Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

MONITORIZACIÓN ECO-EFICIENTE Y PARALELA CON DISPOSITIVOS MÓVILES

D. VILLALÓN FERNÁNDEZ, Alberto

TUTOR: D. RANILLA PASTOR, José

FECHA: Julio de 2020

Contenido

Índice de Figuras

Índice de Tablas

1. Motivación

La democratización, a nivel mundial y en todos los estratos sociales, de los dispositivos conocidos como llevables o *wearables*, tales como pulseras de actividad o *smartwatches*, es un hecho incontestable. Estos dispositivos cuentan con un conjunto "básico" de sensores destinados a monitorizar el movimiento del dispositivo, su posición, características del entorno, etc. En los últimos años, además, los fabricantes han comenzado a incorporar un conjunto adicional de sensores relacionados con la *Actividad*/*Salud* para medir la frecuencia cardiaca, realizar electrocardiogramas (*Electrocardiogram*, ECGs), conocer la saturación periférica de oxígeno en sangre (*Peripheral Oxygen Saturation*, SpO2), etc.

Así, dependiendo de los sensores incorporados en el wearable es posible obtener gran cantidad de información y, también, realizar una monitorización continua de ciertos parámetros, con el fin de mejorar la calidad de vida de las personas. Como se indica en [1], el ámbito de aplicación de estos dispositivos es muy diverso. En medicina se utilizan para monitorizar el estado de salud del paciente, permitiendo prevenir enfermedades, contribuyendo a la detección temprana de posibles dolencias, etc. En profesiones de riesgo son una herramienta inestimable de prevención y ayuda para evitar accidentes laborales. En el ámbito del deporte, tanto profesional como *amateur*, son capaces de medir diferentes parámetros de la actividad física, como por ejemplo distancias recorridas, frecuencia cardiaca instantánea, etc., y, con los datos recogidos, hacer recomendaciones para evitar lesiones, proponer ritmos de trabajo personalizados y adaptativos o fomentar el aumento de actividad. En el ámbito de la educación se utilizan para monitorizar los niveles de estrés de los estudiantes y, de esta forma, poder elaborar planes de estudio personalizados. Nótese que los mecanismos de recomendación/alerta son trasversales, aplicables a distintos ámbitos, con solo modificar el enfoque o los umbrales. Por ejemplo, la velocidad en la recuperación de la frecuencia cardiaca después de un esfuerzo es útil en medicina, deporte, estudio, trabajo, etc.

Los llevables estándar o de uso masivo son dispositivos periféricos en lo relativo a la posición del cuerpo donde realizan las mediciones, generalmente en una de las muñecas, y suelen recurrir a sensores ópticos para capturar la información relacionada con la salud. Existe otro conjunto de wearables más específicos, tal vez experimentales y menos populares, catalogados como *centrales*, que se ubican en el punto justo del cuerpo donde se produce la información que se desea monitorizar (bandas, camisetas, etc.). Hay un tercer grupo, más preciso, como los tensiómetros o el propio *Omron HeartGuide* que son tanto periféricos como centrales, pero altamente intrusivos. Estos dos últimos grupos usan sensores más específicos como electrodos, pulseras hinchables, etc.

Este trabajo de investigación pone el foco en los llevables pertenecientes al primer grupo por su alta variedad y disponibilidad, por ser económicamente asequibles, por compartir tecnología sensórica y de comunicaciones, por ser "abiertos" en términos del acceso a sus funcionalidades desde programas/códigos externos, por no ser intrusivos y, sobre todo, por ser los wearables que la inmensa mayoría de la población usa o está acostumbrada a usar (no sufren rechazo independiente de la posición social, económica, la edad, etc.).

Obviando los dispositivos sumamente específicos, en muchos casos solo aptos para ser usados por personal cualificado (p. ej. los sistemas de *Monitorización Ambulatoria de la Presión Arterial*, MAPA), el *Sistema Circulatorio o Cardiovascular* es al que más atención prestan los llevables. Esto es así porque en la actualidad existe una gran variedad de sensores precisos, de bajo coste, pequeño tamaño, de reducido consumo y fácilmente integrables en los wearables. También porque los avances de los investigadores han permitido construir modelos, métodos y algoritmos programables para gestionar la información bruta (*RAW*) obtenida por los sensores y para sintetizar conocimiento a partir de ella. No se debe olvidar, además, que las enfermedades cardiacas son una de las

principales causas de muerte en el mundo, siendo la primera en España con casi un 30% del total de fallecimientos [2].

Pero el sistema circulatorio no solo proporciona información susceptible de ser capturada por sensores ópticos o de captación de impulsos eléctricos, también *emite* ondas sonoras, un tipo de información llena de matices. La salud del *Sistema Respiratorio* también es de vital importancia y su monitorización proactiva relevante para la detección temprana de dolencias respiratorias. Más aún, el análisis conjunto de la información ofrecida por ambos sistemas amplia el espectro de dolencias o enfermedades detectables.

El tratamiento de las señales sonoras no es habitual en los llevables. En primer lugar, porque la sensórica actual no se encuentra tan avanzada en términos de integración y no resulta sencillo incorporarla sin ser intrusivo. Y en segundo, porque el tratamiento del sonido es más complejo y precisa de mayor capacidad computacional (p. ej. hay que realizar separación de fuentes sonoras para aislar los sonidos del corazón de los del pulmón, y viceversa). No obstante, la inclusión de este tipo de información en los wearables está cercana. Por ejemplo, el *Apple Watch* ya incorpora funcionalidades para lanzar alertas de entornos ruidosos. Más aún, son varios los grupos de investigación que están iniciando estudios para, partiendo de ciertos sonidos del sistema respiratorio (sibilancias, estertores, *crackles*, etc.) y aplicando técnicas de Inteligencia Artificial (*Deep Learning* normalmente), detectar si el paciente está o no infectado por el virus *SARS-CoV-2* (pandemia conocida popularmente como COVID-19).

Por todo lo dicho, en este Trabajo Final de Grado se estableció como objetivo:

"*Diseñar e implementar un prototipo que, combinando señales heterogéneas, permita la monitorización continua de ciertos parámetros de los Sistemas Cardiovascular y Respiratorio*"

El prototipo usaría los sensores ópticos de un llevable periférico estándar o de consumo (pulseras, relojes, etc.) para detectar los niveles de actividad y obtener cierta información relativa al Sistema Circulatorio, y un micrófono (o varios), *pegado* al pecho, gobernado por un microcontrolador que captura el audio del corazón y de los pulmones (sensor central). La información, periférica y central, se procesará y combinará para la toma de decisiones. La fusión de información, y su procesado, es susceptible de realizarse en el *wearable* periférico si dispone de suficiente capacidad computacional (p. ej. si es un smartwatch de alta gama y última generación), en otro dispositivo como puede ser un smartphone o, si procede y es posible, usando servicios en la nube (*Cloud*).

Sin embargo, la irrupción del SARS-CoV-2 y la posterior pandemia mundial, unido al *Estado de Alarma* decretado en España, ha trastocado el planteamiento inicial. La adquisición del material para construir el prototipo ha sido compleja y los plazos de entrega se han dilatado enormemente. El Estado de Alarma impidió el acceso a los laboratorios de la Universidad de Oviedo para la construcción y validación del prototipo destinado a la captura y tratamiento de las señales sonoras. De igual modo, el Estado de Alarma ha limitado la movilidad y los contactos presenciales, impidiendo la realización de pruebas de campo completas y representativas (obtener medidas con los distintos sensores de diversos estados de actividad de varias personas de ambos géneros y franjas de edad).

Ante la incertidumbre y el desconocimiento sobre cuándo la pandemia, y el Estado de Alarma, remitirían se optó por replantear los objetivos del trabajo, para evitar perjuicios mayores como, por ejemplo, no poder egresarse a tiempo para seguir avanzado en la formación académica.

El nuevo enfoque del Trabajo Final de Grado (TFG) es:

- Realizar trabajos para monitorizar la actividad y ciertos parámetros del Sistema Cardiovascular con llevables periféricos usando sensores ópticos. Una mínima labor de campo: obtención y análisis de datos reales.

- Diseñar un prototipo central para la captura de audio. Pruebas de funcionamiento y comprobación de que cumple los requisitos necesarios para una potencial, y futura, integración en un sistema global.
- Implementar algoritmos eficientes para procesar, detectar y separar en tiempo real audio sintético (procedente de fuentes externas) de corazón y pulmón.

En definitiva, abordar el trabajo planteado inicialmente, pero sin integrar los sistemas y sin profundizar en la validación y corrección de los modelos y métodos usados fruto de la investigación realizada.

Por último, resaltar que este TFG consta, además de este documento, de los manuales para los usuarios y los programadores, los códigos fuente de las implementaciones realizadas, los binarios de las aplicaciones desarrolladas para llevables, los datos obtenidos en la labor de campo realizada y los conjuntos de ejemplos sintetizados a partir de ellos, los archivos con el audio utilizado y algunos ficheros más con indicaciones o explicaciones. Los documentos PDF, esta memoria y los manuales, están disponibles en la plataforma de la Universidad de Oviedo habilitada a tal efecto, mientras que al resto de elementos del TFG se puede acceder desde la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*.

2. Introducción

Las Enfermedades Cardiacas y Cardiovasculares son la principal causa de muerte en España con casi un 30% del total de fallecimientos [2]. Se sitúa por delante de otras enfermedades como el cáncer o las Enfermedades del Sistema Respiratorio. Según la Organización Mundial de la Salud (OMS), en su último estudio, se calcula que fallecieron 17,7 millones de personas en el año 2015 por enfermedades cardiovasculares, lo que supone un 31% del total de fallecimientos [3]. Además, la OMS establece que para las personas con un alto riesgo cardiovascular es muy importante realizar una detección precoz, para implantar tratamientos efectivos lo antes posible y, de esta forma, reducir la probabilidad de fallecimiento.

Para realizar un primer diagnóstico el método más utilizado es la auscultación, ya que es barato y no invasivo, aunque cuenta con la desventaja de depender de la experiencia y la capacidad de apreciación de los facultativos. Normalmente la auscultación se complementa con otro tipo de pruebas no invasivas, como el electrocardiograma que recoge la actividad eléctrica del corazón en diferentes localizaciones de la superficie corporal. Existen otro tipo de pruebas como la ecografía, la resonancia magnética o el escáner que ya requieren una inversión económica elevada. Todos ellos son métodos que precisan de una planificación y de la intervención de personal cualificado (médicos, enfermeros, auxiliares, etc.) y no son aptos para una monitorización continua. Por lo general son reactivos (no proactivos o preventivos) y se realizan bajo demanda ante la sospecha de la existencia de alguna anomalía.

Debido a la importancia de la detección precoz de enfermedades cardiacas o cardiovasculares cada vez son más habituales los sistemas de monitorización de la frecuencia cardiaca (*Heart Rate Monitoring System*, HRMS). Como se indica en [4], una anomalía en la frecuencia cardiaca (*Heart Rate*, HR) puede ser síntoma de una enfermedad grave. Un ejemplo típico de HRMS es el *Holter*, que registra los ritmos cardíacos de forma

continua durante 24 o 48 horas de actividad normal. Finalizado el periodo de observación, los datos registrados son interpretados por los especialistas. Aunque los Holter han evolucionado fuertemente, son muy precisos y capturan gran cantidad de información, presentan algunas limitaciones, como por ejemplo escasa autonomía (bien por batería, bien por capacidad de almacenamiento, o bien por ambas) o distorsión de sus medidas fruto de campos magnéticos, zonas del alto voltaje, etc. Pero, tal vez, su mayor limitación es el "factor de rechazo"; no son cómodos, lo que influye especialmente en la información capturada durante los periodos de descanso (el paciente no descansa como en condiciones normales, está pendiente del dispositivo).

El nivel de actividad física es otro factor a tener en consideración cuando se trata de medir la frecuencia cardiaca, dado que incide directamente en la HR. De esta forma, si una persona se encuentra realizando una actividad física con una cierta exigencia (p. ej. correr, nadar, etc.) su HR será más elevada que en condiciones de reposo. Además, según sea el patrón de recuperación de la HR es posible que exista una determinada dolencia. En consecuencia, es necesario conocer el contexto, nivel de actividad física, existente cuando se realizan las mediciones.

Como se ha comentado, la auscultación es el método más utilizado a la hora de realizar un primer diagnóstico. Lo que el facultativo hace es *interpretar* los sonidos que percibe (oye), al objeto de estimar la frecuencia de trabajo del corazón y detectar la existencia, o no, de anomalías. Pero los sonidos que percibe no son solo del corazón, son una mezcla de "ruidos" del cuerpo, donde los procedentes del Sistema Respiratorio son importantes dada su naturaleza y la posición de los pulmones respecto al corazón, y de las condiciones acústicas ambientales. Disponer de dispositivos que capturen, filtren, separen, etc. los distintos sonidos en tiempo real y que ofrezcan información con valor añadido (p. ej. que indiquen la HR, la frecuencia respiratoria, sibilancias, etc.) es, evidentemente, un avance cualitativo. Estos dispositivos, obviamente, ya existen, sobre todo en entornos ambulatorios y/o hospitalarios, pero en general adolecen de los mismos problemas que el

Holter: son herramientas complejas, altamente especializadas, costosas, pensadas para su uso/manipulación por expertos y no buscan la integración con otros dispositivos de uso común por parte de los pacientes.

Si el usuario (paciente) dispusiese de un dispositivo mínimamente intrusivo, con una autonomía razonable, que permitiese la monitorización continua, que se integrase fácilmente con la electrónica de consumo (p. ej. smartphone), aunque fuese menos preciso, mejoraría su calidad de vida. Si, además, la información aportada por dicho dispositivo gozase de la aceptación por parte de los sistemas de salud se elevaría sustancialmente el nivel de proactividad y la detección precoz de dolencias. En la actualidad el *Apple Watch Series 5* es el más avanzado, cuyas mediciones son *aceptadas* por la *American Heart Association* y la Unión Europea, entre otros países o asociaciones. Detecta caídas, monitoriza de forma continua la HR con alarmas de defecto/exceso de pulsaciones, puede realizar ECG, detecta e informa de entornos ruidosos, etc. Sin embargo, es un dispositivo costoso, no pertenece al ecosistema dominante (Android) y su integración fuera del *mundo* Apple no es sencilla. Y, aunque el Apple Watch y otros pueden capturar y tratar audio, son llevables periféricos, no tienen acceso a los sonidos que los facultativos detectan en la auscultación. Para ello necesitarían complementos hardware/software que, por ahora, no están disponibles.

Y es, precisamente, esta carencia el foco o núcleo de esta propuesta de investigación. En este Trabajo Final de Grado (TFG en adelante) se propone:

- Diseñar e implantar un HRMS sencillo, apoyado por un módulo capaz de diferenciar distintos niveles de actividad física, para un amplio espectro de wearables periféricos. La arquitectura debe ser flexible, permitiendo que la información sea procesada en el propio wearable, si dispone de capacidades para ello, o enviada a otro dispositivo para su tratamiento.
- Diseñar y construir un prototipo central (micrófono + microcontrolador) que capture los sonidos del corazón y de los pulmones.

- Aplicar al audio capturado algoritmos de separación de fuentes para que, con posterioridad, se pueda inferir información de valor añadido (HR, frecuencia respiratoria, etc.) o, simplemente, que el facultativo pueda escuchar ambas señales por separado en las mejores condiciones ambientales.
- Implementar los algoritmos sujetos a restricciones de tiempo real, para permitir la monitorización continua del audio. Se recurrirá al uso de técnicas de *High Performance Computing* y de Computación Paralela para maximizar el aprovechamiento de las capacidades del hardware donde se ejecuten.
- Considerar, en la medida de lo posible, la relación FLOP[1]/VATIO para reducir el consumo energético, dado que los desarrollos que se realicen deberán ser ejecutados en algún tipo de wearable, incluidos los smartphones.

Para concluir, en los siguientes apartados de esta introducción se incluyen algunas nociones básicas del sonido, así como la caracterización de los sonidos cardio-pulmonares.

2.1 Características básicas del sonido

El sonido consiste en la propagación de ondas mecánicas generadas como resultado de la vibración de un cuerpo o foco [5]. Dichas ondas se propagan a través de un medio elástico en forma de ondas sonoras. La velocidad de propagación en el aire es, aproximadamente, de 343 m/s, y es un valor que depende del medio de transmisión. En general los medios que tienen las partículas que lo componen más juntas, permiten una mayor velocidad de transmisión. De esta forma, los medios sólidos tienen una velocidad de transmisión mayor que los medios líquidos, que a su vez tienen mayor velocidad de trasmisión que los medios gaseosos.

El sonido es una onda mecánica porque su energía no se puede transmitir a través del vacío, sino que necesita un medio físico y elástico para propagarse. Además, se engloba

[1] Operación en Punto Flotante, *Floating Point Operation* en inglés.

dentro de las ondas longitudinales ya que la vibración de las partículas del medio de propagación es paralelo a la dirección en la que se propaga la energía [5]. Las cuatro cualidades que definen un sonido son:

- **Altura o tono**. El sonido se produce por la vibración de un cuerpo. En función de la frecuencia con la que vibre, el sonido será más grave o más agudo. Una vibración más lenta conlleva una frecuencia menor y por tanto un sonido grave. Por otro lado, una vibración más rápida genera un sonido más agudo.
- **Intensidad**. Es la potencia acústica por unidad de superficie y permite diferenciar entre sonidos fuertes y sonidos débiles. Los factores que influyen en la intensidad del sonido son la amplitud de la vibración del cuerpo que genera el sonido, la superficie de ese cuerpo, la distancia al cuerpo y la naturaleza del medio de transmisión. Cuanto mayor es la amplitud de vibración del cuerpo mayor es la intensidad del sonido.
- **Timbre**. Cualidad que permite distinguir dos sonidos con la misma altura e intensidad.
- **Duración**. Intervalo de tiempo durante el cual el sonido se emite de forma continua.

El sonido está compuesto por ondas de diferentes frecuencias. Utilizando un espectro de frecuencias es posible visualizar las diferentes frecuencias que lo componen. Para ello es necesario utilizar la Transformada Rápida de Fourier (FFT) que se encarga de convertir una señal expresada en dominio del tiempo a una representación en dominio de la frecuencia. La Transformada Inversa Rápida de Fourier (IFFT) es la operación inversa que permite convertir una señal en dominio de la frecuencia a dominio del tiempo. De esta forma es posible, partiendo de una señal en domino del tiempo, realizar la transformación mediante la FFT para obtener la señal en dominio de la frecuencia y obtener la información necesaria para realizar su tratamiento. Posteriormente se puede volver a reconstruir la señal original utilizando la IFFT.

2.2 Sonidos de corazón

En este apartado se realiza una explicación del funcionamiento del aparato circulatorio de los seres humanos, y más concretamente del corazón [6]. El funcionamiento del corazón

se divide en cuatro fases en las que se generan diferentes sonidos que son útiles para obtener información sobre dicho órgano, y para diagnosticar posibles enfermedades.

El aparato circulatorio está formado por un conjunto de conductos que son las arterias, las venas y los capilares por los que circula la sangre. La sangre es impulsada por el corazón, que es un músculo situado entre ambos pulmones y por encima del diafragma.

El corazón presenta cuatro cavidades: dos aurículas y dos ventrículos (ver Figura 2.1). Para controlar la dirección de la sangre, y que siempre circule en la misma dirección, el corazón tiene cuatro válvulas. Entre las aurículas y los ventrículos se encuentra la válvula *mitral* y la válvula *tricúspide*. La primera se encarga de que la sangre no retorne desde el ventrículo izquierdo a la aurícula izquierda, mientras que la válvula tricúspide se encarga de que la sangre no retorne desde el ventrículo derecho a la aurícula derecha.

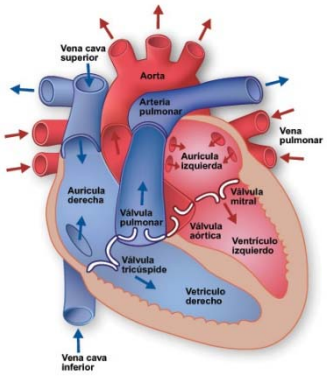

Figura 2.1. Partes del corazón. Imagen procedente de [6].

Entre los ventrículos y los grandes vasos se localizan las dos válvulas restantes, que son la válvula *sigmoidea aórtica* y la válvula *semilunar pulmonar* (ver Figura 2.1). La primera no permite que la sangre retorne desde la arteria aorta al ventrículo izquierdo. La segunda se encarga de que la sangre no retorne desde la arteria pulmonar hasta el ventrículo derecho.

Los acontecimientos que ocurren desde que se inicia el latido cardiaco hasta que se produce el inicio del siguiente latido se conocen como *ciclo cardiaco*. El bombeo de la sangre por el corazón genera el pulso que se puede detectar en diferentes zonas de cuerpo,

como en zonas próximas al corazón o en zonas donde se encuentran arterias como las radiales, carótidas o femorales.

Cada latido cardiaco se compone de una fase de *sístole* donde los ventrículos se contraen y una fase de *diástole* donde los ventrículos se dilatan y se llenan de sangre procedente de las aurículas. El ciclo cardiaco se compone de cuatro fases que son:

- **Fase 1 o Periodo de Llenado**. El ventrículo izquierdo se llena de sangre. La sangre pasa de la aurícula al ventrículo durante la diástole. Al aumentar la presión en el ventrículo, las válvulas situadas entre las aurículas y los ventrículos se cierran.
- **Fase 2 o Periodo de Contracción Isovolumétrica**. En este punto las válvulas aurico ventriculares se encuentran cerradas y las situadas entre los ventrículos y los grandes vasos todavía no se han abierto. El ventrículo se contrae y aumenta la presión en su interior ya que la sangre no puede salir en ninguna dirección.
- **Fase 3 o Periodo de Eyección**. La presión en el ventrículo sigue aumentando hasta que dicha presión es suficiente para abrir la válvula aórtica. En ese punto la sangre empieza a salir por la arteria aorta.
- **Fase 4 o Periodo de Eyección Volumétrica**. Al final de la sístole, cuando la presión del ventrículo izquierdo es menor que la presión de la arteria aorta, se cierra la válvula aórtica. En este punto comienza la diástole. Cuando la presión en el ventrículo es inferior a la de la aurícula se abre la válvula mitral y empieza de nuevo el ciclo cardiaco.

El corazón para contraerse necesita un impulso eléctrico que se produce en el propio corazón, concretamente en el nódulo sinusal localizado en la aurícula derecha. Es posible registrar la actividad eléctrica del corazón mediante un electrocardiograma para detectar posibles anomalías o enfermedades.

El corazón durante su funcionamiento normal emite dos sonidos diferenciados en cada ciclo cardiaco. El primer ruido cardiaco (R1) se corresponde con el cierre de las válvulas

situadas entre las aurículas y los ventrículos. Se produce durante la Fase 1 del ciclo cardiaco explicado anteriormente. Aunque las válvulas mitral y tricúspide no se cierran exactamente al mismo tiempo, se suele percibir como un sonido único. El segundo ruido cardiaco (R2) se corresponde con el cierre de las válvulas aórtica y pulmonar durante la Fase 4 del ciclo cardiaco. El cierre de la válvula aórtica produce un sonido más intenso y precoz que el cierre de la válvula pulmonar. El tiempo transcurrido entre R1 y R2 es el periodo necesario para que se complete la sístole, mientras que el tiempo transcurrido entre R2 y el siguiente R1 se corresponde con la diástole. Además de los dos ruidos cardiacos mencionados anteriormente, en ciertas situaciones es posible escuchar dos ruidos más. El tercer ruido cardiaco (R3) se produce después del R2 y es apreciable sobre todo en niños y adultos jóvenes. Está relacionado con un llenado brusco del ventrículo. El cuarto ruido cardiaco (R4) es menos frecuente y está relacionado con alguna patología. Se produce justo antes del R1 y está relacionado con una vibración que se produce durante la contracción auricular.

La *Frecuencia Cardiaca* (HR) es el número de veces por unidad de tiempo que se contrae el corazón. Normalmente se expresa en pulsaciones por minuto. Según la *American Heart Association* (AHA)[2] la frecuencia cardiaca en reposo de una persona sana debe estar entre 60 y 100 pulsaciones por minuto. Existen múltiples factores que afectan a la frecuencia cardiaca en reposo como el estrés, la ansiedad o la forma física en la que se encuentre la persona. Los atletas profesionales pueden llegar a tener una frecuencia cardiaca en reposo de 40 pulsaciones por minuto. Se ha demostrado que una frecuencia cardiaca elevada en reposo está relacionada con una baja forma física, una alta presión sanguínea y con el sobrepeso.

AHA define, además, la frecuencia cardiaca máxima (FC_m) cuando se está realizando una actividad física muy intensa como:

[2] www.heart.org

$$FC_m = 220 - e \quad (2.1)$$

donde e denota la edad de la persona.

Algunos de los trastornos más comunes relacionados con la frecuencia cardiaca son:

- **Taquicardia**. Consiste en una frecuencia cardiaca irregular y acelerada que puede oscilar entre los 100 y 175 pulsaciones por minuto en estado de reposo. Como se mencionó anteriormente, en el nódulo sinusal se genera el impulso eléctrico necesario para el correcto funcionamiento del corazón. Esta señal debe viajar desde las dos cavidades superiores del corazón hasta las cavidades inferiores. Cuando se produce fibrilación auricular la transmisión de esta señal se vuelve caótica y se empiezan a recibir un gran número de impulsos eléctricos que provocan un aumento significativo de la frecuencia cardiaca.
- **Bradicardia**. Consiste en una frecuencia cardiaca irregular y anormalmente baja con valores que se sitúan por debajo de las 60 pulsaciones por minuto en reposo. La bradicardia por sí sola no es un problema grave, pero puede ser el reflejo de otra dolencia más seria. En algunos casos puede requerir la implantación de un marcapasos en el corazón.

  Una frecuencia cardiaca en reposo por debajo de 60 pulsaciones por minuto es normal en personas jóvenes y atletas. En este último caso la frecuencia cardiaca puede llegar a bajar de las 40 pulsaciones por minuto sin representar un riesgo para la salud.

2.3 Sonidos de pulmón

Al igual que sucede con el corazón, los pulmones también producen diferentes sonidos durante su funcionamiento que pueden resultar útiles para obtener información sobre el estado de dicho órgano y realizar un diagnóstico médico. Los pulmones son los órganos del aparato respiratorio que se encargan de realizar la función de la respiración. Dichos órganos se localizan a cada lado del corazón protegidos por las costillas. En su interior se

produce el intercambio de gases donde la sangre recibe oxígeno procedente del aire inspirado y se desprende de dióxido de carbono que pasa al aire expirado.

La ventilación es el proceso por el cual entra aire rico en oxígeno en los pulmones y se expulsa aire con poco oxígeno y cargado de dióxido de carbono. El proceso se divide en dos fases, que son:

- **Inspiración**. Durante esta fase el aire con oxígeno entra en los pulmones desde el exterior a través de las vías aéreas, que son las diferentes partes por las que circula el aire hasta llegar a los pulmones. Están compuestas por las fosas nasales, la boca, faringe, glotis, epiglotis, laringe, tráquea, bronquios, bronquiolos y alveolo pulmonar. Durante esta fase el diafragma se contrae para permitir el aumento de la caja torácica reduciendo la presión interna en los pulmones. De esta forma el aire se introduce en los pulmones.
- **Expiración**. En esta fase se realiza el proceso opuesto a la inspiración. En este caso el diafragma se relaja provocando una reducción del volumen de la caja torácica y por tanto una contracción en los pulmones. Como consecuencia de ello la presión en el interior de dichos órganos aumenta siendo esta superior a la presión atmosférica. Así el aire empieza a salir de los pulmones a través de las vías aéreas hasta que la presión en su interior se iguala a la presión atmosférica. De esta forma el aire con dióxido de carbono es expulsado de los pulmones completando el proceso de la ventilación pulmonar.

Los diferentes sonidos pulmonares se producen principalmente al pasar el aire por las vías aéreas. Los sonidos auscultados en posiciones cercanas a la pared torácica se generan principalmente en los bronquios. Los sonidos que llegan a zonas periféricas tienen una menor frecuencia ya que los pulmones funcionan como un filtro para los sonidos de las frecuencias más altas. Los ruidos normales que se generan durante la respiración son los siguientes:

- **Ruido Traqueal**. Sonido que se escucha normalmente al realizar la auscultación en la zona de la tráquea. Es perceptible durante la inspiración y durante la expiración, y se produce cuando el aire atraviesa la tráquea.
- **Ruido Traqueobronquial**. Sonido muy parecido al ruido traqueal, pero de menor intensidad. La auscultación se realiza en el tórax y en la espalda.
- **Murmullo Pulmonar**. Sonido de baja intensidad que se ausculta en las zonas cercanas a los pulmones tanto en el tórax como en el costado y la espalda. Es el sonido que llega a la pared torácica al atravesar el aire las vías aéreas. El pulmón actúa de filtro para las frecuencias más altas.

Además de los sonidos normales, es posible percibir otro tipo de sonidos cuando existe alguna enfermedad en los pulmones [7]. Los más habituales son:

- **Crepitaciones**. Sonidos de poca intensidad y normalmente con una duración menor a 20ms que se producen durante la inspiración. Existen diferentes enfermedades, como la neumonía o la fibrosis pulmonar, que provocan la aparición de este tipo de sonido.
- **Sibilancias**. Son sonidos continuos de alta frecuencia parecidos a un silbido. Se produce por el paso del aire a través de las vías aéreas cuando presentan algún tipo de obstrucción.
- **Roncus**. Es un sonido que se produce, al igual que las sibilancias, por una obstrucción en las vías aéreas. Es un sonido de frecuencia más baja y similar a un ronquido. Aparece cuando se producen secreciones en los bronquios.

3. Estado del arte

3.1 Wearables para monitorizar aspectos sobre la salud

El número y tipo de sensores presentes en este tipo de dispositivos es muy variado. En [1] se indican los sensores habituales para detectar diferentes parámetros como la frecuencia cardiaca, la temperatura o la aceleración. En concreto, para detectar la frecuencia cardiaca se utilizan sensores de fotoconductividad, sensores infrarrojos y el acelerómetro. Para detectar la aceleración se utiliza el acelerómetro, y el giroscopio para gestos y movimientos.

Dentro de los llevables periféricos comerciales y de uso masivo, que son en los que este trabajo centra su atención, los más habituales son aquellos que el usuario lleva en la muñeca, existiendo dos grandes grupos: a) pulseras y b) relojes.

La mayor parte de pulseras comerciales disponen de un sensor óptico para medir la frecuencia cardiaca y un acelerómetro con el que pueden determinar el número de pasos que ha dado el usuario. Por ejemplo, la pulsera *Xiaomi Smart Band 4* añade a los sensores mencionados un giroscopio, un sensor de proximidad capacitivo y *Bluetooth* 5.0 de baja energía. *Fitbit Charge 3* agrega, respecto a la Xiaomi, un altímetro barométrico y la posibilidad de medir la saturación periférica de oxígeno en sangre (*Peripheral Oxygen Saturation*, SpO2) [8], multiplicando por 5 el precio de la primera. La pulsera *Garmin Vivosmart 4* tiene los mismos sensores que la Fitbit a un precio ligeramente inferior.

Respecto a los relojes inteligentes, algunos de ellos permiten instalar aplicaciones (Apps) que pueden, entre otras funcionalidades, recoger la información que proporcionan los diferentes sensores integrados en el reloj. Esta información se puede utilizar para diferentes finalidades como, por ejemplo, detectar el tipo de actividad física. En el mercado existen relojes que cuentan con diferentes sistemas operativos. El *Samsung Galaxy Watch*

usa el sistema operativo *Tizen OS* e incluye los sensores necesarios para este trabajo. El *Huawei Watch GT 2* lleva el sistema operativo *Lite OS*, pero no es posible desarrollar aplicaciones que se instalen en el dispositivo. El *Apple Watch 5*, tal vez el más avanzado, utiliza *watchOS* como sistema operativo. Por otro lado, hay relojes que llevan el sistema operativo *Wear OS*. Wear OS es el sistema operativo de Google pensado para wearables que usan *Android*. La primera versión basada en Android 4.4 fue lanzada en 2014 y desde entonces se han realizado 4 actualizaciones. Es posible desarrollar aplicaciones con *Android Studio* usando el SDK (*Software Development Kit*) de Android. Dichas aplicaciones son similares a las que se pueden desarrollar para los teléfonos con Android, pero cuentan con opciones de diseño diferentes para poder adaptarse a las pequeñas pantallas de los relojes.

Para seleccionar la pulsera y el reloj a utilizar en este trabajo se tuvieron en consideración los siguientes aspectos:

- Que disponga del conjunto de sensores mínimo necesario.
- Que su índice de penetración y aceptación fuese el más elevado posible.
- Que económicamente fuese asequible.
- Que fuese estándar, accesible y programable.
- La reputación y fiabilidad del fabricante.

Teniendo en consideración las restricciones mencionadas, los productos del fabricante Apple se consideraron no aptos por precio, gama e índice de penetración. Seguidamente se realizó un estudio de mercado analizando las diferentes opciones, siendo la decisión tomada:

- **Xiaomi Smart Band 4** como pulsera de actividad. Incorpora los sensores necesarios, tiene un precio muy bajo y se encuentra entre las más vendidas.
- Para el reloj inteligente se optó por aquellos con sistema operativo Wear OS. En [9] se puede encontrar una lista con los relojes compatibles con Wear OS. De todos ellos, una primera preselección quedó formada por *Ticwatch S2*, *Puma Smartwatch* y *Fossil Sport*. Los tres incorporan sensores como acelerómetro, giroscopio, sensor óptico

para medir la frecuencia cardiaca y GPS (Sistema de Posicionamiento Global, en inglés, *Global Positioning System*) para determinar la ubicación del dispositivo. Finalmente, el elegido fue el **Ticwatch S2** por ser el más barato.

Además de los dispositivos mencionados también se valoró el *Omron HeartGuide*. Este reloj es capaz de medir la presión sanguínea con una precisión clínica. Para ello incorpora un brazalete que se infla y ejerce presión sobre la muñeca. El procedimiento de medida es similar al de un tensiómetro tradicional y, por ello, los resultados tienen una precisión elevada. Los motivos por los que se decidió descartar este dispositivo fueron su elevado precio, la falta de disponibilidad inicial, la baja accesibilidad desde el punto de vista de la programación (la permite, pero no es *open*), la poca comodidad de uso y una monitorización continua compleja porque los procesos de medida requieren atención e intervención por parte del usuario.

3.2 Detección de la actividad física

Los sistemas de detección de la actividad física tuvieron un gran impulso con la aparición de los smartphones, ya que son dispositivos que las personas llevan pegados al cuerpo durante bastante tiempo. Además, cuentan con sensores como acelerómetros y giroscopios que permiten recoger datos para poder estimar la actividad física que se encuentra realizando el usuario.

Más recientemente, con el considerable aumento de ventas de los *wearables*, tanto pulseras como relojes inteligentes, parece razonable estudiar las diferentes formas de detectar la actividad que pueden ser factibles con estos dispositivos.

La Organización Mundial de la Salud [10] realiza una clasificación de la actividad física en función de METs (*Metabolic Equivalents*). Un MET es la relación entre la energía que consume una persona desarrollando un determinado trabajo y su metabolismo basal, es decir, la energía que consume en estado de reposo. La OMS distingue entre actividad física

moderada y actividad física intensa. La primera se corresponde con un MET entre tres y seis, es decir, el consumo de calorías aumenta entre tres y seis veces respecto a un estado de reposo. Durante este tipo de actividad, el ritmo cardiaco aumenta de forma *perceptible* [10]. Ejemplos de actividades que se encuentran dentro de esta categoría son: a) caminar a paso rápido, b) tareas domésticas o c) desplazamientos de cargas menores de 20 kilogramos. La actividad física intensa se corresponde con un MET superior a seis. En este caso, la respiración se vuelve más rápida y el ritmo cardiaco aumenta de forma *sustancial* [10]. Algunos ejercicios que se engloban dentro de este tipo de actividad son: a) *running*, b) deportes competitivos o c) desplazamientos de cargas mayores de 20 kilogramos.

Existen estudios sobre cómo utilizar la información que proporcionan los diferentes sensores para detectar la actividad. En [11] utilizan los datos que proporcionan acelerómetro, giroscopio y magnetómetro de un smartphone para detectar 5 tipos de actividad. Todos los sensores cuentan con los ejes X, Y, Z. Para reducir el sesgo y el ruido que pueda generar el sensor se aplica un pre-procesado de los datos, dividiendo los mismos en ventanas de 1,6 segundos con un 50% de solapamiento. Para la recolección de los datos utilizan siete voluntarios que realizaron las cinco actividades contempladas, al mismo tiempo que se guardaban los datos de los tres sensores mencionados anteriormente. Las actividades sobre las cuales se realiza el estudio son reposo, caminar, correr, subir escaleras y bajar escaleras. Una vez obtenidos los datos de todos los sensores, para cada medida del acelerómetro y giroscopio se calcula su vector magnitud para conseguir una independencia de la orientación y la posición en la que se encuentre el dispositivo. Para ello en [11] se aplica la ecuación (3.1).

$$A_3 = \sqrt{X^2 + Y^2 + Z^2} \tag{3.1}$$

Una vez realizados los cálculos se seleccionan los vectores magnitud del acelerómetro y del giroscopio y los valores de los tres ejes del magnetómetro, por lo que se cuenta con 5 señales (características, atributos o *features*) en el conjunto de datos. Todas las señales fueron extraídas utilizando ventanas con solapamiento para su posterior análisis estadístico.

Para cada ventana se calculan seis medidas estadísticas, para cada una de las 5 señales, que se utilizarán posteriormente para determinar el nivel de actividad. Estos estadísticos son:

- **Media**. Se calcula el valor medio de cada señal para cada ventana.
- **Mediana**. Se calcula la mediana de cada señal para cada ventana.
- **Desviación Estándar**. Desviación estándar de cada señal en cada ventana.
- **Asimetría** (*Skewness*). Determina el grado de asimetría que presenta la distribución. Se aplica para cada señal en cada una de las ventanas.
- **Curtosis**. Grado de concentración de los datos alrededor de la media. Se aplica para cada señal en cada una de las ventanas.
- **Rango Intercuartílico**. Diferencia entre el primer y el tercer cuartil de los valores de cada señal en cada ventana.

Los estadísticos obtenidos, junto con las etiquetas (clases o categorías), que identifican los tipos de actividad, conforman el conjunto de ejemplos. En [11] usan tres algoritmos de clasificación para poder extraer las reglas de asignación del nivel de actividad. Los algoritmos utilizados son SMO (*Sequential Minimal Optimization*), *Naive Bayes* y *J48* (implementación *open source* del reputado algoritmo C4.5 de R. Quinlan), siendo este último el que tuvo el mayor nivel de acierto.

[11] concluye que el enfoque seguido permite clasificar, con elevada precisión, los tipos de actividad que realizan las personas en el discurrir cotidiano de sus vidas. Además, el coste computacional del sistema propuesto (en la etapa de clasificación, no en la de aprendizaje que es *offline* y no tiene que realizarse en el llevable) es muy bajo, lo cual es un punto positivo añadido.

En [12] se utilizan los datos del acelerómetro y giroscopio para determinar 5 tipos de actividades diarias. Estas actividades son reposo, caminar, correr, subir escaleras y bajar

escaleras. Además, se indica que la frecuencia de muestreo adecuada de los sensores para la actividad física humana debe ser 8Hz. Al igual que en [11], se calcula el vector magnitud para cada medida de ambos sensores y posteriormente se calcula la media y la desviación estándar para determinar la actividad física. Además, se utiliza el valor mínimo del eje Y del acelerómetro. Para la recogida del conjunto de datos se utilizó un teléfono con Android que integra ambos sensores.

El nivel de acierto del sistema propuesto supera el 97%. Las actividades que presentan un nivel de acierto menor son el reposo y bajar escaleras. Debido a los buenos resultados, [12] plantea como trabajo futuro su extensión a un mayor número de actividades.

Por su parte [13] realiza un estudio sobre la precisión del reconocimiento de la actividad, utilizando el acelerómetro y giroscopio individualmente y de forma combinada. También estudia la posibilidad de utilizar el magnetómetro tanto individualmente como combinado con alguno de los otros dos sensores, para mejorar la precisión del sistema. Los tipos de actividades contempladas en el estudio son caminar, correr, subir escaleras, bajar escaleras, estar sentado y estar de pie. [13] incide en la importancia de detectar los sensores más discriminantes para clasificar las actividades, dado que el uso de muchos sensores hace que aumente significativamente el consumo de batería. El estudio se realiza utilizando los sensores de cuatro teléfonos Android ubicados en cuatro posiciones diferentes del cuerpo, entre ellas la muñeca. La frecuencia de muestreo elegida fue 50Hz, aunque se indica que una frecuencia menor también sería suficiente para reconocer la actividad física. Tras realizar la recolección de datos con cuatro voluntarios se realiza un pre-procesado similar al de [11]. En este caso se utiliza un tamaño de la ventana de 2 segundos y un factor de solapamiento del 50%. Para cada sensor se calcula el vector magnitud de los tres ejes utilizando la misma fórmula que en [11], pero en este caso no se descarta la información de cada uno de los tres ejes de cada sensor. Por lo tanto, para cada sensor existen cuatro dimensiones y para cada una de ellas se calcula la media y la desviación típica obteniendo un total de 24 características. [13] descarta, al igual que la

inmensa mayoría de los trabajos, analizar las características en dominio de la frecuencia debido al alto coste computacional de la transformada de Fourier. Para la experimentación [13] usa el entorno WEKA (*Waikato Environment for Knowledge Analysis*, software libre de Universidad de Waikato, Nueva Zelanda), *validación cruzada* (*cross-validation*) y distribución equiprobable de las categorías en los conjuntos para la evaluación de los resultados. Tras el análisis concluyen que la solución de mayor precisión es la que combina datos del acelerómetro y del giroscopio. En concreto, para la posición de la muñeca, utilizando únicamente el acelerómetro se obtienen unos resultados con un alto nivel de precisión, que mejora aún más al combinarlo con el giroscopio. Los resultados del magnetómetro no fueron satisfactorios, ya que los datos recogidos no tienen un patrón determinado y provoca un sobreajuste en los algoritmos de clasificación utilizados. Por tanto, [13] no recomienda su uso.

En [14] se utilizan dos acelerómetros para reconocer actividades estáticas, como estar sentado o de pie y actividades dinámicas, como caminar o correr. Para la detección de actividades estáticas usa un planteamiento similar al de [11], [12] y [13]: calcula el vector magnitud a partir de los tres ejes de cada sensor y diferentes medidas estadísticas a partir de los datos, como la media y la desviación típica. Las diferentes señales se dividen en ventanas pequeñas, donde para cada una de ellas se realizan los cálculos necesarios para clasificar el tipo de actividad. Es posible aumentar la resolución utilizando ventanas móviles superpuestas. Para las actividades dinámicas [14] recurre a un análisis en el dominio de la frecuencia, en lugar del análisis en el dominio del tiempo aplicado para las estáticas. El movimiento en este tipo de actividades, las dinámicas, tiene patrones que se repiten de forma periódica y por ello se puede obtener información *adicional* analizando el dominio de la frecuencia. En primer lugar, determina si la actividad es estática o dinámica utilizando la desviación estándar de los valores del vector magnitud. Una vez realizada esta clasificación, si la actividad es dinámica aplica la Transformada Rápida de Fourier (FFT) y calcula el *periodograma* utilizando los coeficientes de Fourier. Los máximos locales del periodograma se utilizarán posteriormente para clasificar las actividades. También calcula

la Densidad Espectral de Potencia (PSD), que describe como está distribuida la potencia de la señal, y la Entropía Espectral (SE), que permite diferenciar entre las partes deterministas de una señal y las partes aleatorias. Utilizando todas las características mencionadas realiza la clasificación de las actividades de igual forma que en las referencias anteriores: usando algoritmos de clasificación basados en árboles de decisión y en el teorema de Bayes. Pero, además, experimenta con redes neuronales intentando limitar el sobreajuste que se pueda producir en la fase de entrenamiento. El subconjunto de datos con el que se comprueba la efectividad de la red es disjunto con el usado durante el entrenamiento. Aunque los resultados obtenidos son ligeramente mejores, el análisis en el dominio de la frecuencia tiene un coste computacional muy superior al del dominio del tiempo, por lo que habría que valorar si el rendimiento ligeramente superior compensa el mayor desgaste de la batería y el mayor tiempo de espera.

A la vista de lo expuesto en la sección anterior se puede concluir que el procedimiento estándar para el reconocimiento de la actividad consta de los siguientes 4 pasos:

1. **Obtención**. Adquirir los datos en formato RAW (bruto). Los datos de los diferentes sensores se almacenan para ser utilizados posteriormente.
2. **Pre-procesado**. Los datos obtenidos en 1. se dividen en segmentos de la misma duración en segundos. Se aplican técnicas de “ventana deslizante” con un factor de solapamiento del 50% (el denominador común en la mayoría de estudios previos).
3. **Extracción de características**. Para cada ventana se extraen los diferentes estadísticos (p. Ej. media, desviación típica, grado de asimetría, etc.) que serán usados con posterioridad para clasificar las actividades.
4. **Aprendizaje**. Para realizar la clasificación es necesario un proceso previo de entrenamiento o aprendizaje que concluye en un conjunto de reglas (o funciones) de clasificación. En los trabajos consultados se observa que, en general, los algoritmos basados en árboles de decisión construyen conjuntos de reglas reducidos (árboles pequeños) y de elevada precisión.

Clasificación. No es, en sí, un paso del proceso. Es, simplemente, la aplicación en tiempo real de las reglas (funciones) sintetizadas en 4. A los datos que el llevable está registrando o adquiriendo.

3.3 Clasificadores utilizados en la detección de la actividad

En [11], [12] y [13] los clasificadores utilizados para determinar los diferentes tipos de actividades que se contemplan en dichos artículos son clasificadores basados en árboles de decisión y en el teorema de Bayes. También se utilizan algoritmos de Máquinas de Soporte Vectorial (*Support Vector Machines*, SVM) [15].

Como ya se ha comentado, el algoritmo J48 es la implementación de código abierto del C4.5 que utiliza WEKA. La salida del J48 es un árbol de decisión que se genera a partir del conjunto de datos cargado previamente. Las diferentes decisiones que contiene el árbol se van determinando desde el nodo raíz hasta las hojas etiquetadas con una clase o categoría.

El clasificador Naive Bayes, también implementado en WEKA, basa su funcionamiento en el teorema de Bayes, que asume la independencia de los atributos que forman parte del conjunto de datos.

El clasificador BinarySMO resuelve un problema binario de clasificación. Las clases quedan caracterizadas mediante rectas que dividen los puntos que representan las diferentes instancias de las dos clases. Esta recta separa las clases con la mayor distancia posible.

En *Python* existe la biblioteca *scikit-learn*, que es una de las bibliotecas más utilizadas para aprendizaje automático. Es de código abierto y contiene un conjunto de algoritmos, tanto de aprendizaje supervisado como de aprendizaje no supervisado. Para realizar la clasificación contiene la clase *DecisionTreeClassifier* que, a partir de un conjunto de datos, genera un árbol que clasifica las diferentes clases presentes en el conjunto de datos. Toma

como entrada dos vectores, uno con los datos de entrenamiento y otro con las etiquetas de las clases del conjunto. Una vez inicializado, hay que realizar el entrenamiento con el conjunto de datos de entrenamiento, que se obtiene realizando una partición del conjunto de datos original. Tras esta partición, se obtiene también el conjunto de datos de test. Esta operación se realiza con la función *train_test_split()* incluida también en scikit-learn. Una vez entrenado es posible dibujar el árbol para poder visualizar las diferentes decisiones que este contiene, como producto del entrenamiento. El algoritmo que utiliza es una versión optimizada de algoritmo CART (*Classification And Regression Trees*) [16], muy similar al C4.5. Scikit-learn también incorpora algoritmos de máquinas de soporte vectorial como SVC (*Support Vector Classification*, implementado usando la biblioteca *libsvm*), NuSVC (similar a SVC con un parámetro para controlar el número de vectores de soporte) y LinearSVC (como SVC, pero usando un *kernel* lineal; implementado usando *liblinear* en vez de libsvm), que permiten realizar clasificaciones binarias y multi-categoría. En [14] utilizan redes neuronales para clasificar las actividades físicas. Scikit-learn incorpora la clase *MLPClassifier* que implementa un *Perceptrón Multicapa* (*MultiLayer Perceptron*, MLP). MLP aprende una función no lineal que utilizará para realizar la clasificación.

Para evaluar los resultados de los clasificadores es habitual recurrir a estimadores insesgados o, al menos, lo menos sesgados posible. Uno de ellos, el más usado, es la validación cruzada (CV) anteriormente comentada. La CV consiste en dividir el conjunto de datos en k subconjuntos de, aproximadamente, igual tamaño (*k-folds*). Posteriormente, se utilizan $k-1$ subconjuntos para realizar el entrenamiento del modelo. El subconjunto restante se utiliza como test para validar el modelo. Este procedimiento se realiza de forma iterativa k veces, donde cada subconjunto se utiliza exactamente una vez como test.

Cuando el número de subconjuntos generados coincide con el tamaño del conjunto de datos (k igual al cardinal del conjunto) se está ante el estimador menos sesgado, conocido como *Leave-one-out* (uno fuera) o, también, como *Leave-One-Out Cross-Validation* (LOOCV). En este caso, en cada iteración se realiza el entrenamiento con todo el conjunto

de datos, dejando fuera solamente un ejemplo, que se utiliza para testar lo aprendido. Aunque LOOCV es un estimador excelente es, computacionalmente, muy costoso por lo que, para conjuntos grandes, o sistemas lentos, no se suele usar.

En cualquiera de las variantes de validación cruzada comentadas, el error se calcula como la media aritmética de los errores de todas las iteraciones que se han realizado.

3.4 Recuperación de frecuencia cardiaca tras actividad física

La recuperación de la frecuencia cardiaca (*Heart Rate Recovery,* HRR) es el descenso de la frecuencia cardiaca, que se produce en los instantes inmediatamente posteriores a la realización de una actividad física. Este parámetro constituye una medición establecida de la funcionalidad del sistema nervioso autónomo (SNA). En el año 1994, Imai publicó un estudio que mostraba cómo la rápida deceleración inicial de la frecuencia cardiaca (FC) tras el ejercicio podía atenuarse con *Atropina* [17]. Además, observó que los pacientes con insuficiencia cardiaca, que padecen un marcado desequilibrio del sistema nervioso autónomo, no presentaron la rápida caída inicial de la FC observada en personas sanas y atletas.

Posteriormente, el grupo de la Clínica Cleveland Cole y algunos colaboradores publicaron un artículo, que pretendía examinar la utilidad a la hora de realizar un pronóstico, de la HRR [18]. Para ello, realizaron un estudio durante más de 6 años con 2.428 adultos, que fueron sometidos a pruebas de esfuerzo limitada en cintas de correr. Tras finalizar la prueba de esfuerzo, se inició un periodo de recuperación de 2 minutos a un ritmo de 2,4 km/h con una pendiente del 2,5%. Durante ese periodo, se midió el valor de recuperación de la frecuencia cardiaca definido como “la reducción en la frecuencia cardiaca entre la frecuencia en el máximo esfuerzo y la frecuencia 1 minuto tras el cese del ejercicio”. El valor medio de recuperación, para los individuos que participaron en el estudio, fue de 17 pulsaciones por minuto durante el primer minuto de reposo tras realizar ejercicio. Se demostró como un valor bajo está muy relacionado con una posible

enfermedad cardiaca. [18] concluye que una recuperación de menos de 12 pulsaciones por minuto aumenta las probabilidades de fallecimiento por cualquier causa. Este valor es independiente de la intensidad de la actividad física realizada.

En [19] se llevó a cabo un estudio con 12 voluntarios que realizaron un calentamiento previo. A continuación, realizaron 3 minutos de ciclismo llegando a un 60% de la frecuencia cardiaca máxima de cada voluntario. Para finalizar se evaluaron tres modos diferentes de recuperación con una duración de 5 minutos. El primero de ellos consistió en un reposo pasivo, donde el sujeto se encontraba inactivo sentado en una silla. El segundo fue un reposo activo, donde se redujo la carga física del ejercicio, mientras que el tercero fue un reposo pasivo, donde el sujeto continuaba realizando pedaladas, pero en una bicicleta de tándem donde otro compañero realizaba el esfuerzo. En la Figura 3.1 se puede observar el valor de la recuperación de la frecuencia cardiaca en los 5 minutos posteriores a la realización de las actividad física, en cada uno de los tres modos de recuperación estudiados. Los valores que se muestran en el gráfico corresponden con el valor medio de los 12 sujetos que participaron en el estudio, y la línea de puntos es la frecuencia cardiaca previa a la realización del ejercicio. Se observa de forma clara como la frecuencia cardiaca disminuye más lentamente si el periodo de recuperación es activo. También se puede comprobar como la recuperación de la frecuencia cardiaca se produce fundamentalmente en los dos minutos posteriores a la finalización del ejercicio, y especialmente durante el minuto inmediatamente posterior.

En [20] se realizó un estudio con 50 pacientes que habían sufrido un infarto de miocardio con una edad media de 51 años. Tras realizar un proceso de rehabilitación de los pacientes durante 12 semanas, la recuperación de la frecuencia cardiaca en el primer minuto después del ejercicio físico fue, en media, de 24,7 pulsaciones por minuto. Durante los dos minutos posteriores a la finalización del ejercicio la recuperación fue de 38,9 pulsaciones por minuto. En este estudio se indica que una recuperación inferior a 12 y 22 pulsaciones por

minuto, en el minuto posterior y en los dos minutos posteriores respectivamente a la realización de la actividad física, se considera como anormal.

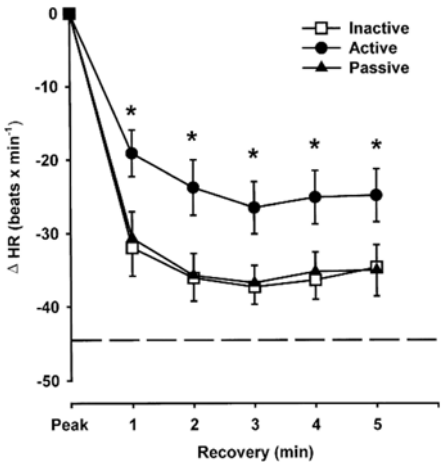

Figura 3.1. Recuperación de la frecuencia cardiaca. Imagen procedente de [19].

En [21] se realizó un estudio con 40727 participantes sin enfermedades cardiacas con una edad media de 56 años. El estudio se centra en la recuperación de la frecuencia cardiaca en los 10 segundos posteriores al cese de la actividad física. También se estudian los 20, 30, 40 y 50 segundos posteriores a dicho cese. Durante los 10 segundos posteriores de media se recuperan 18,4 pulsaciones por minuto. Se llega a la conclusión de que el mejor predictor de la mortalidad se basa en los resultados obtenidos en los 10 segundos posteriores al cese de la actividad física. Cuanto más aumenta el periodo utilizado para medir la recuperación, la predicción de la mortalidad es más inexacta.

3.5 Sensores y frecuencia cardiaca en wearables

Los sensores que se encargan de medir el movimiento en los llevables son el acelerómetro y el giroscopio. El acelerómetro obtiene las aceleraciones que sufre el dispositivo realizando mediciones en tres ejes ortogonales entre sí mismos (X, Y, Z). Para cada eje se puede obtener la fuerza de aceleración que está sufriendo en un instante en m/s^2. Hay que tener en cuenta que la fuerza de la gravedad siempre forma parte de los resultados que se extraen del sensor. En la Figura 3.2 se puede observar cómo se sitúan los tres ejes en el dispositivo.

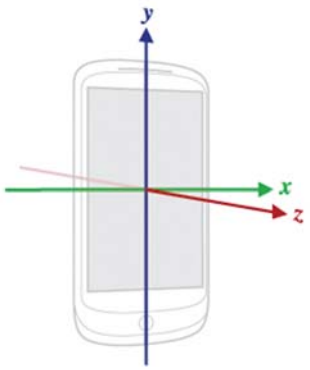

Figura 3.2. Disposición de los ejes del acelerómentro respecto al dispositivo.

El giroscopio mide el movimiento de rotación del dispositivo alrededor de cada uno de los ejes. Para cada eje se pude obtener la velocidad angular en radianes por segundo (rad/s). Al igual que en el acelerómetro, los tres ejes utilizados son los de la Figura 3.2.

Como sensor de posición se suele utilizar un magnetómetro. Este sensor se encarga de medir el campo magnético terrestre en cada uno de los tres ejes de la Figura 3.2. Con la información que proporciona sería posible crear una brújula. La unidad de medida habitual es el μT, que es una unidad de inducción magnética.

Estos 3 sensores constituyen un conjunto conocido como *Sensores Hardware* (SH), que se caracterizan por un elevado nivel de integración en el hardware del dispositivo y su bajo consumo energético. Al resto de sensores se les conoce como *Sensores Software* (SS). Muchos de los SS no son dispositivos físicos, sino que los datos que proporcionan los obtienen de uno o varios SH. Por ello, los SS tienen un mayor consumo de batería. Evidentemente no son conjuntos cerrados, la familia de los SH va creciendo a medida que las necesidades de nuevas funcionalidades aumentan y los avances en la tecnología lo permiten.

Así, la mayoría de llevables con capacidad para medir la frecuencia cardiaca usan un sensor óptico, junto con una fuente de iluminación, típicamente un LED de color verde. Este sensor basa su funcionamiento en la *Fotopletismografía* (*PhotoPlethysmoGraphy*,

PPG) [22]. Esta tecnología no invasiva parte de la hipótesis de que la sangre es de color rojo porque refleja la luz roja y absorbe luz verde. La sangre de las arterias tiene un nivel de absorción de la luz diferente al del resto de tejidos que las rodean. Además, esta absorción no es constante, cuando el corazón bombea sangre aumenta el flujo sanguíneo y, por lo tanto, aumenta la absorción de luz. Hasta que se produce el siguiente bombeo el flujo sanguíneo va disminuyendo y el nivel de absorción de luz se reduce. De esta forma, conociendo cada cuanto tiempo aumenta el flujo sanguíneo es posible determinar la frecuencia cardiaca. Además de la frecuencia cardiaca es posible obtener otros parámetros del sistema cardiovascular a partir de la señal generada por el sensor óptico. Por ejemplo, el nivel periférico de saturación de oxígeno en sangre (*Peripheral Oxygen Saturation*, SpO2) [8] se obtiene utilizando el espectro correspondiente al rojo dado que es el más sensible al cambio del nivel de oxígeno en sangre. También es posible realizar una aproximación a la estimación de la presión sanguínea.

En [23] se realiza un estudio sobre la exactitud de este tipo de sensores para medir la frecuencia cardiaca. Para ello se utilizan los sensores ópticos incorporados en algunas de las pulseras más vendidas, entre ellas la *Xiaomi Smart Band 3*. Se estudiaron diferentes situaciones que pueden influir en la exactitud de los resultados como el tono de la piel, si el usuario se encuentra realizando algún tipo de actividad física o se encuentra en estado de reposo. La utilización de este tipo de sensor sobre una piel oscura reduce la exactitud en, aproximadamente, un 15% ya que este tipo de piel absorbe más luz verde que la piel más clara. Algunos fabricantes incluso recomiendan utilizar sus dispositivos únicamente sobre pieles claras. Otro factor que reduce la exactitud en la medición es el movimiento de la pulsera sobre la muñeca, que puede provocar deformaciones en la piel que hacen que se obtengan falsos latidos o se pierdan algunos. La actividad física también influye en la precisión de la medición obtenida. En [23]se demuestra como el error aumenta cuando el usuario se encuentra realizando algún tipo de actividad física y disminuye cuando se encuentra en reposo. Por último, [23] destaca que en ninguno de los dispositivos probados es posible obtener datos brutos (RAW) directamente de los sensores, para realizar un

procesamiento distinto de la señal y, así, poder estimar otros parámetros como la variabilidad de la frecuencia cardiaca.

Algunos dispositivos de alta gama como el Apple Watch Series 5 incorporan un conjunto de electrodos para medir las señales eléctricas del corazón. De esta forma puede hacer electrocardiogramas (ECGs) que permiten detectar posibles problemas en el corazón. Es importante destacar que un electrocardiograma tradicional realizado en un hospital se obtiene utilizando doce electrodos mientras que el Apple Watch solo cuenta con un electrodo. En cualquier caso, este dispositivo cuenta también con un sensor óptico para monitorizar la frecuencia cardiaca, por lo que es una opción muy completa para monitorizar el corazón.

En cualquier caso, estos sensores ópticos permiten monitorizar a un paciente sin la necesidad de portar equipos médicos que pueden resultar complejos o molestos. Además, la monitorización puede ser continua y en tiempo real durante el desempeño de una actividad (p. Ej. realizar deporte) obteniendo, de esta forma, información que puede ser valiosa de cara a un diagnóstico médico.

Android agrupa los sensores en tres categorías:

- **Movimiento**. Se encargan de medir fuerzas de aceleración y de rotación. En este grupo se encuentran los acelerómetros y giroscopios.
- **Ambientales**. Miden propiedades del entorno como la temperatura, la humedad o la iluminación. Dentro de este grupo se encuentran termómetros, barómetros o fotómetros.
- **Posición**. Se encargan de medir la posición en la que se encuentra el dispositivo. En este grupo se encuentra el magnetómetro.

La forma de obtener los datos de los diferentes sensores es mediante un evento *SensorEvent*, que debe ser escuchado por un *SensorEventListener* para recoger los datos.

El SensorEvent incluye en sus campos un *array* con los valores resultantes de la medición del sensor. La longitud del *array* depende del tipo de sensor. Por ejemplo, para un acelerómetro es posible obtener el valor de cada eje de la siguiente forma: "*SensorEvent.values[0], SensorEvent.values[1], SensorEvent.values[2]*".

Para cada sensor se pueden especificar cuatro frecuencias de muestreo diferentes, aunque el sistema Android lo toma como una recomendación. Por tanto, es posible que la frecuencia real no corresponda con la frecuencia que se le ha indicado. La frecuencia *normal* es, aproximadamente, 5Hz. Las otras frecuencias son: 1) "UI" a 16,6Hz, 2) "Fastest" se corresponde con la máxima permitida por el dispositivo y 3) "Game" a 50Hz.

3.6 Bluetooth de Baja Energía

Bluetooth es una especificación industrial para *Redes Inalámbricas de Área Personal* (*Wireless Personal Area Network*, WPAN) creada por *Bluetooth Special Interest Group, Inc*. Bluetooth permite el intercambio de información entre dos dispositivos mediante un enlace por radiofrecuencia en la banda de 2.4GHz. En función de la potencia de transmisión existen cuatro clases de dispositivos Bluetooth:

- **Clase 1**. Los más potentes con un alcance que puede llegar hasta los 100 metros.
- **Clase 2**. Tiene una potencia de transmisión cinco veces inferior a los de la Clase 1 y su alcance se reduce, como máximo, a unos diez metros.
- **Clases 3 y 4**. Cuentan con una menor potencia de transmisión y reducen su alcance hasta un metro y medio metro, respectivamente.

A mayor potencia de transmisión mayor consumo de batería. Por lo tanto, es importante seleccionar adecuadamente la clase del dispositivo dependiendo de la función que vaya a desempeñar la aplicación.

Desde la primera especificación de Bluetooth el ancho de banda que permite cada especificación ha aumentado de forma considerable. En sus primeras versiones apenas

llegaba a 1 Mbit/s. Desde entonces se han ido desarrollando nuevas versiones de la especificación con numerosas mejoras. Gracias a ello, el ancho de banda en la última versión alcanza los 50 Mbit/s.

Con la llegada de la versión v4.0 de Bluetooth en 2010, se completó su especificación incluyendo el Bluetooth clásico, el Bluetooth de alta velocidad y el Bluetooth de bajo consumo (*BluetoothLE*). El Bluetooth de bajo consumo es, por tanto, un subconjunto del estándar Bluetooth v4.0, aunque no es compatible con versiones anteriores de Bluetooth ni con versiones que no son de baja energía. BluetoothLE se desarrolló principalmente para dar soporte a pequeños dispositivos alimentados con una pila de botón. La gestión de la batería es un factor crítico y BluetoothLE ayuda de forma notable en esta tarea.

BluetoothLE incorpora un protocolo de *Perfil de Acceso Genérico* (*Generic Access Profile*, GAP) para la detección de dispositivos. De esta forma se proporciona un entorno de trabajo (*framework*) para permitir que los diferentes dispositivos se descubran entre sí, difundan datos y establezcan conexiones seguras. GAP define dos tipos de roles para los dispositivos. Por un lado, se encuentran los periféricos o dispositivos de poca potencia, con recursos limitados y un consumo bajo de energía. Este rol es utilizado por relojes y pulseras inteligentes, monitores de ritmo cardiaco y cualquier otro dispositivo pequeño que cumpla con los requisitos anteriormente citados. Por otro lado, los dispositivos centrales, que cuentan con una mayor capacidad de memoria y procesamiento. Normalmente son teléfonos móviles.

Los periféricos anuncian su presencia para que los dispositivos centrales puedan establecer una conexión. Los dispositivos centrales son los que se encargan de iniciar la conexión una vez han escuchado los diferentes paquetes de anuncio de dispositivos periféricos. Para ello envía un paquete solicitando la conexión al dispositivo periférico, que debe ser aceptado. Una vez establecida la conexión, los dispositivos periféricos dejan de anunciar su presencia. De esta forma solamente se pueden conectar a un dispositivo

central mientras que estos últimos pueden estar conectados a varios periféricos. Cuando un periférico y un dispositivo central se encuentran conectados es posible modificar determinados parámetros de la conexión. Solamente los centrales pueden modificar estos parámetros, aunque los periféricos pueden solicitar un cambio a los dispositivos centrales.

BluetoothLE utiliza un *Perfil de Atributos Genéricos* (*Generic Attribute*, GATT) que define la forma en que dos dispositivos transmiten sus datos. Se establecen las estructuras de datos a utilizar y los modos de funcionamiento. Los tipos de atributos que se contemplan son servicios, características y descriptores.

Los servicios se utilizan para dividir los datos en entidades lógicas y contienen diferentes datos que son las características. Cada servicio puede tener una o varias características. Los servicios se identifican por medio de un ID único llamado UUID que puede ser de 16 o 128 bits. Uno de los servicios contemplados en la especificación es el Servicio de Ritmo Cardiaco. Se identifica con el UUID de 16 bit 0x180D y puede contener hasta tres características dependiendo de la implementación.

Las características se encuentran dentro de un servicio GATT y representan el nivel más bajo en las transacciones entre dos dispositivos conteniendo un único tipo de dato. Como ocurre con los servicios, las características también se identifican con un UUID de 16 o 128 bits. Siguiendo con el ejemplo del Servicio de Ritmo Cardiaco, dentro de este se pueden encontrar hasta tres características. Una de ellas es la medición del ritmo cardiaco que se corresponde con el UUID 0x2A37. Realizando una operación de lectura de esta característica es posible acceder a la información contenida en dicha característica. En concreto, los primeros 8 bits indican el formato en el que se encuentran los datos de la característica. En los siguientes bits se encuentra el valor del ritmo cardiaco en formato uint8 o uint16.

Por último, los descriptores son atributos que proporcionan información adicional sobre una determinada característica. En la Figura 3.3 se observa la jerarquía entre servicios, características y descriptores.

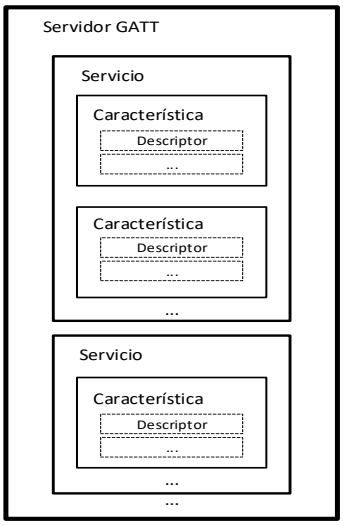

Figura 3.3. Componentes y jJerarquía de un GATT usado en BluetoothLE.

El dispositivo que contiene los datos, normalmente un sensor, una pulsera, reloj inteligente, etc., es el servidor GATT. El dispositivo que utiliza esos datos para realizar un determinado procesamiento es el cliente GATT.

Bluetooth, y más concretamente BluetoothLE, está presente en la práctica totalidad de smartphones y wearables del mercado. Los llevables utilizan esta especificación para conectarse a los smartphones y, de esta forma, recibir notificaciones o transmitir información. Para la comunicación entre smartphones y wearables el BluetoothLE es recomendable porque no se precisa de un elevado alcance, suelen estar muy cerca uno de los otros, no se requiere un ancho de banda elevado y el número de datos que se transmiten es reducido. La principal ventaja del BluetoothLE es ser una tecnología barata que habilita conexiones sin cables entre dos dispositivos. La desventaja es que el número máximo de dispositivos conectados simultáneamente es limitado. Nótese que, como ya se ha comentado, el alcance no es una variable relevante para el trabajo que se desarrolla en este Trabajo Final de Grado.

Android tiene soporte tanto para la versión clásica de Bluetooth como para BluetoothLE. Por razones de consumo energético en este trabajo se ha optado por utilizar BluetoothLE, explicándose a continuación cómo se utiliza esta especificación a través de la *Interfaz de Programación de Aplicaciones* (*Application Programming Interface*, API) de Android. Desde la versión 4.3 de Android (conocida como *JellyBean*) se incluye soporte para BluetoothLE. El principal caso de uso que se contempla es la transmisión de pequeños volúmenes de datos entre dos dispositivos cercanos.

En primer lugar, para que una aplicación Android tenga acceso a las características Bluetooth se debe declarar el permiso BLUETOOTH en el fichero *AndroidManifest.xml*. También es recomendable declarar el permiso ACCESS_FINE_LOCATION. Durante el inicio de la aplicación es recomendable que se compruebe si el dispositivo es compatible con BluetoothLE. A continuación, se debe comprobar si el BluetoothLE está activado y, en caso de que no lo esté, permitir al usuario que lo active. Esta acción se puede realizar sin salir de la aplicación a través de los objetos *BluetoothAdapter* y *BluetoothManager*.

El siguiente paso consiste en realizar un escaneo de los dispositivos BluetoothLE disponibles. Para ello se utiliza el método *BluetoothAdapter.startLeScan*() que recibe como parámetro un *BluetoothAdapter.LeScanCallback* encargado de recibir la información de los dispositivos disponibles. Se recomienda fijar un tiempo máximo para la duración del escaneo con el objetivo de limitar la descarga de la batería.

Una vez obtenidos todos los dispositivos BluetoothLE disponibles, es posible conectarse con uno de ellos mediante el método *BluetoothDevice.connectGatt*() para realizar la conexión con un servidor GATT. Uno de los parámetros que recibe este método es un *BluetoothGattCallback,* que se encarga de recibir eventos relativos al estado de la conexión y a diferentes operaciones realizadas con el servidor GATT, como el descubrimiento de servicios o características y la lectura de una determinada característica.

Por último, para leer una determinada característica de un servidor GATT desde la aplicación Android, primero, se debe obtener la característica a partir de la lista de servicios soportados. A partir de un servicio, se puede obtener la lista de características utilizando el método *BluetoothGattServices.getCharacteristics*(). De ese listado se extrae la característica que se quiere leer, para, con el método *BluetoothGatt.readCharacteristic*() obtener el valor deseado. Dicho valor se retorna a través del BluetoothGattCallback especificado, al realizar la conexión con el servidor GATT.

Además de leer directamente una determinada característica de un servidor GATT, es posible que la aplicación solicite recibir una notificación cuando se produce un cambio en la característica. Para ello es necesario realizar dos pasos. El primero de ellos consiste en activar las notificaciones de la característica mediante el método *BluetoothGatt.setCharacteristicNotification*(). Posteriormente hay que obtener el descriptor asociado a la característica con el método *getDescriptor*() y escribir en dicho descriptor el valor necesario para activar las notificaciones. Esto se realiza con los métodos *BluetoothGattDescriptor.setValue*() y *BluetoothGatt.writeDescriptor*(). Los cambios en la característica se obtienen a través del BluetoothGattCallback en el método *onCharacteristicChanged*().
Para terminar la conexión desde la aplicación Android se llama al método *BluetoothGatt.close*().

3.7 Separación de sonidos de corazón y pulmón

Como se ha comentado, la auscultación es uno de los métodos más extendidos en entornos médicos a la hora de realizar un primer diagnóstico. El facultativo, utilizando un estetoscopio, *interpreta* los sonidos que oye al objeto de detectar la existencia, o no, de anomalías. Lo que el médico escucha es una mezcla de sonidos que, en el mejor de los escenarios, está compuesta únicamente por sonidos que provienen del corazón y de los pulmones. La auscultación es mínimamente invasiva, muy rápida, barata y, además, ayuda

a detectar dolencias y enfermedades de dos sistemas sumamente importantes como lo son el Circulatorio/Cardiovascular y el Respiratorio. Como punto negativo, el diagnóstico depende de las condiciones ambientales (ruido), de la pericia o percepción del facultativo, etc.

También se ha comentado que el sonido es una fuente de información muy rica y llena de matices. Bien sea porque los sensores disponibles no se encuentran tan avanzados en términos de integración y aceptación, bien porque su procesamiento es más complejo y precisa de mayor cantidad de recursos computacionales, etc., el caso es que su tratamiento en los llevables no es habitual. No obstante, la inclusión del audio en los wearables, con fines de salud, está cercana.

Los sonidos procedentes del corazón y del pulmón se solapan tanto en el dominio del tiempo como en el dominio de la frecuencia. El solapamiento temporal se debe a que ambos órganos deben *funcionar* de forma continua y no es posible cesar la actividad de uno de ellos sin consecuencias muy serias. El solapamiento en frecuencias tiene que ver con que ambos emiten sonidos en una banda de frecuencia similar. En concreto, como se indica en [24], el problema del solapamiento se circunscribe a la banda de frecuencias 60Hz-320Hz. El corazón emite sonidos en la banda 10Hz-320Hz. Más concretamente, el primer ruido del corazón (R1), que ocurre durante la Fase 1 del ciclo cardiaco, se sitúa en la banda 10Hz-300Hz, mientras que el segundo ruido del corazón (R2), que ocurre durante la Fase 4, se sitúa en la banda 50Hz-320Hz. El pulmón durante el proceso de inspiración y expiración concentra su energía en la banda de frecuencia 60Hz-1000Hz.

Hay supuestos donde no es necesario realizar una separación (p. ej. una *estimación aceptable* de la frecuencia cardiaca) y otros donde con una separación *razonable* sería suficiente (p. ej. la detección de sibilancias). Sin embargo, una separación de calidad (fina y precisa) es válida para los casos anteriores y para todo un espectro de usos y dolencias más exigentes.

Los primeros métodos usados para realizar la separación se basaban en Filtros Paso Alto (*High Pass Filter*, HPF) y Paso Bajo (*Low Pass Filter*, LHF). Los HPF, como su nombre indica, eliminan las frecuencias que no superan un umbral establecido, mientras que los LSF filtran las frecuencias que lo superan. Estos métodos eliminaban información importante de la señal que proviene del corazón y, por ello, no se utiliza en la actualidad.

Hay más propuestas y trabajos de la comunidad científica para realizar la separación. En este trabajo se ha seleccionado la presentada en [24] por motivos tales como: 1) que es un trabajo reciente, 2) que la calidad de su separación es muy elevada y 3) porque permite una aproximación, desde la perspectiva de la computación paralela y de altas prestaciones, interesante y útil.

[24] propone el uso de la Factorización No-Negativa de Matrices (*Non-Negative Matrix Factorization*, NMF) para separar las señales emitidas por el pulmón y el corazón. La NMF es en una herramienta muy útil en problemas como el agrupamiento de documentos, la extracción de datos, el análisis de datos e imágenes, la separación de fuentes o la bioinformática (ver, por ejemplo, [25], [26], [27] y [28]). Cuando los valores son no negativos, como es este caso, una forma de calcular la factorización es minimizar el error de reconstrucción entre el espectrograma observado y el modelado.

La NMF aproxima una matriz $X \in \mathbb{R}^{m \times n}$, con todos sus elementos no negativos, por el producto de dos matrices $W \in \mathbb{R}^{m \times k}$ y $H \in \mathbb{R}^{k \times n}$ de menor rango ($k \leq \min(m, n)$), también con todos sus elementos no negativos, de forma que $X \approx WH$. La factorización se aborda como el problema de calcular dos matrices W_0 y H_0 tal que

$$\|W_0 H_0 - X\|_F = \min_{W,H \geq 0} \|WH - X\|_F, \tag{3.2}$$

donde $\| \quad \|_F$ denota la norma de *Frobenius*. Otro tipo de normas pueden ser usadas, como la divergencia de *Kullback-Leibler* [29] o la divergencia de *Itakura-Saito*.

La factorización se resuelve, generalmente, a través de un problema de minimización $\min_{W,H\geq 0} Dist(X|WX)$ donde $Dist(X|WX)$ viene definida por la ecuación (3.3), donde $d(x|y)$ representa una función de coste escalar.

$$Dist(X|WX) = \sum_{i=1}^{m}\sum_{j=1}^{n} d([X]_{i,j}|[WH]_{i,j}) \tag{3.3}$$

En [29] Lee propone usar, como funciones de coste escalar, la distancia euclidiana ($d(x|y) = 1/2\,(x-y)^2$) y la divergencia de Kullback-Leibler ($d(x|y) = x\log(x/y) - x + y$). Así, por ejemplo, considerando la distancia euclidiana y el algoritmo del gradiente descendente para resolver el problema de minimización se obtienen las reglas de actualización mostradas en la ecuación (3.4), donde el símbolo $\odot$ denota productos *Hadamard* (productos a nivel de elemento, *element-wise multiplication*), también los cocientes son a nivel de elemento.

$$\begin{aligned} H &\leftarrow H \odot \frac{W^T X}{(W^T W)H} \\ W &\leftarrow W \odot \frac{XH^T}{W(HH^T)} \end{aligned} \tag{3.4}$$

En muchas aplicaciones de la NMF es más útil usar otras métricas para medir la cercanía entre dichas matrices. En [30] se comprueba empíricamente que las métricas βeta-*divergencia* proporcionan resultados más precisos para ciertos problemas, manteniendo bajo ciertas condiciones la convergencia. La βeta-*divergencia* puede ser definida como se muestra en la ecuación (3.5) [31]. Nótese que la βeta-*divergencia* incluye en su definición otras funciones de coste como la euclidiana ($\beta = 2$), la Kullback-Leibler ($\beta = 1$) o la Itakura-Saito ($\beta = 0$).

Partiendo de [29] y del criterio del gradiente se obtienen las reglas de actualización mostradas en la ecuación (3.6), donde $X^{\cdot n}$ denota $([X]_{i,j})^n$.

$$_{\beta}(x|\hat{x}) = \begin{cases} \frac{1}{\beta(\beta-1)}(x^{\beta} + (\beta-1)\hat{x}^{\beta} - \beta x\hat{x}^{\beta-1}) & \beta \in \mathbb{R}\backslash\{0,1\} \\ x\,log\left(\frac{x}{\hat{x}}\right) + \hat{x} - x & \beta = 1 \\ \frac{x}{\hat{x}} - log\frac{x}{\hat{x}} - 1 & \beta = 0 \end{cases} \quad (3.5)$$

$$H \leftarrow H \odot \frac{W^T((WH)^{\cdot\beta-2} \odot A)}{W^T(WH)^{\cdot\beta-1}}$$
$$W \leftarrow W \odot \frac{((WH)^{\cdot\beta-2} \odot A)H^T}{(WH)^{\cdot\beta-1}H^T} \quad (3.6)$$

Volviendo al problema de este TFG, W es la matriz cuyas columnas son los patrones espectrales de cada base, H la matriz cuyas filas denotan las activaciones temporales de cada una de las bases utilizadas y k el número de bases. [24] propone la función de Kullback-Leibler porque su valor es 0 cuando $a = b$, es una función convexa en $(0, +\infty)$.

Debido a la dispersión de las fuentes a separar, [24] añade a la función de coste un término de penalización para la matriz H y, así, conseguir una factorización donde el valor de la dimensión k sea lo menor posible. Dicha penalización consiste en calcular la $\| \quad \|_1$ de la matriz H (ver ecuación (3.7)).

$$\|H\|_1 = \max_{1\leq j\leq t} \sum_{i=1}^{k} |h_{i,j}| \quad (3.7)$$

Lo que se persigue es que la función de coste aumente al aumentar el número de elementos de H distintos de 0 (penalización de matrices densas). El peso de la penalización se controla con un parámetro (λ). En consecuencia, la función de coste queda como se muestra en la ecuación (3.8), donde $d(X|WH)$ es una de las funciones de coste mencionadas anteriormente y λ es el parámetro que mide la dispersión en el rango $[0, 1]$. [24] demuestra que la inclusión del factor λ mejora la velocidad de convergencia del método.

$$D(X|WH) = \|d(X|WH)\|_f + \lambda\|H\|_1 \tag{3.8}$$

Teniendo en consideración las ecuaciones (3.7) y (3.8), las reglas de actualización para las matrices W y H en [24] se muestran en la ecuación (3.9). Como criterio de parada [24] utiliza la ecuación (3.10), donde $D^i(X|\hat{X})$ y $D^{i-1}(X|\hat{X})$ representan la función de coste en la iteración $i-ésima$ y en la iteración $i-1$, respectivamente, y ξ un umbral previamente establecido. También es habitual ver propuestas donde el criterio de parada se basa única y exclusivamente en un número de iteraciones previamente establecido [32], o una mezcla de ambos.

$$\begin{aligned} H &\leftarrow H \odot \frac{W^T((WH)^{\odot -1} \odot X)}{W^T + \lambda} \\ W &\leftarrow W \odot \frac{((WH) \odot^{-1} X)H^T}{H^T} \end{aligned} \tag{3.9}$$

$$\frac{D^i(X|\hat{X}) - D^{i-1}(X|\hat{X})}{D^i(X|\hat{X})} < \xi \tag{3.10}$$

Dependiendo de qué matrices se actualizan durante la factorización, el sistema puede ser *Supervisado* (sNMF), *Semi-Supervisado* (ssNMF) o *No-Supervisado* (nsNMF). sNMF comienza con una fase de entrenamiento para obtener los patrones espectrales de cada fuente sonora, de forma que la fase de separación solo calcula las matrices de activaciones temporales (H) de cada señal a separar. ssNMF únicamente entrena los patrones espectrales de una de las fuentes sonoras y, en la fase de separación, calcula el resto de matrices de patrones y todas las matrices de activación temporales. nsNMF no realiza un entrenamiento previo, la fase de separación calcula todas las matrices para todas las fuentes sonoras.

[24] recurre a un sistema ssNMF, es decir, no realiza el entrenamiento de los patrones espectrales de todas las señales. Supone, para la separación pulmón/corazón, que las dos señales están mezcladas, aproximadamente, de forma lineal, es decir, que $X(i) = Xc(i) +$

$Xp(i)$. A continuación se detalla el algoritmo propuesto en [24] para realizar la separación, que se ha dividido en 5 etapas.

Primera etapa: Obtener el espectrograma

Esta primera etapa consiste en obtener una representación de la señal de entrada en dominio de la frecuencia a partir de su representación de dominio del tiempo. Para ello se utiliza la Transformada de Fourier de Tiempo Reducido (STFT) con ventanas de *Hanning*. Estas ventanas se utilizan para minimizar los problemas derivados de trabajar con una señal que contiene un número no entero de ciclos. Cada elemento de la ventana se calcula aplicando la ecuación (3.11).

$$v(n) = a_0 - a_1 \times \cos\left(\frac{2\pi n}{N-1}\right) \qquad a_0 = 0{,}53836; \quad a_1 = 0{,}46164 \tag{3.11}$$

Como resultado de la STFT se obtiene el espectrograma X de la mezcla corazón/pulmón, que es una matriz de dimensiones $f\ x\ t$, siendo t el número de *frame* en el dominio del tiempo y f los correspondientes valores en el dominio de la frecuencia. Cada elemento de la matriz $X_{i,j}$ representa la energía de la señal en la frecuencia i y en el tiempo j.

Segunda etapa: Descomposición NMF

Aplicar la ssNMF para obtener, a partir de la señal mezclada contenida en la matriz X, las matrices W y H de los patrones espectrales y de las activaciones temporales de la señal, como se ha explicado en las secciones anteriores. A partir de ellas, se obtienen dos nuevas matrices, $\widehat{Xc}$ y $\widehat{Xp}$, que son el espectrograma estimado del corazón y del pulmón, respectivamente. Así, la matriz $\widehat{Xc}$ es el resultado del producto matricial entre $\widehat{Wc}$ y $\widehat{Hc}$, que son los patrones espectrales y las activaciones temporales estimadas para el sonido del corazón. De forma análoga, $\widehat{Xp}$ es el resultado del producto matricial entre $\widehat{Wp}$ y $\widehat{Hp}$. Para ello es necesario realizar una clasificación de los elementos de las matrices W y H diferenciando entre sonido de corazón o sonido de pulmón. Esto se realizará en la cuarta

etapa. El procedimiento aplicado en esta etapa se muestra en la ecuación (3.12), donde X es el espectrograma de los sonidos mezclados, Xc y Xp los espectrogramas originales de ambas fuentes, Wc y Hc los patrones espectrales y las activaciones temporales originales del sonido del corazón, $\widehat{Xc}$ y $\widehat{Xp}$ los espectrogramas estimados, $\widehat{Wc}$ y $\widehat{Hc}$ las bases y activaciones estimadas de la señal del corazón y W y H las bases y activaciones obtenidas por la NMF.

$$\begin{aligned} X = Xc + Xp = [Wc \;\; Wp] \begin{bmatrix} Hc \\ Hp \end{bmatrix} \approx \hat{X} = \widehat{Xc} + \widehat{Xp} \\ = [\widehat{Wc}\,\widehat{Wp}] \begin{bmatrix} \widehat{Hc} \\ \widehat{Hp} \end{bmatrix} = WH \end{aligned} \tag{3.12}$$

Como ya se comentó, en [24] la función de coste usada es la divergencia de Kullback-Leibler. La ecuación (3.13) muestra las reglas de actualización para las matrices W y H en el proceso iterativo de factorización, donde $\oslash$ denota la división de matrices elemento a elemento. Finalmente, el criterio de parada es el mostrado en la ecuación (3.10).

$$\begin{aligned} W &\leftarrow W \odot ((X \oslash WH)H^T \oslash (1_{F,T}H^T)) \\ H &\leftarrow H(W^T(X \oslash WH) \oslash (W^T 1_{F,T} + \lambda)) \end{aligned} \tag{3.13}$$

Tercera etapa: Cálculo del patrón rítmico del corazón

Esta etapa detecta el patrón de los sonidos del corazón a partir de la señal de entrada con una mezcla de sonidos del corazón y del pulmón. El resultado será una secuencia de ceros y unos, donde 1 denota que se ha localizado un sonido del corazón y 0 lo contrario.

Es sabido que los sonidos de ambos sistemas se solapan en la banda 60Hz-320Hz, y que la energía de los provenientes del corazón se concentra en su mayor parte en la banda 10Hz-144Hz. En consecuencia, la búsqueda del patrón rítmico del corazón se centra en la banda de frecuencias por debajo de 60Hz.

En [32] se propone realizar la estimación de la frecuencia cardiaca, utilizando los cambios de energía que se producen en el la banda de frecuencia 0Hz-60Hz del espectrograma que contiene los sonidos de ambas señales. [32] utiliza una función *Onset* basada en el cálculo del flujo espectral. A continuación, se explica el procedimiento para estimar la frecuencia cardiaca propuesto en [32].

El primer paso consiste en calcular el espectrograma de la señal que contiene los sonidos del corazón y pulmón. Para ello se utiliza la STFT con una longitud de ventana n (ecuación (3.14)), donde fs representa la frecuencia de muestreo de la señal de entrada.

$$n = \frac{fs}{8} \tag{3.14}$$

El resultado de la STFT es la matriz X, que representa su espectrograma. La búsqueda de sonidos del corazón en X se realiza considerando únicamente la parte de la matriz con información relativa a las frecuencias inferiores a 60Hz. El resultado es la matriz S (ecuación (3.15)).

$$S = X(1{:}\,F_{60Hz}, 1{:}\,T) \tag{3.15}$$

Seguidamente se calcula el flujo espectral de la matriz S, usando la ecuación (3.16), donde $i = 2 \dots n_l$, siendo n_l el número de ventanas temporales usadas en la STFT.

$$H(i) = \sum_{f=1}^{F_{60Hz}} |S(f,i)| - |S(f,i-1)| \tag{3.16}$$

A continuación se calcula la función Onset con un rectificador de media onda, obteniéndose la matriz R (ecuación(3.17)), donde los valores obtenidos siempre serán mayores o iguales que 0.

$$R(i) = \frac{H(i) + |H(i)|}{2} \tag{3.17}$$

El siguiente paso consiste en realizar una búsqueda de los máximos locales de R. Para ello se utiliza el algoritmo *Automatic Multiscale-based Peak Detection* (AMPD) [33], que consiste en:

1. Construir un *escalograma* de máximos locales, aplicando a R un análisis de regresión lineal, para calcular la recta que mejor se adapta a la tendencia de los datos de la señal. Seguidamente se inicializa una matriz M, de dimensiones $l\ x\ m$ donde m es la longitud de la señal y l se calcula como $(m/2)-1$. Cada elemento de la matriz se inicializa con una valor $r+1$, donde r son valores aleatorios uniformemente distribuidos.

   A continuación, se buscan máximos locales en k escalas, con $k=1\ldots l$, tal que para cada escala el rango viene determinado por $i=k+2\ldots m-k-1$. En los recorridos se modifican los valores de la matriz M siguiendo la ecuación (3.18). El resultado es que la matriz M contiene ceros en las posiciones donde se encuentren los máximos locales.

$$m_{k,i}=\begin{cases}0 \ \text{ si } \ R_{i-1}>R_{i-k-1} \ \wedge \ R_{i-1}>R_{i+k-1}\\ \text{No se modifica en otro caso}\end{cases} \tag{3.18}$$

2. Obtener el vector φ siguiendo la ecuación (3.19). φ es la suma de todos los elementos de cada escala de la matriz M. La posición del vector φ con el valor más bajo sirve para identificar la escala donde se encuentra el mayor número de máximos locales. Dicho valor se denominará λ.

$$\varphi_k=\sum_{i=1}^{N} m_{k,i} \tag{3.19}$$

3. Construir la matriz Mr copiando las filas de M desde la primera hasta la λ. Por último, se calcula la desviación típica de cada columna de la matriz Mr. Las posiciones de la matriz donde dicho valor sea igual a cero son los lugares donde se localizan los picos de la función R. Utilizando los picos de la función R es posible determinar los instantes temporales donde se localizan los sonidos del corazón.

En [32] se indica que, en algunas situaciones, el primer pico no es detectado por el algoritmo AMPD. Para solventar este problema se propone un mecanismo de corrección a partir del periodo cardiaco basado en el cálculo de la auto correlación de la señal R. El

periodo cardiaco se calcula como la diferencia de tiempo entre el máximo absoluto y el siguiente máximo de la función. Considerando el primer pico detectado por el algoritmo AMPD como $tiempo(1)$, y el segundo pico como $tiempo(2)$, se añadirá, o no, un primer pico siguiendo la ecuación (3.20).

$$\text{añadir} \begin{cases} \text{tiempo}(1) - \text{periodo}, & \text{si tiempo}(1) - \text{periodo} > 0 \\ \text{tiempo}(2) - \text{periodo}, & \text{si tiempo}(2) - \text{periodo} > 0 \end{cases} \tag{3.20}$$

Por último, para cada uno de los picos se buscan las ventanas temporales más cercanas. De esta forma se obtienen los índices i e $i-1$. Así, es posible construir el patrón de los sonidos del corazón, que se corresponde con un vector de tamaño n_l, que es el número de ventanas utilizadas.

El primer ruido cardiaco (R1) tiene una duración aproximada de $140ms$, mientras que el segundo ruido cardiaco (R2) dura unos $110ms$. Para cubrir ambos sonidos completos, es necesario seleccionar el número de ventanas temporales en función del tamaño de la ventana, considerando el factor de solapamiento, que en este caso es del 50%. Para un tamaño de ventana 512, la diferencia temporal entre cada ventana se sitúa en $32ms$. Por ello, es necesario utilizar 5 ventanas temporales para cubrir ambos sonidos del corazón. Si el tamaño de la ventana se aumenta a 1024, la diferencia de tiempo entre cada ventana es de $64ms$, siendo necesarias 3 ventanas para cubrir los dos tipos de sonidos del corazón. Con las consideraciones mencionadas ya es posible construir el patrón de sonidos del corazón usando la ecuación (3.21).

$$m_l \begin{cases} 1 \text{ si } \begin{cases} (i-1)-1\text{:}\,(i)+2 & \text{si } N_{NMF} = 512 \\ (i-1)\text{:}\,(i)+1 & \text{si } N_{NMF} = 1024 \\ (i-1)\text{:}\,(i) & \text{si } N_{NMF} = 2048 \end{cases} \\ 0 \text{ en otro caso} \end{cases} \tag{3.21}$$

Cuarta etapa: Clasificación de bases

Partiendo de las matrices W y H obtenidas en la ssNMF, esta fase aplica una serie de pasos destinados a clasificar (agrupar, *clustering*) las bases (W) y las activaciones (H) entre sonidos del corazón y del pulmón. Aplica dos métodos, basados en medidas espectrales, para la matriz W y un método, basado en medidas temporales, para la matriz H. Estos son:

- **Correlación Espectral Supervisada** (*Supervised spectral Correlation*, SC). Calcula la similitud entre los patrones espectrales (W) obtenidos por la ssNMF y un diccionario (Wg) obtenido a partir de señales del corazón no mezcladas. Cada columna de la matriz W se corresponde con los patrones espectrales de una base. Se compara columna a columna usando la distancia del coseno (ecuación (3.22)) para obtener la correlación espectral, donde $SC(i,j)$ es un valor en $[0,1]$ próximo a 1 o 0 dependiendo de si la base pertenece a un sonido del corazón o del pulmón. Para cada base $W(:,i)$ se calcula el máximo de sus $SC(i,j)$ y, si supera un umbral establecido, se clasificará como un sonido del corazón. En caso contrario se clasifica como sonido del pulmón.

$$SC(i,j) = \frac{W(i) \times Wg(j)}{\|W(i)\| \, \|Wg(j)\|} = \frac{\sum_{f=1}^{F} W(f,i) Wg(f,i)}{\sqrt{\sum_{f=1}^{F} W^2(f,i)} \sqrt{\sum_{f=1}^{F} Wg^2(f,i)}} \qquad (3.22)$$

- **Roll-off** (RO). La densidad espectral de potencia (PSD) permite conocer cómo se distribuye la energía de una determinada señal sobre la banda de frecuencia. En [24] se explica que dicha distribución es diferente en las señales del pulmón y corazón. Por lo tanto, dicha información se puede utilizar para clasificar las bases $W(:,i)$ entre corazón y pulmón. La señal del corazón concentra gran parte de su energía en el rango de frecuencias $[0 - fo]$ y el pulmón en el rango $[fo - fs/2]$, donde fs es la frecuencia de muestreo y fo es la frecuencia máxima donde es posible encontrar sonidos del corazón. En [24] se recomienda utilizar un valor de 260Hz para fo. La ecuación para el cálculo del RO usada se muestra en (3.23).

$$RO(i) = \sum_{t=1}^{T} \sum_{f=0}^{fo} \|X_i(f,t)\|^2 \quad (3.23)$$

La componente $i-ésima$ de la señal que contiene la mezcla de corazón y pulmón se obtiene a partir de W y H, realizando la inversa de la FFT del producto de estas dos matrices. De esta forma se reconstruye el espectrograma y se calcula su RO en el rango $[0-fo]$. Para realizar la clasificación entre corazón y pulmón se establece que la energía total en el rango $[0-fo]$ debe ser un 85% de la existente en $[0-fs/2]$. Es decir, la base $W(:,i)$ pertenece al corazón o al pulmón dependiendo de si su $RO(i)$ es inferior (pulmón) o superior (corazón) al resultado de la ecuación (3.24).

$$Er(i) = 0{,}85 \times \sum_{t=1}^{T} \sum_{f=0}^{fs/2} \|X_i(f,t)\|^2 \quad (3.24)$$

- **Correlación Temporal** (*Temporal Correlation*, TC). Como ya se ha comentado, los sonidos del corazón concentran su energía en la banda de frecuencia 10Hz-320Hz, mientras que los sonidos del pulmón la concentran en la banda 60Hz-1000Hz. Existe, por tanto, solapamiento en la banda 60Hz-320Hz. Como la energía del corazón es predominante en frecuencias inferiores a 60Hz, la TC trabaja en dicha banda para detectar sonidos del corazón. En primer lugar, obtiene el patrón del ritmo cardiaco a partir del espectrograma, donde los sonidos del pulmón y corazón se encuentran mezclados, usando la información presente en la banda 0Hz-60Hz. En segundo lugar, construye una función basada en los cambios de energía de la señal y se buscan picos que puedan corresponderse con un latido del corazón. Con ello, se obtiene un valor 1 si en ese instante de tiempo existe un sonido procedente del corazón, y un 0 en caso contrario. Esto genera un patrón que se puede utilizar para estimar el ritmo cardiaco. Para poder comparar las activaciones H con el patrón del ritmo cardiaco generado es necesario realizar un pre-procesado de la matriz de activaciones H, como se indica en la ecuación (3.25).

$$H(i,t) = \begin{cases} 1 \quad \text{si} \quad H(i,t) \geq \sum_{z=1}^{T} \frac{H(i,z)}{T} \\ 0 \quad \text{si} \quad H(i,t) < \sum_{z=1}^{T} \frac{H(i,z)}{T} \end{cases} \tag{3.25}$$

De esta forma la matriz H estará compuesta por ceros y unos, igual que el patrón de ritmo cardiaco. El siguiente paso calcula el coeficiente de correlación entre cada una de las i bases presentes en la matriz H y el patrón del ritmo cardiaco. Para ello se utiliza la ecuación (3.26), donde $\sigma_{H(i)}$ es la desviación típica de la activación $i-$ $ésima$ de la matriz H, $\mu_{H(i)}$ la media de la activación $i-ésima$ de la matriz H y σ_p y μ_p la desviación típica y la media del patrón del ritmo, respectivamente.

$$TC(i) = \frac{1}{T-1}\left(\sum_{j=1}^{T} \frac{H(i,j) - \mu_{H(i)}}{\sigma_{H(i)}}\right) \times \left(\frac{P(i,j) - \mu_p}{\sigma_p}\right) \tag{3.26}$$

Cada $TC(i)$ es un valor en el rango $[-1,1]$, donde -1 denota que el sonido pertenece al pulmón y 1 que pertenece al corazón. Se trata de una variable continua, por lo que se establece 0 como valor frontera, de forma que $W(:,i)$ pertenece a un sonido del pulmón o de corazón según se indica en la ecuación (3.27).

$$W(i) \rightarrow \begin{cases} \text{pertenece a } \widehat{Wc} \text{ si } TC(i) \geq 0 \\ \text{pertenece a } \widehat{Wp} \text{ si } TC(i) < 0 \end{cases} \tag{3.27}$$

Los tres métodos explicados se pueden aplicar individualmente o de forma combinada para clasificar cada una de las bases presentes en la matriz de patrones espectrales W. En particular, en [24] se utiliza una combinación de los tres métodos, de tal forma que, para clasificar $W(i)$ como sonido del corazón, se debe satisfacer, al menos, la condición de uno de los métodos.

Etapa 1	1. Calcular el espectrograma X para obtener la representación de la señal en dominio de la frecuencia.
Etapa 2	2. Inicializar las matrices W y H con valores aleatorios no negativos. 3. Mientras (3.10) hacer 3.1 Actualizar matriz W con ecuación (3.13) 3.2 Actualizar matriz H con ecuación (3.13) Fin mientras
Etapa 3	4. Calcular el patrón rítmico del corazón
Etapa 4	5. Para cada base $W(:,i)$ 5.1 Determinar si la base $W(:,i)$ pertenece a $\widehat{Wc}$ o a $\widehat{Wp}$ utilizando el método SC. 5.2 Determinar si la base $W(:,i)$ pertenece a $\widehat{Wc}$ o a $\widehat{Wp}$ utilizando el método RO. 5.3 Determinar si la base $W(:,i)$ pertenece a $\widehat{Wc}$ o a $\widehat{Wp}$ utilizando el método TC. Fin para cada
Etapa 5	6. Calcular el espectrograma $\widehat{Xc} = \widehat{Wc}\widehat{Hc}$ con las matrices obtenidas de la clasificación realizada en la Etapa 4. 7. Calcular el espectrograma $\widehat{Xp} = \widehat{Wp}\widehat{Hp}$ con las matrices obtenidas de la clasificación realizada en la Etapa 4. 8. Reconstrucción de la señal del corazón en dominio del tiempo utilizando la inversa de la STFT del espectrograma $\widehat{Xc}$. 9. Reconstrucción de la señal del pulmón en dominio del tiempo utilizando la inversa de la STFT del espectrograma $\widehat{Xp}$.

Tabla 3.1. Tabla resumen del algoritmo de separación de sonidos cardio-pulmonares.

Quinta etapa: Reconstrucción de la señal

En este punto se cuenta con las matrices $\widehat{Wc}$ y $\widehat{Hc}$. Realizando el producto matricial de ambas matrices se obtiene $\widehat{Xc}$, que es el espectrograma estimado de la señal del corazón. Para el caso de los pulmones se obtiene la misma información.

Cada fuente de sonido estimada $\widehat{Xc}(t)$ y $\widehat{Xp}(t)$ se puede obtener a partir de la mezcla original $X(t)$ utilizando las máscaras de *Wiener*, que representan la contribución de

energía que realiza cada fuente a la mezcla $X(t)$. Las máscaras se calculan según muestra la ecuación (3.28).

$$Mc = |\widehat{Xc}|^2 \oslash \left(|\widehat{Xc}|^2 + |\widehat{Xp}|^2\right)$$
$$Mp = |\widehat{Xp}|^2 \oslash \left(|\widehat{Xc}|^2 + |\widehat{Xp}|^2\right) \quad (3.28)$$

Posteriormente se reconstruyen los espectrogramas de los sonidos del corazón y del pulmón a partir del espectrograma original siguiendo la ecuación (3.29). Por último, se calcula la inversa de la STFT de $\widehat{Xc}$ para obtener la señal del corazón en dominio del tiempo.

$$\widehat{Xc} = Mc \odot X$$
$$\widehat{Xp} = Mp \odot X \quad (3.29)$$

Esta sección finaliza mostrando en la Tabla 3.1 un resumen los diferentes pasos seguidos en el algoritmo de separación.

3.8 Tecnologías y herramientas de computación

La computación de altas prestaciones (*High Performance Computing*, HPC) consiste en un conjunto de técnicas, herramientas y procedimientos para obtener la mayor eficiencia posible, para un problema concreto, usando sistemas complejos (con varios nodos, cada nodo con varios núcleos de proceso -procesadores o *cores*- asistidos por aceleradoras –GPUs generalmente-).

Para explotar al máximo las capacidades de estos sistemas (computadores), es necesario utilizar simultáneamente el mayor número de recursos computacionales. Esto se consigue mediante la Computación Paralela (*Parallel Computing*, PC). Para ello, es necesario dividir el problema en tantas partes como núcleos de proceso contenga el sistema. De esta forma, cada parte del problema se asigna a un núcleo y todos ellos trabajan de forma simultánea en la resolución del problema. La descomposición del problema completo en diferentes tareas es un procedimiento que requiere un análisis detallado, ya que es relativamente fácil generar código correcto, pero altamente ineficiente. Es necesario estudiar las

dependencias de las tareas y realizar un correcto balanceado de la carga, intentando que todas ellas tengan el mismo tamaño para conseguir la máxima eficiencia posible. Si a esto se añade la hibridación (el sistema consta de varios computadores conectados por una red de datos) y la heterogeneidad (cada computador consta de procesadores de arquitecturas distintas -CPU + GPU-) resulta evidente que es necesario un estudio detallado, riguroso y complejo para evitar pérdidas de rendimiento.

A la vista de lo comentado en las secciones anteriores de esta memoria, es evidente que la separación de las señales sonoras del corazón y del pulmón, tal como se aborda en este Trabajo Final de Grado, conlleva una elevada carga computacional. Si se añade la restricción, o el deseo, de que todo el proceso se puede efectuar en tiempo real, incluso en llevables cuyas capacidades computaciones son inferiores a los computadores *clásicos*, es evidente que en la implementación de los algoritmos diseñados se debe recurrir al uso masivo de la PC con técnicas de HPC. Seguidamente se describen las tecnologías y herramientas usadas.

OpenMP

Proyecto iniciado por Sillicon Graphics Inc. para arquitecturas de memoria compartida. En este tipo de arquitecturas no existen restricciones de acceso a las diferentes áreas de memoria por parte de los núcleos del procesador. OpenMP[3] proporciona una API que es soportada, entre otros, por lenguajes de programación tales como C/C++ y Fortran. En resumen: OpenMP facilita la implementación de programas paralelos en memoria compartida.

Su funcionamiento se basa en el uso de directivas del compilador, por lo que es necesario que dicho compilador tenga soporte para OpenMP. En la actualidad GNU C/C++ y Clang tienen soporte para esta API. Cabe destacar que las últimas actualizaciones del NDK

[3] www.openmp.org

de Android han dejado soportar GNU C/C++, siendo Clang el compilador por "defecto" del NDK de Android. Por ello, es imprescindible que la versión de Clang usada soporte OpenMP, dado que Android, y sus derivados, son el sistema operativo base de los dispositivos móviles/llevables usados en este trabajo, todos ellos con una CPU con varios núcleos de proceso compartiendo memoria. Además de las directivas, también ofrece una serie de funciones y variables de entorno que se pueden utilizar en el programa.

OpenMP soporta paralelismo a nivel de dato, de tarea e, incluso, vectorial (*Single Instruction, Multiple Data*, SIMD). OpenMP se encarga de la de creación de los hilos, donde se ejecutan las diferentes tareas de forma concurrente, y del reparto de hilos/tareas/etc. buscando un correcto balanceo. Aunque tiene fijado un comportamiento por defecto (estrategia), es posible modificar su comportamiento, utilizando ciertas directivas, o parámetros de directivas, existentes a tal efecto.

El punto más conflictivo para el programador reside en la gestión de la memoria (variables compartidas). La forma en que los algoritmos exploten la Localidad Espacial y la Localidad Temporal, unido a cómo gestionen la contención, la coherencia, los conflictos de acceso, etc., determinará la eficiencia (el rendimiento) del algoritmo.

FFTW

FFTW[4] es una biblioteca (librería en el lenguaje informático cotidiano) eficiente de código abierto, escrita en C, para el cálculo de la Transformada Discreta de Fourier en una o varias dimensiones. Puede trabajar tanto con datos reales como complejos. También incluye la Transformada Discreta del Seno y la Transformada Discreta del Coseno. Puede realizar las transformaciones de forma paralela, usando OpenMP, explotando de esta forma las características del sistema donde se ejecuta para obtener la mayor eficiencia posible.

[4] www.fftw.org

Es posible utilizar FFTW en plataformas que contengan un compilador C. En concreto, se puede utilizar con el NDK de Android usando Clang como compilador. Además, las últimas versiones de FFTW incluyen soporte para arquitecturas ARM, habilitando su uso en dispositivos móviles y/o llevables.

BLAS/LAPACK

BLAS (*Basic Linear Algebra Subprograms*)[5] / LAPACK (*Linear Algebra PACKage*)[6] son bibliotecas de Álgebra Lineal inicialmente implementadas en Fortran. En la actualidad, dispone de una interfaz para C y de una re-implementación casi completa en C. Dichas rutinas se encuentran organizadas en tres niveles. El primer nivel (BLAS 1) contiene las operaciones entre vectores, el segundo (BLAS 2) las operaciones entre vectores y matrices y el tercero (BLAS 3) operaciones a nivel matricial. Entre las operaciones soportadas se encuentran productos vectoriales y matriciales, normas vectoriales y matriciales, resolución de sistemas de ecuaciones, factorizaciones, cálculo de valores singulares/propios, mínimos cuadrados, etc.

Las rutinas incorporan técnicas de PC y HPC para explotar al máximo las características del sistema donde se ejecuta. Las operaciones de nivel 3 (BLAS 3) son extremadamente eficientes, al explotar tanto la localidad espacial como la temporal.

La primera implementación de BLAS fue realizada por *NETLIB*. *GotoBLAS* es una implementación de código abierto de BLAS (actualmente no se encuentra en desarrollo), que incorpora algunas técnicas para mejorar la comunicación entre procesadores, aumentando así la eficiencia del procesamiento. *OpenBLAS*, basada en GotoBLAS 1.13, es

[5] www.netlib.org/blas/

[6] www.netlib.org/lapack/

la biblioteca usada en este trabajo que añade, respecto a GotoBLAS, las rutinas de LAPACK y soporte para arquitecturas ARM de 32 y 64 bits.

4. Trabajo realizado

En la presentación de este trabajo ("Motivación") se fijaba el objetivo general, que era:

"Diseñar e *implementar* un prototipo que, combinando señales heterogéneas, permita la monitorización continua de ciertos parámetros de los Sistemas Cardiovascular y Respiratorio"

El sistema usaría los sensores de wearables periféricos para detectar los niveles de actividad y obtener información del sistema circulatorio, mientras que un micrófono pegado al pecho gobernado por un microcontrolador, capturaría el audio del corazón y de los pulmones (sensor central). La información, periférica y central, sería procesada y combinada para la toma de decisiones. La fusión de información, y su procesado, se podría realizar en los llevables periféricos, si su potencia computacional fuese suficiente, en otro dispositivo adicional (p. ej. un smartphone) o, si fuese necesario y factible, usando servicios en la nube.

Como ya se ha explicado, la irrupción del SARS-CoV-2 y la posterior pandemia mundial, unido al *Estado de Alarma* decretado en España, trastocaron el planteamiento inicial en varios planos: a) en la adquisición del material, b) en el acceso a los laboratorios de la Universidad de Oviedo para la construcción y validación de los prototipos y c) en la movilidad, limitando fuertemente las pruebas de campo (obtener medidas de los distintos estados de actividad con varias personas de ambos géneros y franjas de edad).

Se decidió, ante la incertidumbre y el desconocimiento sobre cuándo la pandemia (y el Estado de Alarma) remitiría, replantear los objetivos del trabajo, para evitar perjuicios en el avance de la formación académica, siendo los nuevos objetivos:

- Realizar trabajos para monitorizar la actividad y ciertos parámetros del sistema cardiovascular con llevables periféricos usando sensores ópticos. Una mínima labor de campo: obtención y análisis de datos reales.

- Diseñar un prototipo central para la captura de audio. Pruebas de su funcionamiento y comprobación de que cumple los requisitos necesarios para una potencial, y futura, integración en un sistema global.
- Implementar algoritmos eficientes para procesar, detectar y separar en tiempo real audio sintético (procedente de fuentes externas) de corazón y pulmón.

En definitiva, abordar el trabajo planteado inicialmente, pero sin integrar los sistemas ni profundizar en la validación y corrección de los modelos resultantes de la investigación realizada.

En las siguientes secciones se detallan los aspectos más relevantes del trabajo realizado para cada nuevo objetivo, partiendo, obviamente, de los fundamentos teóricos descritos en el capítulo "Estado del arte" de esta memoria.

4.1 Separación de sonidos cardio-pulmonares

El objetivo era construir un dispositivo que recogiese los sonidos emitidos por el corazón y los pulmones (sensor central). Sonidos que serían procesados en tiempo real para extraer información relevante de cada sistema. La construcción englobaría tanto la parte hardware como la vertiente software. Finalmente se procedería con las pruebas de validación y ajuste necesarias.

4.1.1 Prototipo hardware

Como ya se ha comentado reiteradamente, la reciente pandemia COVID-19 y los *Estados de Alarma* decretados a nivel mundial han hecho imposible cumplir al 100% las tareas relacionadas con la vertiente hardware de este objetivo. Concretamente con la construcción sólida del prototipo hardware y con las pruebas de validación y de integración. No obstante, sí se ha llevado a cabo un trabajo importante que, a continuación, se detalla.

En primer lugar, se realizó un estudio de mercado para determinar el tipo de micrófono más adecuado, atendiendo a las características del sonido a capturar, su tamaño, su facilidad de uso, su precisión, calidad, etc. Seguidamente, se concluyó un estudio similar para detectar el *microcontrolador* más apropiado, es decir, que soportase a nivel hardware/software el micrófono seleccionado, con las funcionalidades necesarias, facilidad de programación, lo más estándar posible, capacidades de comunicación, aspectos físicos como el tamaño, etc.

Para el micrófono, se observó que el abanico de posibilidades no era muy amplio, encontrándose en el mercado varios modelos que, en esencia, eran todos iguales. Se optó, finalmente, por adquirir varios INMP441.

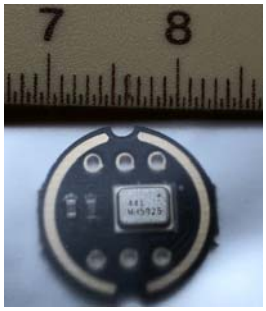

Figura 4.1. Imagen del micrófono INMP441.

El INMP441 (ver Figura 4.1) es un micrófono omnidireccional plano, de tamaño muy reducido, alto rendimiento, baja potencia, salida digital, acondicionamiento de señal, convertidor analógico/digital, filtro anti-solapamiento, administración de energía, con un sensor MEMS (*MicroElectroMechanical Systems*) e interfaz I2S (*Inter-IC Sound* o *Integrated Interchip Sound*) estándar de 24 bits. La interfaz I2S permite conectarlo directamente a procesadores digitales como DSPs (*Digital Signal Processor*) y microcontroladores sin necesidad de un códec de audio. Según sus especificaciones, su relación señal/ruido es de 61 dBA, la sensibilidad de -26 dBFS, el consumo de 1.4 mA, PSR (rechazo de fuente de alimentación) de -75 dBFS, respuesta de frecuencia de banda ancha plana estable en el intervalo [60 Hz, 15 kHz]. En resumen, una buena opción para aplicaciones de campo cercano, como es el caso de este trabajo.

En lo que respecta al microcontrolador, la elección fue un ESP32 (ver Figura 4.2). El ESP32 es un SoC (*System on Chip*) diseñado por la compañía *Espressif* y fabricado por *Taiwan Semiconductor Manufacturing Company, Limited* (TSMC). Integra en un único chip un procesador *Tensilica Xtensa* de doble núcleo de 32bits a 160Mhz (con posibilidad de hasta 240Mhz), conectividad WiFi y Bluetooth, encriptación por hardware, reloj de tiempo real (RTC), puertos, etc. En el mercado hay actualmente numerosas placas de desarrollo que integran el ESP32 con baterías LiPo (Litio y Polímero), pantallas TFT-LCD (*Thin Film Transistor-Liquid Crystal Display*) y OLED (Organic Light-Emitting Diode), sensores de todo tipo, etc. En definitiva, un dispositivo de enorme potencial que se ha convertido en un estándar dentro de las aplicaciones de IoT (*Internet of Things*).

Figura 4.2. Imagen del microcontrolador ESP32.

Otras características del ESP32 interesantes para este trabajo son que dispone de un conversor analógico digital (*Analog-to-Digital Converter*, ADC) de 12bits y 18 canales, 2 conversores digital analógico (*Digital-to-Analog Converter*, DAC) de 8bits, 11 conversores analógico/digital de 10 pines, 3x UART (Transmisor-Receptor Asíncrono Universal, *Universal Asynchronous Receiver-Transmitter*), 2x I2S, 4x SPI (*Serial Peripheral Interface*) y 2x I2C (bus y protocolo *Inter-integrated Circuit*).

En cuanto a la programación del ESP32 hay varias opciones como, por ejemplo, el IDE (Entorno de Desarrollo Integrado, *Integrated Development Environment*) de Arduino[7], MicroPython[8] (variante de Python 3 optimizada para microcontroladores), etc.

Adquiridos los elementos hardware y construido el prototipo, como se muestra en la Figura 4.3, se implementaron programas, usando el IDE de Arduino, para comprobar la integración del INMP441 con el ESP32, capturar audio y, en definitiva, testar su correcto funcionamiento. Las pruebas se realizaron a diferentes frecuencias de muestreo, todas ellas dentro del rango útil descrito en el capítulo 3 de esta memoria, con audio reproducido por un equipo musical. Se comprobó que la captura era correcta y estable en las frecuencias usadas.

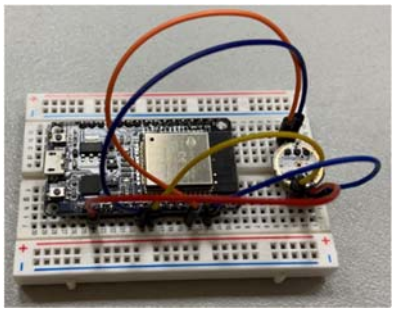

Figura 4.3. Prototipo hardware para la separación del sonido cardio-pulmonar.

No hubo posibilidad de realizar más pruebas (p. ej. capturar audio del pulmón/corazón en tiempo real en condiciones ambientales cotidianas) ni avanzar más en la parte de integración y/o comunicaciones, por las limitaciones ya comentadas.

4.1.2 Algoritmos, complejidades e implementaciones

Dada la naturaleza de esta parte del trabajo, no limitada por las restricciones comentadas para el hardware, aquí sí se han cumplido los objetivos fijados. Se han implementado algoritmos eficientes para procesar, detectar y separar audio en tiempo

[7] www.arduino.cc

[8] micropython.org

real. Como la integración completa no se ha podido efectuar, el audio de corazón y pulmón usado será sintético, procedente de fuentes externas, almacenado como ficheros WAV (o WAVE, apócope de *Waveform*).

Más concretamente:

- Se han implementado en Matlab[9] (*MATrix LABoratory*) / Octave[10] todos los algoritmos necesarios. Esto ha permitido disponer con rapidez de un prototipo para validar los resultados de cada elemento de computación del proyecto, también del conjunto, hacer distintos tipos de pruebas, visualizar resultados, etc. Además, estos códigos se han usado como punto de referencia para comprobar la corrección de las implementaciones en otros lenguajes y entornos.
- Se han implementado en lenguaje C los algoritmos para la captura, separación, reconstrucción y grabación (también en formato WAV) de las señales sonoras. Estas implementaciones recurren masivamente al uso de la PC (*Parallel Computing*) y del HPC (*High Performance Computing*) para cumplir con la restricción de tiempo real. Los métodos y funciones se han implementado contemplando que los casos de uso en producción pueden ser tanto sistemas clásicos (computadores personales o servidores) como SoC llevables (relojes inteligentes, smartphones, etc.). Por tanto, se ha verificado su compatibilidad tanto con el sistema operativo Linux como con Android.
- Se han realizado pruebas de rendimiento y se ha comprobado que las implementaciones en C son correctas y en tiempo real.
- Finalmente, se ha implementado una App demostrativa, pero completamente funcional, para dispositivos compatibles con los sistemas operativos Android. Desde la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/* se puede descargar la aplicación, así como acceder a su código fuente.

[9] www.mathworks.com

[10] www.gnu.org

Los algoritmos diseñados, y sus implementaciones, siguen los principios descritos en la sección 3.7. Los códigos no se incluyen en esta memoria por cuestiones de operatividad, están disponibles en la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*, pero sí se detallan los aspectos más destacables de cada uno de ellos, así como un análisis completo de su complejidad temporal. Para su explicación se seguirá un esquema de etapas similar al presentado en la sección 3.7 (ver Tabla 3.1).

Fase 0: Lectura y escritura del audio en ficheros WAV

El audio con la mezcla de sonidos del corazón y de los pulmones está almacenado en ficheros tipo WAV. Los ficheros WAV constan de una cabecera con meta-información (p. ej. número de canales, frecuencia de muestreo, bits por muestra, etc.) seguida de los datos. Los ficheros WAV sencillos (formato *Pulse Code Modulation*, PCM), los usados en este trabajo, representan los datos en formato entero con 16 bits por muestra y canal, es decir, utilizando el tipo *short* se pueden representar todos los valores posibles. No obstante, los algoritmos diseñados trabajan con números reales de precisión razonable (tipo *double*). Por tanto, en el momento de la lectura/escritura se deben convertir los datos de un tipo a otro, usando algún tipo de normalización.

La documentación de Matlab explica en qué consiste su proceso de normalización, pero, se ha observado que la implementación, en realidad, es diferente. Se ha comprobado que lo que Matlab hace es dividir los valores leídos por el valor absoluto del máximo, que se puede representar en aritmética entera con 16 bits, esto es, 32768. Obviamente, en la generación (escritura) de ficheros WAV se aplica el proceso inverso, multiplicar por 32768. Esta normalización no es formalmente correcta, el rango del tipo short es [-32768, …, 32767], pero por compatibilidad con Matlab será la que se use.

Como se ha comentado, la primera etapa de la separación aplica un proceso de *ventando de Hanning* antes de calcular las FFTs. Por tanto, partiendo del número de

muestras, la frecuencia de muestreo, el porcentaje de solapamiento entre ventanas y el tamaño de ventana deseado, se calcula el número de ventanas (*frames* en terminología coloquial) usando la ecuación (4.1), donde N_{Frame} será el número de frames, N_{Sample} el número de muestras en el fichero, N_{window} el tamaño de ventana y Ov el porcentaje solapamiento entre ventanas. El término $\mathrm{nextPow2}(N_{window})$ también se denominará como N_{FFT}, porque será el tamaño o resolución que se usará en la FFT y es, simple y llanamente, la siguiente potencia de 2 del tamaño de ventana usado. Esto se hace así por razones de eficiencia computacional, como se verá más adelante. No obstante, los valores usados para N_{window} son, en general, potencia de 2, por lo que se cumple que $N_{window} = N_{FFT}$. Más concretamente, en esta fase $N_{window} = N_{FFT} = 512$.

$$N_{Frame} = \mathrm{floor}\left(\frac{N_{Sample} - \mathrm{round}(\mathrm{nextPow2}(N_{window}) \times Ov)}{N_{window} - \mathrm{round}(\mathrm{nextPow2}(N_{window}) \times Ov)}\right) \quad (4.1)$$

El resultado de la lectura será una matriz (X), de dimensiones $N_{window} \times N_{Frame}$, con el audio normalizado según el proceso descrito y estructurado en ventanas (la escritura es equivalente pero en sentido inverso). En realidad será una matriz de números complejos, con la parte imaginaria a 0 por ahora, y de dimensiones $N_{NMF} \times N_{Frame}$, siendo N_{NMF} igual a $\mathrm{round}(\mathrm{nextPow2}(N_{window}) \times Ov) + 1$. Estos cambios vienen impuestos por las necesidades de la FFT y de la factorización matricial aplicada. Nótese que N_{NMF} es, aproximadamente, la mitad de N_{window}, es decir, es una optimización en el consumo de memoria. Considerando todo lo dicho ($\mathrm{nextPow2}(N_{window}) = N_{window} = N_{FFT}$, $Ov = 0.5$ y que $N_{NMF} = \mathrm{round}(\mathrm{nextPow2}(N_{window}) \times Ov) + 1 \cong \frac{N_{FFT}}{2}$) la ecuación (4.1) se puede aproximar por $N_{Frame} = \mathrm{floor}\left(\frac{2N_{Sample} - N_{FFT}}{N_{FFT}}\right) \cong \frac{2N_{Sample}}{N_{FFT}} - 1 \simeq \frac{N_{Sample}}{N_{FFT}}$. Esta acotación ($N_{Sample} = N_{FFT} N_{Frame}$) se usará en futuras explicaciones. Esta fase no precisa del uso de la PC ni aplicar HPC, es un proceso de lectura de un fichero secuencial.

Fase 1: Obtener el espectrograma

Consiste en obtener la representación de la señal en dominio de la frecuencia. Partiendo de la matriz X de la fase anterior, el proceso se reduce a computar N_{Frame} transformadas de Fourier de tiempo reducido (STFT) unidimensionales, una por cada frame de forma

independiente, previa adecuación de los datos (por columnas, por frame), según el proceso descrito en la ecuación (3.11). La aplicación de la ecuación (3.11) consiste en un producto elemento a elemento entre dos vectores, la columna de X que se está evaluando y un vector, de tamaño N_{window}, previamente calculado, aplicando (3.11). Esta fase retorna una matriz con el espectrograma de la señal de entrada. Esta matriz es la propia X (*in-place*), para ahorrar espacio, lo que justifica su elección de "tipo complejo" y con un número de filas igual a N_{NMF}.

Para las STFTs se usa la biblioteca FFTW. La complejidad temporal de una STFT unidimensional es $N_{FFT} \times log(N_{FFT})$, salvo que N_{FFT} sea potencia de 2, en cuyo caso es $N_{FFT} \times log_2(N_{FFT})$. Esta es la razón por la que se ha impuesto que N_{FFT} sea potencia de 2. Nótese, además, que solo las $\frac{N_{FFT}}{2}$ primeras componentes del resultado de aplicar la STFT son necesarias, lo que aclara que el número de filas de la matriz X sea N_{NMF} y no N_{window}.

FFTW también puede calcular las transformadas de forma paralela, en cuyo caso la complejidad temporal se puede aproximar por $\frac{N_{FFT} \times log_2(N_{FFT})}{p}$, siendo p el número de *cores* o procesadores.

En distintas pruebas se observó que ejecutar secuencialmente N_{Frame} veces esta fase, usando la implementación paralela de FFTW, tenía un rendimiento peor que ejecutar en paralelo las N_{Frame} veces usando FFTW de forma secuencial. Es decir, cambiar paralelismo de grano fino (*Fine-Grained Parallelism*, FGP) por "grano grueso" o "compulsivo" (*Coarse-Grained Parallelism*, CGP). Es posible usar CGP porque que el cálculo de las STFTs unidimensionales, y de las operaciones previas/posteriores necesarias, es independiente entre sí. Si, además, se tiene en consideración que el valor de N_{FFT} es relativamente pequeño, 512, queda justificado el mayor rendimiento de CGP. Nótese que si N_{FFT} es pequeño, también lo será N_{window} y, por tanto, la aceleración que se pueda conseguir en

los cálculos previos/posteriores de baja intensidad computacional será reducida o menor que con CGP.

Por todo lo dicho, la complejidad temporal de esta fase se puede aproximar por la ecuación (4.2). Para alcanzar el paralelismo CGP se recurre a OpenMP, concretamente a las directivas "*#pragma omp parallel*" y "*#pragma omp for*", y a la directiva "*#pragma GCC ivdep*" de GNU C para evitar la comprobación de dependencias y maximizar el uso de los registros vectoriales.

$$\begin{aligned} \text{Secuencial} &\rightarrow N_{Frame}\left(N_{FFT} \times log_2(N_{FFT})\right) \\ \text{Paralelo} &\rightarrow \frac{N_{Frame}}{p}\left(N_{FFT} \times log_2(N_{FFT})\right) \end{aligned} \tag{4.2}$$

Es importante destacar que, salvo "*fft_execute*()", las funciones de la API de FFTW no son *thread-safe*. Consecuentemente, el resto de llamadas a la API se deben realizar fuera de las regiones paralelas (en secuencial). Como paso previo a usar fft_execute() hay que crear una estructura, *plan* en terminología FFTW, que contiene todos los datos necesarios para computar la FFT (p. ej. tamaño de la FFT, punteros a las direcciones de memoria con los datos de entrada y salida, sentido de la FFT, etc.). En la creación del plan hay un *flag* de vital importancia, especialmente cuando el tamaño de la FFT es elevado. Este flag puede tomar los siguientes valores: *FFTW_MEASURE*, *FFTW_ESTIMATE* y *FFTW_WISDOM_ONLY*. Se usa para optimizar, computacionalmente, el plan y el cómputo de las futuras FFTs. En otras palabras, busca el algoritmo óptimo para el tipo de FFT que se desea ejecutar bajo dicho plan. Si es FFTW_ESTIMATE, se elige de forma *rápida* un algoritmo entre los implementados en FFTW, que será sub-óptimo. Con FFTW_MEASURE la búsqueda es exhaustiva, y para tamaños de FFT grandes puede tardar horas, mucho más que el tiempo necesario para la futuras FFTs (fft_execute()). FFTW_WISDOM_ONLY permite cargar planes creados, almacenados en fichero, para las condiciones deseadas y optimizados para la computadora donde se ejecutará la aplicación. En otras palabras, con un programa independiente se crean, y guardan en fichero, todos los planes necesarios usando

FFTW_MEASURE. Nótese que mientras no cambien las condiciones, tipo de planes y computadora, estos ficheros no hay que recalcularlos.

Fase 2: Factorización NMF

Aplica la factorización NMF al espectrograma calculado en fase anterior. El resultado serán las matrices W y H correspondientes a los patrones espectrales y a las activaciones temporales, respectivamente. El núcleo de este proceso iterativo es la ecuación (3.13) y el número de repeticiones se determina aplicando la ecuación (3.10). Por simplicidad, pero sin pérdida de generalidad, en esta explicación se asume que el número de iteraciones es *k*.

La NMF recibe 3 matrices, la X calculada en el fase anterior y las matrices W y H inicializadas aleatoriamente. El resultado serán las matrices W y H actualizadas de forma que se cumpla que $\|W_0H_0 - X\|_\beta = \min_{W,H\geq 0}\|WH - X\|_\beta$. Con el operador $\|\quad\|_\beta$ se está indicando que se usa βeta-*divergencia* y no la norma euclidiana.
Cabe recordar que $X \in \mathbb{R}^{N_{NMF}\times N_{Frame}}$, $W \in \mathbb{R}^{N_{NMF}\times N_b}$ y $H \in \mathbb{R}^{N_b\times N_{Frame}}$ con ($N_b \leq \min(N_{NMF}, N_{Frame})$). En este problema N_b es el resultado de sumar 55, número de bases utilizadas para la factorización de sonidos de corazón, y 64, el equivalente para los pulmones. Esto es, el número de bases usado en la factorización es una constante de valor $N_b = 119$. Obviamente, $Y \in \mathbb{R}^{N_{NMF}\times N_{Frame}}$, siendo $Y = W_0H_0$.

En la implementación en C para la ssNMF se ha llevado a cabo un estudio exhaustivo de los cálculos necesarios, atendiendo a las condiciones particulares del problema. En la experimentación de [24], y en otros estudios, se concluye que un valor de $\beta = 1.0$ arroja buenos resultados, lo que habilita ciertas optimizaciones en la ecuación (3.13) (p. ej. elevar a β implica que no es necesario computar dicha operación). Más aún, siempre que ha sido posible se ha buscado plantear los cálculos de la ssNMF como operaciones del nivel 3 de BLAS (operaciones a nivel matricial), por lo que se han realizado transformaciones más allá de la mera aplicación de ciertas propiedades matemáticas como, por ejemplo, que $AB^T =$

$(BA^T)^T$. En concreto, ciertos productos matriciales se han podido reducir a productos matriz por vector, aprovechando todas las particularidades del modelo.

Finalmente, el coste computacional del criterio de parada, ecuación (3.10), es aproximadamente $(N_b + N_{NMF})N_{Frame}$.

Para ilustrar la complejidad temporal de la ssNMF se incluye en la Tabla 4.1 una descripción en términos de las funciones de BLAS usadas, así como indicaciones de OpenMP. En la Tabla 4.1 se ha sustituido N_{NMF} por $\frac{N_{FFT}}{2}$, de igual forma que en la fase anterior, al objeto de minimizar el número de variables en las expresiones de la complejidad temporal. Por tanto, la complejidad temporal del algoritmo secuencial de la ssNMF es el mostrado en la ecuación (4.3), y para la variante paralela en la ecuación (4.4). En las ecuaciones (4.3) y (4.4) se ha aplicado el *Principio de Invarianza* (las constantes no modifican la cota de complejidad) y que en un polinomio la cota de complejidad es la del término de mayor orden.

$$\begin{aligned}(4k+1)N_{Frame}N_b + \frac{3k}{2}N_{FFT}N_{Frame} + \frac{4k+1}{2}N_{FFT}N_{Frame}N_b \\ + \frac{7k+2}{2}N_{FFT}N_b \\ \simeq k(N_{Frame}N_b + N_{FFT}N_{Frame} + N_{FFT}N_{Frame}N_b + N_{FFT}N_b) \\ T_s(N_{FFT}, N_{Frame}, N_b) \simeq kN_{FFT}N_{Frame}N_b\end{aligned} \tag{4.3}$$

$$T_p(N_{FFT}, N_{Frame}, N_b, p) \simeq \frac{kN_{FFT}N_{Frame}N_b}{p} \tag{4.4}$$

Paso	Descripción	Paralelismo	Coste
Inicio	Norma y actualizar W y H	BLAS (dnrm2 + dscal)	$(N_{FFT} + N_{Frame})N_b$
	Inicializar Y	BLAS (dgemm)	$\frac{N_{FFT}N_{Frame}N_b}{2}$
Repite k veces			
Paso 1	Obtener vector auxiliar	BLAS (dgemv)	$N_{Frame}N_b$
	Actualizar Y	OpenMP (parallel for)	$\frac{N_{FFT}N_{Frame}}{2}$
	Obtener matriz auxiliar	BLAS (dgemm)	$\frac{N_{FFT}N_{Frame}N_b}{2}$
	Actualizar W	OpenMP (parallel for)	$\frac{N_{FFT}N_b}{2}$
	Norma y actualizar W y H	BLAS (dnrm2 + dscal)	$(N_{FFT} + N_{Frame})N_b$
	Actualizar Y	BLAS (dgemm)	$\frac{N_{FFT}N_{Frame}N_b}{2}$
Paso 2	Obtener vector auxiliar	BLAS (dgemv)	$\frac{N_{FFT}N_b}{2}$
	Actualizar Y	OpenMP (parallel for)	$\frac{N_{FFT}N_{Frame}}{2}$
	Obtener matriz auxiliar	BLAS (dgemm)	$\frac{N_{FFT}N_{Frame}N_b}{2}$
	Actualizar H	OpenMP (parallel for)	$N_{Frame}N_b$
	Actualizar Y	BLAS (dgemm)	$\frac{N_{FFT}N_{Frame}N_b}{2}$
Parada	Criterio parada	OpenMP (parallel for)	$\left(N_b + \frac{N_{FFT}}{2}\right)N_{Frame}$

Tabla 4.1. Complejidades temporales en el cálculo de la ssNMF.

A la vista de las ecuaciones (4.3) y (4.4), se puede concluir que la paralelización efectuada es adecuada dado que la eficiencia es 1 ($\text{speedup} = p$), al menos desde el punto de vista teórico.

Fase 3: Cálculo de patrones rítmicos

Esta fase obtiene el patrón rítmico de los sonidos del corazón (*Heart Sound Detection*, HSD) aplicando el algoritmo descrito en la tercera etapa de la sección 3.7.

HSD comienza con el equivalente a "**Fase 1: Obtener el espectrograma**", pero usando un tamaño de ventana mayor (1024). Su complejidad temporal se muestra en la ecuación (4.5), expresada en términos del tamaño de ventana (y de FFT) de la Fase 1, aprovechando que es la mitad de la aquí necesitada.

$$\frac{N_{Frame}}{p}\left(2N_{FFT} \times log_2(2N_{FFT})\right) \cong \frac{2N_{Frame}}{p}\left(N_{FFT} \times log_2(N_{FFT})\right) \quad (4.5)$$

El siguiente paso es obtener el vector R mediante una función *Onset* con un rectificador de media onda (ecuación (3.17))). Para ello se calcula el flujo espectral (ecuación (3.16))) considerando únicamente las frecuencias inferiores a 60Hz (ecuación (3.15))). Es por ello que el número de filas de la matriz resultante al obtener el espectrograma, que se denota por X, es $N_{F_{cut}} = \frac{60 \times 1024}{8000} = 8$, donde 60Hz es la frecuencia de corte, 1024 el tamaño de la ventana y 8000Hz la frecuencia de muestreo usada en los datos de entrada. Nuevamente una optimización, en este caso espacial. Computacionalmente hablando es un recorrido para determinar si la componente $i-esima$ (frame $i-esimo$) de R debe ser 0, el flujo espectral (ecuación (3.16))) es negativo, o el propio flujo espectral, si es positivo. Se puede paralelizar porque el cálculo de cada R_i es independiente, siendo su complejidad temporal la mostrada en la ecuación (4.6). Aunque sus cálculos dependen de N_{Frame} y la duración del audio, su intensidad computacional por frame/iteración es baja (pocos y sencillos cálculos que se repiten $N_{F_{cut}} = 8$ veces).

$$\begin{aligned} \text{Secuencial} \rightarrow \ & N_{F_{cut}} N_{Frame} \cong N_{Frame} \\ \text{Parlelo} \quad \rightarrow \ & \frac{N_{F_{cut}} N_{Frame}}{p} \cong \frac{N_{Frame}}{p} \end{aligned} \quad (4.6)$$

Hay partes de HSD que no son especialmente aptas para la paralelización por la falta de intensidad computacional de sus operaciones, por no abundar operaciones de niveles 2 y

3 de BLAS, por dependencias ligadas a resultados de procesos de búsqueda, por recorridos que no explotan la localidad espacial, etc. Es decir, fragmentos de código donde la ganancia al paralelizar cálculos puede ser inferior a la pérdida que supone la creación y gestión de los distintos hilos (*overheads*), que, a su vez, depende del número de *cores* y de su potencia.

En consecuencia, en la paralelización de HSD se han realizado experimentos de calibración. Así, hay fragmentos que en la configuración por defecto se ejecutan en secuencial, y otros que dependiendo del número de frames y/o procesadores se ejecutan en secuencial o en paralelo. Estrictamente hablando se debería decir "en la compilación por defecto" y no "configuración". Esto es así porque en el código fuente en C se han incrustado directivas de compilación condicional que permiten, sin modificar el código, activar/desactivar el paralelismo en ciertas zonas.

El código que calcula el vector R se ejecutará en paralelo cuando el número de procesadores no sea elevado. En consecuencia, en la Tabla 4.2 se muestra la cota superior de complejidad del cálculo de R.

Seguidamente se calcula la auto-correlación del vector R, para estimar el periodo de la señal de entrada. Este nuevo vector, de tamaño $2N_{Frame} - 1$, se denominará Cx. La complejidad temporal para la obtención de Cx se muestra en la ecuación (4.7). Nótese que al ser un vector simétrico es suficiente con calcular la mitad de sus componentes (N_{Frame} y no $2N_{Frame} - 1$).

$$\frac{N_{Frame} \times \left(\frac{N_{Frame}}{2}\right)}{p} = \frac{(N_{Frame})^2}{2p} \tag{4.7}$$

Nuevo Ranilla Posterior a la defensa del TFG (23/07/2020)

La auto-correlación así planteada tiene un coste cuadrático. Cuando N_{Frame} no es muy grande no tiene mayor importancia, por ejemplo con los audios de 7 segundos. El problema surge con audios largos, por ejemplo 700 segundos. Más aún, si se añade al final del

proceso, después de la Fase 5, la estimación de las BPM con el modelo en el dominio del tiempo planteado por Jaén en julio de 2020 el problema se agrava. Esta forma de estimar las BPM computa la auto-correlación de un vector de dimensiones N_{Sample} que, incluso para audios no muy largos, resulta costosa.

En consecuencia, se cambió la forma de computar la auto-correlación de un vector, pasando a usar un planteamiento similar al de Matlab. El procedimiento consiste en aplicar la FFT al vector de entrada, un pequeño cálculo y, finalmente, la inversa de la FFT. En consecuencia, el coste temporal ahora es

$$4N_{Frame}\left(1+\frac{\log_2(N_{Frame})}{p}\right)\approx\frac{N_{Frame}\log_2(N_{Frame})}{p}$$

Nótese que en la acotación final de la complejidad temporal de esta fase este cambio no influye. "*En cualquier caso, HSD necesita una revisión más "profunda" para ver dónde se pierde tanto tiempo*".

Fin Nuevo Ranilla Posterior a la defensa del TFG (23/07/2020)

Estimado el periodo de la señal se deben detectar los picos, para lo que HSD recurre al algoritmo *Automatic Multiscale-based Peak Detection* (AMPD) [33], cuya cota de complejidad secuencial se muestra en la ecuación (4.8). AMPD comienza rellenando aleatoriamente, usando una variante del algoritmo "*Mersenne Twister pseudorandom number generator*" (ver [34] y [43]), una matriz de dimensiones $\frac{N_{Frame}}{2}\times N_{Frame}$, y seguidamente pone a 0 aquellas posiciones tal que sus homónimas en Cx cumplan una determinada condición. Con posterioridad realiza el producto de dicha matriz por un vector, de dimensión N_{Frame} inicializado a unos y, sobre el resultado del producto, busca la posición, de las $\frac{N_{Frame}}{2}$ primeras componentes, cuyo contenido sea el mínimo. Para estimar la cota de complejidad superior del resto de cálculos de AMPD se supondrá que el problema de búsqueda concluye encontrando el mínimo en la última posición ($\frac{N_{Frame}}{2}$). Para detectar los índices (los frames) que identifican los picos de la señal para cada columna de la matriz creada, de N_{Frame} columnas, se efectúan ciertas operaciones escalares que, si

el resultado cumple una determinada condición, implica que se ha encontrado un pico. Es posible realizar en paralelo la detección, el proceso por columna es independiente del resto, pero al depender del resultado de un problema de búsqueda, ser los cálculos de baja intensidad computacional y que el resultado debe ser una única lista (conflictos de acceso que implican secuencialidad en la ejecución), la paralelización de esta parte no es rentable, especialmente si el número de procesadores es alto y los mínimos se encuentran en las primeras posiciones. AMPD concluye ordenando ascendentemente la lista. Lista que, por razones de eficiencia temporal, que no espacial, se ha implementado como un vector.

$$\begin{aligned} &\simeq \left(\frac{(N_{Frame})^2}{2}+\frac{(N_{Frame})^2}{2}\right)+\left(\frac{(N_{Frame})^2}{2}+\frac{N_{Frame}}{2}\right)+\left(\frac{(N_{Frame})^2}{2}\right) \\ &= 2(N_{Frame})^2+\frac{N_{Frame}}{2} \simeq 2(N_{Frame})^2 \end{aligned} \tag{4.8}$$

En (4.8) no está incluido el coste de la ordenación final y los términos están agrupados (entre paréntesis) por funcionalidad. El primer término son las inicializaciones, el segundo el producto matriz por vector y la búsqueda y el tercero la detección de los frames con los picos de la señal. La ecuación (4.9) es la aproximación, cota superior, de variante paralela de AMPD , donde se observa que solamente se paraleliza, por defecto, la operación de nivel 2 de BLAS (producto matriz por vector) y la búsqueda.

$$\begin{aligned} &\simeq \left(\frac{(N_{Frame})^2}{2}+\frac{(N_{Frame})^2}{2}\right)+\left(\frac{(N_{Frame})^2}{2p}+\frac{N_{Frame}}{2p}\right)+\left(\frac{(N_{Frame})^2}{2}\right) \\ &= \frac{3(N_{Frame})^2}{2}+\frac{(N_{Frame})^2+N_{Frame}}{2p} \simeq \frac{3(N_{Frame})^2}{2}+\frac{(N_{Frame})^2}{2p} \end{aligned} \tag{4.9}$$

Comparando (4.8) y (4.9) se observa que la aceleración de AMPD es baja. En la ecuación (4.10) se muestra la eficiencia de AMPD (E_{AMPD}), que como se puede observar, decrece al aumentar p. La aparente contradicción, $E_{AMPD} > 1$ cuando $p = 1$, es fruto de las aproximaciones. Nótese que la eficiencia se define como: "*cociente entre el tiempo secuencial y el tiempo paralelo multiplicado por el número de procesadores usados* ($\frac{T_s}{pT_p}$)".

$$E_{\text{AMPD}} \simeq \frac{2(N_{Frame})^2}{\left(\frac{3(N_{Frame})^2}{2} + \frac{(N_{Frame})^2}{2p}\right)p} = \frac{4(N_{Frame})^2}{(3p+1)(N_{Frame})^2} \simeq \frac{4}{3p} \qquad (4.10)$$

HSD ejecuta 2 veces el algoritmo AMPD. La primera usa el vector Cx como entrada y la segunda R. HSD sobre Cx (AMPD con Cx) tiene como finalidad encontrar los valores que aseguran que la distancia entre sonidos iguales es mayor que un valor lógico de un latido de corazón. Son una serie de búsquedas encadenadas cuya complejidad temporal es, aproximadamente, $3N_{Frame}$. La segunda vez (AMPD con R) el proceso es similar, varias búsquedas y actualización de valores. En la Tabla 4.2 no se incluyen estos costes, ya que son de poca importancia en comparación con el resto, como tampoco se ha incluido el coste de la ordenación final de AMPD. El exceso de la acotación superior es muy superior al coste de una ordenación de muy pocos valores.

Uniendo todas las expresiones de la Tabla 4.2 se obtiene la ecuación (4.11) con las acotaciones para los tiempos secuencial y paralelo de esta tercera fase.

Descripción	Coste
Obtener espectrograma	$\frac{2N_{Frame}}{p}\left(N_{FFT} \times log_2(N_{FFT})\right)$
Función Onset para obtener R	N_{Frame}
Auto-correlación de R	$\frac{(N_{Frame})^2}{2p}$ $\quad$ $\frac{N_{Frame}\log_2(N_{Frame})}{p}$
Algoritmo AMPD (dos veces)	$3(N_{Frame})^2 + \frac{(N_{Frame})^2}{p}$

Tabla 4.2. Complejidades temporales en el cálculo del patrón rítmico del corazón.

$$\begin{aligned} \text{Secuencial} \rightarrow \quad & \simeq N_{Frame}(N_{FFT} \times log_2(N_{FFT}) + N_{Frame}) \\ \text{Paralelo} \rightarrow \quad & \simeq \frac{N_{Frame}}{p}(N_{FFT} \times log_2(N_{FFT}) + N_{Frame}) + (N_{Frame})^2 \end{aligned} \qquad (4.11)$$

Fase 4: Clasificación

Partiendo de las matrices W y H obtenidas con la factorización NMF (Fase 2: Factorización NMF) aquí se clasifican las bases (W) y las activaciones (H) entre los sonidos del corazón y del pulmón. Como ya se comentó, se aplican dos métodos para la matriz W, basados en medidas espectrales, y un método para la matriz H, basado en medidas temporales. Siguiendo la estructura usada en la sección 3.7 se detalla, a continuación, la complejidad temporal de cada método aplicado.

La Correlación Espectral Supervisada (*Supervised Spectral Correlation*, SC) calcula la similitud entre los patrones espectrales (W) obtenidos por la ssNMF y un diccionario (Wg) obtenido a partir de señales del corazón no mezcladas (ecuación (3.22)). El diccionario, las bases, se lee de fichero y se almacenan en una matriz de dimensiones $N_{NMF} \times N_{Bases}$. Para cada $W(:,i)$ SC calcula la norma euclidiana del vector $W(:,i)$, con coste computacional del orden de N_{NMF}, un producto vector ($W(:,i)$) por matriz (Wg), con coste $N_{NMF} \times N_{Bases}$, y unos cómputos adicionales que involucran, nuevamente, la norma de un vector de tamaño N_{NMF} resultando un coste de $N_{NMF} \times N_{Bases}$. Por tanto, el coste computacional secuencial de SC para cada $W(:,i)$ es el que se muestra en la ecuación (4.12).

$$N_{NMF} + N_{NMF}N_{Bases} + N_{NMF}N_{Bases} \simeq 2N_{NMF}N_{Bases} = N_{FFT}N_{Bases} \qquad (4.12)$$

La Correlación Temporal (*Temporal Correlation*, TC) busca la correlación entre la matriz de activaciones (H) y los patrones generados (obtenidos en HSD) usando el coeficiente de correlación de Pearson, una medida de dependencia lineal entre dos variables aleatorias independiente de su escala de medida. Atendiendo a las particularidades del problema, los valores del vector de patrones son ceros y unos, los cálculos de la covarianza, desviaciones y valores medios se han optimizado. Así, para cada vector $H(i,:)$ el coste computacional de TC es, aproximadamente, $3N_{Frame}$.

Roll-off (RO). RO usa la ecuación (3.23) partiendo de que la señal que contiene la mezcla de sonidos es la FFT del producto de las matrices W y H (reconstrucción del espectrograma). Para realizar la clasificación entre corazón y pulmón establece que la

energía total en el rango $[0 - fo]$ debe ser un 85% de la existente en $[0 - fs/2]$. Es decir, la base $W(:, i)$ pertenece al corazón o al pulmón dependiendo de si su $RO(i)$ es inferior (pulmón) o superior (corazón) al resultado de la ecuación (3.24). Así expresado, los cálculos nativos de RO son sencillos, calcular la energía total de la señal, y luego su $RO(i)$ como una operación simple, donde intervienen las primeras componentes cuya energía acumulada sea inferior al umbral. Es decir, la complejidad mostrada en la ecuación (4.13). El primer término es el cálculo de la energía (frecuencia de Nyquist o $fs/2$) y el segundo una cota superior, suponiendo que el umbral es el 100% de la energía.

$$\frac{\text{nextPow2}(N_{Samples})}{2} + \frac{\text{nextPow2}(N_{Samples})}{2} \cong N_{Samples} \simeq N_{Frame} N_{FFT} \qquad (4.13)$$

El detalle relevante en RO es que, previo a los cálculos descritos para la energía, es necesaria la FFT del producto de W y H. Y no sólo eso, en la ecuación (4.13) el tamaño viene determinado por la siguiente potencia de 2 de $N_{Samples}$. Esto quiere decir que, para audios de larga duración, el tamaño que se deberá usar en las FFTs es elevado, concretamente $N_{Frame} N_{FFT}$ con la acotación que se viene utilizando. Por consiguiente, antes de poder calcular los $RO(i)$ hay que computar la FFT, cuya complejidad se muestra en la ecuación (4.14).

$$N_{Frame}(N_{NMF} + N_{Frame} N_{FFT} \times log_2(N_{Frame} N_{FFT}) + N_{FFT}) \qquad (4.14)$$

En (4.14) el último término N_{FFT}, que en realidad es N_{window} pero como N_{window} y N_{FFT} son iguales se ha usado N_{FFT} por simplicidad, corresponde a un proceso de normalización necesario porque la biblioteca FFTW "*computing an FFT followed by an IFFT transform (or vice versa) will result in the original data multiplied by the size of the transform*". El otro término (N_{NMF}) no ligado a la inversa es el producto optimizado de W y H. No es el producto matricial, es el escalado de la columna de W por el correspondiente elemento de H dependiendo del frame que se esté procesando.

Computados los tres métodos y, en función de lo que retornen, se decide si asignar, acumular, en los sonidos correspondientes al pulmón o el corazón. El coste de este proceso es $N_{NMF}N_{Frame}$.

$$\cong (N_{FFT}N_{Bases}) + (N_{Frame}) + (N_{Frame}N_{FFT}) + \Big(N_{Frame}\big(N_{FFT} + N_{Frame}N_{FFT} \times log_2(N_{Frame}N_{FFT})\big)\Big) + (N_{FFT}N_{Frame}) \quad (4.15)$$

Sumando los términos se obtiene el coste total que se muestra en la ecuación (4.15) donde, al igual que en las fases anteriores, se ha usado que N_{NMF} es $\frac{N_{FFT}}{2}$. Durante las explicaciones dadas se han usado expresiones como: a) en el método SC "… para cada $W(:,i)$…", b) en el método TC "… para cada vector $H(i,:)$…" y c) en el método RO "… la base $W(:,i)$… No es el producto matricial, es el escalado de la columna de W …". Esto es así porque las explicaciones dadas son por base. Cabe recordar que el número de bases en nuestro problema es $N_b = 119$, el resultado de sumar las 55 utilizadas para la factorización de los sonidos del corazón y 64 para los de los pulmones. En consecuencia, el tiempo total en secuencial, aproximadamente, sería el que se muestra en la ecuación (4.16), que es el producto de N_b por una simplificación de la (4.15).

$$\cong N_b(N_{FFT}N_{Bases} + N_{Frame}N_{FFT} + N_{FFT}log_2(N_{FFT})(N_{Frame})^2) \quad (4.16)$$

En las explicaciones de esta fase se ha obviado, intencionadamente, hablar de "paralelismo". Esto es así porque, tal vez, la decisión tomada al respecto se entienda mejor una vez detalladas todas las fases y vistas las ecuaciones. Se podrían haber paralizado todos y cada uno de los métodos de forma individual (paralelismo de grano fino, *Fine-Grained Parallelism*, FGP) pero, como se ha visto, hay partes del código poco aptas (operaciones de baja intensidad, saltos divergentes, falta de localidad espacio-temporal, etc.). Pasaría algo similar a HSD (ver la baja eficiencia mostrada en la ecuación (4.10)). Después de varios experimentos de calibración se ha optado por el paralelismo compulsivo (paralelismo de grano grueso, *Coarse-Grained Parallelism*, CGP) aprovechando que los cálculos para cada valor de b se pueden realizar de forma independiente. No obstante, las operaciones matriciales se siguen planteando con la biblioteca OpenBLAS. Por tanto, aunque el

paralelismo anidado de OpenMP no está habilitado, sí que se activa en las llamadas a OpenBLAS. Como se verá en los experimentos, el despliegue de paralelismo dentro del paralelismo (CGP+FGP), tal como se ha planteado, no supone pérdida de rendimiento.

La estrategia de paralelización seguida añade una sobrecarga temporal final y eleva el consumo de memoria. La cantidad de memoria necesaria para duraciones de audio más que razonables no es alta, consecuentemente, la memoria no es determinante. La sobrecarga temporal corresponde a la fusión, unión, de los sonidos del corazón, y de los pulmones, que cada uno de los procesadores ha obtenido en paralelo. Esta fusión consiste en p productos vector por vector (BLAS *cblas_?axpy*) de tamaño $N_{NMF}N_{Frame}$, pero al hacerse de forma paralela usando, nuevamente, p procesadores, el resultado es el que se muestra en la ecuación (4.17).

$$\frac{p(N_{NMF}N_{Frame})}{p} \cong \frac{N_{Frame}N_{FFT}}{2} \simeq N_{Frame}N_{FFT} \tag{4.17}$$

Sumando las ecuaciones (4.16) y (4.17) se obtiene la cota de complejidad temporal secuencial de esta 4 fase (ecuación (4.18)), siendo el tiempo paralelo el mostrado en la ecuación (4.19) y la eficiencia la indicada en la ecuación (4.20).

$$\simeq N_b(N_{FFT}N_{Bases} + N_{Frame}N_{FFT} + N_{FFT}log_2(N_{FFT})(N_{Frame})^2) \tag{4.18}$$

$$\simeq \frac{N_b}{p}(N_{FFT}N_{Bases} + N_{Frame}N_{FFT} + N_{FFT}log_2(N_{FFT})(N_{Frame})^2) \tag{4.19}$$

$$E_{\text{Fase4}} \simeq \frac{(4.18)}{(4.19)p} = 1 \tag{4.20}$$

Fase 5: Reconstrucción

En este punto se cuenta con las matrices $\widehat{Wc}$ y $\widehat{Hc}$. Realizando el producto matricial de ambas matrices se obtiene $\widehat{Xc}$, que es el espectrograma estimado de la señal del corazón. Para el caso de los pulmones se obtiene la misma información.

Cada fuente de sonido estimada $\widehat{Xc}(t)$ y $\widehat{Xp}(t)$ se puede obtener a partir de la mezcla original $X(t)$ utilizando las máscaras de *Wiener*, que representan la contribución de energía que realiza cada fuente a la mezcla $X(t)$. Las máscaras se calculan según muestra la ecuación (3.28)). Este proceso se realiza en paralelo, siendo su cota de complejidad secuencial $N_{NMF}N_{Frame} \cong \frac{N_{FFT}N_{Frame}}{2}$ y $\frac{N_{FFT}N_{Frame}}{2p}$ la paralela.

Posteriormente, se reconstruyen los espectrogramas de los sonidos del corazón y del pulmón a partir del espectrograma original, siguiendo la ecuación (3.29)) y se calcula la inversa de la STFT. Los costes temporales son los mismos que los de la ecuación (4.14), por lo que el coste de esta última fase es el que se muestra en la ecuación (4.21). Nótese que en (4.14) no se activa el paralelismo. Aquí tampoco se usa el paralelismo de grano fino, pero sí se realizan en paralelo la reconstrucción del corazón con la de los pulmones, por eso el coste es el mismo.

$$\begin{aligned} &\text{Secuencial} \rightarrow && \cong N_{FFT}N_{Frame}\left(1 + 2log_2(N_{FFT})\right) \\ &\text{Paralelo} \rightarrow && \cong N_{FFT}N_{Frame}log_2(N_{FFT}) + \frac{N_{FFT}N_{Frame}}{p} \end{aligned} \tag{4.21}$$

Fase	Tiempos	Eficiencia
1	$T_s(n) = N_{Frame}N_{FFT}log_2(N_{FFT})$ $T_p(n,p) = \frac{N_{Frame}N_{FFT}log_2(N_{FFT})}{p}$	$E_1 \approx 1$
2	$T_s(n) \simeq kN_{FFT}N_{Frame}N_b$ $T_p(n,p) \simeq \frac{kN_{FFT}N_{Frame}N_b}{p}$	$E_2 \approx 1$
3	$T_s(n) \simeq N_{Frame}N_{FFT}log_2(N_{FFT})$ $T_p(n,p) \simeq \frac{N_{Frame}N_{FFT}log_2(N_{FFT})}{p} + (N_{Frame})^2$	$E_3 \to 0$
4	$T_s(n) \simeq N_bN_{FFT}log_2(N_{FFT})(N_{Frame})^2$ $T_p(n,p) \simeq \frac{N_bN_{FFT}log_2(N_{FFT})(N_{Frame})^2}{p}$	$E_4 \approx 1$
5	$T_s(n) \simeq N_{FFT}N_{Frame}log_2(N_{FFT})$ $T_p(n,p) \simeq N_{FFT}N_{Frame}log_2(N_{FFT}) + \frac{N_{FFT}N_{Frame}}{p}$	$E_5 \to 0$

Tabla 4.3. Acotaciones de tiempos y eficiencia de los algoritmos de separación del sonido.

Para concluir, en la Tabla 4.3 se muestran aproximaciones, tanto para el tiempo paralelo como secuencial, de todas las fases, así como para la eficiencia. Las fases 1, 2 y 4 muestran una elevada eficiencia teórica, mientras que la tercera y quinta tienden a 0 para un número de procesadores elevado y tamaños grandes. Las fases 1, 3 y 5 no son especialmente costosas. Por el contrario, la cuarta fase es la de mayor complejidad temporal, seguida de la segunda en función del valor de k (número de iteraciones empleadas en la factorización matricial).

En consecuencia, cabe esperar una eficiencia empírica alta, siempre y cuando la duración del audio sea elevada y/o el número de procesadores usado adecuado a la magnitud del problema. En la siguiente sección se comprobará si las expectativas de las estimaciones teóricas se cumplen en el plano empírico.

4.1.3 Experimentación

En la experimentación relativa a la separación de sonidos cardio-pulmonares se parte de un fichero, formato WAV, con una mezcla de audio de 7 segundos de duración y de un diccionario (Wg) obtenido a partir de señales del corazón no mezcladas. Para tener evidencias suficientes de aceleraciones, limitaciones, etc. y poder comparar y concluir con cierta seguridad, se han generado ficheros de audio de hasta 10 minutos de duración. Los nuevos audios, también en formato WAV, se construyen replicando un número entero de veces el fichero original. Por tanto, se usarán audios de 7, 14, 28, 42, 56, 119, 182, 301 y 602 segundos, respectivamente.

En la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/* el fichero comprimido "Audios.tgz" contiene los ficheros de audio utilizados, así como el resultado de la separación para el WAV de 7 segundos de duración (archivos "sepCorazon007.wav" y "sepPulmon007.wav").

El hardware usado ha sido:

- Servidor SuperMicro SYS-7049GP-TRT. Sistema NUMA (Acceso a Memoria No-Uniforme, *Non-Uniform Memory Access*) con 2 procesadores Intel® Xeon® Silver 4110 CPU @ 2.10GHz, 8 núcleos o *cores* por procesador, 128 GB DDR4 RAM por procesador, 11 MB caché L3 por procesador, 1024 KB caché L2 por núcleo, 32 KB caché L1d/L1i por núcleo, *Hyper-Threading* desactivado y sistema operativo Linux Centros. En resumen, un servidor potente que se debe usar con precaución en experimentos con programas paralelos debido a su arquitectura NUMA (los dos procesadores pueden acceder a los 256 GB de RAM, pero cada procesador "tiene cerca", gestiona, 128 GB) y a la presencia de 2 procesadores (el acceso a los datos almacenados en las cachés será más o menos rápido dependiendo de dónde estén los datos y los procesos o hilos).
- NVIDIA Jetson AGX Xavier Developer Kit. El AGX dispone de una CPU NVIDIA Carmel ARM v8.2 de 64 bits con 8 núcleos, una GPU Volta con 512 núcleos, 16 GB LPDDR4 RAM, 4 MB caché L3, 8 MB caché L2, almacenamiento eMMC 5.1 de 32 GB y sistema

operativo Linux Ubuntu. Las ventajas del Jetson AGX para este trabajo, además de su reducido tamaño y consumo, son: A) es un SoC equivalente a la mayoría de smartphones, tabletas, relojes e, incluso, pulseras de actividad, siempre y cuando no se exprima todo su potencial (GPU, E/S, etc.), y B) con el comando *nvpmodel* se pueden encender/apagar núcleos de proceso, fijar valores constantes para la frecuencia de trabajo de los núcleos de proceso, GPU, buses, memoria, etc. Es decir, un sistema donde se pueden realizar experimentos en condiciones controladas, a diferencia del servidor SuperMicro donde resulta extremadamente complejo.

- Teléfonos inteligentes. En la Tabla 4.4 se muestran las características más relevantes de los smartphones usados en la experimentación. Todos ellos son SoC con sistema operativo Android y CPUs ARM. El Samsung y el Xiaomi MI8 comparten versión de sistema operativo, pero con capas de software superiores de fabricantes diferentes. La CPU del Samsung es la más compleja, con hasta 3 tipos de procesadores. Es habitual que los teléfonos inteligentes recurran a varios tipos de CPU especializadas en grupos de funcionalidades para optimizar el consumo de batería, ejecutar ciertas tareas, etc. Esto, unido a la dificultad para gestionar eventos en tiempo real en Android, se traducirá en mediciones poco *precisas*. Además, los teléfonos se han usado en condiciones normales, en producción, con todos los servicios estándar activos (wifi, datos, etc.). En otras palabras, no se han ajustado (*rooteado* en argot técnico) al objeto de captar mejor la experiencia real del usuario.

Respecto al software se ha usado:

- Los ya mencionados sistemas operativos Linux y Android. En Linux las distribuciones Centos 7.8.2003 y Ubuntu 18.04.1 LTS (GNU/Linux 4.9.108-tegra aarch6416.04). Para Android, las versiones 6.0 *Marshmallow* y 10.0.
- Compiladores GNU C y Clang. En Centos la versión “gcc version 7.3.1 20180303 (Red Hat 7.3.1-5)” y en Ubuntu “gcc version 7.5.0 (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04)”. En Android la versión “3.4.1” de Clang. Ambos compiladores con soporte para la especificación 4.5 de OpenMP.

- Versiones 3.3.8 y 0.3.6 de las bibliotecas FFTW y OpenBLAS, respectivamente.
- Matlab R2015b (8.6.0.267246) 64-bit (glnxa64) y GNU Octave 3.8.2 "x86_64-redhat-linux-gnu".
- Versión "19.2.5345600" del NDK (Native Development Kit) de Android y la versión mínima del SDK (*Software Development Kit*) es 23.

Smartphone	Características
Samsung Galaxy S10+	CPUs: Exynos M4 de 2 núcleos (2,7GHz) + ARM Cortex A75 de 2 núcleos (2,3GHz) + ARM Cortex A55 de 4 núcleos (1,95GHz). Memoria: LPDDR4 de 8 GB. Almacenamiento: UFS 2.1 de 128 GB. Sistema Operativo: Android 10
Xiaomi Redmi Note 3 Pro	CPUs: ARM Cortex A72 de 4 núcleos (1,8GHz) + ARM Cortex A53 de 4 núcleos (1,2GHz). Memoria: LPDDR3 de 3 GB Almacenamiento: eMMC 5.0 de 32 GB. Sistema Operativo: Android 6
Xiaomi MI8	CPUs: ARM Cortex A75 de 2 núcleos (2,8GHz) + ARM Cortex A55 de 4 núcleos (1,8GHz). Memoria: LPDDR4 de 6 GB. Almacenamiento: UFS 2.1 de 64 GB. Sistema Operativo: Android 10

Tabla 4.4. Smartphones usados en la experimentación con sonidos cardio-pulmonares.

Seguidamente se muestran los resultados obtenidos, agrupados por bloques atendiendo a su objetivo primario.

Matlab: Separación y tiempos

Como se ha comentado, se han implementado los algoritmos en Matlab para disponer con rapidez de prototipos que permitan validar la separación del sonido y comprobar la corrección de las implementaciones con otros lenguajes. No se han realizado experimentos con Octave al no disponer de algunos módulos necesarios o, de tenerlos, ser extremadamente lentos.

Matlab es un paquete (*suite*) muy completo que incluye módulos para la ejecución de programas en paralelo. Además, incluye "*Intel® Math Kernel Library Version 11.1.1 Product*

Build 20131010 for Intel® 64 architecture applications. Linear Algebra PACKage Version 3.4.1". En otras palabras, todas las operaciones de álgebra lineal las realiza en paralelo usando una implementación de BLAS/LAPACK optimizada por Intel®.

Los códigos Matlab se han ejecutado en el servidor SuperMicro SYS-7049GP-TRT, el más potente, al no disponer de versión/licencia para el Jetson AGX. El archivo comprimido "matlab.tgz", de la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*, contiene los códigos usados.

En el espectrograma superior de la Figura 4.4 se observa parte de la distribución de la energía de la señal de entrada de duración 7 segundos. Se aprecia como en la banda de frecuencias inferiores a 500Hz aparece un patrón que se repite de forma periódica, que corresponde con los latidos del corazón. También se aprecia otro patrón en el rango de frecuencias inferiores a 1000Hz, con una densidad espectral más baja, proveniente de los sonidos del sistema respiratorio. Nótese que cada vez que aparece un sonido de este tipo, su duración es notablemente superior a la del corazón. El resultado de la separación son dos nuevos espectrogramas, uno para el corazón (figura intermedia de la Figura 4.4, se aprecia la existencia de un patrón repetitivo en frecuencias inferiores a 250Hz) y otro para los pulmones (figura inferior, donde se observan dos ciclos de inspiración y expiración).

En la Figura 4.5 se muestra la densidad espectral de potencia (PSD) de la señal del corazón. A partir de esta información se pueden obtener los instantes temporales donde se localizan los sonidos del corazón, información útil para estimar la frecuencia cardiaca.

En lo que respecta a la validación de la calidad de la separación, no se ha podido avanzar más allá de la valoración de los expertos en audio, que ha sido muy positiva, y de la apreciación visual sobre la Figura 4.4, dado que solo se ha dispuesto de un ejemplo de audio real (el fichero WAV de 7 segundos, el resto son cualitativamente iguales) con la mezcla de sonidos. Se podría haber usado el resultado de la separación y, después de fusionar ambos ficheros, utilizarlo como audio de entrada. Sin embargo, se consideró que este

procedimiento no aportaría información relevante, además de que estaría viciado de inicio. En un futuro, construido el prototipo final y conjuntamente con los expertos en audio, se diseñarán experimentos para validar con precisión este tipo de resultados, usando las técnicas y herramientas estándar en el tratamiento del sonido.

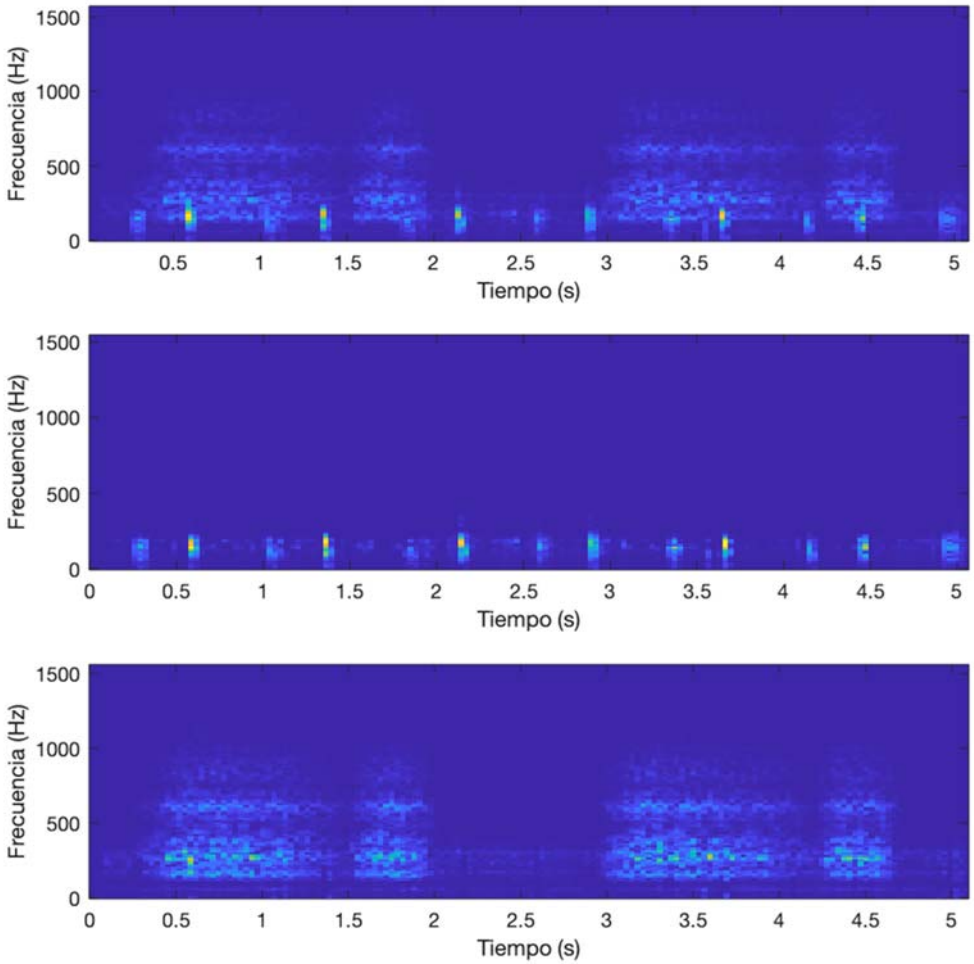

Figura 4.4. Espectrogramas obtenidos con Matlab.

En el plano computacional, las tablas Tabla 4.5 y Tabla 4.6 recogen los tiempos de ejecución de Matlab para todos los audios utilizados en el servidor SuperMicro SYS-7049GP-TRT. En Tabla 4.5 los tiempos paralelos, la ejecución por defecto de Matlab, y en Tabla 4.6 los secuenciales, forzando a Matlab a usar un solo núcleo de proceso. Por su parte, la Tabla 4.7 muestra las eficiencias obtenidas. En todos los casos se presentan los valores totales (tiempos y eficiencias de todo el proceso) como los correspondientes a cada una de las fases descritas en la sección 4.1.2.

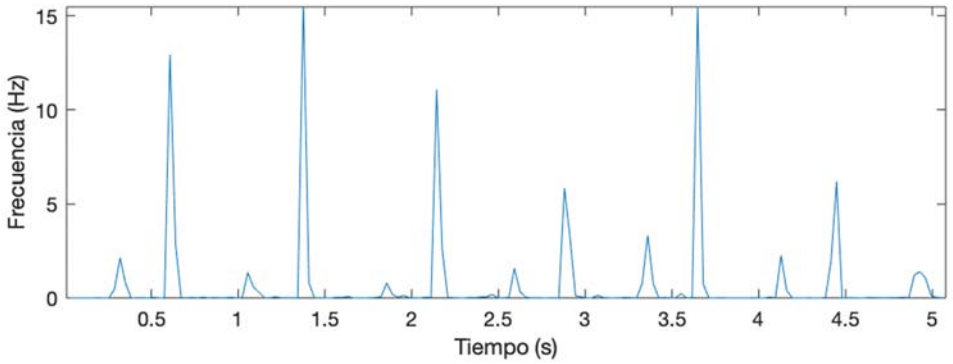

Figura 4.5. Densidad espectral de potencia de la señal del corazón.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,063	0,425	0,314	7,061	0,031	7,924
14	0,069	0,453	0,281	8,038	0,050	8,917
28	0,076	0,624	0,342	9,476	0,073	10,618
42	0,077	0,558	0,309	10,638	0,113	11,723
56	0,126	0,743	0,351	13,756	0,140	15,166
119	0,114	1,124	0,406	22,291	0,293	24,260
182	0,194	1,667	0,691	31,234	0,383	34,221
301	0,188	2,510	1,088	47,291	0,570	51,677
602	0,396	4,095	2,872	92,570	1,114	101,100

Tabla 4.5. Tiempos de ejecución de Matlab en paralelo en el servidor SuperMicro.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,055	0,432	0,301	4,751	0,029	5,606
14	0,084	0,721	0,314	5,743	0,044	6,944
28	0,093	1,139	0,266	7,692	0,079	9,305
42	0,120	1,414	0,274	12,140	0,114	14,111
56	0,122	1,852	0,287	14,228	0,162	16,700
119	0,118	3,624	0,414	23,202	0,280	27,664
182	0,212	5,420	0,593	40,623	0,467	47,365
301	0,295	8,807	1,047	66,193	0,727	77,115
602	0,524	19,728	2,840	139,793	1,442	164,377

Tabla 4.6. Tiempos de ejecución de Matlab en secuencial en el servidor SuperMicro.

A la vista de los resultados de las tablas (Tabla 4.5, Tabla 4.6 y Tabla 4.7) se puede concluir que:

- En todos los escenarios la respuesta de Matlab es en tiempo real. Es cierto que para audios de corta duración la ejecución estándar de Matlab no siempre es en tiempo real: en la Tabla 4.5 el audio de 7 segundos necesita casi 8 segundos. Esto es debido

a la falta de precisión del proceso de medida, como consecuencia de la variabilidad temporal de las sobrecargas. También de la escasa o nula aceleración que se puede obtener al ejecutar en paralelo, en Matlab, problemas de tamaño tan pequeños.

- Solo para audios de larga duración se aprecia que los tiempos paralelos son mejores, pero la ganancia no justifica el coste (el coste se define como "*el producto entre el tiempo de ejecución paralelo y el número de procesadores usados*", una medida de rentabilidad de la inversión). Consecuentemente, las eficiencias son 0 o próximas a cero, valores muy malos que no aconsejan el uso de técnicas de paralelismo.
- Tampoco los tiempos en secuencial son buenos, como se verá al compararlos con los obtenidos en el mismo computador con las implementaciones en lenguaje C.
- Las fases 2 (NMF) y 4 (Agrupamiento) son las más costosas, como cabía esperar. No obstante, la aceleración (eficiencia) de la fase 2 es muy superior a la de la 4, cuando se esperaría que fuesen del mismo orden.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,06	0,06	0,06	0,04	0,06	0,04
14	0,08	0,10	0,07	0,04	0,05	0,05
28	0,08	0,11	0,05	0,05	0,07	0,05
42	0,10	0,16	0,06	0,07	0,06	0,08
56	0,06	0,16	0,05	0,06	0,07	0,07
119	0,06	0,20	0,06	0,07	0,06	0,07
182	0,07	0,20	0,05	0,08	0,08	0,09
301	0,10	0,22	0,06	0,09	0,08	0,09
602	0,08	0,30	0,06	0,09	0,08	0,10

Tabla 4.7. Eficiencias para Matlab en el servidor SuperMicro.

En resumen, Matlab (o la implementación hecha o ambas cosas a la vez) hace una gestión poco eficiente de los recursos, especialmente de la memoria y los procesos/hilos concurrentes, siendo la sobrecarga superior a la ganancia para audios de corta duración. Por ello, se recomienda su uso en el diseño y construcción de los prototipos, pero nunca para la etapa de producción.

Servidor SuperMicro: Implementaciones en lenguaje C

El archivo comprimido "lengC.tgz", de la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*, contiene los códigos creados en lenguaje C. El archivo también incluye el fichero *Makefile* con órdenes para compilar la aplicación bajo diferentes supuestos (paralelo, secuencial, activando o no BLAS, etc.). Cabe recordar que para poder construir la aplicación debe estar instalada la biblioteca FFTW, disponer de una distribución estándar de BLAS/LAPACK y tener soporte para, al menos, la especificación 4.5 de OpenMP.

En primer lugar, se ha comprobado que los resultados de la separación, para todos los audios utilizados, son equivalentes a los obtenidos con Matlab. Nótese que al trabajar con números en coma flotante (número reales en doble precisión) los resultados nunca serán iguales si el número de operaciones que se realizan no son las mismas y en el mismo orden, suponiendo que el resto de factores que intervienen (compilar, versiones de las bibliotecas, arquitectura, norma IEEE, etc.) son los mismos. Las diferencias entre lo obtenido por Matlab y C están dentro del orden de magnitud del error esperado para el tipo de dato usado, esto es, error en torno a 10^{-15} (una magnitud más o menos en función de la duración del audio procesado).

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,003	0,088	0,001	0,305	0,003	0,414
14	0,004	0,173	0,002	0,553	0,005	0,749
28	0,005	0,378	0,006	1,055	0,009	1,474
42	0,008	0,617	0,011	2,613	0,015	3,288
56	0,010	0,825	0,017	2,876	0,019	3,778
119	0,021	1,935	0,056	7,680	0,040	9,782
182	0,029	3,001	0,120	15,780	0,065	19,066
301	0,049	5,184	0,300	31,631	0,106	37,381
602	0,096	10,881	1,107	67,141	0,209	79,642

Tabla 4.8. Tiempos de ejecución secuenciales en lenguaje C en el servidor SuperMicro.

Entrando en los aspectos computacionales, desde la Tabla 4.8 a la Tabla 4.10 se muestran los tiempos de ejecución, secuenciales y paralelos, y las eficiencias. La primera conclusión, comparando con Matlab, es que los tiempos de Matlab son malos, especialmente si se comparan los obtenidos usando paralelismo, llegando a ser Matlab casi

12 veces peor. En secuencial la diferencia es mucho menor, solo 2 veces peor, lo que justifica las bajas eficiencias obtenidas en Matlab y corrobora la afirmación vertida "… gestión poco eficiente de los recursos, especialmente de la memoria y los procesos/hilos concurrentes …".

Y si Matlab cumple con la restricción de tiempo real, obviamente, C también. Aquí, en la implementación en C, el tiempo de respuesta para la peor situación, ejecución secuencial y audios de larga duración, es poco más de la décima parte de la duración del evento. Esto es, hay casi un 90% del tiempo entre eventos disponible para otras tareas o cálculos adicionales. Si el análisis se hace sobre los tiempos de las versiones paralelas el porcentaje crece casi hasta el 98%.

2 Sockets						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,003	0,042	0,001	0,050	0,002	0,111
14	0,004	0,072	0,002	0,077	0,003	0,176
28	0,003	0,123	0,005	0,146	0,007	0,311
42	0,006	0,186	0,011	0,346	0,011	0,588
56	0,005	0,204	0,014	0,384	0,020	0,660
119	0,008	0,395	0,053	0,843	0,033	1,390
182	0,010	0,602	0,119	1,579	0,055	2,447
301	0,014	0,997	0,283	3,063	0,077	4,550
602	0,045	2,852	1,117	6,301	0,165	10,706
1 Socket						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,004	0,052	0,001	0,078	0,002	0,158
14	0,003	0,082	0,002	0,131	0,003	0,236
28	0,004	0,144	0,004	0,237	0,006	0,420
42	0,005	0,190	0,010	0,605	0,018	0,853
56	0,006	0,241	0,016	0,636	0,011	0,940
119	0,011	0,469	0,053	1,459	0,024	2,066
182	0,014	0,719	0,116	2,761	0,036	3,718
301	0,020	1,202	0,277	5,314	0,095	7,026
602	0,035	2,536	1,016	11,431	0,153	15,388

Tabla 4.9. Tiempos de ejecución paralelos en lenguaje C en el servidor SuperMicro.

Se puede observar en Tabla 4.9 (también en Tabla 4.10) que la información se divide en 2 partes, a saber: a) 2 *Sockets* y b) 1 *Socket*. Hay que recordar que este computador dispone de 2 procesadores y que su arquitectura es NUMA (Acceso a Memoria No-Uniforme, *Non-Uniform Memory Access*). Entonces, los tiempos de ejecución de los programas paralelos en memoria compartida (uso de OpenMP) dependen fuertemente de cómo se distribuyan los procesos/hilos entre los procesadores. Así, 2 procesos que "comparten" mucha información deberían ser alojados en el mismo procesador para reducir los tiempos de acceso a memoria, mientras que otros que comparten poco, pero consumen muchos recursos (memoria caché), deberían estar alejados. Además, de una ejecución a otra la distribución de los procesos puede cambiar. Es más, dentro de una ejecución el sistema operativo puede reubicar los procesos tantas veces como estime oportuno. En aplicaciones sencillas hay herramientas para *gestionar* todos estos aspectos, pero en programas complejos, con intereses contrapuestos internamente, es, evidentemente, muy complejo. Por todo ello, las versiones paralelas se han ejecutado bajo dos supuestos. El primero, 2 Sockets, usa los 2 procesadores y le deja libertad al sistema operativo. En el segundo, 1 Socket, solo usa un procesador y fuerza a que los procesos/hilos no conmuten/cambien de núcleo de proceso dentro del procesador (asignación estática).

Entrando de lleno en los datos de las tablas se pueden extraer las siguientes conclusiones, además de lo ya dicho:

- Las Fases 2 y 4 son las más costosas. Esto concuerda con lo observado en Matlab y coincide con lo indicado en el análisis teórico.
- Los tiempos de ejecución paralelos son mejores que los secuenciales. Con 2 Sockets los tiempos paralelos no son la mitad de 1 Socket, pero se aprecia una tendencia positiva.
- En consecuencia, las eficiencias son razonables y se aproximan a las estimaciones teóricas (altas para las fases 1, 2 y 4, y bajas para la tercera y quinta).
- Pasar de 2 Sockets a 1 mejora las eficiencias y empeora el tiempo. Esto es correcto e indica que el diseño es válido y que todavía hay margen de mejora. Es decir, los

tamaños de problema usados no son lo suficientemente grandes para que este sistema llegue al punto de saturación (se están viendo resultados en el régimen transitorio y no en el permanente).

- La Fase 2 (NMF) es intensiva en compartición y dependencia de datos (productos vectoriales, matriciales, etc.), mientras que la Fase 4 es más independiente (paralelismo de grano grueso). Esto justifica la gran mejora de la eficiencia de la Fase 2 al pasar de 2 a 1 Socket, mientras que en el resto la ganancia no es tan significativa. Obviamente, la reflexión en sentido contrario (de 1 a 2) también se puede hacer. Al final, fases con intereses contrapuestos que, en conjunto, se comportan razonablemente bien.

2 Sockets						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,05	0,13	0,06	0,38	0,08	0,23
14	0,06	0,15	0,07	0,45	0,09	0,27
28	0,10	0,19	0,08	0,45	0,08	0,30
42	0,09	0,21	0,07	0,47	0,09	0,35
56	0,13	0,25	0,08	0,47	0,06	0,36
119	0,16	0,31	0,07	0,57	0,08	0,44
182	0,18	0,31	0,06	0,62	0,07	0,49
301	0,23	0,32	0,07	0,65	0,09	0,51
602	0,13	0,24	0,06	0,67	0,08	0,46
1 Socket						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,08	0,21	0,12	0,49	0,19	0,33
14	0,17	0,26	0,17	0,53	0,18	0,40
28	0,17	0,33	0,16	0,56	0,19	0,44
42	0,20	0,41	0,15	0,54	0,10	0,48
56	0,21	0,43	0,13	0,57	0,22	0,50
119	0,25	0,52	0,13	0,66	0,20	0,59
182	0,26	0,52	0,13	0,71	0,23	0,64
301	0,31	0,54	0,14	0,74	0,14	0,67
602	0,34	0,54	0,14	0,73	0,17	0,65

Tabla 4.10. Eficiencias para lenguaje C en el servidor SuperMicro.

Coincidiendo con lo dicho para Matlab, este computador es, tal vez, demasiado potente para la intensidad computacional que exhiben los audios usados (no los audios en sí, su

duración). Se podrían haber realizado experimentos usando un número de núcleos inferior, por ejemplo 2, 4, 6, etc., y de esta forma ajustar la potencia del hardware a la intensidad computacional en función de la duración del audio tratado. Sin embargo, se consideró que este ejercicio teórico no aporta información relevante sobre la experiencia real que el usuario experimentará. En otras palabras, o el usuario no sabrá cómo gestionar los recursos (activar núcleos de proceso, asignar procesos, etc.) o, simplemente, no se preocupará de ello, usará directamente el computador que tiene a su disposición. Los resultados mostrados en las tablas corresponden precisamente a estas situaciones reales de uso (servidores con 1 o 2 procesadores).

Otro aspecto importante es el consumo de memoria. En la Fase 4 de la sección 4.1.2 se comentó que "… La estrategia de paralelización seguida … eleva el consumo de memoria … pero no es determinante". Esto se traduce en que la Fase 4, para lograr la independencia entre los procesos que ejecuta cada core de cada procesador, duplica ciertas estructuras relacionadas con el cálculo de las FFTs (los vectores de entrada, al ser ejecución *in-place* el vector de salida es el de entrada, y los *planes* de la FFTW). Concretamente habrá tantas copias de esas estructuras como número de núcleos de proceso se usen.

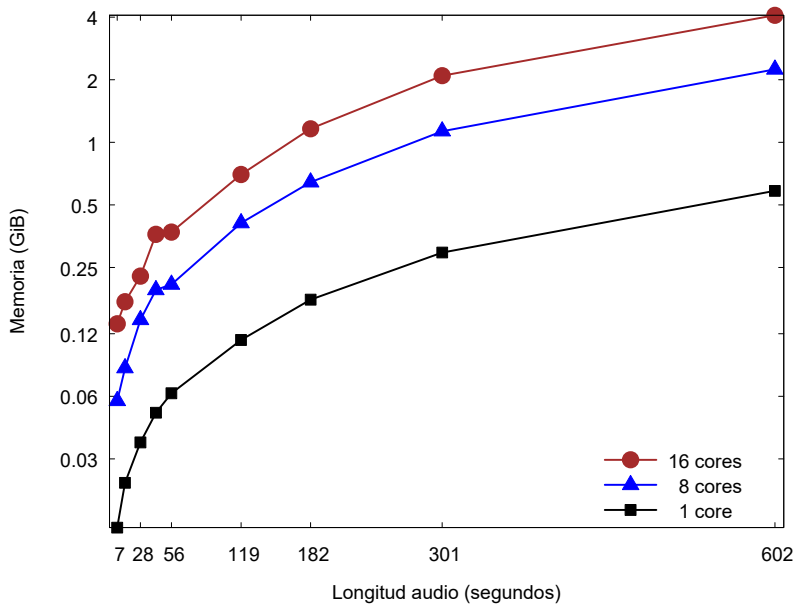

Figura 4.6. Consumo de memoria en el servidor SuperMicro.

La Figura 4.6 muestra la evolución del consumo de memoria RAM[11] en GiB (Giga binary Byte, 2^{30} bytes) en función del número de núcleos de proceso usados y la duración del audio procesado. En la Figura 4.6 la leyenda "16 cores" corresponde con los experimentos usando los 2 Sockets, la leyenda "8 cores" usando 1 Socket y la etiqueta "1 core" corresponde a la ejecución en secuencial. Se observa que hay una cierta proporcionalidad en el consumo: "*para cualquier duración de audio el programa paralelo consume, aproximadamente, la cantidad de memoria usada por el secuencial multiplicada por la mitad del número de cores menos 1*". 4 GiB no es una cantidad importante para este computador, dispone de 256 GB, pero puede ser elevada para equipos menos potentes o wearables. Los equipos menos potentes o llevables no tienen 16 cores, suelen disponer de 4, 6 o, a lo sumo, 8. Así, para 4 cores son necesarios poco más de 0.5 GiB.

Aun siendo un consumo de memoria razonable y, con el ánimo de abarcar el mayor número de dispositivos llevables, la Fase 4 de la sección 4.1.2, que es la que dispara el consumo de memoria, se puede configurar para que use más o menos núcleos de proceso en el despliegue del paralelismo compulsivo o de grano grueso (al compilar se selecciona el escenario adecuado). Nótese que, como se explicó en 4.1.2, esta fase usa un esquema de paralelismo mixto. Primero se fija el número de núcleos de proceso para el bucle externo (paralelismo de grano grueso) y el resto, hasta alcanzar el máximo disponible a nivel físico, será usado cuando sea necesario a nivel interno (*nested parallelism*). Este esquema implica una sobrecarga y puede generar ciertos conflictos de interés (competición por los recursos), pero también un mejor aprovechamiento de los recursos. Todo depende de cómo se alineen la ejecución de las diferentes etapas de cada iteración del bucle externo (relaciones de orden parcial). En todo caso, es sencillo encontrar una configuración que arroje una relación "consumo de memoria" / "tiempo de ejecución" apropiada.

[11] RSS: Resident size memory (non-swapped physical memory) used by the process. La información se ha obtenido ejecutando el comando *system("ps …")* dentro del propio ejecutable.

NVIDIA Jetson AGX Xavier: Implementaciones en lenguaje C

Los códigos usados son los mismos que los utilizados para el servidor SuperMicro (archivo comprimido "lengC.tgz", de la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*), y también el software de apoyo necesario. Como ya se comentó, es un SoC con procesador de arquitectura ARM donde, con facilidad, se controla el encendido/apagado de los cores, la frecuencia de trabajo de los núcleos de proceso, etc. Es decir, un sistema donde se pueden realizar experimentos en condiciones controladas emulando gran parte de los wearables actuales.

Mode 2 (15W): secuencial						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,011	0,382	0,005	2,285	0,014	2,718
14	0,019	0,781	0,012	4,391	0,028	5,251
28	0,033	1,707	0,026	9,264	0,050	11,117
42	0,047	2,789	0,049	10,644	0,064	13,610
56	0,056	3,716	0,074	11,249	0,079	15,194
119	0,111	8,218	0,182	23,095	0,143	31,776
182	0,165	13,041	0,278	44,161	0,225	57,906
301	0,273	22,280	0,752	90,923	0,354	114,631
602	0,540	46,262	3,557	185,865	0,698	237,008
Mode 7 (30W): secuencial						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,006	0,201	0,003	1,203	0,007	1,431
14	0,010	0,416	0,005	2,317	0,013	2,772
28	0,016	0,912	0,012	4,535	0,021	5,516
42	0,023	1,501	0,022	5,257	0,032	6,845
56	0,031	1,989	0,031	5,743	0,040	7,844
119	0,057	4,397	0,100	11,597	0,077	16,242
182	0,083	6,971	0,208	22,928	0,115	30,324
301	0,136	11,960	0,520	46,969	0,187	59,798
602	0,269	24,756	1,946	99,105	0,428	126,549

Tabla 4.11. Tiempos de ejecución secuenciales en lenguaje C en el Jetson AGX Xavier.

Siguiendo la misma metodología, desde la Tabla 4.11 a la Tabla 4.13 se muestran los tiempos de ejecución y las eficiencias, y en la Figura 4.7 los consumos de memoria.

Tanto para los tiempos como para las eficiencias, las tablas son dobles. Las primeras filas, de leyenda "Mode 2 (15W)", corresponden a la configuración por defecto del Jetson. En este modo solo 4 cores de los 8 disponibles están *encendidos*. El sistema ajusta la frecuencia de funcionamiento de los distintos núcleos de proceso en función de la carga y del consumo global del SoC (CPU, GPU, etc.), garantizando que el consumo total no supera los 15 vatios. En "Mode 7 (30W)", últimas filas de las tablas, se ha configurado el sistema con el comando *nvpmodel*, para que todos los cores estén activos con un umbral de frecuencia de trabajo mínima elevado, priorizando la CPU sobre cualquier otro subsistema (p. ej. la GPU). Con estos dos modos se simula la mayoría de smartphones y tabletas actuales.

Mode 2 (15W): 4 cores						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,007	0,207	0,005	0,758	0,008	0,998
14	0,009	0,318	0,010	1,405	0,014	1,777
28	0,015	0,605	0,022	2,870	0,027	3,577
42	0,019	0,911	0,039	2,831	0,033	3,852
56	0,023	1,135	0,057	2,986	0,041	4,264
119	0,038	2,374	0,186	6,063	0,081	8,771
182	0,052	3,759	0,386	11,211	0,119	15,566
301	0,077	6,294	0,962	24,426	0,184	31,997
602	0,151	13,251	3,722	49,319	0,379	66,917
Mode 7 (30W): 8 cores						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,007	0,080	0,003	0,224	0,004	0,324
14	0,004	0,116	0,003	0,420	0,006	0,561
28	0,006	0,208	0,010	0,711	0,012	0,967
42	0,007	0,310	0,016	0,781	0,014	1,141
56	0,008	0,399	0,024	0,878	0,021	1,342
119	0,012	0,753	0,087	1,691	0,042	2,603
182	0,016	1,178	0,198	3,264	0,069	4,748
301	0,024	2,219	0,517	6,715	0,104	9,610
602	0,048	3,835	1,997	13,572	0,236	19,739

Tabla 4.12. Tiempos de ejecución paralelos en lenguaje C en el Jetson AGX Xavier.

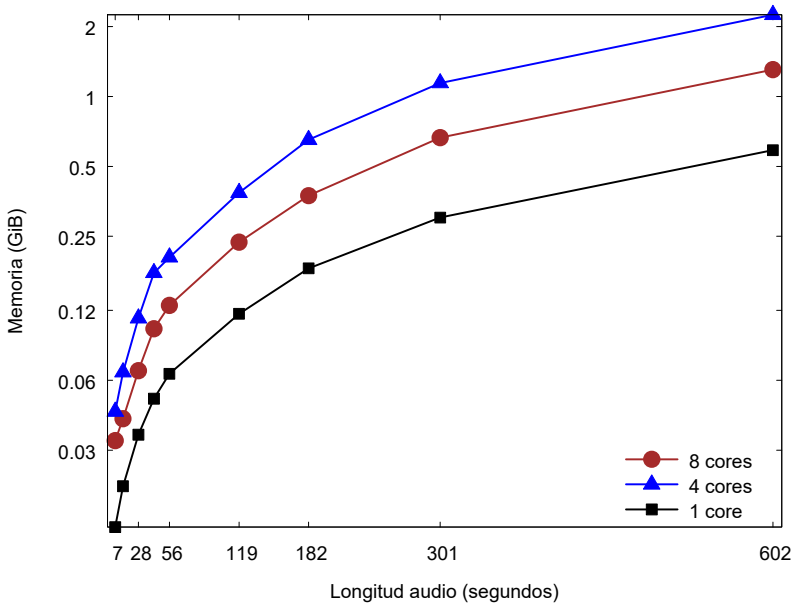

Figura 4.7. Consumo de memoria en el Jetson AGX Xavier.

Mode 2 (15W): 4 cores						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,38	0,46	0,27	0,75	0,47	0,68
14	0,51	0,61	0,30	0,78	0,51	0,74
28	0,56	0,71	0,30	0,81	0,47	0,78
42	0,62	0,77	0,31	0,94	0,49	0,88
56	0,61	0,82	0,33	0,94	0,48	0,89
119	0,73	0,87	0,25	0,95	0,44	0,91
182	0,79	0,87	0,18	0,98	0,47	0,93
301	0,88	0,89	0,20	0,93	0,48	0,90
602	0,89	0,87	0,24	0,94	0,46	0,89
Mode 7 (30W): 8 cores						
Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,10	0,32	0,14	0,67	0,22	0,55
14	0,28	0,45	0,18	0,69	0,28	0,62
28	0,32	0,55	0,15	0,80	0,22	0,71
42	0,39	0,60	0,17	0,84	0,29	0,75
56	0,48	0,62	0,16	0,82	0,24	0,73
119	0,58	0,73	0,14	0,86	0,23	0,78
182	0,65	0,74	0,13	0,88	0,21	0,80
301	0,72	0,67	0,13	0,87	0,22	0,78
602	0,70	0,81	0,12	0,91	0,23	0,80

Tabla 4.13. Eficiencias para lenguaje C en el Jetson AGX Xavier.

A la vista de los resultados experimentales, se puede concluir que el Jetson AGX se comporta como el servidor SuperMicro, aun siendo muy distintos en prestaciones y arquitectura. Por un lado, la evolución del consumo de memoria sigue el mismo patrón y cumple la regla de proporcionalidad comentada en el SuperMicro. Por el otro, la tendencia en los tiempos de ejecución es similar, esto es:

- Las Fases 2 y 4 son las más costosas. Esto concuerda con lo observado y con lo indicado en el análisis teórico.
- Los tiempos de ejecución secuenciales en "Mode 7 (30W)" son casi la mitad de los correspondientes al "Mode 2 (15W)", que concuerda con la posibilidad de trabajar al doble de frecuencia. Los tiempos paralelos en "Mode 7 (30W)" son, aproximadamente, la cuarta parte de los del "Mode 2 (15W)", lo que es correcto al disponer del doble de cores trabajando a casi el doble de frecuencia. Un comportamiento muy lineal y muy parejo al que se deduce desde el plano teórico.
- Comparando con el servidor SuperMicro, los tiempos secuenciales del Jetson en "Mode 7 (30W)" son, aproximadamente, 1.5 veces los del SuperMicro, y lo mismo si la comparación se hace entre las versiones paralelas de las variantes "1 Socket" del SuperMicro con la paralela "Mode 7 (30W)" del Xavier. La relación prestaciones/coste es inmejorable en el Jetson, y lo mismo la relación FLOP/VATIO.
- Obviamente, los tiempos son mejores que los de Matlab y el tiempo de respuesta cumple los requerimientos de tiempo real.
- Las eficiencias son muy buenas, aprovechamiento cercano al máximo teórico estimado, y próximas a las estimaciones teóricas (muy altas para las fases 1, 2 y 4, baja para la tercera y razonable para la última).

En resumen, se puede afirmar que el Jeston AGX Xavier es un sistema más que adecuado para la intensidad computacional que arroja el problema que se está tratando. Cabe esperar, por tanto, que en SoC con una arquitectura similar y simple (un solo tipo de procesador) bajo Android el comportamiento también sea correcto.

Smartphones con Android: Implementaciones en lenguaje C.

Los códigos usados son idénticos a los utilizados en los otros dos sistemas (archivo comprimido "lengC.tgz", de la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*). En este caso es necesario integrarlos en una App utilizando NDK y JNI (*Java Native Interface*). El código de la aplicación se encuentra en la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/* en el archivo comprimido "SeparacionCorazonPulmon.zip". Como ya se comentó anteriormente, los tres dispositivos cuentan con un SoC con procesador de arquitectura ARM. Las CPUs de estos smartphones son complejas, ya que cuentan con hasta 3 tipos diferentes de procesadores incluidas en un mismo dispositivo.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,014	0,571	0,009	2,035	0,016	2,650
14	0,024	0,609	0,018	4,060	0,032	4,751
28	0,040	0,744	0,043	9,049	0,070	9,996
42	0,054	0,610	0,067	19,082	0,104	20,260
56	0,071	0,810	0,096	21,344	0,137	22,470
119	0,139	2,065	0,300	65,745	0,275	68,543
182	0,211	3,198	0,529	131,119	1,101	136,315
301	0,490	6,706	1,693	174,936	1,024	184,918
602	1,211	16,202	6,923	424,349	1,288	450,067

Tabla 4.14. Tiempos de ejecución secuencial en el Samsung S10+.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,032	0,207	0,003	0,918	0,034	1,209
14	0,053	0,301	0,007	1,320	0,076	1,772
28	0,091	0,492	0,024	2,538	0,141	3,314
42	0,067	0,659	0,047	6,091	0,192	7,076
56	0,102	0,798	0,074	6,506	0,496	7,743
119	0,164	1,620	0,289	14,186	0,272	16,552
182	0,110	2,565	0,497	27,576	0,370	31,158
301	0,273	4,539	1,095	54,118	0,486	60,563
602	0,286	11,271	4,418	119,351	0,794	136,221

Tabla 4.15. Tiempo de ejecución en paralelo en el Samsung S10+.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,05	0,34	0,38	0,28	0,06	0,27
14	0,06	0,25	0,31	0,38	0,05	0,34
28	0,05	0,19	0,22	0,45	0,06	0,38
42	0,10	0,12	0,18	0,39	0,07	0,36
56	0,09	0,13	0,16	0,41	0,03	0,36
119	0,11	0,16	0,13	0,58	0,13	0,52
182	0,24	0,16	0,13	0,59	0,37	0,55
301	0,22	0,18	0,19	0,40	0,26	0,38
602	0,53	0,18	0,20	0,44	0,20	0,41

Tabla 4.16. Eficiencias para el Samsung S10+.

Desde la Tabla 4.14 a la Tabla 4.16 se muestran los tiempos de ejecución y las eficiencias para el Samsung S10+. A la vista de los resultados, se puede comprobar que los tiempos obtenidos, tanto en secuencial como en paralelo, son peores que en el Jetson y en el SuperMicro. En este caso no es posible especificar el núcleo de proceso, de los 8 disponibles, donde realizar la ejecución en secuencial.

Además, como se mencionó anteriormente, en Android existe una dificultad para gestionar eventos en tiempo real. Esto, unido a que el teléfono se ha usado en condiciones normales con otros servicios activos, hace que las mediciones no sean tan precisas como en los anteriores dispositivos probados. A pesar de ello es posible extraer algunas consideraciones:

- Para todos los audios y en todas las ejecuciones, la respuesta es en tiempo real.
- Las Fases 2 y 4 continúan siendo las más costosas, hecho que concuerda con el análisis teórico realizado y con los resultados obtenidos por los anteriores dispositivos que aparecen en esta memoria.
- Se puede observar como las eficiencias que se obtienen en este dispositivo son inferiores a las obtenidas en el Jeston, a pesar de que la capacidad computacional de ambos dispositivos es similar. La eficiencia es una relación entre el tiempo de ejecución secuencial y el paralelo. Como el Samsung S10+ incorpora 3 tipos de CPUs, con diferentes arquitecturas y velocidades de reloj, y no se conoce el tipo de procesador usado en la ejecución secuencial, los valores de las eficiencias se deben

interpretar con cierta cautela. Por ejemplo, una ejecución secuencial usando la CPU Exynos M4 a 2,7GHz será, en teoría, un 72% más rápida que usando el Cortex A55 a 1,95GHz. Como la ejecución paralela usa todas las CPUs, la frecuencia real de trabajo viene determinada por la frecuencia de la CPU más lenta, debido a sincronismos, esperas, dependencias, etc. De esta forma, la eficiencia real del diseño se aproximaría a 0.6. En resumen, la eficiencia percibida por el usuario es la mostrada en la Tabla 4.16, pero la real del diseño paralelo es superior.

- Otro indicador de que la eficiencia percibida está afectada por la existencia de 3 tipos de CPUs es que la eficiencia de la Fase 2 es muy baja. La Fase 2 es la que exhibe un paralelismo más acoplado (de grano fino) por lo que es la más sensible a trabajar con CPUs no homogéneas.
- La eficiencia obtenida en la Fase 4 es muy superior a la obtenida en el resto de fases. Esta es la fase más costosa computacionalmente y por ello es positivo que su eficiencia sea lo mayor posible, ya que su impacto en el tiempo de ejecución total es grande.

Para el Xiaomi MI8 los tiempos de ejecución y eficiencias se muestran desde la Tabla 4.17 hasta la Tabla 4.19.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,034	0,183	0,012	3,471	0,018	3,708
14	0,059	0,349	0,026	6,418	0,033	6,867
28	0,083	0,777	0,059	13,231	0,075	14,178
42	0,098	1,188	0,086	35,990	0,119	37,398
56	0,012	1,766	0,156	41,177	0,156	43,589
119	0,233	3,692	0,426	82,708	0,298	87,127
182	0,377	6,166	0,926	157,079	1,234	164,658
301	0,620	10,572	2,213	302,118	1,334	315,693
602	1,255	22,184	7,742	621,458	1,398	652,971

Tabla 4.17. Tiempos de ejecución secuencial en el Xiaomi MI8.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,019	0,190	0,012	1,370	0,042	1,618
14	0,016	0,362	0,020	2,319	0,079	2,737
28	0,097	0,892	0,041	4,102	0,152	5,182
42	0,034	1,343	0,076	10,344	0,202	11,842
56	0,043	1,715	0,111	16,986	0,303	18,424
119	0,086	3,784	0,473	25,583	0,256	29,886
182	0,139	6,008	0,825	49,411	0,446	56,489
301	0,205	9,532	1,963	87,381	0,389	99,247
602	0,397	21,803	7,372	194,491	0,810	224,392

Tabla 4.18. Tiempos de ejecución en paralelo en el Xiaomi MI8.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,23	0,12	0,13	0,32	0,05	0,29
14	0,46	0,12	0,16	0,35	0,05	0,31
28	0,11	0,11	0,18	0,40	0,06	0,34
42	0,36	0,11	0,14	0,43	0,07	0,39
56	0,03	0,13	0,18	0,30	0,06	0,30
119	0,34	0,12	0,11	0,40	0,15	0,36
182	0,34	0,13	0,14	0,40	0,35	0,36
301	0,38	0,14	0,14	0,43	0,43	0,40
602	0,40	0,13	0,13	0,40	0,22	0,36

Tabla 4.19. Eficiencias para el Xiaomi MI8.

Los tiempos obtenidos con el Xiaomi MI8 son peores que los del Samsung S10+. Ni las diferencias en el tipo de CPUs, ni las variaciones en las frecuencias de reloj, justifican que el Xiaomi MI8 sea inferior en prestaciones. Como la versión de Android es la misma, la razón hay que buscarla en las prestaciones del resto del hardware (p. ej. velocidad de la memoria RAM) y/o en las capas intermedias de software del fabricante. A la vista de los resultados obtenidos, se pueden realizar las siguientes consideraciones:

- La respuesta es en tiempo real para todos los audios en las ejecuciones en paralelo. Por el contrario, esto no ocurre en las ejecuciones secuenciales, ya que para los audios de 301 y 602 segundos el tiempo de ejecución es superior a la duración del audio. Nótese que este sistema no mantiene la proporcionalidad entre el tiempo del evento y el tiempo necesario para su procesamiento. En otras palabras, para audios de poco más de 300 segundos el sistema se satura. La saturación también se alcanzará en las ejecuciones paralelas, pero para duraciones muy superiores a las

aquí usadas. En cualquier caso, hay tiempo real en los tamaños estándar de audio usados en este tipo de problemas.

- Las eficiencias obtenidas son ligeramente inferiores a las observadas en el Samsung S10+. Esto es correcto y normal, porque ambos comparten arquitectura y la mencionada saturación del MI8.
- Nuevamente la Fase 4 obtiene una eficiencia elevada comparado con el resto de fases. Este hecho es importante ya que es la fase que mayor incidencia tiene sobre el tiempo de ejecución final.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,060	0,305	0,020	8,325	0,105	8,827
14	0,094	0,583	0,043	10,776	0,079	11,587
28	0,184	1,227	0,096	20,982	0,159	22,667
42	0,175	1,982	0,155	45,813	0,244	48,397
56	0,250	2,631	0,225	50,001	0,374	53,014
119	0,408	6,438	1,075	111,421	0,693	120,074
182	0,518	11,186	1,902	211,773	1,071	226,525
301	0,846	20,989	4,416	425,304	1,985	453,648

Tabla 4.20. Tiempos de ejecución secuencial en el Xiaomi Redmi Note 3 Pro.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,028	0,432	0,013	2,004	0,059	2,535
14	0,020	0,539	0,027	5,197	0,117	5,915
28	0,036	1,200	0,085	7,356	0,208	8,939
42	0,058	1,882	1,089	15,621	0,244	17,491
56	0,075	2,550	0,245	17,836	0,290	20,770
119	0,141	5,527	0,621	45,889	0,457	52,672
182	0,219	9,819	1,627	93,291	0,654	105,692
301	0,347	18,016	3,846	210,321	1,132	233,002

Tabla 4.21. Tiempos de ejecución en paralelo en el Xiaomi Redmi Note 3 Pro.

Por último, desde la Tabla 4.20 hasta la Tabla 4.22 se muestran los tiempos de ejecución y las eficiencias para el Xiaomi Redmi Note 3 Pro.

Duración (sec.)	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Total
7	0,35	0,12	0,26	0,69	0,30	0,58
14	0,78	0,18	0,26	0,35	0,11	0,33
28	0,85	0,17	0,19	0,48	0,13	0,42
42	0,50	0,18	0,02	0,49	0,17	0,46
56	0,56	0,17	0,15	0,47	0,21	0,43
119	0,48	0,19	0,29	0,40	0,25	0,38
182	0,39	0,19	0,19	0,38	0,27	0,36
301	0,41	0,19	0,19	0,34	0,29	0,32

Tabla 4.22. Eficiencia en el Xiaomi Redmi Note 3 Pro.

Es importante mencionar que el Xiaomi Redmi Note 3 Pro es un smartphone de gama media de hace más de 4 años. La frecuencia de reloj de sus 2 tipos de CPU es notablemente inferior a la de los 2 dispositivos Android presentados con anterioridad. Además, la versión de Android en el Redmi es la 6.0, siendo la 10.0 en los otros smartphones analizados.

Debido a que este dispositivo cuenta con una cantidad de memoria inferior, no fue posible realizar pruebas con el audio de 602 segundos. Finalmente:

- La respuesta es en tiempo real para todas las duraciones de audio en la ejecución realizada en paralelo. Esto no se cumple para la variante secuencial, ya que en la práctica totalidad de los audios usados el tiempo de ejecución es superior a la duración del propio archivo de audio. Este hecho pone de manifiesto la importancia de realizar una implementación que aproveche al máximo las capacidades del dispositivo (tanto desde la perspectiva del HPC como de la PC), ya que, de no realizarse la optimización, en este dispositivo no sería posible realizar la separación de las diferentes fuentes sonoras en tiempo real.
- La eficiencia está en la línea de lo observado para los otros dispositivos Android probados.

A modo de conclusión, es posible extraer algunos factores comunes para todos los dispositivos con sistema operativo Android:

- Las eficiencias obtenidas con todos los teléfonos Android son inferiores a las obtenidas por el Jetson Xavier, que es un SoC equivalente a muchos dispositivos

móviles o tabletas. Por ello, se determina que el sistema operativo Android y las diferentes herramientas como el NDK y JNI afectan de forma negativa a la eficiencia de los programas escritos en lenguaje C que utilizan PC (Parallel Computing) y HPC (High Performance Computing).

- Los algoritmos utilizados, cuyas implementaciones recurren al uso de la PC y del HPC, son importantes para cumplir con la restricción de tiempo real. Nótese que en dispositivos que cuentan con CPUs de baja capacidad computacional (p. ej. el Xiaomi Redmi Note 3 Pro), es importante realizar el procesamiento en paralelo, para que sea factible realizar la separación de los sonidos cardio-pulmonares en tiempo real.

4.2 Sistema de detección de la actividad física

Los dos wearables seleccionados para desarrollar este TFG cuentan con un acelerómetro y un giroscopio. El reloj Ticwatch S2 cuenta, además, con un magnetómetro. Todos ellos son sensores hardware, por lo que se encuentran integrados físicamente dentro del dispositivo.

En [11] y [12] se establece que se puede determinar el nivel de actividad con precisión usando únicamente el acelerómetro y el giroscopio. Además, el bajo consumo de energía de ambos sensores hace que este enfoque sea el elegido para obtener los datos necesarios para llevar a cabo esta tarea.

La detección de la actividad física se realiza de forma diferente en la pulsera y en el reloj inteligente. El motivo reside en que no es posible obtener los datos en formato RAW (bruto) de los sensores de la pulsera, mientras que en reloj inteligente sí es posible.

En el caso del reloj inteligente, los sensores devuelven en un vector los valores de los ejes X, Y, Z para cada SensorEvent, correspondiente a cada medición que realiza el sensor. De esta forma se puede acceder a los datos RAW del sensor. Para cada sensor se pueden especificar cuatro frecuencias de muestreo diferentes, aunque el sistema Android

lo *interpreta* como una recomendación. Tras realizar diferentes pruebas con el dispositivo, para cada modo de funcionamiento, se ha concluido que las frecuencias de muestreo son las mostradas en la Tabla 4.23.

Modo	Frecuencia
Normal	5 Hz
UI	25 Hz
Game	62 Hz
Fastest	100 Hz

Tabla 4.23. Frecuencias de muestreo de los sensores Android obtenidas experimentalmente.

Se ha podido comprobar que la frecuencia de muestreo se mantiene más estable en el giroscopio que en el acelerómetro. En el acelerómetro cuando detecta que se están generando valores que corresponden con un movimiento significativo, la frecuencia aumenta a valores que pueden duplicar las frecuencias que aparecen en la Tabla 4.23. También se ha detectado que, al aumentar la frecuencia del acelerómetro, la frecuencia del giroscopio se reduce.

En [12] se afirma que una frecuencia de 8Hz es suficiente para determinar la actividad física. Por ello se indicará al sensor que trabaje en el modo 'UI', para poder tener un buen balance entre el número de muestras y el gasto de batería y recursos del dispositivo. Hay que tener en cuenta que en el caso del acelerómetro la frecuencia puede oscilar entre los $25Hz$ y los $50Hz$.

Los niveles de actividad que se contemplan son los siguientes:

- **Reposo.** Este nivel de actividad se asigna cuando el usuario se encuentra descansando y su nivel de esfuerzo es nulo. El ritmo cardiaco debería encontrarse en valores cercanos al nivel de reposo. El usuario podrá estar sentado, durmiendo o realizando cualquier otro tipo de actividad que no implique ninguna carga física.
- **Actividad moderada.** Este nivel se asigna cuando el usuario se encuentra realizando un esfuerzo leve, que acelera el ritmo cardiaco respecto de su nivel de reposo. En

concreto, en este trabajo, se considera la actividad de caminar como la actividad identificada con un nivel moderado.

- **Actividad intensa.** Este nivel se asigna cuando el usuario está realizando una gran cantidad de esfuerzo, que conlleva un aumento significativo del ritmo cardiaco. En este trabajo se considera la actividad de correr como la actividad de nivel intenso.

4.2.1 Experimentación

Para poder determinar qué medidas estadísticas son las más adecuadas para clasificar el nivel de actividad, una vez obtenidos los datos de los dos sensores, se planteó un experimento. Este experimento consistió en recoger datos de los sensores cuando el usuario está en reposo, caminando a diferentes velocidades y corriendo a distintos ritmos. Para ello se contó con la participación de tres personas, realizándose todas las actividades en *interior* debido a las restricciones impuestas por el estado de alarma derivado de la COVID-19. Cada una de las personas que participó en la recogida de datos realizó el mismo tipo de actividades, que se detallan a continuación.

Cada persona completó un total de tres actividades diferentes, con una duración aproximada de 10 minutos cada una de ellas. Se realizaron, periódicamente, mediciones de 8 segundos de duración obteniendo datos del acelerómetro y giroscopio. Con todos los datos recogidos durante los 8 segundos se aplicó un procesado a los datos consistente en ventanas deslizantes de 2 segundos con 50% de solapamiento, por lo que en 8 segundos existen 7 ventanas con datos. Para cada ventana se calcula el vector magnitud de cada uno de los datos proporcionados por el acelerómetro y giroscopio en formato RAW. Posteriormente se calcula la media, desviación típica, rango intercuartílico y asimetría de cada vector magnitud. En total, cada minuto se recogen aproximadamente 52,5 muestras, por lo que en 10 minutos son 525. En resumen, para cada actividad se recogieron unas 500 muestras por persona.

En primer lugar, se recogieron datos cuando la persona se encuentra en estado de reposo. Para ello, el experimento se realizó con la persona sentada en una silla efectuando pequeños movimientos con la mano para coger algún objeto de encima de una mesa o escribir en un ordenador. De esta forma, se recogen en total unos 1500 datos procedentes de tres personas diferentes asociados al nivel de actividad de reposo.

Posteriormente, se recogieron datos asociados con el nivel de actividad moderado. Para ello, se indicó a los tres individuos que debían caminar a diferentes ritmos durante los 10 minutos que dura la toma de datos. De esta forma se obtiene un conjunto de datos más variado con el que poder realizar un mejor ajuste del sistema. Al igual que para el nivel de actividad de reposo, se obtienen aproximadamente 1500 datos procedentes de tres personas distintas.

Por último, para obtener datos asociados con el nivel de actividad intenso, se indicó a las personas participantes en la prueba que debían correr a diferentes ritmos durante 10 minutos. Al igual que con los anteriores niveles de actividad se obtienen unos 1500 datos.

En total se cuenta con 4723 instancias, cada una de ellas con ocho valores, que se corresponden con la media, desviación típica, rango intercuartílico y asimetría de los vectores magnitud del acelerómetro y giroscopio. Además, el conjunto de datos se encuentra balanceado ya que el número de instancias de cada clase es muy similar para que esto no afecte a la posterior clasificación.

Una vez recogidos todos los datos necesarios, para realizar la clasificación se utilizó el entorno WEKA con cuatro clasificadores diferentes. Se evaluó el rendimiento de los clasificadores J48, RandomTree, Naive-Bayes y SMO. Se utilizó validación cruzada con 10-folds. Los resultados obtenidos fueron muy similares con los cuatro clasificadores. El que obtuvo mejores resultados fue el SMO con un 100% de las instancias clasificadas correctamente. RandomTree obtuvo un 99,98% de las instancias clasificadas

correctamente mientras que J48 consiguió un 99,92%. En último lugar se encuentra el clasificador NaiveBayes con un 99,56%. En la Tabla 4.24 se puede ver la matriz de confusión para el clasificador J48.

	Reposo	Moderada	Intensa
Reposo	1570	1	0
Moderada	0	1533	3
Intensa	0	0	1616

Tabla 4.24. Matriz de confusión para el clasificador J48.

Se puede observar cómo solamente existe una instancia que se clasifica como Moderada, cuando en realidad es Reposo y tres instancias que se clasifican como Intensa, cuando en realidad es Moderada.

Posteriormente se utilizó la librería *sklearn* en *Python* para probar otros algoritmos con los que construir el clasificador para las tres clases. En primer lugar, se utilizó el algoritmo implementado en la clase *DecisionTreeClassifier,* que, como se explicó en el apartado 3.3 de esta memoria, implementa una versión optimizada del algoritmo CART, que es muy similar al algoritmo C4.5. El rendimiento de este algoritmo se evaluó utilizando validación cruzada con 10-folds y validación cruzada dejando uno fuera (Leave-one-out). En el primer caso se obtuvo un 99,996% de las instancias clasificadas correctamente. En el segundo caso el *score* obtenido fue del 99,9998%.

Otro algoritmo de clasificación utilizado fue el implementado en la clase *LogisticRegression* de sklearn. Los resultados obtenidos fueron ligeramente perores que los logrados con el algoritmo anterior, ya que el score en validación cruzada con 10-folds fue del 99,83%, mientras que con validación cruzada dejando uno fuera se obtuvo un 99,85%.

El último algoritmo utilizado de la librería *sklearn* fue el implementado en la clase *MLPClassifier,* que utiliza el perceptrón multicapa. Con validación cruzada con 10-folds se obtuvo un score del 100%. Con validación cruzada dejando uno fuera también se obtuvo un *score* del 100%.

Se ha podido comprobar como los resultados obtenidos con todos los clasificadores utilizados presentan una alta tasa de acierto y por lo tanto es viable realizar la clasificación entre los diferentes tipos de actividad con cualquiera de ellos. Debido a la simplicidad del modelo construido y al gran rendimiento mostrado en las pruebas realizadas, se decide implementar en la aplicación Android el modelo construido por la clase DecisionTreeClassifier de sklearn. Para que el árbol generado cuente con el mayor número de datos posible, se realiza el entrenamiento con todos los datos contenidos en el conjunto inicial, sin realizar una separación entre conjunto de entrenamiento y conjunto de test. Con este modelo solamente se utilizan los datos del acelerómetro, y en concreto las medidas estadísticas de la media, el rango intercuartílico y el grado de asimetría. El árbol resultante es el mostrado en la Figura 4.8.

De esta forma, se demuestra que es posible realizar la clasificación entre los tres niveles de actividad contemplados en este trabajo utilizando únicamente las mediciones que proporciona el acelerómetro. Así se consigue un ahorro de batería, memoria y tiempo de procesamiento respecto a un modelo donde fuera necesario utilizar el giroscopio.

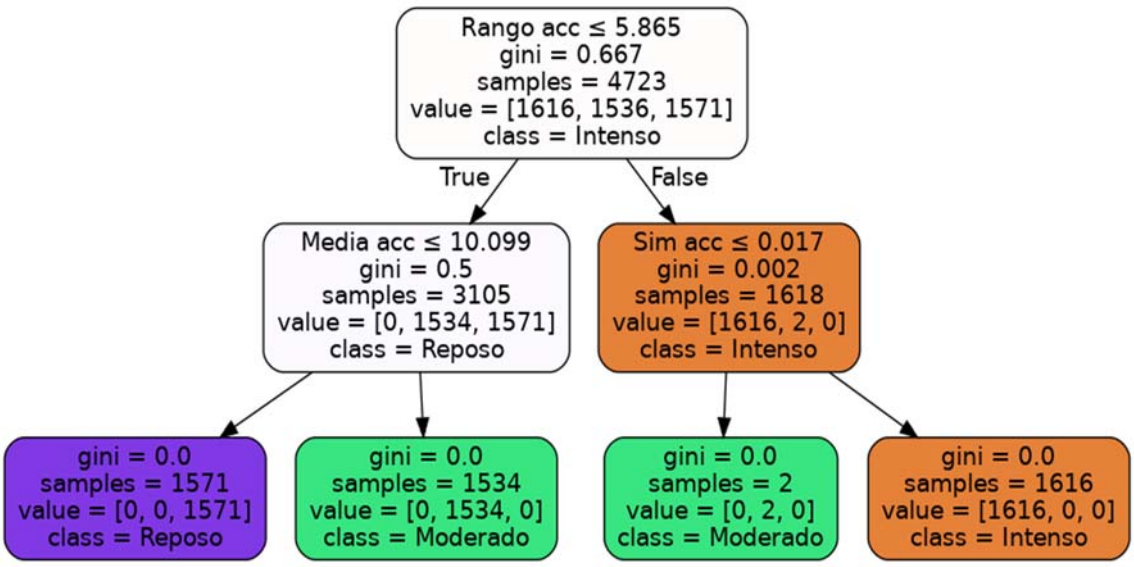

Figura 4.8. Árbol de decisión obtenido para la clasificación del tipo de actividad.

4.2.2 Implementación

La implementación de este sistema es diferente para la pulsera Xiaomi Smart Band 4 y para el reloj inteligente. Como se indicó anteriormente, en el smartwatch es posible obtener los datos en formato RAW de los diferentes sensores, mientras que en la pulsera se puede acceder a información *procesada* internamente en la pulsera a partir de los datos de los sensores. En concreto es posible acceder al número de pasos que ha dado el usuario. Seguidamente se detalla el funcionamiento de este sistema en el smartwatch y posteriormente en la pulsera.

El reloj inteligente se activa cada cierto periodo de tiempo, que puede elegir el usuario, para recoger los datos del acelerómetro necesarios para poder determinar el nivel de actividad en ese instante. De esta forma, el sensor no estará continuamente midiendo, cosa que reduciría notablemente la batería del dispositivo.

Una vez que el reloj se activa para obtener los datos, los sensores permanecen activos durante 8 segundos. En ese tiempo se recogen todos los valores de los tres ejes del sensor para que posteriormente sean procesados. Como se indicó anteriormente, el número de datos recogidos durante esos 8 puede variar ya que, aunque es posible especificar una frecuencia de muestreo, en realidad el sistema lo toma como una recomendación y pueden producirse variaciones en dicho valor.

Para el procesamiento de los datos se utiliza un enfoque con ventana. Los datos se dividen en segmentos de dos segundos y se utiliza un nivel de solapamiento del 50%. Para cada ventana se calcula el vector magnitud de cada medición del acelerómetro y posteriormente la media, el rango intercuartílico y el grado de asimetría. De cada ventana se obtiene un valor que indica el nivel de actividad (reposo, moderado, intenso) utilizando el árbol de la Figura 4.8. De esta forma, en los ocho segundos que permanecen activos los sensores se recogen siete valores del nivel de actividad. Para determinar el nivel de actividad general durante esos 8 segundos se calcula la moda de esos siete valores.

Debido a que la frecuencia de muestreo de los sensores puede variar de forma considerable, en primer lugar, se obtiene el número de muestras recogidas durante esos ocho segundos. En función de ese valor se determina de forma aproximada cual fue la frecuencia de muestreo del sensor dividiendo el número de muestras entre ocho segundos. Una vez conocida la frecuencia del sensor, ya es posible realizar la división de los datos en ventanas de dos segundos. El motivo de realizar este procedimiento reside en que la exactitud al medir los ocho segundos que se encuentran activos los sensores es mucho mayor que la exactitud en la frecuencia de muestreo, que como se ha podido comprobar es muy oscilante.

Para la detección del nivel de actividad física con la pulsera Xiaomi Smart Band 4 se utilizará como *input* los pasos/minuto que realiza el usuario. En este dispositivo no es posible obtener los datos en bruto (RAW) de los diferentes sensores incorporados y por ello se optó por utilizar el número de pasos/minuto que realiza un usuario para determinar la actividad. Este dato no lo proporciona la pulsera directamente, pero es posible calcularlo.

Mediante BLE es posible obtener el número de pasos que ha dado el usuario en el día. Para ello, la pulsera cuenta con un servicio *custom,* no definido en la especificación Bluetooth, a través del cual es posible obtener dicha información. Se realizan consultas periódicas de dicho valor cada cierto tiempo, que puede elegir el usuario. Posteriormente, se calcula la diferencia entre el número de pasos obtenidos en el instante actual y los pasos obtenidos en el anterior instante. Además, se calcula el tiempo transcurrido entre ambas consultas. De esta forma, se pueden calcular los pasos/minuto que ha dado el usuario ya que se cuenta con la diferencia de pasos y la diferencia de tiempo transcurrido entre las dos mediciones. La fórmula utilizada es (4.22).

$$\frac{\text{pasos}_{t_1} - \text{pasos}_{t_0}}{t_1 - t_0} \tag{4.22}$$

Se definen tres niveles de actividad en función del valor resultante del cálculo anterior. Estos niveles son:

- **Reposo**. Este nivel de actividad se asigna cuando el usuario realiza $0\ \mathrm{pasos/min}$. El ritmo cardiaco debería encontrarse en valores cercanos al nivel de reposo.
- **Actividad moderada**. Este nivel se asigna cuando el usuario ha dado algún paso o está caminando. El rango este nivel se encuentra entre $1\ \mathrm{paso/min}$ y $130\ \mathrm{pasos/minuto}$. El ritmo cardiaco se acelera de forma perceptible respecto de su nivel de reposo.
- **Actividad intensa**. Este nivel se asigna cuando el usuario se encuentra corriendo. Para ello el ritmo de pasos/minuto debe ser superior a 130. El ritmo cardiaco aumenta de forma considerable.

4.3 Sistema de monitorización continua

Este sistema se encarga de realizar una monitorización periódica del nivel de actividad del usuario y de su frecuencia cardiaca. Para ello se realizan mediciones cada cierto periodo de tiempo, permitiendo al usuario elegir el valor de dicho periodo entre unas opciones preestablecidas. Cada medición consiste en obtener el valor de la frecuencia cardiaca del usuario en el instante, además de obtener información del acelerómetro en el caso del reloj o el número de pasos en el intervalo temporal para el caso de la pulsera. Con ello se determina el nivel de actividad del usuario en ese momento utilizando el sistema explicado en el apartado 4.2.

Con esa información, se monitoriza el estado del usuario y se realizan diferentes notificaciones en caso de que se detecte alguna anomalía. Cuando el usuario se encuentre en estado de Reposo y su frecuencia cardiaca sea inferior a 40 pulsaciones/min se considera como un síntoma de bradicardia y por ello se realiza un aviso al usuario. Cuando la frecuencia cardiaca sea superior a $220 - edad$ se considera que ha superado la frecuencia cardiaca máxima establecida por la Asociación Americana del Corazón en [35], y en [18] y [36]. Además, se monitorizan los periodos de recuperación de la frecuencia

cardiaca tras realizar una actividad física Moderada o Intensa teniendo en cuenta las consideraciones del apartado 3.4 de esta memoria. Por último, con la información histórica recogida del usuario se obtienen unos rangos de la frecuencia cardiaca *normal* del usuario para cada nivel de actividad, que se utilizará para detectar valores anómalos en la frecuencia cardiaca del usuario. Posteriormente se explica este procedimiento con más detalle, así como el funcionamiento de este sistema desde el punto inicial, donde no existe ningún dato sobre el usuario.

En el sistema de detección del nivel de actividad física se contemplaban los niveles de Reposo, Moderado e Intenso. En [37] se indica que la frecuencia cardiaca se reduce durante el sueño y por ello se decidió añadir un cuarto nivel de actividad para el Sueño. Este nivel deriva del nivel de actividad de Reposo. Cuando el usuario se encuentra en este estado durante 30 minutos de forma ininterrumpida entre las 00h y las 08h del día se considera que se encuentra en estado de Sueño. En [38] se expone la dificultad para detectar el sueño con señales como las proporcionadas por un acelerómetro y un giroscopio. En [39] se propone un sistema que tiene como objetivo realizar una diferenciación entre los estados de Reposo y Sueño, utilizando la señal del acelerómetro y la frecuencia cardiaca proporcionada por un Apple Watch. Tras utilizar diferentes clasificadores, se llega a la conclusión de que los resultados obtenidos no son satisfactorios, ya que los niveles de acierto de dichos clasificadores son bajos. Se propone utilizar un sensor que detecta la cantidad de luz ambiental, que se encuentra integrado en el Apple Watch, pero no es posible acceder a dicho sensor ya que el fabricante no lo permite. En el caso de los dispositivos utilizados en este trabajo ninguno cuenta con este sensor, por lo que su uso para mejorar la detección del sueño quedó descartado. Se valoró un sistema para utilizar el sensor de luz, que incluye el teléfono móvil utilizado para gestionar la pulsera y, de esta forma, utilizar dicho sensor tanto para el sistema implementado en la pulsera como para el sistema implementado en el reloj. La utilización para el sistema de la pulsera es más sencilla ya que es el teléfono el encargado de realizar todo el procesamiento de la información y, por lo tanto, no es necesario realizar ninguna comunicación, simplemente

hay que acceder a los datos del sensor. En el caso del reloj, el procedimiento es más complicado ya que primero hay que realizar una solicitud al teléfono para que realice la medición y posteriormente el teléfono debe enviar la información de nuevo al reloj para que este la procese. Esto implica que el reloj no puede funcionar de forma independiente del teléfono. Además, la comunicación no tiene la velocidad de transmisión necesaria y ralentiza mucho el proceso de cálculo del nivel de actividad. Otra dificultad encontrada fue que dicho sensor solamente funciona cuando la pantalla del teléfono se encuentra encendida, con lo cual, habría que forzar su encendido cada vez que se quiera obtener una medición provocando un gasto de batería considerable. Por todo ello, se decidió no utilizar la información de dicho sensor.

El objetivo del sistema no es determinar con exactitud cuando el usuario se encuentra en estado de sueño y cuando se encuentra en Reposo. Tampoco se trata de medir las horas de sueño del usuario. El objetivo es tratar de diferenciar, lo máximo posible, los datos de Sueño de los datos de Reposo para no provocar distorsiones en la frecuencia cardiaca *normal* de ambos niveles de actividad.

En todo momento se trabaja en escenarios con estados de ánimo normales, sin situaciones psicológicas adversas o de estrés, que pueden influir de forma notoria sobre la frecuencia cardiaca. El tipo de sensórica utilizada no permite discernir entre estas variaciones de índole psicológico.

El funcionamiento del sistema se divide en tres fases. En el punto inicial, el sistema no tiene ningún dato sobre el usuario. Dichas fases se irán completando secuencialmente hasta llegar a la fase 3, que se corresponde con el funcionamiento normal. A medida que se completan las fases 1 y 2 se desbloquean algunas funcionalidades de la aplicación.

La fase 1 corresponde con el punto inicial. En este instante, el usuario debe introducir su edad para poder calcular la frecuencia cardiaca máxima mediante la fórmula $220 - edad$

como indica la Asociación Americana del Corazón en [35]. Además, el usuario debe solicitar un cálculo de su frecuencia cardiaca en reposo. Para ello, el sistema cuenta con la posibilidad de estimar dicho valor recogiendo los datos de la frecuencia cardiaca del usuario durante 5 minutos. Se recomienda que esté sentado, sin realizar ningún movimiento para que los datos obtenidos sean lo más representativos dentro de las posibilidades. A continuación, se busca el valor mínimo de todas las medidas obtenidas para obtener una estimación de la frecuencia cardiaca en reposo del usuario. Este dato se utilizará en fases posteriores.

La fase 2 corresponde con el periodo de aprendizaje. Para cada nivel de actividad (reposo, moderado e intenso) se necesitan 100 mediciones que cumplan una serie de requisitos para que sean consideradas como fiables durante esta fase. En primer lugar, el intervalo de medición seleccionado debe ser de 1 o 2 minutos para que los datos sean considerados en esta fase. Cuando se obtiene una medición de frecuencia cardiaca y nivel de actividad, se verifica que dicho nivel de actividad ha permanecido invariable durante los dos minutos anteriores. El motivo de realizar esta comprobación es descartar mediciones donde el usuario ha realizado otro tipo de actividad en instantes anteriores que puedan distorsionar el periodo de aprendizaje. En [19] se realizó un estudio sobre la reducción de la frecuencia cardiaca una vez finalizada una actividad física. Se pudo comprobar que durante el primer minuto es donde se produce el mayor descenso. En el siguiente minuto desciende de forma menos pronunciada y ya durante los siguientes minutos la diminución es muy pequeña.

Además, se imponen algunas restricciones más en función del tipo de actividad:

- **Reposo**. Se exige que la frecuencia cardiaca de la medición sea igual o superior a la frecuencia cardiaca de reposo calculada en la fase 1, e inferior a 100 pulsaciones por minuto, ya que numerosos organismos como la Asociación Americana del Corazón [35] consideran dicho valor como el máximo para que sea considerado un valor normal.

- **Sueño**. Se exige que las mediciones deben ser iguales o superiores a 40 pulsaciones por minuto, ya que como se indica en [37] la frecuencia cardiaca se reduce durante el sueño. Además, las mediciones deben ser inferiores a 100 pulsaciones por minuto.
- **Moderada**. Se exige que las mediciones de la frecuencia cardiaca sean superiores al 50% de la frecuencia cardiaca máxima calculada en la fase 1 a partir de la edad, e inferiores al 70% de dicho valor. Este rango es el recomendado por la Asociación Americana del Corazón [35] para las actividades moderadas.
- **Intensa**. Las mediciones de frecuencia cardiaca deben ser superiores al 70% de la frecuencia cardiaca máxima, e inferiores al 90% de dicho valor. Al igual que en el caso de las actividades de tipo moderado, este rango es el recomendado por la Asociación Americana del Corazón [35].

Las mediciones que cumplan con todas las restricciones se consideran como mediciones fiables que se utilizarán en el periodo de aprendizaje. Cuando existen más de 100 muestras válidas para una determinada actividad, se considerará que ha finalizado el periodo de aprendizaje para dicha actividad y podrá pasar a la siguiente fase. Los diferentes niveles de actividad son independientes entre sí desde el punto de vista del aprendizaje, de forma que un nivel de actividad se puede encontrar en fase de aprendizaje y otro nivel se puede encontrar en la siguiente fase.

La fase 3 corresponde con el funcionamiento normal de la aplicación. La actividad que ya haya completado el periodo de aprendizaje tendrá desbloqueadas todas las funcionalidades. Los datos obtenidos durante esta fase se utilizarán para seguir ajustando los parámetros del sistema. Al igual que en la anterior fase, el nivel de actividad debe permanecer invariable durante los dos minutos anteriores para que la medición no esté distorsionada y se pueda considerar como fiable. Con todos los datos obtenidos se calcula, para cada actividad, un rango de la frecuencia cardiaca donde se considera que no existe ninguna anomalía. Para construir el rango de cada actividad se valoraron dos posibilidades. La primera consiste en calcular el rango intercuartílico para los datos de cada actividad y

calcular el rango de los valores de la frecuencia cardiaca considerados como normales usando la ecuación (4.23), donde RIQ es el rango intercuartílico.

$$[\text{media} - 1{,}5\mathrm{RIQ}, \qquad \text{media} + 1{,}5\mathrm{RIQ}] \tag{4.23}$$

Además, los nuevos valores recogidos que se encuentren fuera del rango se descartan y no se tienen en cuenta para actualizar el rango. Se realizó un experimento tanto con la pulsera como con el reloj. La Tabla 4.26 sintetiza los datos recogidos en el archivo comprimido "datos.tgz" de la URL *pirserver.edv.uniovi.es/TFG_AlbertoV/*.

Nivel Actividad	Extremo Inferior	Extremo Superior
Sueño	48,06	54,06
Reposo	33,42	99,41
Moderado	53,83	137,83

Tabla 4.25: Rangos obtenidos para la pulsera utilizando (4.23), descartando valores fuera de rango.

Nivel Actividad	Extremo Inferior	Extremo Superior
Sueño	55,21	64,21
Reposo	51,25	69,25
Moderado	81,28	126,28

Tabla 4.26: Rangos obtenidos para el reloj utilizando (4.23), descartando valores fuera de rango.

La segunda posibilidad valorada consistió en no descartar los nuevos valores que se encuentren fuera del rango previamente calculado. En este nuevo escenario se contempló utilizar el rango calculado como se indica en la ecuación (4.23). Además, se incorpora una nueva fórmula que utiliza la desviación típica, cuyo uso se recomienda en [40] y [41] para tratar valores anómalos de frecuencia cardiaca. En este caso el rango se construye como se indica en la ecuación (4.24), donde $Desv$ se corresponde con el valor de la desviación típica de los datos.

$$[\text{media} - 1{,}5\mathrm{Desv}, \qquad \text{media} + 1{,}5\mathrm{Desv}] \tag{4.24}$$

Tras realizar pruebas con diferentes factores que multiplican a la desviación típica se determinó que 1,5 es el valor que mejor ajusta en este caso. Los resultados obtenidos en este caso son los que se muestran en las tablas Tabla 4.27 y Tabla 4.28.

Nivel Actividad	Ext. Inf (4.23)	Ext. Sup (4.23)	Ext. Inf (4.24)	Ext. Sup (4.24)
Sueño	41,75	62,75	35,81	68,69
Reposo	35,70	98,70	45,13	89,26
Moderado	47,06	140,06	70,11	117,02

Tabla 4.27: Rangos obtenidos para la pulsera utilizando (4.23) y (4.24).

Nivel Actividad	Ext. Inf (4.23)	Ext. Sup (4.23)	Ext. Inf (4.24)	Ext. Sup (4.24)
Sueño	54,26	66,26	42,21	78,31
Reposo	37,99	94,99	40,89	92,09
Moderado	83,45	122,45	82,71	123,18

Tabla 4.28: Rangos obtenidos para el reloj utilizando (4.23) y (4.24).

Una vez estudiados los datos recogidos, se optó por utilizar la ecuación (4.24) para construir los rangos de valores normales de la frecuencia cardiaca de los diferentes niveles de actividad. Además, se decidió no descartar ningún valor para realizar el cálculo. Los motivos para elegir esta alternativa son varios. En primer lugar, aunque ambas fórmulas son válidas para detectar valores atípicos, (4.24) ha obtenido resultados satisfactorios en aplicaciones relacionadas con la frecuencia cardiaca (ver [40] y [41]). En segundo lugar los resultados obtenidos con (4.24) generan unos rangos más cercanos a las recomendaciones de la Asociación Americana del Corazón [35]. Utilizando (4.23) se generan algunos rangos donde la diferencia entre sus extremos es muy grande, como es el caso del nivel de actividad Moderado en el caso de la pulsera.

Los valores de frecuencia cardiaca que se encuentren fuera del rango serán considerados como una anomalía. Para las actividades de sueño y reposo, cuando el usuario se encuentre durante 10 minutos de forma consecutiva fuera de rango, se le notificará dicho suceso. Para la actividad de tipo moderado, se notificará cuando se encuentre fuera de rango durante 6 minutos de forma consecutiva. Por último, para la actividad de tipo intenso, se notificará cuando se encuentre fuera de rango durante 4 minutos. Apple, en sus dispositivos Apple Watch, utiliza un periodo de tiempo de 10 minutos para notificar una anomalía en un estado de reposo (ver [42]). Debido a que las actividades de tipo moderado e intenso tienen una duración menor, se decidió utilizar un periodo de tiempo más corto, especialmente para la actividad de tipo Intenso.

Fase 1	▪ Cálculo de FC máxima a partir de la edad ▪ Cálculo de FC reposo
Fase 2	▪ Obtención de 100 datos para cada actividad ▪ Misma actividad durante los 2 minutos anteriores ▪ Intervalos de medición permitidos: 1min y 2min ▪ Restricciones extra según nivel de actividad: ▪ Reposo: FC en el rango $[FC\ reposo, 100]$ ▪ Sueño: FC en el rango $[40, 100]$ ▪ Moderado: FC en el rango $[0.5 * FCmax, 0.7 *$FCmax$]$ ▪ Intenso: FC en el rango $[0.7 * FCmax, 0.9 * FCmax]$
Fase 3	▪ Cálculo de rango de FC normal para cada actividad con (4.24) ▪ Actualización de cada rango con datos obtenidos durante esta fase ▪ Notificación si FC se sale de rango durante 10 min en Sueño y Reposo ▪ Notificación si FC se sale de rango durante 6 min en Moderado ▪ Notificación si FC se sale de rango durante 4 min en Intenso ▪ Notificación si la recuperación de FC es inferior a 12 pulsaciones/min 1 minuto después de reducir el nivel de actividad ▪ Notificación si la recuperación de FC es inferior a 22 pulsaciones/min 2 minutos después de reducir el nivel de actividad ▪ Notificación si FC es inferior a 40 pulsaciones/min ▪ Notificación si FC es superior a $(220 - edad)$ pulsaciones/min

Tabla 4.29. Resumen del sistema de monitorización.

Otra de las funcionalidades implementadas es el estudio de la recuperación de la frecuencia cardiaca una vez el usuario ha terminado de realizar una actividad correspondiente a un nivel superior, ya sea moderado o intenso. Como se indica en [18], [20] y [21], es un indicador para detectar posibles enfermedades cardiacas. Si en el minuto posterior el usuario no ha reducido su frecuencia cardiaca en más de 12 pulsaciones por minuto, será notificado como una anomalía. De igual forma, si no ha recuperado más de 22 pulsaciones por minuto en los dos minutos siguientes a la finalización del nivel de actividad moderado o intenso, se notificará el suceso al usuario.

Además, se controla que la frecuencia cardiaca no disminuya por debajo de 40 pulsaciones por minuto, ya que se considera como un valor anómalo que puede tener consecuencias sobre la salud. De igual forma, se controla que no se supere un valor de frecuencia cardiaca máxima considerado en [18], [35] y [36] para una persona sana. Dicho

valor se calcula como $220 - edad$. En ambos casos también se notificará al usuario la anomalía.

La Tabla 4.29 resume las distintas fases del sistema de monitorización, así como las funcionalidades de cada una de ellas.

5. Conclusiones y trabajos futuros

Cuando se concibió este Trabajo Final de Grado (TFG) se marcó el objetivo de

"*Diseñar e implementar un prototipo que, combinando señales heterogéneas, permita la monitorización continua de ciertos parámetros de los Sistemas Cardiovascular y Respiratorio*"

Sin embargo, las limitaciones derivadas de la irrupción del SARS-CoV-2, y del *Estado de Alarma* decretado en España, obligaron a replantear el objetivo inicial, resumiéndose el nuevo enfoque del TFG en:

- Realizar trabajos para monitorizar la actividad y ciertos parámetros del Sistema Cardiovascular con llevables periféricos. Una mínima labor de campo.
- Diseñar un prototipo central para la captura de audio. Comprobar que cumple con los requisitos para su futura integración en un sistema global.
- Implementar algoritmos eficientes para procesar, detectar y separar en tiempo real señales acústicas provenientes del corazón y de los pulmones.

En definitiva, abordar el trabajo planteado inicialmente, pero sin integrar los sistemas y sin profundizar en la validación de los modelos fruto de las investigaciones realizadas.

En lo relativo al diseño y construcción del prototipo hardware para la captura de audio, se realizaron pruebas capturando el sonido reproducido por un equipo musical. Aunque los resultados fueron satisfactorios, no se pudo avanzar más por las restricciones de movilidad inherente al estado de alarma.

La vertiente software del tratamiento de señales sonoras procedentes del corazón y del pulmón se culminó con éxito, obteniéndose resultados muy prometedores. El sistema diseñado puede ejecutarse tanto en dispositivos móviles (p. ej. smartphones o tabletas) como en sistemas clásicos (ordenadores personales, servidores, etc.). Incluso podría usarse

en otro tipo de SoC (*System on Chip*) menos potentes, como pulseras de actividad o relojes inteligentes, siempre que sus recursos y prestaciones sean razonables (en el reloj inteligente Ticwatch S2 utilizado en este TFG no ha sido posible obtener resultados satisfactorios).

Se ha demostrado como el uso de la HPC (Computación de Altas Prestaciones, *High Performance Computing*) y de la PC (Computación Paralela, *Parallel Computing*) permite cumplir con los requisitos de tiempo real, incluso en dispositivos antiguos y poco potentes (p. ej. el Xiaomi Redmi Note 3 Pro usado en la experimentación). Más aún, se ha demostrado que es posible ser conservador en el drenaje de batería y procesar razonablemente rápido (la relación FLOP/Vatio de los algoritmos es buena).

También se diseñó e implementó un Sistema de Monitorización Continua de parámetros relacionados con la salud (SMCs), utilizando tanto los sensores ópticos como los sensores de posición que integran los wearables. El sistema integra la detección de la actividad física, que se estima a partir de los valores obtenidos por el acelerómetro del llevable. Se han usado técnicas de Inteligencia Artificial (IA) para obtener clasificadores eficaces de la actividad física, seleccionado aquel que, además, es de bajo coste computacional (poco consumo energético).

Para para la obtención de los clasificadores del SMCs se llevó a cabo una labor de campo, obteniendo un conjunto de datos que, con posterioridad, fueron procesados, recurriendo, en parte, a técnicas estadísticas.

El software creado para el SMCs puede ejecutarse en dispositivos móviles con sistemas operativos Android y Wear OS, además de que todo el ecosistema SoC compatible con BluetoothLE puede ser integrado, usado o consultado.

En resumen, se han alcanzado todos los hitos plateados, pero también han quedado abiertos algunos otros. Por ejemplo, probar otros SoC con Wear OS, mejorar la labor de campo y verificar que los algoritmos de IA siguen siendo adecuados, completar el prototipo hardware e integrar todos los sistemas, probar a capturar audio en tiempo real durante la actividad cotidiana, añadir sistemas de toma de decisión para combinar la información de las señales periféricas y centrales, etc.

Y también se han abierto trabajos futuros. Por ejemplo, cualquier anomalía cardio-respiratoria *audible* es susceptible de ser abordada, toda vez que se tiene un sistema en tiempo real que separa, con gran calidad, los sonidos del corazón, y de los pulmones, del resto. Desde las más sencillas, por ejemplo, detección de sibilancias, hasta las más complejas o actuales, detectar la COVID-19 usando modelos de IA.

Bibliografía

[1] A. Kamisalic, I. Fister, M. Turkanovic and S. Karakatic, “Sensors and Functionalities of Non-Invasive Wrist-Wearable Devices: A Review,” *Sensors,* vol. 18, no. 6, p. 1714, 2018.

[2] Sociedad Española de Cardiología, «Insuficiencia cardiaca, la enfermedad cardiovascular que no consigue disminuir la mortalidad,» *Revista Española de Cardiología,* 2019.

[3] Organización Mundial de la Salud, «Enfermedades cardiovasculares,» *Centro de Prensa,* 2017.

[4] U.S. National Library of Medicine, “Heart palpitations,” *MedlinePlus: a service of NLM,* 2018.

[5] L. O. Figura and A. A. Teixeira, Food Physics: Physical Properties - Measurement and Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[6] Texas Heart Institute, "Heart Anatomy," [Online]. Available: www.texasheart.org/heart-health/heart-information-center/topics/heart-anatomy/. [Accessed 30 6 2020].

[7] U.S. National Library of Medicine, “Breath sounds,” *MedlinePlus: a service of NLM,* 2019.

[8] W. H. Spriggs, Essentials of Polysomnography, Jones & Bartlett Learning, 2014.

[9] Google, “Wear OS by Google,” [Online]. Available: https://wearos.google.com/#find-your-watch. [Accessed 30 6 2020].

[10] Organización Mundial de la Salud, «What is Moderate-intensity and Vigorous-intensity Physical Activity?,» [En línea]. Available: www.who.int/dietphysicalactivity/physical_activity_intensity/en/. [Último acceso: 30 6 2020].

[11] F. Miao, Y. He, J. Liu, Y. Li and I. Ayoola, “Identifying typical physical activity on smartphone with varying positions and orientations,” *Biomedical engineering online,* vol. 14, no. 32, 2015.

[12] Chenxi Song, “Master Thesis: User Activity Tracker Using Android Sensor,” Ohio State University, 2015.

[13] M. Shoaib, H. Scholten and P. Havinga, "Towards Physical Activity Recognition Using Smartphone Sensors," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Italy, 2013.

[14] J. Dennis, “Master Thesis: Human Activity Pattern Recognition from Accelerometry Data,” Universidad de Heidelberg, Cologne, 2013.

[15] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning,* vol. 20, p. 273–297, 1995.

[16] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, Classification and Regression Trees, Taylor and Francis, 1984.

[17] I. Katsuji, S. Hideyuki, H. Masatsugu, K. Hideo, O. Hitoshi, Y. Hiroshi, T. Hiroshi, I. Michitoshi and K. Takenobu, "Vagally mediated heart rate recovery after exercise is accelerated in athletes but blunted in patients with chronic heart failure," *Journal of the American College of Cardiology,* vol. 24, no. 6, pp. 1529-1535, 1994.

[18] C. R. Cole, E. H. Blackstone, F. J. Pashkow, C. E. Snader and M. S. Lauer, "Heart-Rate Recovery Immediately after Exercise as a Predictor of Mortality," *New England Journal of Medicine,* vol. 341, no. 18, pp. 1351-1357, 1999.

[19] R. I. Carter, D. E. Watenpaugh, W. L. Wasmund, S. L. Wasmund and M. L. Smith, "Muscle pump and central command during recovery from exercise in humans," *Journal of Applied Physiology,* vol. 87, no. 4, pp. 1463-1469, 1999.

[20] A. Elshazly, H. Khorshid, H. Hanna and A. Ali, "Effect of exercise training on heart rate recovery in patients post anterior myocardial infarction," *Egypt Heart J,* vol. 70, no. 4, pp. 283-285, 2018.

[21] J. v. d. V. Yordi, v. d. H. Pim and W. Niek, "Heart Rate Recovery 10 Seconds After Cessation of Exercise Predicts Death," *Journal of the American Heart Association,* vol. 7, no. 8, p. e008341, 2018.

[22] S. M. López Silva, M. L. Dotor, J. P. Silveira, R. Giannetti y L. Herrera, «Fotopletismografía por reflexión con LEDs infrarrojos para evaluar órganos y tejidos intra-abdominales: estudio inicial en cerdos,» *Opt. Pura Apl.,* vol. 42, nº 1, pp. 23-32, 2009.

[23] B. Bent, B. A. Goldstein, W. A. Kibbe and J. P. Dunn, "Investigating sources of inaccuracy in wearable optical heart rate sensors," *npj Digital Medicine (A Nature Partner Journal),* vol. 3, no. 1, p. 18, 2020.

[24] F. Canadas-Quesada, N. Ruiz-Reyes, J. Carabias-Orti, P. Vera-Candeas y J. Fuertes-Garcia, «A non-negative matrix factorization approach based on spectro- temporal clustering to extract heart sounds,» *Applied Acoustics,* vol. 125, pp. 7-19, 2017.

[25] E. Battenberg, A. Freed y D. Wessel, «Advances in the parallelization of music and audio applications,» *Proceedings of the 2010 International Computer Music Conference (ICMC),* 2010.

[26] K. Davajaran, «Nonnegative matrix factorization: an analytical and interpretative tool in computational biology,» *PLoS Computational Biology ,* vol. 4, nº 7, pp. 1-12, 2008.

[27] J. Wang, W. Zhong y J. Zhang, «NNMF-based factorization techniques for high-accuracy privacy protection on non-negative-valued datasets.,» *Proceedings of the Sixth IEEE International Conference on Computing and Processing, Data Mining Workshops (ICDM Workshops). ,* 2006.

[28] W. Xu, X. Liu y Y. Gong, « Document clustering based on non-negative matrix factorization.,» *Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval.,* 2003.

[29] D. D. Lee y H. Seung, «Algorithms for Non-negative Matrix Factorization. Advances in neural information processing systems.,» *MIT Press,* 2001.

[30] M. Minami y S. Eguchi, «Robust blind source separation by beta-divergence.,» *Neural Computing,* vol. 14, pp. 1859-1886, 2002.

[31] C. Févotte, N. Bertin y J. Durrieu, «Nonnegative matrix factorization with the Itakura-Saito divergence: with application to music analysis.,» *Neural Computing,* vol. 21, pp. 793-830, 2009.

[32] D. Pancorbo Rubio, «Separación y búsqueda de patrones repetitivos procedentes del corazón humano en señales cardiopulmonares,» Jaén, 2016.

[33] F. Scholkmann, J. Boss and M. Wolf, "An Efficient Algorithm for Automatic Peak Detection in Noisy Periodic and Quasi-Periodic Signals," *Algorithms,* vol. 5, pp. 588-603, 2012.

[34] M. Matsumoto y T. Nishimura, «Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator,» *ACM Transactions on Modeling and Computer Simulation,* vol. 8, nº 1, pp. 3-30, 1998.

[35] A. H. Association. [En línea]. Available: https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates. [Último acceso: 20 06 2020].

[36] M. Javorka, I. Zila, T. Balharek y K. Javorka, «Heart rate recovery after exercise: relations to heart rate variability and complexity,» *Brazilian Journal of Medical and Biological Research,* 2020.

[37] T. Penzel, J. W. Kantelhardt, C.-C. Lo, K. Voigt y C. Vogelmeier, «Dynamics of Heart Rate and Sleep Stages in Normals and Patients with Sleep Apnea,» *Neuropsychopharmacology,* vol. 28, 2003.

[38] W. Karlen, «Adaptative Wake and Sleep Detection for Wearables Systemas,» 2009.

[39] O. Walch, Y. Huang, D. Forger y C. Goldstein, «Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device,» vol. 42, nº 12, 2019.

[40] K. J Kemper, C. Hamilton y M. Atkinson, «Heart Rate Variability: Impact of Differences in Outlier Identification and Management Strategies on Common Measures in Three Clinical Populations.,» *Pediatric Research,* vol. 62, pp. 337-342, 2007.

[41] F. Forcolin, R. Buendia, S. Candefjord, J. Karlsson, A. Sjoqvist y A. Anund, «Comparison of Outlier Heartbeat Identification and Spectral Transformation Strategies for Deriving Heart Rate Variability Indices for Drivers at Different Stages of Sleepiness,» *Traffic Inj Prev.,* vol. 19, pp. 112-119, 2018.

[42] Apple, «Apple,» [En línea]. Available: https://support.apple.com/es-es/HT208931. [Último acceso: 20 06 2020].

[43] S. Vigna, «It Is High Time We Let Go Of The Mersenne Twister,» 2019.

[44] A. Nikishaev, «Medium,» [En línea]. Available: https://medium.com/machine-learning-world/how-i-hacked-xiaomi-miband-2-to-control-it-from-linux-a5bd2f36d3ad. [Último acceso: 20 06 2020].

[45] «Bluetooth,» [En línea]. Available: https://www.bluetooth.com/xml-viewer/?src=https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Characteristics/org.bluetooth.characteristic.heart_rate_measurement.xml#tree0:0,6,0,3,0,3,0%7C0,6,0,3,0,3,1%7C0,6,0,3,1,3,0%7C0,6,0,3,1,. [Último acceso: 20 06 2020].

[46] M. Chmielewski, «WithIntent,» [En línea]. Available: https://withintent.com/blog/introduction-to-bluetooth-le-on-ios-mi-band-2-case-study/. [Último acceso: 20 06 2020].

[47] P. «GitHub,» [En línea]. Available: https://github.com/PhilJay/MPAndroidChart. [Último acceso: 20 06 2020].