Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL PARA EL PARSEO Y LA VISUALIZACIÓN DE LOS DATOS DE UN PULSIOXÍMETRO

MEMORIA

Alumno: Mikel Aingeru Palazuelo Sánchez

Director: Oskar Casquero Oyarzabal

Codirector: Harkaitz Eguiraun Martínez

Curso: 2017-2018

Fecha: 18-06-2018

RESUMEN

Existen estudios que avalan el análisis de muestras de la frecuencia cardíaca como método para diagnosticar varios tipos de patologías, tales como las enfermedades cardiovasculares. De estas muestras se puede obtener una de las señales fisiológicas más ricas e interesantes como es la señal pletismográfica.

El proyecto desarrollado como TFG da continuación a un trabajo iniciado en el marco del Convenio del Programa de Inmersión de Ingenieros firmado entre el Hospital de Cruces y la EUITI de Bilbao (hoy día EIB). Dicho trabajo consistió en el desarrollo de una pulsera para el muestreo de la señal pletismográfica y su envío vía Bluetooth.

En este proyecto se tomó ese dispositivo para obtener las muestras. Una vez tomadas, se parsean en una aplicación móvil y se visualizan en una gráfica dinámica que las dibuja en tiempo real. Al finalizar este proceso, se guarda un registro de la muestra en la nube para su posterior análisis con el fin de encontrar patrones que sirvan como indicativos para diagnosticar las patologías de cada paciente.

Palabras clave:

Frecuencia cardíaca, señal pletismográfica, aplicación móvil

LABURPENA

Bihotz-maiztasunaren laginen azterketa, gaixotasun ezberdinak aurkitzeko metodo bezala erabil daizteke, horrela bermatzen dute hainbat ikerketek. Lagin hauetatik, seinale fisiologiko oso interesgarria lor daiteke, seinale pletismografikoa hain zuzen.

GrAL bezala garatu den proiektu hau, beste proiektu baten jarraipena da. Proiektu horrek, **IITUE** (Industria Ingenieritza Teknikoko Unibertsitate Eskola, gaur egun **BIE** –*Bilboko Ingenieritza Eskola*-) eta Gurutzeta Ospitalaren artean sinatutako *Ingenieri Txertatze Programaren* barnean sortu zen. Bere helburua, seinale pletismografikoaren laginak lortzea eta Bluetooth bidez bildaltzen zituen eskumuturreko baten garapena izan zen.

Proiektu honetan, garatu zen eskumuturreko hori erabiliko da laginak lortzeko. Eta behin hauek lortu direla, aplikazio mugikor batean parseatu eta denbora errealean marrazten dira grafika dinamiko baten bidez. Prozesu hau bukatzean, parseatutako erregistro bat hodeira igotzen da. Ondoren, erregistro honen azterketa egingo da, gaixotasun ezberdinen adierazleak aurkitzeko helburuarekin.

Gako hitzak:

Bihotz-maiztasuna, seinale pletismografikoa, aplikazio mugikorra

ABSTRACT

There are studies that support the analysis of samples of the heart rate as a method to diagnose several types of pathologies, such as cardiovascular diseases. From these samples it is possible to obtain one of the most rich and interesting physiological signals as it is the plethysmography signal.

The proposal developed as a Degree Final Project is based on a previous work that was initiated within the framework of the course of "Engineer Immersion Program Agreement" signed between the EUITI (nowadays EIB) and the Cruces Hospital. This work consisted in the development of a wristband for the sampling of the plethysmography signal and its transfer via Bluetooth.

In this project, that device was used for obtaining the samples. Once taken, they are parsed in a mobile application and displayed in a real time graph. When this process is finished, a record of the sample is uploaded to the cloud for its later analysis with the purpose of searching patterns that serve as indicatives for diagnosing the pathologies of each patient.

Keywords:

Heart rate, plethysmography signal, mobile app

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ABREVIATURAS

Ámbito general

TFG: *Trabajo de Fin de Grado*

MOOC[1]: *Massive Open Online Course*

Ámbito de informática

SO[2]: *Sistema Operativo*

IoT[3]: *Internet of Things (Internet de las cosas)*

IDE[4]: *Integrated Development Environment (Entorno de Desarrollo Integrado)*

API[5]: *Application Programming Interface (Interfaz de Programación de Aplicaciones)*

APP: *Application (Aplicación)*

IA[6]: *Inteligencia Artificial*

BT: *Bluetooth*

UART[7]: *Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal)*

MAC[8]: *Media Access Control (Dirección MAC)*

RFCOMM[9]: *Radio Frequency Communication (Comunicación por Radio Frecuencia)*

CHK[10]: *Checksum (suma de verificación)*

CSV: *Comma-Separated Values (Valores Separados por Comas)*

UML: *Unified Modeling Language (Lenguaje Unificado de Modelado)*

MSB: *Most Significant Bit (Bit Más Significativo)*

LSB: *Least Significant Bit (Bit Menos Significativo)*

Ámbito de medicina

HR/FC[11]: *Heart Rate / Frecuencia Cardíaca*

HRV[12]: *Heart Rate Variability (Variabilidad del Ritmo Cardíaco)*

SpO2[13]: *Saturación de Oxígeno en sangre*

BPM/LPM[14]: *Beats Per Minute/Latidos Por Minuto*

Los conceptos con anotaciones están referenciados al final del documento en el apartado de GLOSARIO

1. Introducción

Años atrás, los datos, recursos y herramientas de cada persona estaban localizados en un único punto de acceso. Hoy día, todos esos recursos están "en la nube", es decir, el acceso se amplía a casi cualquier lugar y a través de diferentes dispositivos.

Adicionalmente, cada semana aparece un nuevo dispositivo de los denominados *wearables*[15]. La traducción literal es "vestibles", y hace referencia a un aparato con un conjunto de sensores integrados que podemos ponernos en alguna parte del cuerpo o ropa. Si además, subes los datos a la nube para almacenarlos o acceder a ellos de forma remota, se denominan *IoT* (*Internet of Things*) o Internet de las Cosas. Éstos se encargan de monitorizar aspectos de nuestra forma de vida sin necesidad de interacción. Los más extendidos son los del área de salud, bienestar y deporte, o sea, los que se orientan a controlar signos de nuestro estado fisiológico (pulso, horas y calidad del sueño, tensión…) y también contabilizan otras características para mejorar la salud mediante actividades (pasos andados, km recorridos, calorías consumidas, etc.). Para poder visualizar los resultados de los parámetros comentados, existen aplicaciones móviles de los mismos fabricantes o incluso de terceros.

Ahora bien, teniendo en cuenta todo lo anterior, sería ideal poder ampliar estas herramientas a fines más especializados del ámbito de la medicina. Una de las mayores limitaciones en las instituciones médicas es la siguiente: la falta de recursos, tanto de herramientas como de tiempo, espacio y personal preparado para cada caso. Sabiendo esto, sería un gran avance la posibilidad de democratizar esos recursos para evitar el colapso en las instituciones y liberar a los responsables llevar a cabo todas y cada una de las fases que hoy día se pueden automatizar.

Este es el resumen de la idea de este proyecto: poner al alcance de cada paciente las herramientas necesarias para poder dividir las tareas que no requieren de supervisión especializada en el proceso del tratamiento, de tal forma que el paciente aporta sus datos (análisis, síntomas, etc.) y la parte especializada se encarga de analizarlos y actuar en los casos que así lo requieran.

1.1. Contexto

Diversos estudios han llegado a la siguiente conclusión: examinando muestras de análisis de la frecuencia cardiaca, se pueden detectar indicadores para poder diagnosticar precozmente enfermedades de diferente índole tales como las cardiovasculares. Pero existe un factor bloqueante para la implementación de este tipo de procesos: la falta de recursos; no es viable transformar los hospitales en hoteles para analizar a cada paciente de forma exhaustiva, dado que colapsaría el sistema sanitario. Sin embargo, ¿qué ocurriría si tuviésemos una solución para poder tomar las muestras de manera independiente para cada paciente y sin necesidad de asignar recursos compartidos en cada uno?

El proyecto que ha sido desarrollado como TFG, es la continuación de una necesidad real que nació en el Curso de inmersión de ingenieros del Hospital Universitario de Cruces en el que participó un compañero de la especialidad de ingeniería electrónica: Josu Alonso (Ver Bibliografía).

Aunque inicialmente la idea consistía en validar la plataforma de adquisición denominada *"e-Health"* que obtenía señales biomédicas, tras determinar que no cumplía los requisitos mínimos se decidió construir una que obtuviera esas mismas señales de forma precisa y fiable para poder usarla en el área médica y científica.

Como prueba de concepto de la plataforma, de entre todas las señales muestreables se eligió la señal pletismográfica dado que los clínicos del Hospital tenían interés en observar su correlación con patologías del sueño en enfermos de la Unidad del sueño del Hospital de Cruces. Teniendo en cuenta el área y los casos en los que se iba a aplicar, era imperativo que la solución del desarrollo fuera un producto cómodo, ergonómico y portable. Por ello termino adaptándose a un de los antes mencionados *wearables*, con forma de pulsera.

El dispositivo tiene integrados los siguientes módulos:

Ilustración 1: Sensor NONIN

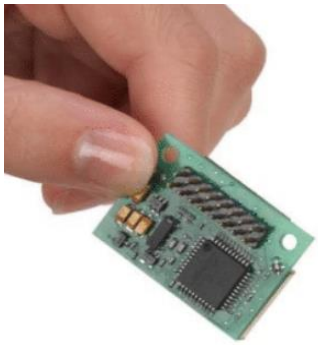


Ilustración 2: Módulo OEM III

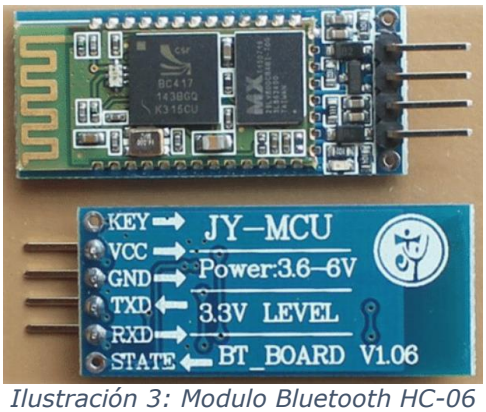


Ilustración 3: Modulo Bluetooth HC-06

Ilustración 4: Batería

Tabla 1: Componentes de la estación pletismográfica

COMPONENTE	FUNCIÓN
Pletismógrafo[16]	Módulo de adquisición de señal OEM III (NONIN) responsable de procesar la señal pletismográfica. En la fase de procesado: muestre, acondiciona y finalmente digitaliza la señal obtenida.
Pulsioxímetro	Adaptador con forma de pinza que se coloca en el dedo para medir la señal analógica del pulso y enviársela al pletismógrafo.
Bluetooth	Adaptador BT HC-06 para la conexión externa con el fin de transmitir los datos obtenidos.
Batería	Batería para convertir la pulsera en portátil por un tiempo limitado.
Sensor NONIN	Sensor periférico encargado de obtener la señal.

Ilustración 5: Pulsera para la estación pletismográfica construida mediante una impresora 3D

Ilustración 6: Carcasa con los componentes de la estación pletismográfica

Ilustración 7: Pulsera con estación pletismográfica integrada

Para poder visualizar los datos muestreados, el módulo OEM III incluía un software para el propósito:

Ilustración 8: OEM Evaluation Program

Una vez contextualizado el proyecto desarrollado por el compañero Josu Alonso, lo tomaremos como base para desarrollar el proyecto descrito en esta memoria, como una continuación del mismo. Como se ha comentado anteriormente en la breve explicación sobre los *wearables*, estos vienen acompañados de una aplicación móvil para presentar los resultados y expandir sus opciones: esa será la misión de este TFG:

La aplicación móvil obtiene los datos muestreados por el pletismógrafo y transferidos por el módulo BT. Una vez en el dispositivo móvil, los datos son parseados siguiendo el *Data Format #7* de la documentación del módulo pletismográfico (*Documentación OEM III – Nonin, ver Anexo I*), visualizados en una interfaz que muestra la gráfica en tiempo real de parseo y que finalmente guarda el registro del muestreo en la nube para su posterior examen en busca de patrones repetitivos que permitan relacionarlos con las patologías estudiadas en los pacientes.

A partir de este punto se tomarán estas referencias para nombrar a los diferentes componentes utilizados en el proyecto:

Tabla 2: Referencia a los componentes

Referencia	Componente	Autor
Aplicación (App)	Aplicación desarrollada en este TFG	Aingeru Palazuelo
Pulsera o *wearable*	Estación pletismográfica integrada en una pulsera	Josu Alonso

2. Objetivos y alcance del trabajo

El objetivo principal del proyecto es desarrollar una app móvil capaz de *parsear*[23] y visualizar los datos de la señal obtenida mediante la pulsera con el pletismógrafo y almacenarlos en la nube para su posterior análisis. Para alcanzar este objetivo se requerirá de una aplicación móvil con diversas pantallas para cada funcionalidad de la misma. Una de estas, la principal, debe mostrar el gráfico de la señal y otros datos significativos obtenidos de la misma señal como pueden ser el HR y el SpO_2. Otra se dedicará a gestionar los permisos, activación, escaneado y conexión Bluetooth. LA estará preparada para recibir los datos enviados desde la pulsera y de parsearlos para seguidamente mostrarlos por pantalla. Además, se guardará un registro de la señal parseada en la nube.

3. Beneficios que aporta el trabajo

Con el desarrollo de este proyecto, estos son los beneficios que le suma a la parte ya desarrollada de hardware en forma de pulsera:

- Primero, al desarrollar una aplicación móvil se expande enormemente el rango de accesibilidad dado que antes sólo se podían ver a través de la aplicación propietaria del pletismógrafo (*OEM Evaluation Program*).
- Segundo, dado que la app se puede conectar directamente con la pulsera, el paciente puede observar los resultados de la muestra en tiempo real sin necesidad de pedir una cita al hospital o de requerir atención de un especialista.
- Tercero, como las muestras se almacenan en la nube ya parseadas, estas podrían ser analizadas a más adelante por el médico y así reproducir cualquiera de ellas en el proceso de investigación de una concreta en la que haya surgido un cambio significativo o simplemente para obtener un evolutivo de ellas.

4. Descripción de requerimientos

En esta sección se describen los requerimientos básicos para obtener los objetivos marcados:

4.1. Aplicación móvil

Desarrollar una aplicación móvil para aumentar la accesibilidad de la muestra de los resultados sin necesidad utilizar el software propietario de la empresa del pletismógrafo (Nonin).

4.2. Conexión vía Bluetooth

Dado que la pulsera tiene un módulo BT integrado, la transferencia de datos se realiza mediante esta interfaz de conexión.

4.3. Gráfica para dibujar la señal pletismográfica

Con el objetivo de mejorar la comprensión del usuario, los datos de la señal pletismográfica se dibujan en un gráfico; para ello se buscará una librería o herramienta especifica que cumpla los requisitos adecuados, tales como: dibujar una gran cantidad de valores, hacerlo en tiempo real y actualizar la gráfica en un ritmo adecuado.

5. Análisis del estado del arte

En este apartado se hace un análisis de aplicaciones de los mercados de apps móviles con una funcionalidad o propósito similares.

5.1. Estudio de mercado

En esta sección se muestran las apps con funcionalidad similar de los mercados de aplicaciones móviles. En este caso, al haber decidido llegar al máximo de audiencia con el SO más utilizado, se ha investigado en *Google Play Store*. La metodología seguida para este fin, se ha basado en descargar las aplicaciones mejor valoradas y más descargas del mercado, entre ellas:

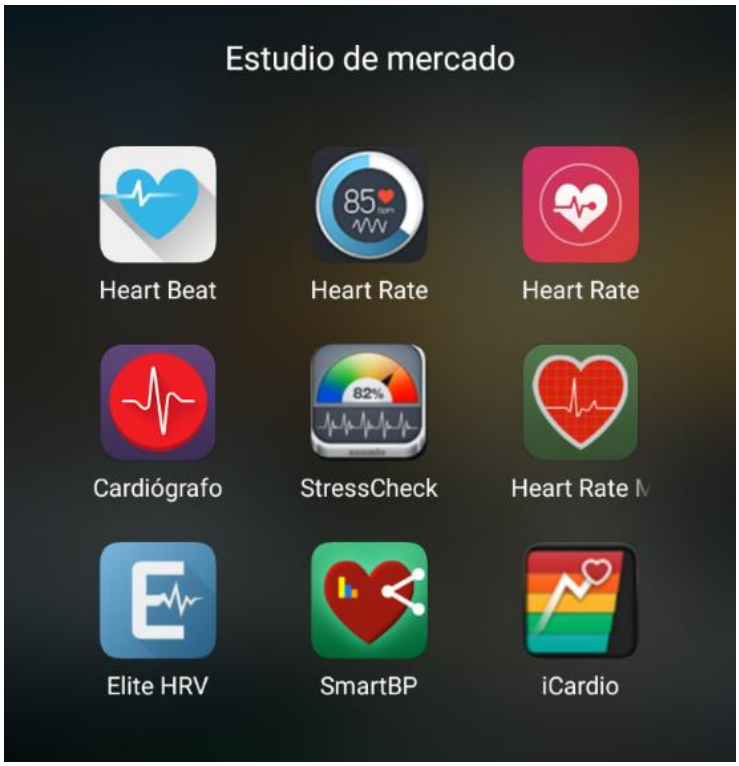


Ilustración 9: Aplicaciones probadas en el estudio de mercado

5.1.1. Elite HRV

El inicio de sesión permite hacerlo mediante la cuenta de Google o Facebook. Una vez dentro, muestra un mensaje para conectar el dispositivo que capture en la frecuencia cardíaca. De no tener tal dispositivo, nos da la opción de conseguir uno redirigiendo a su web (https://elitehrv.com/compatible-devices). Además de esto, la pantalla principal muestra un *feed*[1] de noticias con noticias e información relevante sobre la propia aplicación, información de deporte y otros artículos interesantes sobre los datos que se pueden obtener y utilizar en estas muestras.

Al no disponer de dispositivos de medición compatibles con la app, podemos ver los resultados que obtiene en su web de *Play Store* (https://play.google.com/store/apps/details?id=com.elitehrv.app).

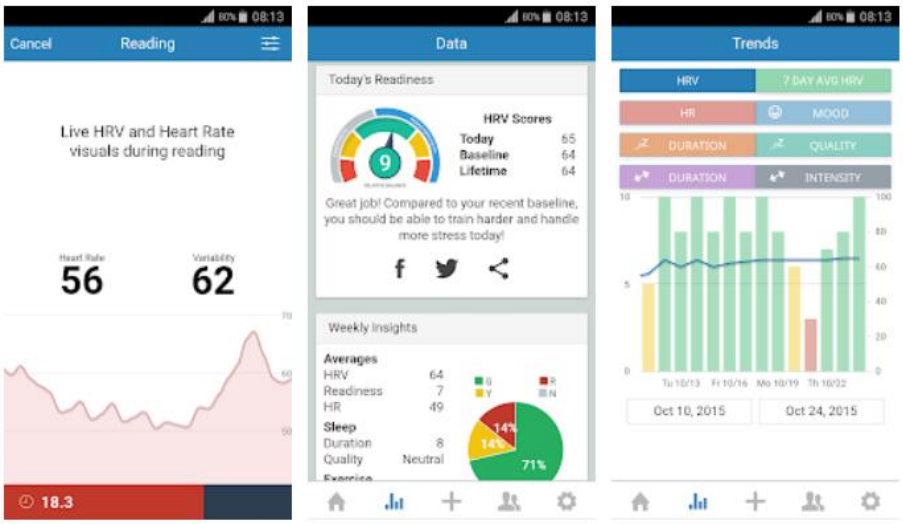


Ilustración 10: Play Store – Elite HRV

Como se observa en la imagen de la izquierda, muestra un gráfico con dos datos: la frecuencia cardiaca (*Heart Rate*) y el HRV (*Variability*). En la parte inferior se oculta el menú de navegación por una barra de progreso de la medición actual.

En la imagen del centro muestra un histórico de los resultados del día actual y semanales.

Y en la imagen derecha se puede ver el histórico entre fechas seleccionadas en un gráfico de barras en el que se puede seleccionar el dato que interese.

Como se observa en el menú también hay pantallas para "equipos" donde pueden agregar otros usuarios (se sugiere agregar al entrenador personal como ejemplo) y una última para el perfil de usuario y la configuración.

[1] **feed:** una columna de artículos que se alimenta o actualiza frecuentemente, sobre noticias relacionadas con el propósito de su contenedor (web, red social, blog…).

5.1.2. Instant Heart Rate (Azumio)

Al acceder a la aplicación por primera vez se recomienda al usuario iniciar sesión para guardar tus mediciones, existe la opción de hacerlo mediante la cuenta de Facebook o Google. Seguido, te pide unos datos básicos como género y fecha de nacimiento.

Te da la opción de actualizar a la versión *premium* para obtener más características.

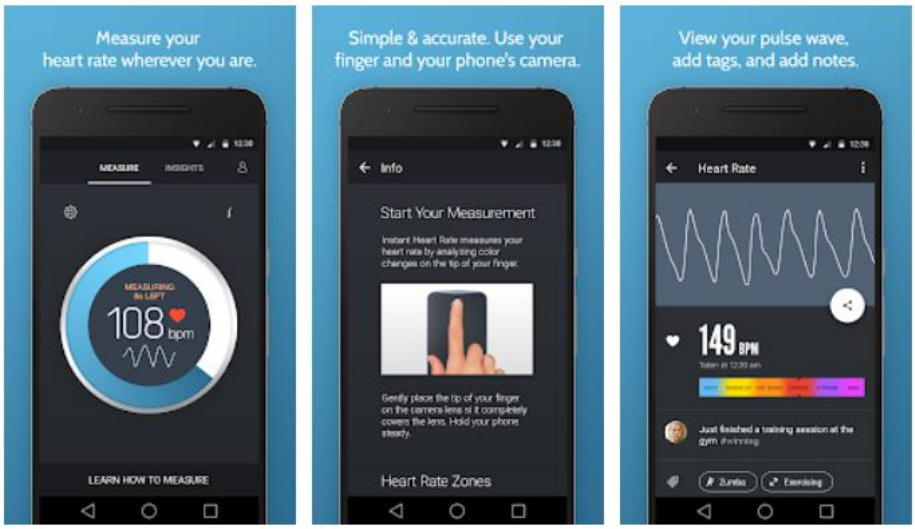


Ilustración 11: Play Store - Instan Heart Rate

Una vez en la pantalla principal de la app (imagen de la izquierda), aparece el característico circulo que muestra durante la medición como barra de progreso circular y dentro de esta los datos: la cuenta atrás para finalizar la medición (de 10 segundos) y el valor del pulso sobre una gráfica que lo representa.

Al finalizar la medición te redirige a otra pantalla en la que te muestra el resultado con la media de pulsaciones y una barra de colores para posicionar esa media según tu edad y género (imagen de la derecha). Para ver más datos como la gráfica es necesario actualizar a la versión premium.

Además de esto, también cuenta con unas instrucciones para llevar a cabo la medición mediante el flash y la cámara. Más adelante se explica cómo funciona este tipo de medición ya que lo usan el resto de aplicaciones que se han estudiado.

5.1.3. Cardiógrafo

Al arrancar la aplicación pide los permisos para usar la cámara, los necesita para llevar a cabo la medición, también da una breve explicación de cómo funciona. Y seguido aparece un *disclaimer*[29] para explicar que, aunque sea lo suficientemente precisa, no puede reemplazar el equipo médico profesional.

Ilustración 12: Play Store – Cardiógrafo

El menú de navegación de esta app es más simple, tiene tres pestañas: La primera y principal, para hacer la medición, una segunda para ver el historial de las mediciones y una tercera para los ajustes de perfil.

5.1.4. Runtastic Heart Rate

Al iniciar la aplicación aparece una pantalla de inicio de sesión en la que la existe la posibilidad de hacerlo con la cuenta de Google, Facebook o registrarse con el email.

Después de iniciar sesión, muestra un pequeño tour con instrucciones para medir el pulso, conocer el historial de mediciones y hacer la medición de FC (Frecuencia Cardíaca) en reposo.

Como se puede observar en la imagen de la izquierda, esta aplicación también ha optado por una gráfica circular para mostrar la barra de progreso como cuenta atrás durante la medición. Dentro de esta muestra el dato del pulso en unidad ppm (pulsaciones por minuto). Bajo esta aparece un gráfico que representa la frecuencia cardíaca.

Al finalizar la medición del pulso te envía a la pantalla de la izquierda donde te posiciona tu resultado en una barra con rangos de estado. Además, puedes editar la medición con el tipo (general, en reposo, antes de ejercicio, después de ejercicio y MÁX RC), la sensación que tuviste durante ella y una nota.

Al salir de esta vista te propone compartir la medición.

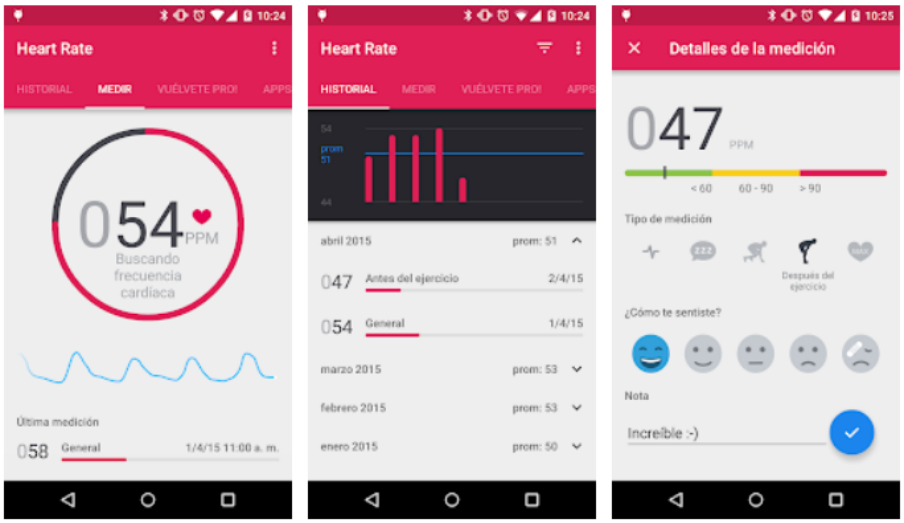


Ilustración 13: Play Store - Runtastic Heart Rate

Mediante el menú superior se puede navegar a la pestaña de historial (imagen central) y otras como "¡Hazte Pro!" para actualizar a la versión premium y otra llamada "Más Apps" que muestra otras aplicaciones de la marca.

5.2. Tabla de aplicaciones

En la siguiente tabla se muestra el resumen de las aplicaciones comparadas con los siguientes datos:

- **Aplicación**
  - **Nombre:** nombre de la app con hipervínculo a su página dentro de la *Play Store*.
- **Descripción:** descripción que aparece en la pantalla de la app en *Play Store*.
- **Núm. de descargas:** número de descargas que tiene la app desde *Play Store*.
- **Valoración:** media de la puntuación que le han dado los usuarios en *Play Store*.

Tabla 3: Estudio de mercado

Aplicación	Descripción	Descargas	Valoración
Elite HRV	Heart Rate Variability with an emphasis on accuracy, professional quality, and ease of use. Track recovery and true stress levels. Boost resilience and your nervous system with guided breathing and live HRV biofeedback.	100m	4.4
Instant Heart Rate (Azumio)	Convierte el dispositivo móvil en un monitor de ritmo cardíaco. Rápida y precisa.	10M	4.3
Cardiógrafo	¿A qué ritmo va tu frecuencia cardíaca? Mide tu pulso.	10M	3.8
Runtastic Heart Rate	Convierte tu Smartphone en tu monitor de frecuencia cardíaca personal!	1M	4.3

5.3. Conclusiones del estudio de mercado

Como hemos resumido en el punto anterior, la primera diferencia significativa entre las aplicaciones estudiadas y la app a desarrollar, es el método de obtención de datos: estas utilizan la cámara del dispositivo en conjunto con el flash del mismo para adquirir los datos, siendo este método poco o menos fiable que mediante un pletismógrafo que está desarrollado para ese propósito.

En la siguiente imagen podemos ver como la app explica en su página de *Play Store* como hacer la medición:

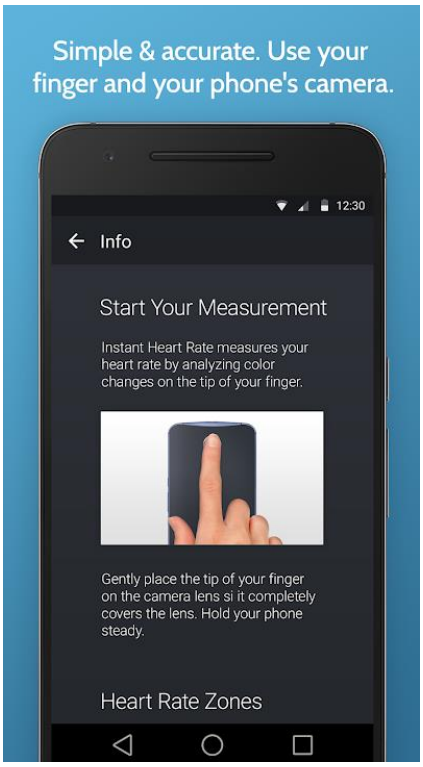


Ilustración 14: Medición vía flash

Con una pequeña investigación sobre el funcionamiento de este método, esta sería una breve explicación:

1. El flash ilumina la punta del dedo que está colocado sobre la cámara.
2. Esta, analiza el cambio del color en la punta del dedo.
3. El cambio surge con el cambio de densidad de la sangre en ese punto, que se efectúa con cada pulsación.
4. Se observa el cambio para simular cada latido y se contabilizan para mostrar el pulso.

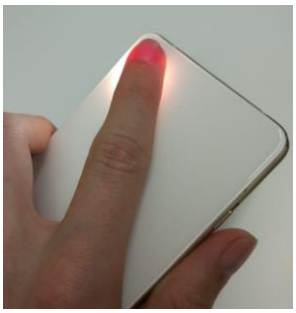


Ilustración 15: Foto de medición con flash

En la ilustración anterior se puede apreciar como el flash del dispositivo ilumina la punta del dedo colocado sobre la cámara.

Este método tiene ventajas del este método:

- Es accesible para la amplia mayoría de dispositivos móviles modernos.
- Se puede utilizar en cualquier momento y lugar.
- Los resultados son rápidos.

Por otro lado, estás son las desventajas:

- Los resultados no son lo suficientemente fiables para el objetivo de audiencia o ámbito para el que se desarrolla este proyecto.
- Tras varias mediciones o una algo más extensa, el flash puede quemar un poco al contacto.

Con este balance, descartamos completamente este método y clasificamos la en una rama diferenciada por si tipo de audiencia y método de obtención de datos.

Respecto a la interfaz de usuario, casi todas las aplicaciones coinciden, en cierto modo, en las pantallas y menú de navegación que muestran en la parte inferior de la pantalla:

- **Ayuda**: pantalla con instrucciones para aprender a manejar la aplicación.
- **Resultado del análisis**: Información extraída de una de las mediciones registradas.
- **Medir pulso**: Actividad principal donde se mide el pulso y se muestra en tiempo real.
- **Historial**: Registro de las mediciones registradas.
- **Ajustes/Perfil**: Ajustes de la aplicación y/o información del perfil.

Se tomarán estas pantallas como base para el desarrollo de la y se ajustarán a las necesidades y funcionalidades propias.

En relación al inicio de sesión, las más populares han implementado la posibilidad de hacerlo mediante la cuenta de Google u otras redes sociales, este punto podría ayudar a sacar provecho para añadir funcionalidades como utilizar los servicios y APIs de Google (Google Maps, Google Drive, etc.) para futuras versiones de la app.

Teniendo en mente todo lo anterior y después de hacer un *brainstorming*[32] y verificar su disponibilidad en el mercado, la aplicación desarrollada se llamará:

HealthyBeat

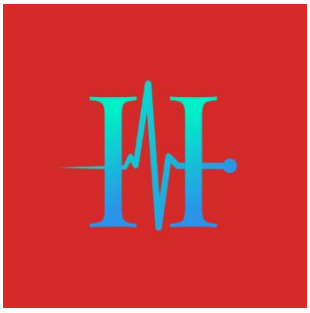


Ilustración 16: HealtyBeat Logo

Siendo un nombre en inglés pensando en opciones de internacionalización y comprensión a mayor escalada y con la conjunción de palabras:

1. ***Healthy*** del inglés "saludable"
2. ***Beat*** del inglés "latido" (o pulso)

Se conforma un nombre del que se puede interpretar "*latido (o pulso) saludable*", algo que sin duda inspira confianza y optimismo para los usuarios.

En la siguiente tabla se muestra la comparativa de las funcionalidades de cada aplicación probada:

- **Datos precisos:** se refiere a la precisión de los datos que muestra y los resultados que obtiene. Para ello es importante usar un método de captura de la muestra fiable, por ejemplo, conectándose a un dispositivo externo preparado para ello.
- **Gráfico:** ¿Muestra un gráfico de la señal?
- **Almacenamiento en la nube:** ¿Almacena los datos en un repositorio en el que se puedan consultar los datos de la muestra?
- **Gratuita:** ¿se puede acceder a todas las funcionalidades sin pagar?

Tabla 4: Resultados de la comparativa

Nombre	Datos precisos	Gráfico	Menú de navegación	Almacenamiento en la nube	Gratuita
Elite HRV	SÍ	SÍ	SÍ	NO	NO
Instant Heart Rate	NO	SÍ	SÍ	NO	NO
Cardiógrafo	NO	SÍ	SÍ	NO	SÍ
Runtastic Heart Rate	NO	SÍ	SÍ	NO	NO

6. Análisis de alternativas

En este apartado se analiza las posibles soluciones para el desarrollo de cada tarea.

6.1. Tipo de aplicación móvil

Para desarrollar una aplicación móvil, existen tres opciones:

- Aplicación nativa
- Web App
- App híbrida

Se decidió desarrollar una aplicación nativa para tener acceso total al dispositivo y así evitar inconvenientes con el hardware, como por ejemplo con el adaptador Bluetooth.

El mayor inconveniente ha sido aprender el lenguaje específico y la estructura de proyecto del SO para el que se ha desarrolla.

6.1.1. App nativa

Aplicación móvil desarrollada en el lenguaje propio del SO en el que se va a ejecutar. Por ejemplo, Android se desarrolla en JAVA para la parte lógica y XML para diseñar la interfaz e iOS[33] en Objective-C o Swift (lenguaje propio y orientado a su software).

6.1.2. Web App

Aplicación web desarrollada en lenguajes web, pero adaptados a dispositivos móviles. Su mayor ventaja es ser multiplataforma, ya que todos los dispositivos tienen un navegador web desde el que acceder a la app. Por tanto, no hace falta publicarlas en un mercado de apps móviles.

6.1.3. App híbrida

Es una mezcla de las dos opciones anteriores, aunando sus ventajas. Es decir, se desarrollan en lenguajes web y se pueden distribuir para diferentes plataformas, pero también pueden a gran parte de las características del hardware. Uno de los framework[34] más utilizados para el desarrollo de estas apps son PhoneGap de Adobe (https://phonegap.com/), Cordova de Apache (https://cordova.apache.org/) e Ionic (https://ionicframework.com/) que utiliza Cordova internamente.

6.2. SO de la aplicación

Una vez tomada la decisión de desarrollar una aplicación nativa, hubo que decidir para que SO hacerlo:

En post de hacer la app lo más accesible posible, se ha desarrollado una aplicación para Android, dado que tiene mayor cuota de mercado en dispositivos móviles y es la opción más accesible.

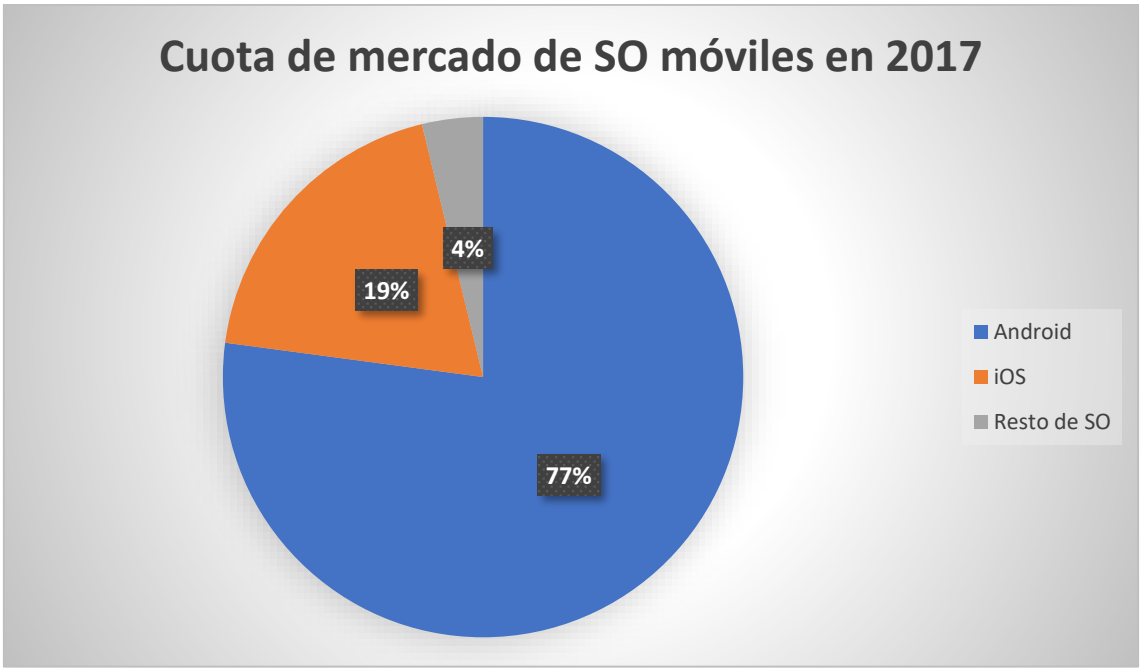


Ilustración 17: Cuota de mercado SO móvil 2017

Como se ve en el gráfico anterior, la ventaja de Android respecto a sus competidores del mundo móvil es abrumadora. Los datos del gráfico han sido obtenidos de la web http://gs.statcounter.com/os-market-share/mobile/worldwide/.

6.3. IDE de desarrollo para app Android

Aunque en los primeros años de su existencia, las aplicaciones para Android se desarrollaban en el IDE Eclipse por ser uno de los más extendidos para proyectos en JAVA, hoy día, Android tiene su propio IDE especializado para desarrollo de aplicaciones para su SO: Android Studio, desarrollado por IntelliJ (JetBrains).

6.4. Repositorio de control de versiones

Para mantener un control de versiones sobre el código del proyecto, se ha utilizado un software de control de versiones web. Por los siguientes puntos a favor se ha decidido utilizar GIT (https://git-scm.com/):

- Experiencia previa
- Es gratuito
- No hay límite de repositorios
- La mayoría de las librerías y código de proyectos utilizados para aprender se alojan en su web, GitHub (https://github.com/).
- Tiene cliente de escritorio:
  GitHub Desktop (https://desktop.github.com/)
- Tiene *plugin*[35] integrados en Android Studio
  Tutorial: https://androidstudiofaqs.com/tutoriales/como-usar-git-en-android-studio

También existen otras alternativas interesantes como Bitbucket (https://bitbucket.org/) de Atlassian (https://es.atlassian.com/), que posibilita la creación de repositorios privados.

6.5. Librería para dibujar la gráfica del pulso

Vistos los requerimientos para dibujar la señal pletismográfica, deberá cumplir los siguientes requisitos:

1. Ser capaz de dibujar una gran cantidad de datos.

2. Mostrarlos en forma de gráfico.

3. Actualizar los datos a medida que se van parseando.

Para cubrir estas necesidades se han probado varias librerías.

6.5.1. Snake View

Librería del repositorio de Android (https://android-arsenal.com), parecía la indicada ya que en el ejemplo muestra un gráfico de pulso mostrando el dato en unidad bpm como se muestra en la siguiente imagen:

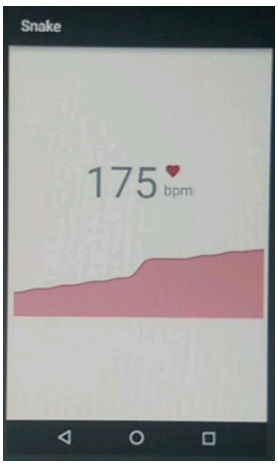


Ilustración 18: Captura librería Snake View

Se utilizo para las pruebas iniciales de interfaz, pero se descartó porque no mostraba el tipo de gráfica indicado para la señal pletismográfica.

6.5.2. Spark

Como segunda opción se probó a implementar la librería Spark del mismo repositorio que la anterior, esta sí que cumplía con el tipo de gráfica que se buscaba:

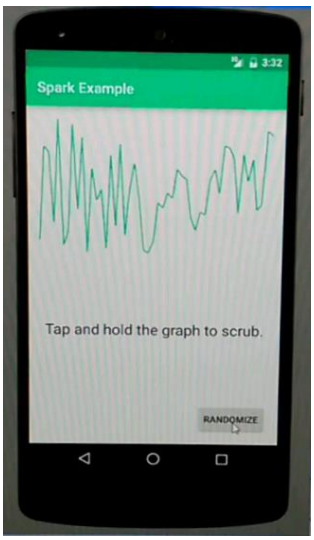


Ilustración 19: Captura de librería Spark

Sin embargo, al pasar a pruebas reales con datos dinámicos, aparecieron las limitaciones de la misma: no estaba preparada para actualizar los datos del gráfico y dibujarlos en tiempo real.

6.5.3. GraphView

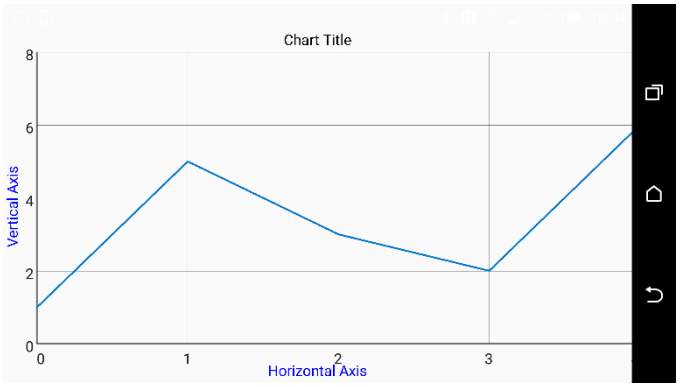


Ilustración 20: Librería GraphView

Por último, se implementó la librería GraphView. Aunque a priori no parecía indicada (basándose en la siguiente ilustración), las opciones de su API llamaban la atención, por ejemplo:

- Varios tipos de gráficas: líneas, barras, puntos, mezclas de las anteriores…
- Dibujado en tiempo real
- Zoom y scroll
- Diferentes formas de crear el gráfico: de forma programática o mediante XML
- Diferentes estilos

Con la opción de dibujar los datos en tiempo real se solventó la limitación de la librería anterior. Esta otra, usaba un *viewport*[1] para mostrar el rango de datos que se muestran por pantalla (ejemplo en la siguiente ilustración):

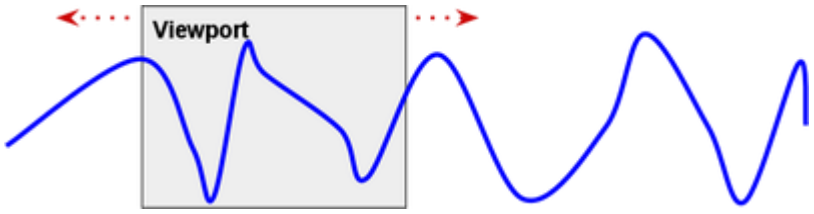


Ilustración 21: GraphView Viewport

Gracias a esto se pueden dibujar los datos de forma dinámica y se le añade la posibilidad de volver a ver los datos ya dibujados y desplazados con la opción de *scrolling*[2].

---

[1] **Viewport:** área de la pantalla que esta disponible para visualizar el contenido. *Véase:* *https://www.arsys.es/blog/programacion/diseno-web/viewport-diseno-responsive/*

[2] **Scrolling:** desplazamiento horizontal/vertical en planos 2D.
Véase: https://es.wikipedia.org/wiki/Scroll

6.6. Repositorio en la nube donde almacenar las muestras

Para guardar los registros de las muestras en un repositorio en la nube, se han estudiado las siguientes alternativas:

1. Google Drive (https://www.google.com/intl/es_ALL/drive/)
2. Dropbox (https://www.dropbox.com/)
3. Microsoft OneDrive (https://onedrive.live.com/)

Se ha utilizado Google Drive por visión de futuro. Se podría añadir el inicio de sesión con la cuenta de Google y así aprovechar la compatibilidad entre los servicios. Desde el punto de vista programático, tiene una API ya familiar dado que se la aplicación es Android y también pertenece a Google.

6.6.1. Google Drive

El servicio en la nube de Google, cuenta con 15GB que pertenecen a la cuenta de Google; se comparten con el servicio de Gmail y de Google Fotos. Esta preparado para compartir los archivos e integrado en Gmail. Tiene un cliente de escritorio para mantener los archivos sincronizados.

6.6.2. Dropbox

También da opción a registrarse e iniciar sesión con la cuenta de Google. La capacidad de almacenamiento es menor (2.5GB) en una cuenta normal. Tiene un cliente de escritorio para mantener los archivos sincronizados.

6.6.3. Microsoft OneDrive

Es el servicio de alojamiento de Microsoft (https://www.microsoft.com/es-es), cuenta con 5GB de almacenamiento gratuito. Tiene un cliente de escritorio para mantener los archivos sincronizados.

6.7. Script para emular la pulsera

Por un problema técnico con la pulsera, ha surgido la necesidad de desarrollar un script[36] de que simule el funcionamiento de la misma para poder sustituirla. Para ello, se han tenido en cuenta dos alternativas: desarrollarlo en JAVA o en Python. Pero dado que Android se programa en el leguaje JAVA, se ha optado por Python.

Por lo tanto, el IDE para desarrollarlo sera PyCharm (JetBrains).

7. Análisis de riesgos

En la siguiente tabla se analizan los riesgos directos e indirectos que podrían surgir durante el desarrollo:

Tabla 5: Análisis de riesgos

Riesgo	Causa	Probabilidad	Impacto	Prevención	Contingencia
ANÁLISIS: Planificación incorrecta	- Inexperiencia - Proyecto de riesgo - Mal cálculo de tiempos	Alta	Alto	Tiempos de cada fase más holgados	Esfuerzos de tiempo puntuales y ejecución de tareas en paralelo
ANÁLISIS: Presupuesto ajustado	- Falta de fuente de ingresos - No rentabilizar el proyecto (de manera inmediata) - Presupuesto personal	Alta	Medio	Buscar recursos públicos y no dañar ni perder los obtenidos	Utilizar recursos personales, públicos, gratuitos
DESARROLLO: Paralizar el progreso del proyecto	- Enfermedad - Accidente - Causa mayor	Baja	Medio	Tener tiempo de margen	Esfuerzos de tiempo puntuales y ejecución de tareas en paralelo

Riesgo	Causa	Probabilidad	Impacto	Prevención	Contingencia
DESARROLLO: Reducción de tiempo posible dedicado	- Contrato laboral de jornada completa - Incompatibilidad de horario - Fatiga o estrés	Media	Alta	Organizar el tiempo y centrarse en el proyecto cuando sea posible	Dedicar tiempos más ajustados pero regulares
DESARROLLO: Pérdidas parciales o totales de código u otros archivos	- Fallo del IDE, SO o máquina - Error de sincronización - Fallo del software de almacenamiento en la nube - Error por parte del programador al guardar u olvidarlo	Media	Alta	Utilizar software de control de versiones y repositorios sincronizados en la nube	Trabajar con la última versión estable del código/archivos y hacer seguir protocolos de sincronización con el repositorio

Riesgo	Causa	Probabilidad	Impacto	Prevención	Contingencia
DESARROLLO: Bloqueo del progreso por dificultades técnicas	- Alta complejidad del código (algoritmos, protocolos, estructura, lenguaje…)	Alta	Medio	Tener a mano la API y otra documentación de cada software	Seguir recomendaciones de la API del propietario o desarrollador del software y buscar en foros de programación
DESARROLLO: Pérdida o fallo de hardware	- Avería o perdida de la maquina u otro elemento de hardware (pulsera)	Baja	Alto	Utilizar los recursos asignados al proyecto con seguridad	Arreglar la avería si es posible o proveerse de nuevos recursos lo antes posible
DESARROLLO: Mal funcionamiento del hardware	- Error, avería o datos incorrectos obtenidos por el hardware	Media	Alta	Guardar registros de muestras fiables para estos casos	Utilizar los registros (backups[1]) de emergencia para estos casos
DESARROLLO: Incompatibilidad de software	- Actualización de alguna tecnología o software (*Deprecated*[2])	Alta	Medio	Estar al tanto de las actualizaciones	Aplicar las recomendaciones de la nueva API de cada caso o buscar alternativas

[1] **Backup:** Copia de seguridad utilizada como prevención para casos de pérdida o error.
Véase: https://es.wikipedia.org/wiki/Copia_de_seguridad

[2] **Deprecated:** termino usado para métodos, clases, librerías… que han quedado obsoletas y han sido sustituidas por otras nuevas.
Véase: https://en.wikipedia.org/wiki/Deprecation#Software_deprecation

Riesgo	Causa	Probabilidad	Impacto	Prevención	Contingencia
	- Nuevas políticas de permisos - Conflictos entre versiones o SO				
DESARROLLO: Incompatibilidad de hardware	- Fallo en la interconexión de los componentes hardware (pulsera y dispositivo móvil)	Media	Alto	Estudiar la compatibilidad previamente	Buscar dispositivos alternativos que sean compatibles

8. Diseño de alto nivel

En el siguiente diagrama de bloques se representa el funcionamiento interno del sistema y la relación entre los diferentes bloques:

1. La **estación pletismográfica** o pulsera para abreviar, muestrea la señal y la envía mediante tecnología Bluetooth al dispositivo Android.

2. La aplicación **HealthyBeat** recibe los datos de la señal desde la pulsera los parsea y los muestra por pantalla. Inicialmente, debía recibir estos datos de la pulsera (Diagrmaa de bloques - versión 1-), pero ha surgido un error con esta y se ha sustituido por un script desarrollado en Python que emula su funcionamiento, enviando una muestra forma cíclica. Dicha muestra ha sido tomada con anterioridad desde la pulsera y guardad en un fichero csv para este tipo de casos.

3. Posteriormente se guarda un registro de la muestra en el servicio en la nube de **Google Drive**.

Ilustración 22: Diagrama de bloques con pulsera (1º versión)

El siguiente diagrama es el que finalmente se ha implementado, por motivo de la avería de la pulsera.

Ilustración 23: Diagrama de bloques con emulador de pulsera (2º versión)

El conjunto de la pulsera consta de los siguientes componentes:

- **Pulsioxímetro periférico**
- Módulo de adquisición de señal **OEM III (Nonin®)**.
- Módulo **Bluetooth HC-06**
- **Batería**: sirve para dar cierta independencia a la pulsera

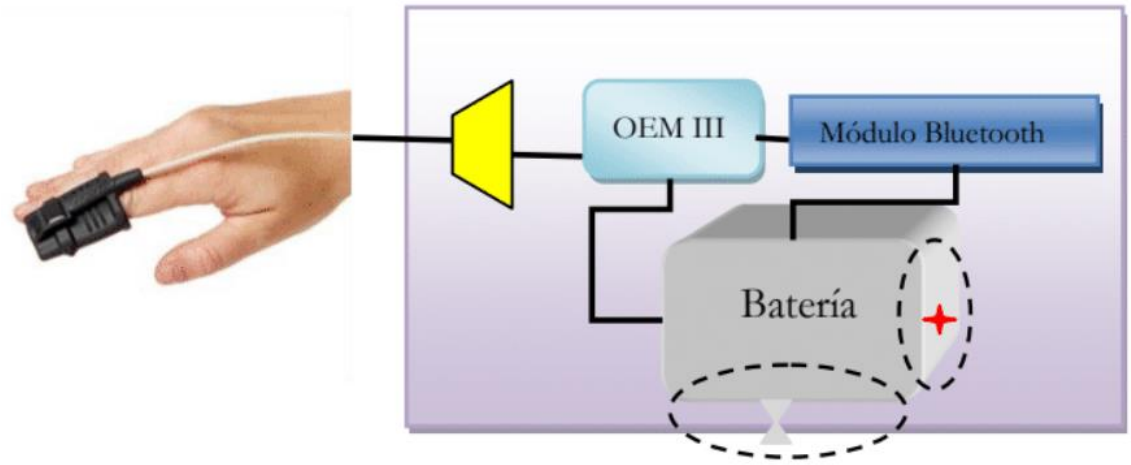


Ilustración 24: Obtención de la señal

La señal pletimográfica analógica es obtenida mediante el pulsioxímetro periférico de la pulsera y transferida al módulo de adquisición OEM III.

El módulo **OEM III** muestrea, acondiciona y digitaliza la señal y seguidamente transfiere esta señal en formato digital al módulo bluetooth mediante con la interfaz de conexión de puertos serie UART (*Universal Asynchonous Receiver-Transmitter*), que comparten los dos y que facilita la transmisión de datos entre ellos. El primero manda los datos a partir del puerto de transmisión (T_x) y el segundo los recibe por el puerto de recepción (R_x).

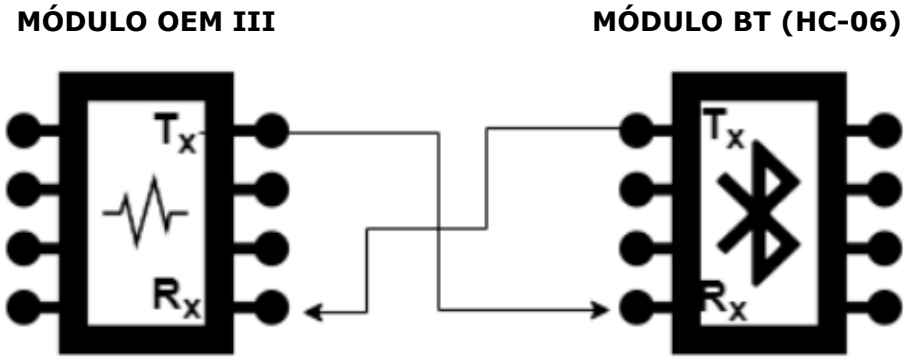


Ilustración 25: Conexión de puertos serie UART

Una vez que la señal has llegado al módulo BT, este se mantiene pendiente de una petición para la conexión BT. Esta parte ha sido sustituida por el módulo de BT del emulador de pulsera, pero el funcionamiento es el mismo ya que el protocolo de conexión no ha cambiado.

Cuando la app *HealthyBeat* se conecta a esta y se vinculan (ver imagen izquierda "Pantalla Bluetooth"), comienza la transmisión de datos de la señal de la pulsera al dispositivo. Sin embargo, dicha señal no se parsea en la app hasta que el usuario pulsa el botón ANALIZAR de la pantalla principal (ver imagen derecha "Pantalla Analizar").

Ilustración 26: Pantalla Bluetooth

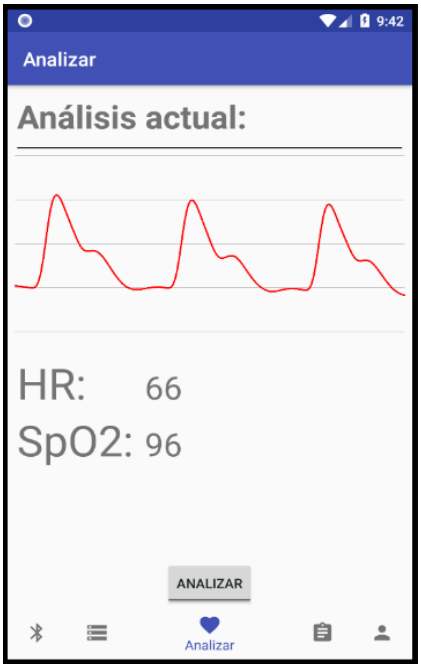


Ilustración 27: Pantalla Analizar

Al pulsar el botón comentado, comienza el parseo de la señal y una vez parseada se mandan los datos a la vista para ser pintados en la gráfica como se observa en la imagen anterior.

Al finalizar el parseo, los datos utilizados para dibujar la gráfica son almacenados en el servicio *cloud* de Google Drive.

Ilustración 28: Registros de las muestras en Google Drive

9. Descripción de fases y tareas

En este apartado se muestran las fases seguidas en el desarrollo del proyecto y la descripción de las tareas.

9.1. Fases del proyecto

Estas son las fases principales que se han llevado a cabo durante el desarrollo del proyecto:

1. **Análisis**: reuniones con el director de proyecto para definir los requerimientos y objetivos. estudio de proyectos similares para aprovechar sus puntos fuertes y buscar el valor añadido. Por otra parte, estudiar las alternativas más adecuadas para el caso.
2. **Formación**: formación necesaria en las tecnologías u otras necesidades del proceso de desarrollo.
3. **Desarrollo**: proceso de implementación del código fuente de la aplicación.
4. **Testing**: evaluación de las funcionalidades desarrolladas en la fase anterior para verificar tu correcto funcionamiento en diversos casos posibles.
5. **Documentación**: redactar y explicar la evolución del proyecto desde la idea inicial hasta la propia redacción de la memoria y preparar la presentación.

9.2. Descripción de tareas

Tabla 6: Tarea 1 - Definir proyecto

FASE	**ANÁLISIS**
Título	Definir proyecto
Descripción	Definir los requisitos y objetivos del proyecto a desarrollar
Duración (horas)	10h
Responsable(s)	Aingeru Palazuelo y Oskar Casquero
Recursos utilizados	*Brainstorming*
Entregas o resultados	Objetivos y requerimientos

Tabla 7: Tarea 2 - Estudio de mercado

FASE	**ANÁLISIS**
Título	Estudio de mercado
Descripción	Estudiar proyectos similares
Duración (horas)	15h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Google Play Store e Internet
Entregas o resultados	Estudio de mercado

Tabla 8: Tarea 3 - Análisis de alternativas

FASE	**ANÁLISIS**
Título	Análisis de alternativas
Descripción	Estudiar ventajas y desventajas de las diferentes alternativas
Duración (horas)	10h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Internet
Entregas o resultados	Análisis de alternativas

Tabla 9: Tarea 4 - Prototipado

FASE	**ANÁLISIS**
Título	Prototipado
Descripción	Diseño del prototipo de papel (*wireframe*) y prototipo digital (mockup)
Duración (horas)	15h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Papel y Cacoo (https://cacoo.com/)
Entregas o resultados	Wireframe y mockup

Tabla 10: Tarea 5 - Formación en Android

FASE	**FORMACIÓN**
Título	Formación en Android
Descripción	Aprender la base del desarrollo de aplicación para el SO Android: Developing Android Apps (by Google)
Duración (horas)	60h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	UDACITY (https://eu.udacity.com/) + Android Studio IDE
Entregas o resultados	Conocimiento base de la estructura y desarrollo de apps Android

Tabla 11: Tarea 6 - Formación en desarrollo de apps móviles

FASE	**FORMACIÓN**
Título	Formación en desarrollo de apps móviles
Descripción	Aprender las peculiaridades de los desarrollos específicos de software orientados a aplicaciones móviles
Duración (horas)	40h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Plataforma Google Activate (https://sites.google.com/site/moocappsmoviles/)
Entregas o resultados	Certificado por la UCM (Universidad Complutense de Madrid)

Tabla 12: Tarea 7 - Configuración del entorno de desarrollo

FASE	**DESARROLLO**
Título	Configuración del entorno de desarrollo
Descripción	Descargar, instalar y configurar las herramientas y recursos necesarios para el desarrollo: Android Studio, JAVA, GIT…
Duración (horas)	10h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Internet: web de cada recurso
Entregas o resultados	-

Tabla 13: Tarea 8 - Menú de navegación

FASE	**DESARROLLO**
Título	Menú de navegación de la app
Descripción	Implementar las pantallas y la navegación entre ellas en la app
Duración (horas)	20h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android
Entregas o resultados	Menú de navegación

Tabla 14: Tarea 9 - Activity Bluetooth

FASE	DESARROLLO
Título	Activity Bluetooth
Descripción	Implementar la actividad (interfaz y controlador) referente a las funcionalidades del Bluetooth: pedir permisos, habilitar el adaptador, escáner, conexión y transferencia.
Duración (horas)	25h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android
Entregas o resultados	Activity Bluetooth

Tabla 15: Tarea 10 - Conexión BT con emulador pulsera

FASE	DESARROLLO
Título	Conexión BT con el emulador de la pulsera (para sustituir la pulsera)
Descripción	Vincular el dispositivo móvil Android el emulador de la pulsera mediante la app HealthyBeat y recibir datos de ella.
Duración (horas)	15h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Pulsera (estación pletismográfica) + Dispositivo móvil Android
Entregas o resultados	Obtención de datos reales en la app

Tabla 16: Tarea 11 - Parsear señal

FASE	DESARROLLO
Título	Parsear señal
Descripción	Parsear los datos de la señal recibidos de la pulsera.
Duración (horas)	20h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android + Datos de la señal + Documentación OEM III
Entregas o resultados	Puntos PLETH de la señal y otros datos significativos (HR, SpO_2)

Tabla 17: Tarea 12 - Mostrar datos y gráfica

FASE	DESARROLLO
Título	Mostrar datos y gráfica
Descripción	Dibujar la gráfica de la señal con la librería para gráficos y mostrar los datos significativos por pantalla.
Duración (horas)	25h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android + Datos de la señal + Librería para gráficos
Entregas o resultados	Mostrar la gráfica construida con los datos PLETH de la señal y los datos significativos (HR, SpO_2)

Tabla 18: Tarea 13 - Guardar registro en la nube

FASE	DESARROLLO
Título	Guardar registro en la nube
Descripción	Guardar registro de la muestra en el servicio en la nube Google Drive.
Duración (horas)	10h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android + Datos de la señal + Cuenta de Google
Entregas o resultados	Registro de la muestra en Google Drive

Tabla 19: Tarea 14 - Pulir la app

FASE	DESARROLLO
Título	Pulir la app
Descripción	Crear un logo para la app, actualizar el gradle, comentar y optimizar el código…
Duración (horas)	15h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Designevo (https://www.designevo.com/)
Entregas o resultados	Código comentado, optimizado y logo de la app

Tabla 20: Tarea 15 - Pruebas

FASE	**TESTING**
Título	Pruebas
Descripción	Probar el funcionamiento correcto de las funcionalidades implementadas
Duración (horas)	15h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Android Studio + Dispositivo móvil Android + Pulsera + Cuenta de Google + Datos de la señal
Entregas o resultados	Código comentado, optimizado y logo de la app

Tabla 21: Tarea 16 - Redactar memoria

FASE	**DOCUMENTACIÓN**
Título	Memoria
Descripción	Redactar la memoria (documento actual).
Duración (horas)	25h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Editor de texto (Microsoft Word)
Entregas o resultados	Memoria (documento actual)

Tabla 22: Tarea 17 - Preparar presentación

FASE	DOCUMENTACIÓN
Título	Presentación
Descripción	Preparar la presentación del proyecto para la defensa.
Duración (horas)	10h
Responsable(s)	Aingeru Palazuelo
Recursos utilizados	Programa de presentaciones (Microsoft PowerPoint)
Entregas o resultados	Memoria (documento actual)

9.3. Estimación de horas

En la siguiente tabla se presenta la estimación de horas según las tareas descritas anteriormente:

Tabla 23: Estimación de horas

TAREA	ESTIMACIÓN DE HORAS
ANÁLISIS	**55h**
Definir proyecto	10h
Estudio de mercado	15h
Análisis de alternativas	10h
Prototipado	20h
FORMACIÓN	**100h**
Desarrollo de apps Android	60h
Desarrollo de apps móviles	40h
DESARROLLO	**140h**
Configurar entorno de desarrollo	10h
Menú de navegación	20h
Activity Bluetooth	25h
Conexión BT entre app y (emulador) pulsera	15h
Parsear señal	20h
Mostrar datos y dibujar gráfica	25h
Guardar registro en la nube	10h
Pulir App	15h
TESTING	**15h**
Pruebas	15h
DOCUMENTACIÓN	**35h**
Memoria	25h
Presentación	10h
TOTAL	**345h**

10. Diagrama Gantt

Tabla 24: Tabla de tareas

FASE	ACTIVIDAD	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	COMPLETADO (%)
1. ANÁLISIS	1.1. Definir el proyecto	1	2	1	2	100%
	1.2. Estudio de mercado	3	3	3	3	100%
	1.3. Análisis de alternativas	6	2	6	5	100%
	1.4. Prototipado	8	4	11	5	100%
2. FORMACIÓN	2.1. Android	12	12	16	20	50%
	2.2. Apps móviles	12	8	16	10	100%
3. DESARROLLO	3.1. Configurar entorno de desarrollo	12	2	16	3	100%
	3.2. Menú de navegación	24	4	26	4	100%
	3.3. Bluetooth	28	5	30	7	100%
	3.4. Conexión con pulsera	33	3	37	3	100 ([1])
	3.5. Parsear señal	36	4	37	3	100%
	3.6. Mostrar datos y gráfica	40	5	40	15	100%
	3.7. Guardar registro de medición	40	2	40	3	100%
	3.8. Otras tareas	45	3	55	2	100%
4. TESTING	4.1. Pruebas	48	3	57	3	80%
5. DOCUMENTACIÓN	5.1. Memoria y presentación	51	7	60	10	100%
TOTAL (días)		Previstos:	69	Reales:	94	96%
TOTAL (horas)		Previstas:	345	Reales:	469	

[1] Dada la avería de la pulsera, la conexión se ha efectuado contra el emulador de la pulsera

Diagrama Gantt con secuencia de tareas asignadas en la tabla superior. Cada columna representa una jornada de 5h.

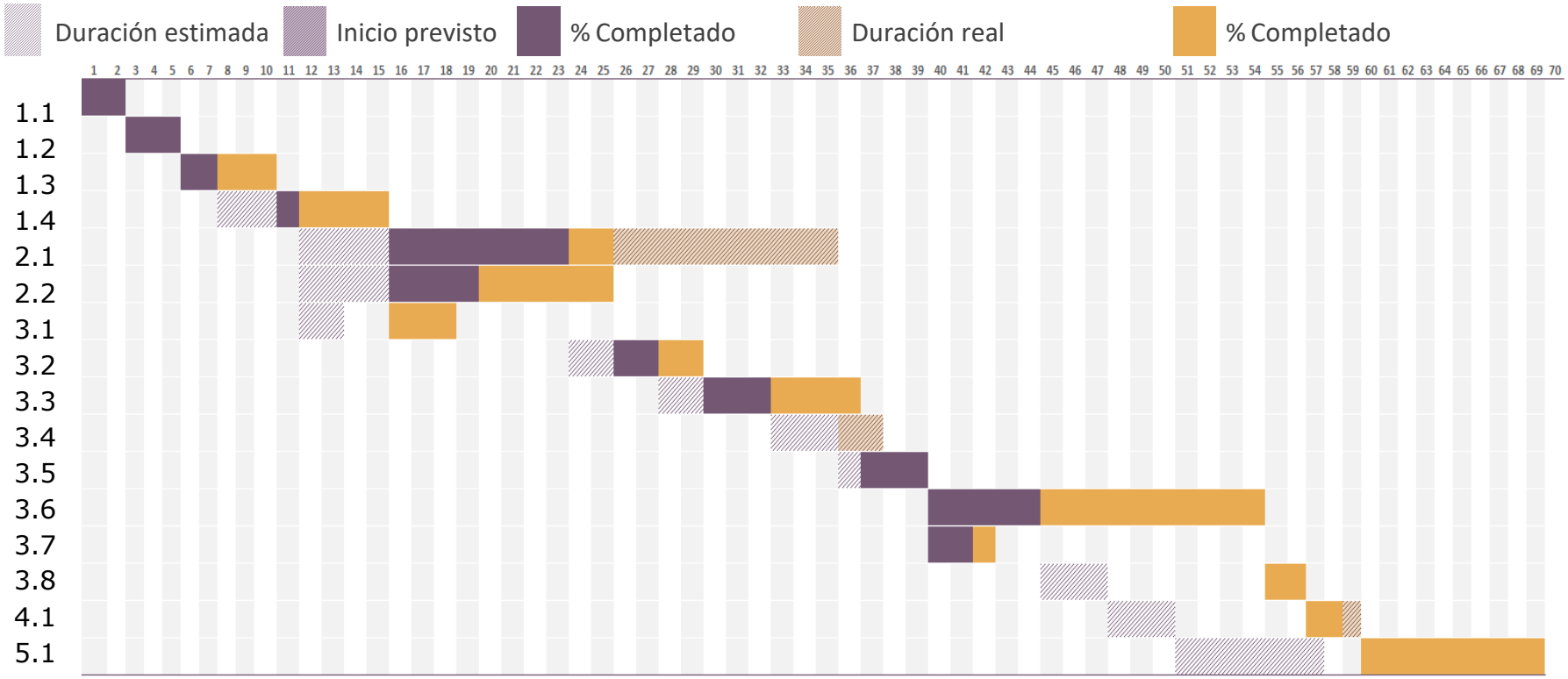


Ilustración 29: Diagrama Gantt

11. Diseño de bajo nivel

En esta sección se explica de forma más detallada y técnica el proceso de desarrollo que se ha seguido y el funcionamiento de los diferentes apartados del proyecto.

11.1. Wireframe

Antes de diseñar un prototipo digital que consume más tiempo, se ha esbozado una idea de lo que debería ser la actividad principal de la aplicación. De esta manera, en caso de surgir algún cambio se puede discutir y cambiar fácilmente.

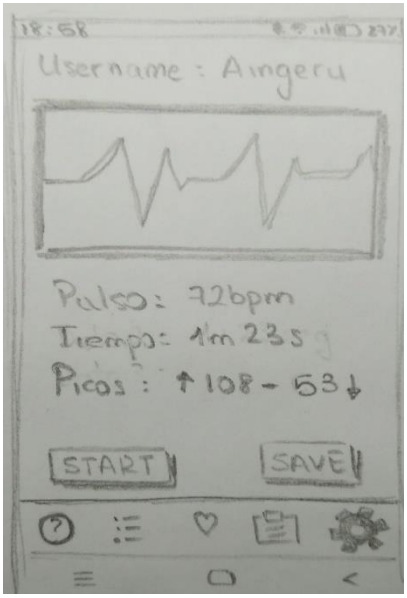


Ilustración 30: Wireframe

En la imagen anterior se puede ver la interfaz de lo que será el núcleo de la app: la actividad donde se mostrará el pulso y los datos recogidos y procesados.

En la mitad superior de la pantalla se renderizará el grafico del pulso y en la mitad inferior otros datos; como se aprecia en los ejemplos del boceto:

1. **Pulso:** dato HR
2. **Tiempo**: tiempo de medición desde el inicio del análisis.
3. **Picos**: máximos y mínimos (servirán para obtener el dato HRV que se explica más abajo).

Bajo estos datos se sitúan los botones para empezar (START) la medición y para guardarla (SAVE).

Y finalmente, pegado al borde inferior de la pantalla: el menú de navegación para navegar entre las pantallas de la app.

HEART RATE VARIABILITY (HRV):

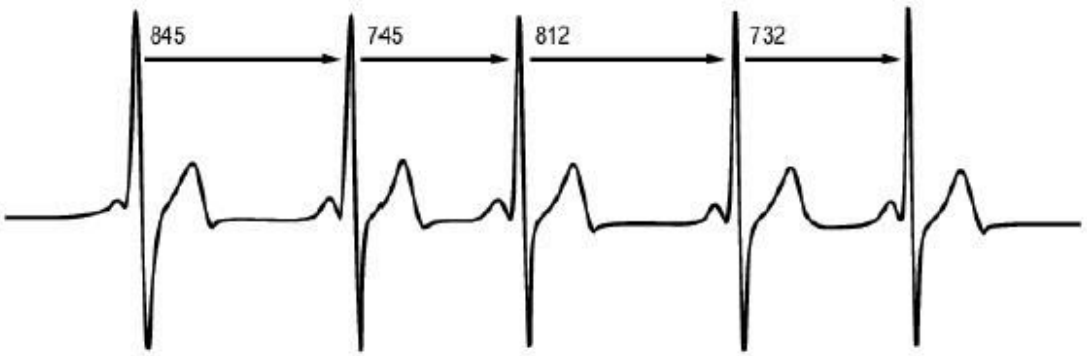


Ilustración 31: HRV

El Heart Rate Variability (HRV) o Varibilidad del Ritmo Cardíaco en castellano, es el intervalo de tiempo entre los latidos consecutivos medido en ms. Como se observa en la imagen anterior, no es constante, fluctúa dependiendo de diferentes factores, tales como: estado físico, psicológico, emocional… Del mismo modo también sirve para obtener indicativos de estos y aplicarlo para diferentes fines: desde controlar el estado físico para entrenamientos hasta detectar patrones que indique patologías (que es para lo que se busca aplicar en los pacientes con trastorno del sueño).

11.2. Mockup

Una vez aceptado el boceto anterior, se desarrolla un prototipo digital que requiere de más esfuerzo, pero da una imagen más ajustada al resultado final.

En este caso, se ha utilizado la web Cacoo (https://cacoo.com/).

Ilustración 32: Prototipo digital

El resultado es bastante similar al prototipo anterior. En este, aparece un indicador sobre la gráfica que indica que se está llevando a cabo el análisis. Además, también se ha cambiado el botón para que muestre el mensaje de "ANALIZAR" para comenzar con el proceso de obtención, procesado y muestra de datos y al mismo tiempo cambie su texto a "GUARDAR" para detener y guardar la muestra. Los datos que aparecen ahora bajo la gráfica son:

- **Pulso:** (el dato HR) los latidos por minuto como muestra la unidad ***lpm (latidos por minuto).***
- **SpO_2:** la oxigenación en sangre o saturación de oxígeno en sangre.

11.3. Casos de uso

En esta sección se determinará el caso de uso típico del usuario medio:

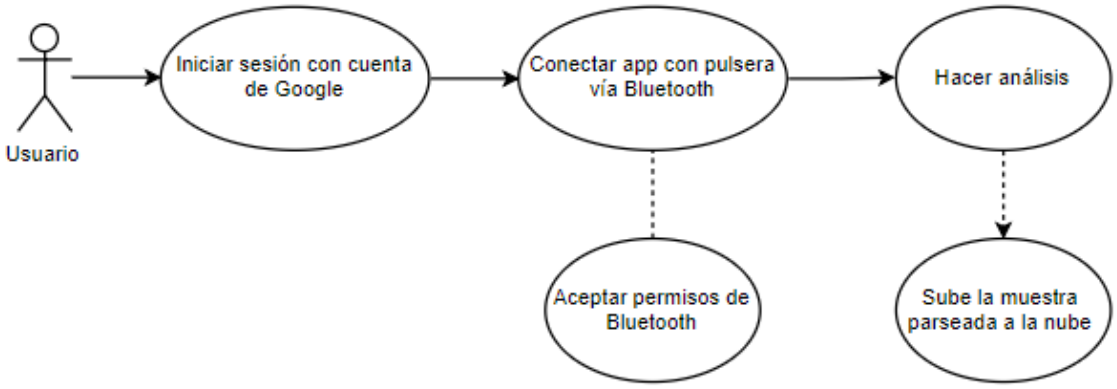


Ilustración 33: Caso de uso – Analizar

Dado que la funcionalidad principal de la app es hacer la medición o el análisis, estos serán los pasos a seguir para llegar a ese fin:

Siendo esta la primera vez que inicia la app:

1. Seleccionar cuenta de Google para posteriormente guardar los registros en Drive.
2. Ir a la pestaña de Bluetooth usando el menú de navegación inferior.
3. Aceptar los permisos para que la app pueda conectarse con otros dispositivos vía bluetooth (y activar el adaptador BT en caso de tener deshabilitado).
4. (*) Escanear en busca de dispositivos BT cercanos y conectarse con la pulsera.
5. (Una vez conectado) Volver a la pantalla principal mediante el menú.
6. Pulsar el botón de ANALIZAR para comentar el análisis.
7. Al detener el análisis pulsando DETENER, se subirá el registro de la muestra automáticamente a Drive.

(*) Dadas las circunstancias de la pulsera, el proceso será idéntico, pero se conectará al emulador de la pulsera.

11.4. Diagrama de clases

En el siguiente diagrama se muestran las clases que conforman las actividades JAVA de la app y la relación entre ellas.

Se ha generado en Android Studio mediante el plugin SimpleUMLCE (https://plugins.jetbrains.com/plugin/4946-simpleumlce).

Clases principales:

- **BaseActivity**
  Implementa: BottomNavigationMenu (Menú de navegación)
  - **Analizar**
    Hereda de: BaseActivity
    Implementa: BluetoothDataReceiver (BroadcastReceiver)
    - **BluetoothDataReceiver (Receiver)**
      Hereda de: BroadcastReceiver
  - **Bluetooth**
    Hereda de: BaseActivity
    Implementa: DeviceListFragment
    - **DeviceListFragment (Fragment)**
      Implementa: OnItemClickListener (Listener)
      - **DeviceListAdapter**
      - **DeviceItem**
  - **Archivo**
    Hereda de: BaseActivity
  - **Resultado**
    Hereda de: BaseActivity
  - **Perfil**
    Hereda de: BaseActivity
- **Aplicación**
  Hereda de: Aplication

11.4.1. BaseActivity

En la parte superior de la jerarquía de herencia, se sitúa la clase **BaseActivity** de la que heredan el resto de actividades principales, que serán las pantallas por las que el usuario navegue a través del menú de navegación. Esta clase tiene los atributos y métodos que las clases *hijo* hereda, en este caso sirve para que todas compartan el menú inferior y los métodos para comunicarse entre ellas.

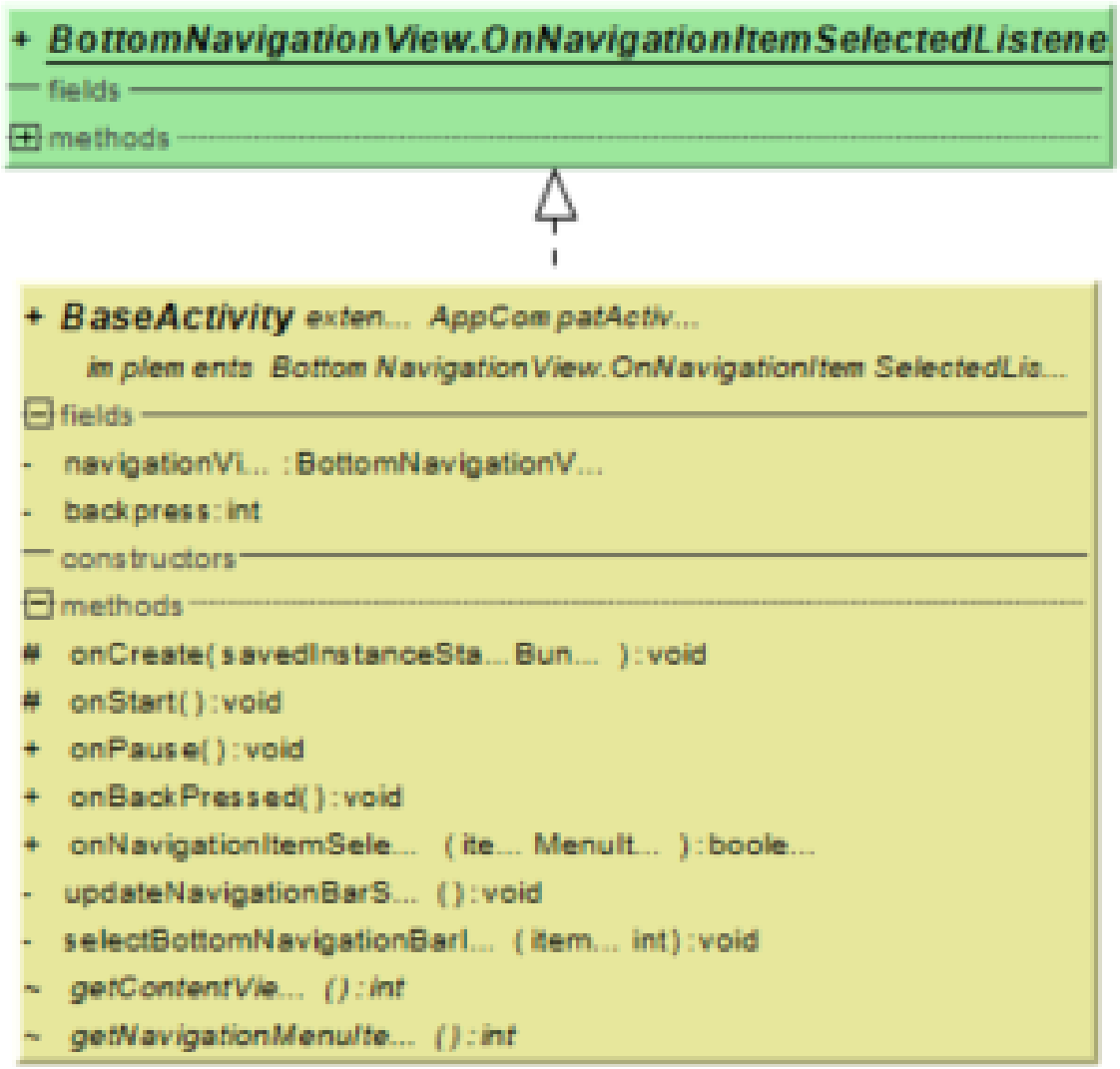


Ilustración 34: UML - BaseActivity

11.4.2. Analizar

Analizar es la actividad principal de la app. En esta actividad aparecen los ítems necesarios para mostrar e iniciar los datos del análisis: botones para iniciar/detener el análisis, la gráfica donde se renderiza el pulso y otros datos del análisis.

Esta a su vez llamará a **BluetoothDataReceiver** al pulsar el botón para iniciar el análisis. Este botón desencadenará de forma asíncrona un "BroadcastReceiver" que es un escuchador, en este caso el que recibirá los datos enviados por la pulsera vía Bluetooth. Y en esta misma clase es donde se parsearán esos datos y se procesarán mediante un algoritmo basado en la API del pletismógrafo (*OEM III Specifications 4518-001 Rev 11, ver Anexo I*) para finalmente mandar y renderizarlos de vuelta en la pantalla de **Analizar**.

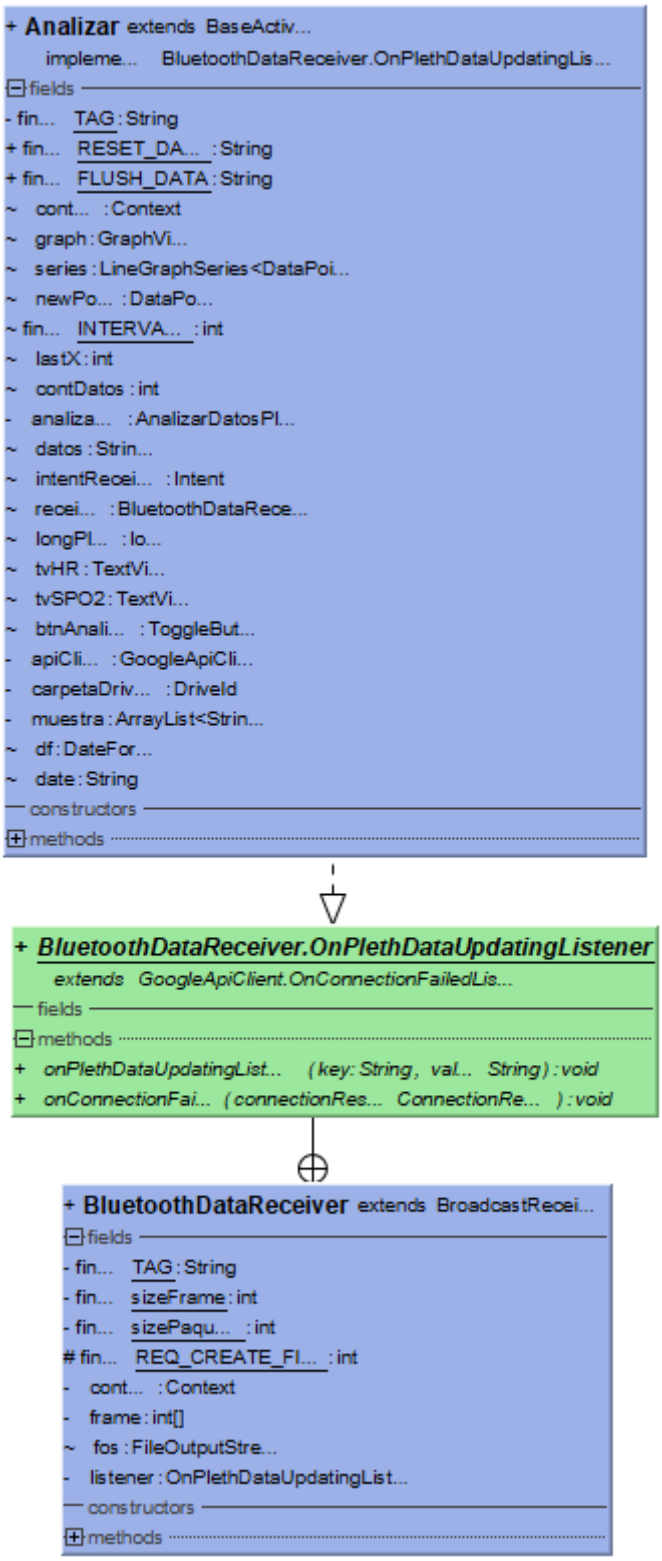


Ilustración 35: UML - Analizar

11.4.3. Bluetooth

La actividad **Bluetooth** es como su nombre indica, para gestionar la conexión vía Bluetooth con la pulsera (o el emulador de la pulsera en este caso). Al entrar por primera vez se le pedirá al usuario que acepte los permisos necesarios para que se pueda acceder al adaptador BT del dispositivo y poder activarlo/desactivarlo, hacer búsquedas, entablar conexiones y mandar/recibir datos por ella.

Como se observa en el diagrama, la clase implementa la interfaz **DeviceListFragment** puesto que va a contener un Fragment dentro de su vista para pintar la lista de dispositivos BT emparejados o encontrados en el escáner.

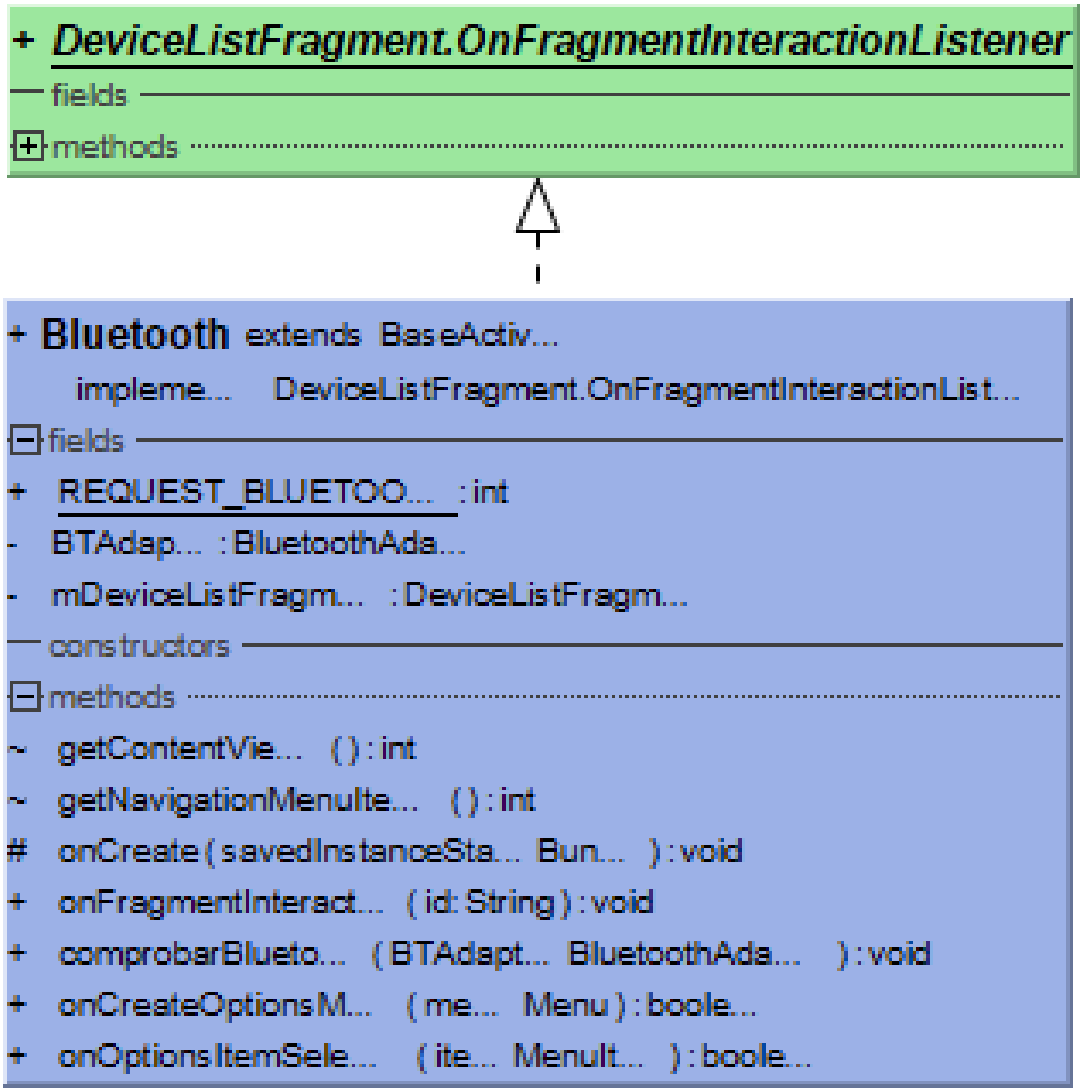


Ilustración 36: UML - Bluetooth

Desde esta activity se "inflará" (inflate) **DeviceListFragment**, un fragmento de la interfaz que se visualiza dentro de la vista de la actividad, para listar los dispositivos bluetooth con lo que establecer una conexión.

Este fragmento, dentro contiene un **DeviceListAdapter** que como su nombre indica es el componente para mostrar la lista de dispositivos (BT).

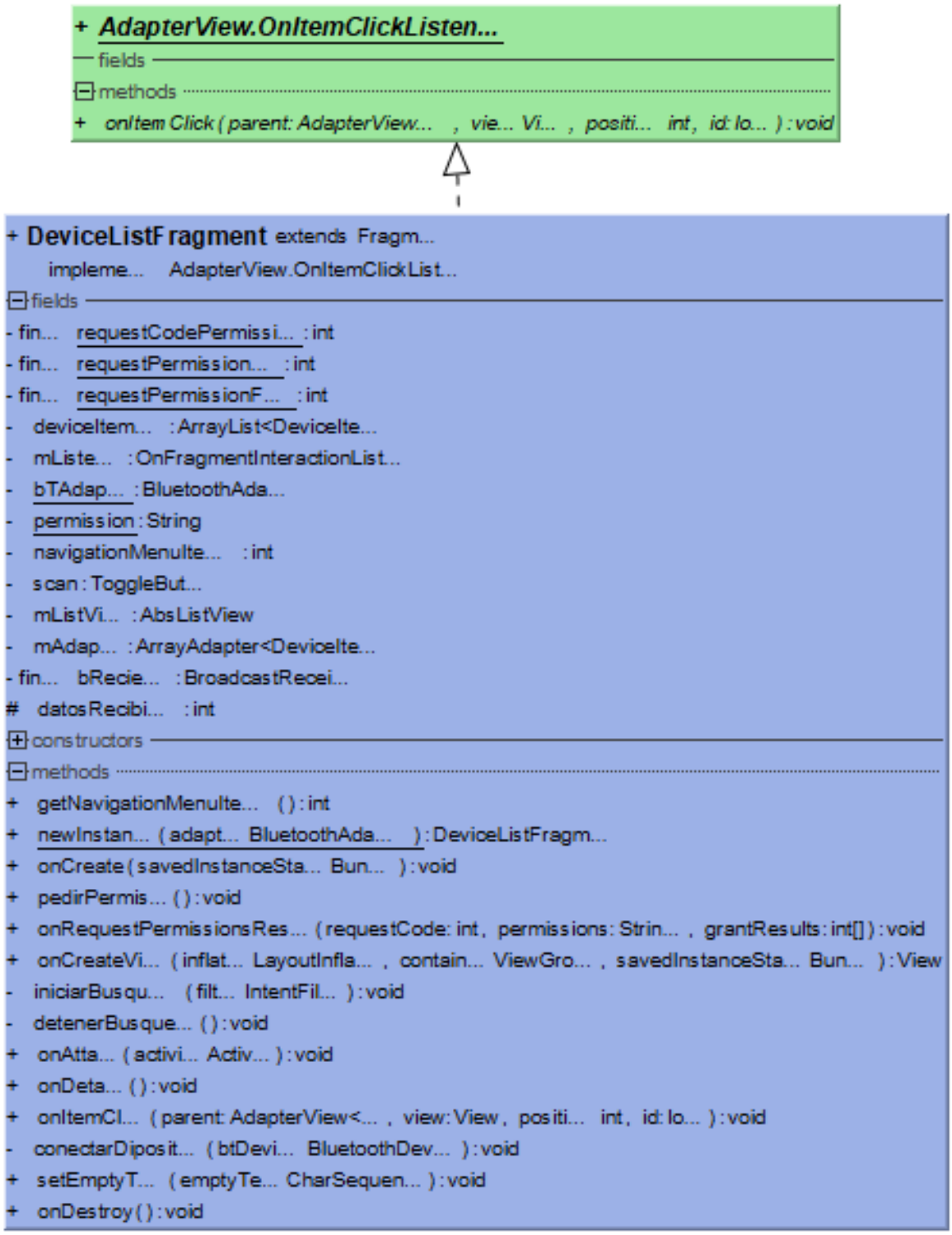
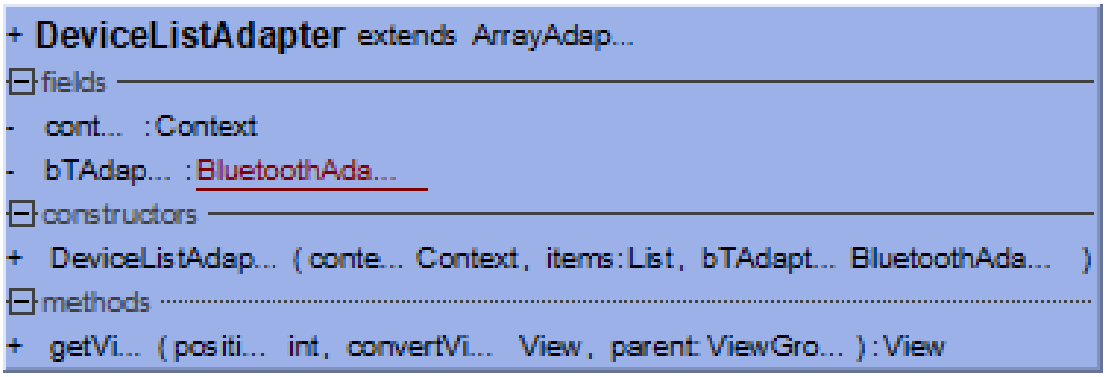


Ilustración 37: UML - DeviceListFragment

Dentro de esta, cada dispositivo se muestra mediante un **DeviceItem** con la información de cada dispositivo detectado vía Bluetooth: nombre y MAC.



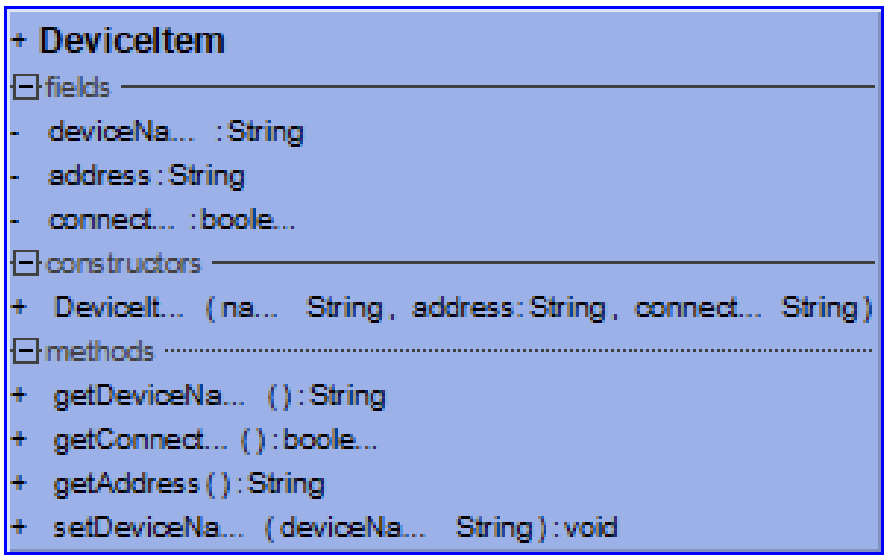


Ilustración 38: UML - DeviceItem

11.4.4. Otras

A parte de los anteriores, tenemos las actividades **Archivo, Resultado y Perfil** que no son más que el resto de pantallas a las que se viaja a través del menú, basadas en las pantallas que tiene el resto de aplicaciones probadas en el estudio de mercado, pero que actualmente no tiene funciones implementadas.

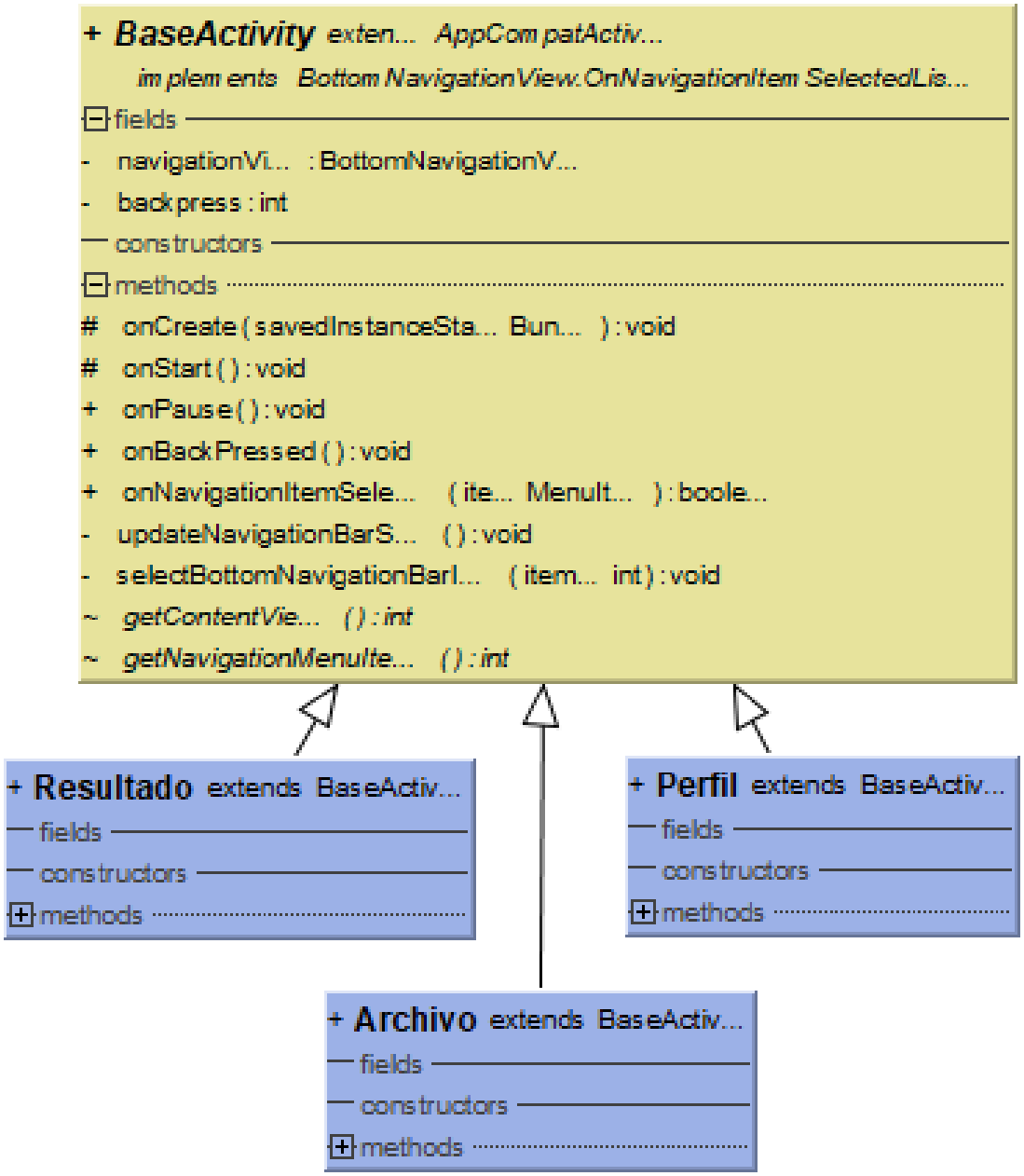


Ilustración 39: UML - Archivo | Resultado | Perfil

11.4.5. Aplicación

Por último, está **Aplicación** una clase hija de *Application (*similar a la clase *Object* de JAVA) en la que se usa para almacenar variables globales. En este caso, el BroadcastReceiver cargará de aquí atributos clave para el parseo de datos.

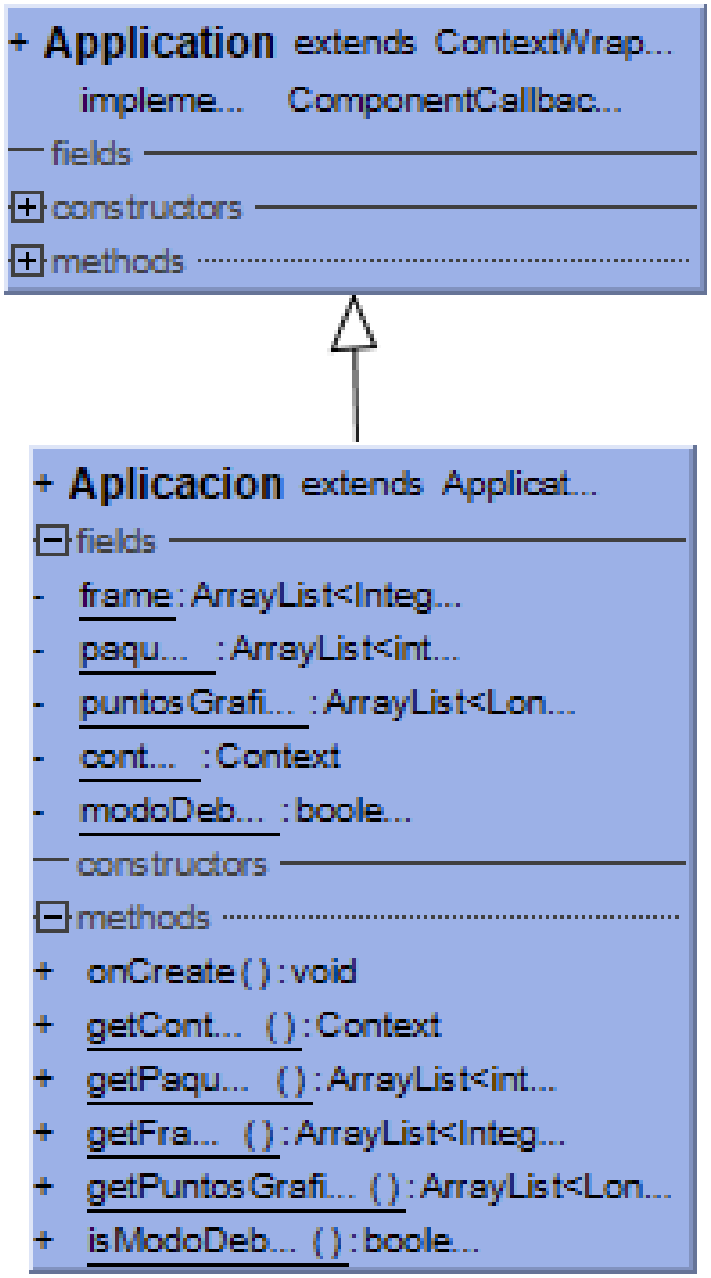


Ilustración 40: UML - Aplicacion

En la siguiente ilustración se muestra una visión general de las clases importantes estructura da una manera ordenada para mejorar su visualización:

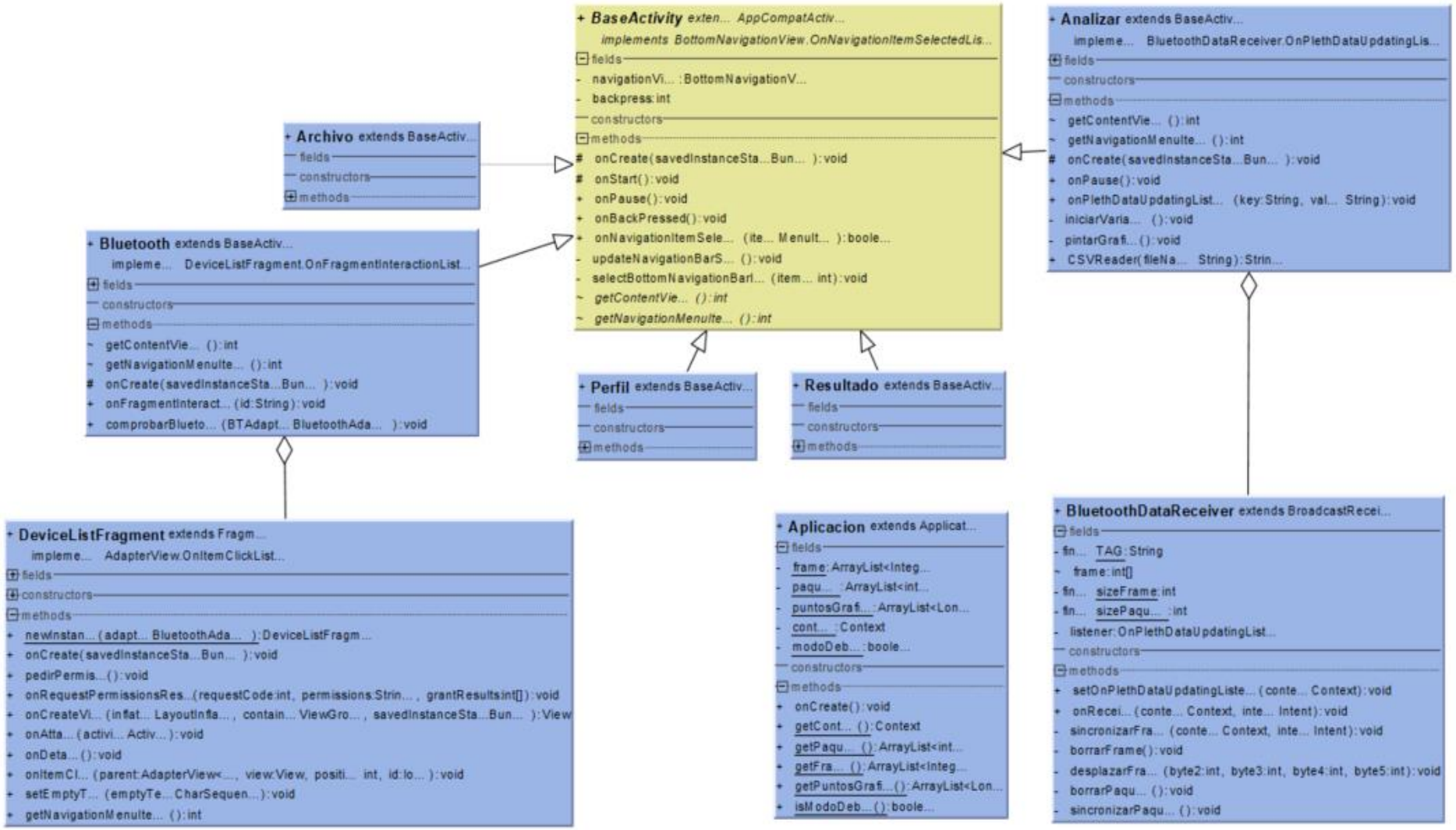


Ilustración 41: UML – Diagrama general

11.5. Diagramas de secuencia

En esta sección se muestran los diagramas de secuencia de llamadas entre clases y sus métodos, para visualizar el flujo de llamadas.

Se han creado en Android Studio a partir del plugin SequenceDiagram (https://plugins.jetbrains.com/plugin/8286-sequencediagram).

Estos son los procesos esenciales de la app

11.5.1. Bluetooth

En esta secuencia se muestra cómo funciona la actividad de Bluetooth:

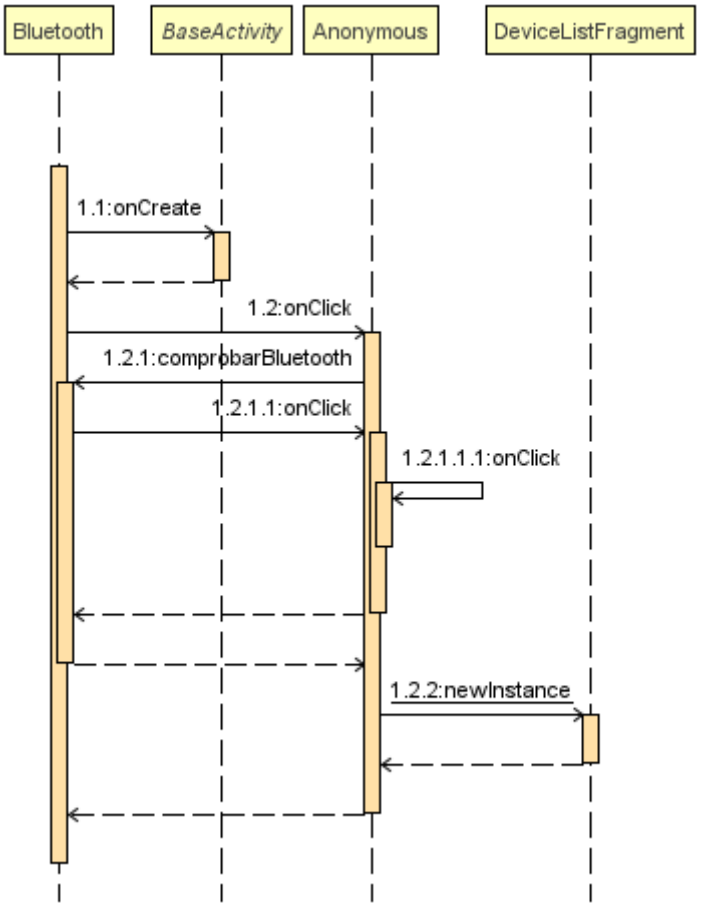


Ilustración 42: Secuencia – Pantalla Bluetooth

Lo primero a verificar al entrar por primera vez en la pantalla de Bluetooth, es si el dispositivo lo soporta:

```
// Verficar si el dispositivo soporta Bluetooth
if (BTAdapter == null) {
    // Device does not support Bluetooth
    new AlertDialog.Builder( context: this)
            .setTitle("Dispositivo no compatible")
            .setMessage("Este dispositivo no soporta Bluetooth")
            .setPositiveButton("Aceptar", (dialog, which) → {
                    System.exit( status: 0);
            })
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show();
}
```

Ilustración 43: Código - Verificar soporte de Bluetooth

En caso contrario la aplicación no podría completar su función, por ello se le informa al usuario y se cierra la aplicación.

La otra verificación se hace para detectar si el adaptador Bluetooth del dispositivo está encendido o no. En caso negativo se le informa al usuario mediante un mensaje y se muestra una petición para activarlo:

```
public void comprobarBluetooth(BluetoothAdapter BTAdapter) {

    if (!BTAdapter.isEnabled()) {

        new AlertDialog.Builder( context: this)
                .setTitle("Bluetooth desactivado")
                .setMessage("¿Desea activarlo?")
                // Si acepta se activara el Bluetooth una vez habilitado el permiso
                .setPositiveButton("Aceptar", (dialog, which) → {
                        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_R
                        startActivityForResult(enableBtIntent, REQUEST_BLUETOOTH);
                })
                // Si no acepta, se mostrará un mensaje
                .setNegativeButton("Cancelar", (dialog, which) → {
                        Toast toast = Toast.makeText(getApplicationContext(), R.stri
                        toast.setGravity( gravity: Gravity.CENTER | Gravity.CENTER,  xC
                        toast.show();
                })
                .setIcon(R.drawable.ic_bluetooth_black_24dp)
                .show();
    }
}
```

Ilustración 44: Código - Verifica adaptado BT

El siguiente paso es la petición de permisos al usuario:

```
public void pedirPermisos() {

    // Petición de permisos de localización, en caso de no haberlo aceptado anteriormente
    if(ContextCompat.checkSelfPermission(getActivity(), permission) != PackageManager.PERMISSION_GRANTED) {
        requestPermissions(new String[]{permission}, requestCodePermission);
    }
}

@Override
public void onRequestPermissionsResult (int requestCode, String[] permissions, int[] grantResults) {
    // Check which request we're responding to
    if (requestCode == requestCodePermission) {

        for (int i = 0, len = permissions.length; i < len; i++) {
            String permission = permissions[i];
            if (grantResults[i] == PackageManager.PERMISSION_DENIED) {
                // user rejected the permission
                boolean showRationale = shouldShowRequestPermissionRationale( permission );
                if (! showRationale) {
                    new android.app.AlertDialog.Builder(getActivity())
                        .setTitle("Información importante")
                        .setMessage("Debe aceptar los permisos para el correcto funcionamiento de la aplicac
                        .setPositiveButton("Aceptar", (dialog, which) → {
                                Toast.makeText(getActivity(), text: "Entre en Ajustes para aceptar los permis
                                System.exit( status: 0);
                        })
                        .setIcon(android.R.drawable.ic_dialog_info)
                        .show();
                } else if (Manifest.permission.ACCESS_COARSE_LOCATION.equals(permission)) {
                    new android.app.AlertDialog.Builder(getActivity())
                        .setTitle("Información importante")
                        .setMessage("Debe aceptar los permisos para el correcto funcionamiento de la aplicac
                        .setPositiveButton("Aceptar", (dialog, which) → {
                                dialog.cancel();
                                pedirPermisos();
                        })
                        .setNegativeButton("Cancelar", (dialog, which) → {
                                System.exit( status: 0);
                        })
                        .setIcon(android.R.drawable.ic_dialog_info)
                        .show();
                }
            }
        }
    }
}
```

Ilustración 45: Código - Pedir permisos BT al usuario

Al pasar las verificaciones del adaptador y permisos, se rellena la lista de dispositivos. Inicialmente con los dispositivos emparejados anteriormente (dispositivos conocidos), en caso de tener ninguno se muestra un ítem con el texto “No hay dispositivos”.

```
Set<BluetoothDevice> pairedDevices = bTAdapter.getBondedDevices();
if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
        DeviceItem newDevice= new DeviceItem(device.getName(), device.getAddress(), connected: "false");
        deviceItemList.add(newDevice);
    }
}

// En caso de no encontrar dispositivos, añadir un item que lo muestre.
if(deviceItemList.size() == 0) {
    deviceItemList.add(new DeviceItem( name: "No hay dispositivos",  address: "",  connected: "false"));
}

Log.d( tag: "DEVICELIST",  msg: "DeviceList rellenada\n");
mAdapter = new DeviceListAdapter(getActivity(), deviceItemList, bTAdapter);
Log.d( tag: "DEVICELIST",  msg: "Adapter creado\n");
```

Ilustración 46: Código - Rellenar lista de dispositivos BT

Al pulsar sobre uno de los dispositivos BT disponibles tras el escáner, se lanza el hilo para crear un canal RFCOMM, por el cual se recibirán los datos.

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    Log.d( tag: "DEVICELIST",  msg: "onItemClick position: " + position +
            " id: " + id + " name: " + deviceItemList.get(position).getDeviceName() + "\n");
    if (null != mListener) {
        // Notificiar a la interfaz padre (activity en este caso)
        // que se ha pulsado sobre un item
        mListener.onFragmentInteraction(deviceItemList.get(position).getDeviceName());
        // Vincular con el dispositivo pulsado
        BluetoothDevice btDevice = bTAdapter.getRemoteDevice(deviceItemList.get(position).getAddress());
        btDevice.createBond();
        conectarDipositivo(btDevice);
        // Detener el escanear una vez vinculado a un dispositivo
        if (bReciever != null) {
            detenerBusqueda();
        }
    }

}

private void conectarDipositivo(BluetoothDevice btDevice) {
    ConnectThread ct = new ConnectThread();
    if (ct.connect(btDevice, UUID.randomUUID() ) ) {
        BluetoothSocket btSocket = ct.getSocket();

        ManageConnectThread conThread = new ManageConnectThread();
        try {
            datosRecibidos = conThread.receiveData(btSocket);
            Log.d(TAG,  msg: "Datos recibidos vía BT: " + datosRecibidos);
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        Log.e(TAG,  msg: "No se ha podido crear el canal RFCOMM");
    }
}
```

Ilustración 47: Código - Crear conexión RFCOMM

A continuación, se muestra el flujo de manera visual:

Si es la primera vez que se entra en esta actividad, se pedirán los permisos correspondientes (imagen izquierda). En caso de rechazarlos, se le muestra al usuario un mensaje informando de que debe aceptarlos para el correcto funcionamiento de la aplicación (imagen central).

Ilustración 48: Pantalla Bluetooth - Pedir permisos

Ilustración 49: Pantalla Bluetooth - Mensaje sobre permisos BT

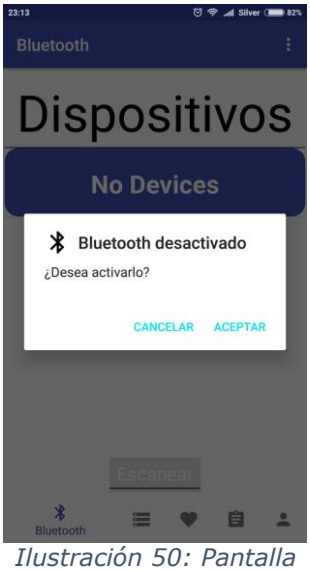


Ilustración 50: Pantalla Bluetooth - Activar BT

Aunque no se dé el caso anterior, si al entrar en esta pantalla el adaptador BT no está habilitado, se le informa al usuario y se muestra un pop-up para activarlo (imagen derecha).

Ilustración 51: Pantalla Bluetooth

Se crear un por cada dispositivo Bluetooth cercano que se haya encontrado y asigna la información de los dispositivos: nombre y MAC. Y se muestran los dispositivos en la lista del **DeviceListAdapter**.

En la siguiente secuencia se puede observar como al pulsar sobre el botón ESCANEAR de la pantalla Bluetooth, se cambiar el estado de este *ToggleButton*y compieza la búsqueda. Esta búsqueda se detiene por los siguientes motivos:

- Pulsar el mismo botón para detener la búsqueda.
- Pulsar en alguno de los dispositivos BT encontrados.
- Erros: salir de la pantalla, salir de la aplicación u otros errores no contemplados.

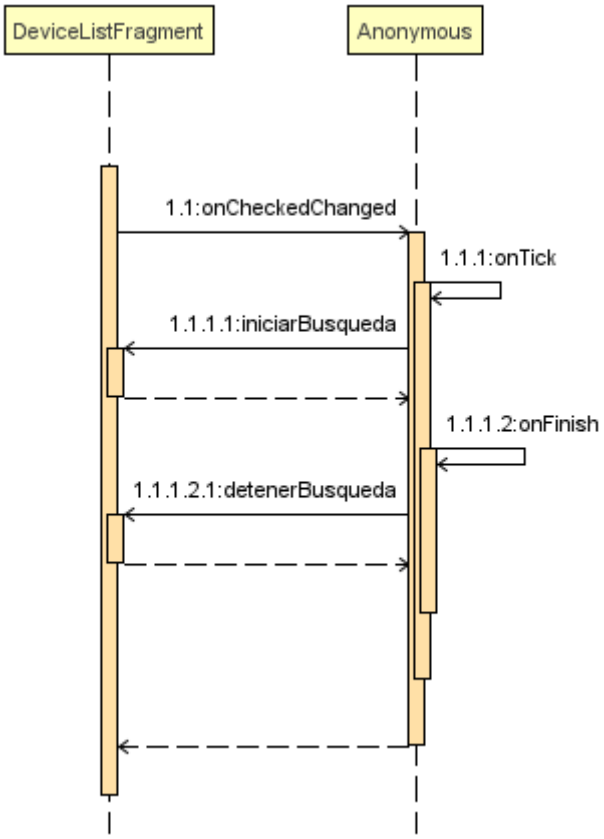


Ilustración 52: Secuencia - Escáner BT

11.5.2. Parsear y mostrar los datos de la señal

Esta secuencia muestra el flujo de parseo, procesado y devolución de datos a la vista Analizar para mostrar datos y pintar el gráfico en tiempo real:

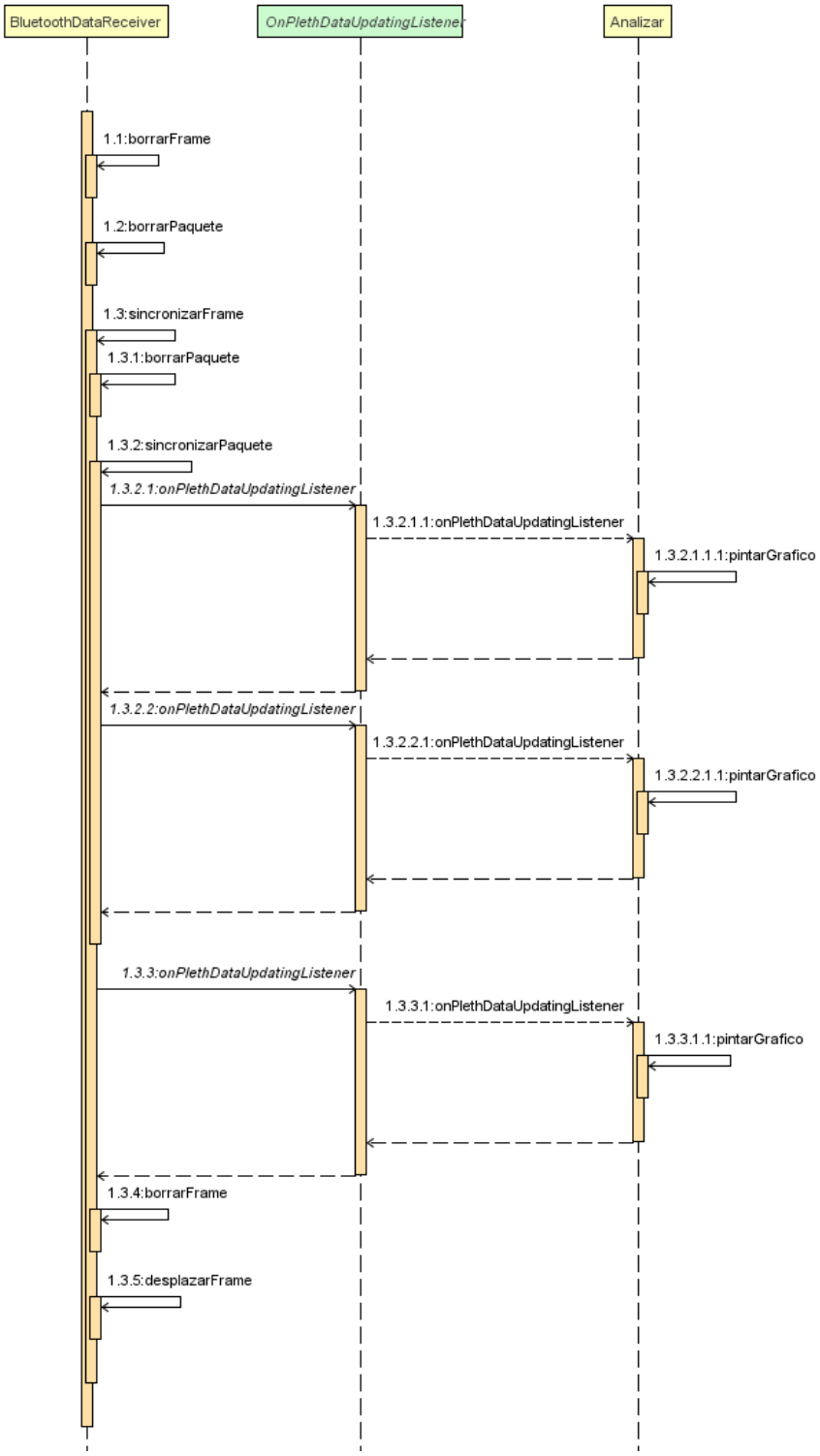


Ilustración 53: Secuencia - Parsear datos

Al llegar a esta clase se resetean las estructuras de datos para almacenar y manipular los frame y paquetes que más adelantes se explican.

```
// Empezar cuando el ArrayList de Frame tenga 5 valores
if (Aplicacion.getFrame().size() == sizeFrame) {
    int byte1 = Aplicacion.getFrame().get(0);
    int byte2 = Aplicacion.getFrame().get(1);
    int byte3 = Aplicacion.getFrame().get(2);
    int byte4 = Aplicacion.getFrame().get(3);
    int byte5 = Aplicacion.getFrame().get(4);
```

Ilustración 54: Código - Rellenar Frame

Una vez que el array de Frame de 5 posiciones se rellena, se llama al método **sincronizarFrame** para sincronizarse a nivel de frame.

```
// CHK: Si el resto de la suma de los 4 primeros bytes modulo 256,
// es igual al 5.byte: estamos sincronizados a nivel de Frame
if ((byte1+byte2+byte3+byte4) % 256 == byte5) {
    frame = new int[]{byte1, byte2, byte3, byte4, byte5};
    // Sí el 1.byte del Frame es '129' o '131': estamos sincronizados a nivel de Paquete
    if((byte1 == 129 || byte1 == 131) || Aplicacion.getPaquete().size() >= 1) {
        // Si ya hemos procesado un paquete
        if ((byte1 == 129 || byte1 == 131) && Aplicacion.getPaquete().size() == sizePaquete) {
```

Ilustración 55: Código - Comprobar CHK Frame

Cuando se logra este nivel de sincronización se llama al método **sincronizarPaquete** para hacer lo propio, pero a nivel de paquete. algoritmo para sincronizar los datos a nivel de paquete
(En este punto ya tenemos una estructura de datos sincronizada, como la que se muestra en la documentación OEM III del Anexo 1, un paquete).

En este punto se obtiene los datos que se van a pintar por pantalla. Por un lado, se calcula el punto *pleth* (punto del gráfico) que viene separado en 2 byte, y se transmite a la vista para pintarlo. La frecuencia de transmisión de este dato es muy hasta, dado que en cada paquete viene 25 datos de este tipo, uno por cada frame.

```
// Devolver los datos Pleth (Byte2 y Byte3) para mostrar el HRV en UI
// Juntar los 2 Bytes de Pleth y mostrarlos en la UI
long puntoPleth = (frame[1]*256) + frame[2];

// Mandar dato pleth para dibujar el gráfico a la UIThread mediante Listener
if (listener != null) {
    listener.onPlethDataUpdatingListener("Pleth", String.valueOf(puntoPleth));
}
```

Ilustración 56: Código - Calculo del punto del gráfico (PLETH) y mandarlo a la vista con el listener

El resto de datos se calculan siguiendo la documentación de OEM III

```
if (Aplicacion.getPaquete().size() == sizePaquete) {
    // Devolver los datos significativos (HR, SpO2...) para mostrar en UI
    int HR_MSB = Aplicacion.getPaquete().get(0)[3];
    int HR_LSB = Aplicacion.getPaquete().get(1)[3];
    long HR = HR_MSB*256 + HR_LSB;
    int SPO2 = Aplicacion.getPaquete().get(2)[3];
    // CHANGES: No se muestran en UI
    /*int E_HR_MSB = Aplicacion.getPaquete().get(13)[3];
    int E_HR_LSB = Aplicacion.getPaquete().get(14)[3];
    long E_HR = E_HR_MSB*256 + E_HR_LSB;
    int E_SPO2 = Aplicacion.getPaquete().get(15)[3];*/
```

Ilustración 57: Código - Calculo del resto de datos de la señal

Cuando se devuelven los datos al controlador principal, este gestiona que hacer con cada uno, es decir, los datos PLETH (puntos del gráfico) los añade a la lista de punto para dibujar y con el resto de datos actualizar el valor que se muestra:

```
// Método para actualizar la UI con los datos procesados en BluetoothDataReceiver
@Override
public void onPlethDataUpdatingListener(String key, String value) {

        switch (key) {
            /*
            Para guardar todos los datos en crudo
            case "Dato":
                //Log.d(TAG, "Dato: " + value);
                muestra.add(value);
                break;*/
            case "Pleth":
                // Añadir a muestra para subir a Drive
                muestra.add(value);
                // Dibujar gráfico
                if ( !Aplicacion.getPuntosGrafico().isEmpty() ) {
                    if(Aplicacion.isModoDebug()) {
                        longPleth = Long.parseLong(value);
                        // tvPleth.setText(String.valueOf(longPleth));
                        Log.d(TAG,  msg: "PLETH: " + longPleth);
                    }
                    pintarGrafico();
                }
                break;
            case "HR":
                tvHR.setText(value);
                Log.d(TAG,  msg: "HR: " + value);
                break;
            case "SPO2":
                tvSPO2.setText(value);
                Log.d(TAG,  msg: "SPO2: " + value);
                break;
        }
}
```

Ilustración 58: Código - Mostrar datos por pantalla

Para añadir un nuevo punto a dibujar en el gráfico, simplemente hay que hacer uso de la API de GraphView y añadirlo mediante el método *appendData* al objeto *series* que representa la lista de puntos del gráfico.

```
// Metodo para insertar los nuevos puntos del eje Y
private void pintarGrafico() {

    long graphPoint = Aplicacion.getPuntosGrafico().remove( index: 0);
    series.appendData(new DataPoint(lastX, graphPoint), scrollToEnd: true, maxDataPoints: 1000);
    lastX++;
}
```

Ilustración 59: Código - Pintar nuevos puntos en el gráfico

Previamente, hemos debido añadir las dependencias de la librería en el Gradle → build.gradle (Module: app)

```
dependencies {

    ...

    // GraphView dependency
    implementation 'com.jjoe64:graphview:4.2.2'

}
```

Ilustración 60: Código - Dependencias GraphView

ESTILOS DEL GRÁFICO GRAPHVIEW

Estos son los estilos definidos en la app, pero se puede personalizar siguiendo la API: http://www.android-graphview.org/style-options/

```
// ---------------------------------- GRAPHVIEW ---------------------------------------- //
// GraphView: editar aspecto del gráfico en UI
graph = (GraphView) findViewById(R.id.graph);
// VIEWPORT: Segmento de datos que se ve por pantalla
graph.getViewport().setMaxX(200);                       // MAX: Rango de 200 datos en el EJE X (default=200)
graph.getViewport().setXAxisBoundsManual(true);         // Calcula los límites del eje X automáticamente (default=true)
graph.getViewport().setScalable(false);                 // No activar (default=false)
graph.getViewport().setScrollable(true);                // Activar el scroll horizontal (default=true)
graph.getViewport().setBackgroundColor(TRANSPARENT);    // Color de fondo (default=TRANSPARENT)
// GRIDLABELRENDERER: Estilos de la cuadricula que se usa como canvas (o View) del gráfico

/* Estilo de las rejillas de fondo
    GridLabelRenderer.GridStyle.
    NONE: Ninguna
    HORIZONTAL
    VERTICAL
    BOTH: Horizontal y Vertical
*/
if (Aplicacion.isModoDebug()) {
    graph.getGridLabelRenderer().setGridStyle(GridLabelRenderer.GridStyle.BOTH);
    graph.getGridLabelRenderer().setHorizontalLabelsVisible(true);
    graph.getGridLabelRenderer().setVerticalLabelsVisible(true);
    graph.getGridLabelRenderer().setHighlightZeroLines(true);       // Remarcar ejes en el valor 0
} else {
    graph.getGridLabelRenderer().setGridStyle(GridLabelRenderer.GridStyle.HORIZONTAL);
    graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);
    graph.getGridLabelRenderer().setVerticalLabelsVisible(false);
    graph.getGridLabelRenderer().setHighlightZeroLines(false);      // Remarcar ejes en el valor 0
}

// ---------------------------------- END GRAPHVIEW ------------------------------------ //
```

Ilustración 61: Código - Definir estilos GraphView

11.5.3. Subir muestra a Google Drive

En la siguiente secuencia se muestra el proceso para guardar el registro de la muestra en el servicio en la nube de Google Drive:

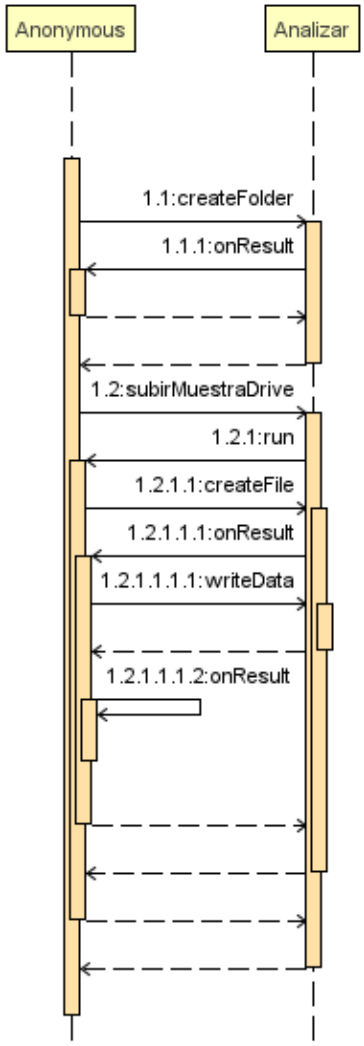


Ilustración 62: Código - Secuencia para subir muestra a Google Drive

Después de seguir los pasos para crear un proyecto en la consola de desarrolladores de Google y añadir las respectivas dependencias para el servicio Drive:

```
dependencies {

    ...

    // Servicios de Google: Drive
    implementation 'com.google.android.gms:play-services-drive:15.0.1'
}
```

Ilustración 63: Código - Dependencia Google Drive

Lo primero es crear la apiClient:

```
// Crear DRIVE API CLIENT
apiClient = new GoogleApiClient.Builder(context)
        .enableAutoManage( fragmentActivity: this,  onConnectionFailedListener: this)
        .addApi(Drive.API)
        .addScope(Drive.SCOPE_FILE)
        /* Para manejar los errores manualmente
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        */
        .build();
```

Ilustración 64: Código - Crear GoogleApiClient para Drive

Sólo con esto, al iniciar la aplicación, se mostrará una ventana para seleccionar la cuenta de Google en la que el usuario desea guardar los registros (imagen izquierda). Después de seleccionar una de ella, se deberán aceptar los permisos de acceso a la cuenta (imagen derecha). En este caso, acceso a los archivos y carpetas de Drive.

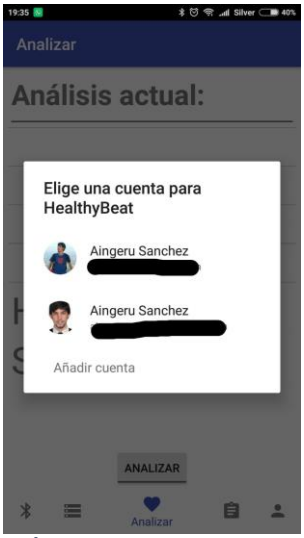


Ilustración 65: Seleccionar cuenta de Google

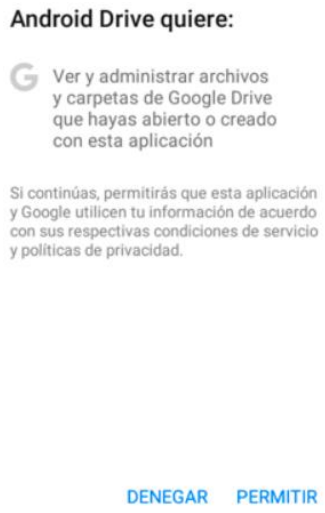


Ilustración 66: Permisos para Drive

En el siguiente *snippet*[1] se muestra la creación de la carpeta en el directorio raíz de Drive:

[1] **Snippet:** fragmento de código.
Véase: https://es.wikipedia.org/wiki/Snippet

```
private void createFolder(final String foldername) {

    MetadataChangeSet changeSet =
            new MetadataChangeSet.Builder()
                    .setTitle(foldername)
                    .build();

    // Directorio raíz de Google Drive
    DriveFolder folder = Drive.DriveApi.getRootFolder(apiClient);

    // Opción 2: Carpeta de Aplicación (App Folder)
    //DriveFolder folder = Drive.DriveApi.getAppFolder(apiClient);

    folder.createFolder(apiClient, changeSet).setResultCallback(
            (ResultCallback) (result) → {
                    if (result.getStatus().isSuccess()) {
                        Log.i(TAG,  msg: "Carpeta creada con ID = " + result.getDriveFolder().getDriveId());
                        carpetaDriveId = result.getDriveFolder().getDriveId();
                    } else {
                        Log.e(TAG,  msg: "Error al crear carpeta");
                    }
            });
}
```

Ilustración 67: Código - Crear fichero en Drive

Se debe guardar el ID de creación de la carpeta, para poder crear el archivo dentro de esta y no de nuevo en la raíz del directorio.

Para crear un nombre de carpeta identificativo concatenado la fecha del día que se ha tomado la muestra:

```
// Al pulsar DETENER:
// Guadar muestra en Google Drive
createFolder( foldername: "Muestra Healthybeat " + date);
subirMuestraDrive();
```

Ilustración 68: Código: nomenclatura de capetas de registros

Ilustración 69: Carpetas de muestras en Drive

Como se detalla más arriba, es al finalizar el análisis cuando se genera el fichero de la muestra parseada que se va a subir a Drive:

```
private void createFile(final String filename) {

    Drive.DriveApi.newDriveContents(apiClient)
            .setResultCallback((ResultCallback) (result) → {
                if (result.getStatus().isSuccess()) {

                    writeData(result.getDriveContents());

                    MetadataChangeSet changeSet =
                            new MetadataChangeSet.Builder()
                                    .setTitle(filename)
                                    .setMimeType("text/plain")
                                    // .setMimeType("text/excel")
                                    .build();

                    // Carpeta creada al inicio del muestreo
                    DriveFolder folder = carpetaDriveId.asDriveFolder();

                    folder.createFile(apiClient, changeSet, result.getDriveContents())
                            .setResultCallback((ResultCallback) (result) → {
                                if (result.getStatus().isSuccess()) {
                                    Log.i(TAG,  msg: "Fichero creado con ID = " + result.getDriveFile().getDriveId());
                                } else {
                                    Log.e(TAG,  msg: "Error al crear el fichero");
                                }
                            });
                } else {
                    Log.e(TAG,  msg: "Error al crear DriveContents");
                }
            });
}
```

Ilustración 70: Código - Crear fichero de la muestra en Drive

Para la nomenclatura del fichero se ha seguido el mismo método que con la carpeta. En la siguiente línea de código es en la que definimos la capeta en la que se guardará la muestra, que es una variable con el ID generado al crear la capeta:

```
// Carpeta creada al inicio del muestreo
DriveFolder folder = carpetaDriveId.asDriveFolder();
```

Ilustración 71: Código - Carpeta de la muestra

Finalmente se ejecuta la subida a Drive en un hilo asíncrono para no bloquear el hilo principal (*UI*[1] *Thread*[2]):

```
// ------------------ API GOOGLE DRIVE --------------------
private void subirMuestraDrive() {
    new Thread() {
        @Override
        public void run() {
            createFile( filename: "Muestra_" + date);
        }
    }.start();
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Toast.makeText(context,  text: "Error de conexión", Toast.LENGTH_SHORT).show();
    Log.e(TAG,  msg: "OnConnectionFailed: " + connectionResult);
}
```

Ilustración 72: Código - subir muestra a Drive de manera asíncrona

[1] **UI:** *User Interface* o Interfaz de Usuario.
Véase: https://es.wikipedia.org/wiki/Interfaz_de_usuario

[2] **Thread:** en el ámbito de la programación, se refiere a los hilos de ejecución de la aplicación.
Véase: https://es.wikipedia.org/wiki/Hilo_(informática)

11.6. Algoritmo de parseo de señal

11.6.1. Explicación textual

SINCRONIZAR DATOS:

Este algoritmo está basado en la API del pletismógrafo (*OEM III Specifications 4518-001 Rev 11, ver Anexo I*), en este caso concreto, para el *Serial Data Format #7:*

Serial Data Format #7:

Packet Description

A frame consists of 5 bytes; a packet consists of 25 frames. Three packets (75 frames) are transmitted each second.

Frame

Packet

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
1	STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK
2	STATUS	PLETH (MSB)	PLETH (LSB)	HR LSB	CHK
3	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2	CHK
4	STATUS	PLETH (MSB)	PLETH (LSB)	REV	CHK
5	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
6	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
7	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
8	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
9	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2-D	CHK
10	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 Fast	CHK
11	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 B-B	CHK
12	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
13	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
14	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR MSB	CHK
15	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR LSB	CHK
16	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2	CHK
17	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2-D	CHK
18	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
19	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
20	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D MSB	CHK
21	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D LSB	CHK
22	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D MSB	CHK
23	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D LSB	CHK
24	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
25	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK

Notes:
- Refer to DF#2 for definitions on Status Byte, Byte 4 and Checksum
- PLETH(MSB) and PLETH(LSB) make up a 16-bit pleth (ex. PLETH(MSB)*256 + PLETH(LSB))

Ilustración 73: Documentación OEM III - Serial Data Format #7

Siguiendo la documentación OEM III se pueden *parsear* los datos que llegan desde el pletismógrafo siguen esta estructura:

- *BYTE* 1 → **STATUS**
- *BYTE* 2 y 3 → **PLETH**
- *BYTE* 4 → HR, SpO_2… y otros datos significativos de la medición
- *BYTE* 5 → **CHK**

Como describe este formato de datos, esa matriz de datos denominada "**paquete**" consta de 25 "***frames***" (o fila) y se envían 3 de estos cada segundo vía Bluetooth. La dificultad en el parseado de datos consta en que el primer dato que recibas o

procesos no tiene por qué ser el 1. del paquete ni del frame. Por lo tanto, este es el procedimiento a seguir para sincronizar la llegada de datos a nivel de frame y posteriormente a nivel de paquete:

SINCRONIZAR DATOS A NIVEL DE **FRAME**:

Para este propósito se envía el *BYTE* 5 (**CHK**) o *Checksum*. Si el módulo^{256} de la suma de los 4 byte anteriores es igual al 5º byte, esto significa que este 5. byte es el CHK y por tanto que el conjunto de los 5 byte representa una fila o frame.

Una vez asegurado que podemos leer los datos en conjuntos de 5 o a nivel de frame, el siguiente paso es sincronizar a nivel de paquete:

SINCRONIZAR DATOS A NIVEL DE **PAQUETE**:

El primer byte de **STATUS** de cada paquete es diferente, y podemos saber cuál de ellos de nuevo gracias a la documentación OEM III:

Byte 2 - Status							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
1	SNSD	ARTF	OOT	SNSA	YPRF		SYNC
					RPRF	GPRF	
*Note: Bit 7 is always set.							

The following are all active high:

SNSD:	Sensor Disconnect	– Sensor is not connected to oximeter or sensor is inoperable.
ARTF:	Artifact	– A detected pulse beat didn't match the current pulse interval
OOT:	Out Of Track	– An absence of consecutive good pulse signals
SNSA:	Sensor Alarm	– Sensor is providing unusable data for analysis
RPRF:	Red Perfusion	– Amplitude representation of low signal quality (occurs only during pulse)
YPRF:	Yellow Perfusion	– Amplitude representation of medium signal quality (occurs only during pulse)
GPRF:	Green Perfusion	– Amplitude representation of high signal quality (occurs only during pulse)
SYNC:	Frame Sync	(occurs 1 of 25)

Ilustración 74: Documentación OEM III – Status

Se puede conocer el significado de cada *BIT* viendo la ilustración anterior, pero para lo que nos ocupa, vamos a centrarnos en el *BIT* 0. Siendo el resto variables (excepto el BIT7 que es constante), este último BIT es el que tiene el valor 1 cuando es el 1º byte del paquete, siendo la diferencia respecto a los siguientes BYTE de estados en 1 unidad.

Ejemplo:

*70,**128**,117,43,0,32,**128**,116,195,96,23,**128**,116,124,96,208,**128**,116,86,96,170,**128**,116,74,0,62,**128**,116,86,0,74,**128**,116,117,0,105,**128**,116,162,73,223,**128**,116,216,96,44,**128**,117,16,96,101,**128**,117,71,0,60,**128**,117,122,0,111,**128**,117,167,0,156,**128**,117,201,72,6,**128**,117,225,0,214,**128**,117,238,74,45,**128**,117,239,0,228,**128**,117,228,0,217,**129**,117,210,0,200,**128**,117,188,70,247,**128**,117,164,96,249,**128**,117,136,45,170,**128**,117,105,96,190,**128**,117,71,0,60,**128**,117,34,0,23,**128**,117,4,0,249,**128**,116,253,96,81,**128**,117,58,96,143,**128**,118,11,96,97,**128**,119,217,0,208,**128**,122,253,0,247,**128**,127,147,0,146,**128**,133,88,73,166,**128**,139,181,96,32,**128**,145,225,96,82,**128**,151,35,0,58,**128**,155,2,0,29,**128**,157,90,0,119,**128**,158,72,72,174,**128**,158,12,0,42,**128**,156,239,74,85,**128**,155,57,0,84,**130**,153,42,0,69,**131**,150,243,0,12,**130**,148,181,68,15,**130**,146,**130**,96,246,**130**,144,87,45,150,**130**,142,49,96,161,**130**,140,22,0,36,**130**,138,26,0*

Color negro: BYTE STATUS (1. Dato del *frame*)
En negrita: 1. STATUS del paquete
Subrayado: paquete completo y ordenado
Rojo: falso BYTE de STATUS

En esta muestra real de 250 datos podemos estar seguros de que hay al menos un paquete completo (se muestra subrayado), dado que cada uno tiene 25 *frames* de 5 bytes cada uno, por lo tanto 25 x 5 = 125 datos en cada paquete (se reciben 3 paquetes cada segundo).

Se puede observar que el número con más apariciones es el '128', dado que este es el BYTE de **STATUS** (se muestra de color negro) y aparece una vez cada 5 datos. Pero hay que tener cuidado, puesto que uno de los datos internos del *frame*, a parte del **STATUS** puede coincidir con este valor. Para eso está el BYTE de CHK. Probemos con la primera secuencia subrayada: *128,117,43,0,32*

1. **STATUS** → 128
2. PLETH(MSB) → 117
3. PLETH(LSB) → 43
4. BYTE 4 → 0
5. CHK → 32

Si le aplicamos la ecuación del *checksum*: Modulo^{256} (128 + 117 + 43 + 0) = 32

Calcular modulo → => (288 % 256) = 32; donde (a): sumatorio, (b): 256

**Web para calcular el módulo:* *Ejemplo*

- *(a):* $\sum(\text{Byte1} + \text{Byte2} + \text{Byte3} + \text{Byte4})$ => $\sum(128 + 117 + 43 + 0) = 288$
- *(b): 256*
- *Mod valor de (a % b): debe coincidir con el valor de Byte5, para confirmar la sincronización a nivel de frame.*

https://www.easycalculation.com/es/algebra/modulo-calculator.php

Esto verifica el sincronismo a nivel de *frame*. Más adelante en la muestra, el **STATUS** pasa a tener el valor 130 por la activación del BIT1 que representa la calidad de señal en la obtención de datos. Pero la ecuación sigue el mismo procedimiento.

Una vez que tenemos el 1. byte de cada *frame* controlado (se muestra en **negrita**), sólo debemos recorrer los siguientes datos en grupos de 5 para controlar que avanzamos ordenadamente de *frame* en *frame*.

Una vez sincronizados a nivel de *frame* pasamos a sincronizar a nivel de paquete. Para eso solo hay que encontrar el **STATUS** con valor mayor en una unidad, en este ejemplo, 128 (en los casos de calidad media de la señal) o 131 (en los casos con buena señal). A partir de aquí, solo habría que ordenar los datos en matrices con la estructura que presenta la documentación *OEM III*.

Advertencia: como se ha comentado anteriormente, pueden aparecer valores idénticos a los que deben tener los **STATUS** entre los BYTE intermedios, por ejemplo, el '131' marcado en rojo.

Después de este proceso, adquirir los datos es simple:

OBTENER DATO **PLETH**:

Para adquirir los puntos para construir el gráfico del pulso, se deben unir los BYTE 2 y 3 de cada *frame.* Usando el mismo *frame* de ejemplo anterior:

128,117,43,0,32

El dato **PLETH**, o punto del grafico se obtiene con la ecuación que se muestra bajo la matriz:

Notes:
- Refer to DF#2 for definitions on Status Byte, Byte 4 and Checksum
- PLETH(MSB) and PLETH(LSB) make up a 16-bit pleth (ex. PLETH(MSB)*256 + PLETH(LSB))

Ilustración 75: Documentación OEM III - Obtener Pleth

PLETH Data = BYTE 2 & BYTE 3 => PLETH Data = [PLETH(MSB) x 256] + PLETH(LSB) => PLETH Data = (117 x 256) + 43 = 29995

Y para el resto de datos significativos del BYTE 4 sólo hay que referenciar la tabla de documentación de OEM III para obtenerlos. En el siguiente ejemplo conseguiremos el dato HR.

OBTENER DATO **HR**:

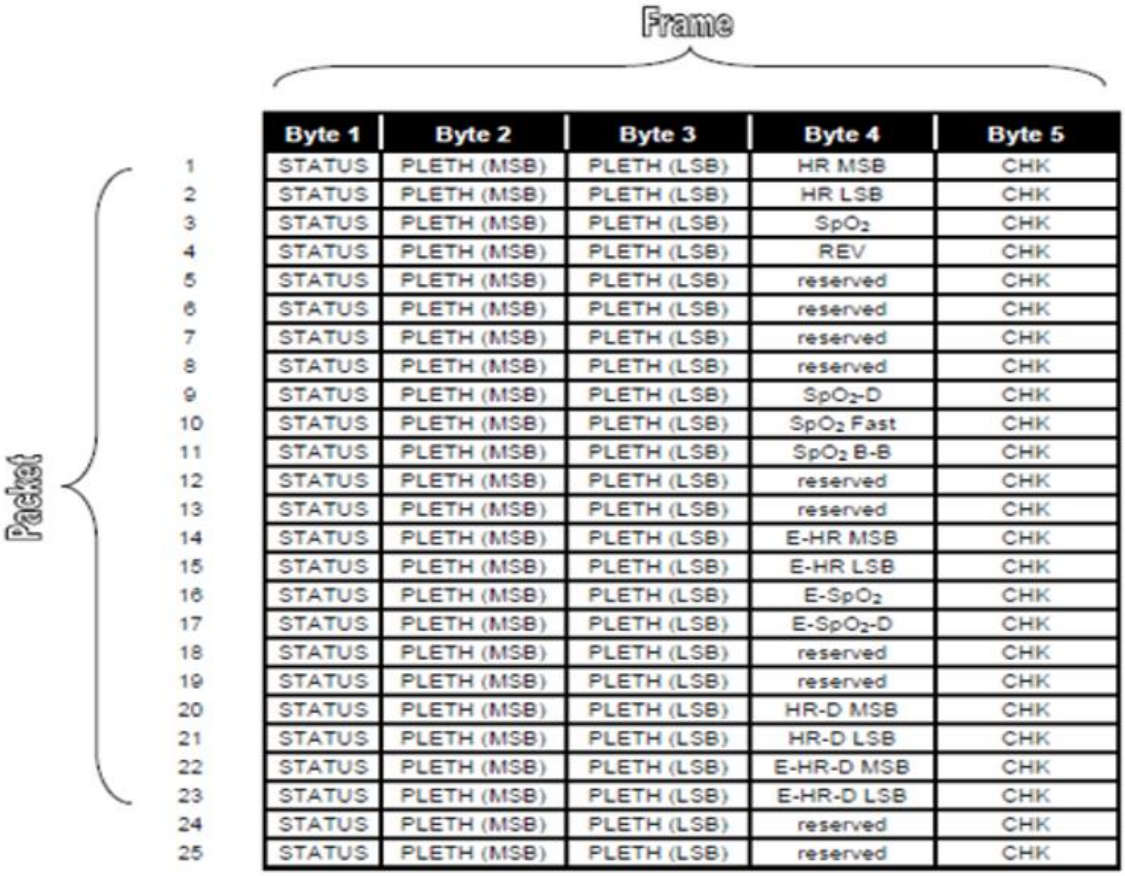
Siguiendo la tabla del formato utilizado, el dato HR aparece en el BYTE 4 de cada *frame*, como el resto de datos significativos. Para obtener este en particular, hacen falta 2 datos, el BYTE 4 del 1. FRAME y el BYTE 4 del 2. FRAME. Una vez localizados, sólo hay que seguir el mismo proceso que con el dato PLETH.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK
STATUS	PLETH (MSB)	PLETH (LSB)	HR LSB	CHK

Ilustración 76: Documentación OEM III - Dato HR

11.6.2. Explicación gráfica

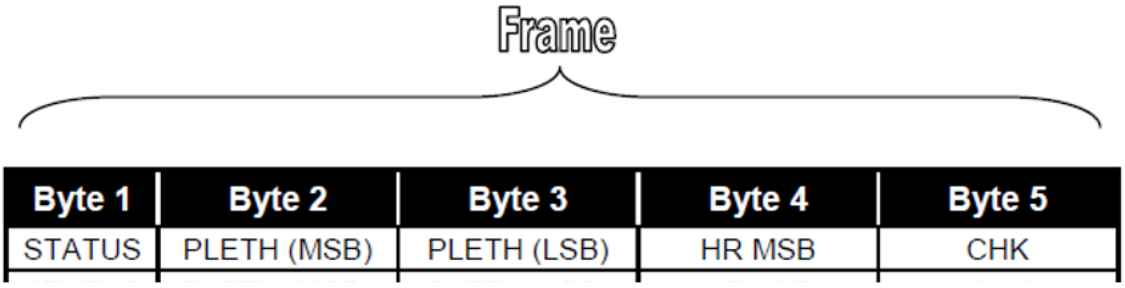
Para comenzar, esta es la estructura de un PAQUETE de la señal, en el formato *Serial Data Format #7* de OEM III:



	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
1	STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK
2	STATUS	PLETH (MSB)	PLETH (LSB)	HR LSB	CHK
3	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2	CHK
4	STATUS	PLETH (MSB)	PLETH (LSB)	REV	CHK
5	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
6	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
7	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
8	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
9	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2-D	CHK
10	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 Fast	CHK
11	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 B-B	CHK
12	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
13	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
14	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR MSB	CHK
15	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR LSB	CHK
16	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2	CHK
17	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2-D	CHK
18	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
19	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
20	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D MSB	CHK
21	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D LSB	CHK
22	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D MSB	CHK
23	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D LSB	CHK
24	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
25	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK

Ilustración 77: Documentación OEM III - Paquete

Contiene 25 FRAMEs. Y cada uno de ellos tiene 5 byte:



Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK

Ilustración 78: Documentación OEM III - Frame

De estos 5 byte, el dato Pleth que se necesita para pintar el gráfico se calcular a partir del 2º y 3º byte:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK

Ilustración 79: Documentación OEM III - Bytes de PLETH

De este modo, la resolución de onda pletimográfica es de 16 bits, es decir, 2^{16} = 65536 valores, razón que la hace tan precisa. A continuación, se muestra el valor de uno de los puntos pleth que más adelante se pintaron en forma de gráfico:

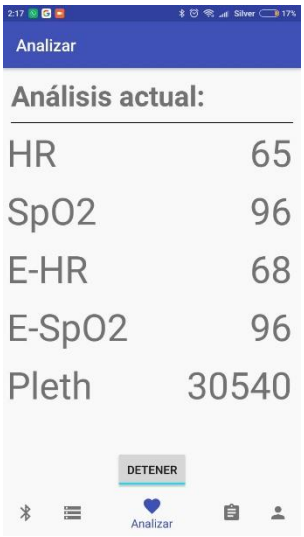


Ilustración 80: Pantalla Analizar - Valor dato PLETH

Una vez que tenemos los datos de la señal listas para pasear, el primer paso es la sincronización a nivel de frame. Para ese propósito existe el 5º Byte, el CHK o *Checksum*.

Este es el proceso a seguir para conseguirlo:

- Se crea un Array de 4 posiciones y se rellena con 4 byte secuenciales.

Ilustración 81: Array de 4 posiciones

- Se calcular el módulo 256 de la suma de los 4 bytes.

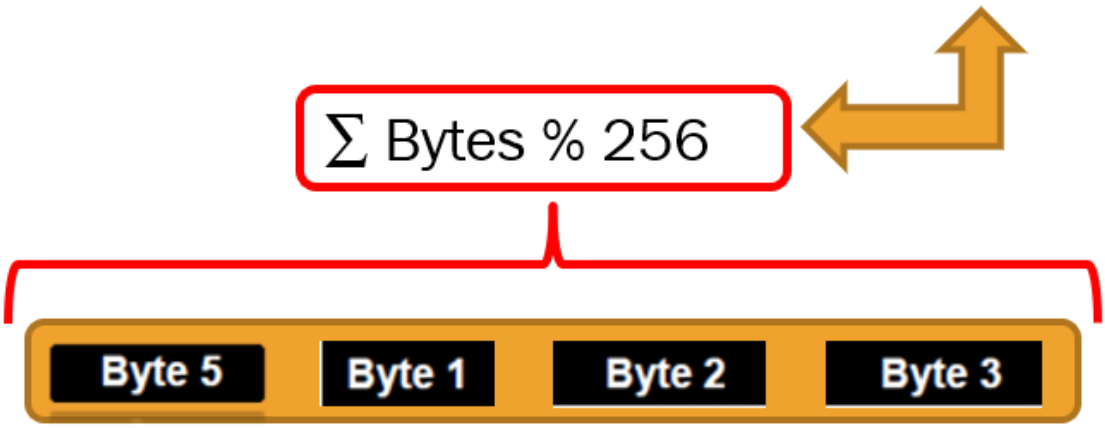


Ilustración 82: Módulo 256 del sumario de valores del array

- Y se compara con el que sería el 5º byte, es decir, el dato inmediatamente posterior al último insertado en el array.

Si hay sincronismo a nivel de Frame:

- $\text{CHK} = \sum Array[4] \ \% 256$

En caso contrario, los valores no serán coincidentes.

En la siguiente imagen se muestra un ejemplo de frame no sincronizado, en consecuencia, el valor del cálculo no coincide con el valor del 5º byte, porque no es el CHK.

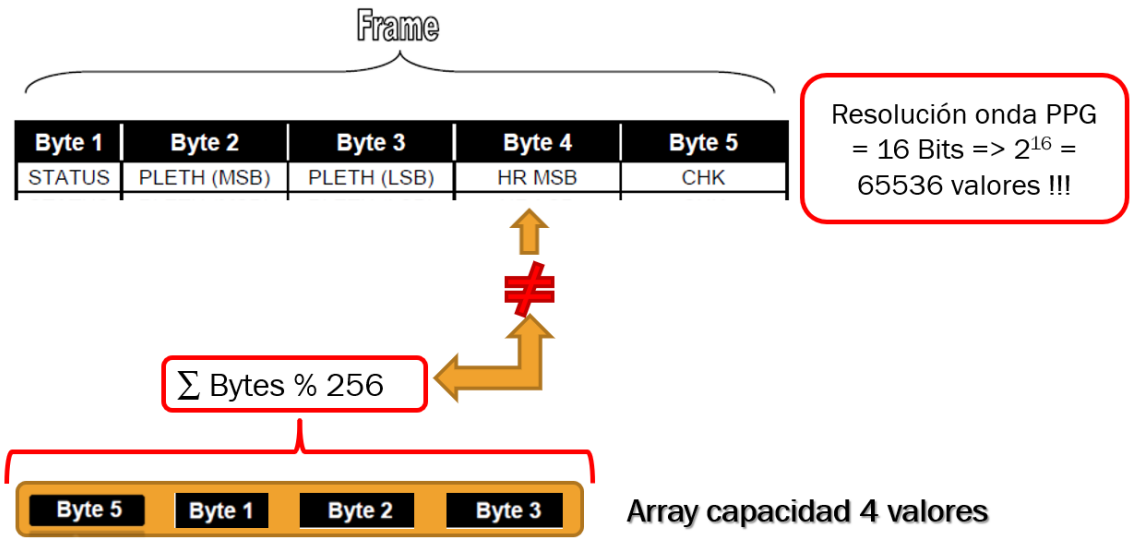


Ilustración 83: Sincronización incorrecta

El procedimiento a seguir después de este caso, comienza por desplazar los bytes del array, del tal modo que el 1º elemento del array se elimina, y el que era el 5º elemento a comparar en la iteración anterior se inserta en la última posición. Se ha de repetir este proceso hasta conseguir la coincidencia como en la siguiente imagen:

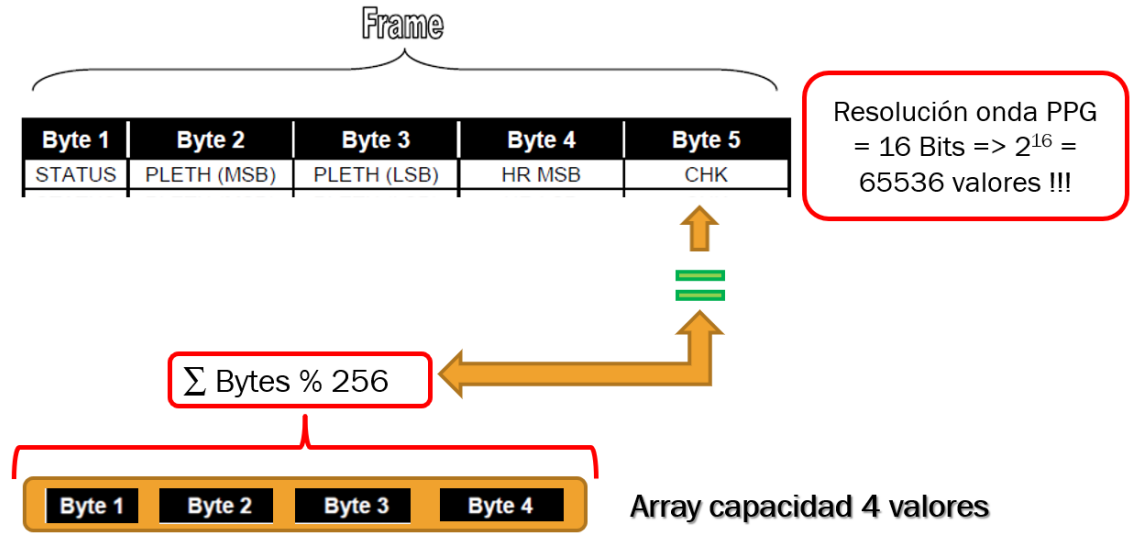


Ilustración 84: Sincronización correcta

11.7. Crear proyecto en Google Developer Console

Para subir el registro a Drive, se ha seguido la API de Google Drive (https://developers.google.com/drive/). Y el primer paso es crear un proyecto y su respectivo ID para que puedan acceder a los recursos de la API específica.

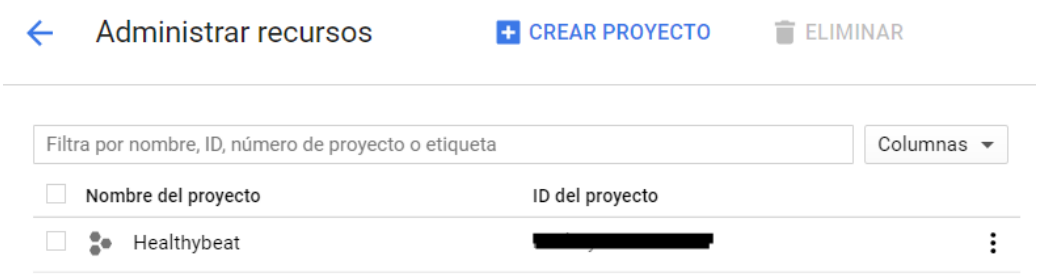


Ilustración 85: Google Developers Console – Crear proyecto

En este caso, se ha habilitado Google Drive API:

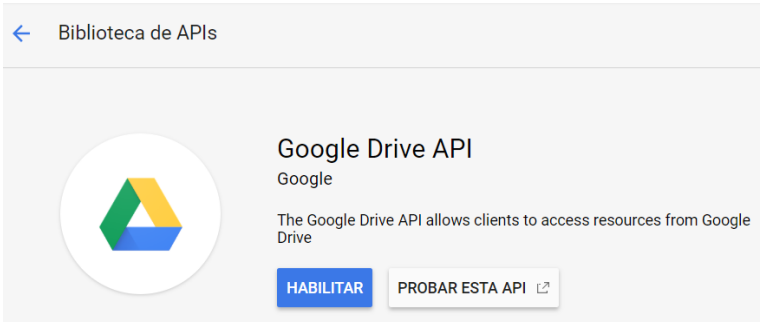


Ilustración 86: Google Developers Console - Google Drive API

Después de este paso, hace falta crear las credenciales para permitir y del mismo modo poder gestionar en el futuro el uso de la API por la app.

Ilustración 87: Google Developers Console - Panel de control de API

A continuación, hay que añadir las credenciales del proyecto:

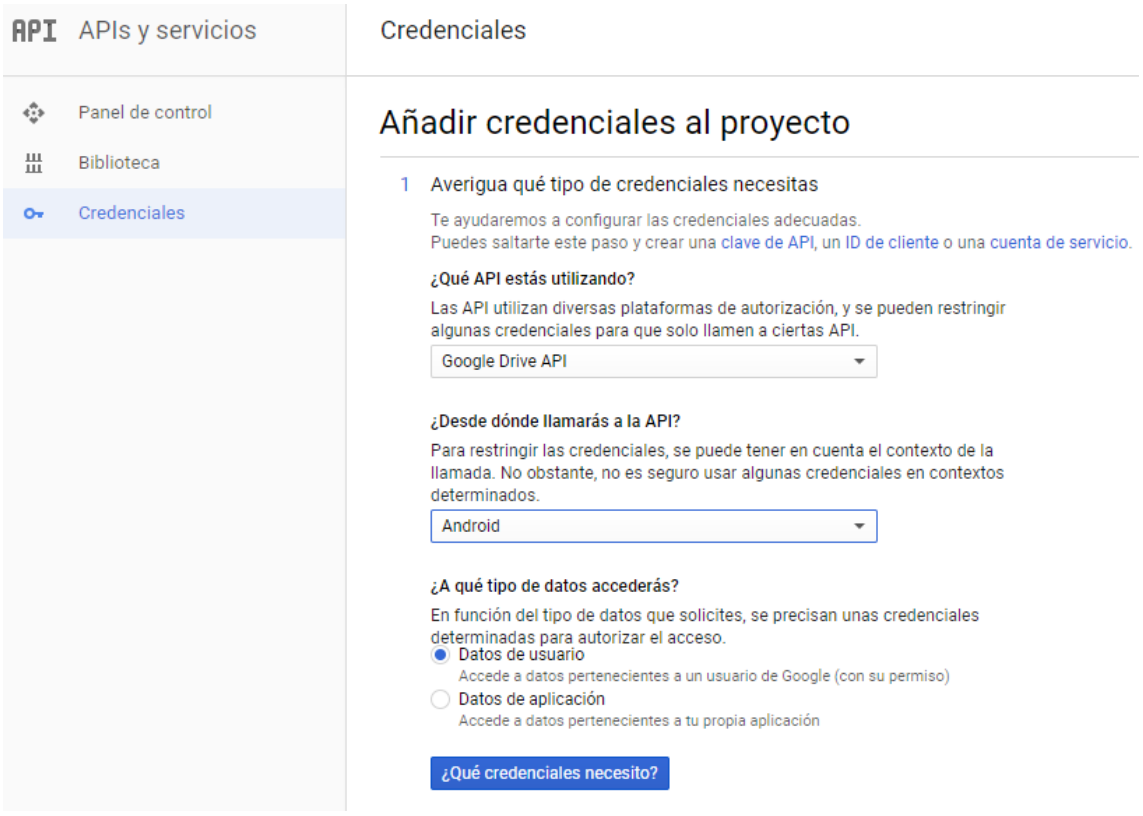


Ilustración 88: Google Developers Console - Añadir credenciales

El siguiente paso es generar las credenciales:

- Nombre de ID
- Hay que generar una huella digital de certificado de firma (SHA-1[43]) copiando el comando y ejecutándolo en la consola, mediante el keytool alojado en la capeta /bin de JAVA.
- Nombre del paquete: es el nombre del paquete de la aplicación, aparece en el AndroidManifest.xml

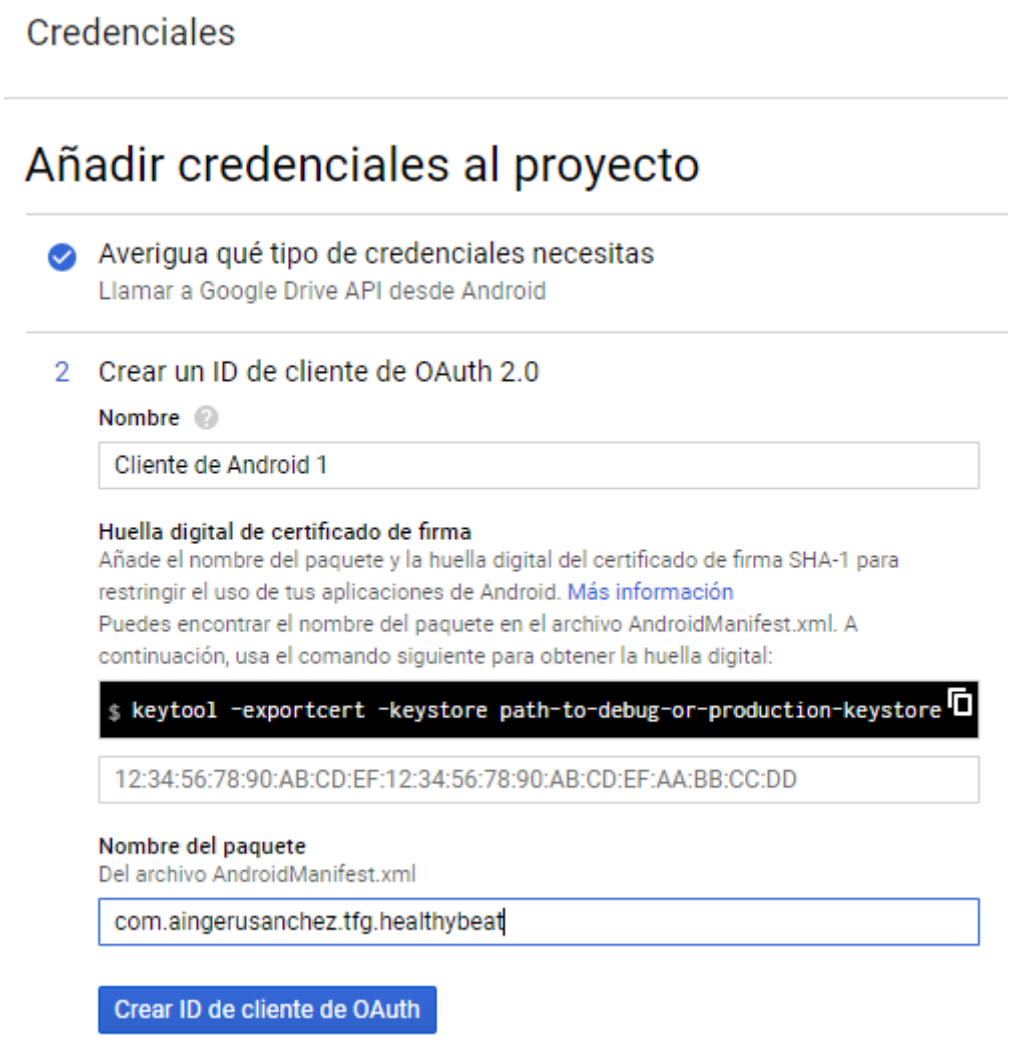


Ilustración 89: Google Developers Console - Generar credenciales

Una vez creado el ID de cliente de OAuth, aparece la clave de API que necesitaremos en la aplicación.

11.8. Emulador de la pulsera

La razón del desarrollo de esta tarea, ha sido una avería en la pulsera. En las mediciones de esta, se empezaron a detectar datos poco fiables y/o incorrectos. Para solventarlo, se ha decidido crear un script que simule la parte de la pulsera de la siguiente manera:

El script se arranca desde un ordenador y se conecta a un puerto COM (o puertos Serie[44]) de salida Bluetooth. Permanece a la espera a que la aplicación se vincule y comienza a mandar los datos de una muestra (previamente tomada con datos reales) de forma cíclica mientras haya conexión.

Antes de ejecutar el código del script, se ha de vincular el dispositivo móvil con el ordenador vía Bluetooth para poder asignarle un puerto serie como se muestra en la siguiente imagen:

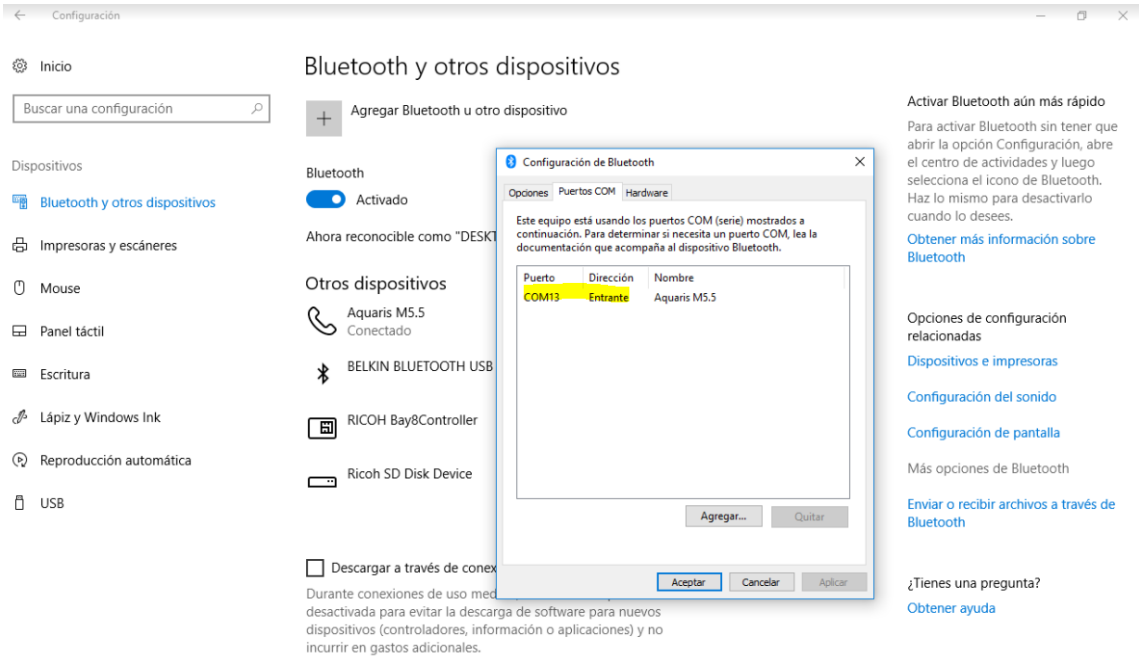


Ilustración 90: Puerto COM entrante vinculado con dispositivo móvil

A parte de la conexión, hay que instalar las dependencias que necesita el script:

```
C:\Users\Aingeru Sanchez\Dropbox\mapalazuelo\emulador_pulsera>c:\Python27\Scripts\pip.exe install pyserial
Collecting pyserial
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3915cbe3c8cc06e209f59c30
/pyserial-3.4-py2.py3-none-any.whl (193kB)
    100% |################################| 194kB 1.7MB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.4
You are using pip version 9.0.3, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Ilustración 91: Instalar imports del script Emulador Pulsera

A continuación, se explica el código del emulador de pulsera:

Primero se crea una lista con los 256 puertos serie virtuales:

```
if sys.platform.startswith('win'):
    ports = ['COM' + str(i) for i in range(256)]
```

Ilustración 92: Código Emulador Pulsera - Crear puertos virtuales

Segundo, se obtiene los puertos COM que están activos:

```
conn_ports = []
for each in ports:
    try:
        s = serial.Serial(each)
        s.close()
        conn_ports.append(each)
    except:
        pass
```

Ilustración 93: Código Emulador Pulsera - Obtener puertos COM activos

Tercero, se le muestra la lista de puertos activos al usuario para que escoja el número del puerto entrante vinculado con el dispositivo móvil que se ha conectado mediante la app.

```
print conn_ports
port_num = raw_input("Select COM number: ")
```

Ilustración 94: Código Emulador Pulsera - Seleccionar puerto COM

Se debe escoger el mismo puerto que aparece como entrante del dispositivo móvil vinculado vía Bluetooth.

A partir de aquí, el emulador de la pulsera se encarga de enviar datos de la señal simulando el funcionamiento real de la pulsera.

```
serial_conn = serial.Serial(port, 9600) # establecemos conexion serie con el modulo de BT a traves del COM

f = open("Datos.csv", 'r')
samples = f.readlines()
num_samples = len(samples)
f.close()

i = 0
while i < num_samples: # buque para enviar las muestras de forma indefinida
    sample = hex(int(samples[i])) # string --> int --> hex
    line = serial_conn.write(sample)
    serial_conn.write("\r\n") # para poder ver datos en BT terminal (quitar en version final)
    if i == (num_samples - 1):
        i = 0
    else:
        i += 1
    time.sleep(1)
```

Ilustración 95: Código Emulador Pulsera - Envió de datos a la app

Como se muestra en el código superior, primero se establece la conexión serie con el módulo BT a través de la interfaz de puertos serie, a 9600 baudios.

Después se carga el fichero con la muestra que se va a enviar iterativamente.

12. Descripción de resultados

En esta sección se describen los resultados obtenido en comparación a los objetivos marcados al inicio del proyecto.

Se ha desarrollado una aplicación móvil Android capaz de parsear y visualizar las muestras tomadas por el pletismógrafo. Aunque se han probado varías librerías para conseguir el resultado deseado, se ha conseguido pintar la señal en forma de gráfica.

También se ha logrado el objetivo de guardar un registro de las muestras en el servicio en la nube de Google Drive.

En lo que respecta a la conectividad Bluetooth, se ha implementado la actividad encargada de gestionar las conexiones, pero dada la avería de la pulsera, no se ha podido conectar con esta. Sin embargo, para solventar este inconveniente, se ha desarrollado un script que la sustituye simulando su función.

En general, se han alcanzado los objetivos propuestos y se han encontrado soluciones a los problemas surgidos durante el desarrollo.

A continuación, se explica un breve tour como resumen del proceso de desarrollo con imágenes para poder visualizar los resultados.

Al iniciar la fase de desarrollo, la base para comenzar fue crear las diferentes pantallas para cada propósito, siguiendo las conclusiones obtenidas del apartado de análisis de mercado . Y después de esto se implementó un menú de navegación para comunicarlas:

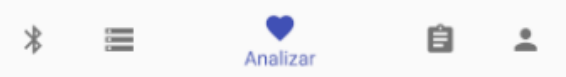


Ilustración 96: Menú de navegación

Una vez implementada esta base, se pasó a la actividad del Bluetooth, puesto que para conseguir el resto de funcionalidades primero necesitábamos obtener la señal para trabajar con ella

Ilustración 97: Pantalla – Bluetooth

Dadas las especificaciones del dispositivo móvil usado para las pruebas de conexión Bluetooth, hubo bastantes complicaciones: el dispositivo es un Xiaomi Mi5, con SO MIUI basado en Android, pero con protocolos de BT reimplementados y diferentes en cada versión. Aun así, se consiguió realizar la vinculación con otros dispositivos Android.

El siguiente paso y al mismo tiempo la prueba para confirmar que la app se conectaba con otros dispositivos correctamente, debería haber sido la conexión con la pulsera y la recepción de la señal muestreada. Sin embargo, por razones técnicas, la pulsera no obtenía la señal con la precisión que debería y los datos no eran fiables. Por ese motivo se ha desarrollado un emulador de pulsera que simula su función.

Para este caso de riesgo, se obtuvo y almaceno en su momento un *backup* de datos con el que se pudo continuar el desarrollo. Por lo tanto, se pasó a la fase de parseo de esta señal y seguidamente a dibujar datos en pantalla:

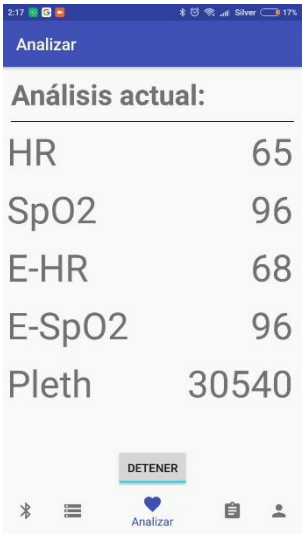


Ilustración 98: Pantalla - Analizar con datos

Ilustración 99: Pantalla - Analizar con gráfica

Como se muestra en las imágenes anteriores, primero se mostraron los datos parseados (imagen izquierda) y después de dibujaron en la gráfica con las diferentes librerías hasta obtener el resultado buscado (imagen derecha).

Al finalizar este proceso, se preparó la subida automática de la muestra tomada al servicio de almacenamiento en la nube de Google Drive:

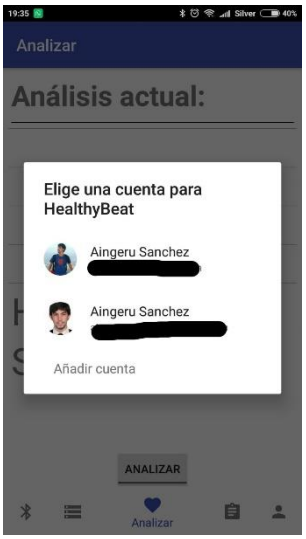


Ilustración 100: Elegir cuenta de Google

Ilustración 101: Registros en Drive

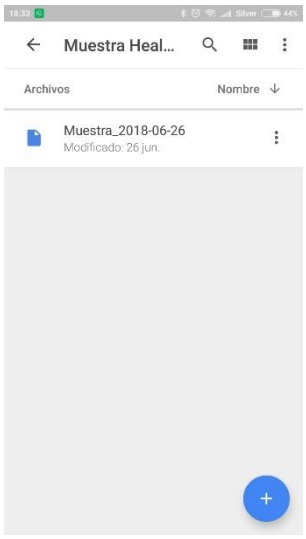


Ilustración 102: Muestra guardada en Drive

Al iniciar la app se ha elegido una cuenta de Google en la que se almacenarán las muestras (imagen izquierda), al subirlas se creará una carpeta en la raíz de Drive con un nombre identificativo con la fecha (imagen central) y dentro de esta se encuentra el archivo con los datos ya parseados (imagen derecha).

En la siguiente dirección URL está disponible el repositorio de código de la aplicación HelathyBeat:

https://github.com/Aingeru72/HealthyBeat

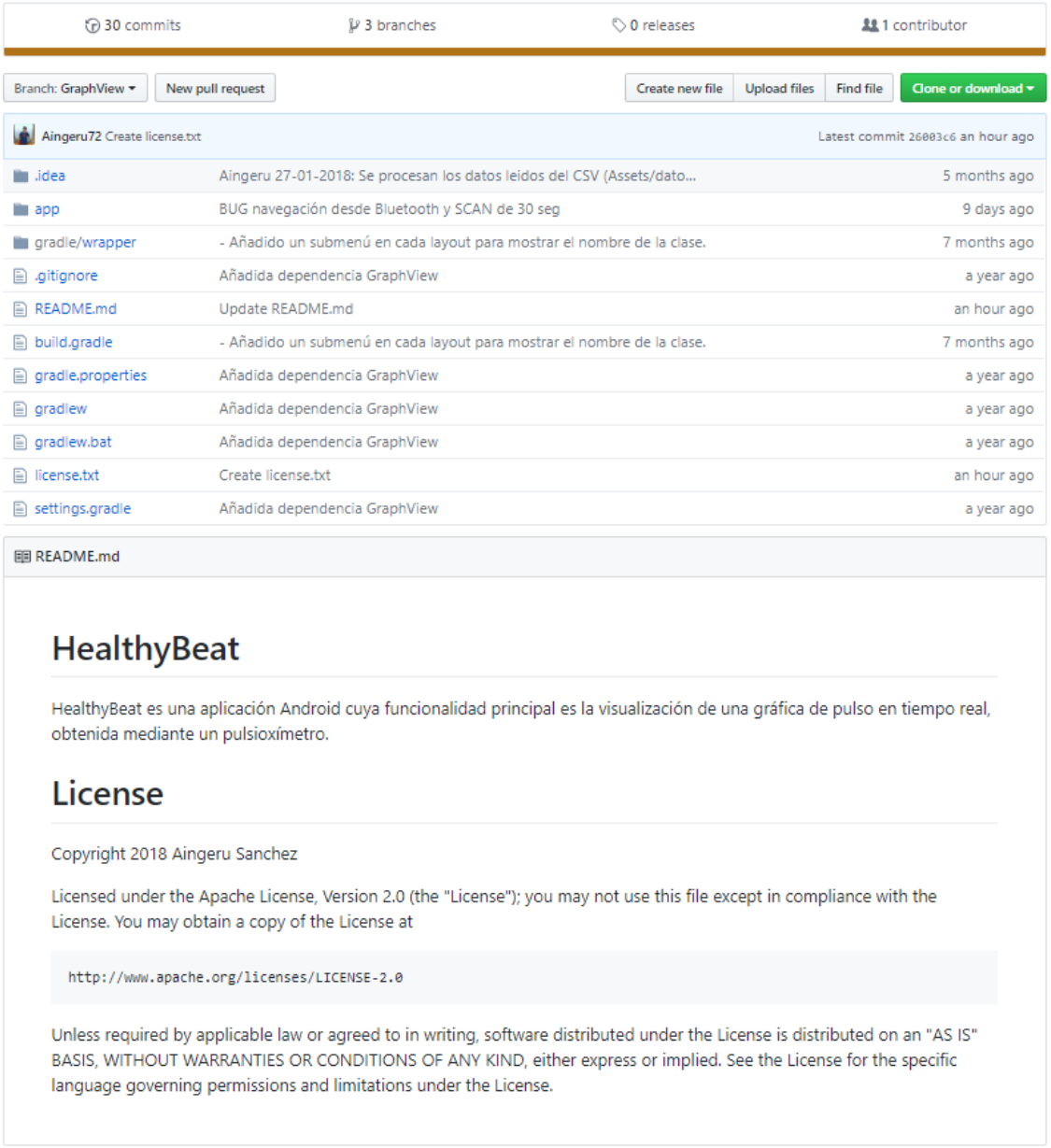


Ilustración 103: Repositorio en GitHub de HealthyBeat

13. Aspectos económicos

En este apartado se evalúan los aspectos económicos del proyecto.

13.1. Personal

Los siguientes salarios han sido obtenidos del *BOE-A-2009-5688*:

Grupo III. *Técnicos de oficina*	Salario mensual (€)	Salario anual (€)
Analista y Analista de sistemas	1.538,48	21.538,72
Analista-programador/a y Diseñador/a de página Web	1.509,50	21.133,00
Programador/a senior, Jefe/a de operación y Programador/a en Internet	1.081,41	15.139,74
Delineante-proyectista	1.001,09	14.015,26
Programador/a junior, Operador/a ordenador, Programador/a máquina auxiliar, Monitor/a de grabación y Técnico/a mantenimiento página Web.	968,32	13.556,48
Delineante	869,31	12.170,34
Administrador/a de test.	824,44	11.542,16

Ilustración 104: Salarios base técnicos de oficina (BOE-A-2009-5688)

En este proyecto se han asumido los roles de Analista y de programador Junior. Obteniendo las horas de cada rol de la tabla de estimación de horas, se han desglosado los siguientes sueldos base estipulados por el BOE en €/hora:

Tabla 25: Gastos de personal

Rol	Sueldo (€/Hora)	Horas	TOTAL
Analista	6.5 €/hora	55h	357,5 €
Programador Junior	4 €/hora	290h	1160 €
TOTAL		**345h**	**1517,5 €**

13.2. Software

En la siguiente tabla se muestran los gastos relativos al software utilizado en el proyecto:

Tabla 26: Gastos de Software

Software	Licencia	Coste
Licencia Windows (10 Education)	Licencia gratuita por el acuerdo entre la EHU/UPV y Microsoft: Plataforma DreamSpark	0€
Android Studio (IDE Android)	Licencia gratuita	0€
Google Drive (Cuenta de Google)	Cuenta gratuita	0€
JAVA, Python	Descarga gratuita	0€
GIT	Descarga gratuita	0€
Repositorio en GitHub	Repositorio gratuito	0€
Navegador Chrome	Navegador gratuito	0€
TOTAL		**0€**

13.3. Hardware

En la siguiente tabla se muestran los gastos relativos al hardware utilizado en el proyecto:

Tabla 27: Gastos de Hardware

Hardware	Coste total	Vida útil	Coste proporcional
Maquina: *Portátil MSI GE60 2PC Apache*	950 €	4 años	9,38 €
Dispositivo móvil: Xiaomi Mi5	350 €	2 años	0,6 €
Pulsera: estación pletismográfica	Cedida por el desarrollador de la misma: Josu Alonso	2 años	0€
TOTAL			**9,98 €**

Para calcular los costes relativos:

f(x) = [Coste del HW / Vida útil / Semanas (año) / Días (semana) / Horas (día)] * Horas dedicadas en el proyecto

- Máquina portátil: (950 / 4 / 52 / 7 / 24) * 345 = 9.38€
- Dispositivo móvil: (350 / 4 / 52 / 7 / 24) * 30 = 0.6€

La máquina portátil se ha utilizado durante todo el proyecto, el dispositivo móvil sin embargo se ha dedicado para la sección de pruebas y para el estudio de mercado (descargar y probar las aplicaciones similares de *Play Store*).

Resumiendo, esta es la tabla que resume los gastos totales del proyecto:

Tabla 28: Gastos totales

Gastos	Coste
Personal	1.517,5 €
Software	0 €
Hardware	9,98 €
TOTAL	1.527,48 €
IVA (21%)	320,77 €
TOTAL (Con IVA)	**1.848,25 €**

14. Conclusiones

Para finalizar, se explican las conclusiones alcanzadas:

Respecto a los objetivos definidos, se pueden decir que se han completado casi en su totalidad. La única tarea que ha quedado sin implementar ha sido la conexión entre la pulsera y la app vía Bluetooth. Ha de añadirse que esta circunstancia está justificada, la pulsera dejo de funcionar correctamente. Aun así, se utilizó la muestra de datos que se creó a modo *backup* o copia de seguridad para poder continuar con el desarrollo. Y de esta forma, se han completado el resto de tareas, que es una parte muy valorable ya que la posibilidad de encontrar contratiempos del estilo en otros proyectos reales es muy alta, y se ha de aprender a encontrar soluciones.

Al ser un proyecto en el que no existía experiencia previa, una de las conclusiones ha sido el desvío en los plazos de cada tarea, la diferencia entre las horas previstas y las reales es significativa. Sin embargo, se ha adquirido una importante experiencia que no se había visto durante el grado, motivo importante de la elección del mismo.

Pensando en el trabajo a futuro, se podría implementar las siguientes funcionalidades, mejoras, optimizaciones:

- El inicio de sesión con la cuenta de Google:
  - De este modo se reutilizaría a la hora de guardar los registros de muestras en Google Drive.
  - Se podrían guardar los ajustes de cada usuario en su perfil (edad, sexo y otros datos relevantes para el estudio) y conectarla con el servicio de Firebase (https://firebase.google.com/?hl=es-419) de Google para almacenar sus datos en persistencia.

- Sacar provecho al resto de pantallas y menú para darle al usuario nuevas opciones de personalización y ajustes (tamaños de fuente, contraste de colores, idiomas…).
  - Crear una pantalla que muestre los registros guardados en Drive e implementar la opción para volver a dibujarlos o simplemente ver los resultados de la misma.

- Optimizar la aplicación investigando las herramientas que brinda la API de Android:
  - Guardar los registros en el propio dispositivo para poder trabajar momentáneamente off-line, podría utilizarse la clase *SharedPreferences*, aunque está más orientada para guardas los ajustes del usuario.

- Ampliar la visión del proyecto: igual que este TFG ha servido como continuación del que ya desarrolló el compañero Josu Alonso, se podrían proponer ideas a nuevos alumnos, tales como:
    - Guardar los registros de las muestras de todos los pacientes en BBDD[45] para el estudio exhaustivo de estas, paradigma de *Big Data*[46]. Por ejemplo, mediante la implementación de minería de datos[47] para detectar patrones indicativos de las patologías investigadas.

15. Bibliografía

García, J. A. (23 de 04 de 2016). Desarrollo de una estación pletismográfica inalámbrica para el almacenamiento y procesado de su señal. Aplicación en pacientes con trastornos del sueño. *Memoria*. Bilbao, País Vasco, España.

Gehring, J. (s.f.). *GraphView*. Obtenido de GraphView: http://www.android-graphview.org

Google. (s.f.). *Google Play Store*. Obtenido de Play Store: https://play.google.com/store

Ministerio, T. I. (2009). III. Otras disposiciones. *BOE*, 32.

Statcounter. (Diciembre de 2017). Obtenido de Statcounter: http://gs.statcounter.com

Wikipedia. (s.f.). Obtenido de https://es.wikipedia.org

15.1. Recursos gráficos

- Ilustraciones de la 1 a la 6, 7 y 24 han sido obtenidas de TFG de Josu Alonso García referenciado en la Bibliografía.
- Ilustraciones de la 9 a la 14 sobre aplicaciones de *Play Store,* han sido obtenidas de Google Play Store: https://play.google.com/store
- Ilustración 16 del Logo de *HealthyBeat* ha sido creado mediante la web *Designevo*: https://www.designevo.com/
- Ilustraciones 20 y 21 sobre librerías Android para gráficos, han sido obtenidas de Android Arsenal: https://android-arsenal.com/
- Ilustraciones 22, 23, 25, 32 y 33 han sido creadas mediante la web Draw.io: https://www.draw.io/
- Las ilustraciones 26, 27, de la 48 a la 51, 65, 66, 80, y de la 96 a la 100 son capturas de pantalla de la aplicación *HealthyBeat*.
- La ilustración 31 sobre HRV ha sido obtenida de la web www.desarrollandojuntos.com (https://i0.wp.com/www.desarrollandojuntos.com/wp-content/uploads/2016/01/Grafico-Variabilidad-Frecuncia-Cardiaca.jpg)
- La ilustración 32 sobre el mockup ha sido creado mediante la web Cacoo: https://cacoo.com/
- Las ilustraciones de la 34 a la 41 sobre los diagramas UML, han sido creadas en Android Studio mediante el plugin *SimpleUMLCE* (JetBrains): https://plugins.jetbrains.com/plugin/4946-simpleumlce
- Las ilustraciones 42, 52 y 53 sobre los diagramas de secuencia, han sido creadas en Android Studio mediante el plugin *SequenceDiagram* (JetBrains): https://plugins.jetbrains.com/plugin/8286-sequencediagram
- Las ilustraciones de la 43 a la 47, de la 54 a la 64, 67, 68 y de la 70 a la 72 sobre el código de la aplicación, son capturas de pantalla de Android Studio.

- Las ilustraciones de la 73 a la 79 son capturas de la documentación OEM III del Anexo I.
- Las ilustraciones de la 85 a la 89 son captura realizadas en la web de *Google Developers Console*: https://console.cloud.google.com/
- Las ilustraciones de la 92 a la 95 sobre el código del emulador de la pulsera, son capturas de pantalla de PyCharm.
- Las ilustraciones 101 y 102 son capturas de pantalla de la aplicación Google Drive.

15.2. Librerías de software

- Librería de gráficos **Snake View**:
  - Android Arsenal: https://android-arsenal.com/details/1/2675
  - GitHub: https://github.com/txusballesteros/snake?utm_source=android-arsenal.com&utm_medium=referral&utm_campaign=2675

- Librería de gráficos **Spark**:
  - Android Arsenal: https://android-arsenal.com/details/1/3446
  - GitHub: https://github.com/robinhood/spark?utm_source=android-arsenal.com&utm_medium=referral&utm_campaign=3446

- Librería de gráficos **GraphView**:
  - Web del proyecto: http://www.android-graphview.org/
  - API (Javadoc): http://jjoe64.github.io/GraphView/javadoc/
  - GitHub: https://github.com/jjoe64/GraphView

16. Glosario

[1] **MOOC (*Massive Open Online Course*):** Curso On-line Masivo y Abierto. Son cursos en línea dirigidos a un número ilimitado de participantes a través de Internet según el principio de educación abierta (es.wikipedia.org/wiki/Educación_abierta) y masiva.
Véase: https://es.wikipedia.org/wiki/Massive_Open_Online_Course

[2] **SO (Sistema Operativo):** Conjunto de órdenes y programas que controlan los procesos básicos de un dispositivo y permiten el funcionamiento de otros programas en el mismo.
Véase: https://es.wikipedia.org/wiki/Sistema_operativo

[3] **IoT (Internet of Things):** Dispositivos cotidianos e independientes con conexión a Internet.
Véase: https://es.wikipedia.org/wiki/Internet_de_las_cosas

[4] **IDE (Integrated Development Environmnet)** Entorno de Desarrollo Integrado: es el programa o plataforma que se utiliza para desarrollar software.
Véase: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[5] **API (*Application Programming Interface*)** Interfaz de programación de aplicaciones: es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
Véase: https://es.wikipedia.org/wiki/Interfaz_de_programación_de_aplicaciones

[6] **Inteligencia Artificial (IA):** se aplica cuando una máquina imita las funciones «cognitivas» que los humanos asocian con otras mentes humanas, como por ejemplo: "aprender" y "resolver problemas".
Véase: https://es.wikipedia.org/wiki/Inteligencia_artificial

[7] **UART (Universal Asynchronous Receiver-Transmitter):**
dispositivo encargado de controlar los puertos y dispositivos serie para comunicación.
Véase: https://es.wikipedia.org/wiki/Universal_Asynchronous_Receiver-Transmitter

[8] **MAC (Media Access Control):** identificador único de cada dispositivo con algún tipo de conexión.
Véase: https://es.wikipedia.org/wiki/Dirección_MAC

[9] **RFCOMM:** (*Radio Frequency Communication*) Son las siglas de *Comunicación por radio frecuencia*. Es un protocolo de conexión basada en puertos de entrada/salida.
Véase: https://es.wikipedia.org/wiki/Protocolos_Bluetooth#RFCOMM

[10] Checksum (CHK): Suma de verificación o checkeo. Tiene como propósito principal detectar cambios accidentales en una secuencia de datos para proteger la integridad de estos.
Véase: https://es.wikipedia.org/wiki/Suma_de_verificación

[11] **HR (*Heart Rate*):** Frecuencia cardíaca, es el número de pulsaciones por unidad de tiempo.
Véase: https://es.wikipedia.org/wiki/Frecuencia_cardíaca

[12] **HRV (*Heart Rate Variability*):** Variabilidad de la Frecuencia Cardíaca es el intervalo de tiempo que transcurre entre dos latidos de corazón consecutivos medido en milisegundos.
Véase: http://www.desarrollandojuntos.com/variabilidad-del-ritmo-cardiaco-una-medida-de-equilibrio-fisico-y-psicologico/

[13] **SpO_2:** Saturación de oxígeno en sangre es la medida de la cantidad de oxígeno disponible en el torrente sanguíneo.
Véase: https://www.gasometria.com/saturacion_de_oxigeno_en_sangre

[14] **lpm/bpm: Latidos por minuto (beats per minute)** Unidad de medida para la frecuencia cardíaca.
Véase: https://es.wikipedia.org/wiki/Pulsaciones_por_minuto

[15] **Wereable:** Dispositivo electrónico que se lleva (o viste) en alguna parte del cuerpo y que sirve para monitorizar diferentes atributos relacionados con el área al que están orientados.
Véase: https://es.wikipedia.org/wiki/Tecnología_vestible

[16] **pletismógrafo**: es la herramienta responsable de llevar a cabo el método basado en la medición de cambios de presión y volumen que se utiliza para medir parámetros orientados al diagnóstico de enfermedades pulmonares y cardiovasculares, también llamada pletismografía.
Véase: *https://es.wikipedia.org/wiki/Pletismografía*

[23] **Parsear**: adaptar una muestra de datos para traducirlos a una estructura de datos comprensibles en nuestro contexto. Ejemplo: en nuestro caso recibiremos una cadena de datos muy extensa de la que tendremos que separar los datos útiles para usarlos posteriormente.
Véase: http://www.alegsa.com.ar/Dic/parseo.php

[29] **Disclaimer (Aviso legal):** descargo de responsabilidad o cláusula de exención de responsabilidad.
Véase: https://es.wikipedia.org/wiki/Aviso_legal

[32] ***Brainstorming* (tormenta de ideas):** es una herramienta de trabajo grupal que facilita el surgimiento de nuevas ideas sobre un tema o problema determinado.
Véase: https://es.wikipedia.org/wiki/Lluvia_de_ideas

[33] **iOS**: Sistema Operativo para dispositivos de Apple.
Véase: https://es.wikipedia.org/wiki/IOS

[34] **Framework:** marco o entorno de trabajo preparado para un desarrollo específico.
Véase: https://es.wikipedia.org/wiki/Framework

[35] **Plugin** (o plug-in)**:** complemento de un software para agregarle una función específica.
Véase: https://es.wikipedia.org/wiki/Complemento_(informática)

[36] **Script:** Programa, por lo general pequeño, que ejecuta una secuencia de comandos o código.
Véase: https://es.wikipedia.org/wiki/Script

[43] **SHA-1** (*Secure Hash Algorthm)* función de cifrado.
Véase: https://es.wikipedia.org/wiki/Secure_Hash_Algorithm#SHA-1

[44] **Puerto serie**: interfaz de comunicaciones de datos digitales, donde la información es transmitida bit a bit.
Véase: https://es.wikipedia.org/wiki/Puerto_serie

[45] **Bases de Datos** (**BBDD**): Conjunto de datos del mismo contexto almacenados para su posterior obtención, modificación o borrado.
Véase: https://es.wikipedia.org/wiki/Base_de_datos

[46] **Big Data (Macrodatos):** Conjunto de datos ingente utilizado para obtener estadísticas y patrones mediante minería de datos.
Véase: https://es.wikipedia.org/wiki/Macrodatos

[47] **Minería de datos** **(*Data Mining*):** campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos
Véase: https://es.wikipedia.org/wiki/Minería_de_datos

Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL PARA EL PARSEO Y LA VISUALIZACIÓN DE LOS DATOS DE UN PULSIOXÍMETRO

ANEXO I – *DOCUMENTACIÓN OEM III (NONIN)*

Alumno: Mikel Aingeru Palazuelo Sánchez

Director: Oskar Casquero Oyarzabal

Codirector: Harkaitz Eguiraun Martínez

Curso: 2017-2018

Fecha: 18-06-2018

OEM III Module Specification
and
Technical Information

Nonin® Medical, Inc.
13700 1st Avenue North
Plymouth, Minnesota 55441-5443
USA

+1 (763) 553-9968
(800) 356-8874 (USA and Canada)

Fax: +1 (763) 553-7807

E-mail: info@nonin.com

nonin.com

4518-001-11 Rev K

Table of Contents

Nonin OEM III Specifications

1.	**Displayed Oxygen Saturation Range (SpO_2)**		0 to 100%	
2.	**Displayed Pulse Rate Range**		18 to 321 beats per minute (BPM)	
3.	**Measurement Wavelengths and Output Power****			
		Red:	660 nanometers @ 0.8 mW maximum average	
		Infrared (using Nonin PureLight® Sensor):	910 nanometers @ 1.2 mW maximum average	
4.	**SpO_2 Accuracy (A_{rms}*)**		70-100%	
			Adults/Pediatrics	**Neonates**
	No Motion			
	REUSABLE:	Finger Clip:	± 2 digits	± 3 digits
		Flex:	± 3 digits	± 3 digits
		Soft Sensor:	± 2 digits	N/A
		8000R:	± 3 digits	N/A
		8000Q2:	± 4 digits	N/A
	DISPOSABLE:	6000C Series:	± 2 digits	± 3 digits
		7000 Series	± 3 digits	± 4 digits
		6500 Series	± 2 digits	N/A
	Motion			
	REUSABLE:	Finger Clip:	± 2 digits	± 3 digits
		Flex:	± 3 digits	± 4 digits
		Soft Sensor:	± 3 digits	N/A
	Low Perfusion	All Sensors:	± 2 digits	± 3 digits
5.	**Pulse Rate Accuracy**			
			Adults/Pediatrics	**Neonates**
	No Motion (18-300 BPM)			
	REUSABLE:	Finger Clip:	± 3 digits	± 3 digits
		Flex:	± 3 digits	± 3 digits
		Soft Sensor:	± 3 digits	± 3 digits
		8000R:	± 3 digits	± 3 digits
		8000Q2:	± 3 digits	± 3 digits
	DISPOSABLE:	6000C Series:	± 3 digits	± 3 digits
		7000 Series:	± 3 digits	± 3 digits
		6500 Series	± 3 digits	N/A
	Motion (40-240 BPM)			
	REUSABLE:	Finger Clip:	± 5 digits	± 5 digits
		Flex:	± 5 digits	± 5 digits
		Soft Sensor:	± 5 digits	± 5 digits
	Low Perfusion (40-240 BPM)	All Sensors:	± 3 digits	± 3 digits

Notes:

Reusable Group:

Finger Clip Sensors: 8000AA-1, 8000AA-3, 8000AP-1, 8000AP-3
Flex Sensors: 8000J-1, 8000J-3, 8008J, 8001J
Soft Sensors: 8000SS, 8000SM, 8000SL

Disposable Group:

Flexi-Form® III (7000 Series) Sensors: 7000A, 7000P, 7000I, 7000N
6000C Cloth Series Sensors: 6000CA, 6000CP, 6000CI, 6000CN
6500 Durafoam Series Sensors: 6500SA, 6500MA

* ±1 A_{rms} represents approximately 68% of measurements.

** This information is especially useful for clinicians performing photodynamic therapy.

Nonin OEM III Specifications

6.	**Temperature (Operating)**	0°C to +50°C (32°F – 122°F) • Specified operating temperature is for the module. • Operating temperature of Sensor is not to exceed 40°C (104°F)
	Temperature (Storage/Transportation)	-20°C to +70°C (-4°F - +158°F)
7.	**Humidity (Operating)**	10 to 90% non-condensing
	Humidity (Storage/Transportation)	10 to 95% non-condensing
8.	**Power Draw**	29 mW typical operation @ 3.3VDC input voltage 45 mW typical operation @ 5.0VDC input voltage
9.	**Voltage Input**	+3.3VDC (3.2V to 3.5V), w/50mV max. ripple +5.0VDC ±0.25VDC, w/50mV max. ripple
10.	**I/O Signals**	0 to +3.3VDC (nominal) @ +3.3VDC input voltage 0 to +5.0VDC (nominal) @ +5.0VDC input voltage
11.	**Dimensions**	1.35" x 0.95" x 0.235" (34.3 x 24.1 x 6.2 mm)
12.	**Weight**	5.3g (0.19 oz.) (with shield)
13.	**Ruggedness (Shock)**	IEC 60068-2-27
	Ruggedness (Vibration)	Sinusoidal – IEC 60068-2-6 Random – IEC 60068-2-64
14.	**Sensors**	Designed to use Nonin-branded PureLight sensors only (see Accessories)
15.	**Shielding**	An RF shield is included (placed over the analog components)

Serial Output Formatting Options

The format for serial output data is determined by the amount of resistance present between the serial data format switch (J1, pin 9) and ground (J1, pin 15). If J1, pin 9 is left unconnected, then the default format is serial data format #2 (see “Serial Output Formatting Options”).

Serial Output Formatting Options

Serial Format #	J1, Pin 9 Status
#1	0Ω to 626Ω
#2	297KΩ to ∞Ω
#7	4.3KΩ, ± 5%

The serial transmission rate for all data formats shall be as follows:

Bits per Second	Data Bits	Parity	Stop Bits	Flow Control
9600	8	None	1	None

Serial Data Format #1:

Packet Description

Three bytes of data are transmitted 1 once per second.

Byte 1 - Status							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
1	SNSD	OOT	LPRF	MPRF	ARTF	HR8	HR7
***Note**: Bit 7 is always set							

Byte 2 - Heart Rate							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
0	HR6	HR5	HR4	HR3	HR2	HR1	HR0
***Note**: Bit 7 is always clear							

Byte 3 - SpO2							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
0	SP6	SP5	SP4	SP3	SP2	SP1	SP0
***Note:** Bit 7 is always clear							

The following are all active high:

SNSD:	Sensor Disconnect	– Sensor is not connected to oximeter or sensor is inoperable.
OOT:	Out Of Track	– An absence of consecutive good pulse signals.
LPRF:	Low Perfusion	– Amplitude representation of low signal quality (holds for entire duration).
MPRF:	Marginal Perfusion	– Amplitude representation of medium signal quality (holds for entire duration).
ARTF:	Artifact	– A detected pulse beat didn’t match the current pulse interval.
HR8 – HR0:	Heart Rate	– Standard 4-beat average values not including display holds.
SP6 – SP0:	SpO_2	– Standard 4-beat average values not including display holds.

When SpO_2 and HR cannot be computed, the system will send a missing data indicator. For missing data, the HR equals 511 and the SpO_2 equals 127.

Serial Data Format #2:

Packet Description

A frame consists of 5 bytes; a packet consists of 25 frames. Three packets (75 frames) are transmitted each second.

Frame

Packet

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
1	01	STATUS	PLETH	HR MSB	CHK
2	01	STATUS	PLETH	HR LSB	CHK
3	01	STATUS	PLETH	SpO_2	CHK
4	01	STATUS	PLETH	REV	CHK
5	01	STATUS	PLETH	reserved	CHK
6	01	STATUS	PLETH	reserved	CHK
7	01	STATUS	PLETH	reserved	CHK
8	01	STATUS	PLETH	reserved	CHK
9	01	STATUS	PLETH	SpO2-D	CHK
10	01	STATUS	PLETH	SpO_2 Fast	CHK
11	01	STATUS	PLETH	SpO_2 B-B	CHK
12	01	STATUS	PLETH	reserved	CHK
13	01	STATUS	PLETH	reserved	CHK
14	01	STATUS	PLETH	E-HR MSB	CHK
15	01	STATUS	PLETH	E-HR LSB	CHK
16	01	STATUS	PLETH	E-SpO_2	CHK
17	01	STATUS	PLETH	E-SpO_2-D	CHK
18	01	STATUS	PLETH	reserved	CHK
19	01	STATUS	PLETH	reserved	CHK
20	01	STATUS	PLETH	HR-D MSB	CHK
21	01	STATUS	PLETH	HR-D LSB	CHK
22	01	STATUS	PLETH	E-HR-D MSB	CHK
23	01	STATUS	PLETH	E-HR-D LSB	CHK
24	01	STATUS	PLETH	reserved	CHK
25	01	STATUS	PLETH	reserved	CHK

Notes:

- Byte number 1 in each frame is set to a value of 1.
- Reserved bytes are undefined.

Byte 2 - Status							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
1	SNSD	ARTF	OOT	SNSA	YPRF		SYNC
					RPRF	GPRF	
*__Note__: Bit 7 is always set.							

The following are all active high:

SNSD:	Sensor Disconnect	– Sensor is not connected to oximeter or sensor is inoperable.
ARTF:	Artifact	– A detected pulse beat didn't match the current pulse interval
OOT:	Out Of Track	– An absence of consecutive good pulse signals
SNSA:	Sensor Alarm	– Sensor is providing unusable data for analysis
RPRF:	Red Perfusion	– Amplitude representation of low signal quality (occurs only during pulse)
YPRF:	Yellow Perfusion	– Amplitude representation of medium signal quality (occurs only during pulse)
GPRF:	Green Perfusion	– Amplitude representation of high signal quality (occurs only during pulse)
SYNC:	Frame Sync	(occurs 1 of 25)

Generic HR Format:

	7	6	5	4	3	2	1	0
HR MSB	X	X	X	X	X	X	HR8	HR7

	7	6	5	4	3	2	1	0
HR LSB	X	HR6	HR5	HR4	HR3	HR2	HR1	HR0

Generic SpO_2 Format:

	7	6	5	4	3	2	1	0
SPO_2	X	SP6	SP5	SP4	SP3	SP2	SP1	SP0

HR: 4-beat average values in standard mode.

SpO_2: 4-beat average values in standard mode.

HR-D: 4-beat average displayed values in display mode

SpO_2-D: 4-beat average displayed values in display mode

SpO_2 Fast: Non-slew limited saturation with 4-beat averaging in standard mode.

SpO_2: B-B: Un-averaged, non-slew limited, beat to beat value in standard mode.

E-HR: 8-beat average values in standard mode.

E-SpO_2: 8-beat average values in standard mode.

E-HR-D: 8-beat average displayed values in display mode

E-SpO_2-D: 8-beat average displayed values in display mode

PLETH: 8-Bit Plethysmographic Pulse Amplitude

SREV: Oximeter Firmware Revision Level

CHK: Checksum = (Byte 1) + (Byte 2) + (Byte 3) + (Byte 4) modulo 256

When SpO_2 and HR cannot be computed, the system will send a missing data indicator. For missing data, the HR equals 511 and the SpO_2 equals 127.

Mode	In Track	Out of Track
Standard	SpO_2 and pulse rate updated on every pulse beat	SpO_2 and Heart Rate values are set to missing data values and out of track indicated.
Display	SpO_2 and pulse rate updated every 1.5 seconds	Last in track values transmitted for ten seconds and out of track indicated. After ten seconds, values are set to missing data values.

Serial Data Format #7:

Packet Description

A frame consists of 5 bytes; a packet consists of 25 frames. Three packets (75 frames) are transmitted each second.

Frame

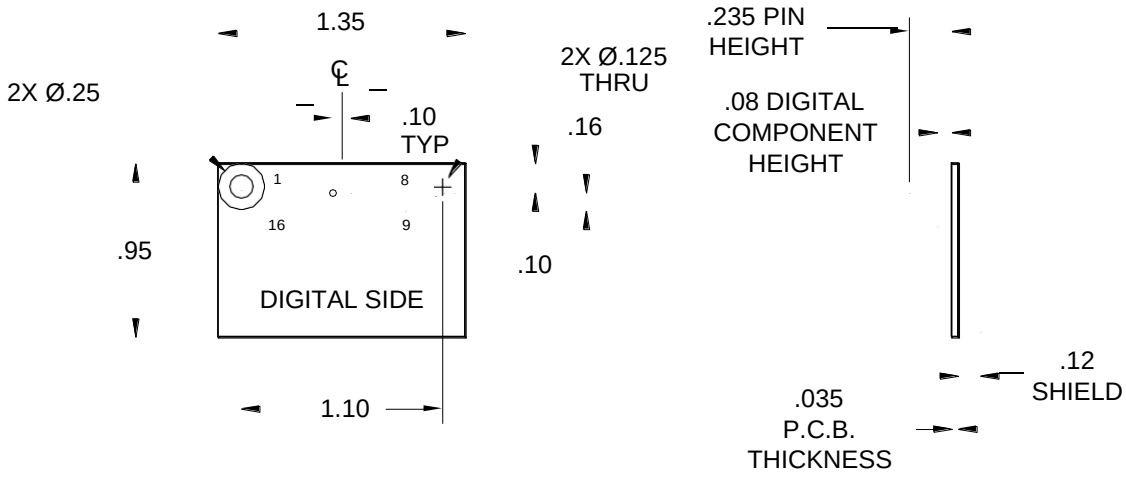
Packet

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
1	STATUS	PLETH (MSB)	PLETH (LSB)	HR MSB	CHK
2	STATUS	PLETH (MSB)	PLETH (LSB)	HR LSB	CHK
3	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2	CHK
4	STATUS	PLETH (MSB)	PLETH (LSB)	REV	CHK
5	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
6	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
7	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
8	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
9	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2-D	CHK
10	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 Fast	CHK
11	STATUS	PLETH (MSB)	PLETH (LSB)	SpO_2 B-B	CHK
12	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
13	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
14	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR MSB	CHK
15	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR LSB	CHK
16	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2	CHK
17	STATUS	PLETH (MSB)	PLETH (LSB)	E-SpO_2-D	CHK
18	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
19	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
20	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D MSB	CHK
21	STATUS	PLETH (MSB)	PLETH (LSB)	HR-D LSB	CHK
22	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D MSB	CHK
23	STATUS	PLETH (MSB)	PLETH (LSB)	E-HR-D LSB	CHK
24	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK
25	STATUS	PLETH (MSB)	PLETH (LSB)	reserved	CHK

Notes:

- Refer to DF#2 for definitions on Status Byte, Byte 4 and Checksum
- PLETH(MSB) and PLETH(LSB) make up a 16-bit pleth (ex. PLETH(MSB)*256 + PLETH(LSB))

(DIMENSIONS IN INCHES)



OUTPUTS:
J1-11 = Serial Output

INPUTS:
J1-16 = +3.3VDC (3.2V to 3.5V), 50mV max. ripple
+5.0VDC ±0.25VDC, 50mV max ripple
J1-15 = Ground
J1-13 = Reset (optional) [1]
J1-09 = Serial Data Format Switch [2]
J1-10 = Serial Input (future use)[1]

SENSOR CONNECTION:

	9 Pin	UNI EXT Patient Extension	
J1	D-Sub	Cable Color	Description
J1-01	7	Cable Shield	Cable shield
J1-02	5	Coax Signal	Photo diode signal
J1-04	9	Coax Shield	Photo diode bias
J1-05	6	Green	Sensor type #1 line
J1-06	2	Red	LED drive line
J1-07	3	Black	LED drive line
J1-12	1	Yellow	Sensor type #2/1 wire

PIN ASSIGNMENTS:
J1-01 = Cable shield
J1-02 = Photo diode signal
J1-03 = Signal shield
J1-04 = Photo diode bias
J1-05 = Sensor type #1 line
J1-06 = LED drive line
J1-07 = LED drive line
J1-08 = Reserved [3]
J1-09 = Serial Data Format Switch
J1-10 = Serial Input (future use)[1]
J1-11 = Serial Output
J1-12 = Sensor type #2/1 wire
J1-13 = Reset (Optional) [1]
J1-14 = Photo Plethysmogram Digital Output [1,4]
J1-15 = Ground
J1-16 = +3.3VDC (3.2V to 3.5V), 50mV max. ripple
+5.0VDC ±0.25VDC, 50mV max ripple

Notes:
1. Pins may be left un-terminated
2. Pin 9 may be left un-terminated for Data Format #2
3. Pins marked “Reserved” should be left un-terminated
4. Additional details can be found in technical note T-0604.

Indications for Use

The OEM III module is intended to provide medical device manufacturers with a small, low-power device that can be easily integrated into a host device. The module measures functional oxygen saturation of arterial hemoglobin (%SpO_2) and pulse rate (BPM) for adult, pediatric, infant and neonatal patients. When integrated with a medical device manufacturer's host system, the OEM III module may be used in any environment where pulse oximetry measurements are made.

Contraindications

- Do not use this device in an MR environment.
- Explosive Hazard: Do not use this device in an explosive atmosphere or in the presence of flammable anesthetics or gases
- This module does not meet defibrillation-proof requirement per IEC 60601-1.

Warnings

- Inspect the sensor application site at least every 6 to 8 hours to ensure correct sensor alignment and skin integrity. Patient sensitivity may vary due to medical status or skin condition. Discontinue use of adhesive tape strips if the patient exhibits an allergic reaction to the adhesive material.
- Avoid excessive pressure to the sensor application site as this may cause damage to the skin beneath the sensor.
- To avoid patient injury, use only with Nonin-branded PureLight pulse oximeter sensors. These sensors are manufactured to meet the accuracy specifications for Nonin pulse oximeters. Using other manufacturers' sensors can result in inaccurate pulse oximeter performance.
- Loss of monitoring can result if any objects hinder the pulse measurement. Ensure that no blood flow restrictors (e.g., blood pressure cuff) hinder pulse measurements.
- As with all medical equipment, carefully route cables and connections to reduce the possibility of entanglement, strangulation, or injury to the patient.
- Operation of this module below the minimum amplitude of 0.3% modulation may cause inaccurate results.
- The use of accessories, sensors, and cables other than those specified by Nonin (may result in increased emission and/or decreased immunity of this device.
- Do not use a damaged sensor. If the sensor is damaged in any way, discontinue use immediately and replace the sensor.

⚠ Cautions

- The accuracy of the SpO_2 measurement may be affected if the total sensor cable length (including extension cables) is greater than 3 meters.
- Follow local, state, or national governing ordinances and recycling instructions regarding disposal or recycling of the device and device components.
- In compliance with the European Directive on Waste Electrical and Electronic Equipment (WEEE) 2002/96/EC, do not dispose of this product as unsorted municipal waste. This device contains WEEE materials; please contact your distributor regarding take-back or recycling of the device.
- This pulse oximeter module is designed to determine the percentage of arterial oxygen saturation of functional hemoglobin. Significant levels of dysfunctional hemoglobin, such as methemoglobin, might affect the accuracy of the measurement. Factors that may degrade pulse oximeter performance or affect the accuracy of the measurement include the following: excessive ambient light, excessive motion, electrosurgical interference, blood flow restrictors (arterial catheters, blood pressure cuffs, infusing lines, etc.), moisture in the sensor, improperly applied sensor, incorrect sensor type, poor pulse quality, venous pulsations, anemia or low hemoglobin concentrations, cardiogreen or other intravascular dyes, carboxyhemoglobin, methemoglobin, dysfunctional hemoglobin, artificial nails or fingernail polish, or a sensor not at heart level.

⚠ Cautions
• This device has motion tolerant software that minimizes the likelihood of motion artifact being misinterpreted as good pulse quality. In some circumstances, however, this device may still interpret motion as good pulse quality. This covers all available outputs (i.e. SpO_2, HR, PLETH, PPG).
• This equipment complies with IEC EN 60601-1-2 for electromagnetic compatibility for medical electrical equipment and/or systems. This standard is designed to provide reasonable protection against harmful interference in a typical medical installation. However, because of the proliferation of radio-frequency transmitting equipment and other sources of electrical noise in healthcare and other environments, it is possible that high levels of such interference due to close proximity or strength of a source might disrupt the performance of this device. Medical electrical equipment needs special precautions regarding EMC, and all equipment must be installed and put into service according to the EMC information specified in this manual.
• Portable and mobile RF communications equipment may affect medical electrical equipment.
• Oximeter readings may be affected by the use of an electrosurgical unit (ESU)
• The oximeter sensor may not work on cold extremities due to reduced circulation. Warm or rub the finger to increase circulation, or reposition the sensor.
• A functional tester cannot be used to assess the accuracy of a pulse oximeter monitor or sensor.

For more information about required safety and regulatory requirements for pulse oximeters, refer to ISO 80601-2-61 and IEC 60601-1. Additional safety information can be found in the labeling provided with each Nonin sensor.

Equipment Response Time

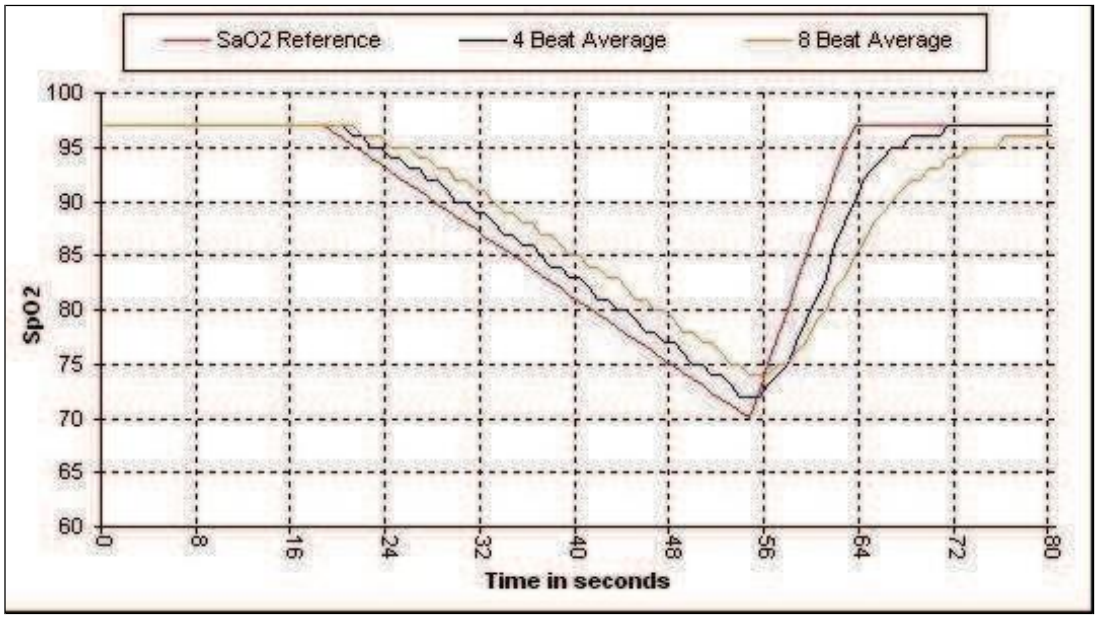
SpO_2 Values	Average	Latency
Standard/Fast Averaged SpO_2	4 beat exponential	2 beats
Extended Averaged SpO_2	8 beat exponential	2 beats

Pulse Rate Values	Average	Latency
Standard/Fast Averaged Pulse Rate	4 beat exponential	2 beats
Extended Averaged Pulse Rate	8 beat exponential	2 beats

Example – SpO_2 Exponential Averaging

SpO_2 decreases 0.75% per second (7.5% over 10 seconds)

Pulse Rate – 75 BPM



Specific to this example:

- The response of the 4-beat average is 1.5 seconds.
- The response of the 8-beat average is 3 seconds.

Accessories

The following Nonin accessories may be used with the OEM III module. See the respective sensor instructions for detailed information regarding specified sensor use (patient population, body/tissue, and application).

Model Number	Description
8000AA	Adult Articulated Internal Spring Finger Clip, 3 feet / 1 meter cable
8000AP	Pediatric External Spring Finger Clip, 3 feet / 1 meter cable
8000J	Adult Flex, 3 feet / 1 meter cable
8001J	Neonatal Flex, 3 feet / 1 meter cable
8008J	Infant Flex, 3 feet / 1 meter cable
8000Q2	Ear Clip, 3 feet / 1 meter cable
8000R	Reflectance, 3 feet / 1 meter cable
8000SS	Sensor, Reusable, Soft, Small, 1 meter cable
8000SM	Sensor, Reusable, Soft, Medium, 1 meter cable
8000SL	Sensor, Reusable, Soft, Large, 1 meter cable
7000A	Flexi-Form III Adult, 3 feet / 1 meter cable, 10-pack
7000P	Flexi-Form III Pediatric, 3 feet / 1 meter cable, 10-pack
7000I	Flexi-Form III Infant, 3 feet / 1 meter cable, 10-pack
7000N	Flexi-Form III Adult, 3 feet / 1 meter cable, 10-pack
6000CA	Sensor, Disposable, Adult, 45 cm cable
6000CP	Sensor, Disposable, Pediatric, 45 cm cable
6000CI	Sensor, Disposable, Infant, 90 cm cable
6000CN	Sensor, Disposable, Neonate, 90 cm cable
6500SA	Durafoam Sensor, Disposable, Standard, 1 meter cable
6500MA	Durafoam Sensor, Disposable, Small, 1 meter cable
UNI-RA-0	7.5” 90-degree Patient Cable
UNI EXT-X	Patient Extension Cable (1 or 3 meter with 3.3V input, 1,3,6 or 9 meter with +5V input)

Testing Summary

SpO_2 accuracy, motion and low perfusion testing was conducted by Nonin Medical, Incorporated as described below.

SpO_2 Accuracy Testing

SpO_2 accuracy testing is conducted during induced hypoxia studies on healthy, non-smoking, light-to-dark-skinned subjects during motion and no-motion conditions in an independent research laboratory. The measured arterial hemoglobin saturation value (SpO_2) of the sensors is compared to arterial hemoglobin oxygen (SaO_2) value, determined from blood samples with a laboratory co-oximeter. The accuracy of the sensors in comparison to the co-oximeter samples measured over the SpO_2 range of 70 – 100%. Accuracy data is calculated using the root-mean-squared (A_{rms} value) for all subjects, per ISO 80601-2-61, SpO_2 Accuracy of Pulse Oximeter Equipment.

Pulse Rate Motion Testing

This test measures pulse rate accuracy with motion artifact simulation introduced by a pulse oximeter tester. This test determines whether the oximeter meets the criteria of ISO 80601-2-61 for pulse rate during simulated movement, tremor, and spike motions.

Low Perfusion Testing

This test uses an SpO_2 Simulator to provide a simulated pulse rate, with adjustable amplitude settings at various SpO_2 levels.. The module must maintain accuracy in accordance with ISO 80601-2-61 for pulse rate and SpO_2 at the lowest obtainable pulse amplitude (0.3% modulation).

Manufacturer's Declaration

See the following tables for specific information regarding this module's compliance to IEC 60601-1-2.

Table 1: Electromagnetic Emissions

Emissions Test	Compliance	Electromagnetic Environment— Guidance
This module is intended for use in the electromagnetic environment specified below. The customer and/or user of this device should ensure that it is used in such an environment.		
RF Emissions CISPR 11	Group 1	This module uses RF energy only for its internal function. Therefore, its RF emissions are very low and are not likely to cause any interference in nearby electronic equipment.
RF Emissions CISPR 11	Class B	This module is suitable for use in all establishments, including domestic and those directly connected to the public low-voltage power supply network that supplies buildings used for domestic purposes.
Harmonic Emissions IEC 61000-3-2	N/A	
Voltage Fluctuations/ Flicker Emissions IEC 61000-3-3	N/A	

Table 2: Electromagnetic Immunity

Immunity Test	IEC 60601 Test Level	Compliance Level	Electromagnetic Environment— Guidance
This module is intended for use in the electromagnetic environment specified below. The customer and/or user of this device should ensure that it is used in such an environment.			
Electrostatic Discharge (ESD) IEC 61000-4-2	±6 kV contact ±8 kV air	±6 kV contact ±8 kV air	Floors should be wood, concrete, or ceramic tile. If floors are covered with synthetic material, the relative humidity should be at least 30%.
Electrical Fast Transient/Burst IEC 61000-4-4	N/A	N/A	Mains power quality should be that of a typical commercial or hospital environment.
Surge IEC 61000-4-5	N/A	N/A	Mains power quality should be that of a typical commercial or hospital environment.
Voltage dips, short interruptions, and voltage variations on power supply input lines IEC 61000-4-11	N/A	N/A	Mains power quality should be that of a typical commercial or hospital environment. If the user of the module requires continued operation during power mains interruptions, it is recommended that the device be powered from an uninterruptible power supply or battery pack.
Power Frequency (50/60 Hz) Magnetic Field IEC 61000-4-8	3 A/m	3 A/m	Power frequency magnetic fields should be at levels characteristic of a typical location in a typical commercial or hospital environment.
Note: U_T is the AC mains voltage before application of the test level.			

Table 3: Guidance and Manufacturer's Declaration—Electromagnetic Immunity

Immunity Test	IEC 60601 Test Level	Compliance Level	Electromagnetic Environment—Guidance
This module is intended for use in the electromagnetic environment specified below. The customer and/or user of this module should ensure that it is used in such an environment.			
Portable and mobile RF communications equipment should be used no closer to any part of the module, including cables, than the recommended separation distance calculated from the equation applicable to the frequency of the transmitter.			
Conducted RF IEC 61000-4-6	3 Vrms 150 kHz to 80 MHz	3 V	**Recommended Separation Distance** $d = 1.17\sqrt{P}$
Radiated RF IEC 61000-4-3	3 V/m 80 MHz to 2.5 GHz	3 V/m	$d = 1.17\sqrt{P}$ 80 MHz to 800MHz $d = 2.33\sqrt{P}$ 800MHz to 2.5 GHz where *P* is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer and *d* is the recommended separation distance in meters (m). Field strengths from fixed RF transmitters, as determined by an electromagnetic site survey,[a] should be less than the compliance level in each frequency range.[b] Interference may occur in the vicinity of equipment marked with the following symbol: 

Notes:

- At 80 MHz and 800MHz, the separation distance for the higher frequency range applies.
- These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects, and people.

a) Field strengths from fixed transmitters, such as base stations for radio (cellular/cordless) telephones and land mobile radios, amateur radio, AM and FM radio broadcast and TV broadcast cannot be predicted theoretically with accuracy. To assess the electromagnetic environment due to fixed RF transmitters, an electromagnetic site survey should be considered. If the measured field strength in the location in which the device is used exceeds the applicable RF compliance level above, the device should be observed to verify normal operation. If abnormal performance is observed, additional measures may be necessary, such as reorienting or relocating the module.

b) Over the frequency range 150 kHz to 80 MHz, field strengths should be less than 3 V/m.

Table 4: Recommended Separation Distances

The following table describes the recommended separation distances between portable and mobile RF communications equipment and this module.

This module is intended for use in an electromagnetic environment in which radiated RF disturbances are controlled. Customers or users of this module can help prevent electromagnetic interference by maintaining a minimum distance between portable and mobile RF communication equipment (transmitters) and the module as recommended below, according to maximum output power of the communications equipment.			
	Separation Distance According to Frequency of Transmitter		
Rated Maximum Output Power of Transmitter W	150 kHz to 80 MHz $d = 1.17\sqrt{P}$	80 MHz to 800 MHz $d = 1.17\sqrt{P}$	800 MHz to 2.5 GHz $d = 2.33\sqrt{P}$
0.01	0.12	0.12	0.23
0.1	0.37	0.37	0.74
1	1.2	1.2	2.3
10	3.7	3.7	7.4
100	12	12	23
For transmitters rated at a maximum output power not listed above, the recommended separation distance *d* in meters (m) can be estimated using the equation applicable to the frequency of the transmitter, where *P* is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer.			
Notes: • At 80 MHz and 800MHz, the separation distance for the higher frequency range applies. • These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects, and people.			