadas. Esta planificación está dirigida por eventos, los cuales al suceder provoca que se reevalúe la planificación y se produzca un cambio de tarea si es necesario.

Este sistema soporta diversos esquemas para la reserva de memoria, que van desde la reversa estática, en la que no es posible la liberación de dicha memoria para otros usos, hasta la reserva dinámica con coalescencia de memoria, la cual permite no solo liberar memoria reservada, sino que además permite fusionar la memoria contigua liberada en bloques más grandes de manera que se puedan reutilizar para nuevas reservas de memoria. Dicha reserva de memoria se hace en una zona de la memoria RAM que se llama “montículo”, donde se sitúan todos los elementos que necesitan un espacio de memoria constante y definido, como las tareas y los elementos de sincronización.

Por último, es necesario describir los elementos de sincronización de los que dispone el sistema operativo, ya que estos son de vital importancia en el correcto funcionamiento de nuestra aplicación.

Semáforos

El primer elemento a describir es el método de sincronización entre interrupciones y tareas. Dado que los módulos software se comunican mediante interrupciones, es necesario proporcionar algún método que indique a las tareas que se encargan de la gestión de dichos periféricos que pueden comenzar su ejecución.

De la misma manera, la transmisión y recepción de datos de los periféricos a través de DMA necesita un método para indicar que la transferencia ha sido completada. Esto se consigue utilizando la interrupción que produce el DMA cuando la transferencia ha sido finalizada de la misma manera que se utiliza la interrupción de periféricos.

El método elegido para dicha sincronización está definido dentro del sistema operativo FREERTOS y se conoce como semáforo binario. Este semáforo se puede ver como una variable de dos estados que cuando está en estado “libre” permite la ejecución de la tarea que intenta acceder a él cambiando su estado a “cerrado” y que bloquea la ejecución de una tarea si esta intenta acceder a este y se encuentra en estado “cerrado”.

Con el uso de este elemento, podemos sincronizar las tareas con las interrupciones de una manera muy sencilla, ya que cuando se produce una interrupción, se “libera” el semáforo, permitiendo que la tarea que ha sido bloqueada por el mismo vuelva a ejecutarse (siempre que no haya tareas de más prioridad en ejecución). Cuando la tarea termina las acciones e intenta volver a ejecutarse, queda bloqueada por el semáforo a la espera de que se produzca otra interrupción que reanude su ejecución. Estos semáforos se utilizan sobre todo para sincronizar tareas e interrupciones, pero se puede usar de la misma forma para la sincronización entre tareas.

Además de estos semáforos binarios, también existen los semáforos con contador, los cuales tienen un funcionamiento similar, pero en vez de tener dos estados solamente, tienen además un contador recoge cuantas veces puede utilizarse antes de bloquearse. Este contador se puede incrementar externamente hasta un límite, el cual cuando se alcanza bloquea la tarea que intenta incrementarlo, y se decrementa cada vez que una tarea intenta acceder a él para continuar su ejecución.

Por último, estos semáforos disponen de un tiempo de espera para el desbloqueo, de manera que si una tarea está esperando ser reanudada por la liberación de un semáforo con tiempo de espera y dicho tiempo se cumple, la tarea se reanuda realizando otras acciones diferentes a las que realizaría si el semáforo hubiera sido liberado por otra tarea o interrupción. Esto permite la implementación de diferentes comportamientos de una misma tarea. El tiempo de espera puede ser infinito, de manera que una tarea quede suspendida indefinidamente hasta que el semáforo es liberado.

Colas de datos

Además de la sincronización, la mayoría de tareas siguen un modelo de productor-consumidor, esto es, una tarea produce un dato que la otra necesita para su ejecución. Este intercambio de datos necesita dos elementos fundamentales: sincronización y exclusión mutua. La sincronización es necesaria ya que hasta que el producto no haya generado un dato nuevo, el productor no puede continuar su ejecución. La exclusión mutua también es necesaria, ya que hay que garantizar que mientras el productor está escribiendo el dato nuevo el productor no pueda acceder al mismo, pues se producirían inconsistencias.

Para este propósito, FREERTOS ofrece una solución compacta que se denominan “colas de datos”. Dichas colas constan básicamente de un semáforo con contador y un buffer de datos, de manera que con una única estructura se pueda implementar la sincronización, exclusión mutua e intercambio de datos. El buffer está organizado como una cola *first-in-first-out*(FIFO), que elimina el dato cuando este es extraído por el consumidor.

Su funcionamiento se divide en dos partes, la de productor y la de consumidor. Por un lado, cuando el productor intenta poner un dato en el buffer, este incrementa el contador del semáforo para evitar que ponga datos si el buffer está lleno y para reanudar al consumidor si este estaba bloqueado porque el buffer estaba vacío. Por otro lado, cuando el consumidor intenta extraer un dato, se decrementa el semáforo para evitar que se acceda a un buffer vacío.

De la misma forma que los semáforos tienen un tiempo de espera para el desbloqueo, las colas de datos también disponen de un tiempo de espera para la escritura y la lectura. De esta manera, si un consumidor intenta escribir a una cola llena que tiene definido un tiempo de espera, en lugar de bloquearse indefinidamente, al acabar dicho tiempo de espera puede continuar su ejecución con otro comportamiento. Ocurre lo mismo con un consumidor que intente leer de una cola vacía con tiempo de espera; al acabar dicho tiempo de espera, el consumidor se reanuda con un comportamiento distinto al que realizaría si hubiera recibido el dato.

#### Configuración del sistema operativo y definición de tareas.

Una vez visto a grandes rasgos el funcionamiento de FREERTOS, es necesario elegir la configuración del sistema operativo para que se adapte a nuestra aplicación. Esta configuración incluye el esquema de reserva de memoria y la definición de las tareas en términos de prioridad y tipo (esporádicas o periódicas), además de elegir las acciones que se realizaran dentro de ellas.

Para el esquema de reserva de memoria se va a emplear un esquema que permite la liberación de memoria ya que, aunque en principio no se van a crear y destruir tareas, se quiere dejar la posibilidad de un futuro usar esta funcionalidad.

Analizando el funcionamiento del sistema, se ve rápidamente que existe una tarea periódica principal, que tiene que ver con la adquisición y tratamiento digital de la señal, y varias tareas esporádicas relacionadas con eventos de interfaz de usuario o comunicación inalámbrica. Teniendo en cuenta esto vamos a clasificar las tareas en tres grupos: tareas de señal, tareas de comunicación y validación y tareas de interacción con el usuario.

Para la asignación de prioridades de las tareas se definen 5 niveles de prioridad (alta, media-alta, media, media-baja y baja) las cuales, aunque no sean equivalentes a las prioridades que usa el planificador del sistema operativo, son suficientes para el modelado.

Tareas de señal

Aunque estas tareas se podrían englobar en una sola, se ha decidido aumentar el grado de granularidad para poder controlar la asignación de prioridad a las distintas etapas que intervienen. Todas estas tareas son periódicas ya que se sincroniza con un evento periódico, el muestreo de la señal, y además siguen un modelo de productor-consumidor entre ellas: una tarea espera a que otra le proporcione un dato para comenzar a ejecutarse. Visto esto, las tareas que se definen son las siguientes:

* Adquisición: Esta tarea realizará lectura de los datos del AFE por SPI y el formateo de dichos datos, acciones que antes se realizaban en la rutina de atención a la interrupción, actuando como productor para la tarea de filtrado. La prioridad asignada a esta tarea es la más alta, pues es necesario que complete su ciclo de ejecución antes de que empiece la etapa de tratamiento de señal.
* Filtrado: Se encarga de realizar el filtrado digital de las muestras que le llegan desde la tarea de adquisición, la cual envía al finalizar a la tarea de detección de ritmo cardiaco y a la tarea de interfaz de usuario. La prioridad de ejecución es media-alta, pues, aunque debe concluir antes de que llegue la siguiente muestra, es menos prioritaria que la adquisición.
* Detección de ritmo cardiaco: Aplica el algoritmo de detección de ritmo cardiaco a las muestras procesadas y cuando se produce la detección de un intervalo RR, envía dicha información a la tarea de generación de contraseña y a la interfaz de usuario. La prioridad que se le asigna es de media-alta por el mismo motivo que en la tarea de filtrado

Tareas de comunicación

Este grupo de tareas engloba tanto las tareas propias de la comunicación inalámbrica, como la parte de autenticación del protocolo H2H tanto en la generación de la clave como en el propio proceso de validación de la clave recibida. Estas tareas serán esporádicas, pues solo se ejecutarán cuando haya algún evento externo, como un establecimiento de comunicación o una petición de acceso

* Generación de clave: Esta tarea arrancará cuando se inicie un proceso de validación. Su función es utilizar los datos que provienen de la tarea de detección de ritmo cardiaco para generar la clave de validación usada en H2H. Le asignamos prioridad media pues su ejecución debe realizarse entre dos detecciones de intervalo RR, pero no puede bloquear las tareas de adquisición y procesado de datos.
* Validación de clave: Se encarga de llevar a cabo todo el proceso de validación de la contraseña recibida. Como fuente de datos utiliza la clave generada por la tarea de generación y la clave recibida por la comunicación inalámbrica. Su prioridad es media por la misma razón que la tarea de generación
* Gestión de comunicación inalámbrica: Esta tarea se encargará de gestionar las acciones que se pretendan realizar con la comunicación inalámbrica, desde la transmisión/recepción de datos hasta el uso de ciertos comandos para realizar el proceso de descubrimiento-enlace-conexión. Le asignamos la prioridad media-baja ya que, aunque hay tareas menos prioritarias, no debe bloquear la ejecución de las tareas necesarias para el algoritmo de H2H.

Tareas de tareas de interacción con el usuario

Estas tareas son las encargadas de la interacción del dispositivo con el usuario, lo cual engloba la atención del pulsador hardware, la pantalla y el panel táctil. Estas tareas recibirán información de diversas partes, tanto del hardware como de software, y se encargarán de gestionar dicha información. Las tareas que se han definido son las siguientes:

* Atención al pulsador hardware: Esta tarea será la encargada de gestionar las acciones que son necesarias realizar cuando se pulsa el botón hardware. Se ha separado del resto para poder darle una prioridad alta, ya que en caso de querer hacer un apagado por software, no tiene sentido permitir que se terminen de ejecutar otras tareas antes de realizar dicho apagado.
* Atención al panel táctil: En esta tarea se espera a que se notifique que el panel táctil ha sido presionado y una vez se haya producido la pulsación, se lee la posición en la que se ha producido dicha pulsación. Con esta posición se buscará si en dichas coordenadas hay definido algún botón que responda a esa pulsación y se llevaran a cabo las acciones necesarias. La prioridad que se le asigna a esta tarea es la más baja, pues esta tarea no es crítica para el funcionamiento del sistema.
* Atención a la pantalla: En esta tarea se espera la recepción de algún tipo de información que haya que representar por pantalla (batería, hora, señal ECG) y se realizaran las acciones necesarias mostrar dicha información por pantalla. Esta tarea también tiene prioridad baja, pues no es crítica para el sistema.

# Implementación de los prototipos software

En este apartado se detallan la implementación de los diversos módulos software que han sido diseñados en el capítulo anterior. Para la implementación, primero se analizarán las posibles estrategias que existan para abordar la implementación, después se darán los motivos por los que se he elegido la solución concreta y por último se darán los detalles de la realización de la solución escogida.

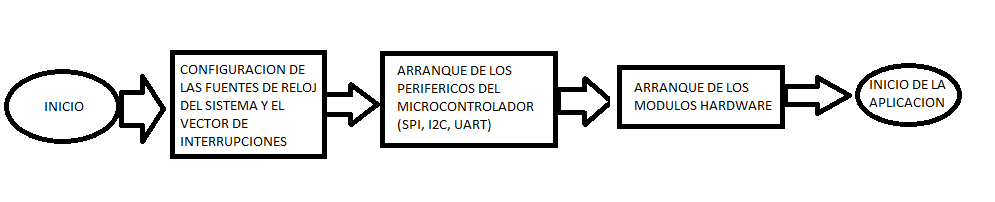
## Implementación del software de la primera versión

En primer lugar, se hará la descripción completa de la implementación de los módulos software que se llegaron a realizar para el primer prototipo. Después, se explicarán los problemas encontrados por los que se descartó este prototipo, detallando tanto el origen de dichos problemas como las soluciones que se plantearon.

### Inicialización del hardware

Para el proceso de inicialización del software se utilizan las funciones proporcionadas por la HAL para configurar todos los periféricos y los módulos hardware. La secuencia de inicio es la siguiente:

* Inicialización de bajo de nivel de la placa, lo cual incluye la configuración de la fuente de reloj para cada uno de los relojes del sistema y la configuración de los periféricos y puertos del microcontrolador.
* Arranque y configuración de los módulos hardware, lo cual se realiza a través de los puertos que hemos configurado en la etapa anterior. Esta configuración consiste en el envío de los comandos correspondientes para que los escribir en los registros de configuración de cada módulo hardware la configuración deseada.



Una vez finaliza esta inicialización, tenemos los periféricos y los módulos listos para comenzar su funcionamiento normal. En esta secuencia de inicialización no se encuentra la configuración de la pila de protocolos porque como se mencionado con anterioridad, no se llegó a hacer la implementación de dicho modulo.

### Etapas de tratamiento de la señal

#### Lectura de datos a través del AFE

Esta lectura de datos se realiza en la rutina de interrupción del puerto uno del microcontrolador. Dicha interrupción se produce cuando el AFE envía un flanco de bajada en la línea DRDY, la cual está conectada a una línea física con capacidad de interrupción.

En la interrupción, se hacen un total de seis lecturas del puerto SPI en la que se reciben 3 bytes de estado que envía el AFE en cada transmisión de datos y tres bytes de datos. Los bytes de estados son descartados y los de datos son convertidos en un entero de 32 bits con la siguiente operación:

La conversión del byte más significativo a entero de treinta y dos bits se hace para respetar el signo del dato y que se puede hacer aritmética con el mas tarde. Una vez que tenemos el dato en un formato que se puede manejar fácilmente, se pasa a la función de filtrado, la cual es un encapsulado de los diversos filtros que intervienen en el preprocesado.

Al finalizar esta rutina de interrupción, la muestra procesada se escribe en un *buffer* del cual será leída por el módulo de procesado digital de la señal. El motivo del uso de un *buffer* circular es porque se implementó con vistas a que la interfaz de usuario pudiera tomar de ese *buffer* con una frecuencia distinta a la de muestreo, pero finalmente no se llegó a implementar este comportamiento.

#### Implementación de los filtros digitales

Para la implementación de los filtros digitales, se utilizaron los coeficientes obtenidos de la herramienta de diseño de Matlab y se realizó una conversión a coma fija, ya que como hemos dicho antes, la falta de unidad de coma flotante hardware hace que los cálculos con coma flotante se tengan que hacer por software, incrementando el tiempo de ejecución y haciendo impracticable su uso para nuestra aplicación. Esos coeficientes después se almacenan en memoria de programa como constantes, lo cual hace que el acceso a estos sea un poco más lento, pero libera el espacio de memoria que ocuparían en memoria RAM, uno de nuestros recursos más limitantes.

La conversión a coma flotante consiste en la multiplicación por un factor de escalado fijo, de manera que la parte decimal se convierta en parte entera y se pueda trabajar con números enteros para los cuales el microcontrolador sí que dispone de multiplicador hardware. Una vez realizada la operación, el factor de escalado se revierte para que el resultado sea el correcto, por lo dicho factor se ha elegido como una potencia de dos para que la reversión se pueda hacer con un desplazamiento aritmético, que es una operación bastante más rápida de ejecutar que una división.

Los filtros digitales que se utilizan para el procesado de señal son de distinto tipo y por lo tanto se implementan de manera diferente. Para los filtros FIR (paso bajo y diferenciador) se ha decidido utilizar la forma directa para un filtro FIR de orden par:

N: Orden del filtro

h(i): coeficiente i-esimo del filtro

El motivo por el que se puede hacer esta operación es porque los coeficientes de los filtros FIR son simétricos. Con esta implementación, un filtro de orden N requiere un buffer de muestras de tamaño N-1, N sumas y N/2 multiplicaciones.

Para el filtro paso alto y su etapa de ecualización de fase, los cuales son ambos IIR, debido a su orden tan reducido se ha optado por utilizar directamente la fórmula que describe este tipo de filtros sin ningún tipo de optimización:

b(i): ceros del filtro

a(i): polos del filtro

Al finalizar la etapa de filtrado, se envía una señal al módulo de detección de ritmo cardiaco para indicar que hay una nueva muestra disponible a procesar.

#### Detección de ritmo cardiaco

La implementación de este algoritmo se encapsula en una única función, la cual devolverá el número de muestras entre ondas R cuando se produzca una detección y 0 el resto del tiempo para que se pueda distinguir entre detección y funcionamiento normal. En la función se distinguen tres comportamientos diferenciados, los cuales se detallan por separado.

Referencia temporal y motorización del máximo

Para el cálculo del ritmo temporal es necesario monitorizar el tiempo que pasa entre detecciones de onda R. Para ello, al principio se planteó usar un temporizador del sistema, pero se vio que era un procedimiento demasiado complejo para algo tan sencillo además que esta sufre un pequeño *jitter* si cuando se está realizando la lectura se produce una interrupción. Finalmente, se decidió usar un método mucho más sencillo que consiste en contar el número de muestras que se reciben entre detección y detección y almacenarlos en una variable, de manera que el tiempo se calcula como el cociente entre el número de muestras y la frecuencia de muestreo.

Por otro lado, es necesario monitorizar la amplitud de la señal para que cuando se produzca una detección se tenga información del valor máximo de la señal en ese intervalo y se pueda actualizar los umbrales de acorde con ello. Para este objetivo se utiliza una variable que almacenara el valor de la muestra siempre que este se superior al que tiene previamente almacenado.

Ambas variables se inicializan cada vez que se produce una detección.

Calculo de umbrales de detección

En este cálculo afrontamos un problema similar al de los coeficientes de los filtros; como no disponemos de unidad de coma flotante, no se pueden utilizar números decimales para el decrecimiento geométrico. La solución por lo tanto es utilizar una aproximación en coma fija, de manera que el coeficiente utilizado sea lo más próximo al utilizado en diseño. Esta aproximación queda de la siguiente forma

* Decrecimiento geométrico:
* Actualización de umbrales:

Por otro lado, se tiene que solucionar el cálculo inicial de los umbrales. Para este cálculo se plantearon diversas opciones, como el de definir unos umbrales iniciales fijos o el de monitorizar la señal durante un periodo de tiempo determinado y luego fijar los umbrales en función de la máxima amplitud detectada en ese periodo. Ambos esquemas presentaban problemas, el primero porque un cambio en la ganancia del AFE obliga a cambiar los umbrales y el segundo porque un “pico” de señal durante la monitorización hacia que los umbrales iniciales fueran excesivamente altos.

Finalmente, la solución que se implementó para este cálculo inicial fue el de fijar ambos umbrales a 0 y dejar que el algoritmo funcionara con normalidad. De esta manera, se producen muchas detecciones que hay que descartar, pero los umbrales se adaptan a la amplitud de la señal y tras el primer complejo QRS alcanzan los valores adecuados para funcionar.

Detección de complejo QRS

Para la detección se utiliza un método muy sencillo que se divide en dos modos de funcionamiento: modo monitorización de amplitud y modo complejo QRS.

Durante la monitorización, se comprueba si la amplitud de la señal es superior al umbral alto y cuando esto se produce se entra en modo complejo QRS poniendo el valor de un indicador a 1.

En el modo complejo QRS, se monitoriza la amplitud igualmente, pero esta vez con el objetivo detectar un paso por cero. Cuando se detecta este paso por cero, se realizan todos los cálculos necesarios (ritmo cardiaco, umbrales y reseteo de referencia temporal y máximo) y se vuelve al modo de monitorización.

Además del propio funcionamiento de detección de ritmo cardiaco, se implementa una función de detección de riesgo cardiaco, el que si se detecta el ritmo cardiaco es persistentemente mayor a 220 pulsación o si no se detecta ninguna pulsación en un periodo de 5 segundos, se envía una señal para que el dispositivo entre en modo promiscuo.

#### Implementación de algoritmo de autenticación

Esta implementación consiste de dos partes: la primera consiste en la propia validación de la clave recibida y la segunda es el cálculo de umbral de validación.

La primera parte es bastante sencilla ya que consiste en una serie de operaciones aritméticas y lógicas para extraer el valor que se utiliza para determinar si la contraseña es legitima o no: . Para dicho calculo, lo que se hace es realizar una operación XOR entre ambas claves, de manera que en las posiciones donde los bits no coinciden toman valor uno y en los que sí coinciden valen cero. Una vez obtenido el vector diferencia, se enmascara dicho vector para obtener el vector asociado a cada bit de la siguiente manera:

Con estos vectores, el cálculo del número de diferencias en cada posición de bit se hace “contando los unos” de cada uno de estos vectores. Una vez que tenemos el número de diferencias, buscamos en memoria los valores asociados a y y hacemos las operaciones necesarias para obtener la puntuación de la contraseña recibida, se compara con el umbral precalculado y decide si la contraseña es válida o no según si la puntuación es menor o mayor que el umbral.

El cálculo del umbral de validación y de y hace fuera del sistema, utilizando un script de Matlab. El script hace las siguientes operaciones:

* Calculo de y :

Donde Be (n, Pe) es el valor de la distribución de Bernoulli para n experimentos y una probabilidad de error de Pe y ei es la probabilidad de error de cada bit ([1]).

* Calculo de para todos los valores de u.
* Construcción de una matriz de manera que M(i) = { ,,}
* Ordenación de dicha matriz en orden ascendente por valor de
* Calculo del umbral de la siguiente manera:

#### Problemática encontrada e inicio de la segunda versión

Durante la implementación del software se han encontrado varios problemas y limitaciones muy importantes, los cuales han obligado a diseñar un segundo prototipo.

Las limitaciones más importantes son las siguientes:

* Interfaz de transmisión de datos a la pantalla implementada por software: El hecho de que fuera necesario controlar la temporización y el envío de datos por software, hacía que dicho envío de datos fuera extremadamente lento y consumiera mucho tiempo de ejecución ya que la transmisión era bloqueante, por lo que en última instancia la representación de la señal ECG hubiera sido muy deficiente.
* Líneas físicas del AFE multiplexadas con líneas del JTAG: debido al número limitado de líneas físicas que disponía el microcontrolador, fue necesario multiplexar físicamente líneas del AFE con líneas del JTAG, por lo que no era posible depurar el funcionamiento del AFE. Esto a su vez hizo que se perdiera mucho tiempo a la hora de identificar uno de los problemas que obligó a descartar el prototipo.

Vistas estas limitaciones, vamos a pasar a los problemas. Dichos problemas se detectaron a medida que se fue implementando el software del sistema, lo cual obligó a rehacer el hardware en una segunda versión que no solo corrigiera las limitaciones que hemos visto antes si no que no presentara los problemas.

El primero de los problemas fue la cantidad de memoria necesaria para la pila de protocolo. Los 2Kbytes de RAM de los que disponíamos eran insuficientes para implementa cualquiera de las pilas de protocolos que se encontraron y se desestimó la implementación de una pila propia, no solo porque el tiempo de desarrollo hubiera sido desproporcionado, sino porque posiblemente tampoco hubiera sido posible que el resto de la aplicación pudiera funcionar con la RAM restante. Aunque eso hubiera sido suficiente para abandonar el prototipo, se decidió seguir trabajando en él y buscar un módulo de Bluetooth que integrara la pila de protocolos dentro del propio chip, reduciendo así las necesidades de memoria para la comunicación inalámbrica.

El segundo de los problemas fue un error de diseño de las librerías software que controlaban el multiplicador. Dicho error fue muy problemático de encontrar ya que se producía cuando intentábamos operar con las muestras de AFE en los filtros, lo cual como hemos comentado antes no era posible depurar mientras se ejecutaba. El error de dicha librería era que los 16 bits más significativos del resultado de la multiplicación hardware era forzosamente almacenado en el registro de propósito general número 14 del microcontrolador, de manera que cualquier dato que hubiera en dicho registro era eliminado sin ningún tipo de guarda. Esto provoca múltiples fenómenos, desde que el resultado de la multiplicación no fuera el correcto hasta que la ejecución se parara por completo. Para solucionarlo, se probaron diversas técnicas como el no usar dicho multiplicador hardware y hacer las multiplicaciones por software, hasta tratar de usar compiladores diferentes para ver si identificaban dicho problema y dedicaban el uso de este registro únicamente a las operaciones de multiplicación. Puesto que ninguna de las soluciones propuestas conseguía solucionar el problema y teniendo en cuenta todo lo anterior, se comenzó el diseño del siguiente modelo.

## Segunda versión (persimmon)

Para la segunda versión del prototipo, se utilizó la experiencia del anterior prototipo para realizar una especificación de requisitos hardware para evitar que se dieran los problemas de falta de recursos que se encontraron en el primer prototipo.

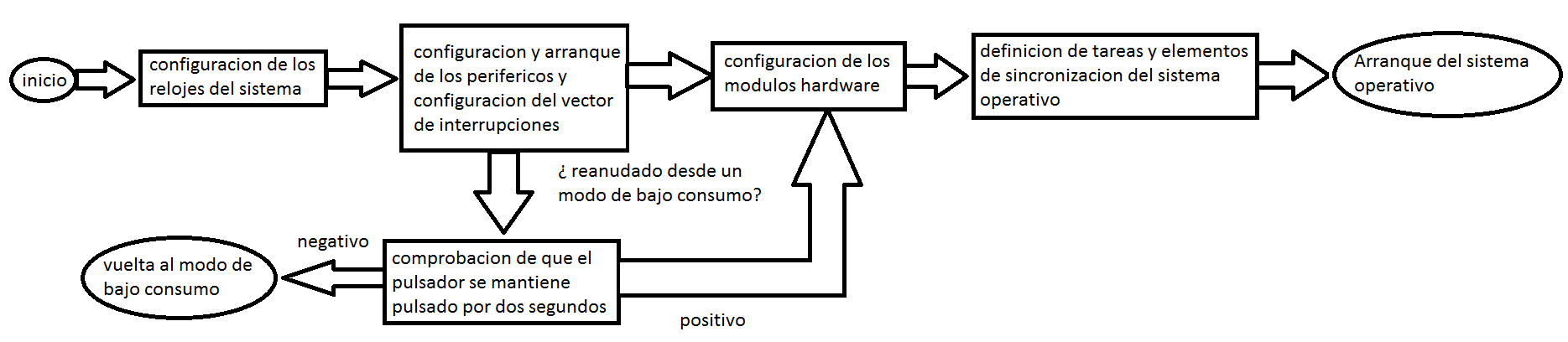
Una vez realizada la segunda versión del prototipo, el primer paso fue probar el código implementado en la primera versión para comprobar que funcionaba correctamente. Las pruebas demostraron que efectivamente dicha implementación era correcta y el que el fallo provenía del hardware anterior, por lo que se comenzó su integración con el nuevo diseño.

Dicho esto, vamos a pasar a la descripción de la implementación de los nuevos módulos software para este prototipo. En la descripción de cada uno de los módulos, se describirán tanto la implementación de la parte funcional como las tareas del sistema operativo que se utilizan para llevar a cabo dicha funcionalidad.

### Inicialización del hardware

El proceso de inicialización del software utiliza las funciones proporcionadas por la HAL para configurar todos los periféricos y los módulos hardware. La secuencia de inicio es la siguiente:

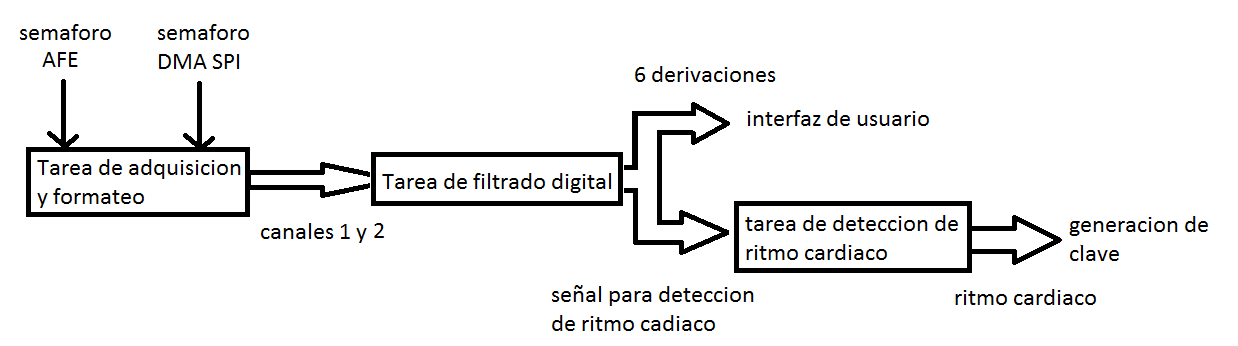
* Inicialización de bajo de nivel de la placa, lo cual incluye la configuración de la fuente de reloj para cada uno de los relojes del sistema y la configuración de los periféricos y puertos del microcontrolador. Adicionalmente, se comprueba si el microcontrolador estaba en un modo de bajo consumo antes de arrancar. Si en efecto se trata de una reanudación desde un modo de bajo consumo, se comprueba que el pulsador hardware se mantenga al menos dos segundos pulsado para continuar con el arranque. Si durante ese tiempo se libera el pulsador, el arranque se detiene y se retorna al modo bajo consumo.
* Arranque y configuración de los módulos hardware, lo cual se realiza a través de los puertos que hemos configurado en la etapa anterior. Esta configuración consiste en el envío de los comandos correspondientes para que los escribir en los registros de configuración de cada módulo hardware la configuración deseada.
* Definición de tareas y elementos de sincronización del sistema operativo, en el cual se añaden todas las tareas al planificador, se les asigna prioridad y se reserva el espacio de memoria para cada una de ellas, además de definir todos los elementos de sincronización (semáforos y colas de datos) que se van a utilizar.



Una vez finaliza esta inicialización, tenemos los periféricos y los módulos listos para comenzar su funcionamiento normal. En esta secuencia de inicialización no se encuentra la configuración de la pila de protocolos porque dicha inicialización será hecha una vez el sistema operativo haya arrancado.

### Procesado de señal

En este módulo, el código utilizado es el mismo que se empleó para el primer prototipo, por lo que vamos a pasar directamente a las tareas. Las tareas se organizan de la siguiente manera:



#### Tareas

Para realizar todo el procesado digital utilizamos un total cinco tareas, las cuales, como ya se ha comentado en el diseño, se relacionan entre sí con un modelo de productor-consumidor. Adicionalmente, la tarea de adquisición debe sincronizarse con las interrupciones que genera que el AFE.

Tarea de adquisición y formateo

Esta tarea utiliza dos semáforos para su sincronización con las interrupciones y deposita los datos leídos en dos colas distintas, las cuales serán leídas después por la tarea de filtrado.

El primero de ellos se desbloquea cuando el AFE produce una interrupción e inicia la lectura, que como indicamos en el análisis de la capa de abstracción hardware utiliza DMA para la lectura del SPI, por lo que la tarea queda suspendida a la espera de que se desbloquee el segundo semáforo.

El segundo de ellos se desbloquea cuando se produce una interrupción de transferencia terminada del DMA y entonces se inicia el formateo de los datos recibidos. El formateo de dichos datos se hace de la misma forma que la descrita para el primer prototipo, pero en este caso se realiza para los dos canales que envía el AFE.

Esta cuando finalizada todo el proceso de recepción y formateo, escribe en las colas de datos definidas para el canal uno y el canal dos del AFE y vuelve al estado suspendida hasta que se produce una nueva interrupción del AFE.

Tarea de filtrado

Esta tarea utiliza como sincronismo con su productor (la tarea de adquisición y formateo de datos) las colas de datos utilizadas para los canales uno y dos del AFE.

Cuando se reciben ambos canales, la tarea se reanuda y se aplica el filtrado previo necesario para poder mostrar la señal por la interfaz de usuario y se calculan las cuatro derivaciones adicionales que se pueden extraer de las dos derivaciones principales que medimos con el AFE (más detalles sobre estos cálculos en [5]), y enviamos las seis derivaciones a la interfaz de usuario para que luego puedan representarse por pantalla. Una vez hecho este primer procesado, aplicamos la etapa final y enviamos esta muestra a la tarea de detección de ritmo cardiaco y vuelve al estado suspendido hasta que se vuelve a recibir ambos canales.

Tarea de detección de ritmo cardiaco

Esta tarea es el consumidor de la tarea de filtrado y por tanto reanuda su ejecución cuando esta pone algún dato en la cola de señal procesada.

Al recibir una muestra nueva, aplica todo el proceso de detección de ritmo cardiaco descrito para la versión anterior (cálculo de distancia temporal RR, actualización de umbrales, etc.) y vuelve al estado suspendido hasta que recibe una nueva muestra. Cuando se produce una detección de intervalo RR, envía la información a la tarea de generación de contraseña a través de la cola definida para ese propósito.

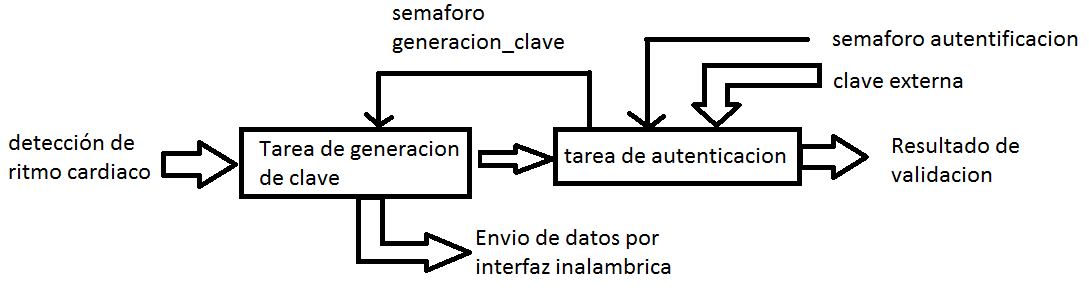
Adicionalmente, para el simulador de marcapasos, en caso de se identifique una situación de riesgo cardiaco, notifica a la tarea de autenticación que debe entrar en modo promiscuo mediante una variable global.

### Algoritmo de autentificación

Este módulo, igual que pasaba con el de procesado de señal, conserva buena parte del código que se implementó para el primer prototipo. Por lo tanto, el cambio que se realiza aquí es la utilización de tareas y elementos de sincronización para llevar a cabo las acciones necesarias para el algoritmo de autentificación.

La implementación de estas tareas se hace como si fueran tareas periódicas, ya que, en lugar de finalizar su ejecución, simplemente entran en estado suspendido. Esto obliga a utilizar un método de sincronización adicional, que coordinara la recepción de un comando por la interfaz inalámbrica con el inicio de estas tareas (los detalles sobre la parte de comunicación inalámbrica en el apartado 4.2.4)

El esquema organizativo de este módulo sería el siguiente:



Tarea de generación de contraseña

La sincronización de esta tarea es un tanto compleja ya que tiene dos comportamientos que debe realizar para el correcto funcionamiento del algoritmo. Por un lado, debe consumir los datos de la tarea de detección de ritmo cardiaco para que esta no se bloquee y por otro lado debe espera a que la tarea de autenticación le notifique puede comenzar a generar la contraseña para que pueda comenzar a funcionar.

Esta sincronización se implementa usando una limitación temporal para el semáforo que permite de la generación de la clave. De esta manera, la tarea puede estar consumiendo los datos (aunque estos sean descartados) de manera continua sin que llegue a generarse la clave. Cuando el semáforo se libera, los datos consumidos pasan a ser utilizados para la generación de la contraseña y cuando este proceso finaliza, se envía tanto a la tarea de autenticación como a la tarea de transmisión de datos.

Tarea de autenticación

De forma similar a la tarea de generación de clave, esta tarea tiene que manejar varios elementos de sincronización para llevar a cabo sus funciones.

En primer lugar, esta tarea se encuentra suspendida a la espera que la tarea de recepción Bluetooth reciba un comando que inicie el proceso de autenticación. Cuando se recibe este comando, se libera el semáforo que inicia la autenticación y esta tarea a su vez libera un semáforo que iniciara la generación de clave. Una vez liberado el semáforo, la tarea entra en estado de suspensión de nuevo a la espera de recibir tanto la clave externa a través de la tarea de recepción de Bluetooth como la clave interna a través de la tarea de generación de contraseña.

Una vez que dispone de ambas contraseñas, aplica el algoritmo de autentificación que se implementó para el primer prototipo para obtener la respuesta de si correcta la contraseña o no. Si esta respuesta es positiva, la conexión actual se marca como autenticada y si no, se fuerza una finalización de la conexión inalámbrica.

### Comunicación inalámbrica

Este módulo no fue implementado en el primer prototipo por los motivos que se mencionan en el apartado 4.1.2.5. En primer lugar, describiremos el proceso que se ha seguido para implementar la pila de protocolos que hemos utilizado y después se describirá cómo se gestiona el establecimiento de la conexión entre el dispositivo de acceso y el simulador de marcapasos

#### Pila de protocolos

La pila de protocolos elegida es la versión para ARM-Cortex-M3 de Bluetopia, que es la pila de protocolos que proporciona de forma gratuita Texas Instruments.

Esta pila necesita de ciertas adaptaciones para su funcionamiento en nuestro microcontrolador. Dichos cambios consisten en la adaptación de la capa de transporte físico y el núcleo de tiempo real que incluye en su versión adaptada para el uso con un sistema operativo.

Los cambios en la capa de transporte se basan en adaptar la implementación por defecto que se ofrece cuando descargas la pila, adaptada para la familia de microcontroladores STM32F4, a nuestro microcontrolador. Esta capa física está basada en el uso de las transferencias y recepciones de datos por UART utilizando DMA para evitar en la medida de lo posible la intervención de la CPU en la transferencia de datos entre el microcontrolador y el chip de Bluetooth. Dicha implementación utiliza una tarea del sistema operativo para atender a las interrupciones de transferencia completada del DMA, de una manera muy similar a la se ha empleado en este proyecto para atender módulos hardware.

Sin embargo, este planteamiento presenta el problema que ni las transmisiones ni las recepciones tienen un tamaño fijo lo cual obliga a utilizar una lógica un tanto compleja para analizar el número de bits que se han transmitido o recibido y así decidir si hay que continuar con la transmisión/recepción para completar el proceso.

En nuestro proyecto se ha decido simplificar toda esta lógica limitando la recepción a un único *byte*, que es la unidad mínima de intercambio entre el microcontrolador y el chip de Bluetooth. Esta decisión implica que la CPU va tener que intervenir un mayor número de veces en la trasmisión de datos, pero esta sobrecarga nos permite evitar todos los cálculos y estructuras de datos intermedias que había definidos en la pila original. Para la parte de transmisión, lo que se implementa es una transferencia bloqueante cada vez que se quiere enviar algún dato, ya que el chip de Bluetooth tiene un tiempo límite en la recepción de *bytes* consecutivos, y si se utilizaba otra forma de transmisión, en algunas situaciones se producían fallos en la comunicación lo que llevaba a un bloqueo de la pila de protocolos.

Una vez adaptada la capa de transporte físico, fue necesario adaptar el núcleo de tiempo real que utilizaba la pila de protocolos. Este núcleo estaba basado en FREERTOS como ya se ha dicho en el diseño de esta parte, por lo que los cambios a introducir han sido mínimos.

El cambio más significativo ha sido cambiar el esquema de reserva de memoria que se utilizaba en la pila original. Dicho esquema se basaba en definir un área de memoria RAM que se utilizaba de montículo, pero que no estaba gestionada directamente por el sistema operativo. Esto presentaba numerosos problemas, ya que no solo no se podían aplicar los métodos de control de montículo de los que dispone el sistema operativo, si no que la reserva de memoria era ineficiente y aumentaba innecesariamente el espacio de memoria ocupado por la pila de protocolos. El cambio que se hizo, por lo tanto, es sustituir las funciones que se utilizaban en la pila de protocolos por las funciones del sistema operativo, con lo que la gestión de memoria y declaración de tareas y elementos de sincronización está controlada directamente por el sistema operativo.

#### Gestión de la red

Para la gestión de red, el primer paso fue definir las funciones de atención a eventos de los protocolos y perfiles utilizados, los *callback*. Como se dijo en el diseño, estas funciones se encargan de realizar las acciones necesarias cada vez que se produce un evento asociado a uno de estos perfiles o protocolos.

En este módulo se utilizan el protocolo SDP para la consulta de los parámetros de conexión entre dispositivos para establecer la conexión, el perfil GAP para establecimiento de la comunicación y el perfil SPP para la transferencia de datos.

La forma de implementar estos *callback* ha sido utilizar los prototipos definidos en la especificación de la pila y después encapsular en una única función el comportamiento para todos los eventos posibles, ya disponemos de una estructura de datos que definida para cada callback en la que se incluyen el tipo y todos los datos asociados al evento. Esta idea de implementación fue tomada del código de ejemplo que proporciona la pila de protocolo y que recibe el nombre de “SPPDemo”.

Para el protocolo de SDP, la implementación del callback consiste en atender a las peticiones sobre el estado del servidor SPP, de manera que el dispositivo de acceso pueda conectar dicho servidor generado por el simulador de marcapasos. La respuesta se genera en el propio callback y consistirá en el devolver el número de puerto asociado al servido SPP abierto.

Para el perfil GAP, es necesario responder a varios eventos que se van a producir en este perfil:

* Registro de dispositivos descubierto: Una vez que finalice el proceso de descubrir nuevos dispositivos, el callback debe almacenar las direcciones físicas que se han descubierto para más tarde poder seleccionar una de ellas y conectarnos.
* Respuesta a la creación de enlace: Durante el proceso de vinculación, se establece un enlace con una clave asociada para su identificación. El callback debe encargarse de almacenar esta información del enlace para usarlo más tarde en la comunicación y después informar al dispositivo remoto que la creación de enlace ha sido aceptada. Para la confirmación se utiliza la función de la pila de protocolos *GAP\_Authentication\_Response*, la cual se encarga de construir un mensaje con la respuesta al evento de autenticación del GAP, en este caso, el de creación de enlace.
* Respuesta a la petición de capacidades de entrada/salida: Esta respuesta informa al dispositivo remoto que ha hecho la petición de las capacidades de entrada/salida del dispositivo para que se pueda determinar qué modo de autenticación de la conexión se utiliza. Para esto de nuevo se emplea la función *GAP\_Authentication\_Response,* esta vez construyendo el mensaje con la información de entrada salida del dispositivo.
* Confirmación de conexión: Como en nuestro proyecto se utiliza la conexión no autenticada debido a que el simulador de marcapasos no tiene ninguna capacidad de entrada salida, en la confirmación de conexión lo único que se hace es informar al dispositivo remoto de que la conexión ha sido aceptada utilizando la función *GAP\_Authentication\_Response.*

De manera similar al perfil GAP, el perfil SPP debe responder a los eventos asociados a su funcionalidad, que como se ha dicho anteriormente es la de transmisión y recepción de datos a través de un puerto virtual:

* Apertura de puerto: Este evento se produce cuando se abre un puerto SPP y su respuesta se utiliza para informar al perfil GAP de que existe un puerto virtual al que se pueden conectar dispositivos remotos y que por tanto debe informar al perfil SPP de peticiones de conexión a un puerto virtual. La función de comunicación entre el GAP y el SPP no ha sido implementado en el callback del GAP pues se gestiona internamente en la pila de protocolos.
* Petición a de conexión a puerto virtual: Este evento se produce cuando hay una petición de conexión al puerto virtual del SPP. La como respuesta a este evento, el perfil SPP solicita al perfil GAP el gestor de conexión del dispositivo entrante para poder usarlo en la comunicación.
* Cierre de puerto virtual: Este evento se produce cuando se cierra el servidor virtual y se informa al perfil GAP de que el puerto virtual ya no existe y que debe rechazar estas peticiones de conexión.
* Recepción de datos: Evento producido cuando se reciben datos, sirve para informar a la aplicación de la recepción de dichos datos y enviarlos a la tarea que se encarga de su gestión a través de una cola de datos.
* Finalización de transmisión de datos: Este evento se produce cuando el buffer de transmisión está vacío después de finalizar la transmisión de datos e informa a la aplicación que puede volver a enviar datos.

Una vez implementadas las funciones callback necesarias, se pasaron a implementar las funciones necesarias para el funcionamiento del sistema. Como se ha dicho antes, toda esta implementación está basada en el código de ejemplo contenido “SPPDemo” el cual ya ofrecía prácticamente todas las funciones necesarias, por lo que el trabajo de implementación consistió en modificar dichas funciones para adaptarlas a nuestra aplicación. De cara a mostrar un ejemplo de función se describe la función de inicialización. El resto son las utilizadas en la pila de protocolos.

Inicialización de la pila de protocolos

Esta función implementa la secuencia de arranque de la pila de protocolos. En primer lugar, selecciona la interfaz física que va a utilizar la capa de transporte, lo cual en nuestro caso es la UART, configurando los parámetros de velocidad de símbolo (115200 baud/s) y el control de flujo (control hardware con señal RTS y CTS). Con esta información, se invoca a la función *BSC\_Initialize*, la cual se encuentra en la librería de la pila de protocolos, y que se encarga de múltiples acciones:

* Configura la capa de transporte físico con la información proporcionada
* Envía al chip de Bluetooth la secuencia de inicio necesaria para configurar dicho chip. Este proceso debe realizarse cada vez que se corte la alimentación del chip de Bluetooth.
* Inicializa los recursos del sistema operativo que necesita la pila para funcionar (semáforos, colas, etc.)
* Inicializa los diversos protocolos Bluetooth que están implementados en la pila y configura el “nombre del dispositivo” que será el identificador que podrá consultarse a través de SDP.

Una vez que se ha completado esta tarea, la pila como tal está lista para usarse, así que ahora solo queda la configuración de los parámetros de los perfiles GAP y SPP. Como configuración del GAP, elegimos que el dispositivo sea visible y acepte peticiones de vinculación y conexión. Por último, y solamente en el simulador de marcapasos, abrimos un puerto SPP con un puerto serie para que el dispositivo de acceso pueda conectarse a ese puerto.

#### Envío y recepción de datos

Una vez implementadas las funcionalidades necesarias para establecer un canal en el que se puedan transmitir y recibir datos como de una forma sencilla, queda implementar el método que se utilizara para intercambiar dichos datos.

Como se ha dicho a lo largo de la memoria, el diseño y la implementación de este proyecto se ha hecho para que sea sencillo una futura ampliación de funcionalidades. Con esto en mente, lo que se ha hecho para la transmisión de datos es definir una unidad básica de intercambio de datos, similar a un paquete IP, consistente en un bloque de 32 bytes que contiene dos campos: una cabecera de 8bytes donde se incluye la información asociada al paquete y un campo de datos de 24 bytes. Estructura de la cabecera es la siguiente:



(estructura)

El uso de esta unidad de transmisión nos permite definir qué tipo de contenido estamos enviando. En este momento solo están definidos dos tipos, comando y clave. Los comandos son cadenas de caracteres y por el momento están definidos los siguientes:

|  |  |
| --- | --- |
| Comando | Acción |
| INICIO GENERACION | Indica al otro dispositivo que debe comenzar a generar la clave de autenticación. A este comando le sigue uno de confirmación para que el que lo ha enviado comience también a generar la clave |
| ACK GENERACION | Confirmación de la recepción del comando INICIO GENERACION |
| CONTRASEÑA LISTA | Indica que la contraseña esta lista y que la próxima transmisión será la de la clave |
| ACCESO CONCEDIDO | Informa que petición de acceso ha sido concedida |
| ACCESO RECHAZADO | Informa que petición de acceso ha sido rechazada |

Una vez definido e implementado como se va a realizar la comunicación, queda implementar a la gestión de dicha comunicación. Para esta labor vamos a utilizar dos tareas, una para recepción y otra para transmisión, que se encargaran de la lógica necesaria para generar/interpretar los paquetes.

Tarea de recepción

Esta tarea espera los paquetes que se reciben a través de SPP y que se envían por la cola de datos definida para ese propósito. Esta tarea tiene una pequeña máquina de estados simplificada, que simplemente comprueba que antes de recibir la clave se ha producido el intercambio de comandos adecuado. Dicha máquina de estados tiene un comportamiento distinto si se trata del dispositivo de acceso o del simulador de marcapasos.

(diagrama de la máquina de estados)

El funcionamiento de la máquina de estados para el dispositivo de acceso:

El funcionamiento de la máquina de estados para el simulador de marcapasos:

Tarea transmisión de datos

Esta tarea utiliza una única cola de paquetes Bluetooth, los cuales se envían directamente a través de SPP cada vez que se reciben, reintentando la transmisión siempre que el buffer de transmisión este lleno.

#### Conexión segura

(esto creo que va a quedar vacío)

### Interfaz de usuario

La interfaz de usuario tiene dos partes principales: la parte interactiva y la parte gráfica.

La parte interactiva consiste en una máquina de estado que recibe eventos desde los módulos hardware que permiten que el usuario envíe señales al sistema, es decir, el pulsador hardware y la pantalla táctil. La pantalla táctil tendrá un mediador que se encargará de interpretar el significado de las pulsaciones en la pantalla como se detallará en el apartado 4.2.5.1.

La parte grafica consiste en mostrar por la pantalla toda la información que nos llega tanto de los módulos hardware directamente como de la máquina de estados. Dicha información se formateará en las tareas que gestionan la comunicación con los periféricos y se enviaran en forma de un mapa de pixeles que pueda ser representado directamente a través de las funciones proporcionadas por la HAL para este objetivo.

Para entender el funcionamiento de esta interfaz, primero es necesario describir las estructuras de datos utilizadas.

Área

Esta estructura de datos consiste en tres elementos: una representación gráfica en la pantalla, unas coordenadas asociadas y un evento de la máquina de estados. La función de cada uno es la siguiente:

* La representación gráfica es aquello que se dibuja por pantalla para que se pueda identificar qué área es y donde tenemos que pulsar para activar el evento asociado a dicha área.
* Las coordenadas asociadas sirven tanto para situar donde pintar la representación gráfica en la pantalla como para que el intérprete de clics identifique a que área hay que acceder cuando se detecta un toque en la pantalla en unas coordenadas concretas.
* El evento asociado es el evento que la tarea de intérprete de clics enviara a la máquina de estados cuando se detecte que un área en concreto ha sido pulsada.

Gráfico

Esta estructura de datos se utiliza para representar las señales ECG por pantalla. Igual que la estructura área, el gráfico tiene asociada una representación gráfica y unas coordenadas asociadas, pero en este caso no tiene asociado ningún evento de la máquina de estados, ya que su misión es presentar información, no ofrecer interactividad.

Adicionalmente, guarda información útil para la escritura en dicho gráfico, el rango de eje x(amplitud) y el eje y(tiempo) y el índice de escritura.

Menú

Para esta aplicación, un menú es un conjunto de estructuras área y gráfico, de manera que un menú contiene tanto la información de los objetos a dibujar en la pantalla como los eventos que se pueden generar desde un menú concreto. El definir esta estructura nos permite que distintos estados de la máquina de estados tengan distintas áreas y gráficos, y que el acceso a estos sea sencillo.

Los menús implementados son los definidos en el diseño, más uno adicional de bienvenida y otro de despedida. Estos últimos son menús vacíos, ya que los estados de bienvenida y despedida son transitorios y no tienen ninguna interactividad.

#### Tareas relacionadas con la interactividad

Estas tareas son las encargadas de gestionar la comunicación con los distintos módulos hardware, interpretar la información que les llega desde estos módulos y generar tanto los eventos que se envían a la máquina de estados como otras posibles salidas que se envían directamente a la pantalla (por ejemplo, la tarea del medidor de batería muestra directamente la información por pantalla sin pasar por la máquina de estados)

(esquema de las tareas)

Tarea del panel táctil

Esta tarea gestiona la lectura del panel táctil y utiliza un como método de sincronización un semáforo que se libera cuando se produce una interrupción del mismo. Para comunicar la información a su consumidor, la tarea de gestión de clics, se define una estructura de datos *click\_t*, la cual almacena tanto la posición de dicho clic como el tipo de clic que es (presión, presión mantenida o finalización de presión)

Al reanudar su ejecución, se utilizan las funciones proporcionadas por la HAL para determinar si ha habido una pulsación o se trata de un falso positivo. En caso de que se determine que efectivamente ha habido una pulsación, se hacen lecturas periódicas de la posición de la presión hasta que se detecta que ha finalizado la pulsación, momento en el cual vuelve al estado suspendido.

Durante el tiempo que se está muestreando la posición de la presión, los clics generados se envían a la tarea que gestiona las pulsaciones a la pantalla táctil a través de una cola de datos que estos puedan ser interpretados y se generen los eventos adecuados para la máquina de estados.

Tarea de interpretación de pulsaciones en la pantalla

Como se ha indicado antes, esta tarea se encarga de interpretar las pulsaciones que se hacen en la pantalla. Dicha interpretación se va en determinar si en las coordenadas de la pulsación hay alguna área y en caso de que así sea, enviar el evento de dicha área a la máquina de estados.

Esta tarea utiliza dos elementos de sincronización: por un lado, recibe el menú actual a través de una cola de datos, evitando así que una transición entre menús se produzca alguna lectura errónea de mismo, y por otro lado se reciben los clics desde la tarea de atención al panel táctil a través de otra cola de datos. De esta manera se consigue que siempre se acceda al menú que está actualmente activo y evitemos que se generen inconsistencias en la máquina de estados.

Tarea de gestión del pulsado hardware

Esta es la tarea encarga de gestionar el comportamiento del sistema ante un evento del pulsador hardware. Esta tarea utiliza dos semáforos para detectar si se produce una pulsación de larga duración o de corta duración. Según el tipo de pulsación detectada se enviará un evento u otro a la máquina de estados.

El funcionamiento es el siguiente: cuando se detecta un flanco de subida en la línea a la que está conectada el pulsador hardware, se produce una interrupción que libera el primer semáforo (llamado semáforo de pulsado corto). Cuando se reanuda la ejecución, se reconfigura la línea de interrupción para que sea sensible a flanco de bajada y se espera al segundo semáforo con un tiempo de espera límite de dos segundos. Sin en ese periodo se produce un flanco de bajada producido porque se deja de apretar el pulsador, entonces se envía un evento a la máquina de estado que indica “pulso corto”. Si por el contrario se expira el tiempo de espera al semáforo sin que se haya producido dicho flanco, se reconfigura la línea de interrupción para que se comporte como una línea con capacidad de despertar al microcontrolador y se envía a la máquina de estados el evento “pulso largo”.

Esta tarea, por tanto, es en realidad la encargada de gestionar las dos primeras medidas de reducción de consumo propuestas en el diseño.

#### Tareas relacionadas con la presentación de información

Esta tarea se encarga de gestionar la información que llega de los periféricos, formatearla y enviarla a la tarea de la pantalla. El diagrama organizativo de estas tareas es el siguiente:

(diagrama de las tareas)

Tarea de comunicación con el medidor de batería

Esta tarea se implementa de la misma forma que las otras tareas que atienden a la comunicación de un módulo hardware con la aplicación. Se utiliza un semáforo para sincronizarse con la interrupción que genera el medidor de batería y otro semáforo para la interrupción que se genera cuando se termina la transferencia por DMA del periférico a memoria.

El funcionamiento consiste en reanudar su ejecución cuando la interrupción del medidor libera el semáforo asociado a la interrupción, iniciar la lectura por DMA del módulo hardware y volver al estado suspendido. Cuando esta lectura ha finalizado, se libera el semáforo del DMA y se reanuda la información, enviando la información de la batería formateada a la pantalla.

Dicho formateo se basa en enviar el porcentaje de batería restante y un símbolo definido en la fuente de la capa de abstracción hardware, que representa la batería.

(símbolo de la batería)

Tarea de gestión del buzzer

El funcionamiento de esta tarea es muy sencillo: simplemente espera a que llegue una “nota” a través de una cola de datos, y cuando esta se recibe utiliza una función de la capa de abstracción hardware para producir el sonido correspondiente.

Dicha “nota” es una estructura de datos que contiene tanto la frecuencia del sonido como su duración en milisegundos.

Tarea de gestión del reloj de tiempo real

De forma muy similar a la tarea del medidor de batería, esta atiende a un periférico del microcontrolador, el reloj de tiempo real, utilizando un semáforo para sincronizarse con la interrupción que produce este periférico.

El reloj de tiempo real se encarga de medir el tiempo que transcurre en una “escala humana”, esto es, fecha con información sobre días, meses y años, y hora con información de horas, minutos y segundos. Este periférico está configurado para que interrumpa cada minuto, lo cual libera el semáforo y reanuda la ejecución de esta tarea.

Cuando esto se produce, la tarea formatea la información de tiempo que lee directamente del periférico y la envía a la tarea de la pantalla para que esta la dibuje.

Tarea de gestión de la pantalla

Esta tarea se encarga de dibujar en la pantalla los objetos que le llegan a través de la cola de datos dedicada a esta función. La manera de enviar objetos a través de esta cola consiste en encapsular el objeto y la función de la capa de abstracción hardware que se encarga de dibujarla en una estructura de datos llamada *item\_action\_t*.

Esta tarea no tiene elementos adicionales de sincronización ya que son las tareas de dibujado de objeto la que gestionan el envío de datos a través del puerto paralelo.

Tarea de generación de eventos periódicos

El motivo de incluir esta tarea es debido a que la máquina de estado que hemos implementado (todos los detalles en el apartado 4.2.5.3) es asíncrona y por lo tanto no tiene capacidad de generar u evento periódico por sí misma. Este evento periódico se hace necesario para actualizar los gráficos utilizados para representar la señal cardiaca, pues se necesita de una referencia temporal para imprimir nuevas muestras en la pantalla.

La manera de implementar este evento periódico es utilizar un semáforo con tiempo de espera que no es liberado desde ningún punto, de manera que cada vez que expira el tiempo de espera se produce el evento periódico.

Tarea de gestión de eventos

Esta tarea se encarga sencillamente de recibir los eventos a través de la cola de eventos y pasárselos a la máquina de estados, la cual realiza las acciones necesarias en función de su estado y del evento que recibe.

#### Máquina de estados

(hablar con samu que maneja más esta parte y creo que tiene todos los diagramas hechos)

### Gestión de consumo

En este prototipo si ha sido posible el implementar dos de las tres medidas de reducción de consumo que se diseñaron. La tercera de las medidas no ha podido llegar a implementarse porque después del envío del comando HCI para que el chip de Bluetooth entrara en bajo consumo, no era posible que volviera al funcionamiento normal sin reiniciar el chip, lo cual hacia que perdiera la configuración inicial y no respondiera a comando enviados por la pila de protocolos.

No obstante, el apagado por software y la reducción de consumo de pantalla han sido satisfactoriamente implementadas en la tarea de atención al pulsador hardware. En dicha tarea, se generan los eventos que, una vez recibidos por la máquina de estados, hacen que la pantalla apague o encienda su retroiluminación (pulsado corto) o que entre en el modo de más bajo consumo “*standby”* (pulsado largo).

El apagado por software consigue por tanto extender la batería muy notable y además conserva la configuración del reloj de tiempo real, con lo que no es necesario reconfigurarlo cada vez que se inicia. Además de esta ventaja, el hecho para despertarse haga falta pulsar el pulsador hardware durante dos segundos seguidos, evita que se produzcan reanudaciones involuntarias.

Respecto a la reducción de consumo de la pantalla, la ventaja que obtenemos es que podemos apagar la retroalimentación de la pantalla mientras no vayamos a utilizar activamente el dispositivo y reducir así su consumo enormemente, y cuando vayamos a volver a usarlo no es necesario esperar todo el proceso de reiniciado desde el modo de bajo consumo.

# Integración y pruebas

En este capítulo describimos el proceso de integración de los módulos individuales que han sido diseñados e implementados a lo largo del proyecto. Después, se describen las pruebas hechas, tanto a los algoritmos como al sistema general, para comprobar su funcionamiento y si en efecto se ajustan a los requisitos definidos al principio

## Integración de los módulos software

El proceso de integración ha sido razonablemente sencillo, ya que al utilizar un sistema operativo y gestionar toda la lógica a través de tareas, incluir o no un módulo a la ejecución de la aplicación consistía simplemente en definir o no sus tareas y elementos de sincronización.

Dado que cada módulo ha sido diseñado para que sea autocontenido y se relacione con el resto simplemente a través de elementos del sistema operativo, las pruebas que se han podido realizar antes de la integración han permitido depurar previamente el comportamiento y han acelerado el proceso de integración.

El proceso de integración ha seguido este orden:

* Primero, se comenzó a ejecutando todas las tareas del módulo de tratamiento de señal (adquisición, filtrado, detección de ritmo cardiaco y generación de contraseña) para comprobar que la ejecución de este módulo no copaba un excesivo tiempo de ejecución, ya que como se comentó en el diseño, el tiempo de ejecución de las etapas de tratamiento digital de señal era un elemento crítico. Utilizando la herramienta *FreeRTOS+Trace from Percepio* se ha obtenido una traza de la ejecución de dichas tareas, lo cual muestra que el tiempo de ejecución está entorno al 20% del tiempo entre muestra y muestra.
* Después, se añadieron las tareas de los módulos de comunicación inalámbrica y validación, para comprobar que el funcionamiento era el esperado de ambos módulos, y la coordinación con las tareas de señal era la adecuada.
* Por último, se añadieron las tareas del módulo de interfaz de usuario, con el objetivo de comprobar que la interfaz respondía lo suficientemente rápido como cumplir con los requisitos de usabilidad en presencia de otras tareas más prioritarias.

Todos los test de integración se pasaron satisfactoriamente gracias a la depuración previa hecha a los módulos individualmente. Una vez finalizada la fase de integración, el sistema está listo para las pruebas en un escenario real.

## Pruebas del algoritmo de detección de ritmo cardiaco

El algoritmo de detección de ritmo cardiaco fue testeado y caracterizado mediante un experimento en Matlab. En dicho experimento se reconstruía el algoritmo de detección de ritmo cardiaco en lenguaje Matlab para simular el comportamiento de este algoritmo. Como entrada, se utilizaron señales cardiacas previamente capturadas y que han sido descargadas de (referencia a la web). En el experimento se han comprobado su funcionamiento en dos aspectos: primero, la exactitud con la que se detectaba el ritmo cardiaco y segundo, cuan de robusto era ante el ruido.

Para la primera parte, se calcula el ritmo cardiaco utilizando la función de Matlab que encuentra las posiciones del máximo de la señal y calculando la distancia RR como las diferencia entre dos máximos detectados con la función de Matlab. Calculando la diferencia entre el ritmo cardiaco detectado de ambas formas, no hay ninguna diferencia salvo en el transitorio inicial, donde se producen falsas detecciones, y en las señales cardiacas afectadas por alguna anomalía, como en el caso de una de las señales que registra un corazón infartado. Por lo tanto, damos por valido el algoritmo de detección de ritmo cardiaco implementado.

Para la segunda parte, se construyeron diversas señales cardiacas que, partiendo de la misma, se iba añadiendo más y más ruido para reducir la relación señal a ruido. Con este experimento se observó que la detección no comienza a degradarse fuertemente hasta una relación señal a ruido de 10dB, con lo que este sería el limite funcionamiento del algoritmo de detección.

## Pruebas del algoritmo de autenticación

Las pruebas realizadas a este algoritmo han intentado simular la situación en la que el simulador de marcapasos hace una lectura sin ruido y el dispositivo de acceso hace una lectura ruidosa.

Para ello, se han aprovechado las señales con ruido añadido que se generaron en el experimento anterior, y con ellas se han generado claves, para enfrentarlas contra la clave generada a partir de la señal sin ruido. Como cabía esperar, para una relación señal a ruido alta (> 20dB) la clave se da por válida en un 100% de los casos, y a medida que el ruido aumenta, el porcentaje de contraseñas valida se degrada muy rápidamente. Esto es debido a que para una relación señal a ruido de menos de 15 dB, el algoritmo detector de ritmo cardiaco comienza a cometer muchos fallos y por ello la generación de la clave es incorrecta.

Para simular un ataque, lo que se ha hecho es generar las claves con señales del corazón distintas. En este caso, la clave es rechazada en un 100% de los casos.

# Conclusiones y líneas futuras

En este proyecto se ha realizado el diseño y la implementación del servicio de acceso autenticado H2H. A lo largo de los capítulos anteriores, se ha descrito el proceso de desarrollo de este servicio.

En primer lugar, se ha realizado un estudio teórico del trabajo en el que está basado este proyecto, así como de los algoritmos necesarios y del hardware disponible para implementar dicho servicio, a partir de lo cual se han definido los requisitos que debe cumplir el software. Una vez realizado este estudio y definidos los requisitos, se ha realizado el diseño de todos los módulos software necesarios para poder realizar las funciones del servicio de acceso H2H y se comenzado la implementación de dicho diseño

Durante el proceso de implementación, se han detectado una serie de problemas, los cuales han sido identificados y caracterizados, y se ha tomado la decisión, junto con el proyecto [2] de realizar un segundo prototipo el solucionó esto problemas. En este segundo prototipo sí que ha sido posible llevar a cabo la implementación de todo el software diseñado, con lo que ha conseguido que el sistema, compuesto por dispositivo de acceso y simulador de marcapasos, pueda llevar a cabo el protocolo de acceso autenticado H2H.

Una vez finalizado el proceso de implementación, se ha realizado la integración de todos los módulos en el sistema final y se han realizado pruebas para comprobar que funcionan correctamente. Además de las pruebas con el sistema, se ha hecho uso de simulaciones con Matlab para comprobar el comportamiento de los algoritmos desarrollando, atendiendo sobre todo a su funcionamiento cuando la señal del corazón presenta ruido.

Las pruebas han arrojado resultados positivos y por lo tanto se considera que se han cumplido los objetivos planteados inicialmente.

A continuación, se detallan las conclusiones a las que se ha llegado tras el desarrollo, comenzando por el dispositivo de acceso y siguiendo con el simulador de marcapasos.

## Conclusiones sobre el software desarrollado

El software de este dispositivo se ha desarrollado utilizando como bases el sistema operativo FREERTOS y la capa de abstracción software proporcionado por [1] y la pila de protocolos de Bluetooth proporcionada por Texas Instrument. A partir de estas bases, se ha implementado el código necesario de forma específica para este sistema, por lo que es necesaria una valoración de los resultados obtenidos, tanto desde el punto de vista de consumo de recursos del sistema como desde el punto de vista de adecuación a los requisitos del sistema.

En primer lugar, hay que mencionar que el diseño modular que se ha realizado ha permitido no solo la identificación y la depuración rápida de los problemas que han ido surgiendo a lo largo del desarrollo, sino que además ha hecho posible que los distintos módulos puedan incluirse de una manera sencilla al sistema, lo cual es positivo de cara a una posible ampliación de las funcionalidades del sistema.

Desde el punto de vista de consumo de recursos, la utilización de la memoria de programa es correcta ya que no alcanza el 50% del máximo, pero hay un grave problema en cuanto a la ocupación de la RAM ya que esta es cercana al 99% sin optimizaciones del compilador. Esto representa que sería imposible añadir más funcionalidad al sistema tal y como esta en este momento, lo cual no es coherente con la intención de diseño que se ha seguido a lo largo del desarrollo. No obstante, se han identificado las causas de este consumo excesivo de RAM y se ha estimado que, con una optimización del código razonablemente sencilla, este consumo se puede reducir hasta el 70% de la memoria RAM (más detalles sobre esto en el apartado 6.3.2). Además, [2] se propone como línea futura el uso de un microcontrolador con más capacidad de memoria, lo cual amplia aún más el margen de RAM para la ampliación de funcionalidad.

En cuanto al nivel de adecuación a los requisitos, se ha conseguido que cada módulo software realice su función a la perfección, cumpliendo con todos los requisitos principales del sistema. No obstante, no se ha cumplido con el requisito opcional de añadir una capa de seguridad extra a la pila de protocolos, lo cual se incluirá en las líneas de trabajo futuras de este proyecto.

Para el software del simulador de marcapasos, se ha aprovechado la estructura modular del software desarrollado para reutilizar el código ya generado, incluyendo solamente el módulo de tratamiento de señal, el de algoritmo de autenticación y el de comunicación inalámbrica. La única modificación que se ha hecho ha sido en la máquina de estados, ya que esta no incluye la parta de interfaz de usuario. Aquí de nuevo vemos las ventajas del diseño modular que se ha seguido durante todo el desarrollo.

Desde el punto de vista de consumo de recursos, la memoria de programa esta utilizada a un 35% el consumo de RAM en este caso es de un 60%, muy inferior al 99% del dispositivo de acceso. Si bien esto es un punto positivo respecto al caso del dispositivo de acceso, surge la pregunta si sería suficientemente baja como para utilizarlo tal cual en un IMD real, donde la capacidad de memoria es mucho más limitada. El estudio de dicha posibilidad queda propuesta para líneas de trabajo futuras.

En cuanto al nivel de adecuación a los requisitos, de la misma forma que el software del dispositivo de acceso funciona correctamente y cumple con los requisitos, este software también lo hace pues es esencialmente el mismo código.

## Líneas futuras de trabajo

Una vez elaboradas las anteriores conclusiones, se pueden establecer una serie de líneas de trabajo futuras en las que trabajar para mejorar el sistema. El objetivo es por un lado incluir los requisitos opcionales que no han podido incluirse en este proyecto y por otro lado ampliar la funcionalidad de este sistema.

### Optimización de uso de memoria RAM

Como se ha comentado en las conclusiones, el uso de RAM en el dispositivo de acceso es prohibitivamente alto e imposibilita la ampliación de funcionalidad que se pretende realizar en este sistema.

Todo esto lleva a plantear como línea futura hacer un estudio de las posibles optimizaciones código para reducir el consumo de RAM que se puedan realizar y después realizar esas optimizaciones.

### Implementación de una capa de encriptación de datos para la comunicación inalámbrica

Como se explicó en los requisitos, es conveniente que además del protocolo de autenticación se incluya una capa de encriptación de datos para que la comunicación inalámbrica no pueda ser interceptada por un atacante remoto.

Esta posible capa de encriptación tiene además soporte hardware ya que el microcontrolador incluye un módulo de encriptación hardware de 128 bits, lo cual permite hacer estas operaciones a mayor velocidad y sin que tenga que intervenir la CPU.

Con todo esto, se propone como línea futura el estudio e implementación de una capa de encriptación de datos para el sistema.

### Detección automática situaciones de riesgo cardiaco

Este servicio está pensado para que cuando se produzca una situación de riesgo, entre en modo promiscuo para que se pueda acceder sin ningún retraso al marcapasos. No obstante, la identificación de estas situaciones de riesgo es más complejo que simplemente analizar el ritmo cardiaco.

Por lo tanto, se propone como línea futura la implementación de un módulo que monitorice la señal y detecte de forma automática y precisa situaciones de riesgo como arritmias e infartos.

### Ayuda al diagnóstico

Además de la función propia de generación de la clave para el protocolo, la adquisición de la señal cardiaca puede utilizarse para, a través de cierto procesado digital, ofrecer información útil al diagnóstico.

Entre estas posibles herramientas de ayuda se encuentran el cálculo del eje eléctrico del corazón y el análisis de la forma de onda para determinar anomalías en el corazón, las cuales se pueden llevar a cabo de forma razonablemente sencilla implementando un módulo adicional de señal que analice la señal cardiaca adquirida.

Por lo tanto, la línea de trabajo futuro que se propone es el estudio de las distintas técnicas de ayuda al diagnóstico que se pueda realizar en este sistema y su implementación en el mismo.

# Referencias

[1] “Heart-to-Heart (H2H): Authentication for Implanted Medical Devices”

[2] “Signal Processing Methods for Heart Rate Variability”

[3] Compromising a Medical Mannequin

https://es.mathworks.com/matlabcentral/fileexchange/45404-ecg-q-r-s-wave-online-detector