Smart Footwear Feedback System: sistema electrónico para realimentación y monitoreo de movimientos en extremidades inferiores

Edisson Javier Ariza Ariza
Cristian Camilo Bohórquez Buitrago

Trabajo de grado para obtener el título de
Licenciados en Electrónica

Director
David Peña Morales

Universidad Pedagógica Nacional
Facultad de Ciencia y Tecnología
Licenciatura en Electrónica
Bogotá
2021

Agradecimientos

Cristian Camilo Bohórquez Buitrago

Un agradecimiento especial al profesor David Peña por su constante apoyo para que este proyecto fuera desarrollado con la calidad que conseguimos finalizarlo, a mis padres y hermanos que me respaldaron en todo momento y a mi compañero Edisson que fue parte fundamental para la conclusión de este trabajo. Agradezco también a la UPN y a UNIVATES que me permitieron desarrollarme académicamente y me dieron tantas oportunidades y alegrías.

Edisson Javier Ariza Ariza

Agradezco a todos los profesores de la Licenciatura en Electrónica por guiar mi camino profesional, en especial al profesor David Peña por sus orientaciones y compromiso en el desarrollo de este proyecto; a mi familia, especialmente a mi mamá Amanda Ariza quien me acompañó y apoyó todo este tiempo; a mi compañero Cristian por todo el esfuerzo y constancia que dedicó para culminar este trabajo.

Tabla de contenido

Tabla de ilustraciones

Tabla de tablas

Resumen

El presente trabajo propone el desarrollo de un sistema electrónico portátil para la realimentación y el monitoreo de movimientos en extremidades inferiores (SFFS: Smart Footwear Feedback System) basado en tecnologías que se puedan vestir y una interfaz humano-computadora. Se diseñó una plantilla para calzado con sensores resistivos de presión y una unidad de medición inercial, el cual se encarga de medir la velocidad y orientación, usando una combinación de acelerómetros y giróscopos. Los sensores se colocaron en posiciones ampliamente estudiadas por diferentes grupos de investigación para la vigilancia de la actividad y la marcha del usuario. SFFS utiliza la comunicación inalámbrica Bluetooth® de la placa de desarrollo ESP32 y una aplicación móvil diseñada e implementada para el sistema operativo Android® en donde el usuario recibe realimentación visual y auditiva de su actividad mientras usa los zapatos. Se realizaron diversas pruebas para validar la efectividad de la realimentación y el correcto funcionamiento del sistema, proponiendo diferentes mecanismos factibles de aplicar en procesos de sonificación interactiva, además de ser ampliamente flexible y adaptable según se requiera.

Abstract

The project proposes the development of a portable electronic system for feedback and monitoring of movements in lower extremities (SFFS: Smart Footwear Feedback System) based on wearable technologies and a human-computer interfaces. A footwear insole was designed with resistive pressure sensors and an inertial measurement unit, which is responsible for measuring speed and orientation of the foot, using a combination of accelerometers and gyroscopes. Sensors were placed in positions widely studied by different research groups around the world for monitoring the user's activity and gait. SFFS uses the Bluetooth® wireless communication protocol of the ESP32 development board, and a mobile application designed and implemented for Android® where the users receive visual and auditory feedback of their activity while wearing the shoes. Several tests were performed to validate the effectiveness of the feedback and the correct system performance, proposing different mechanisms feasible to apply in interactive sonification of movement, in addition to being highly flexible and adaptable as required.

1. Introducción y aspectos generales

1.1 Planteamiento del problema

Investigaciones recientes como las realizadas por Eskofier et al. (2017), Ohnishi, Terada y Tsukamoto (2018) y Saidani et al. (2019), resaltan la importancia de desarrollar tecnologías que se puedan vestir (WT: *Wearable Technology* por sus siglas en inglés) basadas en el internet de las cosas (IoT: *Internet of Things* por sus siglas en inglés) que ayude al usuario a monitorear su actividad diaria, con la intención de favorecer un estilo de vida saludable. Una aplicación pertinente en este contexto es la vigilancia continua de la actividad física y su realimentación en el menor tiempo posible, usando dispositivos como zapatos inteligentes que puedan captar señales relevantes, de forma automática, relacionadas con las condiciones físicas y de salud de los usuarios. Los resultados de los estudios de WT orientadas a la promoción de la vida sana proponen que aún hay mucho camino por recorrer en este tema y además sugieren algunos parámetros para tener en cuenta en trabajos futuros en este tipo de tecnologías emergentes. Por lo anterior, la presente investigación propuso el desarrollo de un sistema electrónico de realimentación auditiva llamado: SFFS (*Smart Footwear Feedback System*) que utiliza los datos recolectados por diferentes sensores integrados en el calzado y entrega a los usuarios, de forma inalámbrica, información relevante a través de una aplicación móvil (APP: *application* por sus siglas en inglés) para teléfonos inteligentes. SFFS propende por la promoción de hábitos saludables como la actividad física, adicionalmente es posible su implementación en procesos de rehabilitación física o como apoyo para la detección temprana de enfermedades (Eskofier et al. 2017).

1.2 Pregunta de investigación

¿Cómo desarrollar un dispositivo de tecnología que se pueda vestir mediante la integración de un sistema electrónico al calzado y una aplicación para teléfonos inteligentes, que permita

realizar una realimentación auditiva y visual a partir del monitoreo constante de los movimientos en extremidades inferiores?

1.3 A lJustificación

En la última década, industrias y académicos dirigieron su atención a diferentes dispositivos que se puedan vestir también conocidos como *Werable Devices* (WD) o *Werable Techology* (WT), como el reloj inteligente (Smart Watch), la pulsera (Smart Band) o los zapatos; pues, el creciente desarrollo de sistemas informáticos de bajo consumo energético, comunicados de manera inalámbrica, discretos y socialmente aceptables que se puedan llevar puestos, se ha convertido en un tema de investigación cada vez más importante (Hegde, Bries y Sazonov 2016). Esto obedece al enfoque emergente de tecnologías para la internet de las cosas de la salud (IoHT: *Internet of Healt Things* por sus siglas en inglés). Pasluosta et al. (2015) e Islam et al. (2015) proponen el cuidado de la salud como una aplicación de las WT que captan información de la actividad diaria del usuario. Con esto se busca apoyar potencialmente la vida sana de las personas, cuando se realimentan señales de variables físicas captadas al realizar actividades que involucren las extremidades inferiores con sensores integrados en el calzado; por ejemplo, Tajadura-Jiménez et al. (2015) expone la posibilidad de modificar la percepción del peso corporal al realimentar auditivamente al usuario con el sonido que produce al caminar. De este modo, se estimula a las personas para seguir con esta actividad por más tiempo, influyendo positivamente en su salud.

Los resultados obtenidos en los diversos proyectos desarrollados en estas temáticas sugieren la posibilidad de realizar desarrollos tecnológicos relacionados con neurociencia y rehabilitación física. En ese mismo camino, la investigación desarrollada por Wang et al. (2015) demostró que estudiando el patrón de la marcha de las personas se consigue prevenir caídas y detectar patrones anormales al caminar. Sin embargo, también sugieren que es necesario sacar del

laboratorio estos exámenes y permitirle al usuario conocer en tiempo real[1] esta información para que la utilice en su día a día. Lo anterior es posible dado que en la actualidad hay una tendencia de las personas a utilizar dispositivos móviles y, además, las industrias dirigieron su atención en la fabricación de dispositivos inteligentes que hacen parte del vestuario (Kim, Lee y Hong 2018). Así pues, los sistemas de realimentación con zapatos inteligentes se están convirtiendo en el presente y futuro de IoHT para favorecer la vida sana (Eskofier et al. 2017). Un conjunto de sensores inerciales o para la detección de la presión (relacionado con la fuerza aplicada en una superficie) pueden ser usados para el seguimiento de la postura y el reconocimiento de la actividad física (Hegde, Bries y Sazonov 2016) o para apoyar la prevención de lesiones, el trabajo diagnóstico de enfermedades, la toma de decisiones terapéuticas o el control individual de recuperación. Lo anterior sustenta la relevancia de diseñar y producir sistemas de realimentación de datos en tiempo real que le permitan al usuario monitorear diferentes variables físicas involucradas en el desarrollo de actividades o movimientos de las extremidades inferiores y con esto, contribuir a mejorar su estado de salud.

1.4 Objetivos

1.4.1 Objetivo general.

Desarrollar un sistema electrónico para la realimentación auditiva y monitoreo de variables de presión, fuerza e inercia involucradas en movimientos de extremidades inferiores.

1.4.2 Objetivos específicos.

- Construir un prototipo de dispositivo adaptado al calzado convencional para la adquisición de datos que integre un conjunto de sensores inerciales y de presión.

[1] El concepto de tiempo real utilizado en este proyecto se explica en la sección 2.3

- Implementar una interfaz de usuario a partir de una conexión inalámbrica entre el sistema electrónico en el calzado y una aplicación móvil para el tratamiento y trasmisión de la información procesada.
- Analizar y parametrizar los datos obtenidos para definir el tipo de respuesta que será realimentada al usuario.

1.5 Antecedentes

Diversos proyectos de investigación se han desarrollado en este campo, por ejemplo, para el análisis de la marcha longitudinal usando plantillas equipadas con sensores de presión y seguimiento del movimiento, Wang et al. (2015) demostraron que al usar un sistema inalámbrico se aumenta la distancia de transmisión de los datos obtenidos de los sensores y se alarga la duración de la batería. Hegde, Bries y Sazonov (2016) y de Fazio et al. (2021), basan su investigación en varios estudios que usan platillas y zapatos equipados con sensores inerciales (acelerómetro y giroscopio) y de fuerza (sensores de presión) para el análisis de la marcha.

Tajadura-Jiménez et al. (2015) desarrollaron un sistema compuesto por zapatos con sensores y auriculares para la realimentación auditiva, demostrando así, el cambio en el estado de ánimo y la percepción del peso corporal al permitirle a los usuarios escuchar el sonido que producían al caminar. Esta investigación abrió la posibilidad de desarrollar avances relacionados con neurociencia al utilizar sonidos producidos por el movimiento, se sugiere diseñar tecnologías para cambiar la forma en que se percibe el cuerpo y fortalecer desarrollos de interacción humano-computadora (HCI: *Human Computer Interface* por sus siglas en inglés).

En la propuesta de Carbonaro, Lorussi y Tognetti (2016) se analizaron las características técnicas de los zapatos comerciales FootMoov, producidos por la empresa italiana CARLOS SRL (Fucecchio, Firenze, Italia, 2021) para mostrar la posibilidad que existe de utilizar zapatos

inteligentes de bajo costo como herramienta para aplicaciones biomédicas y electrónicas en la salud. El trabajo se enfocó en el análisis de la marcha y, si bien se reconoce el uso del sistema en personas sanas para que mantengan un estado óptimo de salud, proponen la posibilidad de utilizar esta tecnología para contribuir en la recuperación de personas que padezcan una patología física o neurológica en ambientes menos controlados que permitan analizar condiciones diferentes a las estudiadas en laboratorio.

Para Saidani et al. (2019) el monitoreo en tiempo real de pacientes usando sistemas de detección con WT que no afecten la calidad de vida de los pacientes, permitirá a los médicos tener información actualizada del estado de salud de estos. Una gran contribución de esta tecnología se encuentra en la rápida detección de enfermedades; por esta razón, para los autores fue importante hacer una comparación de los sistemas de detección plantar usados en la actualidad y mostrar las cualidades y debilidades de cada uno de ellos; este es un trabajo importante si se desea desarrollar un sistema de detección con sensores, pues entrega un panorama acerca de dónde, cómo y cuales sensores se recomienda usar.

En Colombia y a la fecha de elaboración del presente proyecto de grado, la implementación de sistemas de realimentación o sonificación[2] interactiva, utilizando el calzado como una WT, que permita recopilar datos de las personas asociados a sus hábitos al caminar o características de postura o marcha cuando realiza actividadfísica y además entregue realimentación visual y auditiva constante de forma inalámbrica, es un tema que no se ha explorado. Permitiendo que SFFS pueda fungir como una apertura e invitación a la comunidad académica para desarrollar proyectos aplicados a situaciones reales de salud que puede afectar la calidad de vida de los colombianos.

[2] El concepto de sonificación utilizado en este proyecto se explica en la sección 3.5 referente a la realimentación auditiva.

2. Marco teórico

En el siguiente apartado se dará a conocer la base teórica y conceptual, seleccionada académica y técnicamente, para dar viabilidad a este proyecto.

2.1 Fuerza e inercia

Con relación al movimiento de los cuerpos, Newton percibió que, si todas las fuerzas aplicadas en cierto cuerpo se anulan mutuamente, este sólo puede ser encontrado en dos estados: reposo o movimiento (Gardelli, 1999). Por lo anterior, cada vez que una persona da un paso los músculos empujan el suelo hacia atrás y este empuja su cuerpo hacia adelante, haciéndole cambiar su posición en el tiempo. Esta acción constante es lo que le permite avanzar. Ahora bien, en el momento que la persona deja de empujar hacia abajo su movimiento hacia el frente se detiene y queda en estado de reposo del que sólo puede salir cuando sus músculos vuelvan a empujar el suelo. A este empuje se le denomina fuerza (N) en Newton como unidad de medida para el Sistema Internacional y a la capacidad que tiene el suelo de resistirla se le llama inercia. En este sistema, las medidas inerciales se tomarán para determinar la inclinación (en grados o radianes) y aceleración (en m/s^2) de los pies a cada paso.

2.2 Presión

La presión es una magnitud física escalar definida como la razón entre la fuerza (N) y el área (m^2) sobre el cual la referida fuerza es aplicada perpendicularmente (Saraiva, 2014), la ecuación 1 muestra esta relación donde P es presión, F es fuerza y A es área. Existe una presión ejercida por el pie con respecto a la superficie del piso cuando las personas caminan, esta magnitud puede ser medida usando dispositivos que captan sus variaciones y las transforman en señales eléctricas.

$$P = \frac{F}{A} \tag{1}$$

2.3 Sistema de tiempo real

Young (1982) establece que un sistema de tiempo real (RTS: *real-time system* por sus siglas en inglés) es: "[…] Cualquier actividad o sistema de proceso de información que tiene que responder a un estímulo de entrada generado externamente en un periodo finito y especificado". Se observa entonces que el tiempo de respuesta de salida con respecto a la entrada es determinante. Sobre esto el *Oxford Dictionary of Computing* define un RTS como:

> *"Cualquier sistema en el que el tiempo en el que se produce la salida es significativo. Esto generalmente es porque la entrada corresponde a algún movimiento en el mundo físico, y la salida está relacionada con dicho movimiento. El intervalo entre el tiempo de entrada y el de salida debe ser lo suficientemente pequeño para una temporalidad aceptable*"

En este punto es importante señalar que los RTS pueden ser estrictos o no estrictos. Burns y Wellings (2003) explican que "los sistemas de tiempo real no estrictos son aquéllos en los que los tiempos de respuesta son importantes pero el sistema seguirá funcionando correctamente, aunque los tiempos límite no se cumplan ocasionalmente". Para este proyecto se tendrá en cuenta la definición de los autores arriba citados y será usado un RTS no estricto, dado que el SFFS busca tomar muestras de un conjunto de sensores de entrada a intervalos regulares, pero puede tolerar retrasos intermitentes cortos.

2.4 Sensores

Se pueden medir de diversas formas las magnitudes físicas involucradas en la acción de caminar para obtener información útil de este movimiento; gracias al desarrollo de la ciencia y la tecnología es posible hacerlo con mucha exactitud ya que en la actualidad existen para esto gran cantidad de sensores. Según Wengling (2010), sensor es un término usado para designar aquellos dispositivos que son sensibles a alguna forma de energía del ambiente y relacionan estas

informaciones con una magnitud física, para entregar otra señal a la salida susceptible de ser almacenada y procesada.

2.4.1 Sensores de presión.

Un sensor de presión es un dispositivo que entrega señales en función de la fuerza a la que es sometido, para el caso de este proyecto se usaron ocho sensores que ofrecen como salida una señal de tipo eléctrico. En la Ilustración 1 se muestra la representación de un pie que presiona de forma perpendicular la superficie de una resistencia (FSR: *Force Sensing Resistor* por sus siglas en inglés) y modifica su valor en Ohmios en las terminales de salida, en una relación inversamente proporcional a la fuerza ejercida sobre su área de contacto. Es importante señalar estos sensores tienen forma circular y la fuerza ejercida por una masa (la del pie en este caso) se distribuye de manera homogénea sobre el área de contacto efectiva. Para este proyecto se usó un sensor con área de 45.36mm^2 (7.62mm de diámetro como máximo) y un grosor que no supera los 1.2mm.

Dado que estos sensores tienen un límite de fuerza de 20N[3] al que pueden ser sometidos, en este proyecto se usaron ocho FSR dispuestos de tal forma que se distribuya el peso del usuario sin sobrepasar los límites soportados por cada dispositivo. La ecuación 2 expresa la fuerza máxima total soportada F_{Tmax} como la suma de la fuerza que tolera cada uno de las ocho FSR de F_{s1} hasta F_{s8} respectivamente.

$$F_{Tmax} = F_{s1} + F_{s2} + F_{s3} + F_{s3} + \cdots + F_{s8} \quad (2)$$

$$F_{user} = m_{user} * g \quad (3)$$

$$F_{user} < F_{Tmax} \quad (4)$$

[3] Véase hoja de especificaciones del fabricante Interlink Systems® disponible en: https://www.interlinkelectronics.com/fsr-402

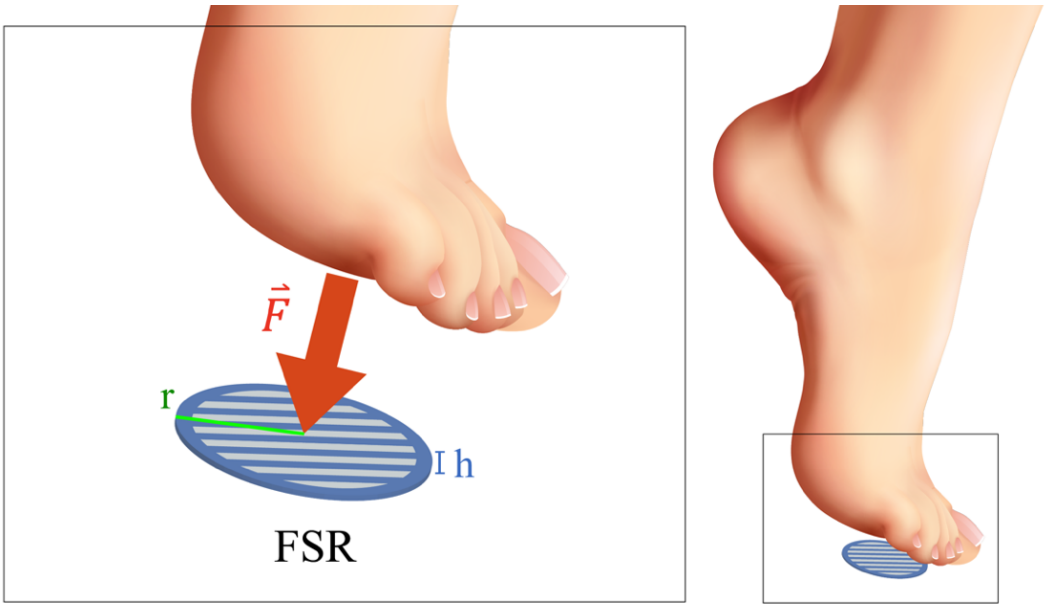


Ilustración 1. Sensor de presión resistivo para detección de la fuerza aplicada por una persona en la actividad de la marcha.

En la ecuación 3 se expresa la fuerza ejercida por el usuario F_{user} como el producto de su masa m_{user} por la aceleración de la gravedad en la tierra $g = 9.8\, m/s^2$. Es importante señalar que la fuerza ejercida por el usuario debe tener una magnitud menor a la fuera máxima total soportada por el sistema como se muestra en la ecuación 4, para evitar que se exceda el límite de presión tolerado por las FSR.

2.4.2 Sensores inerciales.

2.4.2.1 Giroscopio.

El giroscopio es un dispositivo que mide la velocidad y la posición angular de un objeto en torno de su eje de rotación (Acar y Shkel, 2009). Son sensores capaces de detectar cambios repentinos y tienen gran precisión con bajo ruido en la señal de salida siempre que se usen rangos cortos de tiempo para la medición. Estas características permiten conocer el ángulo de inclinación

del pie y su velocidad angular mientras una persona realiza movimientos con las extremidades inferiores.

2.4.2.2 Acelerómetro.

El acelerómetro es un dispositivo que permite conocer la variación de la velocidad producida mientras se realiza un movimiento, a lo largo del eje en el cual se está trabajando (Pozo, 2010). Este tipo de sensores son normalmente de tres ejes (x, y & z), gracias a ello se puede determinar la magnitud y dirección de la aceleración medida. Así se determinará el cambio de velocidad de los pies a cada paso y se mide el ángulo absoluto de inclinación que forma el sensor con respecto al suelo.

2.5 Convertidor analógico a digital

Un convertidor analógico a digital (ADC: *Analog-to-Digital Converter* por sus siglas en inglés) es la etapa en el procesamiento de señales que permite transcribir señales de naturaleza analógica, generalmente continuas en tiempo y amplitud, a una señal discreta en tiempo muestreado y amplitud cuantificada (Rodrigues, 2011). En SFFS las señales analógicas de voltaje que entregan los sensores, son convertidas a un formato digital con el propósito de facilitar su procesamiento, codificación, comprensión, y hacer la señal resultante más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

2.6 Interfaz de usuario

Se conoce como interfaz de usuario el apartado visual donde el usuario puede efectuar acciones que alteran los parámetros del sistema y al mismo tiempo puede tener una realimentación gráfica. Los zapatos de SFFS estarán en constante vigilancia de la actividad física del usuario sin que él lo note y darán en tiempo real una realimentación visual y auditiva que se explicará a continuación.

2.7 Sistema de realimentación

"Cuando una secuencia cerrada de relaciones causa-efecto existe entre las variables de un sistema se dice que hay realimentación" (Kuo, 1996). La Ilustración 2 muestra el sistema de realimentación propuesto en este proyecto, las fases presentadas allí son adaptación de las definidas por Ogata, K. (2010) en el libro Ingeniería de Control Moderna (p. 6).

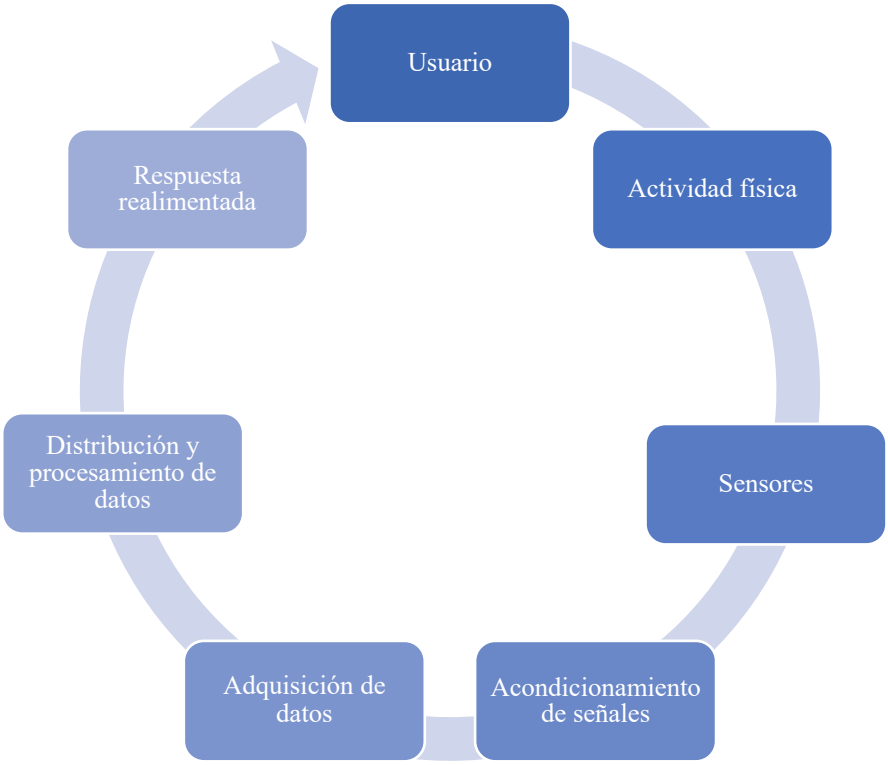


Ilustración 2. Diagrama de bloques que representa las fases que componen el sistema de realimentación usado en SFFS.

Los datos capturados por los sensores cuando el usuario realiza movimientos con las extremidades inferiores son acondicionados por circuitos eléctricos y enviados a los puertos físicos del microcontrolador; desde allí se envían vía Bluetooth a la APP instalada en el teléfono inteligente donde la información se realimenta de forma visual y auditiva al usuario.

2.8 Conceptos musicales básicos

2.8.1 Pulsaciones por minuto (bpm).

En la primera definición del diccionario de inglés Collins English Dictionary publicado por HarperCollins en Glasgow se define pulsaciones por minuto (bpm: *beats per minute* por sus siglas en inglés) como una unidad usada para medir el ritmo cardiaco o el ritmo en la música.

2.8.2 Tempo.

El tempo se refiere al "grado de celeridad en la ejecución de una composición musical y, por ext., de una composición poética." Real Academia Española, (s.f.). En música es usada para hacer referencia a la velocidad de una canción, e.g. una canción que está entre 40bpm y 60bpm es considerada de tempo lento.

2.8.3 MP3.

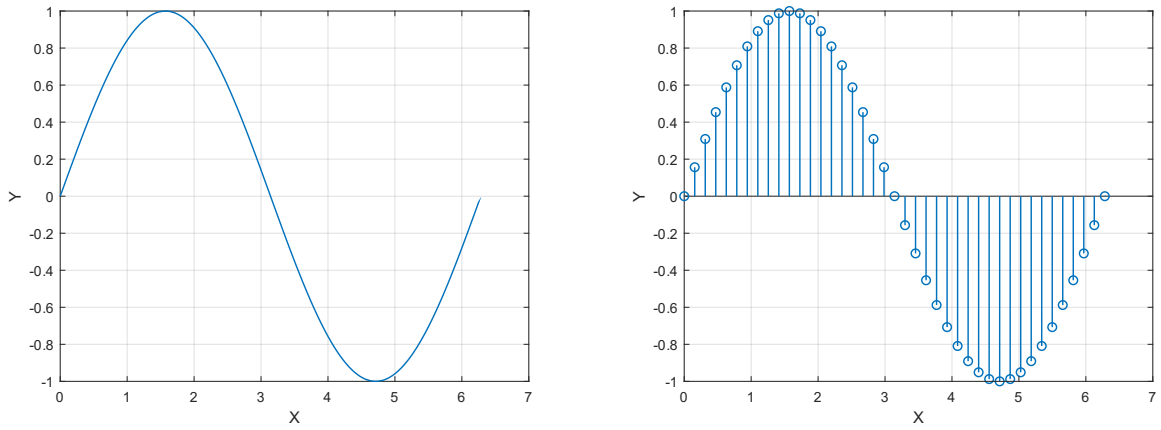
MPEG-1 Audio Layer III o MPEG-2 Audio Layer III es un formato de audio estandarizado por Picture Experts Group (MPEG), el cual permite comprimir, descomprimir y procesar audio digital a velocidades de bits entre 32 a 448 kb/s y con unas frecuencias muestreo entre 44,1 y 48 kHz. A estos dos formatos se les conoce como MP3 (Chiariglione, 2019).

2.9 Filtro digital

Como bien se menciona en Dorf y Svoboda (2006) los filtros son un concepto que ha estado implícito desde comienzos de la humanidad.

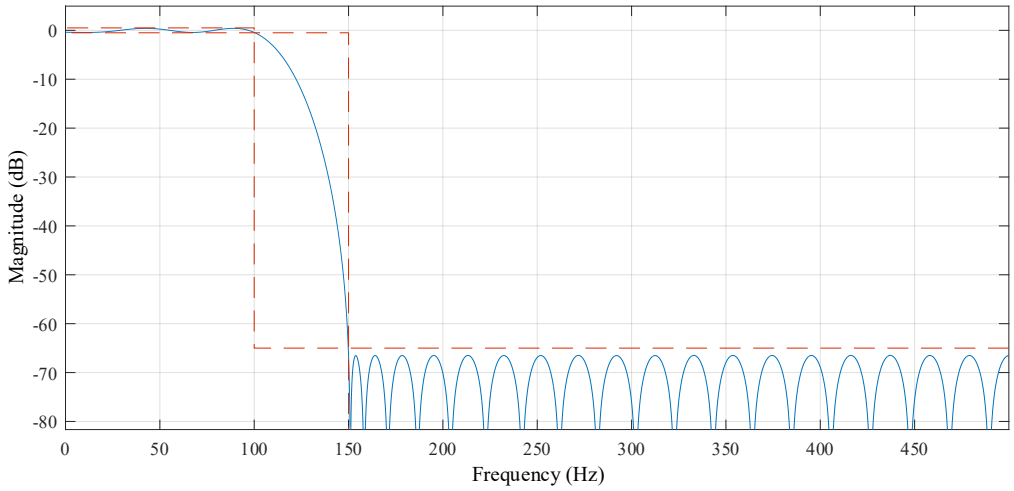
"Se utilizaba un filtro de papel para eliminar del agua y del vino impurezas y sustancias no deseadas. Un material poroso, como un papel, puede servir como un filtro mecánico. Los filtros mecánicos se utilizan para eliminar elementos no deseados como las partículas en suspensión de un líquido. De igual manera, un filtro eléctrico se puede utilizar para eliminar de una señal eléctrica elementos no deseados, como el ruido eléctrico." (p. 793).

Si bien las señales digitales a diferencia de las señales eléctricas son discretas es decir que su variable independiente está definida solo para valores discretos (Ilustración 3), estas señales digitales también pueden ser filtradas ya que los filtros digitales son sistemas que toman una la señal de entrada discreta y cambian las amplitudes de las componentes en frecuencia o elimina por completo algunas mediante proceso matemáticos que pueden ser realizados mediante software Oppenheim y Willsky (1997). En ilustración 4 se muestran algunos ejemplos.

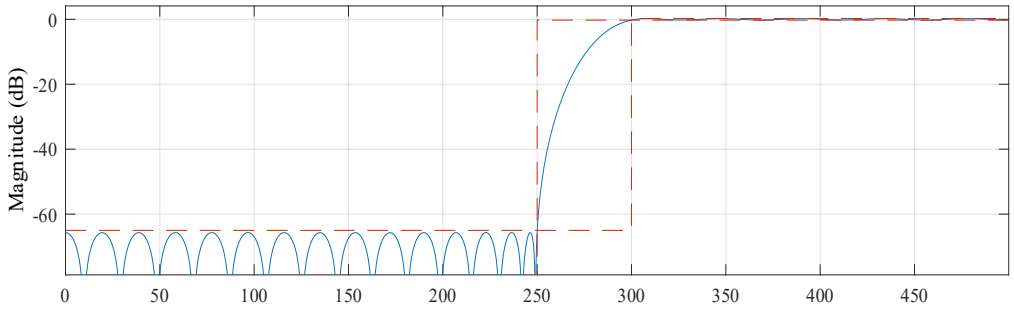


(a) (b)

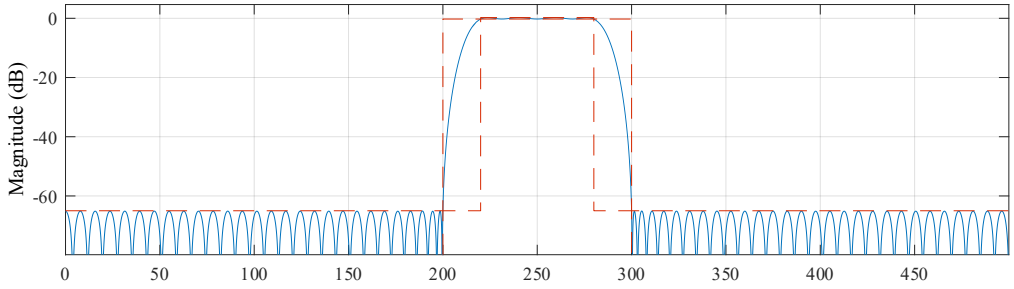
Ilustración 3. Comparación de una señal continua y una señal discreta. a) Gráfica de una señal continua. b) Gráfica de una señal discreta.



(a)



(b)



(c)

Ilustración 4. Filtros digitales de respuesta finita al impulso (FIR: Finite Impulse Response por sus siglas en ingles). a) Gráfica de una un filtro FIR pasa bajos. b) Gráfica de una un filtro FIR pasa altas. c) Gráfica de una un filtro FIR pasa banda.

3. Marco procedimental

Para el desarrollo de este proyecto se toma como base la metodología Top-Down que consiste en establecer algunas especificaciones iniciales en un nivel jerárquico mayor. Luego este nivel superior se transfiere sucesivamente de forma hereditaria a cada parte del proyecto con nivel inferior de jerarquía (Aleixos et al., 2003). Se elige esta metodología ya que brinda la posibilidad de desarrollar tareas secuenciales y de forma progresiva para alcanzar el objetivo del proyecto. Crespi, Galstyan y Lerman (2008) comentan la pertinencia de un enfoque centralizado donde la resolución ordenada de componentes forme el conjunto de especificaciones de un estado global del sistema. La Ilustración 5 muestra las fases seguidas para realizar el SFFS.

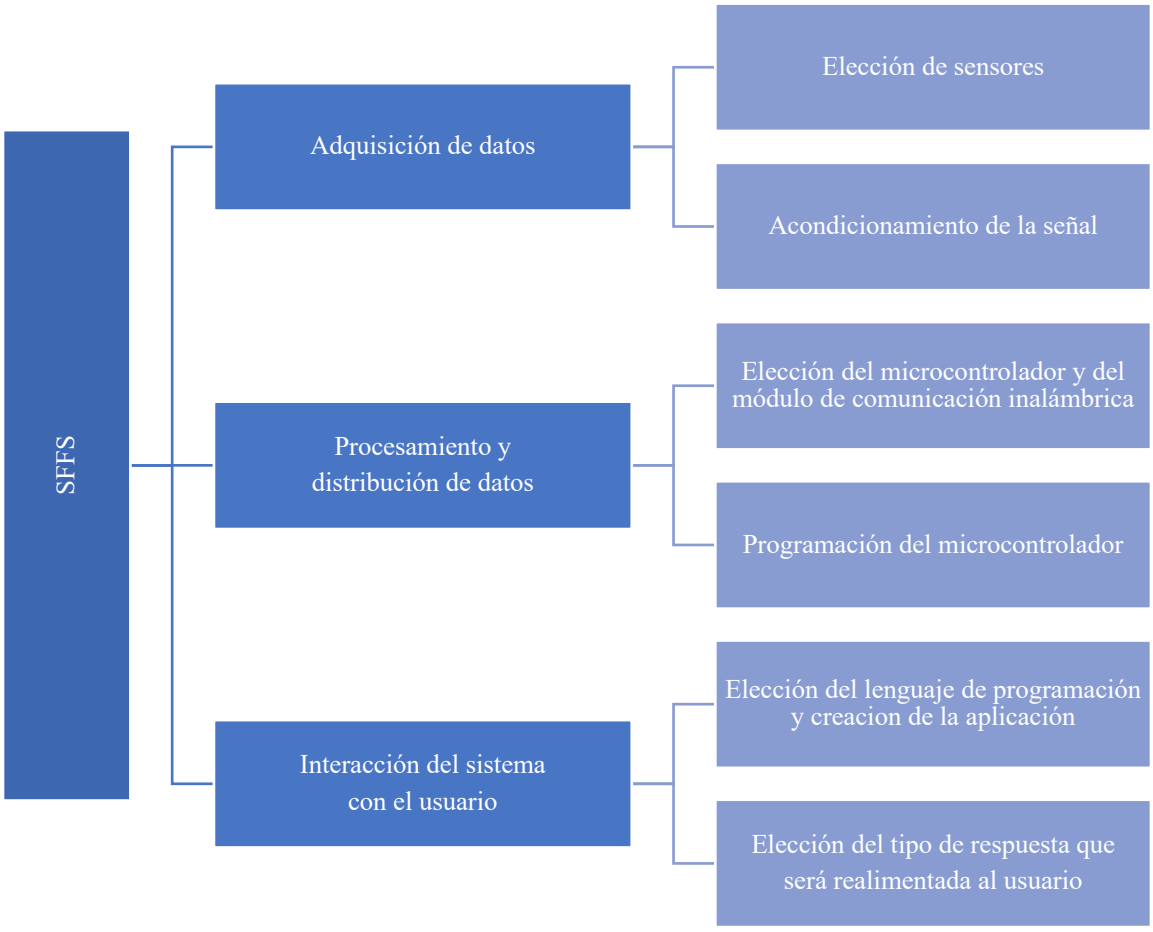


Ilustración 5. Diagrama jerárquico que representa el marco metodológico propuesto para desarrollar SFFS.

3.1 Adquisición de datos

Hay diversas maneras de capturar datos de las variables que intervienen en la actividad física realizada por una persona, sin embargo, SFFS es un sistema pensado como WT y por ello se hace necesario usar sensores integrados a los zapatos siguiendo el enfoque de IoHT. Se encontraron evidencias suficientes, como se menciona en los antecedentes, que demuestran la utilidad de usar FSR, acelerómetro y giroscopio para la vigilancia de la actividad física y la marcha. El tamaño de los sensores y su distribución es determinante para realizar una medición no invasiva que permita obtener datos precisos con bajo consumo de energía, por ello el proceso de selección se hizo importante en este proyecto y se explica a continuación:

3.1.1 Elección y ubicación de los sensores

La elección de los sensores usados en los zapatos fue un aspecto importante, teniendo en cuenta las recomendaciones hechas por Saidani et al. (2019) referentes a la comodidad que debe sentir el usuario al utilizar WT, pues de ello depende la fiabilidad de las señales obtenidas. En este contexto se seleccionó el integrado MPU-92/65 de aproximadamente 15mm x 25mm que cuenta con acelerómetro y giroscopio de tres ejes (IMU: *Inertial measurement unit* por sus siglas en inglés). La ubicación de esta IMU se hizo basada en los resultados obtenidos en la investigación de Lidstone et al. (2019) enfocada en la medición del área de contacto de presión plantar de usuarios al caminar, usando una técnica de estudio óptico de los campos de presión que actúan entre la superficie plantar del pie y una superficie de apoyo.

La Ilustración 6 muestra el lugar seleccionado para ubicar la IMU en SFFS, pues salvo en situaciones de deformidad de pie plano, en esta zona no hay ningún contacto con el suelo y el arco longitudinal del pie amortigua las fuerzas durante las fases de pisada y media de la marcha

(Klimiec et al., 2017). Como la parte media del arco del pie evita el contacto con el suelo, se decidió ubicar la IMU en este lugar dado que se garantiza que el dispositivo no va a sufrir daño.

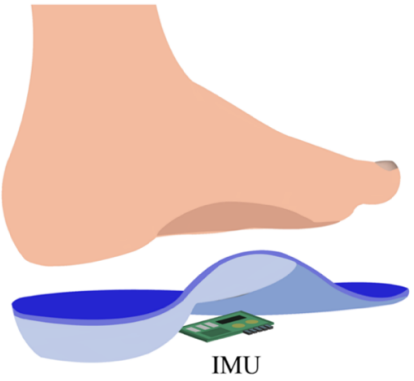


Ilustración 6. Ubicación de la IMU en la plantilla de SFFS para evitar incomodidad al usuario o daños del dispositivo por la presión generada por el pie.

Para capturar los datos de la presión ejercida por los pies al caminar en SFFS se utilizaron los sensores FSR-402 de Interlink Electronics®, que son dispositivos robustos de película gruesa de polímero que exhiben una disminución de la resistencia cuando se aumenta la fuerza aplicada a la superficie del sensor. Los FSR se seleccionaron debido a su tamaño, flexibilidad y bajo consumo de energía; las características determinantes se resumen en la Tabla 1.

La distribución de los FSR usada en SFFS está basada en los resultados obtenidos por Ohnishi, Terada y Tsukamoto (2018), donde se seleccionaron cuatro puntos para ubicar sensores de presión en una plantilla de zapato, después de una evaluación de veintidós posturas corporales. Es posible observar en la parte relacionada con el pie izquierdo en la Ilustración 7 varias líneas que unen pares de puntos, estas representan los lugares del pie donde se ejerce mayor presión al ejecutar las posturas para diferentes usuarios. La posición óptima de los sensores de presión para reconocimiento del movimiento se realizó en esa investigación usando un algoritmo de agrupación k-means (Ohnishi, Terada y Tsukamoto, 2018); los óvalos sombreados demarcan el área de mayor

contacto del pie y con base en ello, se propuso para SFFS la ubicación de cuatro FSR como se muestra en la plantilla del pie derecho en la Ilustración 7.

Tabla 1. Adaptación de la hoja de datos del FSR 400 series de Interlink Systems®[4]

Característica	Valor
Fuerza de accionamiento	0.1 Newtons
Rango de sensibilidad a la fuerza	0.2 – 20 Newtons
Tamaño	7.6mm de diâmetro
Gama de espesores	0.2 - 1.25 mm
Resistencia a la desconexión	>10M Ohms
Tiempo de subida del dispositivo	<3 microsegundos
Rango de temperatura para funcionamiento	-30° - +70° C

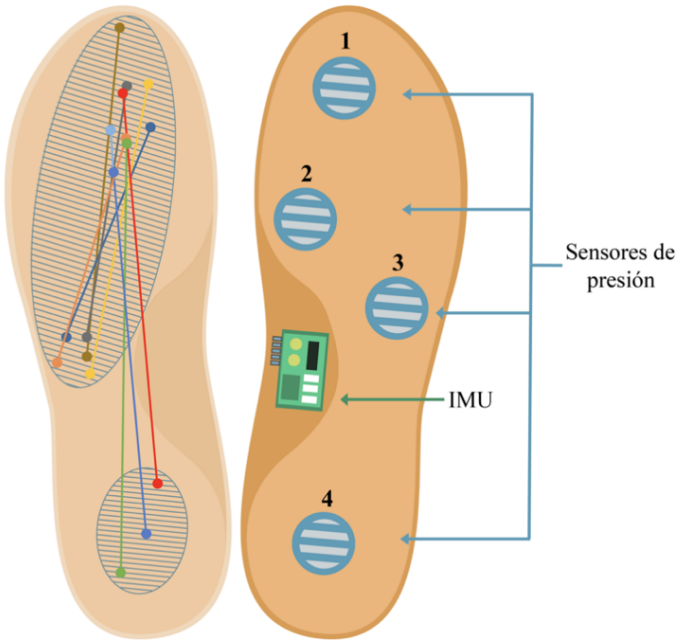


Ilustración 7. Adaptación de los resultados obtenidos por Ohnishi, Terada y Tsukamoto (2018) para la ubicación de las FSR y la IMU en SFFS.

[4] Véase hoja de especificaciones del fabricante Interlink Systems® disponible en: https://www.interlinkelectronics.com/fsr-402

3.1.2 Elección del microcontrolador y la batería

La elección del microcontrolador (MCU: *Microcontroller Unit* por sus siglas en inglés) usado para la adquisición y procesamiento de los datos capturados por los sensores, tuvo en cuenta algunos aspectos determinantes que se explican a continuación: inicialmente se buscaron tarjetas integradas que tuvieran un tamaño menor a 7cm de longitud dada la necesidad de ubicarlas en un zapato y la comodidad del usuario es prioridad en SFFS como ya se expuso previamente. Con esto como premisa, se realizó una búsqueda de los MCU disponibles en el mercado que pudieran integrarse directamente en los zapatos sin utilizar accesorios extra.

Con la primera selección de posibles tarjetas a usar entró en consideración el segundo aspecto importante, la disponibilidad de puertos analógicos; como se vio en la Ilustración 7, SFFS usa una plantilla con cuatro FSR y una IMU para cada zapato, eso implica que la tarjeta elegida tuviese por lo menos seis entradas analógicas que permita conectar todos los sensores. Si bien en el mercado hay variedad de tarjetas que cumplen con las primeras dos condiciones, el tercer aspecto para tener en cuenta para la elección redujo las posibilidades. La conectividad inalámbrica fue siempre una prioridad en SFFS, en Wang et al. (2015) se explicaron las ventajas que tenían los sistemas inalámbricos para la transmisión de datos, la fiabilidad de los resultados obtenidos en cuanto a la naturalidad de los usuarios al usar el sistema y la prolongación de la vida útil de la batería. En ese sentido, se buscó una tarjeta que tuviera diferentes tipos de comunicación inalámbrica y así poder determinar el que mejor se acomodase a los propósitos establecidos para el proyecto.

En cuanto a la alimentación del sistema, se tuvo en cuenta la relación de proporcionalidad directa que hay entre la disponibilidad de almacenamiento de energía y el tamaño de las baterías eléctricas. Así, el uso de una tarjeta con bajo consumo de energía garantizaría que la batería usada

también fuese de un tamaño que pueda ser integrado en los zapatos sin afectar la comodidad del usuario. Los cuatros aspectos considerados: tamaño, disponibilidad de puertos analógicos, conectividad inalámbrica y bajo consumo de energía eléctrica; combinados a la importancia de usar dispositivos de bajo costo que permitan al usuario tener acceso a estos sistemas, como se explicó en Carbonaro, Lorussi y Tognetti (2016), hicieron que la placa elegida fuese la ESP32-WROOM32 de Espressif Systems® que se muestra en la Ilustración 8.

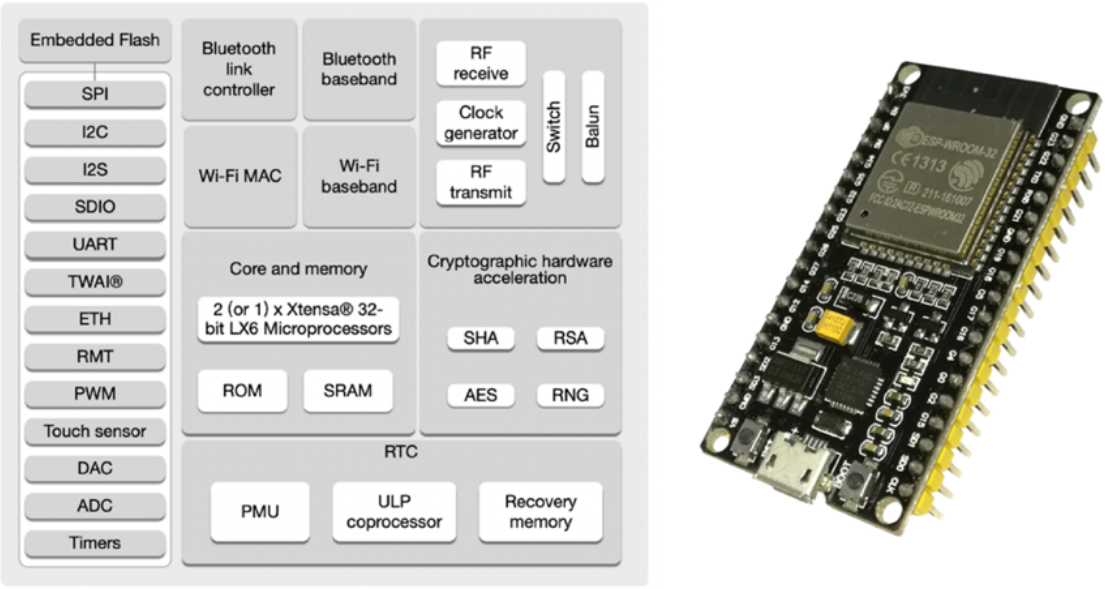


Ilustración 8. Diagrama de bloques funcional y vista superior de la placa de desarrollo ESP-WROOM-32 de Espressif Systems®.

La ESP32-WROOM-32 en una tarjeta de desarrollo de la familia ESP32-DevKitC que incluye el MCU ESP32 de Espressif Systems®. Las características determinantes para su elección en este proyecto se encuentran resumidas en la Tabla 2.

Teniendo en cuenta las características de alimentación de la ESP32 expuestas en la Tabla 2, se usaron dos baterías recargables de polímero de litio idénticas para energizar los circuitos

(ESP32 y plantillas con cuatro FSR y una IMU) en cada zapato. El voltaje de salida es de 3.7v y la corriente de 1A[5]. Las medidas son 5cm de largo, 3cm de ancho y 0.5cm de profundidad.

Tabla 2. Adaptación de la hoja de datos del ESP32-WROOM-32 de Espressif Systems®.

Característica	**Especificación**
Bluetooth v4.2 BR/EDR	Receptor NZIF con sensibilidad de –97 dBm
Cristal integrado	Cristal de 40MHz
Memoria Flash SPI integrada	4MB
Tensión de funcionamiento/alimentación	2.7 V ~ 3.6 V>10M Ohms
Corriente de operación	Aproximadamente 80mA
Corriente mínima de alimentación	500mA
Rango de temperatura de funcionamiento	–40 °C ~ +85 °C
Tamaño de la placa	5 cm x 2.9 cm

La distribución de los sensores se presentó en la Ilustración 6 y se siguió ese diseño para fabricar la plantilla que está en la parte interior de los zapatos; para la parte exterior se establecieron dos puntos donde no se genera grandes niveles de presión por el pie (ver Ilustración 9) y allí se ubicaron la batería y la placa ESP32. El diseño de los zapatos usados en SFFS teniendo en cuenta los puntos escogidos mencionados anteriormente se muestra en la Ilustración 9.

En el zapato izquierdo de la Ilustración 9, es posible observar una vista de costado donde se presenta el diseño del bolsillo fabricado con tela elástica y la solapa que sirve de tapa al usar velcro para fijarse. La vista superior del zapato derecho muestra los dos compartimientos que cada zapato tiene a los costados. En el bolsillo interior de cada zapato se ubica la ESP32 y en el exterior

[5] Véase hoja de especificaciones del fabricante LiPol Battery Co® disponible en: https://www.li-polymer-battery.com/wp-content/uploads/2021/03/LP103035-1000mAh-Datasheet.pdf

la batería. Estos compartimientos sobresalen a cada costado cinco milímetros como máximo, sin afectar la comodidad del usuario al usar SFFS.

Ilustración 9. Vista lateral y superior de los zapatos diseñados para SFFS con señalización de los bolsillos exterior e interior y la solapa con velcro que cierra el bolsillo.

3.2 Distribución y procesamiento de datos

Los datos de presión (en Pa), velocidad angular (en rad/s) y aceleración (en m/s^2) capturados por los sensores se acondicionaron eléctricamente y se envían a la tarjeta ESP32 por los puertos físicos de la placa (pines 39, 34, 35 y 36 para las FSR 1, 2, 3 y 4 respectivamente; 21 y 22 para la IMU). En la tarjeta se realiza el proceso de filtrado y acondicionamiento de la señal que será transmitida a la APP instalada en el teléfono inteligente usando la comunicación Bluetooth. La Ilustración 10 muestra las fases de recolección y distribución de datos usadas en SFFS que permitieron la realimentación de información al usuario.

Ilustración 10. Diagrama de bloques que representa las fases de recolección, acondicionamiento y distribución de los datos en SFFS.

Una vez que se seleccionaron los sensores, se determinó su ubicación y se hizo la conexión eléctrica correspondiente, se fijaron en la plantilla del zapato para garantizar que la fase de recolección de datos que se muestra en el diagrama de bloques de la Ilustración 10 se cumpliera satisfactoriamente.

3.2.1 Acondicionamiento de la señal

Al realizar pruebas con resultado satisfactorio referente a la conexión y recepción de los datos capturados por los sensores, se procedió a diseñar el circuito electrónico que entregara una señal de voltaje relacionada a la presión ejercida por los pies en cada sensor. Para ello se usó un divisor de voltaje donde cada FSR fue alimentada por la salida de 3.3V de la ESP32 y una resistencia de 3.3KΩ conectada a tierra, teniendo en cuenta la relación de voltaje-fuerza de la hoja de datos de la FSR402. La tensión en el nodo del divisor se conectó directamente en los puertos con ADC integrados de la placa. Para las señales entregadas por la IMU no hizo falta ningún circuito adicional de acondicionamiento dado que la ESP32 cuenta con dos entradas específicas para acelerómetro y giroscopio, de hecho, esta fue una de las razones que contribuyeron en la elección de la placa.

Una vez que los datos capturados por los sensores estaban en la placa se organizaron, filtraron y codificaron por código para ser transmitidos vía Bluetooth a la APP. Este

acondicionamiento será explicado en detalle en el siguiente apartado, donde también se describe la programación de la ESP32.

3.2.2 Programación del microcontrolador

Las tarjetas ESP32 se programaron en el entorno de desarrollo integrado (IDE: *Integrated Development Environmentace* por sus siglas en inglés) de Arduino® ya que es un software de uso libre. La plataforma cuenta con una interfaz de programación de lenguaje propio basado en Processing, similar a C++ y está orientada para la escritura y carga de programas a placas compatibles con Arduino® y otros microcontroladores como el usado en este proyecto.

Dado que los valores medidos por el acelerómetro se ven afectados por los movimientos del sensor cuando el usuario camina y la señal generada tiene mucho ruido en tiempos cortos de muestreo, hace falta combinar los datos con los capturados por el giroscopio, que a su vez reaccionan eficientemente a los cambios bruscos, pero generan ruido a mediano y largo plazo. Por lo tanto, los dos sensores integrados en la IMU se complementan y la combinación de sus mediciones, acompañada de una fase de filtrado, permiten obtener un valor más preciso de la orientación del pie. Para este proceso se optó por el uso de un filtro complementario, una versión simplificada del filtro de Kálman que une un filtro pasa-altas para el giroscopio y uno pasa-bajas para el acelerómetro.

El cálculo del ángulo de inclinación de los valores obtenidos por el acelerómetro para los ejes x & y (accx y accy) se determinó usando las ecuaciones 5 y 6 respectivamente. Donde x, y, z son los valores obtenidos directamente por el acelerómetro de la IMU.

$$\text{accx} = \tan^{-1}\left(\frac{x}{\sqrt{y^2 + z^2}}\right) \tag{5}$$

$$\text{accy} = \tan^{-1}\left(\frac{y}{\sqrt{x^2 + z^2}}\right) \tag{6}$$

Los valores capturados por el giroscopio se combinaron con los ángulos calculados en las ecuaciones 5 y 6. Usando el filtro complementario expresado en la ecuación 7, se obtiene el ángulo de inclinación para los ejes x, y & z que permiten determinar la orientación del pie.

$$\text{ang} = 0.98 * (\text{angPrev} + \text{gir} * 0.010) + 0.02 * \text{acc} \quad (7)$$

Donde ang es el valor del ángulo, angPrev el valor previo medido, gir el ángulo medido por el giroscopio y acc el valor calculado en la ecuación 5 o 6 para el eje correspondiente. El valor de 0.010 es el tiempo en segundos que pasó desde la última vez que se calculó ang. Los valores constantes 0.98 y 0.02 fueron calculados para satisfacer los requerimientos del filtro complementario. La orientación del pie se determina usando estos valores y esa información se le muestra al usuario en la APP como se explicará en la parte de realimentación visual.

El ángulo para cada eje, calculado con la ecuación 7, es un valor ß entre -90˚ y 90˚ que representa la inclinación máxima y mínima para cada posición del pie, así como se muestra en la Ilustración 11. En la parte a) y b) se presenta el plano transversal del zapato y este es el eje X para SFFS; la parte c) y d) muestran el plano sagital que es el eje Y; finalmente d) y e) muestran el plano coronal y es el eje Y.

Referente a los valores de presión, se tomó la lectura como entradas analógicas de la ESP32 usando su protocolo interno ADC, razón por la cual no hizo falta otra conversión o filtro. Cada dato que se obtuvo del divisor de las resistencias seleccionadas y las FSR es un voltaje entre 0 y 3.3v directamente proporcional a la presión ejercida por el pie. En total se reciben cuatro valores de presión referentes a las FSR distribuidas en el zapato.

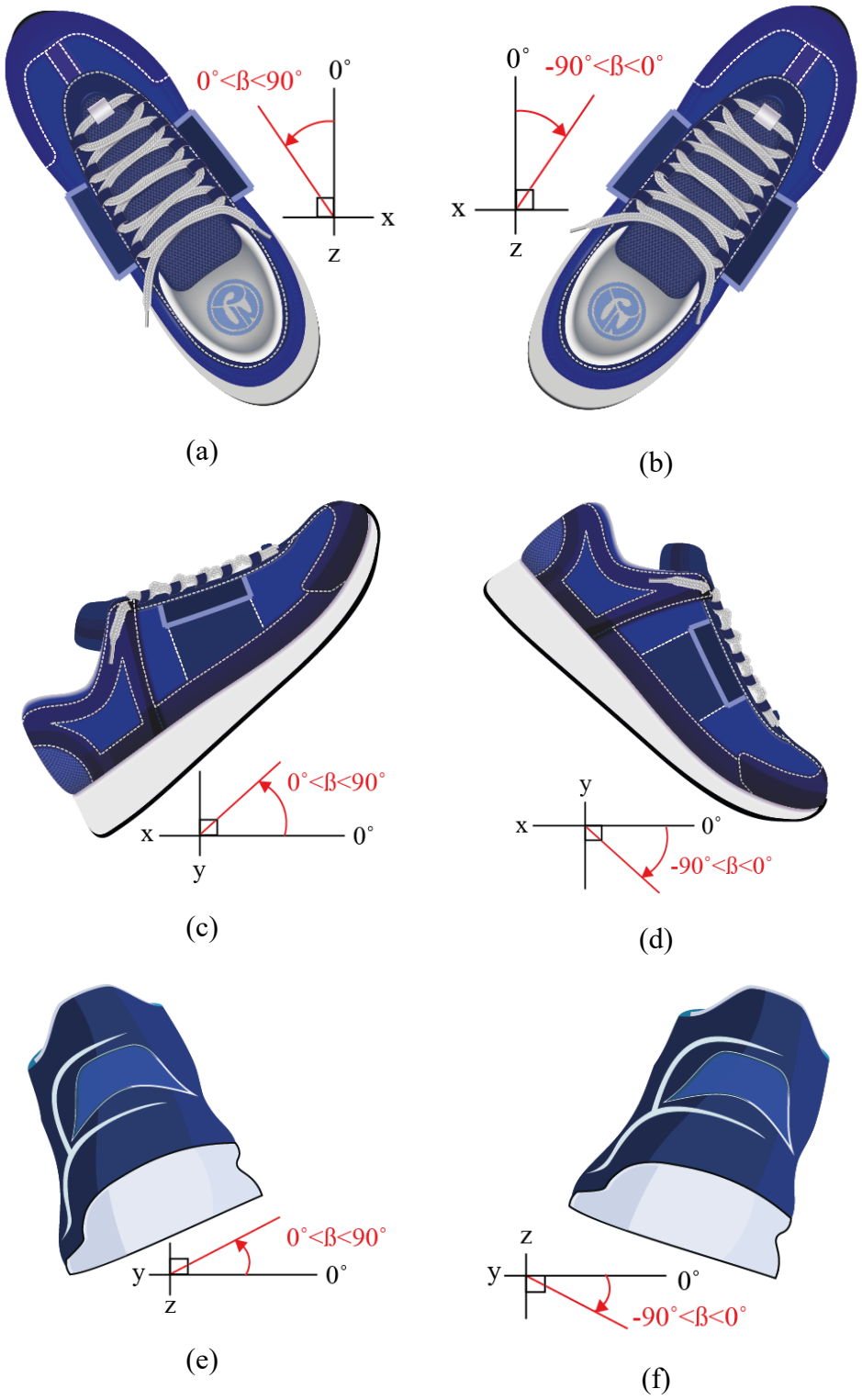


Ilustración 11. Vistas del ángulo de inclinación del zapato en los tres ejes medidos en SFFS. a) Plano transversal en rango de 0˚ a 90˚ b) Plano transversal en rango de -90˚ a 0˚ c) Plano sagital en rango 0˚ a 90˚ d) Plano sagital en rango de -90˚ a 0˚ e) Plano coronal en rango de 0˚ a 90˚ f) Plano coronal en rango de -90˚ a 0˚.

La Ilustración 12 muestra la ubicación de la plantilla con las cuatro FSR y la IMU instaladas en la parte interior del zapato diseñado para SFFS. La conexión de los cinco dispositivos con la ESP32 y la batería ubicada en los bolsillos laterales del zapato es alámbrica y no afecta la comodidad del usuario al usarlos. En el código programado para el MCU los tres valores calculados de la orientación del pie y los cuatro de la presión se almacenaron en variables reales de dos dígitos, suficiente para obtener la resolución requerida. Las siete variables se concatenaron en un arreglo de tipo carácter como se muestra en la Ilustración 13.

Ilustración 12. Ubicación de la plantilla con las cuatro FSR y la IMU dentro del zapato diseñado para SFFS.

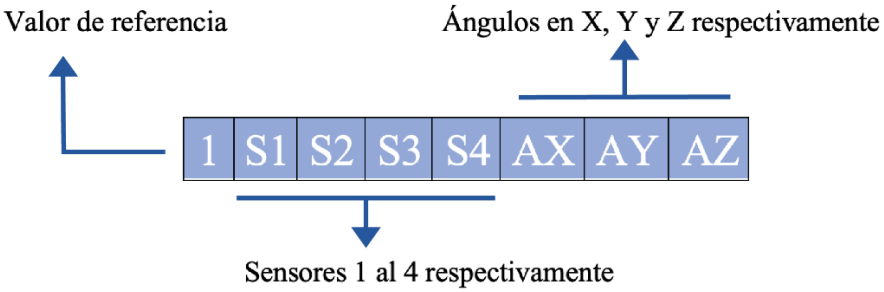


Ilustración 13. Codificación de los valores de los sensores procesados por la ESP32 para el envío por Bluetooth a la APP.

El arreglo codificado por cada ESP32 debe ser enviado a la APP para que el usuario tenga realimentación de ambos pies, sin embargo, esto implica hacer una conexión Bluetooth simultanea entre el teléfono y las dos placas. Debido a las limitaciones de la conexión Bluetooth del teléfono con diferentes dispositivos al mismo tiempo, que podrían afectar la fluidez y la velocidad de respuesta suficiente para que SFFS fuera considerado un RTS, se decidió usar una ESP32 adicional encargada de recibir todos los datos concatenados y enviarlos via Bluetooth a la APP en una conexión única.

En el zapato derecho se agregó la tercera ESP32 (ESPBLU) que enviará el arreglo total con todos los datos a la APP y además, se conecta por puerto serial a la ESP32 (ESPDER) encargada de capturar los datos de las FSR y la IMU. En la ESPDER se usaron los dos núcleos del MCU: uno encargado de capturar los datos de los sensores en el zapato derecho, recibir por Bluetooth los datos de los sensores capturados por la ESP32 del zapato izquierdo (ESPIZQ) y la concatenación del arreglo total; el otro núcleo es usado para la ejecución de una de las realimentaciones auditivas que se explicarán más adelante. En la ESPBLU también se usaron los dos núcleos: uno dedicado a la lectura de los datos por puerto serial desde ESPDER, el otro encargado del envío vía Bluetooth del arreglo total concatenado a la APP.

En la Ilustración 14 se muestra el esquema general de comunicación entre las ESP32. Como se explicó antes, ESPIZQ envía los datos capturados por los sensores del zapato izquierdo por Bluetooth para ESPDER, allí se concatenan con los datos capturados por los sensores del zapato derecho y se envían por puerto serial a ESPBLU. En esta última fase el arreglo codificado con los datos de los sensores de ambos zapatos y la información de la realimentación auditiva se envía por Bluetooth a la APP. El código de programación usado en cada ESP32 se puede ver en detalle en los anexos uno, dos y tres (ESPIZQ, ESPDER, ESPBLU respectivamente).

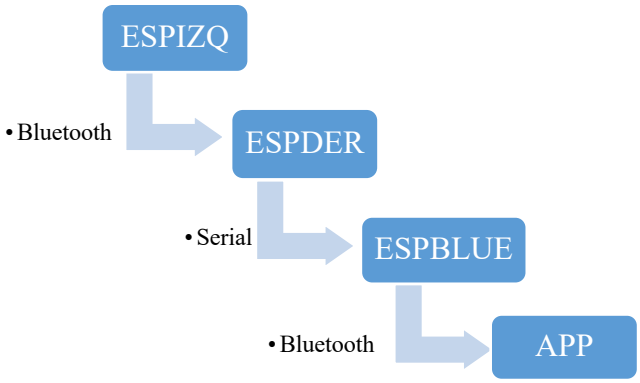


Ilustración 14. Esquema de conexión entre las ESP32 usado en SFFS para el envío de datos a la APP instalada en el teléfono inteligente.

3.3 Interfaz de sistema con el usuario

La HCI implementada en SFFS puede ser representada de forma general como se muestra en la Ilustración 15. La primera parte es el prototipo de dispositivo adaptado al calzado que captura los datos de los sensores y luego del tratamiento explicado anteriormente, los envía por la interfaz de conexión inalámbrica. En la segunda parte la información procesada es presentada al usuario como realimentación auditiva y visual a través de la APP desarrollada.

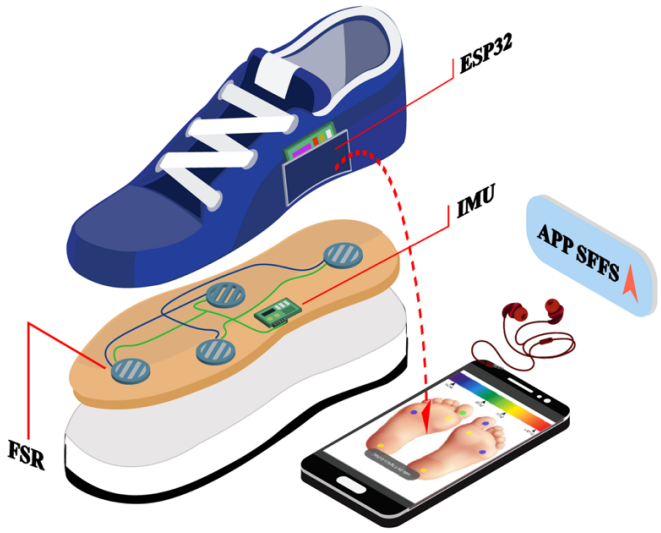


Ilustración 15. Esquema general del prototipo de dispositivo adaptado al calzado y la APP desarrollada para realimentación de información usada en SFFS.

Al iniciar la APP el usuario observa una animación que presenta el logo de SFFS y luego la sincronización del dispositivo para la conectividad Bluetooth®, en esta parte se debe establecer la conexión entre el teléfono y los zapatos. En la pantalla principal hay un menú que se despliega desde la izquierda donde es posible seleccionar las diferentes realimentaciones visuales (explicadas en la sección 3.4), para las auditivas (explicadas en la sección 3.5) se debe acceder desde el menú superior que se presenta en la Ilustración 16. Los botones disponibles en la interfaz gráfica del menú superior permiten controlar la reproducción de la música y cambiar entre los diferentes tipos de respuesta realimentada auditivamente al usuario.

Ilustración 16. Iconos del menú superior de SFFS Heat Maps. a) Inicio de la reproducción y SFFS Equalizer. b) Pausa de la reproducción. c) Cambio de canción. d) Inicio de SFFS Music bpm Select. e) Inicio de SFSS Music Band

Cuando está activa la realimentación visual por gráficas lineales (explicada en la sección 3.4.2) los iconos del menú superior cambian como se observa en la Ilustración 17. En la Ilustración 18 se muestra un diagrama de flujo que representa las opciones que se presentan para que el usuario acceda a las realimentaciones visuales desde el menú desplegado desde la parte izquierda.

Ilustración 17. Iconos menú del superior de SFFS Line Graphs. a) Inicio de la reproducción y SFFS Equalizer. b) Pausa de la reproducción. c) Cambio de canción. d) Activación de plano para las gráficas e) Selección de sensores a graficar

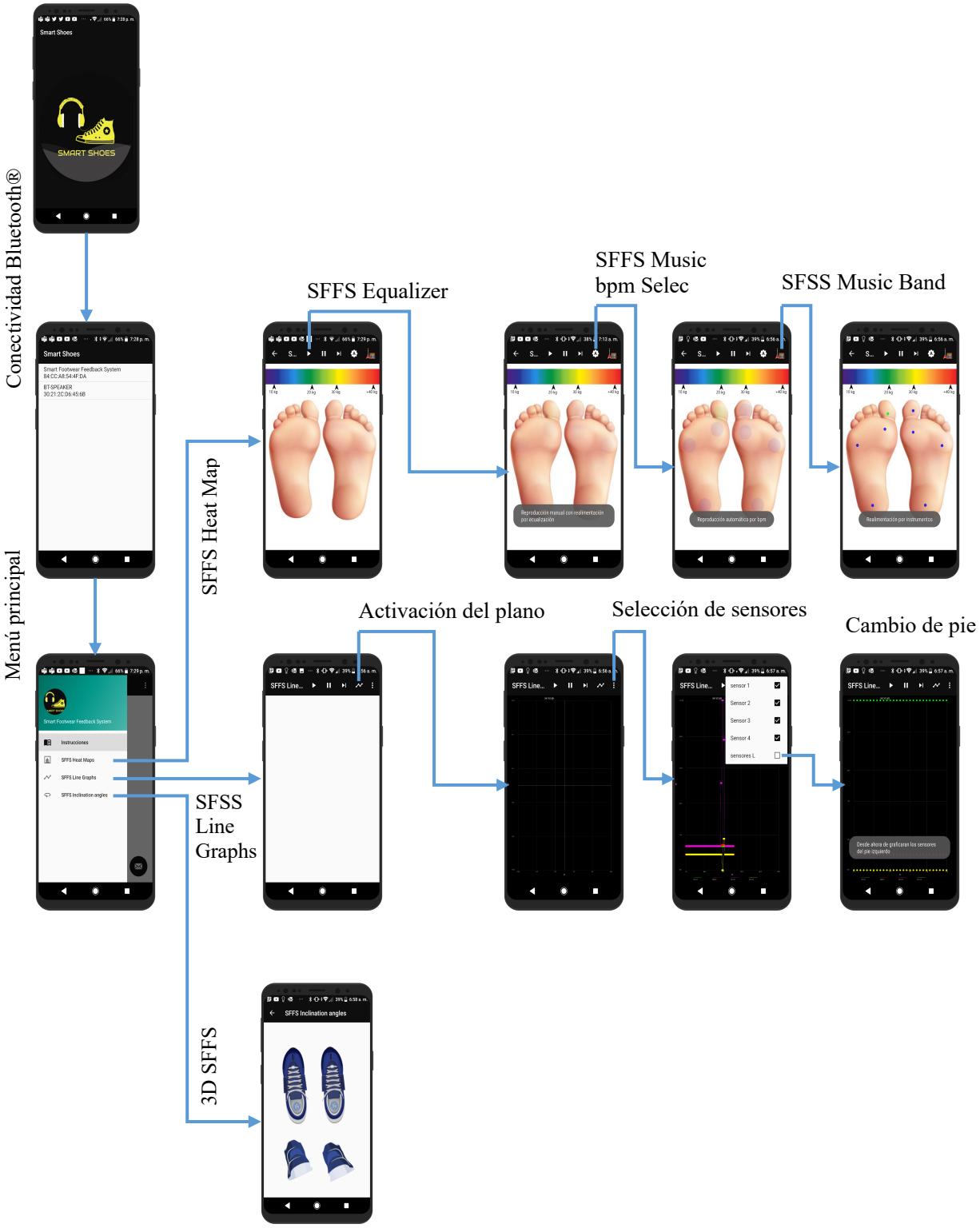


Ilustración 18. Diagrama de flujo de la APP utilizada en SFFS para acceder a los diferentes tipos de realimentación disponibles.

3.3.1 Elección del lenguaje de programación y entorno de desarrollo

Según las cifras del portal estadístico *Statcounter Global Stats* el sistema operativo móvil con la mayor cuota de mercado desde enero del 2013 hasta enero de 2021 es Android®. De hecho, el 72.83% de los dispositivos móviles cuentan con dicho sistema operativo[6], tal como se muestra en la Ilustración 19. Teniendo en cuenta esto, se decidió desarrollar la APP para Android® pensando en la población que podrá instalarla y usar SFFS.

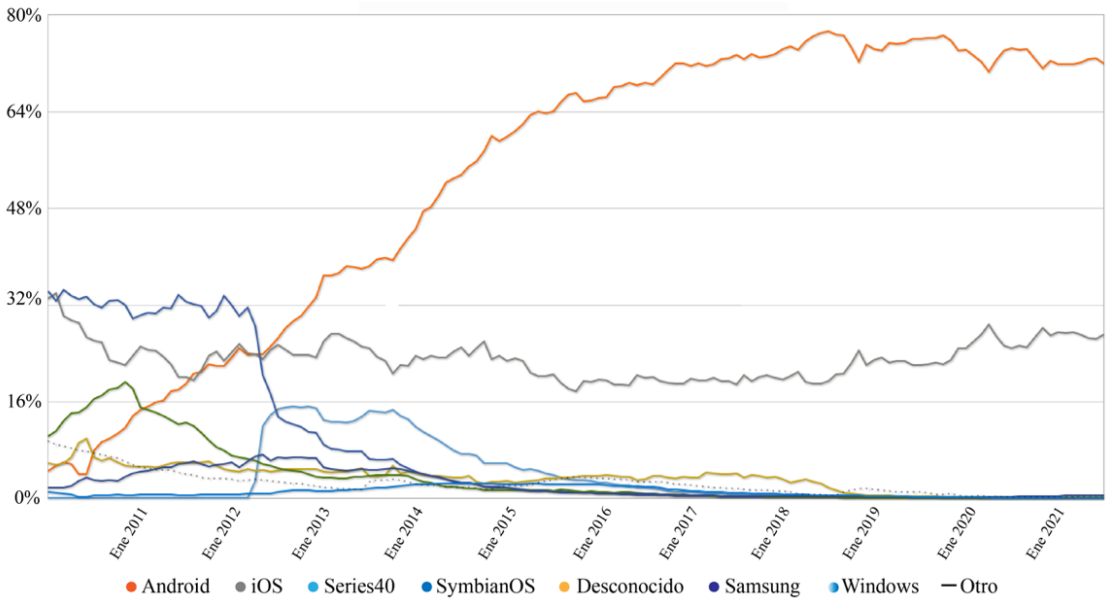


Ilustración 19. Gráfica del porcentaje de los sistemas operativos para dispositivos móviles más usados entre enero de 2010 y enero de 2021. Fuente: Statcounter Global Stats.

El IDE seleccionado para la programación de la APP fue *Android Studio* porque es el que más se destaca, como se menciona en el sitio oficial para desarrolladores de aplicaciones: Android Developers. (s.f.). Introducción a Android Studio. El lenguaje de programación utilizado fue Java debido al fácil acceso a diversas herramientas que proporciona el kit de desarrollo de software

6 Véase la gráfica disponible en: https://gs.statcounter.com/os-market-share/mobile/worldwide#monthly-201001-202107

(SDK: *Software Development Kit* por sus siglas en inglés) y la documentación disponible de la interfaz de programación de aplicaciones (API: *Application Programming Interface* por sus siglas en inglés).

3.3.2 Programación de la APP

Para la programación de SFFS se usaron vararías librerías de repositorios alojados en GitHub que permitieron acortar el tiempo de desarrollo de la APP y dar mayor calidad a la misma. El sitio oficial para desarrolladores de aplicaciones Android Developers fue crucial para encontrar información referente a librerías necesarias, se entrará en detalle sobre este tema cuando se expongan las realimentaciones visuales y auditivas.

Para la implementación de la comunicación Bluetooth en la APP se usó la librería bluetoothjhr[7] Jhr (2020) que establece el teléfono en modo esclavo para la recepción de datos enviados por ESPBLU. El código usado se puede ver en el Anexo 5. Siguiendo las recomendaciones mencionadas en la investigación de Xu et al. (2012) se usaron hilos para la programación de la APP dado que permiten hacer una simulación de gestión de datos en paralelo y posibilitan un correcto funcionamiento cuando se intenta acceder a la interfaz de usuario (UI: User Interface por sus siglas en inglés). Dado que la UI tiene un hilo principal (UIThread) que puede congestionase, se recurrió a la creación de varios hilos que funcionan de manera paralela con ayuda de la clase Runnable y el método run(), y desde allí se gestionan los datos con algoritmos de selección. Ver los Anexos 6, 7, 8 y 9.

[7] Librería disponible en el repositorio: https://github.com/jose-jhr/-bluetoothjhr

3.4 Realimentación visual

El sistema dispone de ocho sensores de presión y dos sensores inerciales encargados de capturar las variables físicas de los zapatos. SFFS ofrece dos tipos de realimentación visual para los sensores de presión y uno para para los sensores inerciales como se explicará a continuación.

3.4.1 Realimentación por mapas de calor (SFFS Heat Maps)

SFFS Heat Maps es un método de realimentación donde se le presenta al usuario una interfaz gráfica en la APP compuesta por la imagen de la planta de dos pies y una barra con colores de referencia que va desde el violeta hasta el rojo, como se muestra en la Ilustración 20.

Ilustración 20. Imagen mostrada al usuario a través de la APP como realimentación de los mapas de calor de la presión capturada por las FSR.

Los puntos de color en las plantas de los pies que se pueden observar en la Ilustración 20 están ubicados en la misma posición que las FSR (ver Ilustración 7) y representan la presión ejercida por el pie en ese sensor específico. La tonalidad de cada punto varía de color entre azul y rojo, de menor a mayor presión respectivamente. La escala de referencia usada en SFFS Heat Maps

se presenta en la Ilustración 21, donde el azul oscuro representa un peso de entre 0kg y 10 Kg, el verde 11Kg y 20Kg, el amarillo 30Kg y 40Kg, el rojo más de 40Kg. Los valores están en concordancia a los resultados obtenidos en la investigación de Ohnishi, Terada y Tsukamoto (2018) y los valores máximos de fuerza soportados por la FSR (ver tabla 1).

Ilustración 21. Referencia de colores usados en la realimentación por mapas de calor (SFFS Heat Maps) para indicarle al usuario la presión ejercida en cada FSR.

Para la representación de los FSR que aparecen en la Ilustración 20 se usó la librería *SpinKitView*[8] ybq (2019) que contiene varias animaciones, en este caso se seleccionó la animación *pulse* que consiste en un círculo que se expande y desparece cíclicamente. Estos objetos son llamados *spinners*. Por medio de un algoritmo de selección se determina el grado de presión que se está ejerciendo y el color de las representaciones se varía utilizando el método *setColor()* que permite cambiar el color de los *spinners* (Ver Anexo 6 y 9).

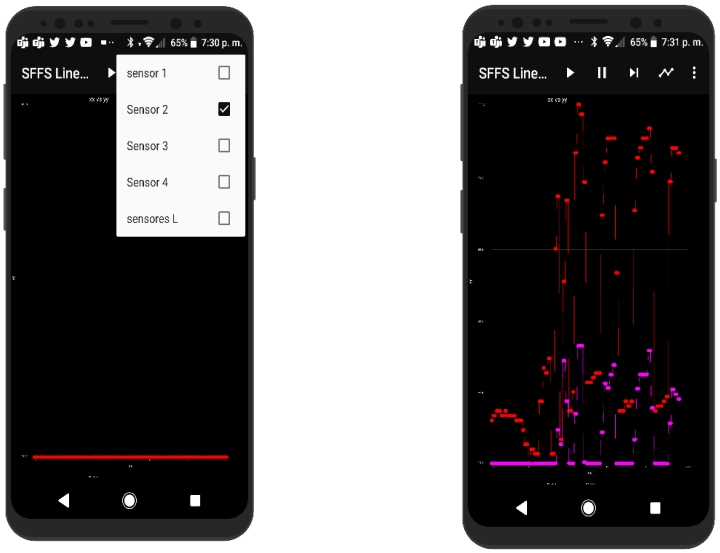
Cuando el usuario comienza un paso con el pie izquierdo, por ejemplo, el zapato de SFFS está en una posición como la mostrada en la Ilustración 11c. En ese caso se muestra en la APP el punto del talón, correspondiente al sensor 4 del pie izquierdo, de color rojo (presión máxima) y los demás sensores de ese zapato en color violeta (presión mínima). A medida que el pie cambia a la posición mostrada en la Ilustración 11d el sensor 4 cambiará de color hasta el violeta, pues la

[8] Librería disponible en el repositorio: https://github.com/ybq/Android-SpinKit

presión ejercida sobre él se irá reduciendo y en los otros sensores del zapato la tonalidad irá cambiando hasta rojo a medida que la presión en ellos aumenta.

3.4.2 Realimentación por graficas lineales (SFFS Line Graphs)

SFFS Line Graphs permite al usuario visualizar individualmente cada sensor mediante graficas lineales en plano dinámico de escala automática, con una resolución de 6 bits y un tiempo de muestreo de 200 ms, que indica con mayor precisión la presión ejercida sobre cada uno de los sensores FSR como se muestra en la Ilustración 22.



(a) (b)

Ilustración 22. Imagen presentada al usuario como realimentación de las gráficas lineales generadas por la presión medida por las FSR. a) Gráfica producida por un solo sensor. b) Gráfica producida por dos sensores mostrados de forma simultánea.

Para realizar SFFS *Line Graphs* se usó la librería *jplot-android*[9] *Johnspice* (2021) que permite realizar gráficos dinámicos. Los datos recibidos por Bluetooth® son procesados y

[9] Librería disponible en el repositorio: https://github.com/johnspice/jplot-android

almacenados en matrices con longitudes de 4000 bytes llamados Y, Y1, Y2, Y3 y Y4. Utilizando el método *plot* perteneciente a la librería *jplot* se grafican los datos en la UI, este método recibe dos parámetros de entrada, X (datos generados por un contador cada 200 milisegundos) y que son las matrices mencionadas anteriormente. Un algoritmo de selección identifica los datos pertenecientes a los FSRs del zapato izquierdo o derecho (ver Anexo 7 y 9).

Para este modo de realimentación visual el usuario puede seleccionar la visualización de uno o más sensores de forma simultánea como se muestra en la Ilustración 22a. Para la gráfica lineal mostrada al usuario, el eje vertical representa el peso al que está siendo sometido el sensor en una escala entre 0 y 40Kg; el eje horizontal se irá desplazando cada 200 milisegundos. Como ejemplo en la Ilustración 22b se muestra los resultados obtenidos al graficar el sensor 1 en verde, el sensor 2 en rojo, el sensor 3 en violeta y el sensor 4 en amarillo del zapato derecho cuando un voluntario caminaba usando los zapatos de SFFS, la última opción al ser seleccionada graficará los sensores del zapato izquierdo.

3.4.3 Realimentación por ángulos de inclinación (3D SFFS)

En 3D SFFS se presenta una interfaz gráfica con imágenes del plano transversal y coronal de los zapatos. Con esta realimentación el usuario puede observar la inclinación de sus pies mientras camina y de esa forma obtener información de como posiciona los pies a cada paso (ver Ilustración 23). La información mostrada es obtenida del cálculo de los ángulos con el filtro complementario aplicado a los datos capturados por el acelerómetro y el giroscopio de la IMU.

Ilustración 23. Imagen de la interfaz gráfica presentada al usuario al activar en la APP la realimentación por ángulos de inclinación.

Para poder realizar 3D SFFS, se aprovecharon los mecanismos que Android proporciona para crear animaciones simples. En este caso se hizo una animación interpolada, que realiza una transformación de rotación basada en la librería *animation* que contiene una extensión llamada *RotateAnimation* y permite controlar el giro de un objeto. El constructor *RotateAnimation,* recibe cuatro parámetros de entrada obtenidos de la IMU por Bluetooth® y permite el desplazamiento del inicio y el final de la animación en las coordenadas X y Y sobre el que gira el objeto (Anexo 8 y 9).

3.5 Realimentación auditiva

Desde el siglo XIX y hasta la fecha se tiene registro, ya sea en partituras como Franz Franz Berwald - Wettlauf - Etude for orchestra o en registro auditivo como el icónico *We Are The Champions de Queen*, de cómo la música es determinante para motivar, preparar y acompañar a quienes participan en los eventos deportivos como se menciona en la investigación desarrollada por Bateman y Bale (2009). Con los resultados obtenidos concluyeron que el uso de la música en

contextos deportivos trae múltiples beneficios para los deportistas al mejorar el estado psicológico en los momentos previos y en la ejecución de actividad física o deportiva.

La Ilustración 24 muestra un esquema donde es posible observar que hay factores internos y externos que afectan de forma positiva el estado psicológico de los deportistas al realizar actividad física. Si se ordenan de forma jerárquica, el ritmo y la musicalidad son determinantes al modificar la percepción que tiene el usuario de su propio cuerpo y la acción realizada, como se demostró en Tajadura-Jiménez et al. (2015) y Newbold et al. (2016). Estas investigaciones, que analizan desde el punto de vista de la neurociencia la influencia de la música en la percepción del cuerpo durante la actividad y rehabilitación física, resaltan la importancia de utilizar sonidos relacionados al movimiento del usuario como un método de realimentación para sistemas como SFFS. El término usado para ello en inglés es *Sonification,* que sería algo como sonificación en español y se refiere a la asignación de sonidos a los movimientos. En adelante se usará este término adaptado al español para facilitar la escritura.

En ese mismo sentido Bateman y Bale (2009) aportan pruebas de cómo el tempo de la música tiene una estrecha relación con el ritmo cardiaco de los deportistas y la sincronía de estos elementos puede aumentar el rendimiento en actividades repetitivas como pedalear o correr. Una adaptación de los resultados obtenidos en la investigación se muestra en la Ilustración 24.

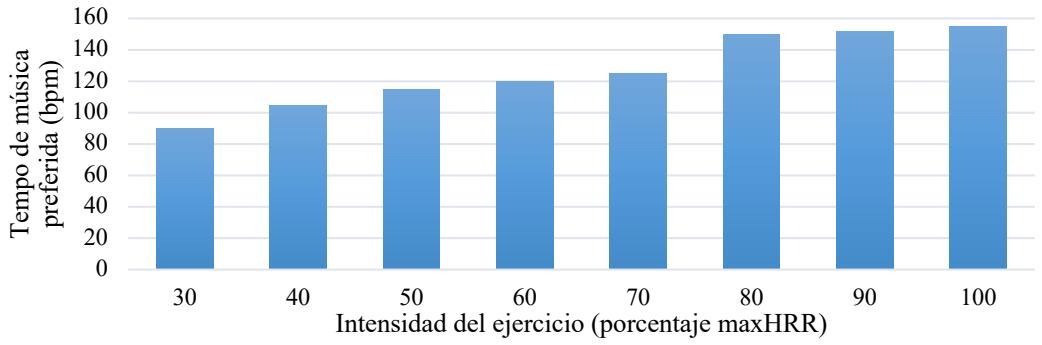


Ilustración 24. Relación entre intensidad del ejercicio en porcentaje maxHRR (Máxima Reserva del ritmo cardíaco) y tempo de la música preferido (Bateman y Bale, 2009).

Basados en los resultados mostrados en la Ilustración 25 en SFFS se decidió realimentar al usuario con tres tipos de respuesta auditiva utilizando un reproductor mp3 y para ello se preseleccionaron 20 canciones de acceso libre para propósitos no comerciales con licencia de *Blanket Barricade*®, de diversos estilos musicales y que se encuentran dentro de la franja de 60 a 160 bpm. Se diseñó y aplicó una encuesta a 87 personas entre 16 y 42 años para determinar las canciones que serían incluidas en el reproductor integrado en la APP de SFFS. En la encuesta, a la que se puede acceder desde el anexo 4, se le permitió escuchar a los participantes un fragmento de 30 segundos de las 20 canciones y se solicitó escoger nueve con las que se sintieran más a gusto para caminar, trotar y correr. Las canciones seleccionadas por los encuestados se relacionan en la Tabla 3.

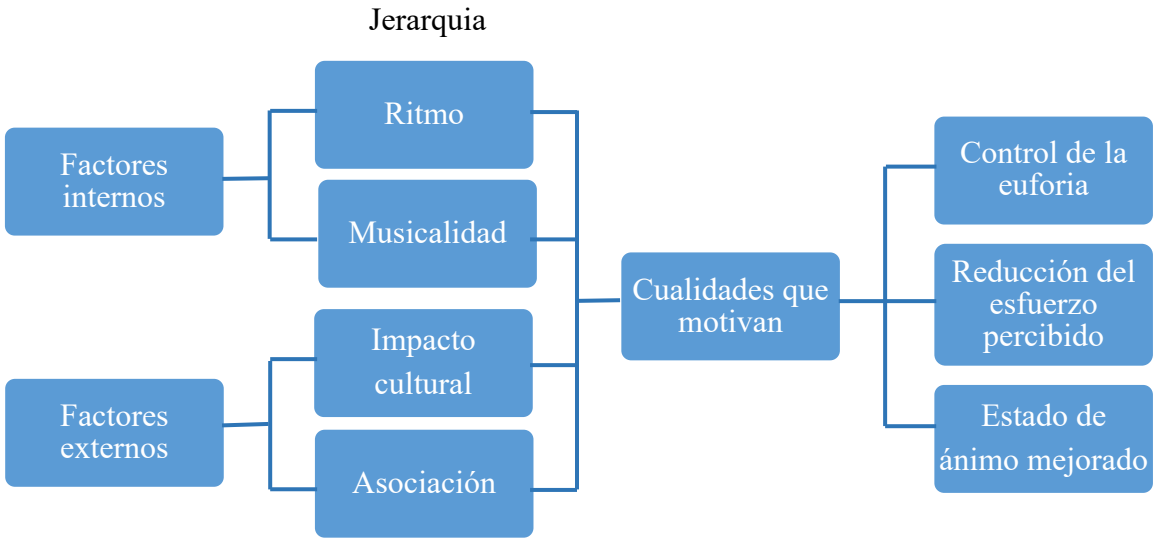


Ilustración 25. Adaptación de los resultados obtenidos por Bateman y Bale (2009) de la relación existente entre la música y la motivación al realizar actividad física.

Ahora bien, SFFS no se limita a las canciones seleccionas en este proyecto, dado que también es posible incluir desde la APP las canciones que el usuario tenga a disposición en su

dispositivo y en ese caso los derechos de autor de la música serán responsabilidad del servicio de *"streaming"* [10] que se use.

Tabla 3. Canciones seleccionadas para la realimentación auditiva, resultado de la encuesta aplicada.

Canción	Género	Tempo (bpm)	Autor
Slow Burn	Instrumental	60	Seattle
SoloAcousticBlues	Blues	75	Jason Shaw
Good Soldiers Don't Cry	Clásica	80	Pr. Mexico City
Blue Jacket	Blues	90	Seattle
HotSalsa	Salsa	100	Jason Shaw
Inside The Box	Rock	116	Seattle
We Fit Together	Instrumental	127	Birmingham
BigCarTheft	Electro	135	Jason Shaw
Sk8board	Electro	160	Jason Shaw

3.5.1 Realimentación por ecualización. (SFFS Music Equalizer)

La musicalidad, entendida como los diferentes factores de una canción que permiten que suene agradable tales como la calidad, el timbre o la armonía de los instrumentos, es el segundo elemento determinante en la jerarquía de los factores internos que repercuten en la psicología del rendimiento deportivo como se mostró en la Ilustración 24. SFFS Music Equalizer hace uso de dicha importancia y realimenta al usuario modificando cuatro anchos de banda de una canción que el usuario esté escuchando, dependiendo de la forma en que camina al usar los zapatos de SFFS.

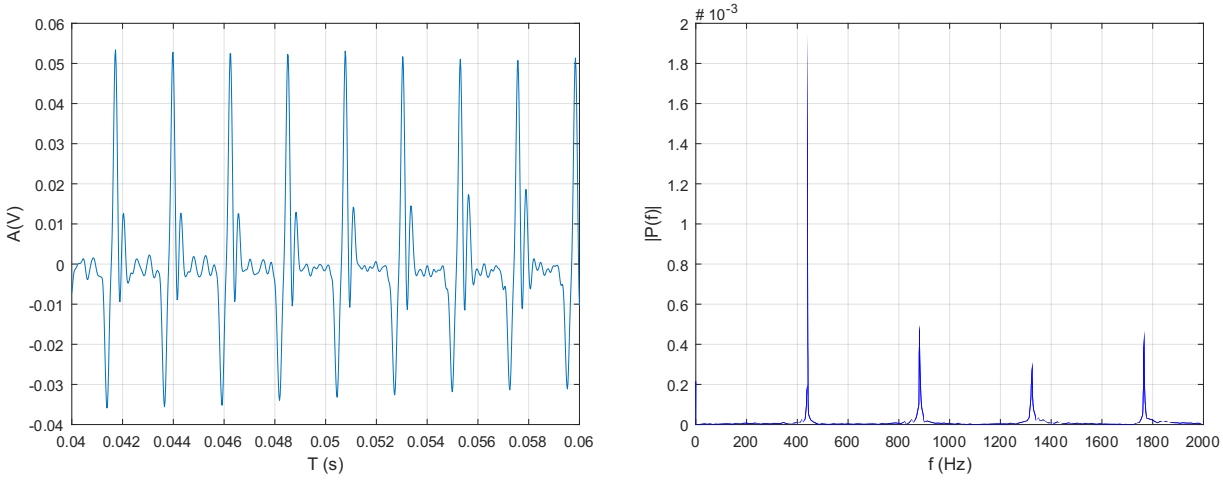
Para explicar de manera más detallada esto se toma como referencia una nota musical, que corresponde a una vibración de 440 Hz producida por una cuerda de guitarra llamada comúnmente

[10] El Streaming entendido como la reproducción de contenido multimedia a través de internet en dispositivos móviles usando plataformas como Spotify®, Deezer® o Apple Music®, como ejemplos de bibliotecas musicales.

La-440. Este sonido produce una gráfica en el dominio temporal que se muestra en la Ilustración 26a. Si a la función que describe esta nota se le aplica la transformada discreta de Fourier expresada en la Ecuación 8, se obtiene la gráfica mostrada en la Ilustración 26b.

$$F(n) = \sum_{K=1}^{N} f(t_k) e^{-j\frac{2\pi n}{N}(k-1)}, para\ 1 \leq n \leq N \qquad (8)$$

Como se puede apreciar en la Ilustración 26b, La-440 tiene una mayor concentración de potencia en la frecuencia de 440 Hz, pero también es interesante ver que esa nota tiene armónicos en los múltiplos de 440 Hz es decir 880 Hz, 1320 Hz y así sucesivamente. Teniendo en cuenta lo anterior, se aplica un filtro digital a 1320 Hz para dejar dos armónicos y eliminar las frecuencias mayores a 1320 Hz como se puede apreciar en la ilustración 27. Siguiendo este principio es posible modificar las diferentes frecuencias de los sonidos en una canción y mediante la programación de varios filtros digitales que en su conjunto forman un ecualizador.



(a) (b)

Ilustración 26. Gráficas de la nota La-440 producida en una guitarra. (a) Gráfica de La-440 en el dominio del tiempo. (b) Gráfica de La-440 en el dominio de la frecuencia.

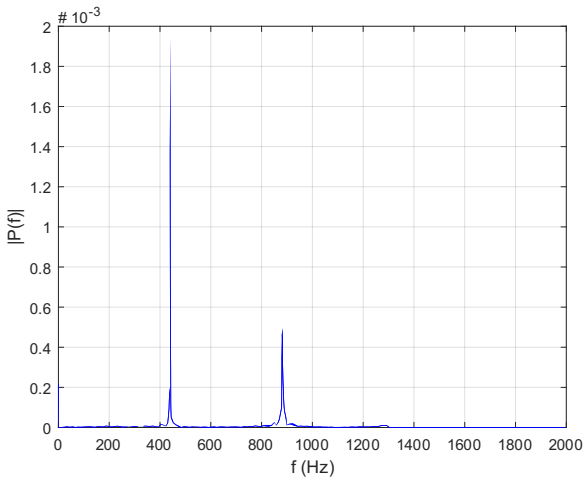


Ilustración 27. Gráfica de La-440 en el dominio de la frecuencia con un filtro pasa bajos a 1300 Hz.

En SFFS se propone la posibilidad de controlar la ecualización con los movimientos que el usuario hace mientras usa los zapatos y generar una sonificación relacionada a los valores de presión capturados por las FSR. Con SFFS Music Equalizer el usuario recibe realimentación auditiva al escuchar como las bandas de frecuencia de la canción se modifican mientras él camina o trota y con esto se mantiene informado de forma constante sobre los movimientos que realiza sin que deba interrumpir abruptamente la actividad física.

Para programar el ecualizador en *Android Studio* se hizo usó de la clase *AudioEffect*[11] que permite controlar los efectos de audio proporcionados por el entorno de desarrollo de audio de Android®. Usando la librería *Equalizer*[12], una extensión de la clase AudioEffect, se puede alterar la respuesta en frecuencia de archivos de audio y de esa manera se ecualiza una instancia de *MediaPlayer,* una clase usada para controlar la reproducción. Mediante el método *setBandLevel,* que recibe dos parámetros de entrada: el número de la banda que corresponde a un rango de

[11] Documentación de la clase disponible en: https://developer.android.com/reference/android/media/audiofx/AudioEffect

[12] Documentación de la librería disponible en: https://developer.android.com/reference/android/media/audiofx/Equalizer

frecuencias y la ganancia en decibeles, se puede alterar la respuesta en frecuencia de cuatro anchos de banda, 60Hz - 230Hz, 230Hz - 910Hz, 910 - 3600hz y 3600Hz - 1500 Hz, entre -15 dB a 15 dB (Anexo 6). e.g. Estas bandas se pueden modificar con combinaciones diferentes de los FSR, por ejemplo, si solo se presiona el sensor 3 (ver Ilustración 28) puede atenuarse los dos anchos de banda de frecuencias más bajas en -15dB. Dependiendo de la aplicación específica para la que se use SFFS se podría informar al usuario que está realizando un movimiento indebido o potencialmente lesivo, él notaría una pérdida significativa de sonidos bajos en la reproducción de audio (calidad y sería un indicador de que está realizando un movimiento indebido.

Ilustración 28. Imagen de realimentación entregada al usuario en la APP al hacer una posición donde solo se ejerce presión en el sensor 1 del pie izquierdo.

3.5.2 Realimentación por bpm (SFFS Music bpm Select).

Bateman y Bale (2009) aportan pruebas de como el ritmo y el tempo son los elementos de la música con más probabilidades de provocar una reacción física en el oyente. Así mismo, la evidencia disponible sugiere que la música sincronizada, se puede aplicar al rendimiento de resistencia aeróbica y anaeróbica en atletas que no son de élite, logrando producir efectos

psicológicos y psicofísicos positivos. Por otra parte, Schaffert (2019) aporta evidencia de como la estimulación auditiva rítmica, tiene incidencia en la recuperación, entrenamiento y mejoramiento de actividades deportivas o cotidianas relacionadas con la marcha. SFFS toma como referencia la importancia del sincronismo y estimulación auditiva rítmica, ofreciendo al usuario música que se adapta con los pasos en un margen de 10 bpm.

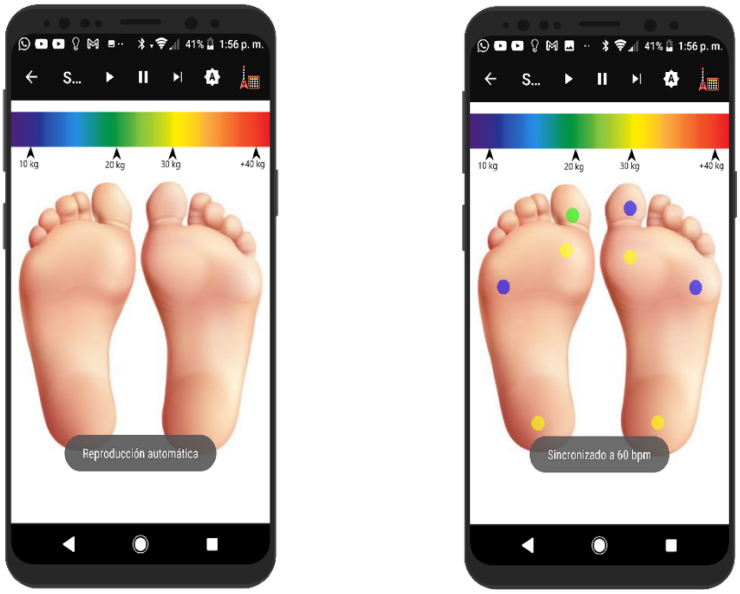
Para conseguir que las canciones reproducidas estuvieran en sincronía con los pasos, se hizo que un voluntario adulto de 30 años caminara usando los zapatos mientras escuchaba un sonido a 60bpm e intentara igualar el ritmo que oía. Se tomaron 200 muestras del tiempo entre cada paso y se calculó el tiempo promedio en milisegundos (ms). Este mismo procedimiento se realizó haciendo que el usuario escuchara sonidos con tempo entre 60 y 160bpm, los resultados obtenidos se relacionan en la tabla 4.

Tabla 4. Resultados obtenidos en las pruebas para calibrar la selección de canciones en SFFS Music bpm Select.

Tempo(bpm)	**Tiempo máximo(ms)**	**Tiempo mínimo(ms)**	**Tiempo promedio(ms)**
60	1968	1746	1822.4
70	1675	1464	1553.8
80	1475	1219	1349.9
90	1289	1069	1137.6
100	1143	958	1046.2
110	977	845	904.9
120	895	766	814.7
130	842	725	754.5
140	741	648	674.3
150	682	569	614.1
160	648	497	551.8

Dependiendo de los datos recolectados por el sensor 2 (ver Ilustración 7) ubicado en el metatarso del zapato derecho, el sistema detecta la frecuencia de los pasos usando el MCU que

envía dicha información vía Bluetooth a la APP encargada de seleccionar una canción con una frecuencia similar. Lo anterior se logró gracias a una función que se encarga de la selección, luego reproduce la canción utilizando la librería *MediaPlayer* y los métodos start(), pause() y stop() usados para reproducir, pausar o finalizar el objeto *MediaPlayer* (Anexo 6) e.g. Si por ejemplo el usuario camina a una frecuencia de 55bpm, el sistema está en la capacidad de detectarlo y reproducir una canción con un tempo similar que en este caso serian 60bpm. La imagen mostrada al usuario que acompaña la realimentación auditiva se muestra en la Ilustración 29.



a) b)

Ilustración 29. Imágenes mostradas al usuario en la APP cuando se utiliza la realimentación *SFFS Music bpm Select*. a) Imagen mostrada con reproducción automática. b) Imagen mostrada con la reproducción de una canción a 60bpm.

La Ilustración 29a muestra la imagen que el usuario vería al iniciar el modo de realimentación SFFS Music bpm Select que comenzaría con una canción automática. Una vez que se detecta presión (ver Ilustración 29b) el MCU detecta la frecuencia a la que está caminando el usuario y reproduce una canción con un tempo similar.

3.5.3 Realimentación por instrumentos (SFFS Music Band)

SFFS Music Band ofrece al usuario una realimentación valiéndose de la musicalidad de las canciones, variando el volumen de cada instrumento que la compone. Cuando una banda realiza la grabación de una canción, por lo general graba uno a uno todos los instrumentos, si se tiene acceso a la grabación original de la canción, es posible tener las pistas de audio de cada instrumento separadas y variar el volumen de cada una por aparte, además se puede sobreponer todas pistas y completar la canción.

SFFS se vale de la disponibilidad de las grabaciones originales de algunas canciones y de manera similar que con SFFS Equalizer, el sistema a través de los sensores detecta un movimiento indebido y alerta al usuario quitando el sonido de un instrumento, varios o silenciado por completo la canción, sin interrumpir abruptamente la actividad física. Esto se logró al usar tres objetos de la clase MediaPlayer, cada uno reproduce un instrumento, que serían voz, melódicos y batería. Con la ayuda del método setVolume() se controla el volumen de cada objeto (Anexo 6). e.g. Cuando se selecciona en la APP la realimentación SFFS Music Band el MCU irá variando el volumen de los instrumentos de la canción reproducida dependiendo de la presión ejercida en las FSR y la inclinación de la IMU a medida que el usuario camina. Es posible configurar por código los sensores específicos que afectan el volumen de cada instrumento. En SFFS el sensor 4 ubicado en el talón controla la batería, el sensor 2 localizado en el metatarso la voz y el sensor 3 de la parte media del pie los melódicos de la canción. Si el usuario está parado generando la misma presión en todos los sensores, escuchará la canción de forma normal (con el volumen de cada instrumento como en la versión original), al comenzar a caminar notará que uno o varios instrumentos de la canción dejan de sonar.

4. Resultados y conclusiones

4.1 Discusión y resultados

Después de las pruebas realizadas se observó que el sensor de presión ubicado cerca del hallux (dedo gordo o primer dedo del pie) no genera valores representativos en comparación a los obtenidos por el segundo sensor, por esta razón es posible prescindir de él en algunas aplicaciones puntuales y generar un diseño utilizando solo tres FSR. Además, se percibió que el uso de una IMU es determinante sólo para aplicaciones donde la inclinación del pie es realmente importante e.g. marcha atlética o halterofilia, pues sólo con los datos de presión es posible extraer suficiente información para la realimentación. Por ello se recomienda probar el diseño propuesto en este trabajo utilizando un prototipo sin IMU y solo con los sensores dos, tres y cuatro, con la finalidad de implementar una versión más sencilla y económica (SFFS Lite).

Para la comunicación inalámbrica entre las ESP32 y la APP se recomienda el uso de comunicación Wi-Fi dado que pueden encontrarse varias limitaciones implementando múltiples conexiones con Bluetooth® 4.2 del teléfono inteligente. Para sistemas en tiempo real que requieran lectura de sensores y gestión de datos como el monitoreo de los dos pies en simultaneo, se recomienda hacer uso de tarjetas multi-núcleo como la utilizada en este proyecto, porque al no contar con los núcleos suficientes es indispensable crear algoritmos o hacer uso de recursos que simulen tareas en paralelo.

Para la programación de una APP, en el sistema operativo Android®, se recomienda hacer uso de hilos (operaciones, lectura de sensores, reproductor musical, etc.) que funcionen de manera paralela a la ejecución principal que controla la parte gráfica de la interfaz de usuario, o de lo contrario se experimentará “lentitud” en la APP y la realimentación no tendrá un tiempo de respuesta lo suficientemente rápido para que sea considerado un RTS.

SFFS es una herramienta con gran potencial para el monitoreo de actividades como la rehabilitación de lesiones lumbares o en extremidades inferiores, ya que permite obtener datos importantes de la presión, la inclinación y la posición de los pies. Con esta información es posible controlar el esfuerzo de un usuario al realizar un ejercicio de recuperación como plantearon Newbold et al. (2016), pero aplicado a las extremidades inferiores. En este proyecto se realizaron diversas pruebas para evidenciar el correcto funcionamiento del sistema, sin embargo, es importante extender las pruebas con un mayor número de personas que permitan estudiar el comportamiento de los métodos de realimentación auditiva y se determine la efectividad de cada uno en actividad física libre y controlada según sea la necesidad especifica. También es posible usar SFFS en aplicaciones de neurociencia como las trabajadas por Tajadura-Jiménez et al. (2015) para mejorar la percepción del propio cuerpo y con ello contribuir en la promoción de hábitos saludables, aprovechando los diferentes métodos de realimentación auditiva basados en la sonificación interactiva del movimiento. Sobre lo anterior es importante resaltar que en este proyecto se utilizó música y no solo sonidos para estimular al usuario. Razón por la cual es importante investigar cómo se modifica la reacción y el comportamiento de los usuarios al escuchar canciones y no solo tonos simples.

4.2 Conclusiones

Dada la importancia que existe en la promoción de hábitos saludables para las personas y la posibilidad de aumentar la motivación que tiene la música al mejorar el estado psicológico en los momentos previos y en la ejecución de actividad física o deportiva, se considera importante este trabajo al abrir una invitación a la comunidad académica de Bogotá y Colombia para desarrollar proyectos de implementación de sistemas de realimentación o sonificación interactiva del movimiento, utilizando el calzado como una WT. Para aplicaciones en situaciones reales de

salud que puedan afectar la calidad de vida de los colombianos, donde la recopilación de datos asociados a los hábitos de las personas al caminar o características de postura y marcha cuando se realiza actividad física, sistemas como SFFS son de gran utilidad al entregar realimentación visual y/o auditiva constante de forma inalámbrica a los usuarios y permitirle a médicos, terapeutas o entrenadores físicos utilizar la información en aplicaciones de rehabilitación con tecnologías relacionadas con IoHT y neurociencia.

Inicialmente se propuso el desarrollo de un sistema electrónico para realimentación y monitoreo de movimientos en extremidades inferiores, no obstante, se logró construir un prototipo de dispositivo adaptado al calzado que cuenta con MCUs, baterías y plantillas con sensores resistivos de presión y una unidad de medición inercial que no incomodan al usuario cuando usa los zapatos y no afectan la fluidez de los movimientos de la actividad física realizada ya que la comunicación entre los zapatos y la APP es inalámbrica. Gracias al avance de las tecnologías en placas de desarrollo como la ESP32 utilizada en este trabajo y las reuniones realizadas con investigadoras internacionales con conocimiento del tema, que permitieron analizar y parametrizar los datos obtenidos para definir el tipo de respuesta para realimentar al usuario, SFFS cuenta con tres tipos diferentes de realimentación visual en tiempo real que entregan información relevante sobre la presión ejercida en cada sensor por cada pie y la inclinación cuando se usan los zapatos. Además, se desarrollaron tres tipos de realimentación auditiva que modifican la musicalidad de las canciones que el usuario oye mientras utiliza el sistema y con ello se contribuye en el trabajo de sonificación interactiva al permitirle al usuario percibir cómo sus movimientos afectan lo que está escuchando.

La información en SFFS es presentada de manera visual y auditiva al usuario de forma constante mientras se usan los zapatos. Sin embargo, el sistema puede ser mejorado al programar

la APP para que almacene el último tramo de datos en la memoria del teléfono y también se sincronice con una base de datos en línea. De esta manera la información puede ser analizada por el usuario o en contextos como la telemedicina por médicos, terapeutas o entrenadores físicos. El prototipo actual no pretende determinar cuáles posiciones de los pies pueden ser lesivas para el usuario o afectan directamente la actividad física; sin embargo, el sistema está diseñado para modificar la musicalidad de las canciones con combinaciones predeterminadas de los sensores de presión. En ese sentido se propone la reprogramación desde la APP de las variables que activan las diferentes realimentaciones auditivas, teniendo en cuenta la aplicación específica para la que se vaya a implementar, pues sistemas como SFFS pueden llegar a remplazar o complementar tecnologías de medición y monitoreo existentes, para detección plantar y posicionamiento del pie.

Hasta la fecha varios autores como Bateman y Bale (2009) o Newbold et al. (2016), han documentado cómo la música puede ser utilizada en contextos deportivos y terapéuticos. Por otra parte, investigaciones como la desarrollada por Tajadura-Jiménez et al. (2015) han diseñado dispositivos portátiles basados en sonificación del movimiento que se enriquecerían con tecnologías de realimentación en tiempo real. En una aproximación sobre este asunto, sistemas como los desarrollados por Hegde, Bries y Sazonov (2016) o de Fazio et al. (2021) ofrecen tecnologías que pretenden crear un puente entre los datos obtenidos por los sensores y la información que le llega al usuario. SFFS ofrece un punto de avance importante al enlazar estos dos campos de investigación, brindando un instrumento que utiliza los datos recolectados por diferentes sensores integrados al calzado y entrega a los usuarios, de forma inalámbrica, información relevante con tres tipos de realimentación auditiva basados en la sonificación interactiva del movimiento en extremidades inferiores.

5. Bibliografía

Acar, C. y Shkel, A. (2009). MEMS Vibratory Gyroscopes Structural Approaches to Improve Robustness. 1ª. Ed. New York: *Springer*. ISBN 978-0-09535-6

Bateman, A. y Bale, J. (Eds.). (2009). Sporting Sounds: Relationships Between Sport and Music, EE. UU. Y Canadá: Editorial Routledge

Burns, A. y Wellings A. (2003). Sistemas de Tiempo Real y Leguajes de Programación 3ª. Ed. Madrid: *Addison Wesley*. ISBN: 84-7829-058-3

Carbonaro, N., Lorussi, F. & Tognetti, A. (2016). Assessment of a Smart Sensing Shoe for Gait Phase Detection in Level Walking. *Electronics* 5(4):78. doi: 10.3390/electronics5040078

Chiariglione, L. (2019). 7 MPEG. Recuperado de https://mpeg.chiariglione.org

Collins English Dictionary 12th edition, (2014). HarperCollins Publishers. Recuperado de https://www.collinsdictionary.com/dictionary ISBN: 978-0007522743

Daintith, J. y Wright, E. (2008). A Dicctionary of Computing. Recuperado de https://www.oxfordreference.com doi: 10.1093/acref/9780199234004.001.0001

De Fazio, R., Perrone, E., Velázquez, R., De Vittorio, M., Visconti, P. (2021). Development of a Self-Powered Piezo-Resistive Smart Insole Equipped with Low-Power BLE Connectivity for Remote Gait Monitoring. *Sensors* 21, 4539. doi: 10.3390/s21134539

Dorf, R. y Svoboda, J. (2006). Circuitos Eléctricos 6ª edición. Alfaomega Grupo Editor. México. ISBN: 970-15-1098-4

Eskofier, B., Lee, S., Baron, M., Simon, A., Martindale, C., Gaßner, H. & Klucken, J. (2017). An Overview of Smart Shoes in the Internet of Health Things : Gait and Mobility Assessment in Health Promotion and Disease Monitoring. *Applied Sciences.* doi: 10.3390/app7100986

Gardelli, D., (1999) A origem da inércia. *Caderno Brasileiro de Ensino de Física*, v. 16, n. 1, p.

43-53. ISBN 2175-7941.

Google LLC (2021). Developer. https://developer.android.com

Hegde, N., Bries, M. & Sazonov, E. (2016). A comparative review of footwear-based wearable systems. *Electronics (Switzerland)* 5(3). doi:10.3390/electronics5030048

Interlink Electronics Inc (11 de marzo de 2021). Serie FSR® 400. https://www.interlinkelectronics.com/fsr-400-series

Jhr, J (2020). Bluetoothjhr. https://github.com/jose-jhr/-bluetoothjhr

Johnspice (2021). Jplot-android. https://github.com/johnspice/jplot-android

Kim, J., Lee, K. y Hong, S. (2018). Random forest based-biometric identification using smart shoes. En *International Conference on Sensing Technology, ICST* 2017-Diciembre:1–4. Sydney, Australia. doi: 10.1109/ICSensT.2017.8304518

Kuo, B. (1996). Sistemas de Control Automático 7ª. Ed. México: *Prentice Hall*. ISBN: 968-880-723-0

Newbold, J., Bianchi-Berthouze, N., Gold, N., Tajadura-Jiménez, A., Williams, A. (2016) Musically Informed Sonification for Chronic Pain Rehabilitation: Facilitating Progress & Avoiding Over-Doing. En *2016 CHI Conference on Human Factors in Computing Systems*. Mayo 5698–5703. doi: 10.1145/2858036.2858302

Ogata, K. (2010). Ingeniería de Control Moderna 5ª edición. Pearson Educación, S.A. Madrid. ISBN: 978-84-8322-660-5

Ohnishi, A., Terada T. y Tsukamoto M. (2018) A Motion Recognition Method Using Foot Pressure Sensors. En AH2018: The 9th Augmented Human International Conference, febrero 7–9 de 2018, Seoul, República de Korea. ACM, New York, NY, USA. doi: 10.1145/3174910.3174938

Oppenheim, A., Willsky, A. y Nawab, S. (1998) Señales y sistemas 2ª edición. Pearson Educación ISBN: 9789701701164

Oxford Dictionary of Computing (2008), Oxford University Press Print Publication. ISBN: 9780191726576. DOI: 10.1093/acref/9780199234004.001.0001

Pasluosta, C., Gaßner, H., Winkler, J., Klucken, J. & Eskofier, B. (2015). An emerging era in the management of parkinson's disease: Wearable technologies and the internet of things. *IEEE Journal of Biomedical and Health Informatics* 19, 9. doi: 10.1109/JBHI.2015.2461555

Pozo, D. (2010). Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio. *Escuela Politécnica Nacional*, Quito.

Rodrigues, S. 2011. 138f. Tese (Doutorado em Engenharia Elétrica), Programa de Pós-Graduação em Engenharia Elétrica, Centro de Engenharia Elétrica e Informática, Universidade Federal de Campina Grande – Paraíba Brasil, 2011.

Saidani, S., Haddad, R., Mezghani, N. y Bouallegue, R. (2019). A survey on smart shoe insole systems. *2018 International Conference on Smart Communications and Networking (SmartNets)* 1–6. doi: 10.1109/smartnets.2018.8707391

Saraiva, F. (2014) Projeto de um sensor de pressão manométrica para ensino de física em tempo real. Caderno Brasileiro de Ensino de Física, Florianópolis, v. 31, n. 1, p. 124-148. ISSN 2175-7941. doi: 10.5007/2175-7941.2014v31n1p124

Tajadura-Jiménez, A., Basia, M., Deroy, O., Fairhurst, M., Marquardt, N. & Bianchi-berthouze, N. (2015). As Light as your Footsteps: Altering Walking Sounds to Change Perceived Body Weight, Emotional State and Gait. CHI '15 - *Proceedings of the 2015 CHI Conference on Human Factors in Computing Systems*, 2943–2952. doi:10.1145/2702123.2702374

Technology, C. (2004) Diccionario de Informatica E Internet: Computer and Internet Technology Definitions in Spanish. Boston Massachusetts: *Thomson*. ISBN: 0619267887, 9780619267889

Vignolo, J. (2008). Introducción al procesamiento de señales. Ediciones Universitarias de Valparaíso.

Wang, B., Rajput, K. S., Tam, W. K., Tung, A. K. H. & Yang, Z. (2015). FreeWalker: A smart insole for longitudinal gait analysis. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* 2015-Novem:3723–26. doi: 10.1109/EMBC.2015.7319202

Wendling, M. (2010) Sensores. *Universidade Estadual Paulista*. São Paulo.

Xu, W., Liu, J., Huang, M.-C., He, M.-C., Amini, N. y Sarrafzadeh, M. (2012) Smart insole: A wearable system for gait analysis. *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments.* V. 18, p. 1–4. doi: 10.1145/2413097.2413120

Ybq (2019). Android-SpinKit. https://github.com/ybq/Android-SpinKit

Young, S., (1982) Real Time Languages Desing and Developments', Ellis Horwood, USA.

Yu, T., Jin, H. y Nahrstedt, K. (2019). ShoesLoc: In-Shoe Force Sensor-Based Indoor Walking Path Tracking. Disponible en: ACM on Interactive, Mobile, Wearabe and Ubiquitous Technologies, 3(1), 1–23, 31. doi: 10.1145/3314418

6. Anexos

Anexo 1. Código SFFS para programación de la ESP32 del zapato izquierdo (ESPIZQ).

```
[1] //Librerías utilizadas en el código
[2] #include "BluetoothSerial.h"
[3] #include "Wire.h"
[4] #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
[5] #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
[6] #endif
[7] BluetoothSerial SerialBT;
[8] //Variables creadas para el contador BPM
[9] unsigned long tiempo1=0, tiempo2=0, diferenciaTiempo=0;
[10] int miArray[100];
[11] int BPM=60;
[12] //Definición de condiciones para funcionamiento de la IMU
[13] #define mpu 0x68
[14] #define a_r 16384.0
[15] #define g_r 131.0
[16] #define rad_deg 57.295779
[17] int16_t ax, ay, az, gx, gy, gz;//Variables para los datos de la IMU
[18] float acc[2], gir[3], ang[3];
[19] int imuX, imuY, imuZ;
[20] float senPre[4];//Variables para los sensores de presión
[21] double concatenado;
[22] String valor0="", valor1="",valor2="", valor3="", valor4="", valor5="", valor6="",valor7="";
[23] void setup() {
[24] SerialBT.begin("SFFSIZ"); //Nombramiento del dispositivo bluetooth
[25] //Definición de terminales analógicos
[26] Serial.begin(115200);
[27] analogReadResolution(12);
[28] analogSetAttenuation(ADC_11db);
[29] //Definición de los puertos de entrada de la IMU usando la librería Wire
[30] Wire.begin(21, 22);
[31] Wire.beginTransmission(mpu);
[32] Wire.write(0x6B);
[33] Wire.write(0);
[34] Wire.endTransmission(true);
[35] }
[36] void acelerometro() {
[37] //Definición de los parámetros de entrada para el acelerómetro
[38] Wire.beginTransmission(mpu);
[39] Wire.write(0x3B);
[40] Wire.endTransmission(false);
[41] Wire.requestFrom(mpu,6,true);
[42] ax = Wire.read()<<8|Wire.read();
[43] ay = Wire.read()<<8|Wire.read();
[44] az = Wire.read()<<8|Wire.read();
[45] //Cálculo de los ángulos de inclinación del pie
[46] acc[0] = atan((ay/a_r)/sqrt(pow((ax/a_r),2) + pow((az/a_r),2)))*rad_deg;
[47] acc[1] = atan(-1*(ax/a_r)/sqrt(pow((ay/a_r),2) + pow((az/a_r),2)))*rad_deg;
[48] }
[49] void giroscopio() {
[50] //Definición de los parámetros de entrada para el giroscópio
[51] Wire.beginTransmission(mpu);
[52] Wire.write(0x43);
[53] Wire.endTransmission(false);
[54] Wire.requestFrom(mpu,6,true);
[55] gx = Wire.read()<<8|Wire.read();
[56] gy = Wire.read()<<8|Wire.read();
[57] gz = Wire.read()<<8|Wire.read();
[58] gir[0] = gx/g_r;
[59] gir[1] = gy/g_r;
[60] gir[2] = gz/g_r;
[61] }
[62] void angulos() {
[63] acelerometro();
[64] giroscopio();
[65] //Aplicación del filtro complementario
```

```
[66] ang[0] = 0.98*(ang[0]+gir[0]*0.010) + 0.02*acc[0];
[67] ang[1] = 0.98*(ang[1]+gir[1]*0.010) + 0.02*acc[1];
[68] ang[2] = 0.98*(ang[2]+gir[2]*0.010);
[69] imuX=map(ang[1],0,360,10,99);
[70] imuY=map(ang[0],0,360,10,99);
[71] imuZ=map(ang[2],0,360,10,99);
[72] }
[73] //Lectura de los datos de la presión de las FSR402
[74] void sensores() {
[75] senPre[0]=map(analogRead(39),0,4095,10,99);
[76] senPre[1]=map(analogRead(36),0,4095,10,99);
[77] senPre[2]=map(analogRead(34),0,4095,10,99);
[78] senPre[3]=map(analogRead(35),0,4095,10,99);
[79] }
[80] void loop() {
[81] String a="";
[82] sensores();
[83] angulos();
[84] //Datos definitivos del zapato izquierdo para el envío por Bluetooth
[85] valor0=String(int(senPre[0]));
[86] valor1=String(int(senPre[1]));
[87] valor2=String(int(senPre[2]));
[88] valor3=String(int(senPre[3]));
[89] valor4=String(imuX);
[90] valor5=String(imuY);
[91] valor6=String(imuZ);
[92] valor7=String(BPM);
[93] Codificación del arreglo con los datos del zapato izquierdo
[94] a=valor0+valor1+valor2+valor3+"10"+"10";
[95] concatenado = a.toDouble();
[96] Serial.println(a);
[97] SerialBT.print(a);
[98] delay(900);
[99]}
```

Anexo 2. Código SFFS para programación de la ESP32 del zapato derecho (ESPDER).

```
[1] //Librerías utilizadas en el código
[2] #include "BluetoothSerial.h"
[3] #include "Wire.h"
[4] #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
[5] #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
[6] #endif
[7] #define RXD2 16
[8] #define TXD2 17
[9] TaskHandle_t Task2;
[11] BluetoothSerial SerialBT;
[12] //Configuración de bluetooth para recepción
[13] String MACadd = "24:6F:28:AE:0A:E6";
[14] String name = "SFFSIZ";
[15] char *pin = "1234"; //<- standard pin would be provided by default
[16] bool connected;
[17] int n=10;
[18] char rx[9];
[19] int i;
[20] //Configuración del contador para realimentación por BPM
[21] unsigned long tiempo1 = 0;
[22] unsigned long tiempo2 = 0;
[23] unsigned long diferenciaTiempo = 0;
[24] int miArray[100];
[25] int BPM=60;
[26] //Definición de condiciones para funcionamiento de la IMU
[27] #define mpu 0x68
[28] #define a_r 16384.0
[29] #define g_r 131.0
[30] #define rad_deg 57.295779
[31] int16_t ax, ay, az, gx, gy, gz;//datos del IMU
[32] float acc[2], gir[3], ang[3];
[33] int imuX, imuY, imuZ;
[34] float senPre[4];/Variables para los sensores de presión
[35] double concatenado;
[36] String valor0="", valor1="",valor2="", valor3="", valor4="", valor5="", valor6="",valor7="";
[37] String zapDer="", zapIzq="";
[38] void setup() {
[39] // Configuración del segundo nucleo del MCU
[40] xTaskCreatePinnedToCore(
[41] loop2,
[42] "Task_2",
[43] 1000,
[44] NULL,
[40] 1,
[46] &Task2,
[47] 0);
[48] SerialBT.begin("SFFS", true); //Nombramiento del dispositivo bluetooth
[49] connected = SerialBT.connect(name);
[50] //Definición de terminales analógicos
[51] Serial.begin(115200);
[52] analogReadResolution(12);
[53] Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
[54] analogSetAttenuation(ADC_11db);
[55] //Definición de los puertos de entrada de la IMU usando la librería Wire
[56] Wire.begin(21, 22);
[57] Wire.beginTransmission(mpu);
[58] Wire.write(0x6B);
[59] Wire.write(0);
[60] Wire.endTransmission(true);
[61] SerialBT.connect();
[62] }
[63] void acelerometro() {
[64] //Definición de los parámetros de entrada para el acelerómetro
[65] Wire.beginTransmission(mpu);
[66] Wire.write(0x3B);
[67] Wire.endTransmission(false);
[68] Wire.requestFrom(mpu,6,true);
[69] ax = Wire.read()<<8|Wire.read();
[70] ay = Wire.read()<<8|Wire.read();
[71] az = Wire.read()<<8|Wire.read();
```

```
[72] //Cálculo de los ángulos de inclinación del pie
[73] acc[0] = atan((ay/a_r)/sqrt(pow((ax/a_r),2) + pow((az/a_r),2)))*rad_deg;
[74] acc[1] = atan(-1*(ax/a_r)/sqrt(pow((ay/a_r),2) + pow((az/a_r),2)))*rad_deg;
[75] }
[76] void giroscopio() {
[77] //Definición de los parámetros de entrada para el giroscópio
[78] Wire.beginTransmission(mpu);
[79] Wire.write(0x43);
[80] Wire.endTransmission(false);
[81] Wire.requestFrom(mpu,6,true);
[82] gx = Wire.read()<<8|Wire.read();
[83] gy = Wire.read()<<8|Wire.read();
[84] gz = Wire.read()<<8|Wire.read();
[85] gir[0] = gx/g_r;
[86] gir[1] = gy/g_r;
[87] gir[2] = gz/g_r;
[88] }
[89] void angulos() {
[90] acelerometro();
[91] giroscopio();
[92] //Aplicación del filtro complementario
[93] ang[0] = 0.98*(ang[0]+gir[0]*0.010) + 0.02*acc[0];
[94] ang[1] = 0.98*(ang[1]+gir[1]*0.010) + 0.02*acc[1];
[95] ang[2] = 0.98*(ang[2]+gir[2]*0.010);
[96] imuX=map(ang[1],0,360,10,99);
[97] imuY=map(ang[0],0,360,10,99);
[98] imuZ=map(ang[2],0,360,10,99);
[99] }
[100] //Lectura de los datos de la presión de las FSR402
[101] void sensores() {
[102] senPre[0]=map(analogRead(36),0,4095,10,99);
[103] senPre[1]=map(analogRead(39),0,4095,10,99);
[104] senPre[2]=map(analogRead(34),0,4095,10,99);
[105] senPre[3]=map(analogRead(35),0,4095,10,99);
[106] }
[107] //Inicio del Loop del primer núcleo
[108] void loop() {
[109] sensores();
[110] angulos();
[111] //Datos definitivos del zapato derecho
[112] valor0=String(int(senPre[0]));
[113] valor1=String(int(senPre[1]));
[114] valor2=String(int(senPre[2]));
[115] valor3=String(int(senPre[3]));
[116] valor4=String(imuX);
[117] valor5=String(imuY);
[118] valor6=String(imuZ);
[119] valor7=String(BPM);
[120] //Codificación del arreglo con los datos del zapato derecho
[121] zapDer="1"+valor0+valor1+valor2+valor3+valor7+valor4+valor5;
[122] delay(200);
[123] Serial2.println(zapDer);
[124] delay(200);
[125] //Lectura del arreglo del zapato izquierdo
[126] for (int cont = 0; cont < n; cont++) {
[127] if (SerialBT.available()) {
[128] rx[i]=(char(SerialBT.read()));
[129] i++;
[130] if(i==n){
[131] zapIzq=(rx);
[132] i=0;}
[133] }
[134] }
[135] }
[136] //Inicio del Loop del segundo núcleo
[137] void loop2(void *parameter){
[138] //Ejecución de la realimentación por BPM
[139] for(;;){
[140] for (int i = 0; i < 100; i = i + 1) {
```

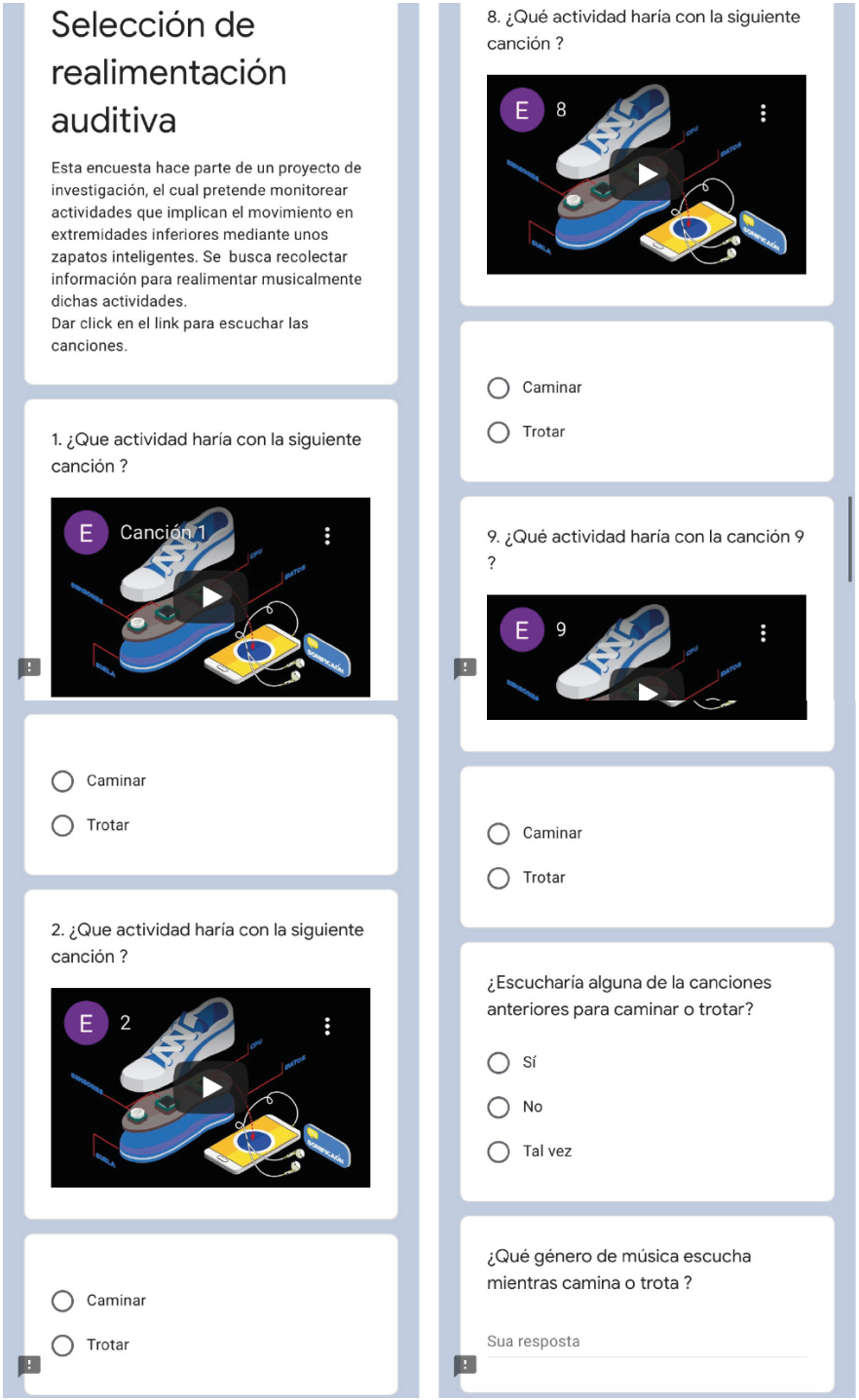
```
[141] if(analogRead(39)<1000){
[142] tiempo1=millis();
[143] }
[144] if(analogRead(39)>1000){
[145] tiempo2 = millis();
[146] diferenciaTiempo = tiempo2-tiempo1;
[147] miArray[i] = diferenciaTiempo;
[148] if (miArray[i]==0){
[149] if(miArray[i-1]> 100 ){
[150] if (miArray[i-1]>1680 & miArray[i-1]<10000 )
[151] BPM=10;//60Bbpm
[152] if (miArray[i-1]>1460 & miArray[i-1]<1680 )
[153] BPM=11;//70bpm
[154] if (miArray[i-1]>1210 & miArray[i-1]<1460 )
[155] BPM=12;//80bpm
[156] if (miArray[i-1]>1110 & miArray[i-1]<1210 )
[157] BPM=13;//90bpm
[158] if (miArray[i-1]>960 & miArray[i-1]<1110 )
[159] BPM=14;//100bpm
[160] if (miArray[i-1]>850 & miArray[i-1]<960 )
[161] BPM=15;//110bpm
[162] if (miArray[i-1]>760 & miArray[i-1]<850 )
[163] BPM=16;//120bpm
[164] if (miArray[i-1]>725 & miArray[i-1]<760 )
[165] BPM=17;//130bpm
[166] if (miArray[i-1]>650 & miArray[i-1]<725 )
[167] BPM=18;//140bpm
[168] if (miArray[i-1]>560 & miArray[i-1]<650 )
[169] BPM=19;//150bpm
[170] if (miArray[i-1]>490 & miArray[i-1]<560 )
[171] BPM=20;//169bpm
[172] if (miArray[i-1]<490 )
[173] BPM=21;//200bmp
[174] }
[175] }
[176] }
[177] }
[178] }
[179] vTaskDelay(10);
[180] }
```

Anexo 3. Código SFFS para programación de la ESP32 que transmite por Bluetooth (ESPBLU).

```
[1] #include "BluetoothSerial.h"
[2] #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
[3] #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
[4] #endif
[5] BluetoothSerial SerialBT;
[6] #define RXD2 16
[7] #define TXD2 17
[8] int n=17;
[9] char a[17];
[10] int i;
[11] void setup() {
[12] SerialBT.begin("Smart Footwear Feedback System");
[13] Serial.begin(115200);
[14] Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
[15] Serial.println("Serial Txd is on pin: "+String(TX));
[16] Serial.println("Serial Rxd is on pin: "+String(RX));
[17] }
[18] void loop() { //Choose Serial1 or Serial2 as required
[19] if (Serial2.available()) {
[20] a[i]=(char(Serial2.read()));
[21] i++;
[22] if(i==17){
[23] SerialBT.print(a);
[24] Serial.print(a);
[25] i=0;
[26] }
[27] }
[28] }
```

Anexo 4. Encuesta para la elección de las canciones para la realimentación auditiva y resultados obtenidos.

Encuesta realizada: https://docs.google.com/forms/d/e/1FAIpQLSe-T9FHAaDcITX15N8vrRNKEU4z0ykC_Pz_eo3I4Nduech1GQ/viewform?pli=1



Selección de realimentación auditiva

Esta encuesta hace parte de un proyecto de investigación, el cual pretende monitorear actividades que implican el movimiento en extremidades inferiores mediante unos zapatos inteligentes. Se busca recolectar información para realimentar musicalmente dichas actividades.
Dar click en el link para escuchar las canciones.

1. ¿Que actividad haría con la siguiente canción ?

Canción 1

Caminar
Trotar

2. ¿Que actividad haría con la siguiente canción ?

2

Caminar
Trotar

8. ¿Qué actividad haría con la siguiente canción ?

8

Caminar
Trotar

9. ¿Qué actividad haría con la canción 9 ?

9

Caminar
Trotar

¿Escucharía alguna de la canciones anteriores para caminar o trotar?

Sí
No
Tal vez

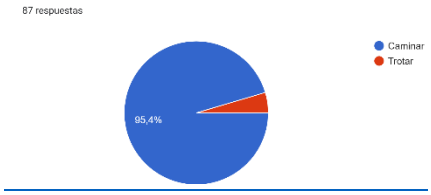
¿Qué género de música escucha mientras camina o trota ?

Sua resposta

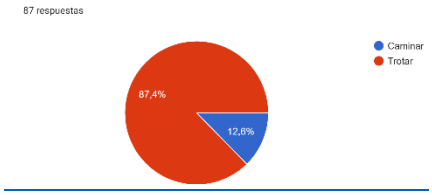
Resultado de la encuesta:

https://docs.google.com/forms/d/1jfBOHDiM7slabc0Ug7tGln7tbH8hi88hQIKhCn2c3ww/viewanalytics

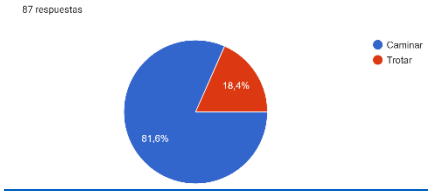
1. ¿Qué actividad haría con la canción 1?



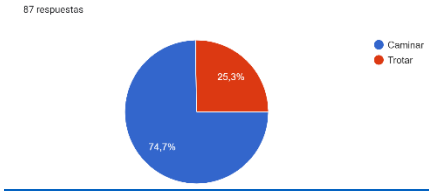
2. ¿Qué actividad haría con la canción 2?



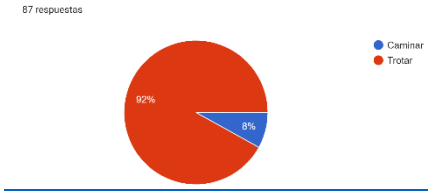
3. ¿Qué actividad haría con la canción 3?



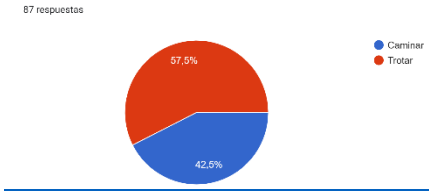
4. ¿Qué actividad haría con la canción 4?



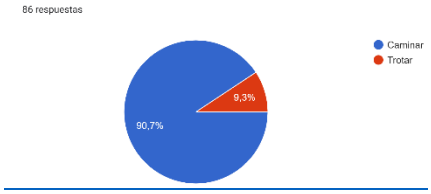
5. ¿Qué actividad haría con la canción 5?



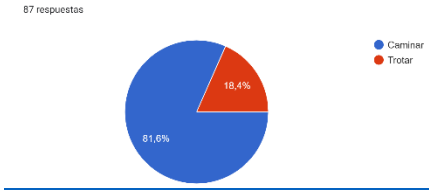
6. ¿Qué actividad haría con la canción 6?



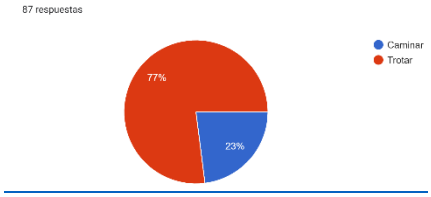
7. ¿Qué actividad haría con la canción 7?



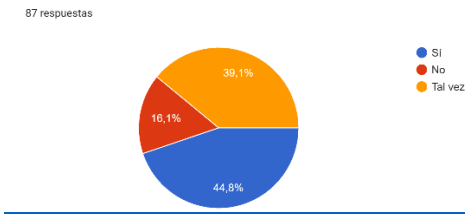
8. ¿Qué actividad haría con la canción 8?



9. ¿Qué actividad haría con la canción 9?



10. ¿Escucharía alguna de las canciones anteriores?



Anexo 5. Código SFFS para la conexión Bluetooth del dispositivo Android®.

```
[1]package com.example.SFFS;
[2]//Librerías utilizadas en el código
[3]import androidx.appcompat.app.AppCompatActivity;
[4]import android.os.Bundle;
[5]import android.view.View;
[6]import android.widget.AdapterView;
[7]import android.widget.ListView;
[8]import com.example.hp.bluetoothjhr.BluetoothJhr;
[9]public class MainActivity2 extends AppCompatActivity {
[10]ListView Lista;
[11]@Override
[12]protected void onCreate(Bundle savedInstanceState) {
[13]super.onCreate(savedInstanceState);
[14]setContentView(R.layout.activity_main2);
[15]Lista = (ListView) findViewById(R.id.Lista);
[16]final BluetoothJhr bluetoothJhr = new BluetoothJhr(this,Lista);
[17]bluetoothJhr.EncenderBluetooth();
[18]Lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
[19]@Override
[20]public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
[21]bluetoothJhr.Disp_Seleccionado(view,position,MainActivity4.class);
[22]}
[23]});
[24]}
[25]}
```

Anexo 6. Código SFFS para realimentación auditiva (SFFS Music Equalizer, SFFS Music Select bpm, SFFS Music Band) y realimentación visual (Heat Maps).

```
[1]package com.example.SFFS;
[2]//Librerías necesarias
[3]import
androidx.appcompat.app.AppCompatActivit
y;
[4]import android.graphics.Color;
[5]import android.media.MediaPlayer;
[6]import android.media.audiofx.Equalizer;
[7]import android.os.Bundle;
[8]import android.view.Menu;
[9]import android.view.MenuItem;
[10]import android.widget.Toast;
[11]import
com.example.hp.bluetoothjhr.BluetoothJhr;
[12]import
com.github.ybq.android.spinkit.SpinKitVie
w;
[13]import java.util.Timer;
[14]import java.util.TimerTask;
[15]public class MainActivity8 extends
AppCompatActivity {
[16]//Variables necesarias para la gestión de
datos
[17]BluetoothJhr bluetoothJhr;
[18]float i;
[19]double
ic,i0,i1L,i2L,i3L,i4L,i5L,i6L,i7L,i1R,i2R,i3
R,i4R,i5R,i6R,i7R;
[20]int[] valor ={0,0,0,0,0,0};
[21]SpinKitView
sensor1,sensor2,sensor3,sensor4,sensor5,sen
sor6,sensor7,sensor8;
[22]String Mensaje;
[23]//Variables necesarias para
realimentación auditiva
[24]boolean instrumentos=false;
[25]int posicion = 0;
[26]MediaPlayer vectormp [] = new
MediaPlayer [11];
[27]MediaPlayer
mediaPlayer1,mediaPlayer2,mediaPlayer3;
[28]public Equalizer mEqualizer;
[29]private MediaPlayer;
[30]@Override
[31]protected void onCreate(Bundle
savedInstanceState) {
[32]super.onCreate(savedInstanceState);
[33]setContentView(R.layout.activity_main
8);
[34]getSupportActionBar().setTitle("SFFS
Heat Maps");
[35]getSupportActionBar().setDisplayHome
AsUpEnabled(true);
[36]bluetoothJhr = new
BluetoothJhr(MainActivity4.class,this);
[37]sensor1 = (SpinKitView)
findViewById(R.id.sensor1);
[38]sensor2 = (SpinKitView)
findViewById(R.id.sensor2);
[39]sensor3 = (SpinKitView)
findViewById(R.id.sensor3);
[40]sensor4 = (SpinKitView)
findViewById(R.id.sensor4);
[41]sensor5 = (SpinKitView)
findViewById(R.id.sensor5);
[42]sensor6 = (SpinKitView)
findViewById(R.id.sensor6);
[43]sensor7 = (SpinKitView)
findViewById(R.id.sensor7);
[44]sensor8 = (SpinKitView)
findViewById(R.id.sensor8);
[45]vectormp[0] = MediaPlayer.create(this,
R.raw.uno);
[46]vectormp[1] = MediaPlayer.create(this,
R.raw.dos);
[47]vectormp[2] = MediaPlayer.create(this,
R.raw.tres);
[48]vectormp[3] = MediaPlayer.create(this,
R.raw.cuatro);
[49]vectormp[4] = MediaPlayer.create(this,
R.raw.cinco);
[50]vectormp[5] = MediaPlayer.create(this,
R.raw.seis);
[51]vectormp[6] = MediaPlayer.create(this,
R.raw.siente);
[52]vectormp[7] = MediaPlayer.create(this,
```

```
R.raw.ocho);
[53]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[54]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[55]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[56]mediaPlayer1= MediaPlayer.create(this,R.raw.vocalm);
[57]mediaPlayer2= MediaPlayer.create(this,R.raw.melodicm);
[58]mediaPlayer3= MediaPlayer.create(this,R.raw.drumm);
[59]final int sessionId = vectormp[posicion].getAudioSessionId();
[60]vectormp[posicion].setLooping(true);
[61]vectormp[posicion].setVolume((float) 1, 1);
[62]mediaPlayer1.setLooping(true);
[63]mediaPlayer2.setLooping(true);
[64]mediaPlayer3.setLooping(true);
[65]mEqualizer = new Equalizer(0,sessionId);
[66]final short lowerEqualizerBandLevel = mEqualizer.getBandLevelRange()[0];
[67]final short upperEqualizerBandLevel = mEqualizer.getBandLevelRange()[1];
[68]mEqualizer.setEnabled(true);
[69]metodoPrincipal();
[70]}
[71]//Método que se ejecuta al iniciar el ciclo de vida del Main Activity
[72]@Override
[73]public void onResume(){
[74]super.onResume();
[75]bluetoothJhr.ConectaBluetooth();
[76]bluetoothJhr.ResetearRx();
[77]}
[78]//Método que se ejecuta al  pausar el ciclo de vida del Main Activity
[79]@Override
[80]public void onPause() {
[81]super.onPause();
[82]bluetoothJhr.CierraConexion();
[83]}
[84]private void Time(int i){
[85]try {
[86]Thread.sleep(i);
[87]} catch (InterruptedException e) {
[88]e.printStackTrace();
[89]}
[90]}
[91]//Metodo principal
[92]public void metodoPrincipal(){
[93]Timer timer = new Timer();
[94]TimerTask task = new TimerTask()
[95]{
[96]//Hilo paralelo al hilo principal
[97]@Override
[98]public void run()
[99]{
[100]//Hilo principal
[101]runOnUiThread(new Runnable()
[102]{
[103]@Override
[104]public void run()
[105]{
[106]Hilo2();
[107]variarfrecuencia1();
[108]if (ic < 2 ){
[109]Hilo3();
[110]}if(ic>2){
[111]Hilo4();
[112]}
[113]if (instrumentos=true) {
[114]variarintrumentos();
[115]}
[116]for (short bandIdx = 0; bandIdx < mEqualizer.getNumberOfBands(); bandIdx++) {
[117]mEqualizer.setBandLevel(bandIdx, (short) valor[bandIdx]);
[118]}
[119]bluetoothJhr.ResetearRx();
[120]}
[121]});
[122]}
[123]};
[124]timer.schedule(task, 1000, 200);
[125]}
[126]//Hilos secundarios
[127]private void Hilo2() {
```

```
[128]runOnUiThread(new Runnable(){
[129]public void run() {
[130]//Se recibe el mensaje mediante bluetooth y se operara para ser usado
[131]Mensaje = bluetoothJhr.Rx();
[132]if(!Mensaje.equals("")) {
[133]try {
[134]i0 = Double.parseDouble(Mensaje);
[135]} catch (NumberFormatException e) {
[136]}
[137]ic = (i0 / 1000000000000000l) % 100;
[138]// ic = (i0 / 10000000000000000l) % 100;
[139]if (ic > 2) {
[140]i1L = (i0 / 10000000000000l) % 100;
[141]i2L = (i0 / 100000000000l) % 100;
[142]i3L = (i0 / 100000000) % 100;
[143]i4L = (i0 / 1000000) % 100;
[144]i5L = (i0 / 10000) % 100;
[145]i6L = (i0 / 100) % 100;
[146]}
[147]if (ic < 2) {
[148]i1R = (i0 / 10000000000000l) % 100;
[149]i2R = (i0 / 100000000000l) % 100;
[150]i3R = (i0 / 100000000) % 100;
[151]i4R = (i0 / 1000000) % 100;
[152]i5R = (i0 / 10000) % 100;
[153]i6R = (i0 / 100) % 100;
[154]i7R = i0 % 100;
[155]}
[156]}
[157]}
[158]});
[159]}
[160]private void Hilo3() {
[161]runOnUiThread(new Runnable(){
[162]public void run() {
[163]variarcolor1L();
[164]variarcolor2L();
[165]variarcolor3L();
[166]variarcolor4L();
[167]}
[168]});
[169]}
[170]private void Hilo4() {
[171]runOnUiThread(new Runnable(){
[172]public void run() {
[173]variarcolor1R();
[174]variarcolor2R();
[175]variarcolor3R();
[176]variarcolor4R();
[177]}
[178]});
[179]}
[180]//Método para variar la frecuencias
[181]public void variarfrecuencia1() {
[182]if (i2L <= 30 & i3L > 20 & i4L > 20 || i2R <= 30 & i3R > 20 & i4R > 20 ) {
[183]valor[1]= -1500;
[184]valor[2]= -1500;
[185]valor[3]= -1500;
[186]valor[4]= -1500;
[187]}else if (i2L < 30 & i3L < 20 & i4L > 20 || i2R < 30 & i3R < 20 & i4R > 20 ) {
[188]valor[1]= -1500;
[189]valor[2]= -1500;
[190]}else {
[191]valor[1]= 1500;
[192]valor[2]= 1500;
[193]valor[3]=500;
[194]valor[4]= 500;
[195]}
[196]}
[197]//Métodos para variar el color de los spin "sensores"
[198]public void variarcolor1L() {
[199]if (i1L >= 80) {
[200]sensor1.setColor(Color.RED);
[201]}
[202]if (i1L < 80 && i1L > 50) {
[203]sensor1.setColor(Color.YELLOW);
[204]}
[205]if (i1L < 50 && i1L > 30) {
[206]sensor1.setColor(Color.GREEN);
[207]}
[208]if (i1L < 30 && i1L>10) {
[209]sensor1.setColor(Color.BLUE);
[210]}
[211]}
[212]public void variarcolor2L(){
[213]if (i2L >= 80) {
[214]sensor2.setColor(Color.RED);
[215]}
```

```
[216]if (i2L < 80 && i2L > 50) {
[217]sensor2.setColor(Color.YELLOW);
[218]}
[219]if (i2L <= 50 && i2L > 30) {
[220]sensor2.setColor(Color.GREEN);
[221]}
[222]if (i2L < 30 && i2L>10) {
[223]sensor2.setColor(Color.BLUE);
[224]}
[225]}
[226]public void variarcolor3L(){
[227]if (i3L >= 80) {
[228]sensor3.setColor(Color.RED);
[229]}
[230]if (i3L < 80 && i3L > 50) {
[231]sensor3.setColor(Color.YELLOW);
[232]}
[233]if (i3L <= 50 && i3L > 30) {
[234]sensor3.setColor(Color.GREEN);
[235]}
[236]if (i3L < 30 && i3L>10) {
[237]sensor3.setColor(Color.BLUE);
[238]}
[239]}
[240]public void variarcolor4L() {
[241]if (i4L >= 80) {
[242]sensor4.setColor(Color.RED);
[243]}
[244]if (i4L < 80 && i4L > 50) {
[245]sensor4.setColor(Color.YELLOW);
[246]}
[247]if (i4L <= 50 && i4L > 30) {
[248]sensor4.setColor(Color.GREEN);
[249]}
[250]if (i4L < 30 && i4L > 10) {
[251]sensor4.setColor(Color.BLUE);
[252]}
[253]}
[254]public void variarcolor1R() {
[255]if (i1R >= 80) {
[256]sensor5.setColor(Color.RED);
[257]}
[258]if (i1R < 80 && i1R > 50) {
[259]sensor5.setColor(Color.YELLOW);
[260]}
[261]if (i1R < 50 && i1R > 30) {
[262]sensor5.setColor(Color.GREEN);
[263]}
[264]if (i1R < 30 && i1R>10) {
[265]sensor5.setColor(Color.BLUE);
[266]}
[267]}
[268]public void variarcolor2R(){ //conecta a
blu
[269]if (i2R >= 80) {
[270]sensor6.setColor(Color.RED);
[271]}
[272]if (i2R < 80 && i2R > 50) {
[273]sensor6.setColor(Color.YELLOW);
[274]}
[275]if (i2R <= 50 && i2R > 30) {
[276]sensor6.setColor(Color.GREEN);
[277]}
[278]if (i2R < 30 && i2R>10) {
[279]sensor6.setColor(Color.BLUE);
[280]}
[281]}
[282]public void variarcolor3R(){ //conecta a
blu
[283]if (i3R >= 80) {
[284]sensor7.setColor(Color.RED);
[285]}
[286]if (i3R < 80 && i3R > 50) {
[287]sensor7.setColor(Color.YELLOW);
[288]}
[289]if (i3R <= 50 && i3R > 30) {
[290]sensor7.setColor(Color.GREEN);
[291]}
[292]if (i3R < 30 && i3R>10) {
[293]sensor7.setColor(Color.BLUE);
[294]}
[295]}
[296]public void variarcolor4R() { //conecta
a blu
[297]if (i4R >= 80) {
[298]sensor8.setColor(Color.RED);
[299]}
[300]if (i4R < 80 && i4R > 50) {
[301]sensor8.setColor(Color.YELLOW);
[302]}
[303]if (i4R <= 50 && i4R > 30) {
[304]sensor8.setColor(Color.GREEN);
```

```
[305]}
[306]if (i4R < 30 && i4R > 10) {
[307]sensor8.setColor(Color.BLUE);
[308]}
[309]}
[310]//Método para variar el volumen de los instrumentos
[311]public void variarintrumentos(){
[312]if (i2L <= 30 & i3L > 20 & i4L > 20 || i2R <= 30 & i3R > 20 & i4R > 20 ) {
[313]mediaPlayer1.setVolume((float)0,0);
[314]mediaPlayer2.setVolume((float)1,1);
[315]mediaPlayer3.setVolume((float)1,1);
[316]}else if (i2L < 30 & i3L < 20 & i4L > 20 || i2R < 30 & i3R < 20 & i4R > 20 ) {
[317]mediaPlayer1.setVolume((float)0,0);
[318]mediaPlayer2.setVolume((float)0,0);
[319]mediaPlayer3.setVolume((float)1,1);
[320]}else {
[321]mediaPlayer1.setVolume((float)0,1);
[322]mediaPlayer2.setVolume((float)0,1);
[323]mediaPlayer3.setVolume((float)0,1);
[324]}
[325]}
[326]//Método para mostrar el menu
[327]public boolean onCreateOptionsMenu(Menu menu){
[328]getMenuInflater().inflate(R.menu.mapadecalor, menu);
[329]return true;
[330]}
[331]//Método para asignar las funciones correspondientes a las opciones.
[332]public boolean onOptionsItemSelected(MenuItem item){
[333]int id = item.getItemId();
[334]if(id == R.id.item1){
[335]Toast.makeText(this, "Graficando sensor 1 ", Toast.LENGTH_SHORT).show();
[336]} else if(id == R.id.item2){
[337]Toast.makeText(this, "Graficando sensor 2 ", Toast.LENGTH_SHORT).show();
[338]}else if(id == R.id.item3){
[339]Toast.makeText(this, "Graficando sensor 3 ", Toast.LENGTH_SHORT).show();
[340]}
[341]else if(id == R.id.play){
[342]if (!vectormp[posicion].isPlaying()) {
[343]if (mediaPlayer1.isPlaying()){
[344]mediaPlayer1.stop();
[345]mediaPlayer2.stop();
[346]mediaPlayer3.stop();
[347]}
[348]vectormp[posicion].start();
[349]}
[350]if (vectormp[posicion].isPlaying()) {
[351]vectormp[posicion].stop();
[352]vectormp[posicion].reset();
[353]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[354]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[355]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[356]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[357]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[358]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[359]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[360]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[361]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[362]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[363]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[364]posicion = 0;
[365]vectormp[posicion].start();
[366]}
[367]vectormp[posicion].start();
[368]Toast.makeText(this, "Reproducción manual con realimentación por ecualización ", Toast.LENGTH_SHORT).show();
[369]}
[370]else if(id == R.id.pause){
```

```
[371]if (vectormp[posicion].isPlaying())
[372]vectormp[posicion].pause();
[373]if (mediaPlayer1.isPlaying()){
[374]mediaPlayer1.pause();
[375]mediaPlayer2.pause();
[376]mediaPlayer3.pause();
[377]}
[378]Toast.makeText(this, "Pause",
Toast.LENGTH_SHORT).show();
[379]}
[380]else if(id == R.id.siguiente){
[381]if (mediaPlayer1.isPlaying())
[382]Toast.makeText(this, "Para este modo
de realimentación solo hay una canción
disponible ",
Toast.LENGTH_SHORT).show();
[383]if(posicion < vectormp.length -1){
[384]if(vectormp[posicion].isPlaying()){
[385]vectormp[posicion].stop();
[386]posicion++;
[387]vectormp[posicion].start();
[388]} else {
[389]posicion++;
[390]}
[391]} else {
[392]Toast.makeText(this, "No hay más
caciones",
Toast.LENGTH_SHORT).show();
[393]}
[394]}
[395]else if(id == R.id.anter){
[396]siguiente();
[397]}
[398]else if(id == R.id.instru){
[399]if(vectormp[posicion].isPlaying()){
[400]vectormp[posicion].stop();
[401]mediaPlayer1=
MediaPlayer.create(this,R.raw.vocalm);
[402]mediaPlayer2=
MediaPlayer.create(this,R.raw.melodicm);
[403]mediaPlayer3=
MediaPlayer.create(this,R.raw.drumm);
[404]mediaPlayer1.start();
[405]mediaPlayer2.start();
[406]mediaPlayer3.start();
[407]instrumentos=true;
[408]Toast.makeText(this, "Realimentación
por instrumentos",
Toast.LENGTH_SHORT).show();
[409]} else {
[410]mediaPlayer1.start();
[411]mediaPlayer2.start();
[412]mediaPlayer3.start();
[413]instrumentos=true;
[414]}
[415]}
[416]if (id == android.R.id.home) {
[417]onBackPressed();
[418]}
[419]final int sessionId =
vectormp[posicion].getAudioSessionId();
[420]mEqualizer = new
Equalizer(0,sessionId);
[421]mEqualizer.setEnabled(true);
[422]return
super.onOptionsItemSelected(item);
[423]}
[424]//Método para cerrar funciones al
terminar el ciclo de vida del Main activity
[425]@Override
[426]public void onDestroy(){
[427]super.onDestroy();
[428]try {
[429]mediaPlayer1.stop();
[430]mediaPlayer2.stop();
[431]mediaPlayer3.stop();
[432]vectormp[posicion].stop();
[433]} catch (Exception ex) {
[434]}
[435]}
[436]//Método para cambiar la canción
[437]public void siguiente(){
[438]if (mediaPlayer1.isPlaying()){
[439]mediaPlayer1.stop();
[440]mediaPlayer2.stop();
[441]mediaPlayer3.stop();
[442]}
[443]Toast.makeText(this, "Reproducción
automática",
Toast.LENGTH_SHORT).show();
[444]if (i5R>=10 & i5R<11){
[445]if(vectormp[posicion].isPlaying()) {
```

```
[446]vectormp[posicion].stop();
[447]vectormp[posicion].reset();
[448]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[449]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[450]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[451]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[452]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[453]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[454]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[455]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[456]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[457]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[458]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[459]posicion = 0;
[460]vectormp[posicion].start();
[461]Toast.makeText(this, "Sincronizado a 60 bpm", Toast.LENGTH_SHORT).show();
[462]}else{
[463]posicion = 0;
[464]vectormp[posicion].start();
[465]}
[466]}
[467]if (i5R>=11 & i5R<12){
[468]if(vectormp[posicion].isPlaying()) {
[469]vectormp[posicion].stop();
[470]vectormp[posicion].reset();
[471]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[472]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[473]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[474]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[475]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[476]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[477]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[478]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[479]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[480]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[481]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[482]posicion = 1;
[483]vectormp[posicion].start();
[484]Toast.makeText(this, "Sincronizado a 70 bpm", Toast.LENGTH_SHORT).show();
[485]}else{
[486]posicion = 1;
[487]vectormp[posicion].start();
[488]}
[489]}
[490]if (i5R>=12 & i5R<13){
[491]if(vectormp[posicion].isPlaying()) {
[492]vectormp[posicion].stop();
[493]vectormp[posicion].reset();
[494]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[495]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[496]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[497]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[498]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[499]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[500]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[501]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[502]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[503]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
```

```
[504]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[505]posicion = 2;
[506]vectormp[posicion].start();
[507]Toast.makeText(this, "Sincronizado a 80 bpm", Toast.LENGTH_SHORT).show();
[508]}else{
[509]posicion = 2;
[510]vectormp[posicion].start();
[511]}
[512]}
[513]if (i5R>=13 & i5R<14){
[514]if(vectormp[posicion].isPlaying()) {
[515]vectormp[posicion].stop();
[516]vectormp[posicion].reset();
[517]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[518]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[519]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[520]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[521]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[522]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[523]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[524]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[525]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[526]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[527]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[528]posicion = 3;
[529]vectormp[posicion].start();
[530]Toast.makeText(this, "Sincronizado a 90 bpm", Toast.LENGTH_SHORT).show();
[531]}else{
[532]posicion = 3;
[533]vectormp[posicion].start();
[534]}
[535]}
[536]if (i5R>=14 & i5R<15){
[537]if(vectormp[posicion].isPlaying()) {
[538]vectormp[posicion].stop();
[539]vectormp[posicion].reset();
[540]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[541]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[542]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[543]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[544]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[545]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[546]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[547]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[548]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[549]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[550]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[551]posicion = 4;
[552]vectormp[posicion].start();
[553]Toast.makeText(this, "Sincronizado a 100 bpm", Toast.LENGTH_SHORT).show();
[554]}else{
[555]posicion = 4;
[556]vectormp[posicion].start();
[557]}
[558]}
[559]if (i5R>=15 & i5R<16){
[560]if(vectormp[posicion].isPlaying()) {
[561]vectormp[posicion].stop();
[562]vectormp[posicion].reset();
[563]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[564]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[565]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
```

```
[566]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[567]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[568]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[569]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[570]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[571]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[572]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[573]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[574]posicion = 5;
[575]vectormp[posicion].start();
[576]Toast.makeText(this, "Sincronizado a 110 bpm", Toast.LENGTH_SHORT).show();
[577]}else{
[578]posicion = 5;
[579]vectormp[posicion].start();
[580]}
[581]}
[582]if (i5R>=16 & i5R<17){
[583]if(vectormp[posicion].isPlaying()) {
[584]vectormp[posicion].stop();
[585]vectormp[posicion].reset();
[586]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[587]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[588]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[589]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[590]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[591]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[592]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[593]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[594]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[595]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[596]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[597]posicion = 6;
[598]vectormp[posicion].start();
[599]Toast.makeText(this, "Sincronizado a 120 bpm", Toast.LENGTH_SHORT).show();
[600]}else{
[601]posicion = 6;
[602]vectormp[posicion].start();
[603]}
[604]}
[605]if (i5R>=17 & i5R<18){
[606]if(vectormp[posicion].isPlaying()) {
[607]vectormp[posicion].stop();
[608]vectormp[posicion].reset();
[609]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[610]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[611]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[612]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[613]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[614]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[615]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[616]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[617]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[618]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[619]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[620]posicion = 7;
[621]vectormp[posicion].start();
[622]Toast.makeText(this, "Sincronizado a 130 bpm",
```

```
Toast.LENGTH_SHORT).show();
[623]}else{
[624]posicion = 7;
[625]vectormp[posicion].start();
[626]}
[627]}
[628]if (i5R>=18 & i5R<19){
[629]if(vectormp[posicion].isPlaying()) {
[630]vectormp[posicion].stop();
[631]vectormp[posicion].reset();
[632]vectormp[0] = MediaPlayer.create(this,
R.raw.uno);
[633]vectormp[1] = MediaPlayer.create(this,
R.raw.dos);
[634]vectormp[2] = MediaPlayer.create(this,
R.raw.tres);
[635]vectormp[3] = MediaPlayer.create(this,
R.raw.cuatro);
[636]vectormp[4] = MediaPlayer.create(this,
R.raw.cinco);
[637]vectormp[5] = MediaPlayer.create(this,
R.raw.seis);
[638]vectormp[6] = MediaPlayer.create(this,
R.raw.siente);
[639]vectormp[7] = MediaPlayer.create(this,
R.raw.ocho);
[640]vectormp[8] = MediaPlayer.create(this,
R.raw.nueve);
[641]vectormp[9] = MediaPlayer.create(this,
R.raw.diez);
[642]vectormp[10] =
MediaPlayer.create(this, R.raw.once);
[643]posicion = 8;
[644]vectormp[posicion].start();
[645]Toast.makeText(this, "Sincronizado a
140 bpm",
Toast.LENGTH_SHORT).show();
[646]}else{
[647]posicion = 8;
[648]vectormp[posicion].start();
[649]}
[650]}
[651]if (i5R>=19 & i5R<20){
[652]if(vectormp[posicion].isPlaying()) {
[653]vectormp[posicion].stop();
[654]vectormp[posicion].reset();
[655]vectormp[0] = MediaPlayer.create(this,
R.raw.uno);
[656]vectormp[1] = MediaPlayer.create(this,
R.raw.dos);
[657]vectormp[2] = MediaPlayer.create(this,
R.raw.tres);
[658]vectormp[3] = MediaPlayer.create(this,
R.raw.cuatro);
[659]vectormp[4] = MediaPlayer.create(this,
R.raw.cinco);
[660]vectormp[5] = MediaPlayer.create(this,
R.raw.seis);
[661]vectormp[6] = MediaPlayer.create(this,
R.raw.siente);
[662]vectormp[7] = MediaPlayer.create(this,
R.raw.ocho);
[663]vectormp[8] = MediaPlayer.create(this,
R.raw.nueve);
[664]vectormp[9] = MediaPlayer.create(this,
R.raw.diez);
[665]vectormp[10] =
MediaPlayer.create(this, R.raw.once);
[666]posicion = 9;
[667]vectormp[posicion].start();
[668]Toast.makeText(this, "Sincronizado a
150 bpm",
Toast.LENGTH_SHORT).show();
[669]}else{
[670]posicion = 9;
[671]vectormp[posicion].start();
[672]}
[673]}
[674]if (i5R>=20 & i5R<21){
[675]if(vectormp[posicion].isPlaying()) {
[676]vectormp[posicion].stop();
[677]vectormp[posicion].reset();
[678]vectormp[0] = MediaPlayer.create(this,
R.raw.uno);
[679]vectormp[1] = MediaPlayer.create(this,
R.raw.dos);
[680]vectormp[2] = MediaPlayer.create(this,
R.raw.tres);
[681]vectormp[3] = MediaPlayer.create(this,
R.raw.cuatro);
[682]vectormp[4] = MediaPlayer.create(this,
R.raw.cinco);
```

```
[683]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[684]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[685]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[686]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[687]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[688]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[689]posicion = 10;
[690]vectormp[posicion].start();
[691]Toast.makeText(this, "Sincronizado a 160 bpm", Toast.LENGTH_SHORT).show();
[692]}else{
[693]posicion = 10;
[694]vectormp[posicion].start();
[695]}
[696]}
[697]if (i5R>=21 ){
[698]if(vectormp[posicion].isPlaying()) {
[699]vectormp[posicion].stop();
[700]vectormp[posicion].reset();
[701]vectormp[0] = MediaPlayer.create(this, R.raw.uno);
[702]vectormp[1] = MediaPlayer.create(this, R.raw.dos);
[703]vectormp[2] = MediaPlayer.create(this, R.raw.tres);
[704]vectormp[3] = MediaPlayer.create(this, R.raw.cuatro);
[705]vectormp[4] = MediaPlayer.create(this, R.raw.cinco);
[706]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[707]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[708]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[709]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[710]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[711]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[712]posicion = 10;
[713]vectormp[posicion].start();
[714]Toast.makeText(this, "Sincronizado a más de 160 bpm", Toast.LENGTH_SHORT).show();
[715]}else{
[716]posicion = 10;
[717]vectormp[posicion].start();
[718]}
[719]}
[720]}
[721]}
```

Anexo 7. Código SFFS para la realimentación visual (SFFS Line Graphs) adicional también se realimenta auditivamente con (SFFS Music Equalizer).

```
[1]package com.example.SFFS;
[2]//Librerías necesarias
[3]import
androidx.appcompat.app.AppCompatActivit
y;
[4]import android.content.Context;
[5]import android.media.MediaPlayer;
[6]import android.media.audiofx.Equalizer;
[7]import android.os.Bundle;
[8]import android.view.Menu;
[9]import android.view.MenuItem;
[10]import android.view.View;
[11]import android.widget.Button;
[12]import android.widget.LinearLayout;
[13]import android.widget.Toast;
[14]import
com.example.hp.bluetoothjhr.BluetoothJhr;
[15]import
com.github.ybq.android.spinkit.SpinKitVie
w;
[16]import com.juang.jplot.PlotPlanitoXY;
[17]import java.util.Timer;
[18]import java.util.TimerTask;
[19]public class MainActivity10 extends
AppCompatActivity {
[20]//Variables necesarias para graficar
[21]int repetir = 2, posicion = 0;
[22]Timer timer;
[23]private PlotPlanitoXY plot;
[24]private LinearLayout pantalla;
[25]Context context;
[26]int i = 0; // contador de datos
[27]float[] Xd, Yd, Yd2, Xd2, Xd3, Yd3,
Xd4, Yd4,Xd5, Yd5;
[28]boolean Lef=false;
[29]boolean plot1=false;
[30]boolean plot2=false;
[31]boolean plot3=false;
[32]boolean plot4=false;
[33]boolean plot5=false;
[34]//Arrys para almacenamiento de datos
[35]private float[] X = new float[4000];
[36]private float[] Y = new float[4000];
[37]private float[] X2 = new float[4000];
[38]private float[] Y2 = new float[4000];
[39]private float[] X3 = new float[4000];
[40]private float[] Y3 = new float[4000];
[41]private float[] X4 = new float[4000];
[42]private float[] Y4 = new float[4000];
[43]private float[] X5 = new float[4000];
[44]private float[] Y5 = new float[4000];
[45]//Variables necesarias para la gestión de
datos
[46]BluetoothJhr bluetoothJhr;
[47]float ic,i0, i1L, i2L, i3L, i4L, i5L, i6L,
i7L, i8L;
[48]float i1, i2, i3, i4, i5, i6, i7, i8,it1;
[49]int[] valor = {0, 0, 0, 0, 0};
[50]//Variables necesarias para
realimentación auditiva
[51]MediaPlayer mp;
[52]MediaPlayer vectormp[] = new
MediaPlayer[11];
[53]public Equalizer mEqualizer;
[54]@Override
[55]protected void onCreate(Bundle
savedInstanceState) {
[56]super.onCreate(savedInstanceState);
[57]setContentView(R.layout.activity_main
10);
[58]getSupportActionBar().setTitle("SFFS
Line Graphs");
[59]bluetoothJhr = new
BluetoothJhr(MainActivity2.class, this);
[60]vectormp[0] = MediaPlayer.create(this,
R.raw.uno);
[61]vectormp[1] = MediaPlayer.create(this,
R.raw.dos);
[62]vectormp[2] = MediaPlayer.create(this,
R.raw.tres);
[63]vectormp[3] = MediaPlayer.create(this,
R.raw.cuatro);
[64]vectormp[4] = MediaPlayer.create(this,
R.raw.cinco);
```

```
[65]vectormp[5] = MediaPlayer.create(this, R.raw.seis);
[66]vectormp[6] = MediaPlayer.create(this, R.raw.siente);
[67]vectormp[7] = MediaPlayer.create(this, R.raw.ocho);
[68]vectormp[8] = MediaPlayer.create(this, R.raw.nueve);
[69]vectormp[9] = MediaPlayer.create(this, R.raw.diez);
[70]vectormp[10] = MediaPlayer.create(this, R.raw.once);
[71]final int sessionId = vectormp[posicion].getAudioSessionId();
[72]vectormp[posicion].setLooping(true);
[73]vectormp[posicion].setVolume((float) 1, 1);
[74]mEqualizer = new Equalizer(0,sessionId);
[75]final short lowerEqualizerBandLevel = mEqualizer.getBandLevelRange()[0];
[76]final short upperEqualizerBandLevel = mEqualizer.getBandLevelRange()[1];
[77]mEqualizer.setEnabled(true);
[78]pantalla = (LinearLayout) (findViewById(R.id.pantalla));
[79]context = this;
[80]}
[81]//Método que se ejecuta al iniciar el ciclo de vida del Main Activity
[82]@Override
[83]public void onResume() {
[84]super.onResume();
[85]bluetoothJhr.ConectaBluetooth();
[86]bluetoothJhr.ResetearRx();
[87]}
[88]//Método que se ejecuta al  pausar el ciclo de vida del Main Activity
[89]@Override
[90]public void onPause() {
[91]super.onPause();//cuando salgo no se ejecuta en segundo plano es decir deconecta blu
[92]bluetoothJhr.CierraConexion();
[93]}
[94]private void Time(int i) {
[95]try {
[96]Thread.sleep(i);
[97]} catch (InterruptedException e) {
[98]e.printStackTrace();
[99]}
[100]}
[101]//Métodos para variar la frecuencias
[102]public void variarEQ1() {
[103]if (i1 >= 50 & i1 < 99) {
[104]valor[1]= (int) (15000/(i1));
[105]}
[106]if (i1 < 50 & i1>10) {
[107]valor[1]=(int) (-15000/(i1));
[108]}
[109]}
[110]public void variarEQ2(){ //conecta a blu
[111]if (i2 >= 50 & i2 < 99) {
[112]valor[2]= (int) (15000/(i2));
[113]}
[114]if (i2 < 50 & i2>10) {
[115]valor[2]=(int) (-15000/(i2));
[116]}
[117]}
[118]public void variarEQ3(){ //conecta a blu
[119]if (i3 >= 50 & i3 < 99) {
[120]valor[3]= (int) (15000/(i3));
[121]}
[122]if (i3 < 50 & i3>10) {
[123]valor[3]=(int) (-15000/(i3));
[124]}
[125]}
[126]public void variarEQ4(){
[127]if (i4 >= 50 & i4 < 99) {
[128]valor[4]= (int) (15000/(i4));
[129]}
[130]if (i4 < 50 & i4>10) {
[131]valor[4]=(int) (-15000/(i4));
[132]}
[133]}
[134]//Hilo que recibe el mensaje mediante bluetooth y se operara para ser usado
[135]private void Hilo2() {
[136]runOnUiThread(new Runnable(){
[137]public void run() {
```

```
[138]String Mensaje = bluetoothJhr.Rx();
[139]if (!Mensaje.equals("")) {
[140]try {
[141]i0 = Float.parseFloat(Mensaje);
[142]} catch (NumberFormatException e) {
[143]}
[144]}
[145]ic = (i0 / 1000000000000000l) % 100;
[146]if (Lef==true){
[147]if (ic > 2) {
[148]i1 = (i0 / 10000000000000l) % 100;
[149]i2 = (i0 / 100000000000l) % 100;
[150]i3 = (i0 / 100000000) % 100;
[151]i4 = (i0 / 1000000) % 100;
[152]i5 = (i0 / 10000) % 100;
[153]i6 = (i0 / 100) % 100;
[154]}
[155]}
[156]if (Lef==false){
[157]if (ic < 2) {
[158]i1 = (i0 / 10000000000000l) % 100;
[159]i2 = (i0 / 100000000000l) % 100;
[160]i3 = (i0 / 100000000) % 100;
[161]i4 = (i0 / 1000000) % 100;
[162]i5 = (i0 / 10000) % 100;
[163]i6 = (i0 / 100) % 100;
[164]}
[165]}
[166]it1=i1+i2+i3+i4;
[167]variarEQ1();
[168]variarEQ2();
[169]variarEQ3();
[170]variarEQ4();
[171]for (short bandIdx = 0; bandIdx <
mEqualizer.getNumberOfBands();
bandIdx++) {
[172]mEqualizer.setBandLevel(bandIdx,
(short) valor[bandIdx]);
[173]}
[174]}
[175]});
[176]}
[177]public void metodoPrincipal(){
[178]// inicializamos el grafico dinámico
[179]plot = new PlotPlanitoXY(context,"xx
vs yy","xx","yy");
[180]plot.SetEscalaAutomatica(true);
[181]plot.SetHD(false);
[182]timer = new Timer();
[183]TimerTask task = new TimerTask()
[184]{
[185]//Hilo paralelo al hilo principal
[186]@Override
[187]public void run()
[188]{
[189]//Hilo principal
[190]runOnUiThread(new Runnable()
[191]{
[192]@Override
[193]public void run()
[194]{
[195]//Se gestionan los datos y se grafican
[196]Hilo2();
[197]if (i2>10) {
[198]pantalla.removeAllViews();
[199]Xd = new float[i + 1];
[200]Xd[0] = 3.4f;
[201]Yd = new float[i + 1];
[202]Xd2 = new float[i + 1];
[203]Yd2 = new float[i + 1];
[204]Xd3 = new float[i + 1];
[205]Yd3 = new float[i + 1];
[206]Xd4 = new float[i + 1];
[207]Yd4 = new float[i + 1];
[208]Xd5 = new float[i + 1];
[209]Yd5 = new float[i + 1];
[210]X[i] = i;
[211]Y[i] = i1;
[212]for (int j = 0; j < Xd.length; j++) {
[213]Xd[j] = X[j];
[214]Yd[j] = Y[j];
[215]}
[216]X2[i] = i;
[217]Y2[i] = i2;
[218]for (int j = 0; j < Xd2.length; j++) {
[219]Xd2[j] = X2[j];
[220]Yd2[j] = Y2[j];
[221]}
[222]X3[i] = i;
[223]Y3[i] = i3;
[224]for (int j = 0; j < Xd3.length; j++) {
[225]Xd3[j] = X3[j];
```

```
[226]Yd3[j] = Y3[j];
[227]}
[228]X4[i] = i;
[229]Y4[i] = i4;
[230]for (int j = 0; j < Xd4.length; j++) {
[231]Xd4[j] = X4[j];
[232]Yd4[j] = Y4[j];
[233]}
[234]X5[i] = i;
[235]Y5[i] = it1;
[236]for (int j = 0; j < Xd5.length; j++) {
[237]Xd5[j] = X5[j];
[238]Yd5[j] = Y5[j];
[239]}
[240]if (plot1 == true)
[241]plot.SetSerie1(Xd, Yd, "Sensor1", 7, true);
[242]if (plot2 == true)
[243]plot.SetSerie2(Xd2, Yd2, "Sensor2", 7, true, 1);
[244]if (plot3 == true)
[245]plot.SetSerie3(Xd3, Yd3, "Sensor3", 7, true, 1);
[246]if (plot4 == true)
[247]plot.SetSerie4(Xd4, Yd4, "Sensor4", 7, true, 1);
[248]if (plot5 == true)
[249]plot.SetSerie5(Xd5, Yd5, "Suma pie derecho", 2, true, 1);
[250]pantalla.addView(plot);
[251]i = i + 1;
[252]if (i == 201) {
[253]i = 0;
[254]}
[255]}
[256]bluetoothJhr.ResetearRx();
[257]}
[258]});
[259]}
[260]};
[261]timer.schedule(task, 1000, 100);
[262]}
[263]public void parar(View v){
[264]timer.cancel();
[265]}
[266]//Método para mostrar y ocultar el menú
[267]public boolean onCreateOptionsMenu(Menu menu){
[268]getMenuInflater().inflate(R.menu.graficas, menu);
[269]return true;
[270]}
[271]//Método para asignar las funciones correspondientes a las opciones.
[272]public boolean onOptionsItemSelected(MenuItem item){
[273]int id = item.getItemId();
[274]if(id == R.id.item1){
[275]if (plot1==true) {
[276]item.setChecked(false);
[277]i = 0;
[278]plot1=false;
[279]metodoPrincipal();
[280]}
[281]else {
[282]plot1 = true;
[283]Toast.makeText(this, "Sensor 1 ", Toast.LENGTH_SHORT).show();
[284]item.setChecked(true);
[285]}
[286]} else if(id == R.id.item2){
[287]if (plot2==true) {
[288]plot2=false;
[289]i = 0;
[290]item.setChecked(false);
[291]metodoPrincipal();
[292]}
[293]else {
[294]plot2=true;
[295]item.setChecked(true);
[296]Toast.makeText(this, "Sensor 2 ", Toast.LENGTH_SHORT).show();
[297]}
[298]}else if(id == R.id.item3){
[299]if (plot3==true) {
[300]plot3=false;
[301]i = 0;
[302]item.setChecked(false);
[303]metodoPrincipal();
[304]}
[305]else {
[306]plot3 = true;
```

[307]item.setChecked(true);
[308]}Toast.makeText(this, "Sensor 3 ", Toast.LENGTH_SHORT).show();
[309]}else if(id == R.id.item4){
[310]if (plot4==true) {
[311]plot4=false;
[312]i = 0;
[313]item.setChecked(false);
[314]metodoPrincipal();
[315]}
[316]else {
[317]plot4 = true;
[318]Toast.makeText(this, "Sensor 4 ", Toast.LENGTH_SHORT).show();
[319]item.setChecked(true);
[320]}
[321]}else if(id == R.id.item5){
[322]if (Lef==true){
[323]i = 0;
[324]Lef = false;
[325]item.setChecked(false);
[326]metodoPrincipal();
[327]Toast.makeText(this, "Desde ahora de graficaran los sensores del pie derecho ", Toast.LENGTH_SHORT).show();
[328]}else{
[329]i = 0;
[330]Lef=true;
[331]item.setChecked(true);
[332]metodoPrincipal();
[333]Toast.makeText(this, "Desde ahora de graficaran los sensores del pie izquierdo ", Toast.LENGTH_SHORT).show();
[334]}
[335]}
[336]else if(id == R.id.play){
[337]if (!vectormp[posicion].isPlaying()) {
[338]vectormp[posicion].start();
[339]}
[340]Toast.makeText(this, "play", Toast.LENGTH_SHORT).show();
[341]}
[342]else if(id == R.id.pause){
[343]vectormp[posicion].pause();
[344]Toast.makeText(this, "play", Toast.LENGTH_SHORT).show();
[345]}
[346]else if(id == R.id.siguiente){
[347]if(posicion < vectormp.length -1){
[348]if(vectormp[posicion].isPlaying()){
[349]vectormp[posicion].stop();
[350]posicion++;
[351]vectormp[posicion].start();
[352]} else {
[353]posicion++;
[354]}
[355]} else {
[356]Toast.makeText(this, "No hay más caciones", Toast.LENGTH_SHORT).show();
[357]}
[358]}
[359]else if(id == R.id.plot){
[360]metodoPrincipal();
[361]}
[362]if (id == android.R.id.home) {
[363]onBackPressed();
[364]}
[365]final int sessionId = vectormp[posicion].getAudioSessionId();
[366]mEqualizer = new Equalizer(0,sessionId);
[367]mEqualizer.setEnabled(true);
[368]return super.onOptionsItemSelected(item);
[369]}
[370]//Método para cerrar funciones al terminar el ciclo de vida del Main activity
[371]@Override
[372]public void onDestroy(){
[373]super.onDestroy();
[374]try {
[375]vectormp[posicion].stop();
[376]} catch (Exception ex) {
[377]}
[378]}
[379]}

Anexo 8. Código SFFS para la realimentación visual (3D SFFS)

```
[1]package com.example.SFFS;
[2]//Librerías necesarias
[3]import androidx.annotation.NonNull;
[4]import
androidx.appcompat.app.AppCompatActivit
y;
[5]import android.os.Bundle;
[6]import android.view.MenuItem;
[7]import android.view.View;
[8]import
android.view.animation.Animation;
[9]import
android.view.animation.RotateAnimation;
[10]import android.widget.ImageView;
[11]import
com.example.hp.bluetoothjhr.BluetoothJhr;
[12]import java.util.Timer;
[13]import java.util.TimerTask;
[14]public class MainActivity5 extends
AppCompatActivity {
[15]//Variables necesarias
[16]BluetoothJhr bluetoothJhr;
[17]String Mensaje;
[18]double ic,i0,an1,an2,an3,an4;
[19]ImageView z1, z2,z3,z4;
[20]int angulo,angulo2,angulo3,angulo4;
[21]@Override
[22]protected void onCreate(Bundle
savedInstanceState) {
[23]super.onCreate(savedInstanceState);
[24]setContentView(R.layout.activity_main
5);
[25]getSupportActionBar().setTitle("SFFS
Inclination angles");
[26]getSupportActionBar().setDisplayHome
AsUpEnabled(true);
[27]bluetoothJhr = new
BluetoothJhr(MainActivity.class, this);
[28]z1 = (ImageView)
findViewById(R.id.imageView6);
[29]z2 = (ImageView)
findViewById(R.id.imageView7);
[30]z3 = (ImageView)
findViewById(R.id.imageView4);
[31]z4 = (ImageView)
findViewById(R.id.imageView5);
[32]rotarImagen(z1);
[33]metodoPrincipal();
[34]}
[35]public void metodoPrincipal(){
[36]Timer timer = new Timer();
[37]TimerTask task = new TimerTask()
[38]{
[39]//Hilo secundario
[40]@Override
[41]public void run()
[42]{
[43]//hilo principal
[44]runOnUiThread(new Runnable()
[45]{
[46]@Override
[47]public void run()
[48]{
[49]Hilo2();
[50]if (an1 > 0) {
[51]rotarImagen(z1);
[52]}
[53]if (an2 > 0) {
[54]rotarImagen2(z2);
[55]}
[56]if (an3 > 0) {
[57]rotarImagen3(z3);
[58]}
[59]if (an4 > 0) {
[60]rotarImagen4(z4);
[61]}
[62]bluetoothJhr.ResetearRx();
[63]}//hilo2
[64]});
[65]}//hilo1
[66]};timer.schedule(task, 5000, 100);
[67]}
[68]//Hilo que recibe el mensaje mediante
bluetooth y se operara para ser usado
[69]private void Hilo2() {
[70]runOnUiThread(new Runnable(){
```

```
[71]public void run() {
[72]Mensaje = bluetoothJhr.Rx();
[73]if (!Mensaje.equals("")) {
[74]bluetoothJhr.ResetearRx();
[75]try {
[76]i0 = Double.parseDouble(Mensaje);
[77]} catch (NumberFormatException nfe) {
[78]}
[79]ic = (i0 / 1000000000000000l) % 100;
[80]if (ic > 2 ){
[81]an1 = (i0 / 10000) % 100;
[82]an2 = (i0 / 100) % 100;
[83]}if(ic<2){
[84]an3 = (i0 / 100) % 100;
[85]an4 = i0  % 100;
[86]}
[87]angulo = (int) ((an3-58)*2.5); //posterior
iz
[88]angulo2 = (int) ((an3-58)*2.5);
//posterior derecho
[89]angulo3 = (int) ((an4-50)*2.5); //superior
derecho
[90]angulo4 = (int) ((an4-50)*2.5);
//Superior iz
[91]}
[92]}
[93]});
[94]}
[95]@Override
[96]public boolean
onOptionsItemSelected(@NonNull
MenuItem item) {
[97]int id = item.getItemId();
[98]if (id == android.R.id.home) {
[99]onBackPressed();
[100]}
[101]return
super.onOptionsItemSelected(item);
[102]}
[103]//Método que se ejecuta al iniciar el
ciclo de vida del Main Activity
[104]@Override
[105]public void onResume() {
[106]super.onResume();
[107]bluetoothJhr.ConectaBluetooth();
[108]bluetoothJhr.ResetearRx();
[109]}
[110]//Método que se ejecuta al pausar el
ciclo de vida del Main Activity
[111]@Override
[112]public void onPause() {
[113]super.onPause();
[114]bluetoothJhr.CierraConexion();
[115]}
[116]private void Time(int i) {
[117]try {
[118]Thread.sleep(i);
[119]} catch (InterruptedException e) {
[120]e.printStackTrace();
[121]}
[122]}
[123]//Métodos para rotar las imágenes
[124]private void rotarImagen(View view) {
[125]RotateAnimation animation = new
RotateAnimation(angulo, angulo,
[126]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f,
[127]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f);
[128]animation.setDuration(200);
[129]animation.setRepeatCount(Animation.I
NFINITE);
[130]view.startAnimation(animation);
[131]}
[132]private void rotarImagen2(View view)
{
[133]RotateAnimation animation = new
RotateAnimation(angulo2, angulo2,
[134]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f,
[135]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f);
[136]animation.setDuration(200);
[137]animation.setRepeatCount(Animation.I
NFINITE);
[138]view.startAnimation(animation);
[139]}
[140]private void rotarImagen3(View view)
{
[141]RotateAnimation animation = new
RotateAnimation(angulo3, angulo3,
[142]RotateAnimation.RELATIVE_TO_SE
```

```
LF, 0.5f,
[143]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f);
[144]animation.setDuration(200);
[145]animation.setRepeatCount(Animation.I
NFINITE);
[146]view.startAnimation(animation);
[147]}
[148]private void rotarImagen4(View view)
{
[149]RotateAnimation animation = new
RotateAnimation(angulo4, angulo4,
[150]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f,
[151]RotateAnimation.RELATIVE_TO_SE
LF, 0.5f);
[152]animation.setDuration(200);
[153]animation.setRepeatCount(Animation.I
NFINITE);
[154]view.startAnimation(animation);
[155]}
[156]}
```

Anexo 9. Diagrama UML SFFS APP