Universidad Carlos III de Madrid
Escuela Politécnica Superior

Grado de Ingeniería Informática
Proyecto de Fin de Grado

Blueheart: Cálculo y análisis de parámetros fisiológicos cardiovasculares

Autor: Adrián González Rus
Tutores: Juan Miguel Gómez Berbís
Jose Luis González Mora
Fecha: Febrero de 2014

«The art of medicine consists of amusing the patient while nature cures the disease.»

François-Marie Arouet, Voltaire

«Before you become too entranced with gorgeous gadgets and mesmerizing video displays, let me remind you that information is not knowledge, knowledge is not wisdom, and wisdom is not foresight. Each grows out of the other, and we need them all.»

Arthur C. Clarke

Resumen

En los últimos años, la evolución de las nuevas tecnologías en torno al desarrollo de la medicina ha ido en aumento. Los *smartphones* están ocupando un lugar imprescindible en la incorporación de este campo científico en la sociedad [1]. El uso de estas tecnologías no solo está disponible para los profesionales de la medicina, sino que se está convirtiendo en una herramienta que facilita a los ciudadanos el acceso a la consulta y resolución de dudas relacionadas con la salud [2].

En este contexto, el presente proyecto diseña un sistema que mide los niveles de saturación de oxígeno en sangre arterial, la frecuencia cardíaca y otros parámetros fisiológicos derivados de los anteriores, como el ritmo respiratorio y el estado emocional del usuario. El conjunto propuesto se compone de una aplicación que se ejecuta en el terminal Android del usuario, en el que se analiza y procesa sus constantes vitales a partir de valores medidos por un pulsioxímetro y enviados por Bluetooth.

Esta aplicación está dirigida a un colectivo específico, como deportistas de élite, personas con problemas cardiovasculares o profesionales en el área de ciencias de la salud, que necesiten vigilar con frecuencia sus parámetros fisiológicos. *Blueheart* facilita su consulta a través del *smartphone*, lo que proporciona una solución alternativa, portable y asequible económicamente a los equipos médicos que requieren la presencia del usuario en un centro especializado.

En esta memoria, se realiza una comparativa entre la tecnología empleada con otras aplicaciones similares desarrolladas en este ámbito o que han sido probadas en proyectos piloto. Se analizan los requisitos de usuario, se elabora un diseño del sistema y se implementa el prototipo de la aplicación. Por último, se documentan las pruebas comparativas con equipamiento médico homologado, y se plantea el presupuesto y las conclusiones del proyecto.

Abstract

During the last few years the evolution of new technologies around the medical field has greatly increased. The smartphones are becoming essential in the process of incorporating this scientific field into the society [1]. The use of this technology is not only available to the professionals but instead is becoming a tool that helps citizens to ask and solve questions related with their health [2].

In this context, the project designs a system capable of measuring the levels of arterial blood oxygen, heart rate and other related physiological parameters such as breath rate or the emotional state of the patient. The overall setup is composed by an Android application executed in the user terminal which analyzes and process the vital signals sent by a pulsometer using Bluetooth technology.

This application is targeted to a specific group like elite sportsmen, patients with heart problems or professionals in the area of health sciences, who need to check their vital constant with rather frequently. *Bluehearth* allows to monitorize them using the smartphone, which is an economically-suitable portable alternative to the specialized medical equipment that in most cases requires the presence of the patient in the medical center.

The report conducts a comparative between our application and other similar systems developed in this scope or that have been tested in pilot programmes. It also contains the user requirement analysis, system design elaboration and prototype application implementation. Finally the comparative test with the homologated medical equipment is documented, a budget of the project is proposed and the conclusions are presented.

Agradecimientos

Es imposible terminar este proyecto sin agradecer a todas aquellas personas que de una forma u otra han ayudado a la conclusión de esta etapa tan importante de mi vida.

En primer lugar, quiero agradecer a mi padre por darme la posibilidad de formar parte en este gran proyecto y de ayudarme a sacarlo adelante. Sin tu ayuda habría sido imposible.

A mi madre, por estar ahí en todos los momentos de crisis y por ayudarme cuando más lo he necesitado.

A mis compañeros de trabajo, sin ellos esto no habría sido lo mismo. Gracias por soportarme en mis dudas infinitas y guiarme sobre la mejor manera de llevar a cabo esta idea.

Al equipo de electrónica de la ULL, por ayudarme a finalizar una parte muy importante de este gran proyecto y por echarme un cable desde el inicio.

A todos esos amigos que han estado ahí, apoyándome en cada momento, escuchando mis problemas y dándome soluciones para todo.

A mis tutores y jefes, por recomendarme la mejor manera de realizar cada punto del proyecto, sin ellos nada de esto habría sido posible.

Por último, aunque no por ello menos importante, muchas gracias a esa persona tan especial que me ha soportado todos estos meses de agobios, por animarme y ayudarme a seguir adelante.

A todos, muchas gracias
Adrián

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción

1

Introducción

1.1. Motivación

Desde el siglo XIX [3], la evolución de la medicina ha estado íntimamente ligada a los avances tecnológicos. De esta forma, se ha acuñado el término "informática médica", para denotar el uso de las herramientas informáticas en el ejercicio de la medicina.

Desde que en 1983 [4] Motorola lanzara al mercado el que se considera el primer teléfono móvil del mundo (*Dynatac 8000x*) el sector de la telefonía móvil ha evolucionado a gran velocidad suponiendo una estupenda herramienta de trabajo diaria que permite manejar una gran cantidad de información en tiempo real. Las funcionalidades que ofrecen los teléfonos móviles han crecido rápidamente. Los primeros dispositivos únicamente permitían realizar llamadas. Posteriormente se incluyó en los terminales el servicio de mensajes cortos (del inglés, *Short Message System* - SMS). En los años siguientes se incluyeron nuevas funciones como reproducción de música, correo electrónico, agenda electrónica, fotografía digital, grabación y reproducción de vídeo digital, videollamada, sistema de posicionamiento global (del inglés *Global Positioning System* - GPS), Bluetooth, acelerómetro, giroscopio, sensor de proximidad... No obstante, la verdadera revolución en la telefonía móvil llegó de la mano de Internet móvil.

Con la incorporación de Internet en los móviles, cambió el concepto de la telefonía y empezaron a comercializarse los *smartphones* que se conocen en la actualidad. Estos nuevos dispositivos con su gran variedad de sistemas operativos (Android, iOS, Windows Phone, Symbian, RIM...), sus nuevas características y su mejor rendimiento abrieron la posibilidad de desarrollar

todo tipo de aplicaciones móviles que aprovecharan estas ventajas y mejoraran la experiencia del usuario.

En los últimos años el mercado de las aplicaciones móviles ha crecido exponencialmente estimándose que durante 2013 se incrementarán sus ingresos desde 1.411 millones de euros hasta 11.496 millones de euros, lo que supone un aumento del 807 % [5]. Existe una amplia variedad de categorías de aplicaciones: desde herramientas de productividad, de monitorización del dispositivo, de fotografía, de educación, juegos, hasta herramientas de adquisición y canje de tiques de cualquier tipo.

Por otra parte, en el año 1998 surgió lo que se conocen actualmente como la tecnología Bluetooth, creada gracias a un grupo formado por cinco compañías [6]. Ya en el año 2000 apareció el primero teléfono móvil con dicha tecnología, así como las primeras tarjetas de PC y los primeros prototipos de ratón y teclados inalámbricos. Con el paso del tiempo el número de aplicaciones y dispositivos que han hecho uso de esta herramienta ha ido en aumento.

Los desarrolladores aprovechan todas las funcionalidades que ofrecen los *smartphones* para crear nuevas aplicaciones que les permitan innovar en los aspectos más comunes de la vida cotidiana, como por ejemplo la reproducción de música haciendo uso de las posibilidades inalambricas o el control remoto de dispositivos en el hogar.

En el ámbito de la medicina, existe una gran variedad de desarrollos que incluyen el uso del sensor fotoeléctrico de la cámara del *smartphones* (CCD) para realizar diferentes análisis de parámetros fisiológicos. Sin embargo, los resultados obtenidos por estas aplicaciones no son fiables desde un punto de vista médico.

El sistema que se describe en este proyecto consiste en una aplicación que permite recuperar, analizar y representar los datos recibidos por un dispositivo de medición de parámetros fisiológicos en un terminal Android conectado por Bluetooth. Se ha tenido especiamente en cuenta conseguir una alta fiabilidad en la adquisición de datos, de manera que los resultados se ajusten a los obtenidos a partir de equipamiento médico homologado.

Los parámetros fisiológicos que se analizan con la aplicación son el porcentaje de saturación de oxígeno en sangre arterial, la frecuencia cardíaca y respiratoria, y el estado emocional del paciente, lo que permite realizar cierto grado de evaluación de patologías cardiopulmonares.

1.2. Objetivos

El objetivo principal del proyecto es permitir el acceso al análisis y evaluación de determinadas características fisiológicas del usuario sin la necesidad de utilizar equipo altamente cualificado ubicado e un centro médico.

Para alcanzar este objetivo se realizarán las siguientes tareas:

- Estudio del estado del arte, así como de los sistemas existentes y sus características (aplicaciones y/o proyectos piloto en evaluación).
- Planteamiento del sistema.
- Adquisición de los conocimientos necesarios sobre el sistema operativo Android, así como de las tecnologías necesarias para el desarrollo del proyecto.
- Análisis de los requisitos de usuario para el desarrollo de la aplicación.
- Diseño y definición del sistema.
- Implementación de un prototipo.
- Pruebas, evaluaciones y presupuesto.

1.3. Metodología

Una metodología de desarrollo de software consiste en una forma de trabajo que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de software en los sistemas de la información.

En la actualidad existen dos grandes corrientes de metodologías de desarrollo de software. Por un lado, las denominadas metodologías tradicionales, centradas en el control del proceso con un riguroso seguimiento de las actividades involucradas. Por otro lado, las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo, y un incesante incremento de software con iteraciones muy cortas.

En este proyecto, al realizarse por una sola persona, las metodologías tradicionales para el desarrollo de proyectos de software quedan descartadas puesto que están diseñadas para grandes equipos de trabajo y no se podrían adaptar de forma adecuada. Estas metodologías se caracterizan por su estructura rígida y poco flexible a la hora de realizar cambios, el uso de documentos como método de intercambio de ideas y un control prácticamente total de todas las etapas de desarrollo.

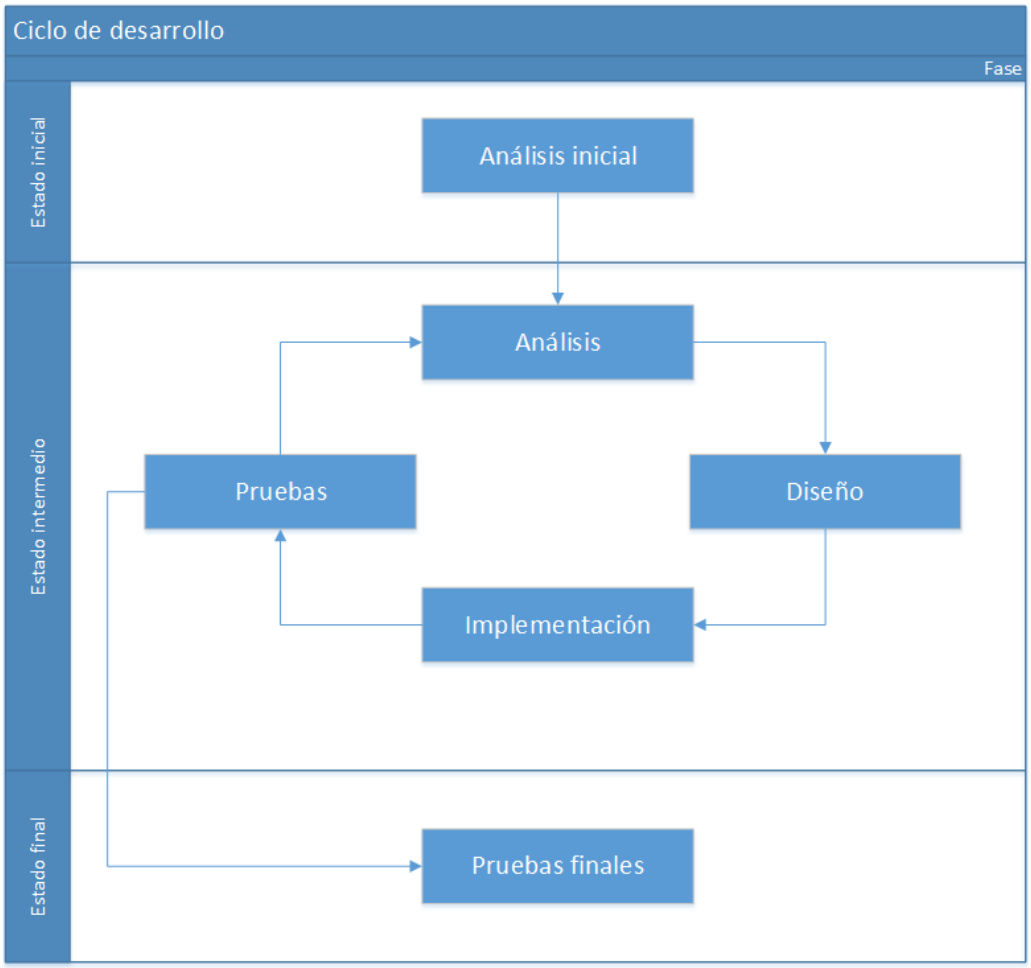

Figura 1.1: Diagrama de desarrollo en espiral

Por esta razón, a este proyecto se adecúa mejor una metodología ágil de desarrollo, teniendo en cuenta que se mantendrán las reuniones con el jefe de equipo, cuya figura está representada por los tutores del proyecto, y se omitirán las posibles reuniones con los miembros del equipo debido a su inexistencia.

Por lo tanto, se decide utilizar una metodología que consiste en reuniones frecuentes con el tutor y un desarrollo incremental, mezclado con fases de diseño e implementación. Junto a ello, se han tomando principios de metodologías clásicas como son la del desarrollo en espiral, cuyo esquema se muestra en la figura 1.1.

En primer lugar, se realizará un análisis del estado del arte, de las decisiones principales del proyecto y de los requisitos de usuario de la aplicación. Los principales temas que se abordan en el análisis del estado del arte son la teoría médica que se desarrolla en torno al proyecto, los protocolos de comunicación de la tecnología Bluetooth y las peculiaridades de la plataforma de desarrollo de Android.

Una vez que se dispone de este análisis, se divide el trabajo en tareas, componiéndose cada una de ellas de cuatro partes: análisis del problema, diseño, implementación y pruebas, incluyendo si es posible la creación de un prototipo, especialmente en fases más tardías del proyecto como se ha comentado. Se intentará que estos pasos sean incrementales incorporando la mayor parte de la integración dentro de las tareas correspondientes.

El desarrollo de la aplicación que abarca este proyecto se realizará de manera paralela al desarrollo del dispositivo externo, por lo que, finalmente, será necesaria una fase de integración y otra de pruebas de la aplicación con el dispositivo.

1.4. Organización del documento

En la parte II del documento se va a desarrollar el análisis inicial del problema. En el capítulo 2 se presenta el estado del arte en el que se describe la teoría física y médica de la espectroscopia, se desarrolla a fondo la tecnología Bluetooth necesaria para la implementación de la aplicación, y se explica en detalle las características del entorno de desarrollo de los terminales Android.

El capítulo 3 trata todos los temas relacionados con el análisis del sistema, la captura de requisitos, tanto de usuario como de software, y el desarrollo de los casos de uso que derivan de los requisitos extraídos. Finalmente, se relacionará toda esta información de proyecto a través de las matrices de trazabilidad, con las que se podrá observar las posibles dependencias de cada una de las características del desarrollo.

La parte III expone el diseño de todo el sistema y la implementación del prototipo. Se divide en capítulos en los que se detalla cada módulo del sistema por separado.

El capítulo 4 contiene los detalles de diseño del sistema en general, con todos los módulos en conexión, así como el funcionamiento y los protocolos de comunicación que se emplearán, incluyendo las necesidades descritas en el capítulo de análisis.

En el capítulo 5 se detallará el proceso de diseño e implementación de la aplicación, incluyendo los diagramas de clases y la explicación de los algoritmos y cálculos empleados en la misma. El capítulo 7 explicará como se han realizado las pruebas de la aplicación emulando el dispositivo externo mediante un servidor de escritorio ejecutado en un ordenador de sobremesa.

El capítulo 6 expondrá brevemente los diseños realizados por el equipo de desarrollo del dispositivo externo. Y, finalmente, en el capítulo 8 se detallarán las pruebas realizadas a diferentes sujetos con el sistema en conjunto.

Por último, la parte IV contiene las conclusiones personales y del proyecto en el capítulo 10, así como el presupuesto estimado y los gastos imputados al proyecto en el capítulo 9.

En la parte V se incluirán los apéndices, el glosario de términos y las referencias del documento.

Parte II

Análisis del problema

2

Estado del arte

En los últimos años, el crecimiento del número de usuarios de smartphones o teléfonos inteligentes, y tablets o tabletas ha sido enorme. Las cifras de estudios y encuestas revelan que, en España, dos de cada tres usuarios dispone de al menos un dispositivo de alguno de los tipos anteriores, y en la mayoría de estos casos, de ambos [7].

Como consecuencia, ha aumentado exponencialmente el número de aplicaciones para estos dispositivos que se encuentran disponibles en el mercado. Entre ellas se incluyen las aplicaciones con un propósito distinto al ocio o tiempo libre. En el sector que nos ocupa, la medicina, informes y análisis del mercado como el realizado por Research2guidance [8] predicen una cifra de 500 millones de personas usando aplicaciones médicas en dispositivos móviles para el año 2015, lo que supone alrededor de un 30 % de usuarios de estos dispositivos.

Sin embargo, no sólo los pacientes recurren al uso de estas aplicaciones, sino que cada vez más, los profesionales del sector de la medicina las utilizan como medio de apoyo para el seguimiento o ayuda para el tratamiento de ciertas enfermedades.

Como ejemplo, en el trabajo de Scully CG y col [9] se describen métodos para detectar el pulso, la respiración, la saturación de oxígeno en sangre arterial y la telemonitorización de estas variables con un dispositivo móvil.

A continuación, se expondrán los principales ámbitos y su estado del arte que tiene como objetivo cubrir el proyecto.

2.1. Espectroscopia

La espectroscopia es una técnica instrumental utilizada para determinar la composición de una muestra, mediante la utilización de patrones o espectros conocidos de otras muestras. El análisis espectral permite detectar la absorción o emisión de radiación electromagnética de distintas energías y relacionarlos con los distintos niveles implicados en una transición cuántica.

2.1.1. Origen

En 1666, Newton [10] observó y se planteó que cuando un rayo de luz natural (blanca) pasa a través de un prisma óptico se descompone en otros colores más simples; es decir, el prima dispersa o separa las luces monocromáticas que componen la luz blanca o cualquier otra luz policromática.

El fenómeno de la dispersión de la luz se debe a que las distintas radiaciones que componen una luz compleja (policromática) se propagan con distinta velocidad en los diversos medios transparentes y, en consecuencia, experimentan distintos ángulo de refracción. Como esta velocidad de propagación es directamente proporcional a la longitud de la onda de la radiación e inversamente proporcional al índice de refracción del medio, resulta que las radiaciones de menor longitud de onda son las más desviadas.

Hasta la segunda mitad del siglo XX, la estructura de una sustancia era determinada usando la información obtenida de las reacciones químicas. Esta información incluía la identificación de grupos funciones con ensayos químicos, junto con los resultados de experimentos de degradación en los cuales las sustancias se rompían en otras más pequeñas, es decir, en fragmentos más fácilmente identificables. Un ejemplo típico de este método es la demostración de la presencia de un doble enlace en un alqueno mediante la hidrogenación catalítica y su localización mediante ozonólosis, reacción de un alqueno con ozono.

Después de considerar todas las evidencias químicas disponibles, el químico proponía la estructura o estructuras candidatas que estaban de acuerdo con las observaciones. La prueba de la estructura se conseguía al convertir la sustancia en un compuesto ya conocido mediante su síntesis.

Los ensayos cualitativos y las degradaciones químicas como pruebas estructurales han sido sustituidas en la química orgánica actual por métodos instrumentales de determinación de la estructura. Las más destacadas de estas técnicas son la espectroscopia de resonancia magnética nuclear (rmn), espectroscopia de infrarrojos (ir), espectroscopia de ultravioleta-visible (uv-vis), y la espectroscopia de masa (em). A pesar de ser distintas, todas ellas están basadas en la absorción de energía por una molécula y todas examinan como la molécula responde a esa absorción de energía.

2.1.2. Ley de Beer-Lambert

La relación entre la absorción de luz y la concentración de la fase absorbente, o concentración de la solución, viene dada por la ley de Beer. Por otro lado, la relación entre la absorción de luz y el camino óptico recorrido por ésta viene dada por la ley de Lambert. Ambas leyes relacionan la absorción de la luz con las propiedades del material atravesado, es decir, la cantidad de luz absorbida por un material depende de:

- El tipo de sustancia que atraviesa la luz.
- La distancia que recorre la luz.
- La concentración de la sustancia.

La ley de Beer-Lambert relaciona la intensidad de luz entrante en un medio con la intensidad de luz saliente después de que en dicho medio se produzca absorción. La relación entre ambas intensidades puede expresarse de la siguiente manera:

$$\frac{I}{I_0} = 10^{-\varepsilon \ell c} = 10^{-A} \tag{2.1}$$

Donde,

- I, I_0 son las intensidades salientes y entrantes respectivamente.
- A es la absorbancia.
- ε es el coeficiente de absorción.
- ℓ es la longitud atravesada por la luz en el medio.
- c es la concentración del absorbente en el medio.

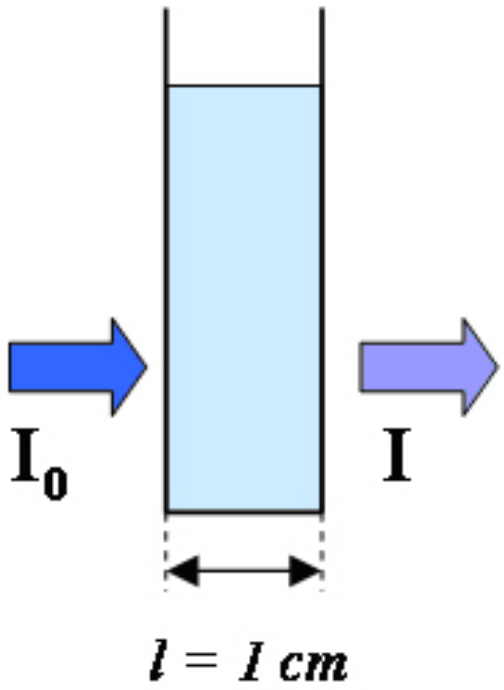

Figura 2.1: Ley de Beer y Lambert

La absorbancia A también puede expresarse como:

$$A = -\log_{(10)} \frac{I}{I_0} = \varepsilon \ell c \tag{2.2}$$

De esta forma, se define transmitancia como:

$$T = \frac{I}{I_0} = 10^{-A} \tag{2.3}$$

La transmitancia se relaciona con la absorbancia A como:

$$A = -log_{10}T = -log_{10}(\frac{I}{I_0}) \tag{2.4}$$

El coeficiente de absorción ε cambia según el material y la longitud de onda. Para el mismo material, en alguna longitud de onda, ε puede ser mayor y, de este modo, absorber más luz que en otras longitudes de onda dónde ε pueda ser más pequeño y el material no absorbe luz. Por lo tanto, ε se debe expresar en función de la longitud de onda:

$$A(\lambda) = \varepsilon(\lambda)\ell c \tag{2.5}$$

En general, una gráfica de $A(\lambda)$ en λ es el espectro de absorción (ver figura 2.2).

2.1.3. Propiedades de la ley de Beer-Lambert

La ley de Beer rige el proceso de absorción en cualquier región del espectro electromagnético, ya sea absorción de radiación visible-ultravioleta, absorción de rayos X, absorción de rayos gamma, etc.

La absorbancia es una propiedad aditiva, es decir, en disoluciones que contengan más de una especie de absorbente, la absorbancia total es la suma de las absorbancias individuales de cada especie absorbente.

Suponiendo que no haya interacción entre las distintas especies absorbentes, es decir, que estas sean independientes entre sí, la absorbancia total para un sistema absorbente multicomponente n, cuyas absorbancias individuales sean An, viene dado por:

$$A_t = \sum_{i=1}^{n} A_i = A_1 + A_2 + ... + A_n \tag{2.6}$$

Los oxímetros de pulso determinan la saturación de oxígeno de la sangre arterial, midiendo la absorción de luz sobre tejido vivo a dos longitudes de onda diferentes y usando la pulsación arterial para diferenciar entre la absorción de la sangre arterial de otras absorciones.

2.1.4. Espectroscopia en tejidos biológicos

La propagación de la luz en los tejidos biológicos puede ser descrita en términos de un proceso de transporte de fotones. En esta perspectiva, la fuente de luz inyecta un determinado número de fotones por unidad de tiempo, de volumen, y de ángulo en una localización específica del tejido. Estos fotones viajan dentro del tejido a lo largo de una determinada trayectoria que es determinada por las propiedades de absorción y dispersión del tejido [11].

Absorción

Los principales absorbentes de la luz cercana al infrarrojo en los tejidos sanguíneos son la hemoglobina oxigenada, la hemoglobina desoxigenada y el agua. Sus espectros de absorción varían entre los 300 y los 1300 nm como se puede observar en la figura 2.2, que son obtenidos de los datos de absorción del agua (Hale and Querry, 1973) [12] y la hemoglobina (Prahl) [13].

Las concentraciones de oxihemoglobina y desoxihemoglobina se asumen que son aproximadamente de 50 μM, que es un valor característico en los tejidos sanguíneos. La llamada "ventana espectral médica" se extiende aproximadamente desde los 700 a los 900 nm, donde la absorción de la luz muestra un mínimo (ver figura 2.2). Como resultado, la luz en esta ventana espectral penetra profundamente en los tejidos permitiendo así investigaciones no invasivas. La profundidad de la penetración óptica de la luz es limitada, en bajas longitudes de onda, por los niveles de absorción de la hemoglobina, y a altas longitudes de onda por los niveles de absorción del agua.

Las propiedades de absorción de los tejidos son definidas por el coeficiente de absorción (μ_a), que se define como la inversa de la distancia media recorrida por un protón antes de ser absorbido.

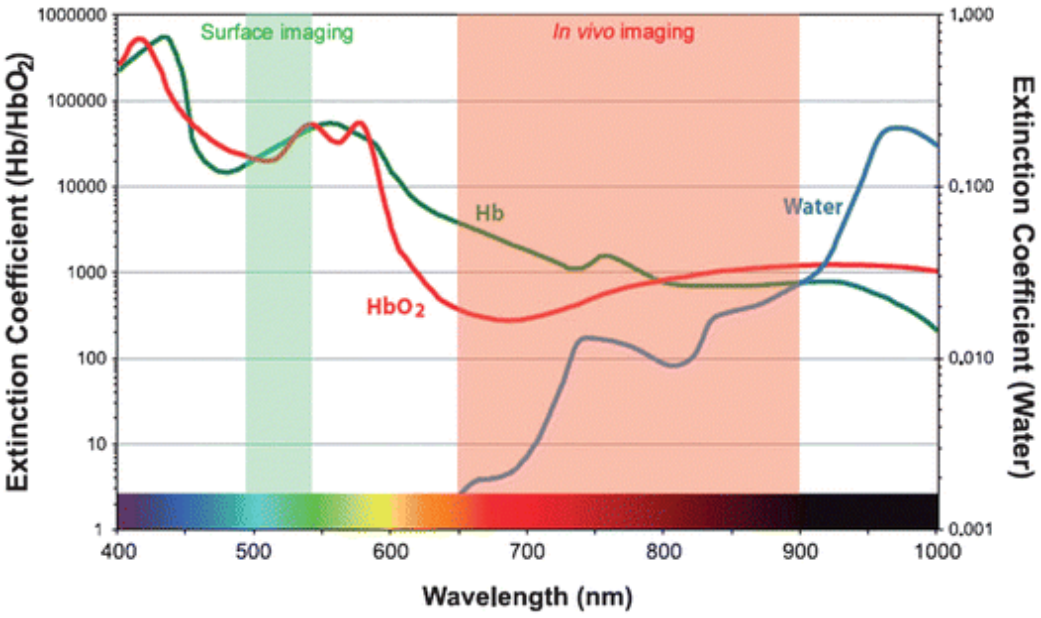

Figura 2.2: Espectros de absorción de la hemoglobina oxigenada (HbO_2), hemoglobina desoxigenada (Hb) y agua.

Dispersión

Las propiedades de dispersión de una muestra son determinadas principalmente por el tamaño de las partículas dispersadas relativas a la longitud de onda de la luz y por la diferencia del índice de refracción de las partículas dispersadas y el medio. En los tejidos biológicos, el centro de dispersión son células, orgánulos celulares, moléculas de alto peso molecular, y cualquier otra estructura que genere dispersión de fotones.

En medios de fuerte dispersión del medio, las propiedades de dispersión son normalmente definidas por el coeficiente de dispersión reducido (μ_s'), que representa la inversa de la distancia media sobre la que la dirección de propagación de un protón es aleatorizada.

Espectroscopía de absorción

Debido a que el coeficiente de absorción de los tejidos se compone de un número de cromóforos es necesario el análisis de varias longitudes de onda para determinar la contribución relativa de cada uno de estos elementos. La idea básica es que la contribución de μ_a del i-ésimo cromóforo puede ser escrito como el producto del coeficiente de extinción molar (ε_i) multiplicado por la concentración (C_i) de ese cromóforo. Como resultado, en presencia de N cromóforos, el coeficiente de absorción μ_a de una longitud de onda λ_i dada se puede obtener por medio de la ecuación 2.7.

$$\mu_a(\lambda_j) = \sum_{i=1}^{N} \varepsilon_i(\lambda_j) C_i \tag{2.7}$$

Si el espectro de extinción $\varepsilon_i(\lambda)$ de todas las especies N es conocido, la concentración C_i puede ser determinada mediante el cálculo de μ_a con N o más longitudes de onda. Este enfoque requiere que μ_a sea calculado de manera independiente de μ_s'.

Oximetría de tejidos

El espectro de absorción de tejidos se puede determinar considerando tan solo tres cromóforos: oxihemoglobina, desoxihemoglobina y agua.

Sin embargo, es posible calcular los niveles de saturación de oxigeno de la hemoglobina en sangre usando solamente dos de estas longitudes de onda. El método para realizar estas mediciones se conoce desde el año 1942, gracias a Millikan, y actualmente es usado por los oxímetros de pulso para calcular la saturación arterial (Mendelson, 1992) [11].

Las dos longitudes de onda usadas para este método λ_1 y λ_2 de oximetría infrarroja son elegidos de tal manera que $\lambda_1 < \lambda_{iso} \leq \lambda_2$, donde λ_{iso} es el punto isobéstico de las longitudes de onda en las cuales los coeficientes de extinción de la oxi y desoxihemoglobina tienen el mismo valor (λ_{iso} es aproximadamente 800 nm como se puede observar en la figura 2.2).

La elección de estas longitudes de onda maximizan la sensibilidad del cálculo óptico de los cambios de los tejidos oxigenados. El cálculo de λ_a en dos longitudes de onda convierten la ecuación 2.7 en un sistema linear de ecuaciones (una para cada longitud de onda) con dos incognitas (la concentración de la hemoglobina oxigenada $[HbO_2]$ y de la hemoglobina desoxigenada $[Hb]$). De su resolución se obtienen las concentraciones de cada una de las hemoglobinas descritas anteriormente, y con ello se puede calcular la saturación de hemoglobina en los tejidos mediante la ecuación 2.8.

$$StO_2 = \frac{[HbO_2]}{[HbO_2] + [Hb]} \tag{2.8}$$

2.1.5. Pulsioximetría

La pulsioximetría consiste en la medición no invasiva del oxígeno transportado por la hemoglobina en el interior de los vasos y tejidos sanguíneos. Este análisis es realizado mediante un dispositivo llamado pulsioxímetro o saturómetro [14].

El pulsioxímetro emite luz con dos longitudes de onda de aproximadamente 660 nm (roja) y 940 nm (infrarroja) que son características de los niveles de absorción de la oxihemoglobina y la desoxihemoglobina respectivamente. La mayor parte de la luz es absorbida por el tejido conectivo, la piel, el hueso y la sangre venosa en una cantidad constante, produciéndose un pequeño incremento de esta absorción en la sangre arterial con cada latido. Esto significa que es necesaria la presencia de pulso arterial para que el aparato reconozca alguna señal.

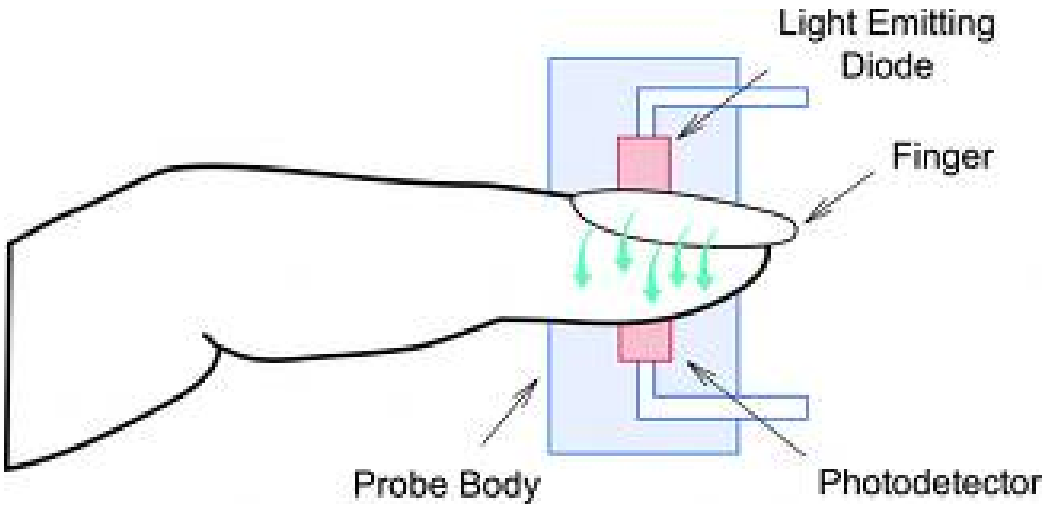

Figura 2.3: Diseño de un pulsioxímetro

Mediante la comparación de la luz que absorbe durante la onda pulsátil con respecto a la absorción basal, se calcula el porcentaje de concentración de oxihemoglobina. Solo se mide la absorción neta durante una onda de pulso, lo que minimiza la influencia de tejidos, venas y capilares en el resultado.

El pulsioxímetro mide la saturación de oxígeno en los tejidos, tiene un transductor con dos piezas, un emisor de luz y un fotodetector, generalmente en forma de pinza y que se suele colocar en el dedo. Después, se espera recibir la información en la pantalla: la saturación de oxígeno, frecuencia cardíaca y curva de pulso.

La pulsioximetría mide la saturación de oxígeno de la hemoglobina en sangre, pero no mide la presión de oxígeno (PaO_2), la presión de dióxido de carbono ($PaCO_2$) o el pH. Por tanto, no sustituye a la gasometría en la valoración completa de los enfermos respiratorios. Sin embargo, supera a la gasometría en rapidez y en la monitorización de estos enfermos.

Los aparatos disponibles en la actualidad son muy fiables para valores entre el 80 % y el 100 %, pero su fiabilidad disminuye por debajo de estas cifras. El punto crítico que debe dar la señal de alarma es el de saturaciones inferiores al 95 % (inferiores al 90 % ó 92 % cuando existe patología pulmonar crónica previa) estos pacientes deben recibir tratamiento inmediato.

Los aparatos actuales son muy fiables cuando el paciente presenta saturaciones superiores al 80 %. Las situaciones que pueden dar lugar a lecturas erróneas son:

- **Anemia severa.** La hemoglobina debe ser inferior a 5 mg/dl para causar lecturas falsas.
- **Interferencias** con otros aparatos eléctricos.
- **El movimiento.** los movimientos del transductor, que se suele colocar en un dedo de la mano, afecta a la fiabilidad (por ejemplo el temblor o vibración de las ambulancias), se soluciona colocándolo en el lóbulo de la oreja o en el dedo del pie o fijándolo con esparadrapo.
- **Contrastes intravenosos.** Pueden interferir si absorben luz de una longitud de onda similar a la de la hemoglobina.
- Si no se protege de la **luz ambiente** al sensor.
- **Luz ambiental intensa:** xenón, infrarrojos, fluorescentes, etc.
- Mala **perfusión periférica** por frío ambiental, disminución de temperatura corporal, hipotensión, vasoconstricción... Es la causa más frecuente de error ya que es imprescindible para que funcione el aparato que existe flujo pulsátil. Puede ser mejorada con calor, masajes, terapia local vasodilatadora, quitando la ropa ajustada, no colocar el manguito de la tensión en el mismo lado que el transductor.
- **La ictericia**, que es la presencia de bilirrubina en sangre, un pigmento biliar que resulta de la degradación de la hemoglobina, no interfiere.
- **El pulso venoso:** fallo cardíaco derecho o insuficiencia tricuspídea. El aumento del pulso venoso puede artefactar la lectura, se debe colocar el dispositivo por encima del corazón para minimizarlo.
- **Fístula arteriovenosa**. No hay diferencia salvo que la fístula produzca isquemia distal.
- La **hemoglobina fetal** no interfiere.
- **Obstáculos** a la absorción de la luz: laca de uñas (retirar con acetona), pigmentación de la piel (utilizar el 5º dedo o el lóbulo de la oreja).

- **Dishemoglobinemias.** La carboxihemoglobina (intoxicación por monóxido de carbono) y la metahemoglobina absorben longitudes de onda similares a la oxihemoglobina. Para estas situaciones son necesarios otros dispositivos como CO-oxímetros o aumentar las longitudes de onda medidas.

2.1.6. Oximetría de pulso por reflectancia

El oxímetro de pulso explicado en el apartado anterior utiliza el método de medición por tramitancia, también conocido como método convencional. Éste se basa en el análisis espectrofotométrico de las propiedades de absorción de la sangre, combinado con el principio de la fotopletismografía. Sin embargo, la StO_2 limita su aplicación a las áreas del cuerpo suficientemente delgadas para que penetre una luz roja e infrarroja.

En 1949, Brinkman y Willem Gerrit Zijlstra [15] describen la medición de la saturación de oxígeno con el uso de la reflexión de la luz mediante la utilización de sensores frontales. En 1983, Yitzhak Mendelson [16] demuestra que la saturación de O_2 de la hemoglobina se puede medir cuantitativamente con la técnica de reflectancia. En 1992, se inicia el desarrollo de sensores ópticos para medir los niveles de oxigenación en las regiones de los tejidos ($rStO_2$).

La $rStO_2$ se sustenta en la reflexión de la sangre en dichos tejidos, y es un método no invasivo que proporciona un monitoreo continuo, exacto, fiable y económico de la saturación de O_2. Los diodos emisores de luz y el fotoreceptor permiten la absorción en la superficie de la piel.

Comparativa

La $rStO_2$ ofrece varias ventajas sobre los oxímetros convencionales. El oxímetro de pulso por tramitancia utiliza la tecnología de la transmisión de luz para calcular la saturación de O_2, sin embargo, el oxímetro de pulso por reflectancia emplea el método de la reflexión de la luz.

En el oxímetro convencional, un lado del sensor envía luz a través del tejido y el otro lado la recibe. El sensor del oxímetro de pulso por reflectancia emite luz a través del tejido y recoge la luz que se refleja calculando la saturación de O_2.

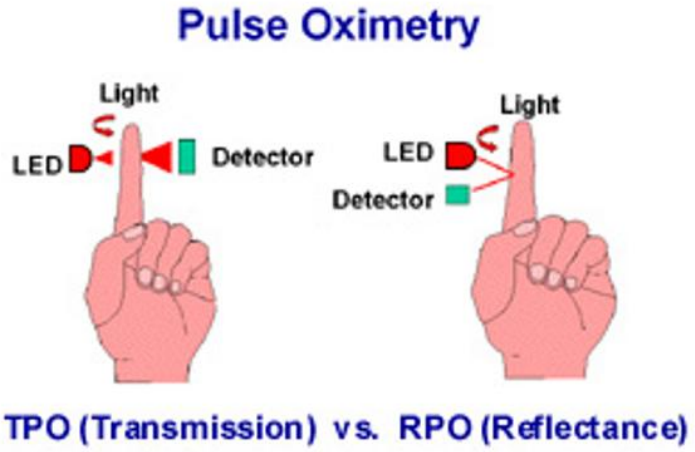

Figura 2.4: Diferencia entre pulsioxímetros de tramitancia y reflectancia

Sensores reflectantes

El sensor convencional envuelve el tejido que analiza, mientras que el sensor reflectante es plano y se adhiere a la piel con un adhesivo. El funcionamiento del sensor incluye la emisión de luz roja e infrarroja en múltiples longitudes de onda, y la detección de campos definida por dos anillos fotodetectores que se colocan concéntricamente alrededor del origen de la luz. Los anillos constituyen una forma anular que permite la obtención de la señal desde una amplia zona tisular debido a la cantidad de luz que se absorbe por la sangre.

Indicaciones

En algunas ocasiones, la oximetría de pulso por tramitancia o convencional dificulta la señal exacta de la oxigenación, debido a una hipoperfusión en brazos y piernas, particularmente en pacientes críticos, y también al continuo movimiento, especialmente en pacientes neonatos. El sistema de $rStO_2$ está diseñado e indicado para resolver estos problemas asociados con los sensores periféricos, ya que, debido a su localización corporal central, proporcionan mayor exactitud en las lecturas de la saturación de O_2. En cirugías de corazón abierto, donde el gasto cardíaco, la temperatura periférica, la perfusión tisular y la presión sistólica es menor, el sensor de reflectancia tiene más probabilidad de obtener lecturas correctas, que un sensor convencional.

Aplicaciones

En la actualidad, la tecnología de reflectancia adquiere gran interés médico, no solo para la monitorización de rutina de pacientes quirúrgicos y de cuidados críticos, sino que también se evalúa su uso en neonatología, medicina materno-fetal y neurología.

2.2. Bluetooth

Bluetooth es un estandar industrial de Redes Inalámbricas de Área Personal (WPAN) que permite la transmisión de datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 2,4 GHz. Los objetivos que se pretende conseguir esta tecnología son:

- Lograr redes *ad hod* simples de bajo coste y consumo.
- Realizar fácilmente comunicaciones entre dispositivos móviles.
- Eliminar las conexiones hardware entre éstos.
- Permitir la posibilidad de crear redes inalámbricas reducidas y facilitar la sincronización de datos entre dispositivos.

La tecnología usada por Bluetooth fue desarrollada en 1940 por un grupo de militares. En 1994, un grupo de ingenieros de una compañía sueca llamada *Ericsson Bluetooth tecnology*. Más tarde, en el año 1998 surgió lo que se conoce actualmente como la tecnología Bluetooth, creado gracias a un grupo formado por cinco compañías [6]. Ya en el año 2000 apareció el primero teléfono móvil con esta tecnología, así como las primeras tarjetas de PC y los primero prototipos de ratón y teclados inalámbricos. Con el paso del tiempo el número de aplicaciones y dispositivos que han hecho uso de esta herramienta ha ido en crecimiento.

Los desarrolladores aprovechan todas las funcionalidades que ofrecen los *smartphones* para crear nuevas aplicaciones que les permitan innovar en los aspectos más típicos de la vida cotidiana como por ejemplo la reproducción de música inalámbrica o el control remoto de dispositivos en el hogar.

2.2.1. Alcance y versiones

Los dispositivos que incorporan el protocolo Bluetooth pueden conectarse y comunicarse entre ellos, siempre y cuando, se encuentre dentro de su rango de alcance. Existen tres tipos de dispositivos clasificados en clases (ver tabla 2.1) según su potencia de transmisión, siendo totalmente compatibles entre sí.

Clase	**Potencia máxima permitida**	**Alcance**
Clase 1	100 mW — 20 dBm	30 metros
Clase 2	2.5 mW — 4 dBm	10-5 metros
Clase 3	1 mW — 0 dBm	1 metro

Tabla 2.1: Clases de Bluetooth

Cuando un dispositivo de clase 2 se conecta a un clase 1, la cobertura efectiva del primero se extiende debido a la mayor sensibilidad y potencia de transmisión del segundo dispositivo. Es decir, el incremento en la potencia del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el dispositivo con menor potencia de señal. La recepción de la señal del dispositivo con mayor potencia aumenta debido a la mayor sensibilidad de este dispositivo en contraposición con la potencia de emisión del dispositivo con menor potencia.

Los dispositivos Bluetooth pueden clasificarse, además, según su ancho de banda siguiendo la tabla 2.2.

Versión	**Ancho de banda**
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Tabla 2.2: Versiones de Bluetooth

2.2.2. Perfiles

Cuando se establece una conexiones entre dos dispositivos Bluetooth, reciben información sobre los protocolos que ofrece el dispositivo asociado, sin embargo, sólo pueden intercambiar datos los dispositibos que comparten el mismo protocolo.

Mientras que la tecnología Bluetooth define la conexion inalámbrica física entre dos dispositivos, los perfiles establecen los comandos y las funciones que estos dispositivos pueden intercambiara través del protocolo. Es decir, los perfiles Bluetooth son descripciones de comportamientos generales que los dispositivos pueden utilizar para comunicarse, que siguen un estándar para favorecer su uso unificado. Las distintas funcionalidades que posee un dispositivo Bluetooth se basa en los perfiles que soporta ese dispositivo, de esta forma, permite que la fabricación de los mismo se adapte a sus necesitades. Una especificación de perfil debe cubrir:

- Posibles dependencias con otros perfiles.
- Formatos recomendados de la interfaz con el usuario.
- Partes concretas de la pila Bluetooth que se utilizan, como opciones particulares y parámetros.

Se puede encontrar una lista de perfiles definidos y adoptados por Bluetooth SIG en [17].

2.2.3. Arquitectura

La tecnología Bluetooth se divide en dos especificaciones: el núcleo y la especificación de perfiles. La especificación del núcleo trata de como funciona la tecnología en las capas inferiores (ver figura 2.5), mientras que la de perfiles se centra en discutir la interoperabilidad entre dispositivos usando la tecnología descrita en el núcleo [18].

Capa de frecuencias de radio

La interfaz de Bluetooth se basa en una potencia de antena de 0dBm (1 mW) con extensiones para operar hasta 20 dBm (100 mW) a lo largo del mundo (ver tabla 2.1). La radio usa frecuencias de salto (*frecuency hopping*) para difundir energía a través del espectro ISM en 79 saltos desplazados por 1 MHz, comenzando en 2.402 GHz hasta los 2.480 GHz. Actualmente, el grupo de desarrollo SIG está trabajando para estandarizar estos 79 canales de radio de manera global y ha propuesto cambios en Japón, España y otros paises.

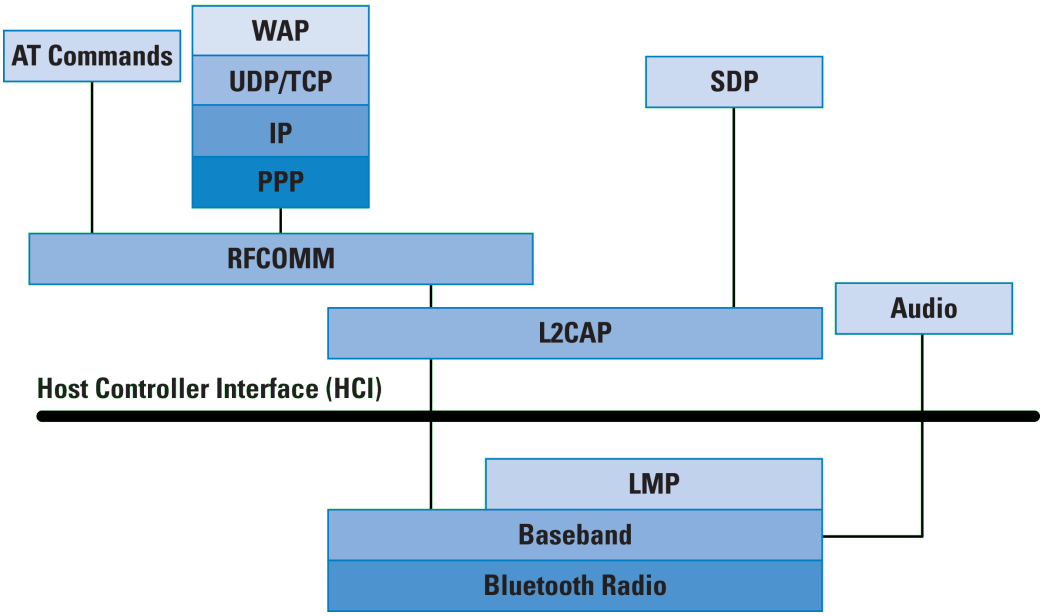

Figura 2.5: Arquitectura Bluetooth

El rango oficial del protocolo es de 10 centímetros hasta los 10 metros, pero puede ser extendido hasta los 100 metros aumentando su poder de transmisión (usando la opción de 20 dBm).

Banda de Bluetooth

Como se comentó anteriormente, la radio básica es un espectro híbrido de difusión de radio. Normalmente, la radio opera en una frecuencia de salto en la que la banda ISM de 2.4 GHz que se divide en 79 canales de 1 MHz, sobre los que va alternando mientras recibe y envía datos.

Se conoce como *piconet* cuando una radio bluetooth se conecta con otra entre sí. Ambas radios pueden saltar juntas los 79 posibles canales. El sistema de radio Bluetooth soporta un gran número de *piconets* proporcionando a cada una de ellas su propio conjunto de patrones de salto. Normalmente, las *piconets* acaban terminando en el mismo canal. Cuando esto ocurre, saltarán a un canal libre y retransmitirán los datos si hay perdidas.

Una trama de Bluetooth consiste en la transmisión de un paquete seguido por la recepción de otro. Cada uno de estos paquetes esta compuesto por distintas ranuras de tiempo de 625 μs cada una. En la figura 2.6 se puede observar una ranura de tiempo de 1600 saltos (*hops*) por segundo.

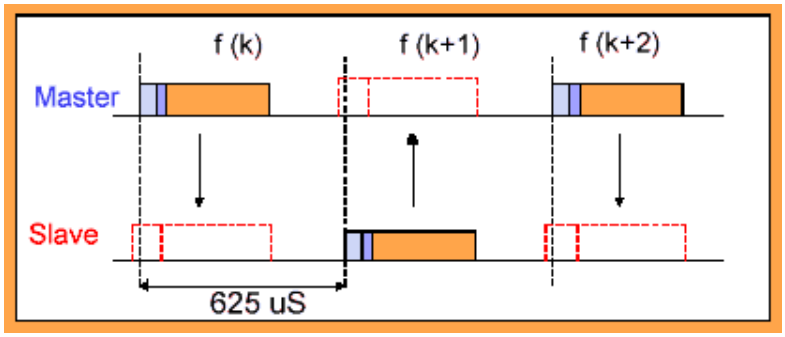

Figura 2.6: Ranuras de tiempo Bluetooth

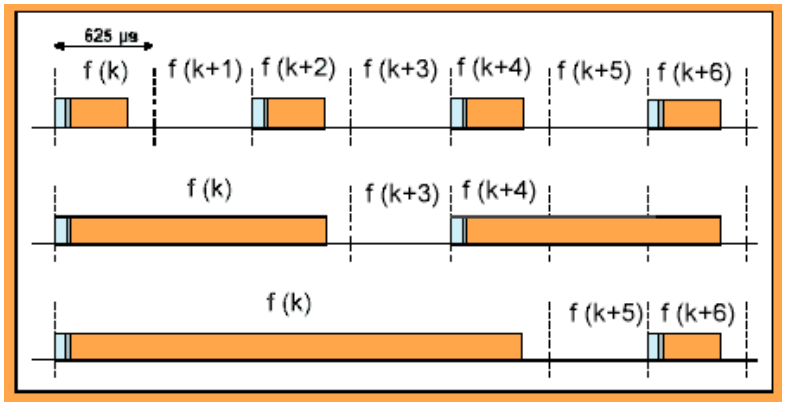

Figura 2.7: Ranuras múltiples de tiempo Bluetooth

Las ranuras múltiples de tiempo permiten las transferencia de mayor cantidad de datos debido a la eliminación del tiempo de regreso entre paquetes y a la reducción del tamaño de cabecera. A modo de comparativa, un paquete de ranura única (*single-slot*) puede alcanzar una tasa de transferencia de 172 kbps, mientras que un paquete de 5 ranuras (*multi-slot*) podría alcanzar los 721 kbps con un ratio de regreso de canal de 57.6 kbps (ver figura 2.7).

Topología de la red

En la figura 2.8 se puede observar una red de conexiones de dispositivos Bluetooth. Cada uno de los rectángulos corresponde con un dispositivo y cada circulo con una *piconet* que está formada siempre por un dispositivo maestro conectado simultáneamente a varios esclavos. Los dispositivos Bluetooth se consideran simétricos debido a que cualquiera puede ser un maestro o esclavo, es decir, la configuración de una *piconet* es determinada solamente por el momento de creación. Como norma general, la radio que conecta será el maestro, sin embargo, puede dar un intercambio de maestro-esclavo

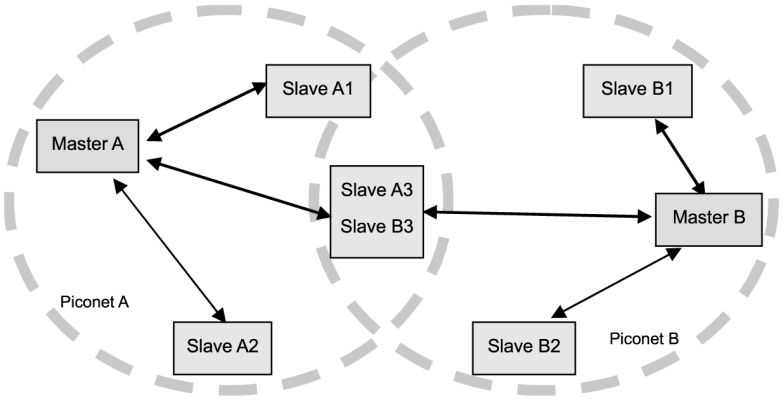

Figura 2.8: Topología de la red Bluetooth

(*master/slave swap*) que permite un cambio de roles entre dispositivos. Un dispositivo solamente puede ser maestro en una red simultáneamente.

Para la formación de una *piconet*, el dispositivo Bluetooth necesita conocer dos parámetros: el patrón de salto de la radio a la que desea conectarse y la fase de ese patrón. Cada dispositivo posee un identificador global único (*Global ID*) que es usado para crear los patrones de salto. Para formar la red, el dispositivo maestro comparte su identificación con los otros dispositivos, que se convierten en dispositivos esclavos, y les proporciona el patrón de salto. El maestro también comparte su desplazamiento de reloj (*clock offset*), representado por un dial de reloj, e incluyéndolo en el patrón de salto. Esta información es transmitida mediante paquetes FHS (*Frequency Hopping Synchronization*).

Los dispositivos normalmente no se conectan a las redes que están en modo *Standby*. En este modo, los dispositivos están a la escucha de otros dispositivos para ser encontrados, estado *inquiry*, y/o escuchan las peticiones para formar redes, estado *page*. Cuando un dispositivo ejecuta un comando *inquire*, los dispositivos a la escucha responderán con un paquete FHS. Con este paquete, los dispositivos cargan el identificador y el desplazamiento de reloj del dispositivo de origen, además de unirse a su red.

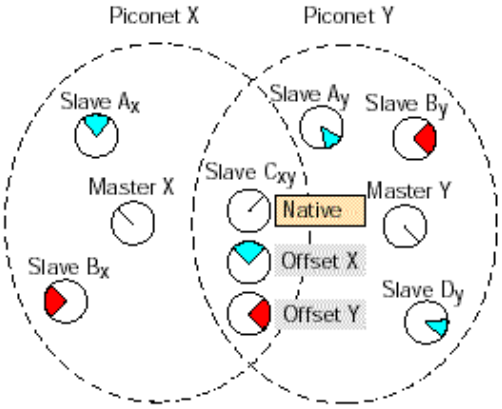

Figura 2.9: Desplazamientos de reloj de una red Bluetooth

Una vez que un dispositivo se une a una red, se le asigna una dirección de miembro de 3 bits (*Active Member Address*, AMA) que permite a los demás dispositivos conectar con él. Cuando una red tiene ocho dispositivos activos, el maestro elegir un dispositivo en espera (*parked*) y coordinarlo de manera que le permite liberar su AMA por un paquete de 8 bits llamado *Passive Member Address* (PMA). La liberación de su AMA puede ser asignada a otro dispositivo que quiera unirse a la red. La combinación de los AMA y PMA permiten más de 256 dispositivos residir en una red simultáneamente, mientras que solo los ocho dispositivos con AMA's pueden transferir datos.

Los dispositivos en espera están a la escucha de una marca de intervalo de información para direccionarlos. De esta manera, el dispositivo maestro puede retransmitir hacia todos los esclavos, tanto activos como en espera.

Los dispositivos que no están conectados activamente a la red están en modo *standby*, y quedan a la escucha de dispositivos en estado *inquiry* o *page*. Cada 1,25 segundos éstos realizan un escaneo en busca de dispositivos en estos modos para encontrar peticiones que hayan sido realizadas.

El proceso de búsqueda (*inquiry*) implica que un dispositivo ejecute una función *page* sobre la identificación de búsqueda (*Inquiry ID*), qué es una dirección global asignada para este tipo de funciones; mientras que los demás dispositivos ejecutan un escaneado de búsqueda. Este proceso se realiza en una secuencia única de 32 canales. El dispositivo que hace el escaneado de búsqueda tiene que permanecer a la escucha cada uno de estos 32 canales cada 1,25 segundos durante 10 ms, después de esto, repitiendo este proceso en el siguiente canal.

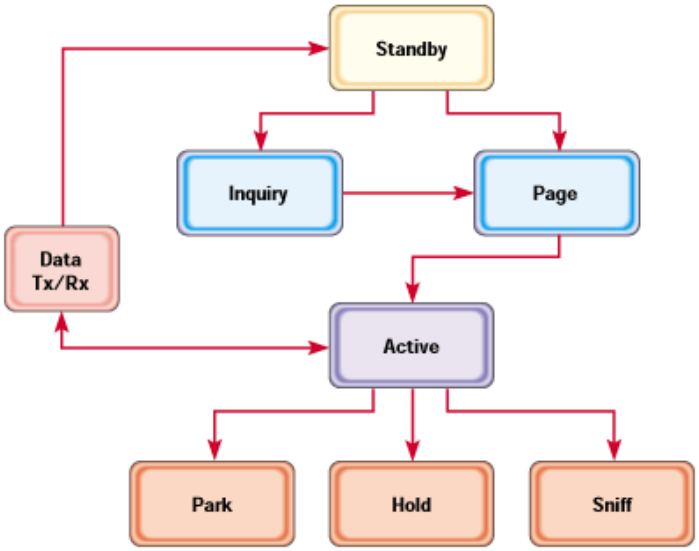

Figura 2.10: Diagrama de modos y estados Bluetooth

Un dispositivo con el escaneado de búsqueda activo continua este proceso hasta que esta función es deshabilitada. El radio de búsqueda incluye un número de *pages* en los canales de búsqueda (doble por cada ranura) y entonces escucha dicha respuesta en su correspondiente frecuencia durante 1,25 segundos por cada 16 de las 32 frecuencias. La secuencia es repetida en las siguientes 16 frecuencias después de que el dispositivo de búsqueda tiene una lista con todos los paquetes FHS que están en su rango.

La función de paginación (*paging*) sigue una secuencia similar a la anterior. Cada radio tiene una secuencia única de 32 frecuencias de paginación y 32 frecuencias de respuesta basadas en su identificación global. Un dispositivo en modo *standby* realizando un escaneo de paginación esta a la escucha de una página de su identificación en cada una de las frecuencias de paginación durante 10 ms por página cada 1,25 segundos. El dispositivo realizando la paginación estima dónde deberían escuchar los dispositivos paginados y continuará paginando 16 de estas frecuencias durante 1,25 segundos. En el caso de que esto haya ido mal, podría tardar hasta un máximo de 2,5 segundos en conectarse.

Una vez que un dispositivo ha sido encontrado, por medio de una función de búsqueda, y es asignado a una red, mediante la función de paginación, se

formará una red.

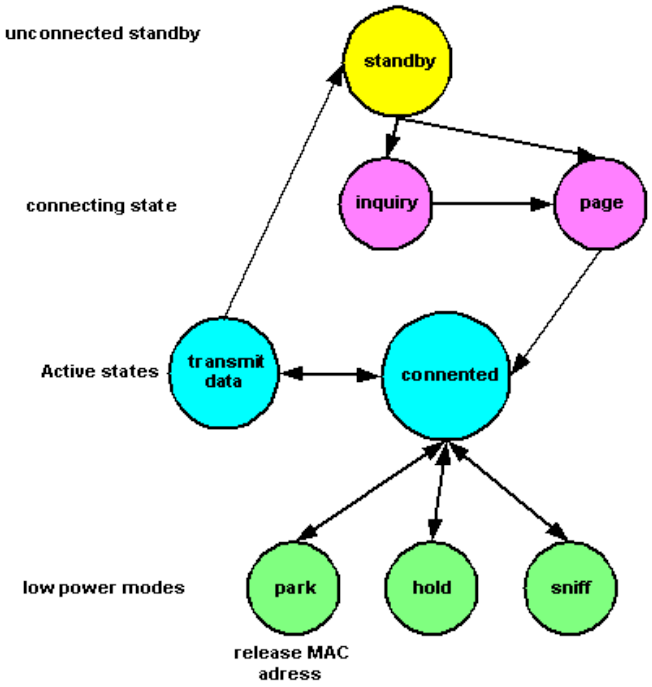

Figura 2.11: Diagrama de estados funcionales Bluetooth

Cuando el dispositivo se encuentra en el estado conectado (*inquiry* o *page*), se le asigna una dirección AMA mediante la que puede dirigir datos entre los diferentes dispositivos de una red. El dispositivo maestro siempre será identificado con la dirección 0. Para permitir que un dispositivo conectado con la red, manteniendo los patrones de saltos y los desplazamientos, se mantenga en un estado de baja potencia (ver figura 2.11), se necesita que el dispositivo se encuentre en estado *park*, *hold* o *sniff*.

En los estado *hold* y *sniff*, los dispositivos tienen que mantenerse despiertos durante un intervalo dado, sin embargo, en el estado *sniff*, el dispositivo puede transferir datos en ese intervalo, mientras que en el estado *hold* no se transmiten datos. En el estado *park*, el dispositivo queda a la espera y se le da la dirección PMA. Una radio en estado *park* a la espera escucha una posible señal de intervalo para ver si el maestro ha pedido que el dispositivo se vuelva activo, si algún dispositivo en espera quiere ser activo, o si existe algún dato para retransmitir.

Mientras que un dispositivo se encuentra en estado conectado, puede tratar dos tipos de paquetes: un paquete orientado a conexiones síncronas (*Synchronous Connection Oriented*, SCO) o un paquete sin conexión asíncrono (*Asynchronous Connectionless Type*, ACL). El tipo SCO esta asociado con datos síncronos para voz, que suele ser un paquete simétrico de 1, 2, o 3 ranuras y donde las tramas están reservadas sean usadas o no en la red. Para poder establecer una conexión SCO, el dispositivo necesita haber establecido una conexión ACL.

El enlace ACL está orientado a paquetes y soporta tanto simetría como asimetría en el tráficos de datos. Los paquetes ACL están creados con un impar de ranuras tales que la trama es siempre un número par de ranura, como por ejemplo $\frac{1}{1}$, $\frac{1}{3}$ o $\frac{1}{5}$.

2.2.4. Seguridad

La manera en que los dispositivos Bluetooth son usados en los dispositivos móviles y el tipo de datos que retransmiten hacen que la seguridad en este protocolo sea un factor de extrema importancia. Mientras que la mayoría de los sistemas sin cables aportan que retransmitir en un espectro de radio mejora la seguridad, el volumen proyectado por la tecnología Bluetooth elimina esta barrera. De esta manera, la capa de enlace y de aplicación son partes fundamentales de las especificaciones de Bluetooth.

A nivel de la capa de enlace, la tecnología Bluetooth proporciona autenticación, encriptación y gestión de claves. El sistema de autenticación implica proporcionar al usuario un número de identificación personal (*Personal Identification Number*, PIN) que es almacenado en una clave de 128 bits usado para autentificar en una o dos vías de dirección.

Una vez que los dispositivos han sido autenticados, el enlace puede ser encriptado con distintas longitudes de clave, hasta 128 bits en incrementos de 8 bits. La arquitectura de seguridad a nivel de la capa de enlace proporciona un patrón de autenticación y un patrón de encriptación flexible que permite a los dispositivos negociar por una longitud de clave. Esto es un factor importante, ya que, de este modo, distintos dispositivos de diferentes países pueden interactuar entre ellos.

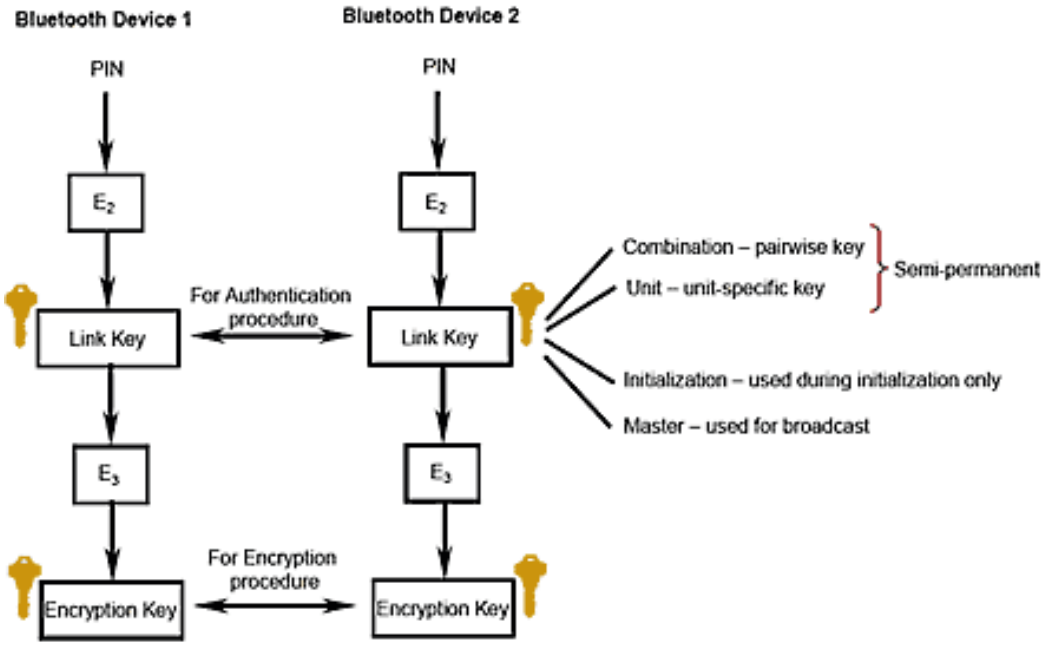

Figura 2.12: Diagrama de seguridad en la capa de enlace Bluetooth

La arquitectura de seguridad de Bluetooth depende de los códigos PIN para establecer conexiones de confianza entre dispositivos. Mientras que no es práctico usar todas las posibles combinaciones de códigos PIN, debe tenerse en cuenta que una vez que se ha establecido el enlace de confianza entre dispositivos, estos códigos pueden almacenarse para permitir conexiones más rápidas y simples. La clave para la simplicidad de este sistema consiste en establecer relaciones de confianza entre dispositivos usados frecuentemente.

2.3. Android

Android Inc. fue fundado en 2003 por Andy Rubin (cofundador de Danger Inc.), Rich Miner (cofundador de Wildfire Communication Inc.), Nick Sears (vicepresidente de T-Mobile) y Chris White (a cargo del diseño y desarrollo de interfaces en WebTV) con el objetivo de desarrollar software para teléfonos móviles que fueran capaces de adecuarse a la localización y las preferencias del usuario [19]. En sus inicios, la empresa era relativamente desconocida.

En julio de 2005, Google compra Android Inc. interesándose especialmente en el equipo de desarrolladores que la componían. Con el apoyo de los recursos de Google, el tirmo de desarrollo de Android se vió incrementado

notablemente y el equipo dirigido por Rubin desarrolló una plataforma para teléfonos móviles que funcionaba sobre el kernel de Linux. Google promocionó la plataforma entre fabricantes de teléfonos y operadores móviles con la promesa de proveer un sistema flexible y actualizable.

En noviembre de 2007, la Open Handset Alliance [20], una alianza de empresas de tecnología como Google, fabricantes de dispositivos como HTC y Samsung, operadores como Sprint Nextel y T-Mobile y fabricantes de chips como Qualcomm y Texas Instruments, se dio a conocer con la finalidad de desarrollar estándares abiertos para dispositivos móviles.

Una semana después, Google presentó una primera versión de Android, una plataforma para teléfonos móviles basada en la versión 2.6 del kernel de Linux y lanzó la primera versión del Android Software Development Kit (SDK). No obstante, no es hasta diciembre de 2008 cuando se puso a la venta el primer terminal móvil con este sistema operativo, el HTC Dream.

Figura 2.13: Open Handset Alliance members

En agosto de 2008, Google anuncia el Android Market donde los desarrolladores podrían subir sus aplicaciones para que todos los usuarios pudieran descargarlas. En un principio no estaba soportado el sistema de pago por las aplicaciones, que sería posteriormente introducido en 2009.

2.3.1. Arquitectura

Para comprender el funcionamiento del sistema operativo Android es necesario entender como está estructurado y que componentes los forman. La arquitectura de Android está formada por varias capas que facilitan el desarrollo de aplicaciones para la plataforma tal y como se muestran en la figura 2.14. Cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones, es por ello que a este tipo de arquitectura se le conoce también como arquitectura de pila. Este modelo hace posible que los desarrolladores puedan acceder a las capas mas bajas del sistema operativo, que permiten usar los componentes hardware de los teléfonos, a través de librerías.

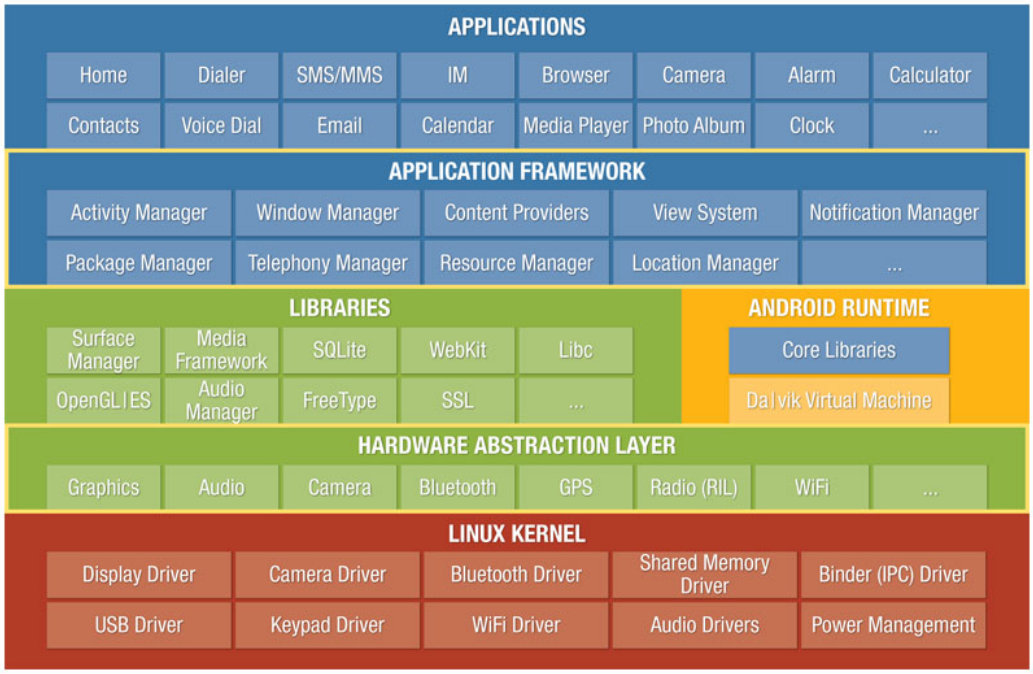

Figura 2.14: Arquitectura de Android

Kernel de Linux El núcleo del sistema operativo Android está basado en la versión 2.6 del kernel de Linux que administra la seguridad, la memoria, los procesos, la pila de red y los drivers, proporcionando las necesidades básicas de software para arrancar y gestionar tanto el hardware como las aplicaciones. El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura.

El kernel de Android está disponible de manera libre en android.git.kernel.org. Sin embargo, el kernel solo suele ser modificado por fabricantes de hardware y dispositivos que quieren asegurarse de que el sistema operativo funcione correctamente en sus dispositivos.

Capa de abstracción de hardware La capa de abstracción de hardware (HAL) es un elemento del sistema operativo que funciona como una interfaz entre el software y el hardware del sistema, proveyendo una plataforma de hardware consistente sobre la cual corren las aplicaciones. Cuando se emplea este capa, las aplicaciones no acceden directamente al hardware, sino que lo hacen desde la capa abstracta provista por la HAL.

Librerías Esta capa se sitúa sobre el kernel y esta formada por librerías nativas de Android escritas en C o C++ que han sido compiladas para la arquitectura hardware específica del teléfono. Estas librerías son desarrolladas normalmente por los fabricantes de dispositivos. Algunas de las librerías más importantes son OpenGL (motor gráfico), Media Framework (códecs de formatos de audio, imagen y vídeo), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto) y SQLite (base de datos).

Entorno de ejecución de Android El entorno de ejecución de Android no es considerado una capa por sí mismo ya que está formado por librerías. Está formado por dos componentes: la máquina virtual Dalvik y las librerías del núcleo de Java. La máquina virtual Dalvik es un tipo de máquina virtual Java optimizado para sistemas que están limitados en términos de procesamiento y memoria.

Las aplicaciones están programadas normalmente en Java y son compiladas en bytecode. El bytecode resultante se transforma en un formato compatible con la máquina virtual Dalvik (.dex) para su posterior instalación en el dispositivo. La máquina virtual Dalvik permite la creación de múltiples instancias simultáneamente proporcionando seguridad, aislamiento, gestión de memoria y soporte para hilos.

Al ser una variante de la máquina virtual de Java permite la instalación de aplicaciones en los dispositivos independientemente del hardware siempre que dispongan de la versión mínima de Android requerida. Las librerías del núcleo de Java son diferentes de las librerías Java SE y Java ME pero proporcionan prácticamente la misma funcionalidad que la primera.

Framework de aplicaciones La siguiente capa está formada por todas las clases y servicios con los que interaccionan las aplicaciones. Estos componentes gestionan las funciones básicas del teléfono y acceden a recursos de las capas anteriores a través de la máquina virtual Dalvik.

Algunos de los bloques más importantes de esta capa son: *Activity Manager*, que se encarga de gestionar las actividades de las aplicaciones así como sus ciclos de vida; *Content Provider*, que encapsula los datos que se comparten entre aplicaciones y permite mantener un control de acceso a dichos datos; *Views*, elementos que permiten construir las interfaces que se muestran al usuario (botones, listas, cuadros de texto...); *Telephony Manager*, que permite realizar/recibir llamadas y enviar/recibir mensajes de texto; *Location Manager*, que hace posible la obtención de la localización del dispositivo mediante GPS o redes disponibles...

Aplicaciones Esta es la última capa de la arquitectura de Android. En esta capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz gráfica como las que no, las nativas (programadas en C/C++) y las administradas (programadas en Java), las preinstaladas y aquellas que ha instalado el usuario.

2.3.2. Máquina Virtual Dalvik

El sistema operativo Android utiliza una máquina virtual llamada Dalvik que se encuentra en la capa de entorno de ejecución y que ha sido especialmente diseñada para optimizar el uso de la memoria y los recursos de hardware en dispositivos móviles. Dalvik también está optimizada para permitir la ejecución de múltiples instancias de la máquina virtual simultáneamente con un impacto muy bajo en el rendimiento de la memoria del dispositivo. Este aspecto de usar varias máquinas virtuales se pensó para proteger a las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma a las demás.

La máquina virtual Dalvik fue diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google. Recibe su nombre en honor a Dalvík, un pueblo de Eyjafjörour, Islandia, donde vivieron antepasados de Bornstein.

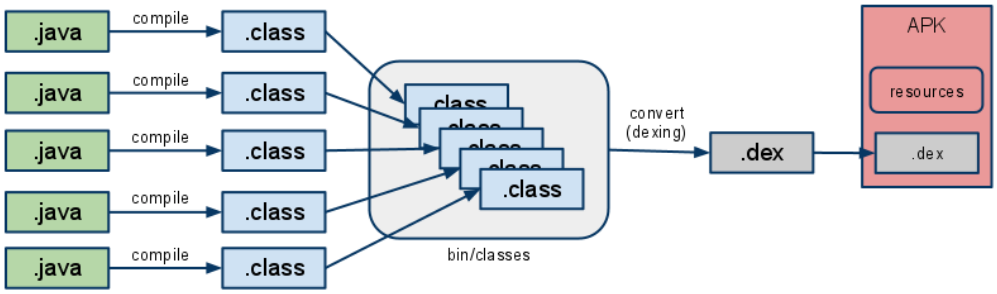

Figura 2.15: Proceso de compilación de Android

Dalvik es una máquina virtual que es gestionada y ejecutada como cualquier otra aplicación, pero su función es la de proporcionar un entorno de programación independiente de la plataforma que permita obviar el hardware instalado por debajo y posibilite la ejecución de los programas realizados para ella en cualquier tipo de plataforma sin tener que realizar modificación alguna en el código de la aplicación. Utiliza el modelo de compilación Just-In-Time (JIT), también conocido como traducción dinámica, que es un híbrido entre los lenguajes interpretados y los compilados y que consiste en traducir el bytecode a código máquina nativo en tiempo de ejecución, lo que mejora el rendimiento considerablemente.

El intérprete toma los archivos generados por las clases Java (.class) y los combina en uno o más archivos ejecutables Dalvik (.dex), los cuales a su vez son comprimidos en un sólo fichero .apk (Android Package) en el dispositivo. De esta forma, reutiliza la información duplicada por múltiples archivos .class, reduciendo así la necesidad de espacio (sin comprimir) a la mitad de lo que ocuparía un archivo .jar.

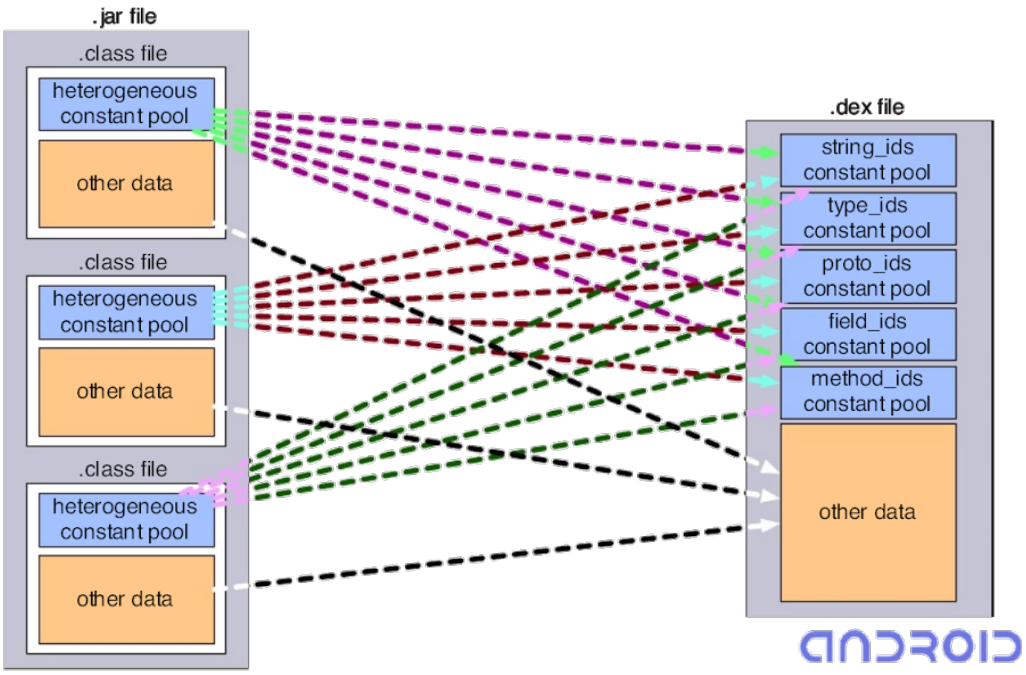

Figura 2.16: Formato .jar vs .dex

A diferencia de la máquina virtual Java (JVM), que es una máquina virtual de pila, Dalvik es una máquina virtual de registro. Este tipo de máquinas virtuales tienen como modelo la máquina de Turing con uno o más registros que sustituyen a la cinta y el cabezal utilizado en dicha máquina teórica.

Las máquinas virtuales de pila utilizan una o más pilas como forma de utilizar la memoria de la máquina y su ventaja respecto a las máquinas virtuales de registro reside en que generalmente tienen una mayor densidad de código lo que facilita la lectura. Sin embargo, las máquinas virtuales de registro, al basarse en estos últimos, suelen ser más rápidas que las de pila dado que estas últimas se basan en memoria, aunque existen máquinas de pila que incluyen caché en registros para acelerar la ejecución en la medida de lo posible.

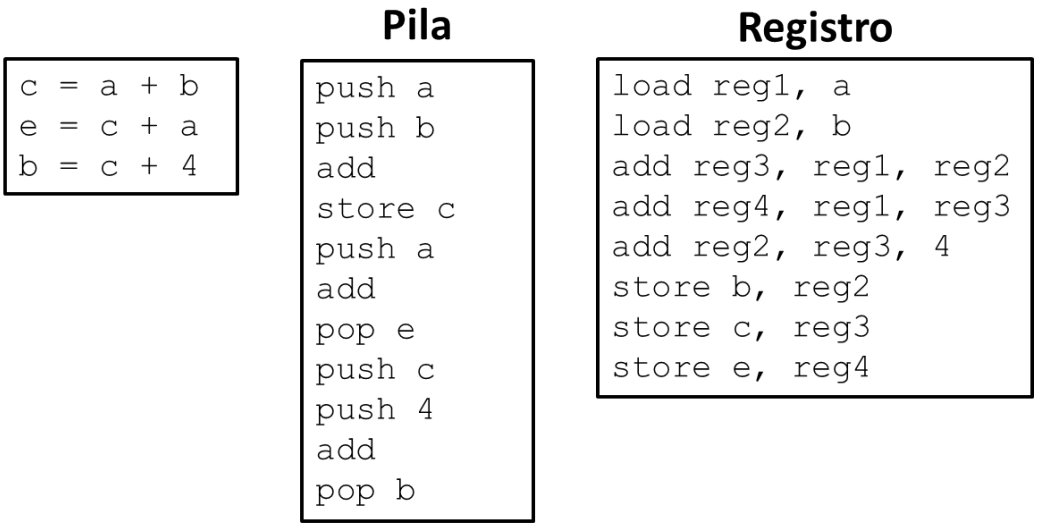

Figura 2.17: Pila vs Registro

2.3.3. Características

Debido a que Android es un sistema operativo de código abierto (bajo licencia Apache), está disponible libremente para que los fabricantes de dispositivos puedan adaptarlo a sus necesidades. Android no tiene una configuración predefinida ni de hardware ni de software. Por si mismo, Android soporta las siguientes características:

- **Diseño.** La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
- **Almacenamiento.** Android usa SQLite, una base de datos relacional ligera, para almacenar datos.
- **Conectividad.** Soporta GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (A2DP and AVRCP), Wi-Fi, LTE, HSDPA, HSPA+ y WiMAX.
- **Mensajería.** Soporta SMS y MMS.
- **Navegador web.** El navegador web incluido está basado en el motor de renderizado de código abierto WebKit, junto con el motor JavaScript V8 de Google Chrome.

- **Soporte multimedia.** Soporta WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

- **Soporte para hardware.** Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.

- **Multitáctil.** Android tiene soporte nativo para pantallas capacitivas con soporte multi-táctil.

- **Multitarea.** Android soporta multitarea real de aplicaciones, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada.

- ***Tethering.*** Permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico.

2.3.4. Versiones

Google ha lanzado numerosas actualizaciones del sistema operativo Android desde su lanzamiento. La tabla 2.3 [21] muestra la relación de todas las versiones de Android lanzadas hasta la fecha. Cada versión se identifica por un número y por un nombre de postre en inglés. En cada versión el postre elegido empieza por una letra distinta siguiendo un orden alfabético.

Versión de Android	Fecha de lanzamiento	Nombre	API
1.0	23 de septiembre de 2008	Apple Pie	1
1.1	9 de febrero de 2009	Banana Bread	2
1.5	30 de abril de 2009	Cupcake	3
1.6	15 de septiembre de 2009	Donut	4
2.0	26 de octubre de 2009	Eclair	5
2.0.1	3 de diciembre de 2009	Eclair	6
2.1	12 de enero de 2010	Eclair	7
2.2	20 de mayo de 2010	Froyo	8
2.3/2.3.2	6 de diciembre de 2010	Gingerbread	9
2.3.3/2.3.7	9 de febrero de 2011	Gingerbread	10
3.0	22 de febrero de 2011	Honeycomb	11
3.1	10 de mayo de 2011	Honeycomb	12
3.2	15 de julio de 2011	Honeycomb	13
4.0/4.0.2	19 de octubre de 2011	Ice Cream Sandwich	14
4.0.3/4.0.4	16 de diciembre de 2011	Ice Cream Sandwich	15
4.1	9 de julio de 2012	Jelly Bean	16
4.2	13 de noviembre de 2012	Jelly Bean	17
4.3	24 de julio de 2013	Jelly Bean	18
4.4/4.4.2	3 de septiembre de 2013	KitKat	19

Tabla 2.3: Versiones de Android

En septiembre de 2008 salió a la venta el primer dispositivo Android con la versión 1.0, la HTC Dream, que incluía una serie de servicios de Google como Gmail (correo), Calendar (calendario), Maps (mapas), Search (búsqueda), GTalk (mensajería instantánea) y otras características y aplicaciones como YouTube, Android Market, WiFi, Bluetooth, notificaciones, etc.

En la versión 1.5 se incluyeron nuevas características como la capacidad para grabar vídeos en formato MPEG-4 y 3GP, el soporte para widgets o la posibilidad de subir videos a Youtube y fotos a Picassa, una aplicación de álbumes de imágenes.

Un año más tarde aparecía Android 2.0 que incorporaba una experiencia mejorada en el Android Market, una mejora en la búsqueda por voz, actualización de soporte para CDMA/EVDO, 802.1x, VPN y *text-to-speech*. Las siguientes revisiones de esta versión optimizaban el rendimiento del sistema operativo y la gestión de memoria y mejoraban otros aspectos tales como ofrecer soporte para Microsoft Exchange, funcionalidad de WiFi hotspot, soporte para Adobe Flash 10.1, soporte nativo para más sensores como giroscopios y barómetros, soporte nativo para múltiples cámaras, soporte nativo para telefonía VoIP, SIP, etc.

En febrero de 2011, Google lanzó la versión 3.0 de Android que estaba dirigida únicamente a las tabletas e incluía una serie de mejoras. Las aplicaciones desarrolladas para versiones anteriores de Android eran compatibles con los dispositivos que tuvieran instalada está versión del sistema operativo, pero no ocurría lo mismo en el caso contrario, es decir, las aplicaciones desarrolladas usando la API de la versión 3.0 de Android no funcionaban en dispositivos con versiones previas del mismo.

Para solucionar esto, en octubre de 2011, Google lanzó la versión 4.0 de Android que traía todas las mejoras incorporadas en la versión 3.0 a los smartphones y además incluía nuevas características como desbloqueo mediante reconocimiento facial, monitorización y control del uso de datos, Near Field Communication (NFC), etc.

La figura 2.18 [22] presenta un gráfico que recoge la cuota de mercado de cada una de las versiones de Android a día de hoy:

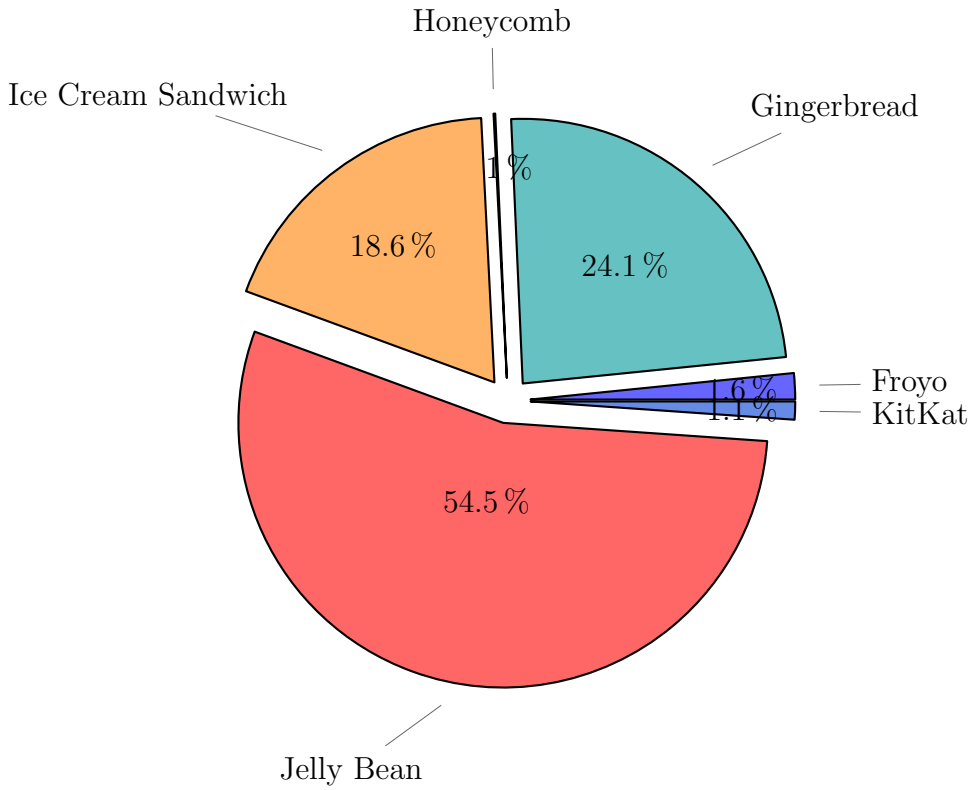

Figura 2.18: Distribución de versiones Android

2.3.5. Estructura de una aplicación

Las aplicaciones Android se escriben usando distintos lenguajes de programación. Las herramientas del Android SDK compilando el código (con los datos y los recursos necesarios) en un paquete Android, un archivo con la extensión .apk. Todo lo incluido en este archivo conforma una aplicación Android y es lo que todos los dispositivos usarán para instalar la aplicación.

Los componentes de las aplicaciones son los bloques esenciales que permiten construir una aplicación Android. Cada componente refleja una forma diferente mediante la cual la aplicación interactúa con el sistema. Hay cuatro tipos diferentes de componentes, cada uno de ellos tiene un propósito distinto y un ciclo de vida diferente que define cuando se crea y destruye el componente.

Activities

Una *activity* o actividad representa una ventana que contiene la interfaz del usuario. Por ejemplo, una aplicación de correo tendría una actividad que mostraría una lista con los correos nuevos, otra que permitiría enviar correos y otra que facilitaría la lectura de los correos. A pesar de que las actividades trabajan juntas para formar una experiencia cohesiva en la aplicación desde el punto de vista del usuaro, cada una es independiente de las otras.

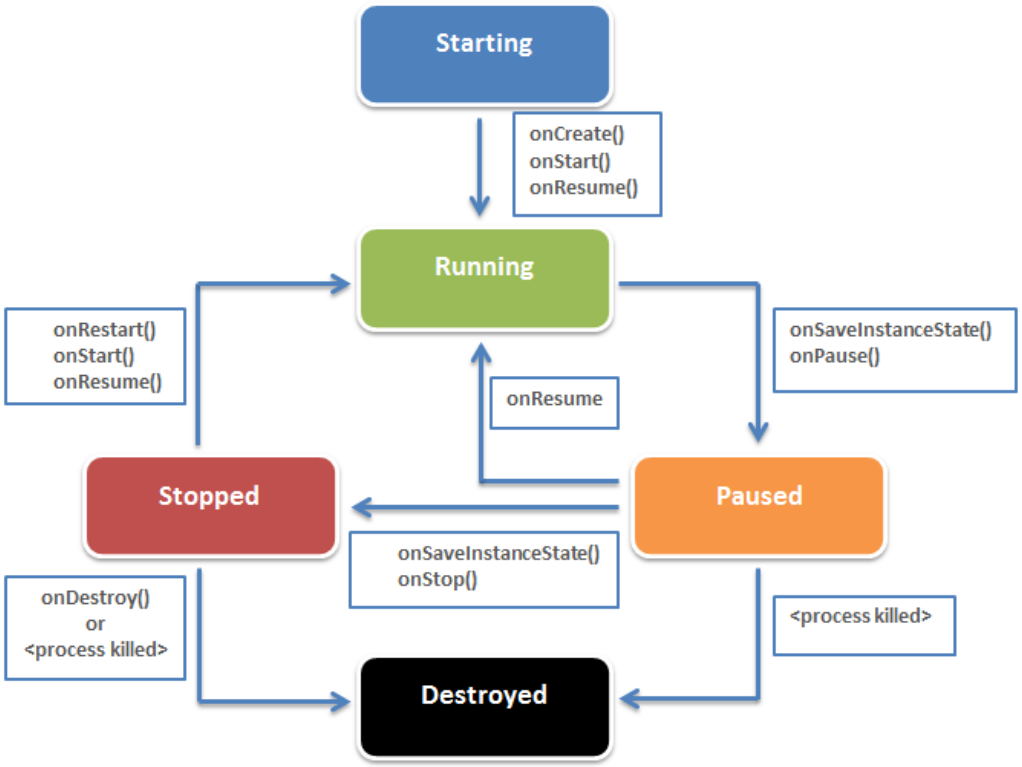

Figura 2.19: Ciclo de vida de una actividad Android

Una actividad puede encontrarse en distintos estados:

- **Activa o en ejecución.** Es la primera en la pila de ejecución, el usuario ve la actividad y puede interactuar con ella.
- **Pausada.** Ocurre cuando una actividad ha pasado a un segundo plano pero todavía es parcialmente visible. Típicamente ocurre cuando se abre un diálogo encima de la pantalla. En este caso, la actividad queda

oculta y puede ser cerrada por el sistema si necesitara liberar recursos de cualquier tipo para la nueva actividad.

- **Parada o detenida.** La actividad ha pasado a segundo plano y está completamente oculta por una nueva actividad, en ese caso el sistema también puede optar por cerrarla si necesitara liberar recursos.
- **Destruída.** La actividad no está disponible por lo que se han liberado todos sus recursos y en caso de ser llamada sería necesario comenzar un nuevo ciclo de vida.

En la figura 2.19 se pueden observar las distintas etapas explicadas anteriormente por las que pasa típicamente una actividad Android.

Services

Un *service* o servicio de Android es un componente que se ejecuta en *background*, es decir, de forma transparente al usuario, para llevar a cabo operaciones de larga duración o para efectuar tareas para procesos remotos. Un servicio no proporciona una interfaz de usuario. Por ejemplo, un servicio es reproducir música en *background* mientras el usuario utiliza una aplicación distinta, o reunir datos a través de la red sin bloquear la interacción del usuario con la actividad principal.

Content provider

Un *content provider* administra un conjunto de datos que pueden ser compartidos entre las aplicaciones. Una aplicación puede almacenar datos en el sistema, en una base de datos, en la web, o en cualquier otra localización persistente a la que pueda acceder. A través de un *content provider* otras aplicaciones pueden acceder o modificar estos datos, siempre que éste lo permita. Por ejemplo, Android proporciona un *content provider* que administra la información de los contactos del usuario. De esta manera, una aplicación con los permisos adecuados puede consultar o modificar la información almacenada sobre los contactos.

Broadcast receivers

Un *broadcast receiver* es un componente que responde anuncios globales del sistema. Muchos de estos anuncios son generados por el propio sistema, por ejemplo, distintos emisores anunciando que la pantalla se ha apagado, el nivel de la batería es bajo o se ha tomado una foto. Nol obstante, las

aplicaciones pueden generar sus propios *broadcasts* para avisar al resto de la aplicaciones de que ha ocurrido algún evento. Los *broadcast receivers* no muestran una interfaz de usuario pero pueden crear una notificación en la barra de estado para indicar cuando ha ocurrido un evento.

Tres de los cuatros componentes previamente descritos (*activities*, *services* y *broadcasts receivers*) son activados por un tipo de mensaje asíncrono llamado *intent*. Los *intents* permiten el paso de información en tiempo de ejecución entre estos componentes, ya sean de la misma aplicación o de aplicaciones distintas, a través de la cola de eventos de Android.

Manifest

El *manifiest* o archivo de manifiesto (AndroidManifest.xml) es un archivo de la aplicación que permite al sistema Android conocer previamente la existencia del componente que se quiere ejecutar. Una aplicación debe tener declarados todos sus componentes en este fichero que está situado en la carpeta raíz paquete ejecutable de la aplicación.

El *manifest* es un archivo en formato XML en el que se definen las características generales de una aplicación Android:

- **Paquete.** Son las etiquetas que identifican de forma unívoca una aplicación. No es posible añadir una aplicación al *Play Store* de Android si ya existe otra aplicación con el mismo nombre de paquete. Del mismo modo, si se instala en un dispositivo una aplicación con el mismo nombre de paquete que otra ya instalada, la nueva sustituirá la anterior o generará un conflicto.
- **Nombre.** Define el nombre de la aplicación que permite que los usuarios la identifiquen.
- **Versión.** Declara la versión mínima (ver tabla 2.3) de Android que necesita un dispositivo para poder instalar la aplicación.
- **Permisos.** Engloba la lista de permisos necesarios para que la aplicación se ejecute correctamente. Esta lista se le presentará al usuario cuando instale la aplicación.
- **Componentes.** Es una lista de los componentes que forman la aplicación.

2.3.6. Seguridad

El sistema operativo Android está basado en la versión 2.6 del kernel de Linux, que es el que proporciona los servicios de seguridad. A lo largo de su historia, el kernel de Linux ha sido investigado, atacado y reparado por miles de desarrolladores convirtiéndose en un kernel estable y seguro en el que confían grandes empresas y profesionales de la seguridad llegando a ser usado en millones de entornos sensibles a la seguridad. En la parte de computación de sistemas móviles, el kernel de Linux proporciona a Android varias característica claves en la seguridad del sistema operativo:

- Modelo de separación de privilegios basado en usuarios.
- Aislamiento de procesos.
- Mecanismo extensible para una comunicación entre procesos segura.
- Habilidad para eliminar partes innecesarias y potencialmente inseguras del kernel.

Como sistema operativo multiusuario, uno de los principales objetivos del kernel de Linux es aislar los recursos de los éstos. De manera que:

- El usuario A no pueda leer los archivos del usuario B.
- El usuario A no consuma toda la memoria del sistema.
- El usuario A no consuma todos los recursos de la CPU del sistema.
- El usuario A no consuma todos los dispositivos (telefonía, GPS, Bluetooth...) del sistema.

Todas las aplicaciones que se ejecutan en Android están sujetas a restricciones de seguridad impuestas por el framework. A continuación se muestran algunos de los aspectos más importantes de la seguridad en Android.

Separación de privilegios

El kernel de Android implementa un modelo de separación de privilegios como medio de identificación de aislamiento de los recursos de las aplicaciones. De esta manera, todas las aplicaciones del sistema operativo se ejecutan en su propia instancia de la máquina virtual Dalvik con su propio identificador de usuario (*uid*) e identificador de grupo (*gid*). Estos evita que aplicaciones o procesos sin permisos puedan acceder a otras aplicaciones o procesos proporcionando un *Application Sandbox* a nivel de kernel. Por defecto, las aplicaciones no pueden interactuar entre sí y tienen acceso limitado al sistema operativo.

Debido a que la puesta en marcha de las máquinas virtuales individuales puede incrementar la latencia entre el inicio de una aplicación y su ejecución, Android utiliza un mecanismo de pre-carga que permite acelerar el procedimiento. Para eso, se utiliza un proceso que recibe el nombre de *Zygote* y que tiene dos funciones: primero, actúa como una plataforma de lanzamiento para nuevas aplicaciones y, en segundo lugar, como un repositorio de bibliotecas al que pueden referirse las aplicaciones durante su ciclo de vida. El proceso *Zygote* se encarga de poner en marcha cada instancia de máquina virtual, la pre-carga y pre-inicializa con las bibliotecas básicas requeridad. A continuación, se mantiene la espera de recibir una señal para iniciar la aplicación. *Zygote* se inicia al arrancar el sistema y funciona de manera similar a una cola. Cualquier dispositivo Android tendrá siempre un proceso principal *Zygote* funcionando.

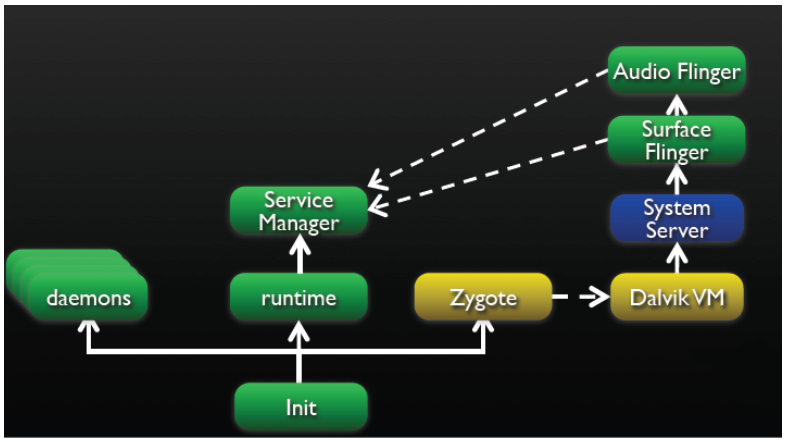

Figura 2.20: Secuencia de arranque de Android

En algunos sistemas operativos, los errores de corrupción de memoria comprometen generalmente la seguridad del dispositivo. Esto no ocurre en Android. Todas las aplicaciones y sus recursos están aislados a nivel de sistema y un error de este tipo únicamente permitiría la ejecución de código arbitrario en el contexto de la aplicación atacada.

Como todas las características de seguridad, el aislamiento de aplicaciones no es irrompible. Sim embargo, para romper el *Application Sandbox* en un dispositivo configurado correctamente, es necesario comprometer la seguridad del kernel de Linux.

Permisos

Por defecto, una aplicación Android solo puede acceder a una serie de recursos limitados del sistema, y es el propio sistema el que gestiona el acceso a los recursos de las aplicaciones de modo que, si se utilizan de manera incorrecta o malintencionada, pueden afectar desfavorablemente a la experiencia del usuario, a la red o a los datos en el dispositivo.

Estas restricciones se aplican de formas diferentes. Algunas funciones están restringidas por una falta intencional de la API de funcionalidad, por ejemplo, no existe ninguna API de Android que manipule de manera directa la tarjeta SIM. En algunos casos, la separación de privilegios proporciona una medida de seguridad, como ocurre con el aislamiento de almacenamiento de cada aplicación. En otros casos, las API's serán usadas por aplicaciones de confianza y protegidas a través de un mecanismo de seguridad conocido como control de permisos. Estas API's protegidas incluyen:

- Funciones de la cámara.
- Funciones GPS.
- Funciones Bluetooth.
- Funciones de telefonía.
- Funciones SMS/MMS de mensajería.
- Funciones de red y/o conexión de datos.

Estos recursos son solamente accesibles a través del sistema operativo. Para poder hacer uso de estas API's en el dispositivo, una aplicación debe defines las funcionalidades que necesita en su archivo *manifest* (AndroidManifest.xml). Cuando se instala una aplicación, el sistema muestra un cuadro de diálogo al usuario que indica los permisos que necesita y le pregunta si desea continuar con la instalación. El sistema asume que el usuario acepta todos los permisos solicitados. No se pueden conceder o denegar permisos de manera individual.

Dentro de los ajustes del dispositivo, los usuarios pueden ver los permisos que tienen las aplicaciones que han instalado previamente. Los usuarios también pueden desactivar algunas funciones a nivel global cuando se elige, por ejemplo, deshabilitar el GPS, la radio o el Wi-Fi. En el caso de que una aplicación intente utilizar una funcionalidad protegida que no se ha declarado en su archivo de permisos, resultará en una excepción de seguridad por parte de la aplicación solicitante.

Firma de las aplicaciones

La firma del código de las aplicaciones permite a los desarrolladores identificar al autor de la aplicación y actualizar las mismas sin necesidad de crear un sistema complejo de interfaces y permisos. Todas las aplicaciones que se ejecuten en la plataforma Android tienen que haber sido firmadas por el desarrollador. Cualquier aplicación sin firmar será rechazada, ya sea por Google Play, a la hora de ser subida al sistema, o por el instalador de paquetes, a la hora de ser instalada en el dispositivo Android. El uso de la firma de aplicaciones en Android permite:

- Identificar el autor del código.
- Detectar si la aplicación ha sido modificada.
- Establecer confianza entre las aplicaciones.

En Google Play, la firma de aplicaciones establece un vínculo de confianza entre Google y el desarrollador, y entre el desarrollador y su aplicación. Las aplicaciones son distribuidas a través de Google Play sin ser modificadas, por lo que los propios desarrolladores son responsables del comportamiento de las mismas.

En Android, la firma de aplicaciones es el primer paso para el aislamiento de procesos ya que el certificado firmado de la aplicación define el identificador asociado a la aplicación. La firma de aplicaciones evita que una aplicación no pueda acceder a otra si no tiene permisos.

Cuando una aplicación se instala en el dispositivo Android, el administrador de paquetes verifica que la APK (archivo de la aplicación) ha sido firmado correctamente con el certificado que incluye. Las aplicaciones pueden ser autofirmadas o firmadas por un tercero. Android permite la firma de aplicaciones utilizando certificados autofirmados que los desarrolladores pueden generar sin asistencia o permisos externos [23].

2.3.7. Bluetooth en Android

La plataforma Android incluye soporte para la red Bluetooth, que permite al dispositivo intercambiar datos inalámbricamente con otros dispositivos que posean Bluetooth. El entorno de la aplicación permite acceso a la funcionalidad de Bluetooth a través de las API del Bluetooth de Android. Estas API permiten realizar las siguientes funciones:

- Escanear y buscar otros dispositivos Bluetooth.
- Consultar al adaptador Bluetooth acerca de información de los dispositivos enlazados.
- Establecer canales RFCOMM.
- Conectar a otros dispositivos a través del servicio de descubrimiento (*discovery*).
- Transferir datos hacia/desde otros dispositivos.
- Gestionar conexiones múltiples.

Android 4.0 introduce soporte para perfiles médicos Bluetooth (*Bluetooth Health Device Profile*, HDP). Con esto se permite la creación de aplicaciones que usen la tecnología Bluetooth para comunicarse con dispositivos médicos que soporten Bluetooth, como monitores de medición del ritmo cardíaco, medidores de sangre, termómetros, etc. Para una lista de todos los dispositivos soportados y sus correspondientes protocolos de datos, ver [24].

2.4. Aplicaciones médicas en Android

Existe un gran número de aplicaciones para móviles que tienen como objetivo la adquisición de ciertas medidas fisiológicas como, por ejemplo, la frecuencia cardíaca o los niveles de saturación de oxígeno en sangre, sobre plataformas Android, iOS y otros sistemas similares. A continuación se detallarán una serie de aplicaciones similares que operan bajo el sistema operativo Android.

Aplicaciones como *Instant Heart Rate* [25] o *Cardiograph* [26] son algunos de los ejemplos existentes que se centran en calcular la frecuencia cardíaca mediante el hardware incorporado en el mismo terminal (ver figura 2.22 y figura 2.21).

Existen también otras aplicaciones mas avanzadas y complejas, como es el caso de *Blood Sugar & Pressure, Oxygen* [27], que son capaces de medir los niveles de azúcar en la sangre, la presión arterial, la saturación de oxígeno, temperatura, peso, etc (ver figura 2.23).

Figura 2.21: Aplicación Android Cardiograph

Figura 2.22: Aplicación Android Instant Heart Rate

Procedimiento

El procedimiento mediante el cual funcionan las aplicaciones mencionadas anteriormente es el siguiente: el usuario sitúa el dedo en la lente de la cámara del dispositivo con la que se toman distintas instantáneas y se analizan observando las variaciones en el color de las imágenes capturadas.

El proceso teórico seguido tiene similitud con el utilizado en la pulsioximetría (ver subsección 2.1.5), sin embargo, en este caso, las variaciones de color son analizadas por medio de la cámara del dispositivo, en vez de con emisión de luz.

Figura 2.23: Aplicación Android Blood Sugar & Pressure, Oxygen

En el mercado se pueden encontrar otros dispositivos basados en smartphones Android, como LifeWatch [28], que integran hardware y software para realizar ciertas medidas utilizando la cámara y otros sensores integrados de los que disponen (ver figura 2.24). El principal problema de estos dispositivos de análisis con hardware dedicado es que son caros, complejos y necesitan de entrenamiento especializado para poder ser manejados correctamente.

Figura 2.24: Dispositivo LifeWatch

3

Análisis

3.1. Introducción

A continuación, se desarrollará el análisis del sistema propuesto en función del estándar ESA PSS-05-0 (*PSS-05 lite*) propuesto por la Agencia Espacial Europea (ESA) de estándares de desarrollo de software que indican las pautas del desarrollo de proyectos de software pequeños [29].

Los apartados en los que se dividirá el capítulo contendrán una descripción general del sistema propuesto, que expondrá una idea aproximada del sistema final que se desarrollará; la especificación de requisitos de usuario y los casos de uso que indicarán las funcionalidades que debe cumplir el sistema a desarrollar, y por último, los requisitos de software extraídos de los requisitos adquiridos anteriormente que establecerán las pautas finales del sistema.

3.2. Descripción general

El objetivo del proyecto consiste en el diseño y la implementación de módulo de pulsioximetría divido en dos entornos: hardware y software.

El entorno de hardware será desarrollado por la Universidad de La Laguna (ULL), y su análisis, diseño e implementación será presentado en este documento como información adicional necesaria para entender el funcionamiento completo del sistema. De este modo, se incluirá dentro de las secciones de análisis, diseño e implementación correspondientes.

El entorno de software será desarrollado por el autor de este documento, y consistirá en la aplicación de la capa de lógica del sistema hardware presentado anteriormente.

La comunicación entre los dos entornos se realizará mediante la tecnología Bluetooth siguiendo un protocolo establecido por los desarrolladores de la aplicación hardware.

3.3. Requisitos de usuario

En el análisis de un proyecto de software se realiza la definición de requisitos de usuario, donde se especifican de forma puntual las condiciones o capacidades que necesita el usuario para resolver o conseguir un objetivo determinado, así como las capacidades y restricciones que deben establecerse para llevar a cabo la resolución de estos objetivos.

El desarrollo de los requisitos se hará en dos secciones:

- **Requisitos de capacidad.** Son los requisitos que definen una funcionalidad o característica requerida del sistema que expresan una capacidad de acción del mismo.
- **Requisitos de restricción.** Son los requisitos que especifican criterios que señalan una limitación de la funcionalidad y que, por lo tanto, no describen funciones a realizar.

La presentación de los requisitos de usuario se realizará a través de la siguiente plantilla.

RU[tipo]-[número]			
Título			
Descripción			
Prioridad		**Impacto**	
Fuente			

Tabla 3.1: Plantilla de tabla de requisito de usuario

- **Identificador** es el código alfanumérico que hace referencia de manera unívoca al requisito. La sintaxis del identificador sigue el patrón *RU*[*tipo*]-[*número*], siendo *RU* el indicador de requisito de usuario, [*tipo*] un carácter que indica si el requisito es de capacidad (*C*) o de restricción (*R*), y [*número*] una cifra incremental de dos dígitos que diferencia al requisito dentro de su tipo.

- **Título** es la cadena que resume la funcionalidad del requisito. Es más sencillo de recordar que el identificador y más breve que la descripción del mismo.

- **Descripción** es un breve texto explicativo del requisito, donde se detalla su información de forma clara y concisa.

- **Prioridad** es el grado de importancia o necesidad del requisito. Puede tomar los valores "Alta" (esencial), "Media" (deseable) o "Baja" (opcional).

- **Impacto** es el nivel de repercusión que se obtendría si el requisito no se llegase a implementar. Puede tomar los valores "Alto" (imprescindible), "Medio" (complementario) o "Bajo" (prescindible)

- **Fuente** indica la persona o documento del cual ha sido extraído el requisito. Los requisitos pueden proceder de exigencias del cliente o de recomendaciones de expertos en el ámbito que abarca el proyecto.

3.3.1. Requisitos de capacidad

RUC-01			
Título	Iniciar y salir de la aplicación del terminal Android.		
Descripción	El usuario podrá iniciar y cerrar la aplicación del terminal Android.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.2: Requisito de capacidad RUC-01

RUC-02			
Título	Activar el dispositivo Bluetooth del terminal Android.		
Descripción	El usuario podrá activar el dispositivo Bluetooth del terminal Android desde la aplicación si éste está desactivado.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.3: Requisito de capacidad RUC-02

RUC-03			
Título	Buscar dispositivos Bluetooth.		
Descripción	La aplicación podrá realizar una búsqueda de dispositivos Bluetooth cercanos para conectarse con ellos posteriormente.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.4: Requisito de capacidad RUC-03

RUC-04			
Título	Conectar y desconectar la aplicación a un dispositivo Bluetooth.		
Descripción	El usuario podrá conectar y desconectar la aplicación del terminal Android a un dispositivo Bluetooth.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.5: Requisito de capacidad RUC-04

RUC-05			
Título	Calibrar los niveles de ruido y de referencia del dispositivo.		
Descripción	La aplicación deberá poder calibrar los niveles de ruido y de referencia del dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.6: Requisito de capacidad RUC-05

RUC-06			
Título	Iniciar y detener la transmisión de datos.		
Descripción	La aplicación será capaz de iniciar y detener la transmisión de datos del dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.7: Requisito de capacidad RUC-06

RUC-07			
Título	Regular la intensidad de los LED.		
Descripción	La aplicación será capaz de regular la intensidad de los LED del dispositivo externo.		
Prioridad	Media.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.8: Requisito de capacidad RUC-07

RUC-08			
Título	Recibir datos del dispositivo.		
Descripción	La aplicación podrá recibir datos del dispositivo externo a través de la conexión Bluetooth.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.9: Requisito de capacidad RUC-08

RUC-09			
Título	Analizar y procesar los datos recibidos.		
Descripción	La aplicación debe ser capaz de analizar y procesar los datos recibidos en busca de incoherencias o errores producidos por el dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.10: Requisito de capacidad RUC-09

RUC-10			
Título	Mostrar los datos procesados.		
Descripción	La aplicación mostrará los datos procesados en cada instante a través de una gráfica.		
Prioridad	Alta.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.11: Requisito de capacidad RUC-10

RUC-11			
Título	Mostrar los niveles de saturación de oxígeno en sangre.		
Descripción	La aplicación mostrará los niveles de saturación de oxígeno en sangre en cada instante con los datos procesados.		
Prioridad	Alta.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.12: Requisito de capacidad RUC-11

RUC-12			
Título	Mostrar media de latidos por minuto.		
Descripción	La aplicación mostrará la media de los latidos por minutos en cada instante con los datos procesados.		
Prioridad	Alta.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.13: Requisito de capacidad RUC-12

RUC-13			
Título	Mostrar la frecuencia cardíaca.		
Descripción	La aplicación mostrará la frecuencia cardíaca en cada latido con los datos procesados a través de una gráfica.		
Prioridad	Alta.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.14: Requisito de capacidad RUC-13

RUC-14			
Título	Mostrar la respuesta galvánica de la piel.		
Descripción	La aplicación mostrará la respuesta galvánica de la piel (GSR) en cada latido con los datos procesados a través de una gráfica.		
Prioridad	Media.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.15: Requisito de capacidad RUC-14

RUC-15			
Título	Mostrar los niveles de saturación de oxígeno en sangre en sístole y diástole ventricular.		
Descripción	La aplicación mostrará los niveles de saturación de oxígeno en sangre en sístole y diástole ventricular con los datos procesados a través de una gráfica.		
Prioridad	Media.	**Impacto**	Medio.
Fuente	Cliente.		

Tabla 3.16: Requisito de capacidad RUC-15

RUC-16			
Título	Activar la reproducción de sonido.		
Descripción	La aplicación dará la opción de reproducir un sonido en cada latido cardíaco.		
Prioridad	Baja.	**Impacto**	Bajo.
Fuente	Cliente.		

Tabla 3.17: Requisito de capacidad RUC-16

RUC-17			
Título	Activar la vibración.		
Descripción	La aplicación dará la opción de hacer vibrar el terminal Android en cada latido cardíaco.		
Prioridad	Baja.	**Impacto**	Bajo.
Fuente	Cliente.		

Tabla 3.18: Requisito de capacidad RUC-17

RUC-18			
Título	Información de depuración.		
Descripción	El usuario podrá optar por activar o desactivar la información de depuración de los datos recibidos y los cálculos realizados por la aplicación.		
Prioridad	Baja.	**Impacto**	Bajo.
Fuente	Cliente.		

Tabla 3.19: Requisito de capacidad RUC-18

3.3.2. Requisitos de restricción

RUR-01			
Título	Sistema operativo y versión del terminal.		
Descripción	La aplicación deberá funcionar en cualquier terminal Android con una versión igual o superior a la 4.0 (ver tabla 2.3).		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.20: Requisito de restricción RUR-01

RUR-02			
Título	Compatibilidad con cualquier pantalla.		
Descripción	La aplicación deberá ser compatible con cualquier tamaño de pantalla ajustándose automáticamente en el arranque.		
Prioridad	Baja.	**Impacto**	Bajo.
Fuente	Cliente.		

Tabla 3.21: Requisito de restricción RUR-02

RUR-03			
Título	Idioma de la aplicación.		
Descripción	La aplicación deberá soportar tanto el idioma inglés como el español.		
Prioridad	Media.	**Impacto**	Bajo.
Fuente	Cliente.		

Tabla 3.22: Requisito de restricción RUR-03

RUR-04			
Título	Disponibilidad de dispositivo Bluetooth.		
Descripción	El terminal donde se ejecute la aplicación deberá disponer de un dispositivo Bluetooth.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.23: Requisito de restricción RUR-04

RUR-05			
Título	Conexión con el dispositivo externo.		
Descripción	La aplicación debe estar conectada siempre al dispositivo externo para permitir cualquier funcionalidad.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.24: Requisito de restricción RUR-05

RUR-06			
Título	Seguridad Bluetooth.		
Descripción	La aplicación deberá solicitar la introducción del código PIN en el caso de que sea necesario para la conexión con el dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.25: Requisito de restricción RUR-06

RUR-07			
Título	Comunicación con el dispositivo externo.		
Descripción	La comunicación entre el terminal Android y el dispositivo externo se realizará únicamente a través de Bluetooth.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.26: Requisito de restricción RUR-07

RUR-08			
Título	Tiempos de respuesta de la aplicación.		
Descripción	Los tiempos de respuesta de la aplicación desde que recibe los datos hasta que los muestra al usuario deben ser inferiores al tiempo de muestreo del dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.27: Requisito de restricción RUR-08

RUR-09			
Título	Información visual.		
Descripción	La aplicación deberá informar al usuario de manera visual del resultado de cualquier acción.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.28: Requisito de restricción RUR-09

RUR-10			
Título	Información de errores.		
Descripción	La aplicación comunicará los posibles errores y fallos de la aplicación al usuario de manera visual.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.29: Requisito de restricción RUR-10

RUR-11			
Título	Estabilidad de la aplicación.		
Descripción	La aplicación debe ser robusta frente a fallos y debe controlar todos los posibles errores que ocurran en tiempo de ejecución.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.30: Requisito de restricción RUR-11

RUR-12			
Título	Tolerancia máxima de errores.		
Descripción	La aplicación debe tener una tolerancia máxima del 10 % de error en los cálculos realizados.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.31: Requisito de restricción RUR-12

RUR-13			
Título	Verificación de datos.		
Descripción	La aplicación debe verificar y comprobar la integridad de todos los datos recibidos desde el dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.32: Requisito de restricción RUR-13

RUR-14			
Título	Comprobación de dispositivo.		
Descripción	La aplicación debe comprobar que el dispositivo al que se quiere conectar es un dispositivo compatible con la aplicación.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.33: Requisito de restricción RUR-14

RUR-15			
Título	Comprobar la calibración del dispositivo.		
Descripción	La aplicación deberá calibrar el dispositivo, al menos una vez, antes de empezar a recibir datos.		
Prioridad	Alta.	**Impacto**	Alto.
Fuente	Cliente.		

Tabla 3.34: Requisito de restricción RUR-15

3.4. Casos de uso

Los casos de uso son la representación gráfica de los requisitos de usuario, por lo que se necesita que hayan sido definidos previamente para el desarrollo de este apartado. El objetivo de este apartado es definir y modelar textualmente los casos de uso de la aplicación. Los casos de uso definen como interactúan los actores o entidades con el entorno, así como las respuestas obtenidas, en los distintos escenarios que se presentan.

Para el desarrollo de los casos de uso sólo se cuenta con un actor, el usuario, y con un entorno, la aplicación. En el siguiente diagrama se muestran los casos de uso obtenidos:

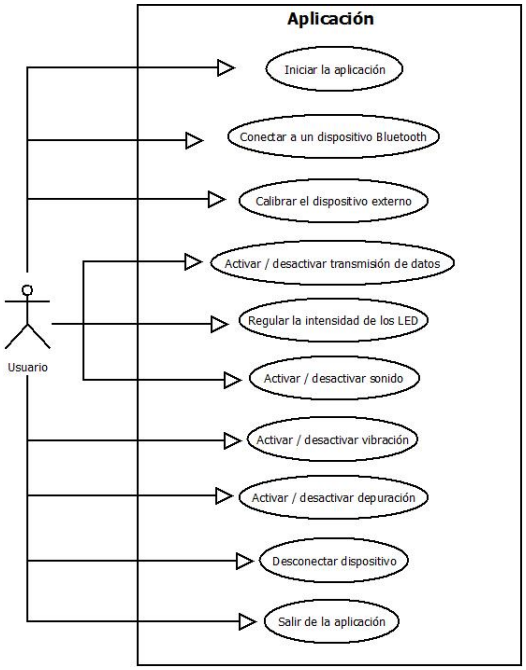

Figura 3.1: Casos de uso

A continuación se describen textualmente y de manera más extendida cada uno de los casos de la aplicación. La plantilla que va a utilizarse para la especificación de los casos de uso es la siguiente:

CU-[número]	
Título	
Escenario	
Precondiciones	
Postcondiciones	
Flujo normal	
Flujo alternativo	

Tabla 3.35: Plantilla de tabla de casos de uso

- **Identificador** es el código alfanumérico que hace referencia de manera unívoca al caso de uso. La sintaxis del identificador sigue el patrón *CU-*[*número*], siendo *CU* el indicador de caso de uso y [*número*] una cifra

incremental de dos dígitos que diferencia el caso de uso dentro de su tipo.

- **Título** es la cadena que resume la funcionalidad del del caso de uso. Es más sencillo de recordar que el identificador.

- **Escenario** es una breve descripción del estado del sistema antes de ejecutarse el caso de uso.

- **Precondiciones** es un listados de las condiciones o acciones que deben cumplirse o deben haber tenido lugar antes del inicio del caso de uso.

- **Postcondiciones** define el estado del sistema una vez ejecutado el caso de uso.

- **Flujo básico** es una enumeración ordenada de la acciones del usuario y de las respuestas del sistema que tendrán lugar durante la ejecución del caso de uso en condiciones normales. Es decir, son los pasos que se deben seguir para ejecutar la funcionalidad del caso de uso.

- **Flujo alternativo** son las posibles bifurnaciones en la secuencia de pasos descrita en el flujo básico que describen los casos de uso en los que la funcionalidad esperada no puede llevarse a cabo por determinadas circunstancias.

3.4.1. Casos de uso

CU-01	
Título	Iniciar la aplicación.
Escenario	El usuario inicia la aplicación desde su terminal.
Precondiciones	▪ El terminal está encendido y la aplicación está instalada.
Postcondiciones	▪ La aplicación se inicia correctamente.
Flujo normal	1. El usuario inicia la aplicación previamente instalada en el terminal Android. 2. La aplicación se inicia con normalidad y muestra el dialogo de conexión de dispositivos.
Flujo alternativo	2. *a*) Si el dispositivo Bluetooth del terminal Android está desconectado, se muestra la posibilidad de activarlo. En el caso de que el dispositivo Bluetooth no se active, la aplicación finaliza. *b*) Si la aplicación ha sido previamente inicializada y se ha conectado correctamente a algún dispositivo Bluetooth, tratará de volver a conectarse. En el caso de que no sea posible realizar la conexión se mostrará el dialogo de conexión de dispositivos.

Tabla 3.36: Caso de uso CU-01

CU-02	
Título	Conectar a un dispositivo Bluetooth.
Escenario	El usuario se quiere conectar con algún dispositivo Bluetooth.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación no está conectada con ningún dispositivo. ▪ La aplicación se encuentra en el dialogo de conexión de dispositivos.
Postcondiciones	▪ La aplicación se conecta con el dispositivo externo.
Flujo normal	1. El usuario selecciona el dispositivo al que se quiere conectar de la lista de dispositivos emparejados. 2. Se verifica que el dispositivo al que se quiere conectar está disponible y es compatible con los dispositivos de la aplicación. 3. La aplicación se conecta correctamente con el dispositivo.
Flujo alternativo	1. *a*) Si el dispositivo Bluetooth no está en la lista de dispositivos emparejados, el usuario puede realizar una busqueda de dispositivos cercanos y seleccionar el dispositivo de dicha lista. *b*) Si el usuario utiliza el botón de "volver" o cierra el dialogo de conexión sin haber realizado ninguna conexión, la aplicación se cerrará. 3. *a*) En el caso de que sea imposible realizar la conexión con el dispositivo, el usuario volverá al dialogo de conexión de dispositivos.

Tabla 3.37: Caso de uso CU-02

CU-03	
Título	Activar / desactivar la transmisión de datos del dispositivo externo.
Escenario	El usuario quiere activar la transmisión de datos del dispositivo externo.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La transmisión de datos del dispositivo externo se encuentra desactivada / activada.
Flujo normal	1. El usuario selecciona el botón para activar / desactivar el envío de datos. 2. La aplicación activa / desactiva el envío de datos.
Flujo alternativo	2. *a*) Si la aplicación no ha realizado la calibración de ruido con anterioridad, se mostrará el dialogo de calibración de ruido, donde se realizará dicha calibración. *b*) Si la aplicación no ha realizado la calibración de referencia con anterioridad, se mostrará el dialogo de calibración de referencia, donde se realizará dicha calibración.

Tabla 3.38: Caso de uso CU-03

CU-04	
Título	Calibrar el dispositivo externo.
Escenario	El usuario quiere realizar una calibración con el dispositivo externo.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La aplicación queda calibrada.
Flujo normal	1. El usuario selecciona la opción de calibración en la aplicación. 2. Se muestra el dialogo de calibración de niveles de ruido, donde se realizará dicha calibración. 3. Se muestra el dialogo de calibración de niveles de referencia, donde se realizará dicha calibración. 4. La aplicación queda calibrada.
Flujo alternativo	2. *a*) Si el dispositivo estaba transmitiendo datos, la aplicación detendrá la transmisión y procederá con la calibración.

Tabla 3.39: Caso de uso CU-04

CU-05	
Título	Regular la intensidad de los LED del dispositivo externo.
Escenario	El usuario quiere regular la intensidad de los LED del dispositivo externo.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ El LED cambia de intensidad.
Flujo normal	1. El usuario selecciona la opción del nivel de intensidad de un LED determinado. 2. La aplicación cambia la intensidad del LED del dispositivo externo.
Flujo alternativo	Inexistente.

Tabla 3.40: Caso de uso CU-05

CU-06	
Título	Activar / desactivar la reproducción de sonido.
Escenario	El usuario quiere activar / desactivar la reproducción de sonido.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La reproducción de sonido se encuentra desactivada / activada.
Flujo normal	1. El usuario selecciona la opción de activar / desactivar la reproducción de sonido. 2. La aplicación activa / desactiva la reproducción de sonido.
Flujo alternativo	Inexistente.

Tabla 3.41: Caso de uso CU-06

CU-07	
Título	Activar / desactivar la vibración del terminal.
Escenario	El usuario quiere activar / desactivar la vibración del terminal.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La vibración del terminal se encuentra desactivada / activada.
Flujo normal	1. El usuario selecciona la opción de activar / desactivar la vibración del terminal. 2. La aplicación activa / desactiva la vibración del terminal.
Flujo alternativo	Inexistente.

Tabla 3.42: Caso de uso CU-07

CU-08	
Título	Activar / desactivar la información de depuración.
Escenario	El usuario quiere activar / desactivar la información de depuración.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La información de depuración se encuentra desactivada / activada.
Flujo normal	1. El usuario selecciona la opción de activar / desactivar la información de depuración. 2. La aplicación activa / desactiva la información de depuración.
Flujo alternativo	Inexistente.

Tabla 3.43: Caso de uso CU-08

CU-09	
Título	Desconectar del dispositivo externo.
Escenario	El usuario se quiere desconectar el terminal del dispositivo externo.
Precondiciones	▪ La aplicación ha sido iniciada. ▪ La aplicación se ha conectado a un dispositivo.
Postcondiciones	▪ La aplicación no está conectada con ningún dispositivo.
Flujo normal	1. El usuario selecciona la opción de desconexión del dispositivo. 2. El terminal queda desconectado del dispositivo externo. 3. Se muestra el dialogo de conexión de dispositivos.
Flujo alternativo	Inexistente.

Tabla 3.44: Caso de uso CU-09

CU-10	
Título	Salir de la aplicación.
Escenario	El usuario quiere salir de la aplicación.
Precondiciones	▪ La aplicación ha sido iniciada.
Postcondiciones	▪ La aplicación está cerrada.
Flujo normal	1. El usuario sale de la aplicación.
Flujo alternativo	1. *a*) Si el dispositivo estaba transmitiendo datos, la aplicación detendrá la transmisión. *b*) Si el terminal estaba conectado con un dispositivo, la aplicación se desconectará de dicho dispositivo.

Tabla 3.45: Caso de uso CU-10

3.5. Requisitos de software

Los requisitos de software sirven de base a los desarrolladores para diseñar el sistema y definen de manera más técnica y detallada las características y capacidades del producto a desarrollar. La presentación de los requisitos se realizará siguiendo la información recogida en el estándar de especificación de requisitos IEEE 830 [30].

A continuación, se detallaran los requisitos de software de la aplicación, obtenidos a través del analisis de los requisitos de usuario y de la información obtenida del cliente. Los tipos de requisitos que se van a especificar son los siguientes:

- **Requisito funcional (F)** define una característica requerida del sistema que expresa una capacidad de acción del mismo. Estos requisitos se subdividen en los siguientes tipos:

    - **Funcionalidad** indican el funcionamiento básico de la aplicación a nivel de software.
    - **Comprobación** especifican las tareas que se deben realizar en determinadas acciones de la aplicación.
    - **Interfaz** definen como deben presentarse las características recogidas.

- **Requisito no funcional (N)** define una característica requerida del sistema que señala una restricción del mismo. Estos requisitos se subdividen en los siguientes tipos:

    - **Compatibilidad** define la restricción que limita lo adaptable que es el sistema a otros entornos.
    - **Operación** especifica cómo debe realizar la aplicación las tareas para las que fue definido.
    - **Verificación** indican los métodos de verificación de la entrada y salida de datos de la aplicación.
    - **Rendimiento** especifican los valores relacionados con la carga que se espera que tenga que soportar el sistema.
    - **Seguridad** indican los elementos que protegerán el software de accesos, usos y sabotajes maliciosos, así como de modificaciones o destrucciones maliciosas o accidentales.
    - **Fiabilidad** especificación aproximada de los valores entre los incidentes permisibles, o el total de incidentes permitidos.
    - **Disponibilidad** especifican los valores en los que el sistema ha de poder estar, como mínimo, operativo.
    - **Mantenibilidad** indican las tareas de mantenimiento que serán necesarias una vez que el software salga a producción.
    - **Portabilidad** especifican los atributos que debe presentar el software para facilitar su traslado a otras plataformas u entornos.

Al contrario que los requisitos de usuario, los requisitos de software han sido extraidos del propio desarrollador del sistema, a partir de los anteriores. Por lo tanto, a la hora de exponer textualmente los requisitos, no se especificará la fuente de los mismos, sino el requisito a partir del cual han sido obtenidos.

A continuación se muestra la plantilla que va a utilizarse para la especificación de los requisitos de software y la explicación de cada uno de los campos:

RS[tipo]-[número]			
Título			
Descripción			
Prioridad		**Impacto**	
Referencia			

Tabla 3.46: Plantilla de tabla de requisito de software

- **Identificador** es el código alfanumérico que hace referencia de manera unívoca al requisito. La sintaxis del identificador sigue el patrón *RS*[*tipo*]-[*número*], siendo *RS* el indicador de requisito de software, [*tipo*] un carácter que indica si el requisito es funcional (*F*) o no funcional (*N*), y [*número*] una cifra incremental de dos dígitos que diferencia al requisito dentro de su tipo.

- **Título** es la cadena que resume la funcionalidad del requisito. Es más sencillo de recordar que el identificador y más breve que la descripción del mismo.

- **Descripción** es un breve texto explicativo del requisito, donde se detalla su información de forma clara y concisa.

- **Prioridad** es el grado de importancia o necesidad del requisito. Puede tomar los valores "Alta" (esencial), "Media" (deseable) o "Baja" (opcional).

- **Impacto** es el nivel de repercusión que se obtendría si el requisito no se llegase a implementar. Puede tomar los valores "Alto" (imprescindible), "Medio" (complementario) o "Bajo" (prescindible)

- **Referencia** indica el requisito a partir del cual ha sido obtenido el requisito actual.

3.5.1. Requisitos funcionales

Requisitos de funcionalidad

RSF-01			
Título	Iniciar la aplicación en el terminal.		
Descripción	El usuario inicia la aplicación en el terminal Android donde debe estar previamente instalada.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-01.		

Tabla 3.47: Requisito funcional RSF-01

RSF-02			
Título	Cerrar la aplicación en el terminal.		
Descripción	El usuario cierra la aplicación en el terminal Android en el que se estaba ejecutando.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-01.		

Tabla 3.48: Requisito funcional RSF-02

RSF-03			
Título	Buscar dispositivos Bluetooth.		
Descripción	La aplicación realizará una busqueda de dispositivos Bluetooth cercanos y se los mostrará al usuario para que pueda realizar una conexión.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-03.		

Tabla 3.49: Requisito funcional RSF-03

RSF-04			
Título	Conectar a un dispositivo Bluetooth.		
Descripción	La aplicación intentará conectarse al dispositivo Bluetooth seleccionado por el usuario.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-04.		

Tabla 3.50: Requisito funcional RSF-04

RSF-05			
Título	Desconectar del dispositivo Bluetooth.		
Descripción	La aplicación intentará desconectarse del dispositivo Bluetooth conectado.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-04.		

Tabla 3.51: Requisito funcional RSF-05

RSF-06			
Título	Calibrar los niveles de ruido.		
Descripción	El usuario podrá calibrar los niveles de ruido del entorno actual y almacenarlos en la aplicación para futuros usos. Se informará al usuario del resultado de esta operación.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05.		

Tabla 3.52: Requisito funcional RSF-06

RSF-07			
Título	Calibrar los niveles de referencia.		
Descripción	El usuario podrá calibrar los niveles de referencia del entorno actual y almacenarlos en la aplicación para futuros usos. Se informará al usuario del resultado de esta operación.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05.		

Tabla 3.53: Requisito funcional RSF-07

RSF-08			
Título	Iniciar la transmisión de datos.		
Descripción	La aplicación solicitará al dispositivo el inicio de la transmisión de datos por medio del protocolo establecido.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-06.		

Tabla 3.54: Requisito funcional RSF-08

RSF-09			
Título	Detener la transmisión de datos.		
Descripción	La aplicación solicitará al dispositivo la detención de la transmisión de datos por medio del protocolo establecido.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-06.		

Tabla 3.55: Requisito funcional RSF-09

RSF-10			
Título	Regular la intensidad de los LED.		
Descripción	La aplicación regulará la intensidad de los LED por medio del protocolo establecido con el dispositivo externo.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-07.		

Tabla 3.56: Requisito funcional RSF-10

RSF-11			
Título	Recibir datos del dispositivo.		
Descripción	Cuando la transmisión de datos haya sido activada, la aplicación recibirá datos desde el dispositivo externo siguiendo un formato preestablecido en un intervalo de tiempo predefinido.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-08.		

Tabla 3.57: Requisito funcional RSF-11

RSF-12			
Título	Procesar los datos recibidos.		
Descripción	La aplicación procesará los datos recibidos. Para ello, necesitará recoger una serie de datos que compondrán la ventana de trabajo. Sobre esta ventana se aplicarán los filtros de suavizado de señales para obtener la frecuencia cardíaca, y los cálculos necesarios para obtener los niveles de saturación en sangre.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-09, RUC-10, RUC-11,RUC-12, RUC-13, RUC-14 y RUC-15.		

Tabla 3.58: Requisito funcional RSF-12

RSF-13			
Título	Reproducción de sonido.		
Descripción	Si la opción de reproducción de sonido se encuentra activada, la aplicación deberá reproducir un sonido predefinido cuando se produzca un latido cardíaco en los datos procesados.		
Prioridad	Baja.	**Impacto**	Bajo.
Referencia	RUC-16.		

Tabla 3.59: Requisito funcional RSF-13

RSF-14			
Título	Vibración del terminal.		
Descripción	Si la opción de vibración del terminal se encuentra activada, la aplicación deberá activar la vibración del terminal cuando se produzca un latido cardíaco en los datos procesados.		
Prioridad	Baja.	**Impacto**	Bajo.
Referencia	RUC-17.		

Tabla 3.60: Requisito funcional RSF-14

Requisitos de comprobación

RSF-15			
Título	Existencia de dispositivo Bluetooth.		
Descripción	La aplicación comprobará que el terminal desde el que se está ejecutando cuenta con un dispositivo Bluetooth funcional.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-02 y RUR-04.		

Tabla 3.61: Requisito funcional RSF-15

RSF-16			
Título	Conexión previa con el dispositivo.		
Descripción	La aplicación comprobará si se ha realizado previamente una conexión con algún dispositivo Bluetooth e intentará reproducirla automaticamente.		
Prioridad	Media.	**Impacto**	Bajo.
Referencia	RUC-04.		

Tabla 3.62: Requisito funcional RSF-16

RSF-17			
Título	Calibración previa de la aplicación.		
Descripción	La aplicación comprobará si se ha realizado alguna calibración anterior y la almacenará como la calibración actual.		
Prioridad	Media.	**Impacto**	Medio.
Referencia	RUC-05.		

Tabla 3.63: Requisito funcional RSF-17

RSF-18			
Título	Calibración de la aplicación.		
Descripción	La aplicación comprobará si se ha realizado alguna calibración y, en caso negativo, la solicitará antes de permitir activar la transmisión de datos.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05, RUC-06 y RUR-15.		

Tabla 3.64: Requisito funcional RSF-18

RSF-19			
Título	Desconexión con el dispositivo.		
Descripción	Antes de cerrarse, la aplicación comprobará que la conexión de datos con el dispositivo externo ha sido cerrada y la aplicación se ha desconectado del mismo.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-01 y RUC-04.		

Tabla 3.65: Requisito funcional RSF-19

Requisitos de interfaz

RSF-20			
Título	Pantalla de conexión de dispositivos.		
Descripción	La pantalla de conexión de dispositivos mostrará una lista con los dispositivos enlazados con el terminal y un botón que permitirá buscar los dispositivos no enlazados que se encuentren cercanos al terminal.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-03, RUC-04 y RUR-05.		

Tabla 3.66: Requisito funcional RSF-20

RSF-21			
Título	Pantalla principal.		
Descripción	La pantalla principal de la aplicación mostrará la información de la media de latidos por minuto y de los niveles de saturación en sangre, así como una gráfica básica de la información que se está procesando desde el dispositivo. Se monstrarán tres botones que permitirán realizar las operaciones básicas de la aplicación: iniciar / detener, desconectar y calibrar.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-10, RUC-11, RUC-12, RUC-04, RUC-05 y RUC-06.		

Tabla 3.67: Requisito funcional RSF-21

RSF-22			
Título	Pantalla de gráficas.		
Descripción	En la pantalla de gráficas se mostrarán la información detallada de los valores procesados, la frecuencia cardíaca, el estado de ánimo y los niveles de StO_2 en sístole y diástole ventricular.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-10,RUC-12, RUC-13, RUC-14 y RUC-15.		

Tabla 3.68: Requisito funcional RSF-22

RSF-23			
Título	Pantalla de opciones.		
Descripción	En la pantalla de opciones se mostrarán las opciones de la aplicación: activar / desactivar sonido, activar / desactivar vibración, activar / desactivar depuración, regular la intensidad de los LED y la información de depuración si ha sido activada.		
Prioridad	Media.	**Impacto**	Bajo.
Referencia	RUC-07,RUC-16, RUC-17 y RUC-18.		

Tabla 3.69: Requisito funcional RSF-23

RSF-24			
Título	Idioma de la interfaz.		
Descripción	La interfaz de la apliación se configurará automaticamente para presentarse tanto en español como en inglés en función del idioma establecido en el terminal.		
Prioridad	Media.	**Impacto**	Bajo.
Referencia	RUR-03.		

Tabla 3.70: Requisito funcional RSF-24

RSF-25			
Título	Dialogos secundarios.		
Descripción	La aplicación deberá informar al usuario de manera visual del resultado de cualquier acción, como la calibración, la conexión / desconexión o la recepción de los datos, así como de cualquier error producido durante la ejecución.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-09 y RUR-10.		

Tabla 3.71: Requisito funcional RSF-25

3.5.2. Requisitos no funcionales

Requisitos de compatibilidad

RSN-01			
Título	Sistema operativo del terminal.		
Descripción	Para que la aplicación pueda ser instalada y ejecutada el terminal debe contar con el sistema operativo Android.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-01.		

Tabla 3.72: Requisito no funcional RSN-01

RSN-02			
Título	Versión del sistema operativo.		
Descripción	Para que la aplicación pueda ser instalada y ejecutada el terminal debe contar como mínimo con la versión 4.0 de Android (ver tabla 2.3).		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-01.		

Tabla 3.73: Requisito no funcional RSN-02

RSN-03			
Título	Tamaño de pantalla del terminal.		
Descripción	La aplicación se debe ajustar a cualquier tamaño de pantalla automáticamente.		
Prioridad	Baja.	**Impacto**	Bajo.
Referencia	RUR-02.		

Tabla 3.74: Requisito no funcional RSN-03

RSN-04			
Título	Compatibilidad del dispositivo externo.		
Descripción	La aplicación comprobará que el dispositivo al que se conecta es un dispositivo compatible con la aplicación mediante la petición de una operación y verificación de la respuesta obtenida.		
Prioridad	Media.	**Impacto**	Medio.
Referencia	RUR-14.		

Tabla 3.75: Requisito no funcional RSN-04

Requisitos de operación

RSN-05			
Título	Calibración de los niveles de ruido.		
Descripción	Para realizar la calibración de niveles de ruido, la aplicación debe apagar los LED del dispositivo externo y solicitar la transmisión de datos. Cuando se reciba un conjunto de datos establecidos por el desarrollador, se promediarán y se almacenarán.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05.		

Tabla 3.76: Requisito no funcional RSN-05

RSN-06			
Título	Calibración de los niveles de referencia.		
Descripción	Para realizar la calibración de niveles de referencia, la aplicación debe alertar al usuario de que use un polímero en el dispositivo, despues de esto, encenderá los LED del dispositivo externo y solicitará la transmisión de datos. Cuando se reciba un conjunto de datos establecidos por el desarrollador, se promediarán y se almacenarán.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05.		

Tabla 3.77: Requisito no funcional RSN-06

RSN-07			
Título	Comunicación Bluetooth.		
Descripción	La comunicación con el dispositivo externo se realizará unicamente a través del protocolo Bluetooth.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-07.		

Tabla 3.78: Requisito no funcional RSN-07

Requisitos de verificación

RSN-08			
Título	Analizar los datos recibidos.		
Descripción	La aplicación comprobará que los datos recibidos siguen el protocolo preestablecido y que mantienen una integridad secuencial lógica, tanto numérica como temporal.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-09 y RUR-13.		

Tabla 3.79: Requisito no funcional RSN-08

RSN-09			
Título	Integridad en la calibración.		
Descripción	La aplicación verificará la integridad entre la calibración de ruido y de referencia para que no existan problemas o errores de cálculo posteriores.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUC-05 y RUR-13.		

Tabla 3.80: Requisito no funcional RSN-09

Requisitos de rendimiento

RSN-10			
Título	Tiempo de procesamiento de datos.		
Descripción	La aplicación debe analizar, procesar y mostrar los datos recibidos en un instante de tiempo en un intervalo inferior al tiempo de recepción del siguiente dato recibido. De esta forma se mantendrá la lógica de tiempo real de la aplicación.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-08.		

Tabla 3.81: Requisito no funcional RSN-10

Requisitos de seguridad

RSN-11			
Título	Solicitud de código PIN.		
Descripción	La aplicación solicitará la introducción de un código PIN que debe coincidir con el establecido por defecto en el dispositivo externo.		
Prioridad	Media.	**Impacto**	Bajo.
Referencia	RUR-06.		

Tabla 3.82: Requisito no funcional RSN-11

Requisitos de fiabilidad

RSN-12			
Título	Tolerancia máxima de errores.		
Descripción	La aplicación debe tener una tolerancia máxima del 10 % de error en los cálculos realizados.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-12.		

Tabla 3.83: Requisito no funcional RSN-12

Requisitos de disponibilidad

RSN-13			
Título	Estabilidad de la aplicación.		
Descripción	La aplicación debe ser robusta frente a fallos y debe controlar todos los posibles errores que ocurran en tiempo de ejecución.		
Prioridad	Alta.	**Impacto**	Alto.
Referencia	RUR-11.		

Tabla 3.84: Requisito no funcional RSN-13

3.6. Matrices de trazabilidad

Las matrices de trazabilidad son herramientas que se utilizan para saber que requerimientos de usuario son cubiertos por la documentación obtenida a partir de ellos. Para ello se enfrentarán los requisitos en una tabla y se marcará su relación con cada uno de ellos. En este apartado se van a diferenciar dos tipos de matrices de trazabilidad:

- Matriz de trazabilidad de requisitos de usuario a casos de uso (ver apéndice A).
- Matriz de trazabilidad de requisitos de usuario a requisitos de software (ver apéndice B).

Las matrices se han ubicado en el apéndice del documento para no interferir en el flujo de lectura del mismo.

Parte III

Diseño e implementación del prototipo

4

Diseño del sistema

Siguiendo el análisis realizado en el capítulo 3, se llevará a cabo la fase de diseño del sistema que servirá de introducción de cada una de las partes de las que se compone y que recogerá en detalle el diseño del prototipo del sistema en conjunto y donde se justificarán cada una de las decisiones tomadas.

4.1. Arquitectura

El sistema está compuesto por dos partes, el pulsioxímetro y la aplicación Android, que funcionan en conjunto para proporcionar la funcionalidad definida en los apartados anteriores.

- El pulsioxímetro se encarga de recoger los datos del usuario a través del fotodiodo y de enviarlo por Bluetooth simulando un puerto serial.
- La aplicación Android se ocupa de recibir los datos enviados, analizarlos, procesarlos y mostrarle al usuario la información desglosada.

El objetivo de este capítulo es plantear el diseño y la implementación de la aplicación Android. Sin embargo, debido a que la aplicación sin el dispositivo externo es incapaz de funcionar, se ha decidido incluir el diseño de este dispositivo en el mismo documento para mejorar la calidad de la exposición. Adicionalmente, se incluirá el desarrollo de una aplicación que actúa como servidor de pruebas con el que se ha realizado el desarrollo real de la aplicación.

4.1.1. Modelo cliente-servidor

El sistema en conjunto hace uso del paradigma cliente-servidor, donde el servidor es el pulsioxímetro que recoge y envía los datos, y el cliente es la aplicación que los recibe y los procesa. Es decir, el sistema sigue el modelo de aplicación distribuida donde un cliente realiza peticiones al servidor, quien le da respuesta.

Un sistema distribuido permite a los usuarios finales obtener acceso a la información de forma transparente incluso en entornos multiplataforma. En este modelo, las tareas se reparten entre el cliente y el servidor. De este modo, la capacidad de proceso está repartida entre los clientes y los servidores, aunque son mas importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

El modelo de diseño que se plantea en este documento es una versión reducida del modelo cliente-servidor, ya que únicamente existirá la conexión entre un servidor y un cliente simultáneamente en cada sistema. Esto simplifica en gran medida el diseño y la implementación global del entorno.

Figura 4.1: Modelo cliente-servidor

4.2. Despliegue

El despliegue del sistema consiste en diferenciar los elementos hardware necesarios para que el sistema se pueda ejecutar, así como los diversos entornos de ejecución y artefactos software y hardware requeridos para este fin. A través de un diagrama UML se puede representar la disposición de estos elementos como se puede observar en la figura 4.2.

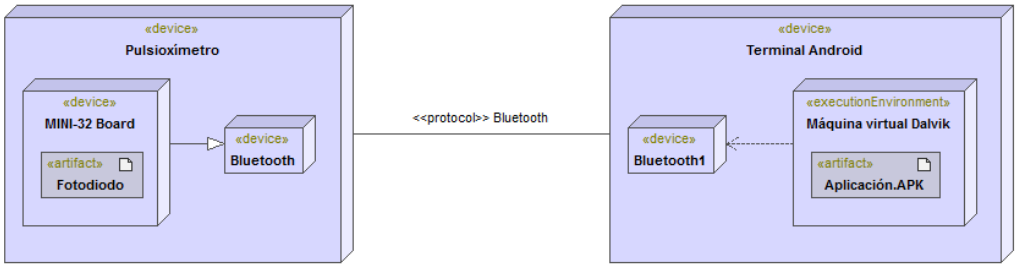

Figura 4.2: Diagrama de despliegue del sistema

En el lado del servidor, se dispone de una placa MINI-32 Board que se encarga de integrar el fotodiodo con los LED emisores y, a su vez, de enviar la señal hacia el dispositivo Bluetooth del mismo, que actua como puerto serial de transmisión de datos.

En el cliente, se dispone de un terminal Android donde se ejecuta la máquina virtual Dalvik que se encarga de ejecutar la aplicación principal. La aplicación se comunica con la máquina virtual a través de su interfaz para interactuar con el dispositivo Bluetooth integrado en el terminal. Los dos dispositivos interactúan a través de sus dispositivos Bluetooth mediante un procolo establecido.

4.3. Funcionamiento

El funcionamiento del sistema en conjunto es simple e intuitivo para el usuario. En la figura 4.3 se observan los elementos principales que lo componen. El funcionamiento paso a paso es el siguiente:

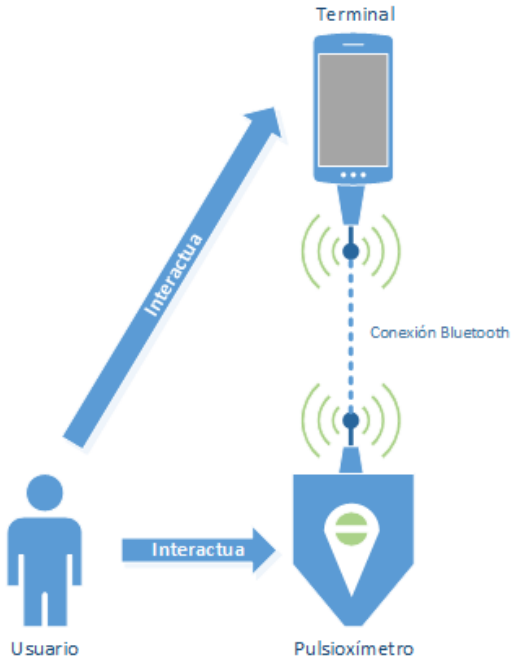

Figura 4.3: Diagrama de funcionamiento del sistema

1. El usuario interactúa con la aplicación del terminal para conectarla con el dispositivo externo o pulsioxímetro.

2. Una vez realizada la conexión, la aplicación solicitará la calibración del dispositivo, con lo que indicará una serie de pasos que debe seguir.

3. Cuando el dispositivo esté calibrado, el usuario debe situar el dedo índice en el pulsioxímetro y activar la transmisión de datos en la aplicación. El dispositivo comenzará entonces a enviar los datos del usuario a la aplicación que los procesará.

4. El usuario puede interactuar con la aplicación para ver el resultado de sus análisis en tiempo real.

Una vez que el usuario haya conectado y calibrado el dispositivo por primera vez, la aplicación tratará de conectarse automáticamente y asumirá la primera calibración como la actual, dando la posibilidad al usuario de recalibrar el sistema en cualquier momento.

4.4. Protocolo de comunicación

Para que la comunicación entre la aplicación y el dispositivo sea efectiva se necesitan establecer una serie de protocolos que permitirán interactuar con cada parte del sistema.

4.4.1. Comunicación pulsioxímetro-aplicación

La comunicación entre el pulsioxímetro y la aplicación se realizará de forma continua y sin pausas, por lo que es necesario establecer un estándar que permita verificar la validez y la coherencia de un determinado paquete emitido en el tiempo. Los datos emitidos por el pulsioxímetro siguen el formato que se muestra a continuación:

Figura 4.4: Formato de comunicación entre dispositivo y aplicación

- **Cabecera.** La cabecera se compone de un byte prefijado al valor 0xFF que permite identificar el inicio de un paquete.
- **Dato 1.** Valor entero de 2 bytes que envía la información recogida por el primer LED del dispositivo. El rango de valores que puede enviar es $[0, 1023]$.
- **Dato 2.** Valor entero de 2 bytes que envía la información recogida por el segundo LED del dispositivo. El rango de valores que puede enviar es $[0, 1023]$.
- **Dato 3.** Valor entero de 2 bytes que envía la información recogida por el tercer LED del dispositivo. El rango de valores que puede enviar es $[0, 1023]$.
- **Dato 4.** Valor entero de 2 bytes que envía la información recogida por el cuarto LED del dispositivo. El rango de valores que puede enviar es $[0, 1023]$.
- **Contador.** Valor entero de un byte que almacena el número de paquete de manera incremental. Cuando el valor se desborda comienza de cero.

4.4.2. Comunicación aplicación-pulsioxímetro

La comunicación entre la aplicación y el pulsioxímetro es necesaria para poder establecer un orden lógico en los datos que se deben enviar desde el dispositivo, ya que éste carece de ningún tipo de lógica de proceso.

El formato del mensaje de comunicación consiste en un byte que almacena el comando de la operación que se solicita. El estado del dispositivo en cada momento debe ser almacenado y gestionado por la aplicación, ya que el dispositivo externo no transmite ningún tipo de información de estados. Los posibles comandos que se pueden enviar son los siguientes:

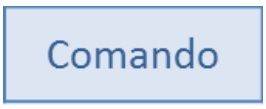

Figura 4.5: Formato de comunicación entre aplicación y dispositivo

- **Start (0x61).** Inicia la transmisión de paquetes desde el dispositivo externo al terminal Android. El muestro se realizará con una frecuencia de 50 veces por segundo, es decir, cada 20 milisegundos.
- **Stop (0x62).** Detiene la transmisión de paquetes.
- **Intensidad.** Cambia la intensidad de los LED. El byte a enviar sigue el formato 0x$\alpha\beta$, donde α es un valor de los siguientes:
  - **A.** Identifica al LED 1.
  - **B.** Identifica al LED 2.
  - **C.** Identifica al LED 3.
  - **D.** Identifica al LED 4.

  Y β, corresponde con el valor de la intensidad de la siguiente forma:
  - **1.** Corresponde con la intensidad baja (*low*).
  - **2.** Corresponde con la intensidad media (*medium*).
  - **3.** Corresponde con la intensidad alta (*high*).
  - **4.** Corresponde con el LED apagado (*off*).

4.4.3. Calibración y datos

Una vez establecido el protocolo de comunicación entre dispositivos, es necesario establecer el orden lógico en el que se ejecutará. El funcionamiento del pulsioxímetro define que una vez que se emita el comando de iniciar la transmisión de datos, el dispositivo comenzará a muestrear las cuatro señales LED y a enviar paquetes de datos sucesivamente y de forma continua.

Por lo tanto, el único factor variante que permitirá modificar las señales recibidas serán las intensidades de los LED. A continuación se van a definir los tres posibles escenarios de transmisión y recepción de datos.

- **Calibración de ruido.** Para muestrear los datos de ruido del entorno se necesita establecer la intensidad de los LED en apagado (*off*) y comenzar la emisión de datos. Es recomendable que el dispositivo se encuentre vacío en el momento de dicha calibración.
- **Calibración de referencia.** En esta calibración es necesario modificar la intensidad de los LED a media (*medium*) e introducir un polímero o elemento que reflecte la luz emitida por el LED.
- **Transmisión de datos.** En este caso la intensidad de los LED podrá ser variable y el usuario podrá introducir el dedo en el dispositivo para comenzar el muestreo de señales.

Los tres casos anteriores los gestionará la aplicación en orden de tener todos los datos necesarios para realizar los cálculos.

5

Diseño e implementación de la aplicación

A continuación se describirá el diseño y la implementación de la aplicación Android desarrollada. Se incluirán diagramas de clases que facilitarán la comprensión del diseño de la aplicación.

5.1. Arquitectura MVC

El propio sistema operativo de Android está diseñado de forma que las aplicaciones desarrolladas para esta plataforma sigan el patrón arquitectónico modelo-vista-controlador (MVC). La principal ventaja de este paradigma de programación consiste en separar los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos que se relacionan para formar la aplicación. De este modo, se puede diferenciar:

- **Modelo.** Es la representación específica de la información con la cual operará la aplicación. El modelo envía a la vista aquella parte de la información que es solicitada para ser mostrada en cada momento. Las peticiones de acceso o manupulación de la información llegan al modelo a través del controlador. En Android, el modelo puede obtenerse de distintas formas, en nuestro caso se consigue a través de los datos recibidos desde el dispositivo.

- **Vista.** Presenta el modelo en un formato adecuado para interactuar con el usuario, es decir, las interfaces de usuario. En Android, las interfaces de usuario se construyen usando XML.

- **Controlador.** Responde a eventos que normalmente son emitidos por las acciones del usuario, e invoca peticiones al modelo cuando se hace

alguna solicitud sobre la información. En resumen, el controlador hace de intermediario entre la vista y el modelo. En Android, el controlador lo forman todas las clases que permiten utilizar las interfaces y permiten desplegar y procesar la información del usuario.

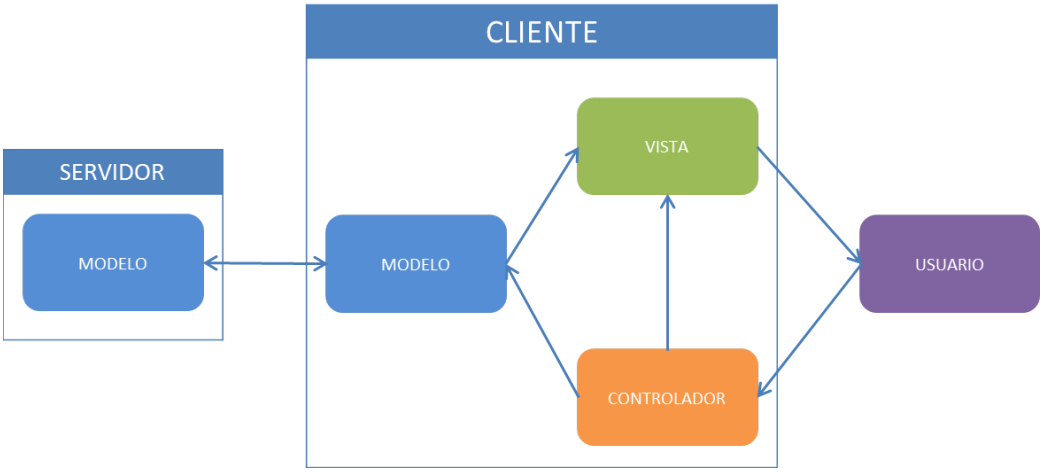

Figura 5.1: Arquitectura MVC

5.2. Framework Android

Una de las principales tareas que hay que definir cuando se realiza un desarrollo de software es la elección de un *framework* o entorno de desarrollo. A la hora de diseñar el proyecto se barajaron dos opciones: Java y Qt.

La mayor parte de las aplicaciones Android se desarrollando en Java y se ejecutan en la máquina virtual Dalvik. Sin embargo, Android solo tomó de este lenguaje la sintaxis de código y no mantuvo las API como las que se pueden encontrar en Java SE y Java ME. Los servicios de los dispositivos Android, como la pantalla táctil y el almacenamiento, funcionan a través de la API de servicios de Google.

Qt es un *framework* multiplataforma que proporciona soporte para varios lenguajes de programación como Python, Ruby y Perl, aunque las aplicaciones están escritas fundamentalmente en C++. El entorno de desarrollo de Qt es conocido por su facilidad de uso, así como por su gran soporte en la interfaz de usuario de los clientes de escritorio. En sus últimos desarrollos se han incluido mejoras para la plataforma móvil bajo el nombre de *Qt Mobility* y *Qt Quick*.

El componente principal de la arquitectura de Android es la reutilización de componentes, que permite compartir actividades, servicios y datos con otras aplicaciones (ver subsección 2.3.1). Sin embargo, Qt contempla de otra forma el manejo de las actividades utilizando diversos componentes, como pueden ser las máquinas de estados. Con las primeras versiones de Qt, el desarrollo de la interfaz de usuario era lento y pesado para los usuarios, sin embargo, con la introducción de *Qt Mobility* y *Qt Quick*, el desarrollo de aplicaciones se llevó a otro nivel.

Figura 5.2: Logo Qt

Qt Quick proporciona un mecanismo de declaración de objetos en arbol usando el lenguaje QML. Con la ayuda de *Qt Declarative* se combina con lo anterior para integrar los objetos QML con los creados en C++, generando de esta forma aplicaciones nativas de una forma rápida y limpia.

La idea inicial a la hora de diseñar el proyecto, fue usar las herramientas que proporciona el *framework* Qt para el desarrollo completo de la aplicación. Sin embargo, y debido a que este *framework* es totalmente ajeno a la comunidad Android, la portabilidad de las API necesarias no estaban totalmente funcionales en el momento de diseño. El principal inconveniente encontrado fue la incapacidad de usar el dispositivo Bluetooth integrado en el terminal de manera nativa con el *framework*, ya que este solo estaba soportado para dispositivos *Nokia*.

Para solucionar esta carencia de funcionalidades en el *framework*, los desarrolladores de Qt proporcionan una herramienta conocida como *Java Native Interface* (JNI). El JNI es un entorno de desarrollo que permite que un programa escrito en Java ejecutado en la JVM pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador. Como continuación del planteamiento anterior, se intentó continuar el diseño en Qt haciendo uso del JNI, sin embargo, su uso no es en absoluto trivial y requería un tiempo de estudio superior al propuesto en el proyecto y por lo tanto se descartó su uso.

El proyecto finalmente se ha realizado usando las herramientas Java oficiales aportadas para los desarrolladores de Android.

5.3. Cálculos y algoritmos

En esta sección se detallarán todos los cálculos y algoritmos realizados en la aplicación para procesar los datos que se reciben del dispositivo. Se va a dividir en dos bloques: por un lado el cálculo de la StO_2 y por otro lado el cálculo de la frecuencia cardíaca a través del filtrado de señales.

Los datos leídos del pulsioxímetro son un conjunto de cuatro valores que corresponden a la reflexión de cuatro longitudes de onda diferentes: 460 nm (azul), 580 nm (verde), 640 nm (amarillo) y 850 nm (infrarrojo). Cada una de estas longitudes de onda poseen propiedades que permiten realizar los cálculos que se presentan a continuación. En la figura 5.3 se pueden observar las equivalencias de las longitudes de onda recibidas.

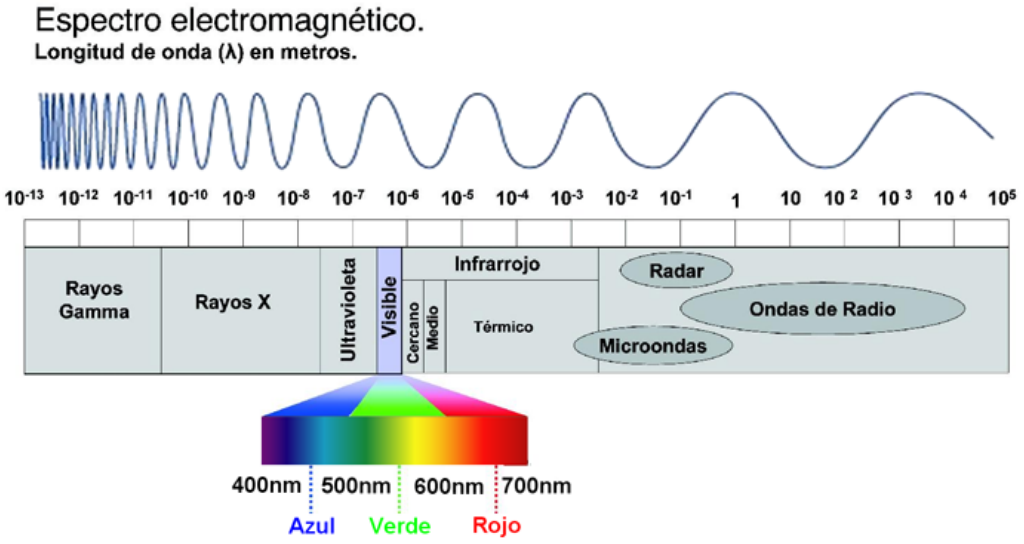

Figura 5.3: Espectro electromagnético

5.3.1. Cálculo de los niveles de saturación de oxígeno en sangre

En el cálculo de los niveles de saturación de oxígeno en sangre se utilizarán dos longitudes de onda: de 640 nm y de 850 nm. La razón de la elección de

estas longitudes de onda se explica en la sección 2.1.4. Para que la explicación se este apartado sea mas clara y sencilla se resolverán las ecuaciones utilizadas a continuación.

La calibración de la aplicación es necesaria antes de realizar este cálculo. Una vez que la aplicación haya sido calibrada se obtendrán dos valores: el nivel de ruido en el ambiente y el valor de luz o de referencia en el mismo. Estos dos valores sitúan un máximo y un mínimo de los valores leídos a continuación. A partir de estos valores de calibración, se puede calcular la densidad óptica de los valores leídos aplicando la siguiente ecuación [31].

$$\mu_a(\lambda_j) = OD(\lambda_j) = -log(\frac{\lambda_j - \lambda_D}{\lambda_R - \lambda_D}) \tag{5.1}$$

Donde,

- $\mu_a(\lambda_j)$ es el coeficiente de absorción de una longitud de onda λ_j determinada.
- $OD(\lambda_j)$ es la densidad óptica de una longitud de onda λ_j determinada.
- λ_D es el valor de la calibración de ruido del entorno leído.
- λ_R es el valor de la calibración de referencia del entorno leído.

Una vez que se ha calculado la densidad óptica de cada una de las dos longitudes de onda, se necesita calcular la concentración de hemoglobina oxigenada y hemoglobina sin oxigenar a partir de esos valores. A partir de la ecuación 2.7 y la ecuación 5.1 se obtiene que:

$$OD(\lambda_j) = \varepsilon_{HbO_2}(\lambda_j)C_{HbO_2} + \varepsilon_{Hb}(\lambda_j)C_{Hb} \tag{5.2}$$

Donde,

- $\varepsilon_{HbO_2}(\lambda_j)$ y $\varepsilon_{Hb}(\lambda_j)$ son los coeficientes de extinción molar de la hemoglobina oxigenada y hemoglobina desoxigenada respectivamente en función de una longitud de onda dada. Estos valores se obtienen a partir de la tabla de coeficientes de extinción molar de hemoglobina en agua de Prahl [13].
- C_{HbO_2} y C_{Hb} son las concentraciones de hemoglobina oxigenada y hemoglobina desoxigenada en sangre respectivamente.

A partir de la ecuación anterior se puede concluir el siguiente sistema de ecuaciones:

$$\begin{cases} OD(\lambda_A) = \varepsilon_{HbO_2}(\lambda_A)C_{HbO_2} + \varepsilon_{Hb}(\lambda_A)C_{Hb} \\ OD(\lambda_B) = \varepsilon_{HbO_2}(\lambda_B)C_{HbO_2} + \varepsilon_{Hb}(\lambda_B)C_{Hb} \end{cases} \tag{5.3}$$

Donde, A y B son las longitudes de onda leídas respectivamente. A partir de esta ecuación, se calcula la concentración de hemoglobina para cada caso y mediante la ecuación 2.8 se calcula el porcentaje de saturación de oxígeno en sangre StO_2.

5.3.2. Cálculo de la frecuencia cardíaca

El cálculo de la frecuencia cardíaca consiste en el análisis de la señal recibida desde el dispositivo para detectar los máximos en la misma. Para realizar los cálculos necesarios de esta sección se han utilizado distintas señales recogidas de dispositivos médicos homologados. Estos dispositivos proporcionan información de 1024 longitudes de onda distintas.

Determinación de la longitud de onda

Una de las primeras fases es la determinación de una buena longitud de onda que proporcionara información suficiente para los cálculos posteriores. Para ello, se ha analizado la variación máxima de la amplitud en cada longitud de onda para observar cual de todas posee una mayor variación en el tiempo. El siguiente experimento se realizó con un análisis de 76 segundos en intervalos de 20 milisegundos.

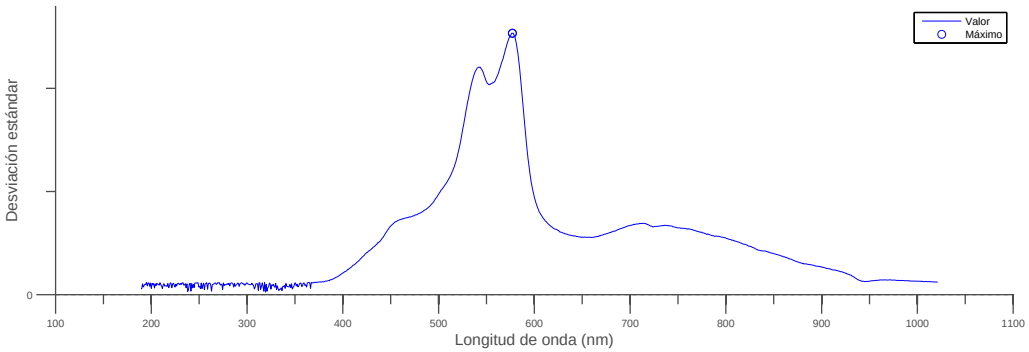

Figura 5.4: Desviación estándar de la amplitud de las longitudes de onda

Como se puede observar en la figura 5.4, la longitud de onda que más variación recibe es la de 577 nm. En el proyecto se ha utilizado la longitud de onda de 580 nm para redondear y simplificar los cálculos numéricos. En la muestra de ejemplo anterior, la longitud de onda de 580 nm corresponde con la frecuencia cardíaca.

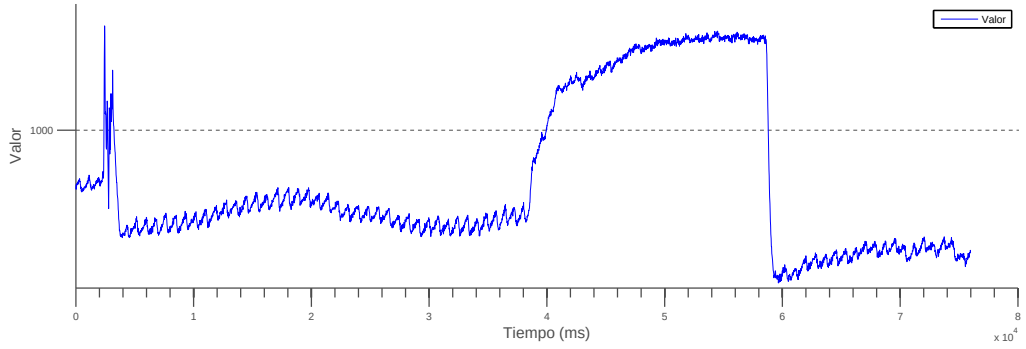

Figura 5.5: Señal completa de ejemplo

Suavizado de la señal

Una vez que se ha determinado cual es la señal que se va a tratar, se ha conocer cómo es dicha señal. Con ese propósito y para simplificar las siguientes gráficas se ha decidido trabajar con un tamaño 500 muestras de las expuestas anteriormente.

En la figura 5.6 se puede observar que la señal obtenida presenta ruido tanto de altas como de bajas frecuencias. El objetivo principal es tratar de conseguir eliminar el ruido de altas frecuencias y recuperar, con el mínimo error posible, la apariencia de la señal sinusoidal real. Se debe tener en cuenta que la capacidad de cálculo que se posee es limitada y no se pueden ejecutar algoritmos de filtrado complejos.

Para la representación de las gráficas se utilizó el programa de cálculo numérico MatLab R2013a. A continuación se muestra la señal parcial de ejemplo sin aplicar ningún filtro.

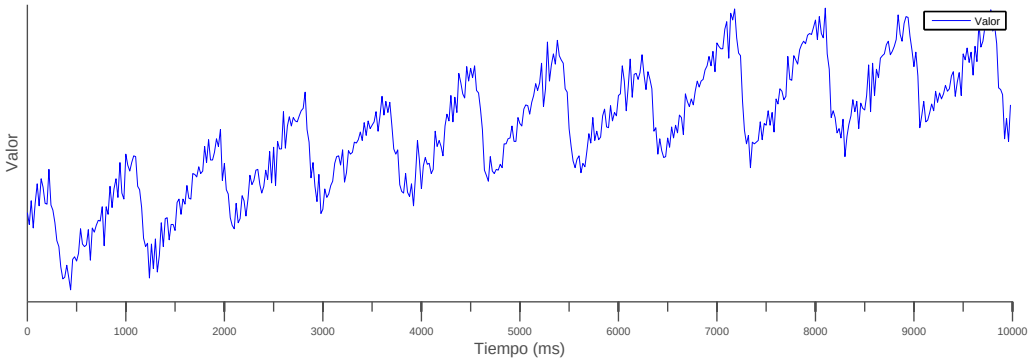

Figura 5.6: Señal parcial de ejemplo sin tratar

Media Móvil Previa El primer algoritmo a probar fue la media móvil previa. La media móvil es un algoritmo de filtrado de paso bajo que consiste en aplicar la media aritmética de los m datos anteriores.

$$MM = \frac{\sum_{i=n-m}^{n} x(i)}{m} \tag{5.4}$$

Donde $x(n)$ es un punto y m son los datos usados para calcular la media.

La media móvil es un método que comparte gran sinergia con la aplicación, ya que no es necesario esperar a los datos posteriores, sino que se ejecuta sobre el histórico de datos recibido. La ejecución del algoritmo se realizó con un valor de $m = 5$. Los resultados obtenidos sobre la señal parcial original son los siguientes:

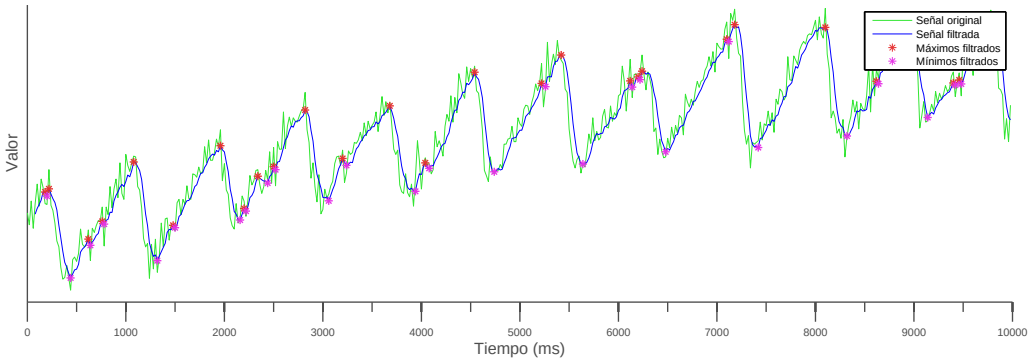

Figura 5.7: Señal parcial de ejemplo filtrada con la media móvil

Como se puede observar, se trata de un buen y rápido mecanismo de filtrado. Sin embargo, no filtra con exactitud la onda de frecuencias que nos da la información de la frecuencia cardíaca.

Transformada rápida de Fourier La transformada rápida de Fourier (*Fast Fourier Transform*, FFT) es un eficiente algoritmo que permite calcular la transformada discreta de Fourier (*Discrete Fourier Transform*, DFT) y su inversa. La FFT tiene gran importancia en una amplia variedad de aplicaciones, desde el tratamiento digital de señales y filtrado digital en general, a la resolución de ecuaciones en derivadas parciales o los algoritmos de multiplicación rápida de grandes números enteros.

La DFT se obtiene por la descomposición de una secuencia de valores en componentes de diferentes frecuencias. Esta operación es útil es muchos campos, pero computacionalmente es muy lenta para trabajar con ella. La FFT es una forma computacional de obtener el mismo resultado de manera mas rápida. Procesar la DFT de N puntos dados tiene un coste computacional de $O(N^2)$ operaciones aritméticas, mientras que la FFT puede procesar el mismo resultado en en solo $O(NlogN)$ operaciones.

Sean $x_0, ..., x_{N-1}$ números complejos. La fórmula de la DFT es la siguiente:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \qquad k = 0, ..., N-1 \tag{5.5}$$

Existen muchos algoritmos para la implementación de la FFT, sin embargo, se ha utilizado la implementación del algoritmo de Cooley-Tukey [32]. Se ha utilizado una implementación optimizada del algoritmo por R. Sedgewick and K. Wayne [33] para realizar las pruebas en los terminales Android. En el ejemplo siguiente se ha aplicado el filtro FFT y sobre el resultado se han eliminado el 87 % de las altas frecuencias.

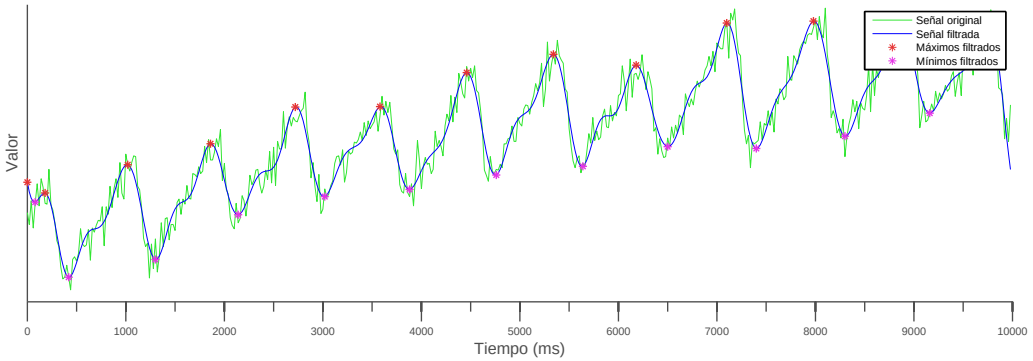

Figura 5.8: Señal parcial de ejemplo filtrada con la FFT

Como se puede observar, es mucho mas fiable y exacta que la media móvil, sin embargo, el coste computacional es muy alto. En la figura 5.8 se puede observar perfectamente la señal sinusoidal del pulso cardíaco definido con los máximos en cada onda.

Filtro de Savitzky–Golay El filtro de Savitzky–Golay es un tipo de filtro digital que se puede aplicar a un conjunto de señales de puntos discretas con el objetivo de suavizar los datos. Es decir, se intenta aumentar el ratio de señal-ruido sin distorsionar la misma. El proceso a seguir se conoce como convolución y consiste en el ajuste de un conjunto sucesivo de datos adyacentes con un polinomio de grado bajo mediante el método de mínimos cuadrados. Este algoritmo de filtrado fue diseñado y publicado por Abraham Savitzky y Marcel J. E. Golay en 1964 [34].

Dado un conjunto de datos $n\{x_j, y_j\}$ con puntos $(j = 1, ..., n)$, donde x es una variable independiente e y_j es un valor observado. Utilizando una serie de m coeficientes de convolución C_i, se obtiene que:

$$Y_j = \sum_{i=-(m-1)/2}^{i=(m-1)/2} C_i \, y_{j+i} \qquad \frac{m+1}{2} \leq j \leq n - \frac{m-1}{2} \tag{5.6}$$

Se ha utilizado una implementación optimizada del algoritmo por Matthew B. Smith [35] basada en la implementación en C del mismo en “Numerical Methods in C: The art of scientific computing second edition”. En el ejemplo siguiente se ha aplicado el filtro de Savitzky–Golay utilizando un polinomio de orden 2 y un tamaño de ventana de 25. Ambos valores se han ido ajustando para obtener el mejor resultado.

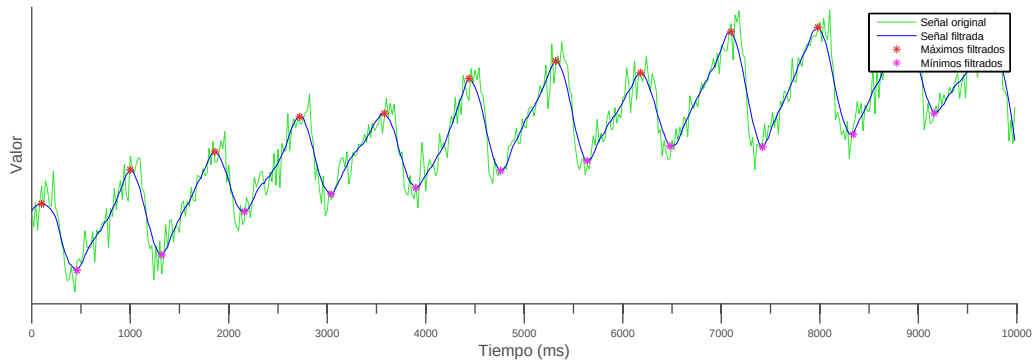

Figura 5.9: Señal parcial de ejemplo filtrada con el filtro Savitzky–Golay

En la figura 5.9 se observa como el algoritmo se ajusta perfectamente con la sinusoidal sin terminar de emularla. Se puede afirmar que es un poco menos preciso que el resultado obtenido por la FFT, pero mucho más exacto que el obtenido con la media móvil.

Tiempos de respuesta Todos los algoritmos fueron probados en distintos dispositivos, donde se analizaron los resultados obtenidos. Los dispositivos usados fueron dos terminales Android distintos: un Samsung Nexus S y un Samsung Galaxy 4.

	Samsung Nexus S	**Samsung Galaxy S4**
Procesador	ARM C-A8	Quad-core C-A15
Velocidad	1024 MHz	1.6 GHz
Memoria RAM	512 Mb	2Gb
Sistema Operativo	Android 4.1.2	Android 4.4.2
Tamaño pantalla	4.0 pulgadas	5.0 pulgadas
Gama	Media/Baja	Alta

Tabla 5.1: Tabla comparativa de dispositivos

En la tabla 5.1 se puede observar que el Samsung Galaxy S4 tiene una capacidad de cálculo muy superior al Samsung Nexus S. Sin embargo, durante el análisis de la aplicación no se solicitó ningún requisito de restricción que incluyera la velocidad de procesamiento del terminal Android. Por lo tanto,

se optó por elegir un algoritmo que pudiera funcionar tanto en un dispositivo de gama alta, como en uno de gama baja.

En la tabla 5.2 se detallan los tiempos de ejecución de los algoritmos descritos anteriormente. Estos se han obtenido realizando tres ejecuciones distintas de cada uno de los algoritmos en cada dispositivo y promediándolas. Finalmente se ha decidido utilizar el algoritmo de filtro de Savitzky–Golay, ya que, aunque el mejor algoritmo de filtrado es la FFT, el tiempo de cálculo es demasiado alto para la potencia de computación limitada del Samsung Nexus S. Es decir, el filtro de Savitzky–Golay tiene un mejor ratio de resultado-tiempo. Sin embargo, si este proyecto se portara a dispositivos con una potencia de cálculo superior, se debería utilizar el algoritmo de la FFT.

	Samsung Nexus S	**Samsung Galaxy S4**
Media móvil previa	0,24 ms	0,07 ms
FFT	18,61 ms	6,7 ms
Filtro Savitzky–Golay	1,46 ms	0,37 ms

Tabla 5.2: Tabla comparativa de tiempos

Detección de picos y distancias

Una vez que se dispone de una señal filtrada es necesario detectar los picos que equivalen al pulso cardíaco. El algoritmo de filtro de Savitzky–Golay no se puede aplicar sobre una señal de datos a tiempo real, por lo que es necesario esperar por una ventana de N valores antes de poder iniciar el filtrado de datos. El tamaño de ventana inicial es de 64 datos, lo que equivale a un segundo, ya que la señal se procesa a 50 datos por segundo.

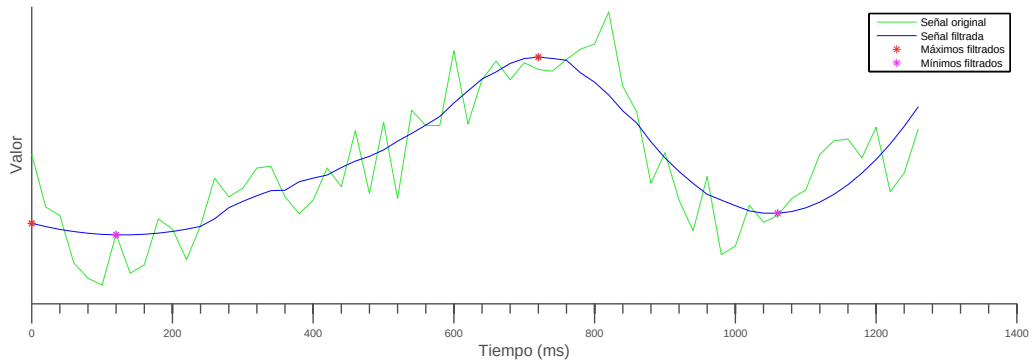

Figura 5.10: Ventana de datos filtrada

Una vez que se dispone de una ventana filtrada, se escogen los valores centrales para su análisis ya que son los que dan los resultados mas fiables. Sobre ellos se calcula la segunda derivada para calcular los máximos y los mínimos relativos, y el valor central se almacena como dato procesado. El conjunto de datos procesados se representa en las gráficas de los valores filtrados de la aplicación.

Según se van recibiendo datos del dispositivo, la ventana de filtro de va actualizando y nuevos valores se van añadiendo a la lista de valores procesados. Una vez que se detecta el siguiente máximo, se compara con el anterior en el tiempo y se calcula la frecuencia cardíaca en PPM. La frecuencia calculada se representa en la gráfica de frecuencia cardíaca correspondiente.

$$f(PPM) = \frac{60}{m_2(s) - m_1(s)} \tag{5.7}$$

Para medir el estado emocional a partir de la señal, que se representa como la amplitud del pulso [36], se utilizan los mínimos procesados. Una vez que se reciben dos mínimos se calcula la recta que pasa entre ellos. Sobre esa recta se calcula la distancia con el máximo que se encuentre entre los dos mínimos anteriores. Dicha distancia se representa en la gráfica del estado anímico del usuario.

$$d = y(max) - \frac{(x(max) - x(min_1))(y(min_2) - y(min_1))}{x(min_2) - x(min_1)} + y(min_1) \tag{5.8}$$

5.4. Interfaz de usuario

La interfaz de usuario (*User Interface*, UI) es el medio con el que el usuario interactúa para comunicarse con la aplicación. El diseño de una buena interfaz de usuario es una de las fases mas importantes a la hora de desarrollar una aplicación o software. Una buena interfaz debe ser sencilla e intuitiva para el usuario.

Es necesario conocer las funcionalidad que ofrece la aplicación a la hora de realizar el diseño de la interfaz de usuario, ya que de esta forma se podrá organizar de manera que sea simple para el usuario. La aplicación cuenta con tres interfaces de usuario principales y otras secundarias. Dentro de las interfaces principales se encuentran las secciones *principal*, *gráficas* y *ajustes*. Estas interfaces se relacionan por medio de un diseño de pestañas desde las que se puede alternar entre una sección u otra. A continuación se detallaran cada una de estas secciones y se comentará como han sido realizadas.

5.4.1. Sección *principal*

La sección *principal* es la sección por defecto a la hora de inicializar la aplicación. En ella se dispone de cuatro subsecciones.

- La subsección de la frecuencia cardíaca, en la que se presenta las pulsaciones por minuto medias del usuario (PPM), junto a un corazón que aumenta su tamaño a través de una animación cada vez que se recibe un pulso en la aplicación.
- La subsección de la StO_2, en la que se indica el porcentaje medio de saturación de oxígeno en sangre. Alrededor del número indicativo, se muestra un circulo junto a un óvalo que crece o decrece según el porcentaje de StO_2, siendo el 100 % el circulo completo y el 0 % el circulo vacío.
- La subsección de los valores leidos y procesados del dispositivo externo, en la que se muestra la información de los valores leídos y procesados en tiempo real en forma de gráfica para comprobar que se ha colocado el dispositivo correctamente. Esta gráfica no presenta ningún tipo de información relevante ni a nivel de valores ni a nivel de posición, simplemente es indicativa.
- La subsección de botones, en la que se disponen tres botones: el botón de inicio/detención de la transmisión de datos, el botón de desconexión del dispositivo Bluetooth y el botón de calibración de la aplicación.

El diseño inicial de esta sección se realizó pensando en la manera más sencilla de implentación, por lo que se desarrollo utilizando tecnología web a través de un *WebView*. Un *WebView* es un *widget* de Android que permite la ejecución de contenido HTML y Javascript en la propia aplicación. Sin embargo, cuando se realizó la primera batería de pruebas de rendimiento en la aplicación, se comprobó que un *WebView* no aportaba el rendimiendo mínimo requerido, ya que las animaciones y los tiempos de muestreo de la información eran excesivamente lentos (del orden de 100 milisegundos).

Figura 5.11: Sección *principal* de la UI

Por lo tanto, se optó por rehacer todo el contenido de esta sección de manera nativa utilizando componentes Android. El anillo circular usado para representar la StO_2 se realizó implementando un *widget* personalizado y dibujando cada circulo en función del porcentaje del valor leído. La gráfica que representa los valores procesados se mantuvo en un *WebView* por simplicidad ya que el rendimiento con los otros cambios mejoró considerablemente.

5.4.2. Sección *gráficas*

La sección *gráficas* se encuentra a la derecha de la sección anterior en el menú de pestañas y se encarga de disponer todas las gráficas detalladas de la aplicación de manera legible para el usuario. Las gráficas que se presentan en esta sección se detallan a continuación.

- La primera gráfica es la gráfica que representa los valores leídos y procesados en el tiempo. Es similar a la gráfica de la sección anterior, pero en ésta se detallan los ejes y los valores. Cada punto en el eje X corresponde con un dato muestreado. La frecuencia de actualización de la gráfica es a tiempo real, es decir, cada vez que se recibe un dato del dispositivo y es procesado. El contenido de está gráfica no aporta

información relevante para el usuario, sin embargo, se decide situarla para poder comprar los valores obtenidos en el resto de gráficas con los valores reales. Es decir, sirve de guía para el usuario que interpreta el resto de gráficas.

- La segunda gráfica es la gráfica que indica la frecuencia cardíaca con a cada latido en el tiempo. Las unidades del eje Y son los PPM y tiene un rango fijado entre 50 y 100 PPM. La frecuencia de actualización de la gráfica es cada vez que se recibe una pulsación, es decir, aproximadamente cada 800 ms se dibuja un punto.

- La tercera gráfica corresponde con la gráfica que representa el estado de ánimo del usuario con cada latido en el tiempo. La frecuencia de actualización de la gráfica es en cada mínimo de la sinusoide procesada, similar a la gráfica anterior.

- La última gráfica representa los valores de la StO_2 en la sístole y la diástole ventricular con cada latido en el tiempo. Las unidades del eje Y corresponden con el porcentaje calculado y no tienen rango fijo. La frecuencia de actualización es la misma que en la segunda gráfica.

Figura 5.12: Sección *gráficas* de la UI

5.4.3. Sección *ajustes*

La sección *ajustes* es la sección que contiene todas las opciones de la aplicación. Normalmente, en las aplicaciones Android, esta sección solo es accesible mediante el menú. Sin embargo, se ha optado por ponerla más visible ya que contiene información necesaria para realizar una buena toma de datos. Los ajustes de los que se dispone son los siguientes:

- Habilitar/deshabilitar sonido.
- Habilitar/deshabilitar vibración.
- Habilitar/deshabilitar información de depuración.
- Modificar la intensidad de los LED. Para cada uno de los cuatro LED es posible modificar su intensidad por separado. Los valores que pueden tomar estas intensidades son: apagado (*off*), bajo (*low*), medio (*medium/par*) y alto (*high*).

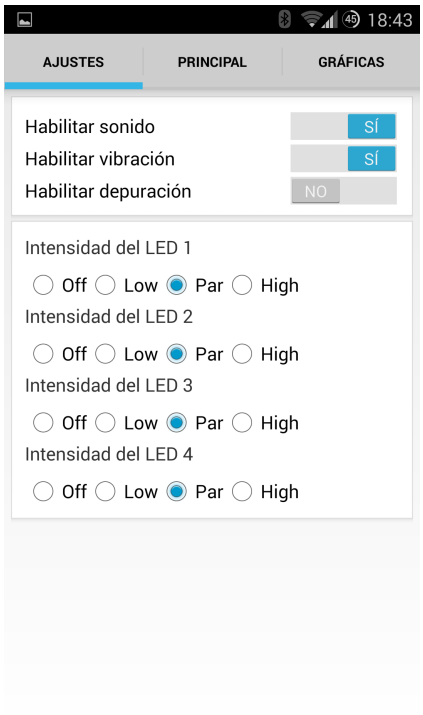

Figura 5.13: Sección *ajustes* de la UI

5.4.4. Interfaces secundarias

Dentro de la aplicación existen otras interfaces de usuario que intervienen en acciones, como la conexión de dispositivos y la calibración de la aplicación, y que no necesitan estar presentes en las secciones principales descritas anteriormente.

La interfaz de conexión de dispositivos permite al usuario elegir un dispositivo enlazado previamente para conectarse. En el caso de que no se encuentre el dispositivo en dicha lista, se muestra un botón que permite realizar una búsqueda de dispositivos cercanos.

Figura 5.14: Interfaz de conexión de dispositivos

La interfaz de calibración instruye al usuario de lo que tiene que hacer para realizar correctamente la calibración de la aplicación. Se divide en dos interfaces con una serie de ordenes según el valor a calibrar.

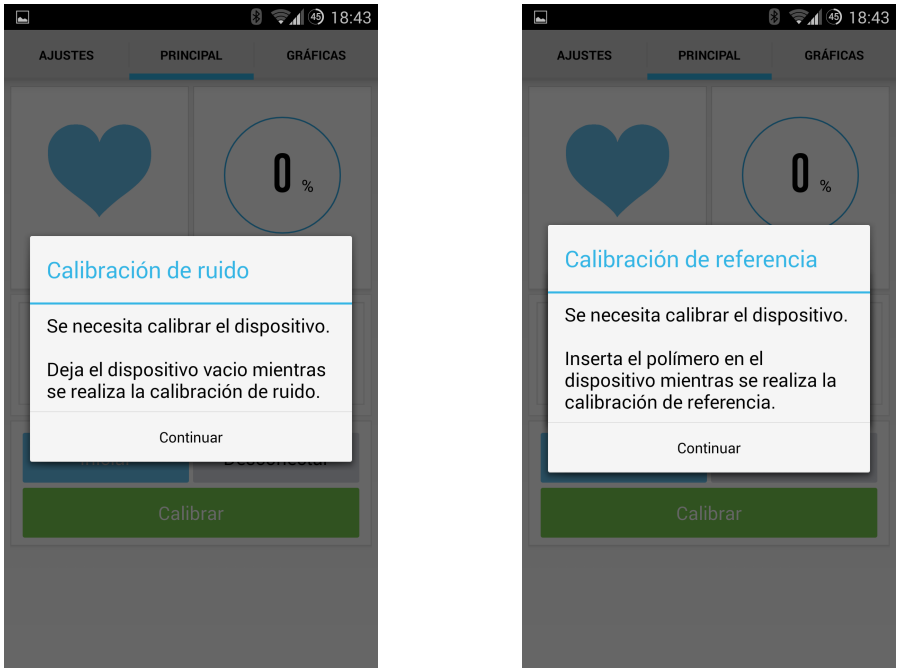

Figura 5.15: Interfaz de calibración

5.5. Diagrama de clases

Un diagrama de clases describe la estructura de una aplicación mostrando las clases con sus métodos, atributos y visibilidad de los mismos, y las relaciones entre ellas.

Los posibles modificadores de visibilidad de los métodos y atributos de las clases son los siguientes:

- ***Public***. Indica que el atributo o método es visible para cualquier clase. Se representa mediante el signo "+" en el diagrama de clases.
- ***Private***, representado con el signo "-", indica que el atributo o método sólo es visible dentro de la clase donde se define.
- ***Protected***. Se representa mediante el signo "#" e indica que el atributo o método es visible en la clase donde se define y en cualquiera de sus subclases.
- ***Package***. Representado con un espacio " " y sin ningún modificador explícito en el código. Indica que el atributo o método solo es visible para las clases dentro del propio paquete.

En este caso, no se van a mostrar los atributos o los métodos en el diagrama general para hacer más sencilla la visión global de la aplicación, pero serán comentados en el siguiente apartado.

En el diagrama, podrán encontrarse tres tipos de relaciones distintas:

- **Asociaciones.** Están ilustradas con flechas trazadas con línea continua y con la punta abierta. Indican que dentro de una clase existe un atributo del tipo de la clase a la que se está apuntando. El nombre del atributo no se muestra junto a la relación, y el valor de visibilidad de ese atributo se coloca delante del nombre en forma de signo.

- **Dependencias.** Se muestran con flechas con líneas discontinuas con la punta abierta y sin ningún texto junto a ellas. Indican que una clase depende de la creación y uso de objetos de otra clase, aunque no los almacene como atributos.

- **Asociaciones de propiedad de elementos.** Son líneas continuas con un signo "+" dentro de una circunferencia en el lado del propietario, que indican que una clase está anidada a otra. La clase anidada solo puede ser utilizada por la clase en la que se encuentra anidada.

Para conseguir una mayor facilidad en la lectura se ha desglosado el diagrama de clases en partes. Las clases serán definidas con más detalle en la siguiente sección. Este primer diagrama representa la relación de los paquetes que contienen las clases de aplicación entre sí. Las clases se han estructurado en paquetes que contienen las clases relacionadas con una determinada funcionalidad.

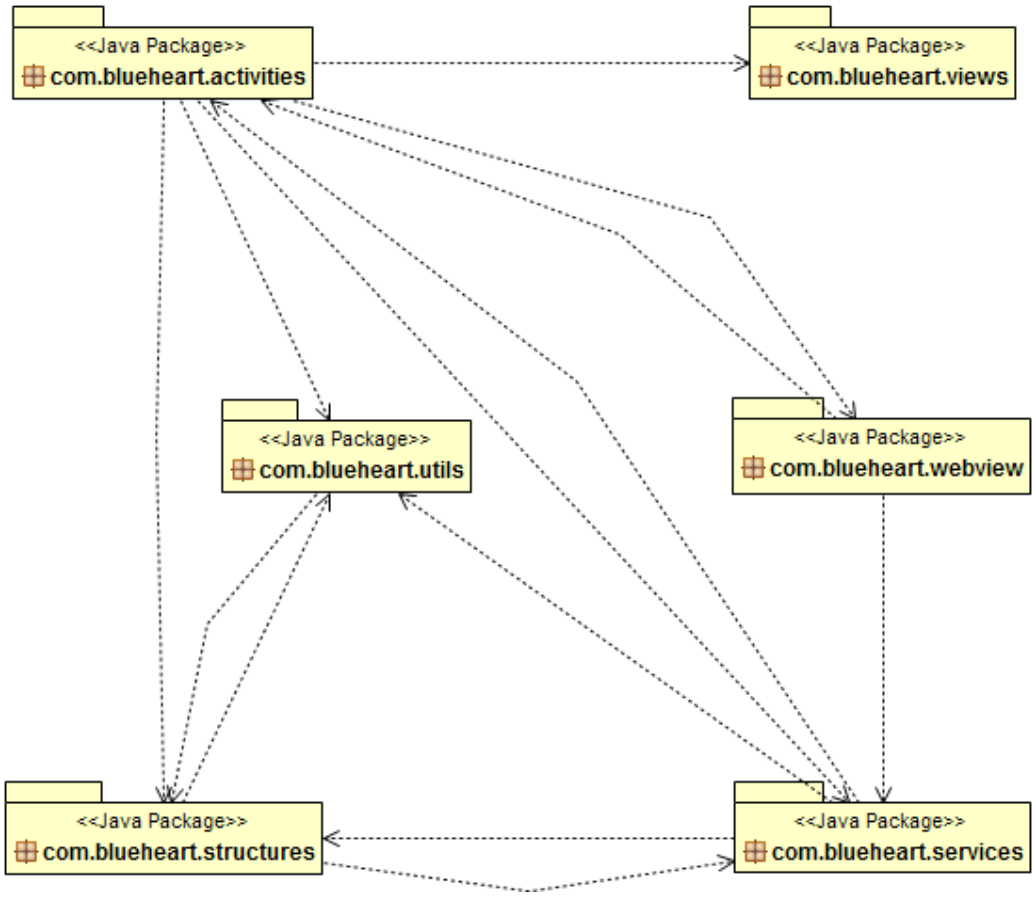

Figura 5.16: Diagrama de clases de paquetes

- ***com.blueheart.activities***. Contiene todas las actividades y secciones de la aplicación.
- ***com.blueheart.services***. Contiene los servicios globales que se emplean para gestionar los datos de la aplicación.
- ***com.blueheart.structures***. Contiene las estructuras de la aplicación.
- ***com.blueheart.utils***. Contiene clases estáticas de algoritmos y funciones.
- ***com.blueheart.views***. Contiene las vistas y elementos visuales específicos de la aplicación.
- ***com.blueheart.webview***. Contiene las clases relacionadas con el *webview*.

A continuación, se muestra una visión global y reducida de los módulos de funcionamiento de la aplicación. En cada uno de ellos, se puede observar como interactúan las distintas clases de la aplicación para obtener la funcionalidad exigida.

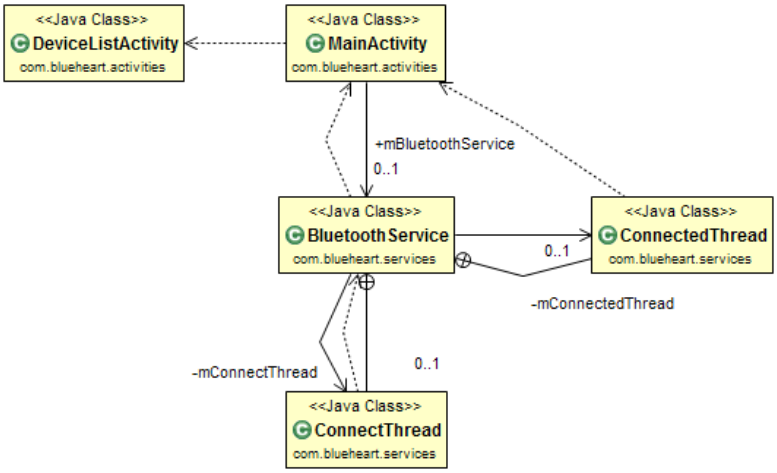

Figura 5.17: Diagrama de clases de conexión Bluetooth

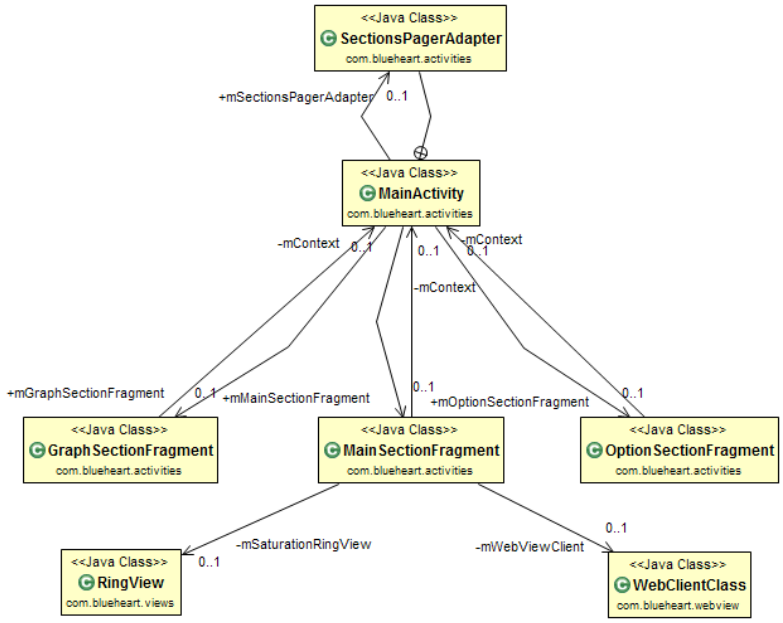

Figura 5.18: Diagrama de clases de actividades

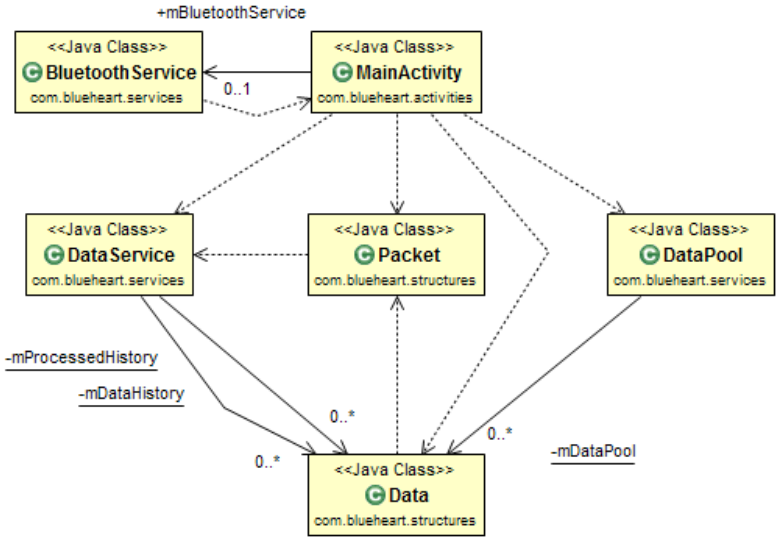

Figura 5.19: Diagrama de clases de recogida de datos

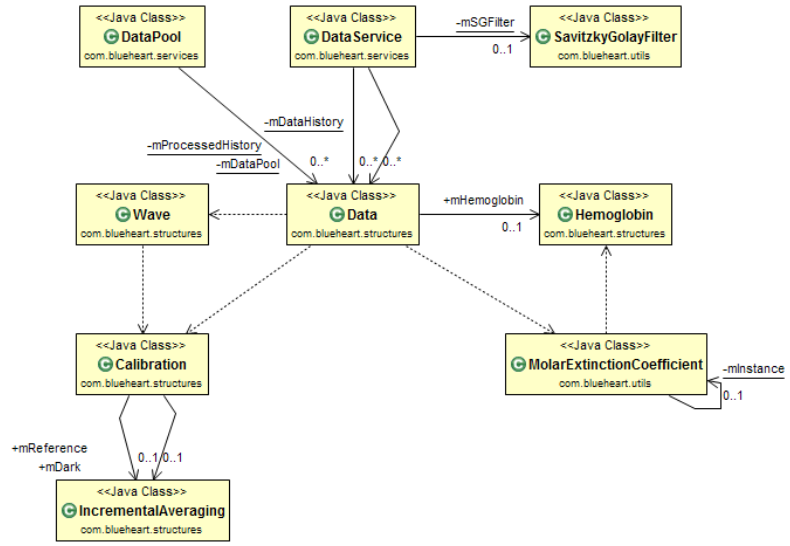

Figura 5.20: Diagrama de clases de procesamiento de datos

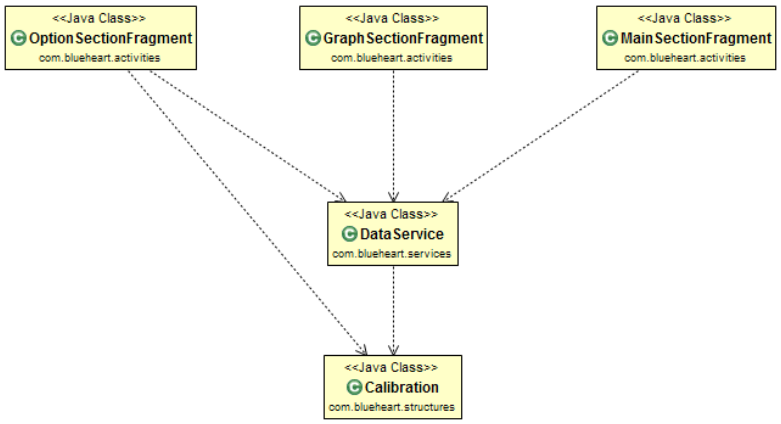

Figura 5.21: Diagrama de clases de representación de datos

5.6. Definición de clases

A continuación se documentarán de manera breve la función de las clases que componen la aplicación, así como los atributos y métodos más relevantes de las mismas. La descripción completa de todas las clases, con sus atributos y métodos, se puede encontrar en la documentación generada por *Javadoc* del proyecto.

5.6.1. Paquete `com.blueheart.activities`

Clase `MainActivity`

Actividad principal de la aplicación. Se encarga de gestionar la funcionalidad de los estados de la aplicación (ver figura 2.19), así como de permitir la navegación por *pestañas* entre las distintas secciones de la aplicación: *ajustes*, *principal* y *gráficas*. Dentro de la propia clase existe la subclase `SectionsPagerAdapter` que permite personalizar esta navegación.

La funcionalidad más importante de la clase `MainActivity` es mantener activos todos los servicios y de hacer de intermediario entre el resto de clases de la aplicación. Por ejemplo, la actividad `BluetoothService`, que recupera los datos recibidos por Bluetooth, los envía a la clase `MainActivity` que se encarga de gestionarlos con sus respectivos servicios y redirigir la salida al *fragment* que se esté mostrando en cada momento. Para permitir esta comunicación entre clases se hace uso de un *Handler* o manejador de mensajes en Android.

Clase `DeviceListActivity`

Clase por defecto al crear un proyecto de Bluetooth en el SDK de Android. No se han realizado modificaciones importantes en esta clase ya que no eran necesarias.

Clase `MainSectionFragment`

Fragmento de actividad que contiene todos los elementos visuales de la sección *principal* de la aplicación. Inicialmente, su diseño se realizó utilizando solo *WebView* por motivos de sencillez. Sin embargo, por motivos de rendimiento, se optó por convertirlo todo a nativo dejando únicamente la gráfica principal en WebView. Se estableció un tiempo de refresco de interfaz de 2 FPS en este fragmento para evitar consumir demasiada batería en el terminal, excepto en la gráfica que se representa a tiempo real.

Clase `GraphSectionFragment`

Fragmento de actividad que contiene todas las gráficas requeridas de la aplicación. La implementación de las gráficas se realizó usando la librería de gráficos *AndroidPlot*. Se estableció un tiempo de refresco de interfaz de 25 FPS en este fragmento para dar sensación de realismo a la representación de los datos.

Clase `OptionSectionFragment`

Fragmento de actividad que agrupa todas las opciones de la aplicación, así como la información de depuración de la misma. Para la información de depuración se estableció un tiempo de muestreo de 3 FPS.

5.6.2. Paquete `com.blueheart.services`

Clase `BluetoothService`

Clase que se encarga de gestionar el servicio conexión, envío y recepción de Bluetooth. Tiene cuatro estados posibles:

- `STATE_NONE.` Es el estado de inicialización e indica que el servicio no esta activo.
- `STATE_LISTEN.` Indica que el servicio esta a la espera de una conexión.
- `STATE_CONNECTING.` Indica que el servicio esta intentando realizar una conexión.
- `STATE_CONNECTED.` Indica que el servicio Bluetooth está conectado a un dispositivo.

Una vez que el servicio ha sido conectado puedo tener cuatro modos distintos que afectan a la recepción de datos:

- `MODE_IDLE`. Es el modo de inicialización e indica que servicio no esta a la espera de ningún dato.
- `MODE_DATA`. Indica que se espera recepción de datos reales.
- `MODE_DARK`. Indica que está recibiendo datos de calibración de ruido.
- `MODE_REFC`. Indica que está recibiendo datos de calibración de referencia.

Se pueden encontrar los siguientes comandos definidos en la clase:

- `EXIT_COMMAND`. Comando que solicita la desconexión del dispositivo. Es enviado automáticamente con la desconexión de la conexión.
- `START_COMMAND`. Comando que solicita la transmisión de datos del dispositivo.
- `STOP_COMMAND`. Comando que solicita la detención de la transmisión de los datos.
- `LED1_LOW_COMMAND`. Comando que establece la intensidad del LED 1 en baja (*low*).
- `LED1_MED_COMMAND`. Comando que establece la intensidad del LED 1 en media (*medium*).
- `LED1_HIGH_COMMAND`. Comando que establece la intensidad del LED 1 en alta (*high*).
- `LED1_OFF_COMMAND`. Comando que establece la intensidad del LED 1 en apagada (*off*).
- `LED2_LOW_COMMAND`. Comando que establece la intensidad del LED 2 en baja (*low*).
- `LED2_MED_COMMAND`. Comando que establece la intensidad del LED 2 en media (*medium*).
- `LED2_HIGH_COMMAND`. Comando que establece la intensidad del LED 2 en alta (*high*).

- `LED2_OFF_COMMAND`. Comando que establece la intensidad del LED 2 en apagada (*off*).
- `LED3_LOW_COMMAND`. Comando que establece la intensidad del LED 3 en baja (*low*).
- `LED3_MED_COMMAND`. Comando que establece la intensidad del LED 3 en media (*medium*).
- `LED3_HIGH_COMMAND`. Comando que establece la intensidad del LED 3 en alta (*high*).
- `LED3_OFF_COMMAND`. Comando que establece la intensidad del LED 3 en apagada (*off*).
- `LED4_LOW_COMMAND`. Comando que establece la intensidad del LED 4 en baja (*low*).
- `LED4_MED_COMMAND`. Comando que establece la intensidad del LED 4 en media (*medium*).
- `LED4_HIGH_COMMAND`. Comando que establece la intensidad del LED 4 en alta (*high*).
- `LED4_OFF_COMMAND`. Comando que establece la intensidad del LED 4 en apagada (*off*).

Dentro de clase `BluetoothService` existen otras dos clases: `ConnectThread`, que se encarga de realizar la conexión, y `ConnectedThread` que se encarga de gestionar la recepción de datos cuando el dispositivo ha sido conectado. Ambos son *threads* independientes.

Clase `DataPool`

Clase estática que se encarga de mantener en memoria los datos procesados útiles y de reutilizarlos cuando no sean necesarios. De esta forma se agiliza el proceso de recolección de basura de Android de manera dinámica.

Clase `DataService`

Clase estática que se encarga de todo el procesado de datos. Dentro de la misma se encuentra toda la configuración de datos de la aplicación.

- `WAVELENGTHS`. Define las longitudes de onda que se recibirán desde el dispositivo externo.

- `SATURATION_MEAN_SIZE`. Define el número de datos necesarios para calcular la media de la StO_2.
- `MOVING_AVERAGE_SIZE`. Define la longitud del algoritmo de media móvil aplicado.
- `WINDOW_SIZE`. Define el número de datos que forman la ventana de proceso de datos del algoritmo de filtrado de Savitzky–Golay.
- `POLYNOMIAL_ORDER`. Define el orden el polinomio del algoritmo de filtrado de Savitzky–Golay.
- `FRAME_WIDTH`. Define el tamaño del marco del algoritmo de filtrado de Savitzky–Golay.
- `PROCESSED_SIZE`. Define el número de datos procesados que se mantendrán simultáneamente en memoria.
- `CALIBRATION_SIZE`. Define el número de muestras necesarias para realizar la calibración.

5.6.3. Paquete `com.blueheart.structures`

Clase `Calibration`

Clase estructural que se encarga de definir cada muestra de calibración.

Clase `CircularList`

Clase estructural que hereda de `ArrayList` y permite definir un *array* circular simulando una cola FIFO (*First-In-First-Out*) de N elementos.

Clase `Data`

Clase estructural que define un paquete que ha sido procesado y debe ser almacenado. Dentro de cada objeto `Data` se encuentra la información del paquete procesado que es a su vez dividido en objectos `Wave` y almacenados en el objeto `Data`. Los atributos más relevantes de esta clase son:

- `mHemoglobin`. Contiene la información del cálculo de las concentraciones de hemoglobina.
- `mSaturation`. Contiene la información de la StO_2 en ese paquete de datos.

- `mHeartbeat`. Contiene la información del valor procesado.
- `mDerivative`. Contiene información de la pendiente del valor anterior.
- `mTime`. Almacena el tiempo en el que se recuperó el paquete.

Clase `Hemoglobin`

Clase estructural básica que define el objeto `Hemoglobin`.

Clase `IncrementalAveraging`

Clase estructural básica que se encarga de almacenar datos en una media incremental de manera dinámica.El cálculo de la media se realiza en cada inserción en vez de en cada consulta.

Clase `Packet`

Clase estructural básica que se encarga de almacenar la información del paquete sin procesar de los datos leídos por Bluetooth.

Clase `Wave`

Clase estructural básica que almacena la información de cada longitud de onda procesada. A la hora de la creación del objeto se realiza el cálculo de la densidad óptica de la longitud de onda correspondiente, por lo que es necesario tener datos de una calibración previa.

5.6.4. Paquete `com.blueheart.utils`

Clase `Algorithm`

Clase funcional estática que permite realizar operaciones de cálculo numérico. Las principales funciones utilizadas son:

- `average`. Calcula la media de un conjunto de elementos dados.
- `getBeatValue`. Calcula el valor de la frecuencia cardíaca en función de dos tiempos. El resultado se devuelve en PPM.
- `getMovingAverage`. Calcula la media móvil de un conjunto de elementos dados.
- `round`. Redondea el valor de manera eficiente.

Clase `MolarExtinctionCoefficient`

Clase instanciable única (*singleton*) que lee los valores de los coeficientes de extinción molar del archivo XML definido en la aplicación.

Clase `Preference`

Clase funcional estática que permite el acceso rápido a la configuración de la aplicación del terminal Android. Permite operaciones tanto de lectura como de escritura en las *SharedPreferences* de la aplicación.

5.6.5. Paquete `com.blueheart.views`

Clase `RingView`

Clase que implementa el *widget* presentado en la pantalla principal de la aplicación responsable de indicar el porcentaje de StO_2 del usuario.

5.7. Modulos y librerias

Algunas implementaciones de la aplicación se optaron por tomarlas de librerías debido al tiempo del que se disponía. A continuación se detallan cuales son dichas librerias y sus principales funcionalidades.

Libreria *AndroidPlot*

AndroidPlot [37] es una API para crear gráficos estáticos y dinámicos en aplicaciones Android. Tiene licencia Apache 2.0 y ha sido diseñado desde sus inicios para terminales Android y es compatible con todas sus versiones desde la 1.6. Actualmente se utiliza en más de 500 aplicaciones publicadas en el *Play Store*.

AndroidPlot soporta distintos tipos de gráficos:

- Gráficos de lineas.
- Gráficos de barras.
- Gráficos circulares.
- Gráficos de puntos.
- Gráficos de secuencias.

La librería *AndroidPlot* se utiliza en el proyecto para representar la sección de gráficas de la aplicación. Se ha empleado dicha librería sobre otras similares por su simpleza, eficiencia y facilidad en representar gráficas en tiempos real.

Libreria *JAMA*

JAMA [38] es una librería de álgebra básica linear programada en Java. Proporciona clases a nivel de usuario para construir y manipular grandes matrices de datos. El paquete *JAMA* se distribuye sin licencia y se compone de una serie de clases que permiten la manipulación de objetos, operaciones elementales, descomposiciones matriciales, solucionador de ecuaciones y de derivadas. Las descomposiciones incluidas son:

- Descomposición simétrica de Cholesky .
- Descomposición LU de matrices rectangulares.
- Descomposición QR de matrices rectangulares.
- Descomposición de matrices cuadradas simétricas y asimétricas.
- Descomposición de valores singulares en matrices rectangulares.

La librería *JAMA* se utiliza para realizar el cálculo del algoritmo de filtro de Savitzky–Golay. Se ha utilizado esta librería sobre otras debido a que se utilizaba en la implementación documentada en la subsección 5.3.2.

5.8. Firma de la aplicación

Para que una aplicación pueda ser instalada en un dispositivo Android es necesario que haya sido firmada con un certificado cuya clave privada sea propiedad del desarrollador de la aplicación. El certificado sirve como medio para identificar al creador de la aplicación, así como otra información de la misma, y para establecer relaciones de confianza entre aplicaciones. Se deben tener en cuenta los siguientes puntos a la hora de firmas una aplicación Android:

- Android no permite la instalación de aplicaciones no firmadas, por lo que la firma de la aplicación es necesaria.
- Android permite firmar aplicaciones con certificados autofirmados.

- Android solo verifica la validez del certificado en el momento de la instalación de la aplicación, por lo que si el certificado expira una vez instalada la aplicación, no afectará a su funcionamiento.
- Se pueden utilizar herramientas como *Keytool* o *Jarsinger* para firmar aplicaciones una vez empaquetadas en un archivo con la extensión `.apk`.

La firma de aplicaciones Android evita que desarrolladores maliciosos modifiquen la misma y la publiquen como legítima, de manera que aumenta la seguridad de las aplicaciones disponibles al público. Un desarrollador malicioso no puede publicar una aplicación modificada en *Google Play*. Si se diera el caso, la aplicación estaría firmada con la clave privada del desarrollador malicioso, de manera que sería reconocido como el autor de la aplicación modificada, eximiendo de cualquier problema al desarrollador legítimo de la aplicación.

Una vez firmado el archivo `.apk` de la aplicación, Google recomienda utilizar la herramienta *Zipalign* sobre el mismo. *Zipalign* es una herramienta que viene incluida en el SDK de Android desde la versión 1.6 y que está pensada para optimizar los paquetes `.apk` adaptándose a los requisitos óptimos del sistema Android.

En Android, los datos almacenados dentro de archivos `.apk` son requeridos por multitud de procesos: el instalador leerá el manifiesto para manejar los permisos asociados con cada solicitud; la aplicación Inicio leerá los recursos para obtener el nombre de la aplicación y el icono; el servidor del sistema leerá los recursos para, por ejemplo, mostrar notificaciones; y, por supuesto, los archivos de recursos son obviamente utilizados por la propia aplicación.

Zipalign garantiza que todos los datos sin comprimir empiezan con una particular alineación de bytes respecto al comienzo del archivo. Establecer una alineación de 4 bytes proporciona una optimización de rendimiento cuando se instala en un dispositivo Android. Cuando están alineados, el sistema es capaz de leer archivos con `mmap()`, incluso si contienen datos binarios con restricciones de alineamiento, en vez de copiar todos los datos del paquete en el caso de no estar alineados.

Si una aplicación no está optimizada con *Zipalign* la lectura de los recursos de aplicaciones será lento y requerirá de mucha memoria. En el mejor de los casos, el único resultado visible es que tanto la aplicación principal como

el inicio de la aplicación será más lenta de lo que deberían. En el peor de los casos, la instalación de varias aplicaciones no alineadas aumentará los requisitos de memoria, provocando que el sistema se sobrecargue por tener que iniciar y terminar estos procesos. En estos casos el usuario terminará con un dispositivo lento y con un consumo de batería excesivo.

6

Diseño e implementación del pulsioxímetro

El diseño del pulsioxímetro no entra dentro del proyecto como tal, sin embargo, al ser una parte necesaria para el funcionamiento del sistema se va a detallar brevemente a continuación.

6.1. Fuente de emisión electromagnética

Para la construcción del prototipo desarrollado se ha elegido la tecnología LED para emitir luz en el visible e infrarrojo cercano. Las razones de la elección de dicha tecnología son de índole técnica y económica.

- Los LED son más baratos que otras fuentes de luz, ya que se hace innecesario el uso de filtros interferenciales que son muy delicados y extremadamente caros.
- La electrónica permite el encendido y apagado rápido sin necesidad de elementos mecánicos complejos.
- Debido a su bajo consumo, permiten ser alimentados con baterías de bajo peso, por ejemplo, una pila de botón.
- La tecnología SMD permite integrarlos en una superficie muy pequeña y ligera.

Se han elegido cuatro longitudes de onda para determinar el pulso y el estado de oxigenación de la hemoglobina. Los cambios debidos al pulso se detectan mejor a longitudes de onda más cortas.

Un LED es un diodo que permite el paso de la corriente en un solo sentido. El dopaje del silicio producido en la fabricación del diodo se hace con cantidades mínimas de impurezas, tales como aluminio, galio o indio. Los diodos comunes de silicio emiten energía cuando la corriente pasa en el sentido positivo. Esta energía está alrededor de los 1100 nm.

El sensor está diseñado para minimizar la pérdida de esta energía, sin embargo, dopando el semiconductor con mezclas de impurezas cuidadosamente escogidas, es posible producir diodos que emitan energía a otras longitudes de onda como las elegidas para el prototipo realizado en este proyecto. Existe una gran gama de longitudes de onda disponibles en el mercado, sin embargo, únicamente se han utilizado cuatro de ellas.

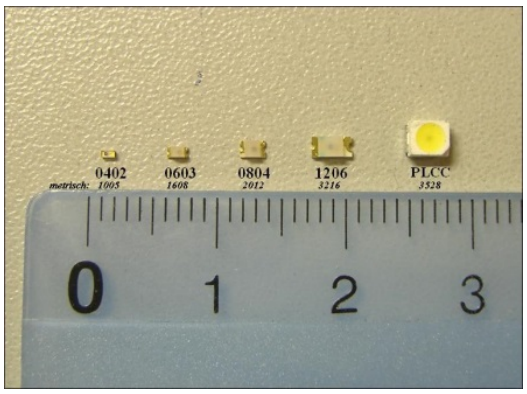

Figura 6.1: LED de diferentes tamaños disponibles

En la figura 6.1 se pueden observar los tamaños relativos de los diferentes LED SMD. Un LED SMD es un LED encapsulado en una resina semirrígida que se ensambla de manera superficial. Esto le ofrece ciertas características muy interesantes para la aplicación.

Las longitudes de onda utilizadas para la cuantificación del porcentaje de saturación de la hemoglobina son 640 y 850. La figura 6.2 muestra la medida obtenida de la respuesta espectral de estos LED. Ambos están localizados entre el punto isosbéstico de la oxi y deoxihemoglobina en la región del infrarrojo cercano a unos 815 nm.

La figura 6.3 muestra la localización espectral de estos LED respecto a las bandas de absorción de la hemoglobina y el agua.

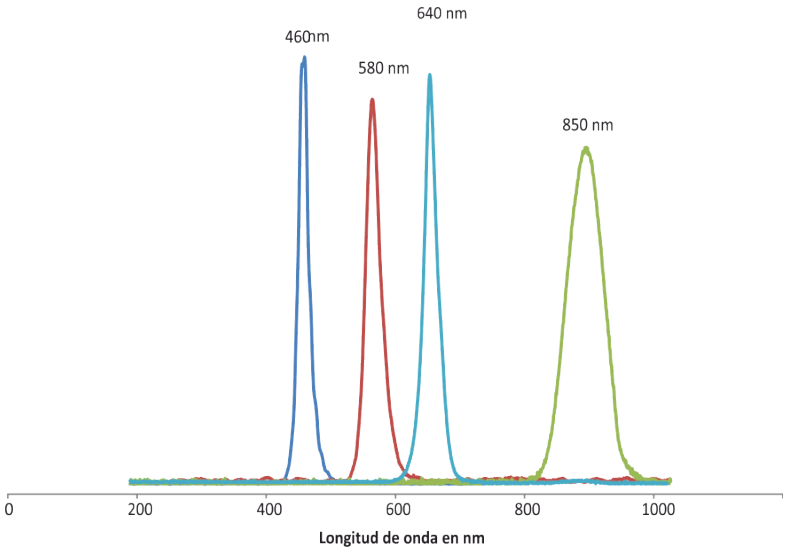

Figura 6.2: Longitudes de onda leídas con el prototipo

6.2. Sensor de detección

Uno de los elementos más importantes del diseño, es la elección del sensor utilizado. Este trabaja utilizando cuatro LED y un fotodetector, que trabajan a longitudes de onda dentro de lo requerido.

Un solo fotodetector es usado para detectar la energía alternante proveniente de los cuatro LED. En la transmisión convencional, éste es posicionado en forma perpendicular y opuesta a los LED con la extremidad puesta cómodamente entre ellos. Este montaje debe ser protegido de luces extrañas con longitudes de onda dentro del rango al cual es sensible el fotodetector.

En el momento de montaje, el fotodetector es situado a una pequeña distancia y en el mismo plano que los LED. El fotodetector es usualmente un fotodiodo de silicio. Estos tienen la ventaja de que no alteran sus propiedades eléctricas cuando son expuestos a energía externa.

La fotosensibilidad está usualmente limitada a un ancho de banda. Esto permite que la selección del dispositivo sea limitada también. El fotodiodo de silicio tiene un gran rango dinámico, con un cambio lineal en su salida proporcional al nivel de la luz incidente en un rango de 10 décadas de energía. Los fototransistores son más sensibles que los fotodiodos pero tienen la desventaja de ser eléctricamente más ruidosos. Además, la sensibilidad de todos los foto detectores varía con la longitud de onda, y es necesario tenerlo en

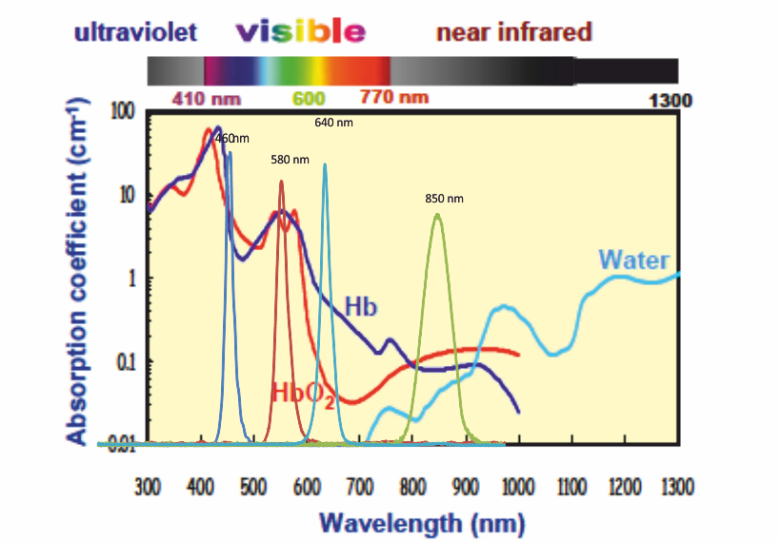

Figura 6.3: Longitudes de onda elegidas con respecto a las bandas de absorción de la hemoglobina y el agua

cuenta en el diseño electrónico y en la calibración de dispositivo. Por estas razones se ha elegido un fotodiodo de propósito general que además de ser muy económico cumple con las especificaciones requeridas para un pulsioxímetro de estas características.

6.3. Diseño del hardware

Teniendo en cuenta que el sensor que se va a utilizar consta de cuatro LED y un fotodiodo, y que las señales correspondientes a cada LED se deben observar independientemente para realizar el cálculo del porcentaje de saturación de oxígeno en el tejido biológico, se debe realizar un diseño que permita la conmutación en el encendido y apagado de los LED, de tal forma que el fotodiodo nunca reciba luz de dos o más longitudes de onda al mismo tiempo. El esquema de la señal de control para el encendido de los LED se encuentra en la figura 6.4.

En éste caso, se utiliza una frecuencia de muestreo de 1 kHz para cada señal, la cual es mucho mayor que las frecuencias que generalmente se encuentran en el pulso arterial.

Esta frecuencia de muestreo se eligió para permitir reconstruir cada onda de pulso independientemente sin perder información en el acto. También se

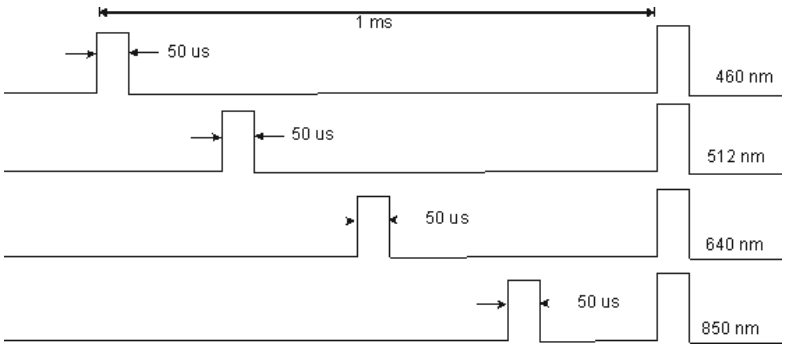

Figura 6.4: Señal de control para encendido secuencial de los cuatro LED

debe utilizar un ancho de pulso que sea lo suficientemente grande como para obtener una respuesta coherente en el fotodiodo, y lo suficientemente pequeño como para permitir el registro de datos, evitando la interferencia entre las lecturas.

6.3.1. Sistema de captación de señales biológicas

El sistema de captación de señales biológicas es el subsistema de los LED y el fotodiodo con la electrónica necesaria para alimentarlo y generar la luz de los LED y las lecturas de la señal procedente del tejido mediante el fotodiodo así como su acondicionamiento y conversión por el ADC.

Estos circuitos se compone de varias partes importantes. Una de ellas es la que permite obtener la energía suficiente para alimentar los LED, y la otra nos permite controlar el encendido y apagado de los mismos en el proceso de conmutación.

La primera parte se ha implementado utilizando una fuente de tensión como la representada en la figura 6.5.

El sistema de alimentación tiene dos voltajes, 12 y 5 voltios, para alimentar los LED y los circuitos electrónicos. Está basado en un conversor DC-DC de bajo ruido modelo *NMH0512SC*.

Los LED se alimentan a 5 voltios y las resistencias variables permiten ajustar los valores de tensión para cada uno de ellos. Cada LED tiene unos valores óptimos de corriente y voltaje que deben ser pre-ajustados para cada dispositivo.

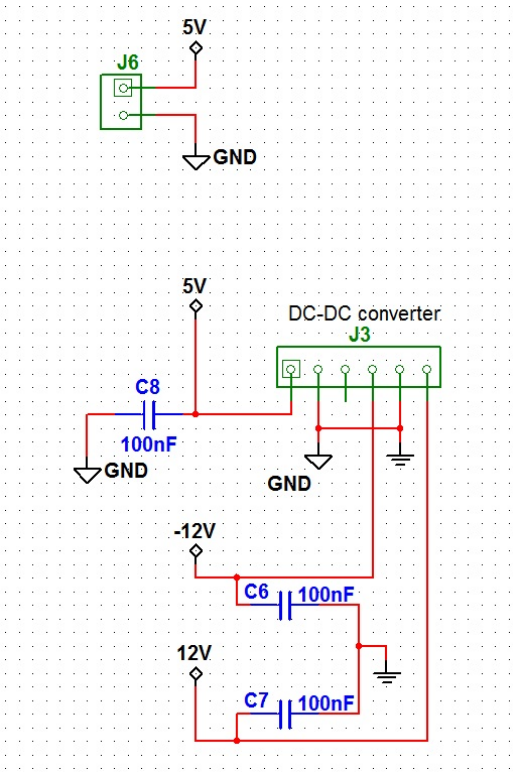

Figura 6.5: Diseño de la fuente de tensión

El circuito integrado *ADG411BR* de *Analog Devices Inc.* es un interruptor cuádruple que permite encender cada LED de forma independiente y en rangos temporales de nanosegundos.

6.3.2. Convertidor de corriente a voltaje

El fotodiodo que se encarga de recibir la luz emitida por los LED, cuando opera en modo foto conductivo se polariza en inversa, de tal manera que, cuando la luz cae en la unión, se producen pares electrón-hueco que debido a la polarización provocan la aparición de una corriente en inverso del diodo. Dicha corriente es proporcional a la cantidad de luz que llega y su valor generalmente es muy pequeño. Para obtener la corriente y convertirla en una señal de voltaje que pueda ser manejada de una mejor manera, se emplea el circuito representado en la figura 6.7.

El esquema de la figura 6.7 corresponde a un amplificador de tras-impedancia y cuya salida, al tener una tierra virtual en la entrada inversora del amplificador, corresponde al producto de la corriente inducida en el fotodiodo por

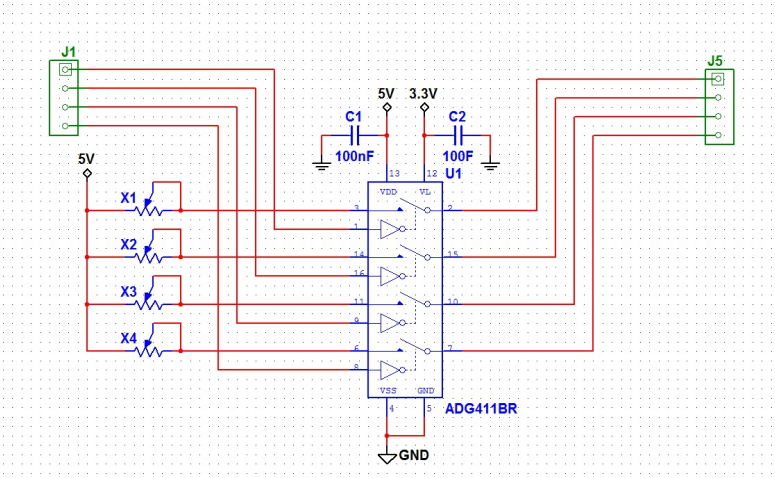

Figura 6.6: Diseño del conversor DC-DC

la resistencia de realimentación del circuito. Para que el valor de voltaje obtenido esté dentro de un rango manejable, la resistencia de realimentación debe tener un valor alto que compense las pequeñas corrientes a convertir.

En éste caso, debido a la alta resistencia que se encuentra en la unión del fotodiodo, se debe utilizar un amplificador operacional de alta precisión de *Texas Instruments OPA277UA*, que nos permita acoplarlo correctamente. Las características de este operacional en cuanto al bajo ruido y alta velocidad así como un ultra bajo voltaje de offset y drift.

6.3.3. Diseño del dispositivo final

Se utilizó un dispositivo PIC para la adquisición de datos y el control de LED. Se eligió una placa Mini-32 Board de *MikroElektronika* [39] por su pequeño tamaño, bajo costo (29$), reducido consumo y gran capacidad de proceso (tiene dos osciladores de 32.768KHz y 8MHz y un *PIC32MX534f064H*).

Se utilizó un dispositivo Bluetooth basado en el integrado RN-41 para transferir los datos al terminal Android. El consumo de este componente es de 30mA haciéndolo idóneo para aplicaciones móviles. El alcance de la radio bluetooth es de 100 m, suficiente para la aplicación que se propone. Su elección de basó en el bajo precio (42$) y reducido consumo.

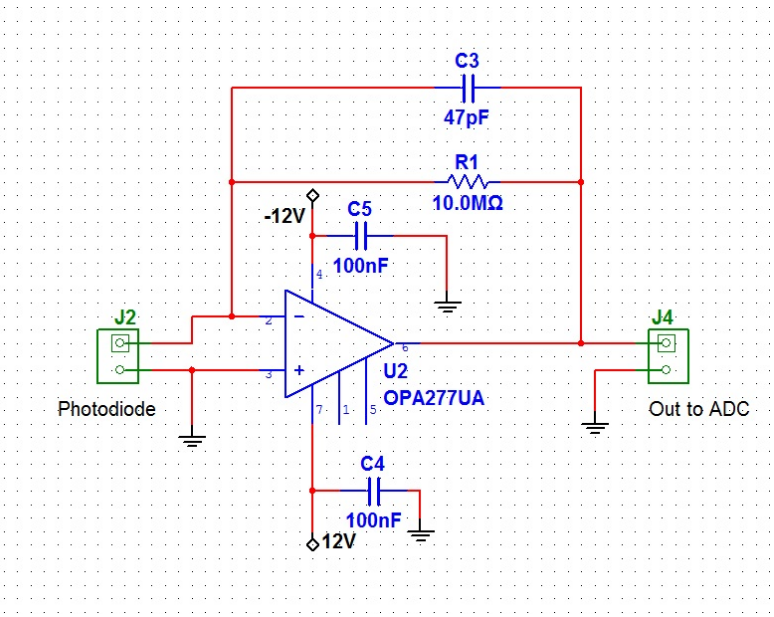

Figura 6.7: Esquema del circuito

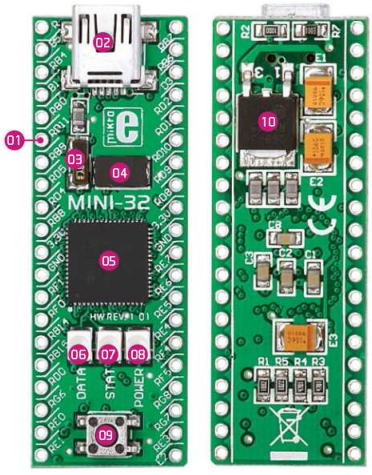

Figura 6.8: Módulo de procesado y adquisición de datos

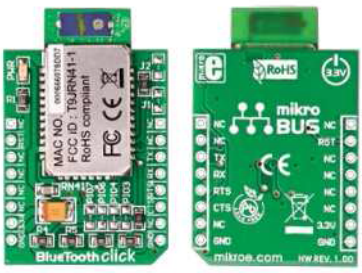

Figura 6.9: Módulo Bluetooth

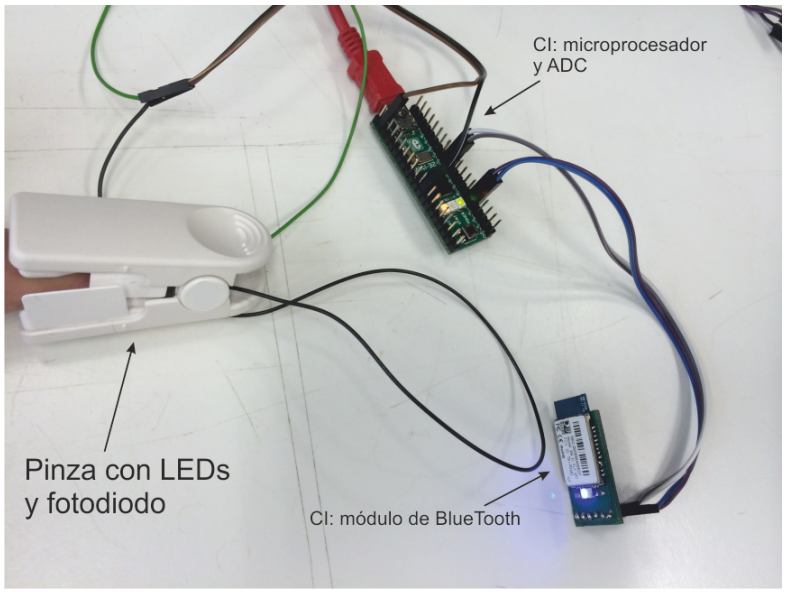

Figura 6.10: Prototipo del pulsioxímetro integrado con el sensor y los elementos de captura, control y transmisión

7

Diseño e implementación del servidor

El diseño y la implementación del pulsioxímetro se realizó paralelamente con la aplicación. Debido a esto y, sobretodo, a problemas de disponibilidad del dispositivo para realizar pruebas, se optó por hacer un software de escritorio que permitiera simular con exactitud los comandos, datos y ordenes del dispositivo original.

Para ello se adquirió un adaptador Bluetooth USB y se ubicó en el en el equipo de desarrollo. A continuación se realizará una breve explicación del diagrama y la definición de las clases que se han usado para realizar la implementación de dicho servidor.

7.1. Funcionamiento

Emulando el comportamiento del pulsioxímetro, el servidor debe leer un conjunto de datos, esperar una conexión Bluetooth e interactuar con los clientes que se conecten al mismo a través de Bluetooth.

La lectura de datos se realiza en la inicialización de la aplicación y consiste en la lectura de archivos externos que han sido escritos por un espectrofotómetro homologado. Los datos leídos son almacenados en estructuras en memoria RAM.

Una vez que se dispone de los datos en el servidor, se crea una conexión Bluetooth en escucha de clientes. Cada conexión se gestiona en un hilo diferente, lo que permite conexiones múltiples de clientes.

Cada cliente dispone de dos hilos o *threads* independientes: el *thread* de entrada y el *thread* de salida. El *thread* de entrada se encarga de recibir todos los datos que envía el cliente, es decir, los comandos de ejecución. El *thread* de salida se encarga de emitir datos de manera continua hasta que se recibe un comando de detención.

Al disponer de una cantidad de datos limitada, cuando el *thread* de salida detecta que ha emitido todos los datos leídos se comienza desde el principio la emisión de los mismo, simulando, de este modo, un *stream* constante de valores. Los comandos de modificación de intensidad de los LED no tienen ninguna repercusión en los datos enviados por el servidor.

Para realizar la calibración, el servidor espera el comando apagado de la intensidad de los LED y se cambia a modo de envío de datos de ruido. Una vez que recibe el comando de detener el flujo de datos, el servidor cambia a modo de envío de valores de referencia. Una vez que vuelve a recibir el comando de detención, se establece el modo normal de envío de datos hasta que se vuelve a recibir un comando de intensidad nula de los LED.

7.2. Diagrama de clases

A continuación, en la figura 7.1, se muestra el diagrama de paquetes usados en la aplicación de servidor y sus relaciones.

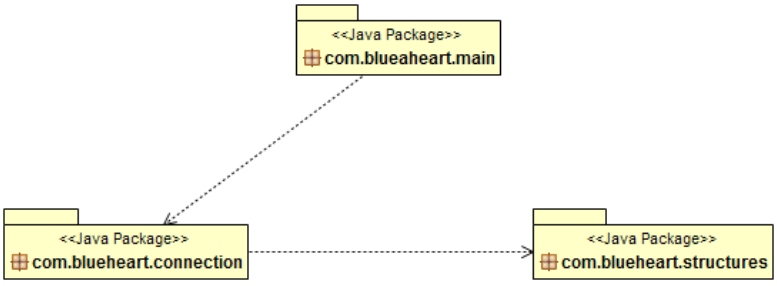

Figura 7.1: Diagrama de paquetes del servidor

- `com.blueheart.main`. Contiene las clases principales de la aplicación, incluyendo la clase de inicialización que contiene el método `main`.
- `com.blueheart.connection`. Contiene todas las clases que se encargan de gestionar las conexiones recibidas por Bluetooth.
- `com.blueheart.estructures`. Contienen las clases de estructuras en las que se almacenan los datos leídos.

7.3. Definición de clases

En la figura 7.2 se muestra el diagrama de clases y sus relaciones. A continuación se detallan las funciones de dichas clases y sus métodos y atributos mas relevantes.

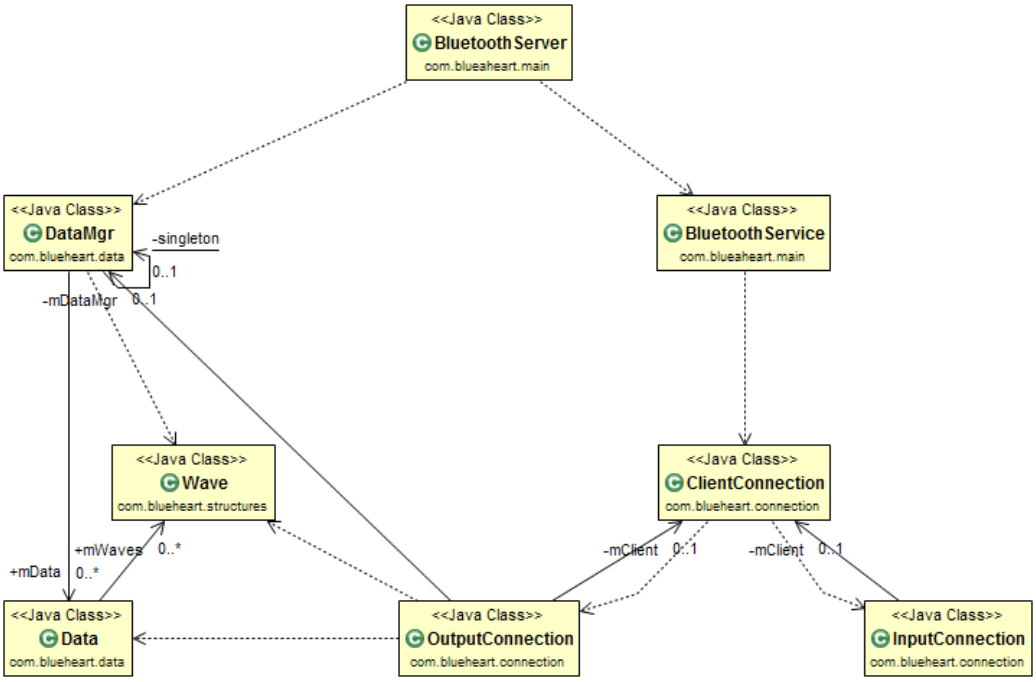

Figura 7.2: Diagrama de clases del servidor

7.3.1. Paquete `com.blueheart.main`

Clase `BluetoothServer`

`BluetoothServer` es la clase principal de la aplicación que contiene el método `main`, que se ejecuta al iniciar la aplicación. En dicho método se llama al servicio de lectura de datos y se crea un thread con el servicio de escucha de Bluetooth.

Clase `BluetoothService`

La clase `BluetoothService` se encarga de gestionar el adaptador Bluetooth USB y ponerlo en modo de escucha de conexiones. Durante la inicialización establece el dispositivo Bluetooth en modo visible. Por cada cliente conectado se crea un *thread* de clase `ClientConnection`.

Clase `DataMgr`

La clase `DataMgr` lee los datos de los archivos de resultado del espectrofotómetro y los almacena en estructuras en memoria RAM. El servidor lee datos de cuatros longitudes de onda y sus respectivos valores de ruido y referencia.

7.3.2. Paquete `com.blueheart.connection`

Clase `ClientConnection`

La clase `ClientConnection` es la clase que gestiona cada conexión individual del dispositivo Bluetooth. En ella se definen todos los comandos, modos y estados de cada cliente. Estas definiciones son las mismas que las presentadas en la sección 5.6.2.

Clase `InputConnection`

La clase `InputConnection` se encarga de gestionar la entrada de datos en el servidor. Según el comando recibido opera de distintas formas. La clase se queda en escucha de comandos hasta que se recibe el comando `EXIT_COMMAND` o se recibe una desconexión del cliente.

- `EXIT_COMMAND`. Detiene todos los threads y cierra la conexión con el cliente.
- `START_COMMAND`. Cambia el modo del cliente a `MODE_DATA`, `MODE_DARK` o `MODE_REFC` según se haya recibido o no distintos comandos de cambio en la intensidad de los LED. La explicación del funcionamiento completo de este comando se puede encontrar en la sección 7.1.
- `STOP_COMMAND`. Detiene el envío del flujo de datos.

Clase `OutputConnection`

La clase `OutputConnection` se encarga de realizar el envío de datos. Dependiendo del modo de la aplicación puede enviar datos de valores, de ruido, de referencia o no enviar ninguno. La clase se encuentra en funcionamiento hasta que se recibe el comando `EXIT_COMMAND` o se recibe una desconexión del cliente.

Es posible variar el tiempo de espera entre paquetes mediante la variable `WAIT_TIME`. Se ha establecido un tiempo por defecto de 20 ms. Una vez que ha enviado un paquete de datos, se queda en espera hasta que se vence el tiempo de espera establecido en la variable anterior. En la ecuación 7.1 se define la formula del tiempo que debe esperar la aplicación entre el envío de cada paquete, siendo t_0 el tiempo de procesamiento de datos.

$$t = \texttt{WAIT\_TIME} - t_0 \tag{7.1}$$

7.3.3. Paquete `com.blueheart.structures`

Clase `Data`

La clase `Data` es una clase estructural que almacena datos de cada una de las longitudes de onda leidas en una lista de estructura `Wave`.

Clase `Wave`

La clase `Wave` almacena y relaciona los datos del valor leído con su correspondiente información de calibración.

7.4. Librerias y módulos

A la hora de implementar el servidor solo se ha utilizado la librería *BlueCove*. *BlueCove* [40] es una librería de Java de Bluetooth con la implementación JSR-82 que soporta tanto Windows como Mac OS como sistemas operativos. La librería se distribuye bajo licencia Apache 2.0. Esta librería se ha utilizado para interactuar con el adaptador Bluetooth USB.

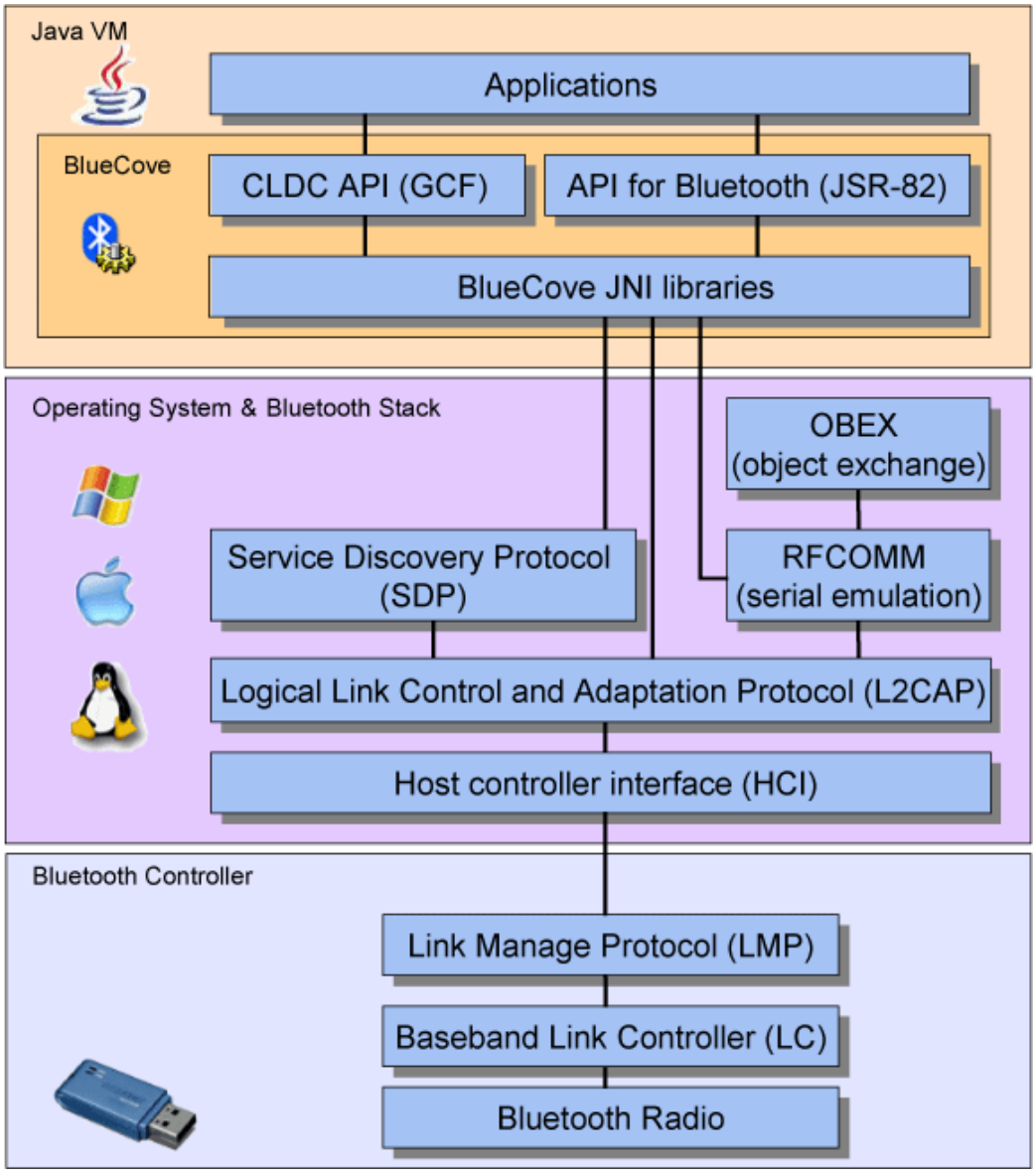

Figura 7.3: Diagrama de capas de *BlueCove*

8

Diseño y resultado de las pruebas

En este capítulo se detallará las realización de las pruebas finales con el sistema integrado y se compararán los resultados con los obtenidos con un equipo médico homologado.

8.1. Realización

Para la realización de las pruebas se han utilizado los datos obtenidos de cuatros usuarios. Cada uno de ellos ha realizado la prueba utilizando simultáneamente, durante un periodo breve de tiempo, el sistema presentado en este documento y un pulsioxímetro médico homologado.

Los datos obtenidos del equipo médico han sido representados en distintas gráficas utilizando un software diseñado para ese fin. Los datos recibidos del dispositivo externo del sistema planteado en este documento han sido almacenados en archivos de datos para poder realizar la comparación en cualquier momento.

8.2. Resultados obtenidos

A continuación se van a presentar los resultados obtenidos tanto por el equipo médico, como por la aplicación. Por último, se planteará una comparativa y se realizará un resumen de lo obtenido.

8.2.1. Equipo médico

Los resultados obtenidos por el equipo médicos de los cuatros usuarios de prueba son los siguientes:

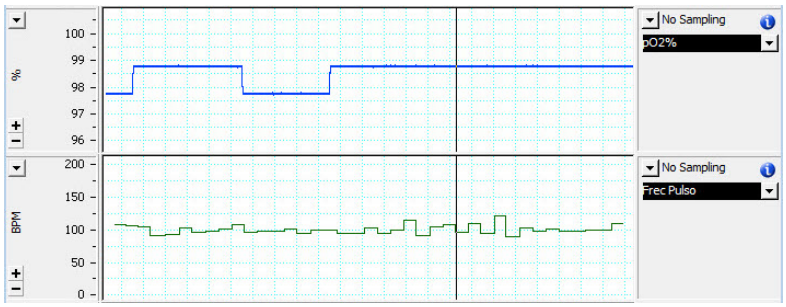

Figura 8.1: Resultados usuario 1 en equipo médico

El usuario 1 presenta un porcentaje de StO_2 entre 97,5 % y 99 % y una frecuencia cardíaca entre 90 y 125 BPM.

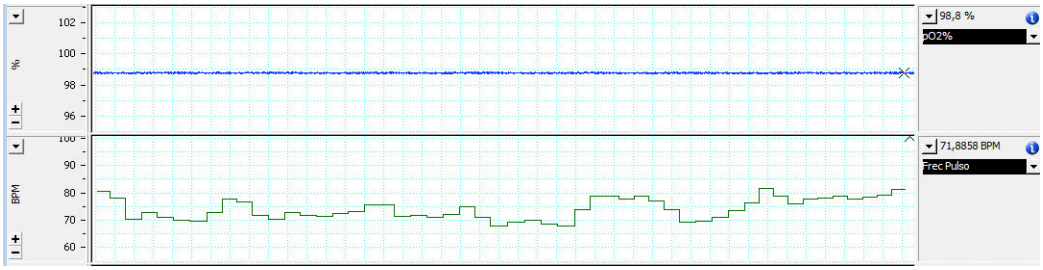

Figura 8.2: Resultados usuario 2 en equipo médico

El usuario 2 presenta un porcentaje de StO_2 del 99 % y una frecuencia cardíaca entre 70 y 80 BPM.

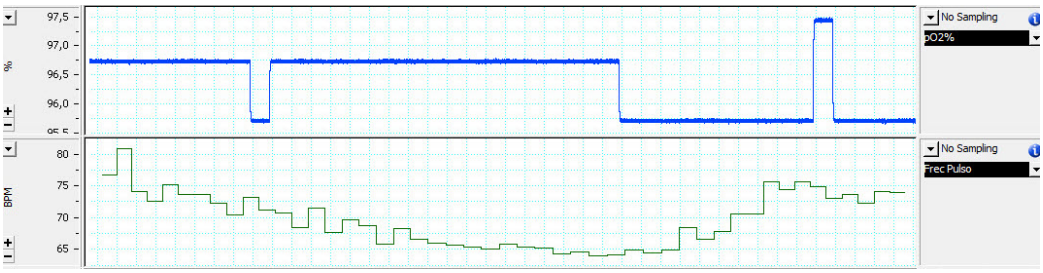

Figura 8.3: Resultados usuario 3 en equipo médico

El usuario 3 presenta un porcentaje de StO_2 entre 96 % y 97 % y una frecuencia cardíaca entre 65 y 80 BPM.

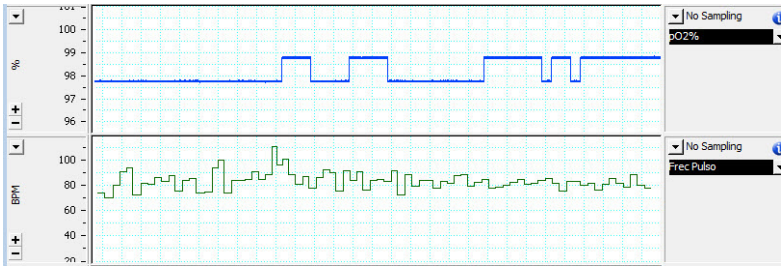

Figura 8.4: Resultados usuario 4 en equipo médico

El usuario 4 presenta un porcentaje de StO_2 entre 97,5 % y 99 % y una frecuencia cardíaca entre 75 y 87,5 BPM.

8.2.2. Sistema desarrollado

Los resultados obtenidos a través del sistema desarrollado de los cuatros usuarios de prueba se pueden observar en la figura 8.5 y en la figura 8.6.

Figura 8.5: Resultados usuario 1 y 2 en sistema desarrollado

El usuario 1 presenta un porcentaje de StO_2 entre 90,8 % y 91,4 % y una frecuencia cardíaca entre 75,2 y 127,5. El usuario 2 presenta un porcentaje de StO_2 entre 92,1 % y 92,7 % y una frecuencia cardíaca entre 65,9 y 87,5.

Figura 8.6: Resultados usuario 3 y 4 en sistema desarrollado

El usuario 3 presenta un porcentaje de StO_2 entre 90,9 % y 91,5 % y una frecuencia cardíaca entre 62,3 y 84,6. El usuario 4 presenta un porcentaje de StO_2 entre 91,2 % y 91,5 % y una frecuencia cardíaca entre 69,7 y 92,5.

8.2.3. Comparación de resultados

Los valores de la StO_2 obtenidos en el sistema desarrollado tienen un desplazamiento de un 7 % en comparación con los obtenidos en el equipo médico, sin embargo, la variación relativa en los cuatro usuarios es prácticamente la misma.

Los valores de la frecuencia cardíaca obtenidos en el sistema desarrollado son prácticamente iguales a los obtenidos en el equipo médico, sin embargo, existen algunos picos de detección que se corregirán al aplicar la media sobre los valores obtenidos.

Comparando los valores máximos y mínimos de los resultados obtenidos en los dos entornos se obtiene la tabla 8.1.

La precisión es la medida que es capaz de apreciar un instrumento. Está relacionada con la sensibilidad, que es la variación de la magnitud a medir que es capaz de apreciar un instrumento. A mayor sensibilidad, menores variaciones es capaz de apreciar, por lo que el instrumento dará medidas mas exactas.

	StO_2	**Frecuencia Cardíaca**
Usuario 1		
Equipo médico	97,50 % - 99,00 %	90,00 - 125,00 BPM
Sistema desarrollado	90,80 % - 91,40 %	75,20 - 127,50 BPM
Usuario 2		
Equipo médico	99,00 %	70,00 - 80,00 BPM
Sistema desarrollado	92,10 % - 92,70 %	65,90 - 87,50 BPM
Usuario 3		
Equipo médico	96,00 % - 97,00 %	65,00 - 80,00 BPM
Sistema desarrollado	90,90 % - 91,50 %	62,30 - 84,60 BPM
Usuario 4		
Equipo médico	97,50 % - 99,00 %	75,00 - 87,50 BPM
Sistema desarrollado	91,20 % - 91,50 %	69,70 - 92,50 BPM

Tabla 8.1: Tabla comparativa de resultados

Para calcular la precisión de un instrumento es necesario calcular previamente el error absoluto y el error relativo. El error absoluto es la diferencia entre el valor de medida y el valor tomado como exacto. En este caso se tomarán como valores exactos los resultantes del dispositivo médico. El error relativo es el porcentaje del cociente entre el error absoluto y el valor exacto.

$$E_{\text{absoluto}} = V_{\text{medida}} - V_{\text{exacto}} \tag{8.1}$$

$$E_{\text{relativo}} = \frac{E_{\text{absoluto}}}{V_{\text{exacto}}} 100 \tag{8.2}$$

$$\text{Precisión} = 100 - |E_{\text{relativo}}| \tag{8.3}$$

A continuación, se calcula el porcentaje de precisión que se ha obtenido con los resultados. A los valores de la StO_2 se les ha sumado el desplazamiento antes de realizar el cálculo.

	StO_2	**F. Cardíaca**	**Total**
Usuario 1	99,69 % - 99,39 %	83,56 % - 98,00 %	95,16 %
Usuario 2	99,60 % - 99,29 %	94,14 % - 90,63 %	95,99 %
Usuario 3	98,02 % - 98,45 %	95,85 % - 94,25 %	96,64 %
Usuario 4	99,28 % - 99,49 %	92,93 % - 94,29 %	96,50 %
Total	**99,19 %**	**92,95 %**	**96,07 %**

Tabla 8.2: Tabla comparativa de precisión

Se ha obtenido una precisión del 96,07 %, lo que indica que los resultados del a aplicación se aproximan mucho a los reales medidos con un dispositivo médico homologado. La desviación resultante de la StO_2 es debida a la elección de las longitudes de onda al realizar el diseño del sistema. Sin embargo, se podría minimizar este error ajustando a otras longitudes de onda distintas.

Parte IV

Presupuesto y conclusiones

9

Presupuesto

En este capítulo se calculará el presupuesto del proyecto, incluyendo el desglose en distintos tipos de gastos, tanto económicos como de tiempo. Todos los cálculos se efectúan con sus valores antes de impuestos y se expresan en euros.

9.1. Gastos directos

Los gastos directos son aquellos que son imputables directamente al proyecto. Esta categoría engloba los gastos de personal, la amortización de equipos y el material fungible utilizado durante el desarrollo del proyecto.

9.1.1. Gastos de personal

Al tratarse de un desarrollo informático, el gasto directo proviene mayoritariamente del coste del personal. Para realizar este cálculo, se estima el tiempo ocupado por cada fase del proyecto, representado en el diagrama de Gantt (ver apéndice C). Los tiempos están calculados para una sola persona con una dedicación parcial alrededor de 30 horas semanales, lo que supondría el 75 % de una jornada completa.

Se ha tomado el salario medio correspondiente a un analista-programador según Infojobs Trends [41], que es de aproximadamente unos 23.251 € brutos anuales. El coste final se calcula multiplicando el coste anual por la duración del proyecto, de unas 28 semanas, y la dedicación del personal al proyecto descrita anteriormente.

$$C_{personal} = 23251€\frac{28}{52}\frac{75}{100} = 9389€ \tag{9.1}$$

9.1.2. Gastos de hardware

Se incluyen otros gastos directos, además de los de personal, como la amortización de equipos informáticos y dispositivos hardware utilizados. Para desarrollar el proyecto ha sido necesario un equipo de desarrollo y dos terminales Android con Bluetooth para realizar las pruebas de implementación.

Equipo de desarrollo

El equipo de desarrollo, que realiza las funciones de servidor de pruebas de la aplicación, cuenta con:

- **Placa base** Intel Socket 1150 - Asus B85M-G. 69,00 €.
- **Cables de red** 5/15 m. 17,45 €.
- **Torre** ATX Cooler Master CMP-350 500W. 63,00 €.
- **Teclado con cable** Gigabyte KM6150 Kit Teclado y **Ratón USB**. 12,95 €.
- **Procesador** Intel Socket 1150 Core I5-4570 3.2Ghz Box Socket 1150. 162,00 €.
- **Memoria RAM** DDR3 1600 Kingston HyperX Blu DDR3 1600 PC3-12800 8GB 2x4GB CL9. 78,00 €.
- **Monitor** Samsung S24B420BW 24"LED. 132,00 €.
- **Adaptador Bluetooth** USB Hama Nano Class 2. 12,95 €.

El precio final del equipo montado son 598,30 €.

Terminales Android

Se han utilizado dos terminales Android durante el desarrollo de la aplicación: Un Samsung Galaxy S4 de coste 429,00 € y un Samsung Nexus S 232,00 €.

Coste final

El periodo de amortización de los equipos empleados es de 3 años, por lo que el coste imputable, teniendo en cuenta que se utiliza en varios proyectos con dedicación igual a la del personal, la fórmula de cálculo del coste imputable al proyecto será:

$$C_{hardware} = \frac{C_{\text{hardware sin IVA}} T_{\text{dedicado al proyecto}}}{T_{\text{amortización}}} \frac{75}{100} \tag{9.2}$$

Equipo	Coste	Dedicación	Amortización	Coste imputable
Equipo de desarrollo	598,30 €	28 semanas	36 meses	80,54 €
Samsung Galaxy S4	429,00 €	11 semanas	36 meses	22,68 €
Samsung Nexus S	232,00 €	11 semanas	36 meses	12,27 €
TOTAL				**115,49 €**

Tabla 9.1: Gastos de hardware

9.1.3. Gastos software

El equipo en el que se ha desarrollado el proyecto utiliza un sistema operativo *Windows 7 Professional* adquirido con una licencia gratuita para estudiantes a través de la universidad.

La documentación del proyecto ha sido realizada con el sistema de composición de textos LaTeX, y empleando software de licencia libre, como el entorno de desarrollo (*Netbeans*). Es decir, no se ha incurrido en ningún gasto de software.

9.1.4. Dispositivo externo

El pulsioxímetro ha sido desarrollado enteramente para este proyecto, por lo que se añadirá el coste total de su fabricación. Se ha empleado a un técnico electrónico durante dos semanas a jornada completa, cuya valoración es de 16.946 € anuales según Infojobs Trends [41].

Para su fabricación se ha utilizado una placa MINI-32 Board de 23,00 € y un Bluetooth Click de 36,90 €.

$$C_{\text{pulsioxímetro}} = 16946€\frac{2}{52} + 23{,}00€ + 36{,}90€ = 711{,}60€ \tag{9.3}$$

9.1.5. Gastos de material fungible

Para la realización del proyecto se han utilizado políticas de conservación del medio ambiente, evitando imprimir documentos y consultándolos en formato electrónico. Por lo que el único coste imputable en esta categoría es la compra del material de papelería con valor de 20,00 €.

9.2. Gastos indirectos, riesgo y beneficio

Se incluyen en el presupuesto los costes indirectos asociados al proyecto, lo que incluye gastos que son difíciles de imputar directamente al mismo, como son el mantenimiento del local o el coste de la electricidad. Para ello se estima un porcentaje del 20 % sobre los gastos directos.

Se añade un porcentaje de riesgo cuya finalidad es sufragar posibles errores en el presupuesto. Dado que es un proyecto de corta duración, un retraso de una semana presenta un error de un 5 % aproximadamente. Para un margen de error de al menos dos semanas, se utilizará como valor de riesgo el equivalente a un 10 % de los costes totales, directos e indirectos, del proyecto.

Se debe tener en cuenta que el principal objetivo del desarrollo del proyecto es obtener beneficios, por lo que se suma al coste total del proyecto el margen de ganancias deseado. En este caso se establece el 10 % del valor de los gastos del mismo.

9.3. Tabla resumen y totales

En la tabla 9.2 se resumen y totalizan todos los gastos del proyecto detallados en las secciones previas.

El coste total del proyecto es de catorce mil setecientos treinta y nueve euros y noventa y siete céntimos (14.739,97 €), sin incluir el impuesto sobre el valor añadido (IVA). Aplicando el IVA vigente a febrero de 2014 (21 %), el total es de diecisiete mil ochocientos treinta y cinco euros y treinta y seis céntimos (17.835,36 €), aunque siempre se aplicará el IVA vigente en el momento del pago del proyecto.

Concepto	Base	Porcentaje	Total
Total gastos directos			**10.236,09 €**
Gastos de personal			9.389,00 €
Gastos de hardware			115,49 €
Gastos de software			0,00 €
Dispositivo externo			711,60 €
Material fungible			20,00 €
Gastos indirectos	10236,09 €	20 %	2047,22 €
Total gastos			**12.283,31 €**
Riesgo	12.283,31 €	10 %	1.228,33 €
Beneficio	12.283,31 €	10 %	1.228,33 €
Total sin IVA			**14.739,97 €**
IVA	14.739,97 €	21 %	3.095,39 €
Total			**17.835,36 €**

Tabla 9.2: Tabla resumen del presupuesto

10

Conclusiones

En este documento se ha descrito un sistema que permite medir y analizar los datos de distintos parámetros fisiológicos del usuario a través de su *smartphone* utilizando un dispositivo externo o pulsioxímetro. La aplicación desarrollada está dirigida a deportistas de élite o personas con problemas cardiovasculares que necesiten vigilar con frecuencia sus constantes vitales. El sistema proporciona una solución alternativa a los equipos médicos que requieren la presencia del usuario en un centro especializado.

A continuación, se comentan posibles lineas de trabajo futuro que permitirán ampliar el sistema y las conclusiones que se han sacado durante el desarrollo e implementación del mismo.

10.1. Trabajo futuro

A pesar de presentarse un sistema completo, existe una gran cantidad de puntos que pueden mejorarse. En este apartado se explican las futuras líneas de trabajo que pueden abarcarse con el objetivo de que se pueda continuar mejorando la aplicación, destacando una serie de aspectos a mejorar con objeto de una futura implementación.

10.1.1. Inclusión de nuevas variables

Durante el diseño del prototipo del sistema se decidió utilizar únicamente cuatro longitudes de onda. Sin embargo, se puede ampliar este rango espectroscópico de manera que la inclusión de nuevas longitudes de onda permita medir y analizar otros valores fisiológicos.

Los parámetros fisiológicos con los que se podría ampliar el proyecto son, por ejemplo, la citocromo oxidasa, la melanina, la bilirrubina, los lípidos y el nivel de saturación de azucar en sangre. Todo este nuevo cálculo de valores llevaría a una nueva investigación sobre el análisis de cada uno de ellos y de las mejores longitudes de onda para realizar su medición.

10.1.2. Mejora de los algoritmos de análisis

Los algoritmos utilizados para el procesamiento de los datos en la aplicación (ver sección 5.3) no son todo lo óptimo que deberían. A pesar de que se logran resultados bastante fiables en comparación con equipamiento médico homologado, no se consigue una fiabilidad del 100 %. Para presentar esta mejora, se necesitaría optimizar el código ejecutado para que fuera más eficiente y rápido.

En la actualidad, existe una gran variedad de terminales Android con diferente potencia de cálculo. Según el terminal en el que se ejecutase la aplicación, se podría tener en cuenta la potencia máxima que se puede obtener para dar paso a un tipo de algoritmo más pesado pero mas efectivo u otro. Esto permitiría distribuir la aplicación según las necesidades de cada uno de los terminales en los que fuese ejecutado.

Para la optimización del código se podría tener en cuenta utilizar la tecnología JNI (del inglés, *Java Native Interface*) que permitiría la inclusión de código de bajo nivel que se ejecutaría de manera mas eficiente.

10.1.3. Almacenamiento del histórico de datos

En su versión final, la aplicación desarrollada solo permite analizar los datos recibidos en el mismo momento de su toma. Esto podría no ser lo más deseado para un usuario que necesite monitorizar su estado vital a lo largo del tiempo.

Para solucionar este problema, se plantea almacenar un histórico de datos con la media de los valores procesados para que el usuario pueda consultarlos una vez que ha terminado el proceso de análisis con el dispositivo. Esto permitiría al usuario acceder a la aplicación para consultar su evolución en función del tiempo.

10.1.4. Portabilidad a otras plataformas

El desarrollo y la implementación de la aplicación se ha realizado únicamente para una plataforma Android debido a la escasez de tiempo y de personal para su realización. Como línea futura, se puede presentar la portabilidad de la aplicación a otras plataformas como iOS o Symbian.

10.2. Conclusiones personales

La realización del proyecto ha sido una gran labor de inmersión e integración de distintas tecnologías entre las que se incluye desarrollo sobre el framework Android, comunicación de red entre un dispositivo externo y la aplicación con la tecnología Bluetooth, y el cálculo y filtro de señales de espectroscopia.

El desarrollo del proyecto cubre prácticamente la totalidad de las materias impartidas en la titulación, exceptuando las de inteligencia artificial y en menor medida las de seguridad. Esto ha permitido completar mi formación principalmente en los ámbitos que formaban parte de mi especialidad en la ingeniería de los computadores. De esta forma, he descubierto la utilidad práctica de muchos conocimientos adquiridos durante la carrera, lo que ha servido de complemento a las prácticas de las asignaturas realizadas.

Además de consolidar los conocimientos adquiridos a lo largo de la carrera, he obtenido otros nuevos, especialmente en el ámbito de la física de señales y del diseño de interfaces, que me han permitido ampliar el rango de disciplinas conocidas dentro de la informática. Esto me proporciona una ayuda en lo referente a la realización de futuros proyectos de manera que pueda aplicar la tecnología correcta para resolver cada tipo de problema que se plantee.

Parte V

Anexos

Apéndice A

Matrix de trazabilidad de requisitos de usuario a casos de uso

	RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUC-09	RUC-10	RUC-11	RUC-12	RUC-13	RUC-14	RUC-15	RUC-16	RUC-17	RUC-18
CU-01	X	X		X														
CU-02			X	X														
CU-03					X	X	X	X	X	X	X	X	X	X	X			
CU-04					X	X	X											
CU-05							X											
CU-06																X		
CU-07																	X	
CU-08																		X
CU-09			X	X														
CU-10	X																	

Tabla A.1: Matriz de trazabilidad RU-CU

Apéndice B

Matriz de trazabilidad de requisitos de usuario a requisitos de software

	RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUC-09	RUC-10	RUC-11	RUC-12	RUC-13	RUC-14	RUC-15	RUC-16	RUC-17	RUC-18	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05	RUR-06	RUR-07	RUR-08	RUR-09	RUR-10	RUR-11	RUR-12	RUR-13	RUR-14	RUR-15
RSF-01	X																																
RSF-02	X																																
RSF-03			X																														
RSF-04				X																													
RSF-05				X																													
RSF-06					X																												
RSF-07					X																												
RSF-08						X																											
RSF-09						X																											
RSF-10							X																										
RSF-11								X																									
RSF-12									X	X	X	X	X	X	X																		
RSF-13																X																	
RSF-14																	X																

	RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUC-09	RUC-10	RUC-11	RUC-12	RUC-13	RUC-14	RUC-15	RUC-16	RUC-17	RUC-18	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05	RUR-06	RUR-07	RUR-08	RUR-09	RUR-10	RUR-11	RUR-12	RUR-13	RUR-14	RUR-15
RSF-15		X																				X											
RSF-16				X																													
RSF-17					X																												
RSF-18					X	X																											X
RSF-19	X			X																													
RSF-20			X	X																			X										
RSF-21				X	X	X				X	X	X																					
RSF-22										X		X	X	X	X																		
RSF-23							X									X	X	X															
RSF-24																					X												
RSF-25																											X	X					
RSN-01																			X														
RSN-02																			X														
RSN-03																				X													

	RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUC-09	RUC-10	RUC-11	RUC-12	RUC-13	RUC-14	RUC-15	RUC-16	RUC-17	RUC-18	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05	RUR-06	RUR-07	RUR-08	RUR-09	RUR-10	RUR-11	RUR-12	RUR-13	RUR-14	RUR-15
RSN-04																																X	
RSN-05					X																												
RSN-06					X																												
RSN-07																									X								
RSN-08									X																						X		
RSN-09					X																										X		
RSN-10																										X							
RSN-11																								X									
RSN-12																														X			
RSN-13																													X				

Tabla B.1: Matriz de trazabilidad RU-RS

Apéndice C

Diagrama de Gantt

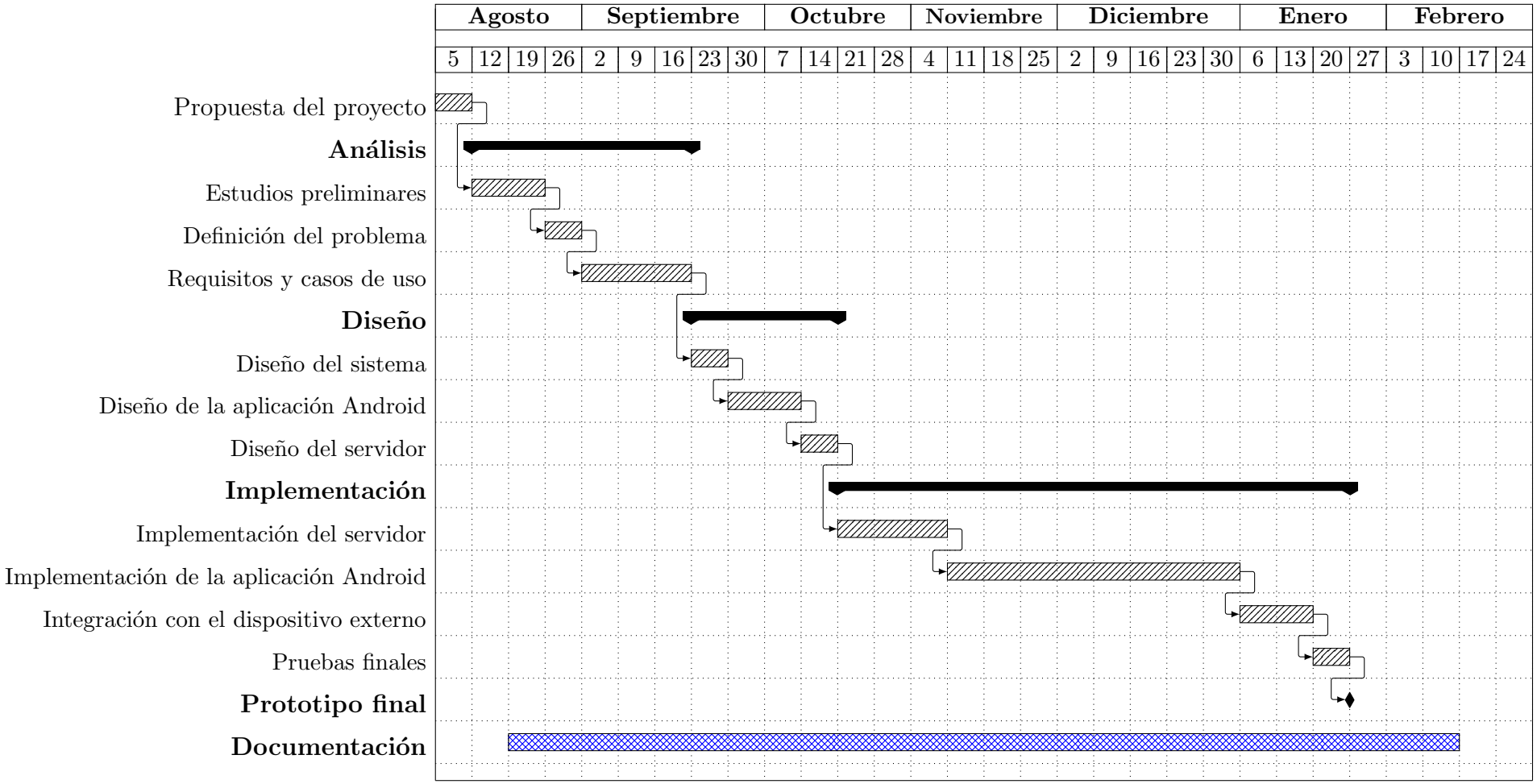

Figura C.1: Diagrama de Gantt del proyecto

Apéndice D

Glosario

api o *Application Programming Interface*, es una interfaz de programación de aplicaciones que contiene el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como capa de abstracción.

bytecode es el código intermedio generado en medio de la fase de compilación que es más abstracto que el código máquina y es tratado habitualmente como un archivo binario que contiene un programa ejecutable similar a un módulo objeto, que es un archivo binario producido por el compilador cuyo contenido es el código objeto o código máquina.

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

cromóforo es una sustancia que contiene gran variedad de electrones capaces de absorber energía o luz visible, y excitarse para emitir diversos colores, dependiendo de las longitudes de onda de la energía emitida por el cambio de nivel energético de los electrones, de su estado excitado a un estado fundamental o basal.

deoxihemoglobina o hemoglobina desoxigenada (Hb) es la hemoglobina que carece de oxígeno..

fhs o *Frequency Hopping Synchronization*, es un paquete especial de control que contiene, entre otras cosas, la dirección del dispositivo Bluetooth y el reloj del dispositivo de origen.

gps o *Global Positioning System*, es un sistema basado en la navegación espacial por satélite que provee información de localización en cualquier situación climática y en cualquier zona de la Tierra que se posea una linea de visión entre cuatro o más satélites de posición simultáneamente.

hal o *Hardware Abstraction Layer*, es un conjunto de herramientas software que proveen una capa de abstracción de hardware para distintos tipos de sistemas permitiendo a las aplicaciones detectar y usar el hardware a través de una API, sin importar el hardware sobre el que se estuviera ejecutando.

hemoglobina es una heteroproteína de la sangre que se encarga de transportar el oxígeno de la sangre desde los órganos respiratorios hasta los tejidos.

ISM o *Industrial, Scientific and Medical*, son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industriales, cientificas y médicas.

kernel o núcleo es el software que constituye la parte más importante del sistema operativo, siendo el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora encargándose de gestionar recursos, a través de servicios de llamadas al sistema.

oxihemoglobina o hemoglobina oxigenada (HbO_2) es la hemoglobina que está unida al oxígeno..

piconet o *pico*, es una red informática cuyos nodos se conectan utilizando la tecnología Bluetooth. Puede estar formada de dos a siete dispositivos en los que siempre existirá un *maestro*, siendo el resto de los dispositivos los *esclavos*.

rfcomm o *Radio Frequency Communication*, es un conjunto simple de protocolos de transporte que proporciona sesenta conexiones simultáneas para dispositivos Bluetooth emulando puertos serie.

sdk o *Software Development Kit*, es un kit de desarrollo de software que contiene el conjunto de herramientas de desarrollo que le permite al programador crear aplicaciones para un sistema concreto.

sim o *Subscriber Identity Module*, es una tarjeta inteligente desmontable usada mayoritariamente en teléfonos móviles que almacenan de forma segura la clave de servicio de subscriptor usada para identificarse ante la red, de forma que sea posible cambiar la línea de un terminal a otro simplemente cambiando la tarjeta.

smartphone o teléfono inteligente, es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y de realizar actividades semejantes a una minicomputadora proporcionando la misma conectividad que un teléfono móvil convencional.

Wi-Fi es un mecanismo de conexión de dispositivos electrónicos de manera inalámbrica por medio de la banda WLAN que permite cubrir grandes áreas de extensión mediante la superposición de múltiples puntos de acceso.

widget es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets. Entre sus objetivos están dar fácil acceso a funciones frecuentemente utilizadas y proveer de información visual al usuario.

WLAN o *Wireless Local Area Network*, es un sistema de comunicación inalámbrico flexible, muy utilizado como alternativa a las redes de área local cableadas o como extensión de éstas.

WPAN o *Wireless Personal Area Network*, es sistema de red de computadoras para la comunicación entre distintos dispositivos cercanos al punto de acceso.

XML o *eXtensible Markup Language*, es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

Apéndice E

Bibliografía

[1] *The Evolution of E-Health – Mobile Technology and mHealth.* URL: http://www.journalmtm.com/2012/the-evolution-of-e-health-mobile-technology-and-mhealth/.

[2] José Luis Palma Gámiz. *¿Qué es la eHealth?* URL: http://www.ediagnostic.es/Blog_de_telemedicina-V4-Que_es_la_eHealth.html.

[3] PlanetSeed. *Historia de la Medicina. El Siglo XX en Adelante. El papel de la tecnología.* URL: http://www.planetseed.com/es/relatedarticle/el-siglo-xx-en-adelante-el-papel-de-la-tecnologia.

[4] *The World's First Mobile Phone.* URL: http://news.bbc.co.uk/dna/place-lancashire/plain/A1082521.

[5] Reserach2guidance. *The Enterprise Mobile App Market Status Report 2012.* URL: http://www.research2guidance.com/shop/index.php/enterprise-mobile-app-market-status-report (visitado 10-06-2013).

[6] *History of the Bluetooth Special Interest Group.* URL: http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx (visitado 17-06-2013).

[7] *España es el país europeo con mayor penetración de smartphones.* URL: http://www.tuexperto.com/2013/04/16/espana-es-el-pais-europeo-con-mayor-penetracion-de-smartphones/ (visitado 16-04-2013).

[8] Reserach2guidance. *m-commerce Global Status Check (2012).* URL: http://www.research2guidance.com/shop/index.php/mcommerce-global-status-check (visitado 17-06-2013).

[9] Scully CG y col. *Physiological parameter monitoring from optical recordings with a mobile phone*. URL: http://www.ncbi.nlm.nih.gov/pubmed/21803676.

[10] *Introducción a la Espectroscopía*. URL: http://rabfis15.uco.es/lvct/tutorial/21/Espectroscopia.htm.

[11] Sergio Fantini's Group, Department of Biomedical Engineering y Tufts University. *Near-infrared spectroscopy for the study of biological tissue*. URL: http://www.docstoc.com/docs/80331658/Studying-brain-function-with-near-infrared-spectroscopy.

[12] Hale y Querry. *Optical Constants of Water in the 200-nm to 200-μm Wavelength Region*. 1 de mar. de 1973. URL: http://www.opticsinfobase.org/ao/abstract.cfm?uri=ao-12-03-555.

[13] Scott Prahl. *Tabulated Molar Extinction Coefficient for Hemoglobin in Water*. URL: http://omlc.ogi.edu/spectra/hemoglobin/summary.html.

[14] Noguerol Casado y Seco González. *Pulsioximetría*. URL: http://www.fisterra.com/material/tecnicas/pulsioximetria/pulsioximetria.pdf.

[15] Willem Gerrit Zijlstra y Anneke Buursma. *Visible and Near Infrared Absorption Spectra of Human and Animal Haemoglobin*. URL: http://books.google.es/books?id=Qn5yenybgtsC.

[16] Mendelson Y y col. *Spectrophotometric investigation of pulsatile blood flow for transcutaneous reflectance oximetry*. URL: http://www.ncbi.nlm.nih.gov/pubmed/6637643.

[17] *Lista de perfiles Bluetooth*. URL: http://es.wikipedia.org/wiki/Perfil_Bluetooth#Lista_de_perfiles.

[18] James Kardach Mobile Computing Group Intel Corporation. *Bluetooth Architecture Overview*. URL: http://www.blueradios.com/bluetooth_architecture.pdf.

[19] *Historia de la informática: Android*. 14 de dic. de 2012. URL: http://histinf.blogs.upv.es/2012/12/14/android/.

[20] *Open Handset Alliance*. 5 de nov. de 2007. URL: http://www.openhandsetalliance.com/.

[21] *Android version history*. URL: http://en.wikipedia.org/wiki/Android_version_history.

[22] *Distribución de versiones de Android*. 3 de dic. de 2013. URL: http://www.xatakandroid.com/mercado/android-4-4-kitkat-solo-esta-presente-en-el-1-1-de-dispositivos-tras-su-primer-mes.

[23] Irene Amador Román. *nfcTicketing: Aplicación para uso de tiques de transporte público*. 2013.

[24] *Bluetooth Assigned Numbers*. URL: https://www.bluetooth.org/en-us/specification/assigned-numbers/health-device-profile.

[25] Azumio Inc. *Instant Heart Rate App*. 12 de mar. de 2012. URL: https://play.google.com/store/apps/details?id=si.modula.android.instantheartrate.

[26] MacroPinch. *Cardiograph App*. 8 de ene. de 2014. URL: https://play.google.com/store/apps/details?id=com.macropinch.hydra.android.

[27] Sunset Apps. *Blood Sugar and Pressure, Oxygen*. 9 de jun. de 2012. URL: https://play.google.com/store/apps/details?id=com.sunsetapps.SanteFR.

[28] Bee-Creations. *LifeWatch Watching Life TM*. URL: http://www.lifewatch.com.

[29] ESA Board for Software Standardisation y Control (BSSC). *Guide to applying the ESA software engineering standards to small software projects*. Mayo de 1996. URL: http://emits.esa.int/emits-doc/e_support/Bssc962.pdf.

[30] IEEE. *Especificación de Requisitos según el estándar de IEEE 830*. 22 de oct. de 2008. URL: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.

[31] Mark Brandt. *Introduction to Absorbance Spectroscopy*. URL: http://www.rose-hulman.edu/~brandt/Fluorescence/Absorbance_Spectroscopy.pdf.

[32] J Cooley y J Tukey. *An algorithm for the machine calculation of complex Fourier series*. 1965. URL: http://www.math.uci.edu/~brusso/mallat51_4-6.pdf.

[33] R Sedgewick y K Wayne. *FFT Java Implementation*. 9 de feb. de 2011. URL: http://introcs.cs.princeton.edu/java/97data/FFT.java.html.

[34] A Savitzky y M Golay. *Smoothing and Differentiation of Data by Simplified Least Squares Procedures*. 1964. URL: http://pubs.acs.org/doi/abs/10.1021/ac60214a047.

[35] M Smith. *SGFilter Java Implementarion*. 2 de oct. de 2010. URL: http://orangepalantir.org/ijplugins/src/SavitzkyGolayFilter.html.

[36] Kusse Sukuta Bersha. *Spectral Imaging and Analysis of Human Skin*. URL: http://cimet.hig.no/content/download/29898/359733/file/Spectral\%20imaging\%20and\%20analysing\%20human\%20skin,\%20Kusse\%20Sukuta\%20BERSHA.pdf.

[37] *AndroidPlot*. URL: http://androidplot.com/.

[38] The MathWorks y NIST. *JAMA: A Java Matrix Package*. URL: http://math.nist.gov/javanumerics/jama/.

[39] *MikroElektronika*. URL: http://www.mikroe.com/.

[40] *BlueCove*. URL: http://bluecove.org/.

[41] *InfoJobs Trends*. URL: http://plandecarrera.infojobs.net/puesto-de-trabajo/analista%20programador.