UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

# Diseño de un *tracker* cefálico para audio 3D

GRADO EN INGENIERÍA DE
SISTEMAS ELECTRÓNICOS

NAVARRETE GÁLVEZ, JUAN JOSÉ

MÁLAGA, 2023

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

# Diseño de un *tracker* cefálico para audio 3D

Autor: Navarrete Gálvez, Juan José.

Tutor: Reyes Lecuona, Arcadio; Pérez Rodríguez, Eduardo Javier.

Departamento: Departamento de Tecnología Electrónica

Titulación: Grado en Ingeniería de Sistemas Electrónicos

Palabras clave: audio 3D, latencia, Bela, sensor inercial, I2C, OSC, sistema empotrado, realidad aumentada.

## Resumen

Actualmente, las tecnologías que engloban a la realidad virtual y aumentada (realidad extendida) están en crecimiento: Apple acaba de lanzar al mercado un dispositivo de este tipo. La generación de audio en tres dimensiones es un reto para cualquier aplicación, pues requiere de un hardware y un software que cumpla con los exigentes requisitos del procesado de audio.

En este documento se detallará cómo se ha creado un sistema que detecta su propia orientación y renderiza audio en colaboración con el Trabajo Fin de Grado presentado por S. Florido [1]. Para ello se ha utilizado una placa Bela mini, la cual utiliza una PocketBeagle para funcionar, y el sensor inercial absoluto BNO055. Todo el sistema incluirá una placa accesoria apilada que añade ciertos controles: potenciómetros para controlar el volumen y la pista de audio, un botón para fijar el punto de referencia y un diodo LED. Además, se incluye una carcasa para el dispositivo, el cual descansará sobre la diadema de unos auriculares.

Con el fin de conocer la viabilidad, se ha diseñado un sistema de pruebas para medir la latencia del dispositivo. Dicho sistema consiste en una cabeza de poliestireno expandido que puede girar sobre sí misma, un generador de evento real y un osciloscopio: cuando se vaya a probar un dispositivo, se medirá la diferencia de tiempo entre el evento real y el evento generado por el sistema de audio. Por último, se probarán otros sistemas a fin de comparar los resultados obtenidos.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

# Design of a head-tracker for 3D audio rendering

Author: Navarrete Gálvez, Juan José.

Supervisor: Reyes Lecuona, Arcadio; Pérez Rodríguez, Eduardo Javier.

Department: Departamento de Tecnología Electrónica

Degree: Grado en Ingeniería de Sistemas Electrónicos

Keywords: Keywords: 3D audio, latency, Bela, inertial sensor, I2C, OSC, embedded system, augmented reality.

## Abstract

Nowadays, the technologies that involve virtual and augmented reality (extended reality) are growing: Apple has just launched a device of this type. The generation of three-dimensional audio is a challenge for any application since it requires specific hardware and software that meet the demanding requirements of audio processing.

This document will detail the design of a system that detects its own orientation and renders audio in collaboration with the End of Degree Project prepared by S. Florido [1]. For this, a Bela mini board has been used, which needs a PocketBeagle board, and the absolute orientation sensor BNO055. The entire system will be complemented with a stacked accessory board that adds various controls to the device: potentiometers for volume and audio track controlling, a button used to set the reference point, and an LED diode. In addition, a case is included for the device, which will rest on a headset's headband.

In order to study the viability, a test system has been designed to measure the latency of the device. Said system consists of a polystyrene mannequin head that can rotate on itself, a real event generator and an oscilloscope: when a device is intended to be tested, the time difference between the real event and the event generated by the system will be measured. Finally, other systems will be tested in order to compare the results obtained.

# Agradecimientos

Me gustaría, en primer lugar, agradecer a mis tutores la ayuda y el apoyo brindado durante la elaboración de este documento. Sin su consejo y sabiduría hubiese tardado mucho más en conseguir mis objetivos.

Agradecer también al resto de profesores que me he encontrado durante estos cinco años de carrera. Mi paso por todas sus clases me ha permitido cambiar mi forma de pensar y ver el mundo de otra manera. Ahora me queda un último paso para cumplir con mi sueño de ser ingeniero.

A los amigos que hice durante el camino. En concreto, a mi compañero Sergio, con el que he vivido muchas batallas y con quién he compartido las vivencias de este Trabajo Fin de Grado. Ha sido un viaje muy enriquecedor y se ha forjado una amistad que perdurará en el tiempo.

Y por supuesto a mi familia, los cuales han tenido que 'soportar' mis incontables días/noches de estudios, mis cabreos y con quienes he celebrado mis victorias. A mis padres por estar constantemente ahí, ayudándome y aconsejándome, y a mi prometida María, la cual ha estado a pie de guerra durante estos años. Ella ha sido la primera 'afectada' durante este proceso, y ha tenido una paciencia formidable.

# Contenido

# Índice de figuras

# Índice de tablas

# Lista de Acrónimos

Adjunto la siguiente lista de acrónimos para facilitar su consulta en caso de ser necesario.

| | |
|---|---|
| RV/RA | Realidad Virtual / Realidad Aumentada |
| RX | Realidad eXtendida |
| ITD | *Interaural Time Difference* o diferencia de tiempo interaural |
| ILD | *Interaural Level Difference* o diferencia interaural de nivel |
| HRIR | *Head Related Impulse Response* o respuesta al impulso relacionada con la cabeza |
| HRTF | *Head Related Transfer Function* o función de transferencia relacionada con la cabeza |
| TFG | Trabajo Fin de Grado |
| IDE | *Integrated Development Evironment* o entorno de desarrollo integrado |
| E/S | Entrada / Salida |
| I2C | *Inter-Integrated Circuit*, abreviatura de Inter-IC |
| SDA | *Serial DAta* o pista de datos serie |
| SCL | *Serial CLock* o pista de reloj serie |
| PRU | *Programmable Real-Time Unit* o Unidad programable de tiempo real |
| DHCP | *Dynamic Host Configuration Protocol* |
| 9-DOF | *9 Degrees of Freedom* o 9 grados de libertad |
| IMU | *Inertial Measurement Unit* o Unidad de Medida Inercial |

| | |
|---|---|
| POR | *Power-on reset* o reset al encendido |
| DPS | *Degree per Second* o grados por segundo |
| MSW | *Mode SWitch* o cambio de modo |
| GPIO | *General Purpose Input/Output* o entrada/salida de propósito general |
| PLA | *PolyLActic acid* o ácido poliláctico |
| OSC | *Open Sound Control* |
| MIDI | *Musical Instrument Digital Interface* |
| UDP/IP | User Datagram Protocol / Internet Protocol |
| BiTa | *Binaural Test aplicaction* |
| VAE | *Virtual Acoustic Environments* o entornos acústico/virtuales |
| NIR | *Near Infra-Red* o infrarrojo cercano |
| DAQ | *Data AQquisition* o sistema de adquisición de datos |
| LDR | *Light Dependent Resistor* o fotorresistencia |
| BOM | *Bill Of Materials* o lista de materiales |

# Capítulo 1. Introducción

Desde el comienzo de la vida, los seres vivos han utilizado diferentes transductores para poder relacionarse con el medio que habitan. El fin está claro, cumplir con los objetivos básicos de perpetuación de la especie. Podríamos afirmar, que gran parte de los recursos se emplean en sobrevivir: el sentimiento de supervivencia es uno de los principales componentes en el comportamiento de cualquier organismo viviente.

El ser humano ha ido evolucionando desde habitar cuevas a construir grandes edificaciones. En los campos de conocimiento científicos como la física y las matemáticas se suceden descubrimientos nuevos casi a diario, lo que conlleva a un desarrollo constante y mantenido de la tecnología. La sociedad ha experimentado una evolución sin precedentes dentro del último siglo, pero este hecho es aún más llamativo en los últimos veinte años.

Entre las tecnologías que han surgido recientemente se encuentran todas aquellas que engloban el mundo digital en 3D. Dentro del mismo, nos vamos a enfocar en el denominado dúo RV/RA (Realidad Virtual, Realidad Aumentada), también conocido como RX (Realidad eXtendida) y, más concretamente, en el mundo de la realidad aumentada y cómo podemos integrarla con el tratamiento del audio.

Si volvemos a los tiempos prehistóricos, el sonido formaba (y sigue formando a día de hoy) un componente fundamental en el desarrollo de la capacidad de supervivencia de cualquier ser vivo. El control del sonido y la creación de música han estado acompañando al ser humano en todos y cada uno de los pasos que éste ha dado durante su existencia: rituales, himnos, música o instrumentos son algunos ejemplos de la evolución del mismo.

Actualmente, el sonido sigue teniendo un papel fundamental en nuestra sociedad, sobre todo en forma de audio digital.

Cuando hablamos de audio y tecnología la relación está clara: existen numerosos elementos *software* y *hardware* que los unen. Sin embargo, ¿qué tiene que ver el mundo de la realidad aumentada con el audio digital?

Es aquí donde entra en juego el presente proyecto, cuya memoria describirá cómo, partiendo de una biblioteca de audio 3D pública desarrollada por el grupo de investigación DIANA, de la Universidad de Málaga, se crea un prototipo de auriculares de diadema que permite detectar el movimiento de la cabeza (función *tracker*). Dicho dispositivo permitirá modificar el audio en consonancia para que dé sensación de que se escuche un objeto fijo (en contraposición de lo que ocurre normalmente, esto es que el objeto seguiría el movimiento de la cabeza), creándose así una sensación de audio 3D más rica y realista.

## 1.1. Contexto tecnológico

Para continuar, dividiremos la presente sección en varias partes diferenciadas: estudio fisiológico, estudio económico/social y estudio tecnológico.

### 1.1.1. Estudio fisiológico

Ambos sistemas auditivos (derecho e izquierdo) se dividen en tres etapas: oído externo, oído medio y oído interno.

- Oído externo: compuesto por el pabellón auricular (comúnmente denominado oreja, el cual es exclusivo de cada persona) y el conducto auditivo externo (nexo de unión entre el oído externo y el medio).
- Oído medio (o cavidad timpánica): empieza con el tímpano, sigue con la cadena de huesecillos (en orden: martillo, yunque y estribo) y termina con la trompa de Eustaquio.
- Oído interno: formado por la cóclea (donde se encuentran las terminaciones nerviosas auditivas), el vestíbulo (lugar de los receptores del equilibrio) y los conductos semicirculares. El estribo se une a la cóclea mediante la ventana oval.

Su funcionamiento se basa en la transformación de ondas sonoras en señales eléctricas: de forma muy resumida, el pabellón auditivo ayuda a redirigir las ondas sonoras hacia el interior, donde viajan a través del oído externo hasta accionar el tímpano. Ahí, las vibraciones se trasmiten por la cadena de huesecillos. La presión del sonido se transmite al líquido del oído interno a través de la estimulación que hace el estribo sobre la ventana oval.

La cóclea, con aspecto de caracol, está enrollada sobre sí misma. Si la desenrolláramos podríamos observar que está formada por una membrana elástica denominada membrana basilar, la cual contiene las células ciliadas (son largas proyecciones con forma de apéndice). De esta manera, la onda de presión viajará por dicha membrana y la deformará en una zona concreta, dependiendo de la frecuencia del sonido (frecuencias altas deformarán la zona basilar o más cercana a la ventana oval y las bajas frecuencias la más alejada). Por último, al moverse dichas células se estimulan unas proyecciones microscópicas denominadas cilios, los cuales se encargarán de transmitir el impulso eléctrico a través del nervio auditivo.

Como podemos observar, nuestro sistema auditivo es capaz de detectar el tono del sonido a través de la frecuencia (cada punto de la membrana se comporta como un filtro paso banda), así como su intensidad (dependiendo de la agresividad con la que se deforma). Por último, el timbre se detecta mediante el análisis de las diferentes frecuencias que componen el sonido que se ha transmitido.

En el ámbito de esta aplicación tiene especial relevancia el oído externo, en particular los pabellones auditivos. Las orejas ejercerán un papel fundamental a la hora de localizar la proveniencia de un sonido en el espacio, tal y como comprenderemos a continuación.

Adoptando un enfoque más técnico, tal y como se expone en [2] y [3], los seres humanos captamos el sonido utilizando la denominada escucha binaural. Es decir, nuestro cerebro percibe el sonido a través de las dos orejas, lo analiza y lo compara para detectar cuál es su origen dentro de un espacio tridimensional. Para ello, usamos de forma combinada varios mecanismos:

- ITD (*Interaural Time Difference*) o diferencia del tiempo interaural, es decir, la diferencia de tiempo con la que un mismo sonido alcanza a ambas orejas.
- ILD (*Interaural Level Difference*) o diferencia interaural de nivel, es decir, la diferencia en intensidad (volumen) que se percibe en un oído frente al otro a partir del mismo sonido. El ILD también se utiliza para determinar la distancia.

Además de los comentados, es importante destacar un mecanismo dependiente de cada oreja:

- Indicios monoaurales [4]. Son aquellos producidos por nuestros pabellones auriculares por separado. En este caso, la forma de la propia oreja, única para cada persona, junto con el conducto auditivo externo, serán los protagonistas: dichos elementos actuarán como filtros dependientes de la dirección, que modificarán el sonido según su naturaleza. A estos filtros se les llama indicios espectrales, dependen de la elevación del sonido y funcionan modificando su espectro.

Tanto el ITD como el ILD combinados nos permiten distinguir un objeto según su ángulo de incidencia respecto al eje interaural (azimut interaural), pero puede darse el caso en que diferentes sonidos estén en un mismo cono formado por un determinado azimut interaural, lo que dificultará la correcta detección de la orientación del sonido. En este ámbito, tiene especial relevancia el plano medial, el cual nos complicará (aún más) la percepción del sonido a la hora de determinar si éste proviene de un posición anterior o posterior a nuestro cuerpo.

En estos casos, nuestra respuesta natural es mover la cabeza: si no lo hacemos existe cierta incertidumbre en la localización del sonido, debido a los conos de confusión. Esto es debido a que el ITD y el ILD son mucho más potentes que los indicios espectrales. Ahora bien, a la hora de mover la cabeza, lo que antes sólo se podía resolver mediante indicios monoaurales ahora se desambigua porque hay cambios en los interaurales.

Esto hace que al renderizar audio binaural, el hecho de tener en cuenta el movimiento de la cabeza aporte mucha más riqueza y realismo al audio 3D.

Aunque es la forma de las orejas las que tienen especial relevancia, además de lo expuesto, también podemos utilizar la forma de nuestro propio cuerpo para modificar el sonido: nuestro torso superior, nuestros brazos y nuestros hombros provocan el rebote de las ondas sonoras, es decir, también actúan como filtros. Mención especial merece la oposición que presenta la cabeza al paso del sonido, la cual genera también una serie de parámetros que se pueden estudiar:

- HRIR (*Head Related Impulse Response*) o respuesta al impulso relacionada con la cabeza, la cual describe cómo se modifica un sonido que viene de una dirección específica.
- HRTF (*Head Related Transfer Function*) o función de transferencia relacionada con la cabeza es la respuesta que caracteriza cómo un oído recibe un sonido desde un punto del espacio. Consiste en el conjunto de HRIR que recoge todas las direcciones alrededor de una persona.

Es importante destacar que, hablando con propiedad, el HRIR debería estar asociado al dominio del tiempo y el HRTF al de la frecuencia, pero históricamente se ha tratado ambos parámetros como se ha descrito en el ámbito en el que se desarrolla este documento.

En definitiva, nuestro sistema auditivo es un mecanismo muy complejo del que sabemos aún muy poco. Por otro lado, la forma exclusiva que presenta el pabellón auditivo de cada persona junto con la forma de su cuerpo y su cabeza hacen que la experiencia sonora sea muy personal. Ambas características suponen un reto a la hora de diseñar sistemas de audio que se asemejen a la realidad.

### 1.1.2. Estudio económico/social

Tal y como se explica en el artículo [5], el término de realidad aumentada nace tras el de realidad virtual. La RA se refiere a aquella tecnología que permite la visualización del mundo real con información añadida. Dicho término surgió, en realidad, en 1994 gracias a P. Milgram [6]. A partir de ahí, se han desarrollado dispositivos relacionados, como las HoloLens de Microsoft (2016), aunque el crecimiento ha sido muy lento desde entonces.

Aunque se espera cierto despegue (las Apple Glasses acaban de anunciarse), actualmente va de la mano de la realidad virtual, y es por ello por lo que hablamos de RV/RA cuando nos referimos a este tipo de tecnologías.

Por otro lado, el virus del COVID-19 produjo un gran cambio a nivel social y económico. Antes del 2020 el mercado de la realidad virtual estaba comenzando a despegar (gracias a Oculus), pero durante el confinamiento, la actividad a nivel digital creció notablemente. El teletrabajo y las conferencias en remoto estuvieron a la orden del día y numerosas empresas florecieron ofreciendo este tipo de servicios.

Ya en 2014 se estudiaban sistemas inmersivos para conferencias enfocados a la participación de varias personas utilizando dispositivos móviles [7]. Es en este punto donde tomamos el testigo para este trabajo fin de grado: buscamos crear un sistema de realidad aumentada que permita asistir a una conferencia de forma telemática, pero escuchando a los diferentes participantes como si estuvieran sentados alrededor nuestro.

En la actualidad, pese a tener que enfrentarse a numerosos obstáculos (como la falta de microcontroladores provocada por diversos problemas logísticos y de producción), el mercado RV/RA está en auge y prevé generar una gran cantidad de ingresos en los próximos años. Analistas como Statista ofrecen informes (de pago) bien detallados sobre dicho tamaño de mercado a nivel mundial y sus previsiones. Podemos ver un ejemplo en [8].

### 1.1.3. Estudio tecnológico

Tras la búsqueda bibliográfica inicial podemos mencionar varios dispositivos que implementan una funcionalidad parecida a la que nos proponemos: existen dispositivos controlados por diferentes microcontroladores, que emplean diversas tecnologías y que procesan el audio de una forma u otra.

Como punto de partida podemos tomar el dispositivo mostrado en [9], donde se diseñó y evaluó un *tracker* cefálico de bajo coste, vestible, para síntesis binaural. Dicho dispositivo utilizaba una placa Arduino Nano como microcontrolador (junto con una interfaz USB/serie) y el sensor BNO055 de Bosch, el cual dispone de un giroscopio, un acelerómetro y un sensor geomagnético.

El procesado de la señal se realizaba en un ordenador. Tras realizar el análisis, se observó que la capacidad de seguimiento de los movimientos podía competir con un sistema de seguimiento óptico (externo) y que, además, le permitía síntesis binaural. No obstante, realmente dicho artículo se centraba en estudiar el seguimiento del movimiento más que en la conjunción con la síntesis de audio.

En nuestro caso, buscamos diseñar un dispositivo similar, pequeño y vestible, que permita detectar el movimiento de la cabeza y pueda sintetizar el audio en tiempo real por sí solo, sin necesidad de comunicarse con una unidad con más capacidad de procesamiento, como por ejemplo un ordenador.

Todo el procesamiento para posicionar el audio en tres dimensiones ya ha sido estudiado por el grupo de investigación DIANA de la Universidad de Málaga, proporcionando como resultado una librería C++: la **3D Tune-In Toolkit**. Podemos leer su artículo según [2] o visitar su repositorio en [10].

Como comentamos anteriormente, en un sistema de audio convencional, cuando giramos la cabeza la fuente de sonido girará con nosotros. Usando esta librería podemos fijar un sonido en un punto del espacio, de manera que cuando movamos la cabeza dicho punto no se mueva (como si fuese un altavoz virtual fijo) y genere la sensación de estar escuchándolo en un espacio abierto y real.

La plataforma elegida para soportar dicho sistema será una placa '**BeagleBone Black'** [11], la cual posibilita una mayor capacidad de procesamiento. Esta placa utiliza un procesador ARM Cortex-A8 que permite velocidades de hasta 1 GHz.

Además, otro motivo por el cual elegimos esta plataforma es porque posibilita añadirle el complemento '**Bela**' [12]. Éste es capaz de procesar con una latencia ultra baja los datos de audio y sensores usando un sistema empotrado. Dispone de audio estéreo y ocho canales de 16 bits, tanto CAD (Conversor Analógico Digital) como CDA (Conversor Digital Analógico) para los sensores y actuadores.

Llegados a este punto es importante comentar que el del presente trabajo se ha realizado de forma conjunta con S. Florido. Su documento [1] versará sobre el software de la aplicación (uso de la librería y despliegue en la placa), mientras que el presente documento tratará sobre el uso del sensor, la comunicación y el hardware que lo engloba.

El procedimiento organizado que nos permite acercarnos, paso a paso, a nuestro objetivo final es:

1. Empezaremos familiarizándonos con la librería 3DTI Toolkit y con la placa BeagleBone Black y Bela. Este punto es común a ambos documentos.
2. Tras esto, nos preocuparemos por incorporar y compilar una versión reducida de la librería en la plataforma elegida, siendo éste el trabajo principal del trabajo fin de grado asociado [1].
3. Luego crearemos un diseño provisional del circuito y lo implementaremos en una placa de pruebas. Dicho hardware se utilizará para dotar al sistema de una interfaz sensor/Bela/usuario e incluirá una serie de componentes electrónicos para dicho propósito (dos potenciómetros, un botón y un diodo LED).
4. Diseñaremos una carcasa que sea adecuada a nuestro dispositivo. Con ella mejoraremos la seguridad y evitaremos movimientos indeseados.
5. Para acabar, estudiaremos la latencia utilizando herramientas diseñadas para este mismo objetivo.

Una vez conseguido el prototipo inicial buscaremos mejorar el dispositivo, aumentando la comodidad, creando una placa de circuito impreso y reduciendo el tamaño del mismo.

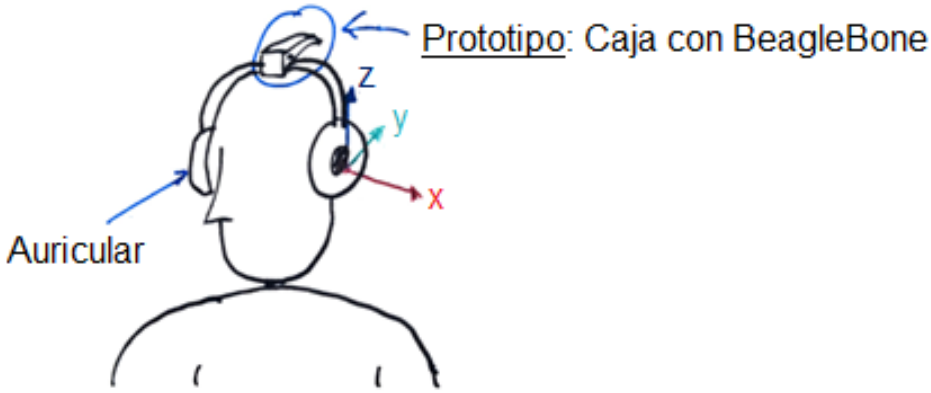


**Figura 1. Boceto del prototipo cefálico.**

## 1.2. Objetivos del Trabajo Fin de Grado

Este documento forma parte de un proyecto más amplio formado por dos trabajos fin de grado (TFG a partir de ahora). El objetivo general del proyecto es el de crear un prototipo de *tracker* cefálico que permita la reproducción de audio 3D, usando para ello la librería 3DTI Toolkit y la plataforma BeagleBone Black con Bela.

Mientras el TFG asociado [1] estudiará el despliegue de la librería en la plataforma, este documento versará sobre el apartado hardware, el diseño del controlador del sensor utilizado en la aplicación y el estudio de las características de dicho prototipo, evaluando la latencia y viabilidad del mismo.

## 1.3. Estructura de la memoria

Antes de entrar en materia, ofrecemos una breve guía sobre cómo está estructurada dicha memoria. Para comenzar, en el capítulo 2, empezaremos estudiando cuáles son las especificaciones del sistema y fijaremos los requisitos necesarios. Podremos estudiar una tabla de requisitos más detallada en el Apéndice C.

En el capítulo tres, procederemos a comentar cómo se ha desarrollado la actividad del proyecto: mostraremos la evolución del mismo, tanto a nivel de software como de hardware, a través de una serie de explicaciones, detalles e imágenes ilustrativas.

Pasaremos al capítulo de pruebas y evaluación, en el cual se mostrará qué procedimientos hemos utilizado para verificar la correcta funcionalidad del dispositivo, esto es, un valor de latencia aceptable para una aplicación de audio. Se realizará, además, una comparación de nuestro dispositivo con otros sistemas. Una vez el desarrollo y las pruebas, reuniremos todos los resultados en un capítulo final, en el cual se expondrá las conclusiones. También se incluirá un análisis de las posibles líneas futuras que quedan pendientes para seguir investigando con este dispositivo.

Por último, daremos paso a los apéndices (lista de materiales, tabla de requisitos, manual de uso, resumen de circuitos) y a la enumeración de las referencias bibliográficas utilizadas.

# Capítulo 2. Especificaciones del sistema

Muy resumidamente, esta podría ser una descripción de los requisitos de nuestro sistema:

La aplicación debe captar el movimiento de la cabeza del usuario para poder renderizar audio en consecuencia, respetando su tridimensionalidad. El usuario podrá girar, ladear o inclinar la cabeza con total libertad. Dichos movimientos serán captados por un sensor, serán transformados en señales digitales y se transferirán a la placa de desarrollo, donde el software diseñado realizará los cálculos pertinentes para generar el sonido buscado. El usuario podrá controlar ciertas variables del sistema, como el volumen o la pista de audio. Por otro lado, el sistema debe colocarse en una posición cómoda y segura para el usuario.

Además del modo de funcionamiento descrito, el sistema será sometido a una serie de pruebas en un entorno controlado: el sistema debe tener una latencia aceptable para el tratamiento del sonido.

Veamos ahora esto de una forma más detallada.

## 2.1. Requisitos de la aplicación

### 2.1.1. Requisitos fundamentales

**R1**. El dispositivo debe ser capaz de detectar su propia orientación, utilizando para ello un sensor inercial que permita su implementación en un sistema global. Dicha medida debe ser rápida y fiable, y debe permitir detectar cualquier ángulo de movimiento que una persona en bipedestación o sentada pudiera realizar con su cabeza.

**R2**. El dispositivo debe, utilizando la información de orientación del sensor, calcular y sintetizar la información de audio en consecuencia. Para evitar perder el realismo de la simulación, el cálculo debe ser rápido. Además, debe existir una salida de audio que permita la reproducción del mismo.

**R2.1**. El sistema debe utilizar la placa BeagleBone Black (en su versión normal o mini) como plataforma sobre la que se desarrollará el sistema.

**R2.2.** La placa BeagleBone Black debe ser complementada con la extensión Bela, la cual ha sido diseñada para el renderizado de audio con ultra baja latencia.

**R2.3.** El sistema debe utilizar el 3D Tune-In Audio ToolKit diseñado por el grupo DIANA. Este requisito es específico del sistema al completo, pero será tratado en el TFG asociado [1].

**R3.** Debe establecerse una comunicación entre el sensor y Bela.

**R3.1.** Debe existir una comunicación software implementada a través de un controlador.

**R3.2.** El sistema debe incluir una conexión física estable que fije el sensor a la placa y evite movimientos indeseados.

**R4.** El usuario debe poder controlar ciertas variables del sistema, tales como la amplitud de la señal, la pista de audio y la posibilidad de fijar un punto de referencia como cero. Para ello se diseñará un hardware específico.

### 2.1.2. Requisitos de Tolerancia a fallos y Seguridad

**R5.** El sistema debe ir contenido en una carcasa diseñada para tal propósito:

**R5.1.** Se debe garantizar la seguridad del usuario en todo momento, evitando todo contacto con cualquier componente electrónico, exceptuando las conexiones externas.

**R5.2.** La carcasa debe permitir un funcionamiento óptimo del dispositivo, asegurando la ventilación de éste y ofreciendo un agarre adecuado que evite posibles movimientos indeseados.

**R5.3.** La carcasa debe contar con aberturas para posibles conexiones (USB, mini USB y Jack), además de las utilizadas para los controles.

**R5.4.** La carcasa debe contar con elementos que permitan utilizar adecuadamente los controles, tales como embellecedores o extensores de botones.

**R6.** El dispositivo debe incluir un diodo LED que indique su estado.

### 2.1.3. Requisitos de Usabilidad

**R7.** El dispositivo debe considerarse vestible. Para ello, el dispositivo debe poder colocarse sin mayor dificultad sobre la diadema de los auriculares que esté utilizando el usuario.

**R7.1.** El dispositivo debe ser lo más cómodo posible: debe tener un tamaño adecuado y no debe pesar demasiado.

**R7.2.** El dispositivo debe ser autónomo: debe poder funcionar por sí mismo, sin necesidad de conectarse a un ordenador o a una toma de luz.

**R8.** En caso de desearse, el dispositivo debe poder mantener una comunicación con un ordenador considerado host. Para ello se deben utilizar mensajes de audio en formato OSC, los cuales contendrán información sobre la orientación de la cabeza.

## 2.2. Requisitos relacionados con la latencia

**R9.** El dispositivo debe tener una latencia conocida, la cual debe estar por debajo de un máximo aceptable para aplicaciones de procesado de audio.

**R10.** Para cumplir con el requisito 10, se debe diseñar un sistema que permita medir los valores de latencia del dispositivo.

**R10.1.** El sistema de pruebas debe sostener adecuadamente el dispositivo a testear (esto es, unos auriculares con diadema).

**R10.2.** El sistema de pruebas debe contar con un disparador de evento real que nos sirva de punto de referencia para realizar la medida. Dicho disparador debe ser rápido.

**R10.3.** El sistema de pruebas debe incluir un medidor que nos permita comparar el evento real con la respuesta del dispositivo medido.

**R11.** Para tener más información del dispositivo se deben comparar sus medidas con las obtenidas de otros sistemas.

# Capítulo 3. Desarrollo del sistema

Estudiaremos el desarrollo del sistema paso a paso, siendo fieles a la línea temporal de realización de este trabajo, desgranando en apartados todos los puntos importantes.

Según orden de realización: comenzaremos estudiando qué es Bela, qué es la BeagleBone Black y el motivo de utilizarlos en conjunto. Continuaremos, descubriendo el sensor elegido, concretamente el BNO055. Lo describiremos a nivel de hardware estudiando sus características y registros, apoyándonos en la hoja de características del mismo.

Una vez comprendidos la base y su funcionamiento, nos enfocaremos en la creación de un controlador software que permita la comunicación entre la placa y el sensor, utilizando para ello el protocolo I2C. Esto nos permitirá dar el salto a un software de más alto nivel que incluya el 3D Tune-In Toolkit, para renderizado de audio. Dicho software necesitará recibir la información sobre orientación en una unidad específica: los cuaterniones.

Conseguido nuestro objetivo, procederemos con el diseño del hardware que acompañará al dispositivo. Incluiremos una serie de controles para controlar distintas variables del sistema (amplitud o volumen, pista y punto de referencia), además de un indicador de estado (diodo LED).

Por último, nos enfrentaremos al problema mecánico que supone fijar el aparato resultante de forma estable y segura, al objeto sobre el que lo queramos colocar (unos auriculares de diadema). Para ello crearemos una carcasa utilizando software de diseño gráfico y la imprimiremos utilizando una impresora 3D.

## 3.1. Complemento 'Bela'

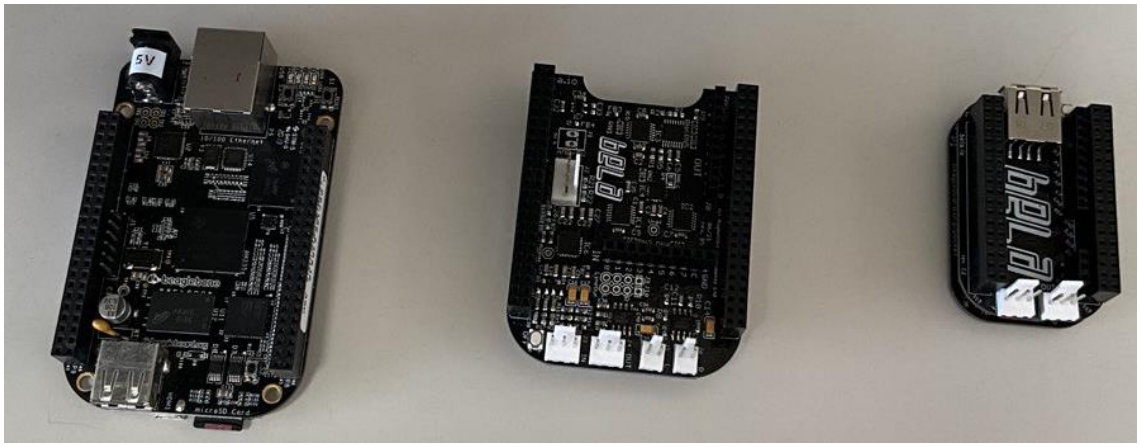

**Figura 2. Detalle de las placas utilizadas: de izquierda a derecha: BeagleBone Black en formato original, Bela en formato original, Bela Mini sobre PocketBeagle.**

El eje angular de este Trabajo Fin de Grado es utilizar la placa Bela como procesador de los datos del sensor y del audio digital. Esto nos lleva a la siguiente cuestión: ¿qué es Bela?

Como ya comenzamos a esbozar en la introducción, Bela es una extensión que se coloca sobre una placa BeagleBone Black (a partir de ahora, se la denominará por las siglas BBB) de forma apilada. Permite procesar con una latencia ultra baja los datos de audio y sensores, usando un sistema empotrado. Además, está disponible en formato normal o 'mini', tal y como podemos observar en la Figura 2.

Una característica muy importante que engloba a Bela es su diseño bajo la premisa de las licencias de código abierto, lo cual permite utilizarla como deseemos, siempre y cuando respetemos sus términos. Fue diseñada por el *Augmented Instruments Laboratory* del *Centre for Digital Music* de la *Queen Mary University* de Londres. Actualmente el producto se encuentra en fase de actualización y mejora, de mano de la empresa *Augmented Instruments Ltd* localizada en Londres.

Bela aprovecha la CPU de la BBB para realizar el procesado: como comentamos en la introducción, nos referimos a un ARM Cortex-A8 con velocidad de 1 GHz junto a 512 Mb de memoria RAM tipo DDR3.

Volviendo a Bela, cuenta con ocho puertos de entrada analógica de 16 bits de resolución, 16 canales digitales E/S (Entrada / Salida), entrada USB y mini USB aprovechados de la BBB, además de dos puertos E/S de audio estéreo propios. Bela en formato original cuenta, además, con 8 canales analógicos de salida de 16 bits de resolución, dos conexiones para altavoces, entrada de alimentación, conexión ethernet y conector I2C directo. La diferencia entre el formato original y el mini se encuentra principalmente en el tamaño, siendo la segunda dos tercios más pequeña.

Respecto al software, utiliza un entorno propio basado en Linux y su extensión '**Xenomai**' para el renderizado de audio en tiempo real (el cual se carga a través de una tarjeta mini SD). Dicho software le permite utilizar búferes de datos de incluso dos muestras, lo que puede reducir la latencia a un milisegundo o menos. Para ello posee un controlador de audio propio que aprovecha la PRU (unidad programable de tiempo real) del sistema para evitar todo el procesado del sistema operativo y conectar directamente con el hardware. Además, la tasa de muestreo de los canales analógicos y digitales se configura automáticamente con la tasa de audio, lo que permite una relación precisa y libre de *jitter* entre el audio y los sensores.

Además, incluye un entorno de desarrollo integrado (IDE a partir de ahora) en el propio software al cual se puede acceder de forma sencilla y directa a través de navegador: al conectarse a un dispositivo por USB, Bela crea automáticamente una red virtual usando un servidor DHCP, luego asigna la dirección x.x.x.1 al host y la x.x.x.2 a sí misma.

De esta forma, si nos conectamos desde un dispositivo con Windows o Mac OS X 10.9 (o más antiguo) lo haremos usando la dirección 192.168.6.2 en nuestro navegador. En cambio, si utilizamos Linux o una versión Mac más moderna utilizaremos la dirección 192.168.7.2.

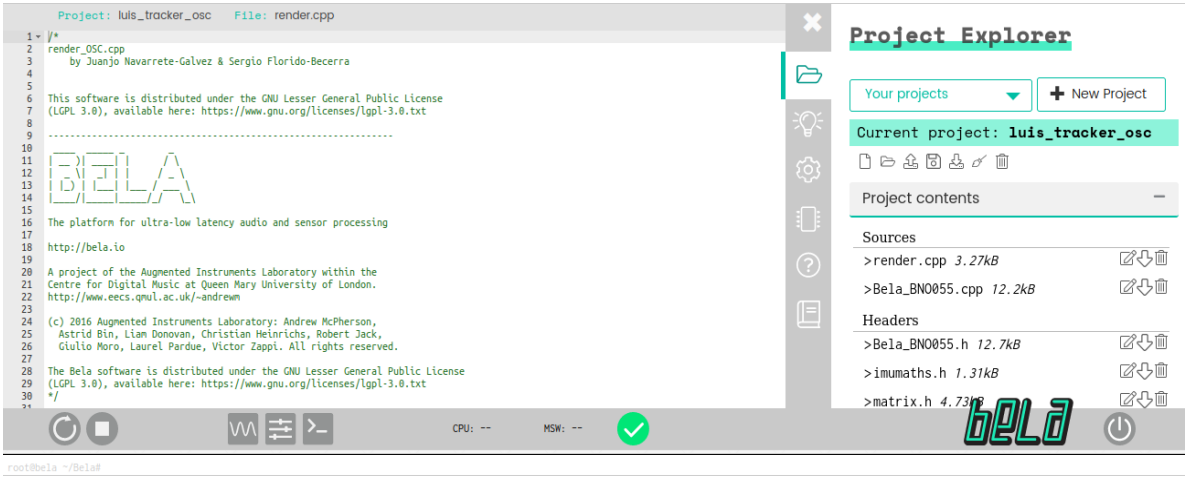


**Figura 3. IDE incluido con Bela.**

El IDE incluye varias aplicaciones y funciones: nos permite cambiar ciertas configuraciones de Bela, acceder a una gran cantidad de ejemplos e incluso al mapa de pines de la placa. Cuenta con un gestor de proyectos y una consola de comandos, la cual podemos utilizar como necesitemos. El propio entorno nos permite escribir nuestro código, el cual se compila en tiempo real de forma automática. Además, incluye un osciloscopio simulado en caso de que necesitemos visualizar alguna curva. La Figura 3 muestra un detalle del mismo.

Bela es compatible con lenguajes de programación como Pure Data y C++ entre otros. Centrándonos en C++, cualquier aplicación desarrollada en esta plataforma contará con tres funciones principales:

- `Setup`: donde se realizan las configuraciones iniciales pertinentes (por ejemplo, iniciar una hebra).
- `Render`: donde acontece el grueso del código. Es semejante al '*loop*' de Arduino, pero destaca porque es una función preparada para ejecutar código en ultra baja latencia. Esto es importante, pues ejecutar ciertas tareas lentas en este espacio puede dar lugar a errores.
- `Cleanup`: donde se realizan las tareas necesarias a la hora de terminar la ejecución.

Para trabajar con la placa se comenzó con la instalación del software y la preparación del sistema, cargando la imagen en una tarjeta microSD y actualizando el código nativo de la BBB. Una vez completados estos pasos, se cargó uno de los proyectos de ejemplo incluidos con la plataforma con el objetivo de aprender su comportamiento. Por último, dedicamos más tiempo a familiarizarnos con el IDE y el osciloscopio.

Esto nos lleva a completar los requisitos 2.1 y 2.2. Para más información sobre el entorno de desarrollo de Bela y su uso recomendamos visitar el TFG asociado [1].

## 3.2. Sensor inercial absoluto BNO055


**Figura 4. Detalle del sensor BNO055 sobre una placa de pruebas.**

El siguiente paso en el desarrollo de este trabajo fue el de elegir el sensor apropiado para nuestra aplicación: se buscó un sensor inercial que incluyera, como mínimo, un acelerómetro y un giróscopo. Entre los candidatos se encontraban el MPU6050 de InvenSense y el BNO055 de BOSCH, siendo éste el elegido principalmente por dos motivos: porque poseía mayor documentación y porque existía una mayor comunidad entorno al mismo. Podemos observar dicho sensor en la Figura 4.

El sensor inercial absoluto BNO055 incluye un acelerómetro, giróscopo y magnetómetro con compensación de temperatura, utilizando un ARM Cortex-M0 de 32 bits en el proceso. A la salida ofrece los datos a una tasa de hasta 100 Hz.

Es definido como 9-DOF (*9 Degrees of Freedom* o 9 grados de libertad), es decir, que incluye tres sensores con tres ejes o coordenadas cada uno, e IMU (*Inertial Measurement Unit* o Unidad de Medida Inercial), es decir, que es un sensor inercial absoluto que procesa los datos. Permite salidas en cuaterniones, ángulos de Euler, vectores de rotación, aceleración lineal, gravedad y temperatura.

Destacamos que aunque el sensor sea considerado 9-DOF, como sólo vamos a estudiar la orientación estamos realmente ante tres grados de libertad.

Concretando características:

- Acelerómetro: sensor triaxial de 14 bits. Permite rangos de configuración desde ±2 g a ± 16 g (fuerza G). Posee un filtro paso bajo configurable desde 1 kHz hasta menos de 8 Hz, diferentes modos de operación y señales de interrupción por movimiento.
- Giróscopo: sensor triaxial de 16 bits. Permite configurar el rango desde ±125 °/s a ±2000 °/s (grados por segundo). Posibilidad de filtro paso bajo entre 523 Hz y 12 Hz. Diferentes modos de operación y generación de señales de interrupción por movimiento.
- Magnetómetro: sensor geomagnético con rango de campo magnético típico de ±1300 μT en ejes x e y, ±2500 μT en eje z (Tesla). Resolución de aproximadamente 0.3 μT. Ofrece diferentes modos de operación y de consumo.

Su alimentación es de 3 V y la comunicación con el host se realiza mediante protocolo I2C. El dispositivo ha sido diseñado para que cargue los valores por defecto al encenderse (POR, *Power-on reset* o reset al encendido) iniciándose en modo configuración.

En cuanto a modos de consumo, ofrece un modo normal (todos los sensores habilitados según el modo de operación elegido), uno de bajo consumo (si no se detecta movimiento durante un tiempo configurable el BNO055 entrará en modo de bajo consumo, dejando sólo el acelerómetro activado) y otro de suspensión (todos los sensores y la CPU están ‘dormidos’). En nuestra aplicación utilizaremos el modo normal continuamente.

Respecto a los modos de operación, disponemos de siete configuraciones sin tratamiento de datos (*non-fusion modes*), los cuales ofrecen la salida sin tratar de los diferentes sensores. Los modos difieren entre sí dependiendo del sensor que decidamos activar, pero permite una configuración total.

Además, el BNO055 ofrece cinco configuraciones con tratamiento de datos (*fusion modes*) entre los que elegir, dependiendo de qué sensores queremos utilizar y qué tipo de orientación necesitemos: relativa o absoluta.[1]

En nuestra aplicación utilizaremos el **modo IMU**, el cual fusiona la información tomada por el acelerómetro y el giróscopo para ofrecer datos de orientación de tipo relativo. Dicho modo permite una tasa de datos de salida de 100 Hz en el acelerómetro, 100 Hz en el giróscopo y 100 Hz en los datos inerciales. Dichos datos pueden ser configurados para que se ofrezcan en dos unidades diferentes: ángulos de Euler (en ángulos de navegación) o cuaterniones.

Además de los modos de operación, podemos configurar otros aspectos, como la orientación (signo), la calibración o las unidades de salida. Concretamente, destacamos que la unidad por defecto de aceleración son los grados, la de velocidad angular los DPS (*Degree per Second* o grados por segundo) y la temperatura en grados Celsius. Además, permite seleccionar el formato de salida del ángulo de rotación según trabajemos en Windows o en Android.

Toda la información contenida en este apartado ha sido obtenida a través de dos fuentes principales: la página oficial del distribuidor Adafruit, disponible en <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview>, y la hoja de características oficial del sensor [13].

Con la elección del sensor, el BNO055, se ha completado parte del primer requisito estipulado (R1).

[1] Los '*fusion mode*' son los que están propiamente pensados para calcular la orientación del dispositivo en el espacio analizando los datos de los sensores. Pueden ser de tipo relativo (los resultados del sensor dependerán de la posición inicial de los mismos) o absoluto (la orientación del sensor se realizará con respecto a la Tierra y su campo magnético).

## 3.3. Controlador del sensor

Una vez elegido el sensor que se utilizará para la aplicación y tras el primer contacto con la placa utilizada (BBB y Bela), ahora es el momento de comenzar a diseñar un controlador (*driver*) software que permita la comunicación entre ambos. Para ello observamos que el BNO055 ofrece una serie de conexiones para realizar una comunicación de tipo I2C.

Comenzaremos recordando cómo funciona este tipo de conexión: el I2C es un protocolo de comunicación serie muy popular de tipo maestro – esclavo (es posible aumentar el número de esclavos). Permite la comunicación entre los diferentes dispositivos mediante cuatro pistas:

- VCC: pista para la alimentación del dispositivo.
- GND: pista de tierra común a todos los dispositivos conectados.
- SDA (*Serial DAta* o pista de datos serie).
- SCL (*Serial CLock* o pista de reloj serie).

El maestro es el encargado de gestionar la comunicación entre los diferentes dispositivos generando una señal de reloj a través de la pista SCL. Mediante la pista SDA podrá transmitir una serie de mensajes a los diferentes dispositivos, los cuales tienen una identificación única.

El objetivo será interactuar con los mismos constatando qué registro del dispositivo esclavo se busca modificar o leer. Para ello se utilizan principalmente dos tipos de mensajes: de escritura o de lectura. Como es muy común que se necesite leer la información contenida en un registro en concreto, generalmente se diseña una función software que escribe y espera a leer de manera automática[2]. En la Figura 5 se puede observar un ejemplo de trama de datos de comunicación.

[2] El flujo sería: el maestro transmite un mensaje de escritura, indicando la identificación del esclavo y la dirección del registro a leer. Luego emite un mensaje de lectura dirigido a ese mismo dispositivo y se queda a la espera. Una vez recibidos los mensajes por parte del esclavo, el dispositivo devuelve al maestro un mensaje con los datos contenidos dentro del registro deseado.


**Figura 5. Trama de bits dentro del protocolo I2C. Fuente imagen: hetpro-store.com, disponible en <https://hetpro-store.com/TUTORIALES/i2c/>**

Otra característica importante de este protocolo es que se escribirán o leerán tantos bits según sea el tamaño del dato que utilicemos, leyendo los registros consecutivamente a partir de la primera dirección. Es decir, si queremos leer cuatro registros de un dispositivo que cuenta con registros de tamaño de 1 byte, el maestro tendrá que transmitir un mensaje en el cual se fija la dirección del primer registro a leer y un tamaño de 4 bytes. El esclavo devolverá la información contenida en los cuatro registros consecutivos, comenzando con el fijado en el campo de dirección.

El principal problema a la hora de diseñar un controlador para el BNO055 tiene que ver con la placa utilizada: generalmente, este sensor es utilizado en Arduino, apoyándose en una librería creada por la empresa Adafruit (este software está disponible en su repositorio <https://github.com/adafruit/Adafruit_BNO055> [14]). Dicha librería utiliza partes vitales del software de Arduino que no se encuentran en la programación de Bela, como la librería *wiring*.

Por ello, es necesario diseñar un código propio que permita utilizar dicho sensor en Linux y sin ninguna dependencia de Arduino. Durante el proceso nos apoyaremos en un proyecto realizado en 2017 denominado *Bela on ur head* (la traducción al castellano sería: Bela en tu cabeza) disponible en <https://github.com/theleadingzero/belaonurhead> [15]. Dicho proyecto utilizaba un BNO055 con Bela, con la diferencia de que el programa principal estaba diseñado con el lenguaje de programación Pure Data.

### 3.3.1. Descripción del controlador

El código del controlador se encuentra en el archivo *Bela_BNO055.cpp*, y va acompañado por su cabecera *Bela_BNO055.h*, ambos añadidos al repositorio adjunto. Éste depende a su vez de los ficheros de cabecera *I2c.h* y *Utilities.h* (forman parte del software nativo de Bela). Además, se crean dependencias con la *IMU maths Library* [16] (*imumaths.h*, de la cual derivan los ficheros *quaternion.h*, *matrix.h* y *vector.h*)[3]. Estos últimos ficheros permiten el cálculo y análisis matemático de cuaterniones, matrices y vectores. Fueron creadas por S. Cowen en calidad de código abierto. El uso que se hace de ellas es principalmente el de dar formato a los diferentes vectores que se utilizarán para almacenar los datos obtenidos de la lectura, según el tipo de unidad.

Tras estudiar y comprender el código de la librería de Adafruit como de *Bela on ur Head*, se procedió a crear un controlador para nuestra aplicación. El código ha sufrido una serie de variaciones a medida que se iba alcanzando la versión final, la cual puede ser consultada en el repositorio adjunto.

Su funcionamiento se basa en la creación de un objeto de la clase `I2C_BNO055`, de la cual se derivan una serie de funciones. Aprovechando la hoja de características del sensor y la información recogida en el fichero de cabecera, podemos conocer las direcciones de los diferentes registros disponibles, así como sus etiquetas. Esto nos permite tener un mejor control de los mensajes y un mayor conocimiento de la arquitectura del sensor.

[3] Para la realización de esta aplicación se actualizaron estas librerías a la última versión disponible: 2021

Procedemos a sintetizar, resumidamente, cómo se comportan las funciones más importantes, a partir de la clase `I2C_BNO055`, basada en [15]:

- `boolean I2C_BNO055::begin(uint8_t bus, uint8_t i2caddr)`: es la encargada de gestionar la inicialización del sensor. Al llamar a esta función, tras esperar unos segundos, pregunta por la identificación del BNO055. Se cambia a modo de configuración y se resetea el dispositivo[4]. Tras esperar de nuevo un tiempo, se cambia a modo de operación y se realiza la configuración inicial (en nuestro caso se configura el modo de consumo normal, aunque este es el punto donde se podría seleccionar el tipo de unidades, cambio del signo de ejes, etcétera). Por último, se cambia al modo IMU y se sale de la función.

- `void I2C_BNO055::setMode(i2c_bno055_opmode_t mode)`: utilizada para seleccionar el modo de operación que se necesite en cada momento.

- `void I2C_BNO055::setExtCrystalUse(boolean usextal)`: permite la selección de un cristal de cuarzo externo, en vez del oscilador interno (es necesario estar en modo configuración).

- `imu::Vector<3> I2C_BNO055::getVector(i2c_vector_type_t vector_type)`: función empleada para leer los datos de los diferentes registros. En la mayoría de las situaciones vamos a necesitar leer tres variables de 16 bits, es decir, un total de seis registros. Por ello, según el valor de `vector_type` podríamos leer datos del magnetómetro, giróscopo, acelerómetro, aceleración lineal, vector gravedad o grados Euler. Todos serán escritos en un vector facilitado por la librería *vector.h* en formato flotante. En la Figura 6 se puede observar la implementación de dicha función.

[4] Realmente, esto se realiza de forma automática al encender el dispositivo: se repite como medida de seguridad.

```
/**************************************************************************
    getVector
    Get sensor vector reading - using readLen
**************************************************************************/
imu::Vector<3> I2C_BNO055::getVector(i2c_vector_type_t vector_type) {
  imu::Vector<3> xyz;
  uint8_t buffer[6];
  memset(buffer, 0, 6);

  int16_t x, y, z;
  x = y = z = 0;

  // Read vector data (6 bytes)
  readRegisterLen((i2c_bno055_reg_t)vector_type, buffer, 6);

  x = ((int16_t)buffer[0]) | (((int16_t)buffer[1]) << 8);
  y = ((int16_t)buffer[2]) | (((int16_t)buffer[3]) << 8);
  z = ((int16_t)buffer[4]) | (((int16_t)buffer[5]) << 8);

  // Convert the value to an appropriate range and assign the value to the Vector type
  switch(vector_type) {
    case VECTOR_EULER:
      xyz[0] = ((double)x)/16.0;
      xyz[1] = ((double)y)/16.0;
      xyz[2] = ((double)z)/16.0;
      break;
  }

  return xyz;
}
```

**Figura 6. Detalle de la implementación de la función `getVector` (versión reducida).**

- `imu::Quaternion I2C_BNO055::getQuat(void)`: procesa los datos de los registros que contienen los cuaterniones. La propia naturaleza de este tipo de unidad nos impide utilizar la función `getVector`, pues al precisar de cuatro variables (x, y, z, w) se deben leer cuatro datos en vez de tres, es decir, ocho registros en vez de seis. Los datos serán transformados a doble flotante y escalados antes de utilizar la librería *quaternion.h* para formatearlos a un vector de cuaterniones.

- `uint8_t I2C_BNO055::readRegister(uint8_t reg)`: función clave del protocolo I2C utilizada para leer un byte de un registro del dispositivo esclavo (incluye la operación de escritura).

- `void I2C_BNO055::readRegisterLen(uint8_t reg, uint8_t *buf, uint8_t length)`: exactamente igual que la anterior pero permitiendo leer varios bytes.

- `void I2C_BNO055::writeRegister(uint8_t reg, uint8_t value)`: utilizada para escribir un número de datos concreto en una dirección dada.

```
/**************************************************************************
    readRegisterLen
    Reads from requested register a number of bytes (I2c.h)
**************************************************************************/
void I2C_BNO055::readRegisterLen(uint8_t reg, uint8_t *buf, uint8_t length){
    uint8_t buffer[1] = { reg };
    ssize_t bytes2Write = 1;
    ssize_t bytes2Read = length;

    ssize_t bytesWritten = writeBytes(buffer, bytes2Write);
    if(bytesWritten != bytes2Write) {
        rt_printf("ERROR - READ_LENGTH: Failed to write register %d on BNO055: written %d of %d bytes.\n",
            reg, bytesWritten, bytes2Write);
        return;
    }


    ssize_t bytesRead = readBytes(buf, bytes2Read);
    if (bytesRead != bytes2Read) {
        rt_printf("ERROR - READ_LENGTH: Failed to read register %d on BNO055: read %d of %d bytes.\n",
            reg, bytesRead, bytes2Read);
        return;
    }
}
```

**Figura 7. Detalle del código contenido en la función `readRegisterLen`.**

Se adjunta una tabla/resumen con la configuración final del sensor:

| Característica |
|---|
| Modo de consumo: normal. |
| Modo de operación: IMU. |
| Unidad de rotación Euler: grados. |
| Formato de los datos de salida: Windows. |
| Uso de cristal externo: sí. |

**Tabla 1. Configuración del sensor BNO055.**

### 3.3.2. Descripción del código de Bela

Una vez implementado el controlador del sensor, nos enfocaremos en analizar el código que utilizará Bela para enviar y recibir los mensajes de orientación. Para ello se el fichero de cabecera *Bela.h* (código nativo de Bela) y *Bela_BNO055.h* (el modificado para esta aplicación a partir de [15]).

La lectura de datos del sensor se ha realizado de forma concurrente utilizando una hebra denominada `loop` (como guiño a Arduino). Dicha hebra se encarga de leer los datos que se deseen (en nuestra aplicación, vectores Euler o cuaterniones) y de introducir el resultado en un arreglo de datos globales de tipo flotante con tres o cuatro posiciones (`float axis[3];` y `float salida_quat[4];`), dependiendo del número de variables necesarias. Por último, se espera un tiempo predeterminado (5 milisegundos) para repetir la lectura: si se hace demasiado rápido se producirán fallos de comunicación I2C.

Antes de continuar, hagamos un paréntesis para comentar una característica importante de Bela: los cambios de modo de Xenomai. Como comentamos en el apartado 3.1. Complemento 'Bela', esta placa posee una función principal denominada `render`, la cual está preparada para tratar audio con ultra baja latencia. Además, el propio IDE ofrece información sobre el rendimiento estimado de la CPU en tiempo real, como también el contador MSW o *Mode SWitch* (cambio de modo).

Gracias a Xenomai podemos utilizar dos modos de trabajo: primario y secundario. El primario es el modo en el que se usa el kernel del propio Xenomai y es el adecuado para aquellas tareas o hebras que requieran procesado de datos en tiempo real, como podría ser el procesado de audio. El modo secundario es el que se sirve del kernel de Linux, y está destinado para tareas que no requieran procesado en tiempo real o que sean menos prioritarias. Los cambios de modo entre primario y secundario (o viceversa) son contabilizados bajo el indicador MSW del IDE.

Como se puede aprender en el apartado técnico de la guía de Bela[5] los cambios de modo deben ser evitados, siendo admisibles hasta dos con el inicio del programa. Cuando estamos en modo secundario y queremos ejecutar una tarea en tiempo real, se realizará un cambio de modo al primario. Cuando estemos en el primario y necesitemos utilizar cualquier servicio que no sea de tiempo real, cambiaremos de modo al secundario.

He aquí la razón por la cual se utiliza una hebra para gestionar la comunicación mediante el protocolo I2C del sensor. La función `render` está considerada una función de procesado en tiempo real que se ejecuta en modo primario. En cambio, la lectura de datos por un protocolo I2C (que además utiliza una librería propia de Linux) no es considerada de la misma forma y se ejecuta en secundario.

[5] Concretamente en <https://learn.bela.io/using-bela/technical-explainers/performance-monitoring/>.

Si intentamos comunicarnos con el sensor de forma continuada dentro de la función `render`, el sistema irá saltando de modo primario a secundario y viceversa cada vez que intentemos leer, lo que se traduce en un indicador MSW con un valor muy elevado, un porcentaje de uso de CPU de valores en torno al 100% y una gran cantidad de errores en la consola del sistema.

Al aislar la lectura de datos del sensor en una hebra propia y utilizar variables globales para gestionar la información evitamos todos esos cambios de modo que generan errores. En la Figura 8 se puede observar en detalle el contenido de dicha hebra.

```
44 // Sleep time for auxiliary task
45 unsigned int gTaskSleepTime = 5000; // microseconds
46
47
48 void loop(void*) {
49     while(!Bela_stopRequested()) {
50         //imu::Vector<3> euler = bno.getVector(I2C_BNO055::VECTOR_EULER);
51         //axis[0] = euler.x();
52         //axis[1] = euler.y();
53         //axis[2] = euler.z();
54
55         imu::Quaternion quat = bno.getQuat();
56         salida_quat[0] = quat.w();
57         salida_quat[1] = quat.x();
58         salida_quat[2] = quat.y();
59         salida_quat[3] = quat.z();
60
61         usleep(gTaskSleepTime);
62     }
63 }
```

**Figura 8. Detalle del contenido de la hebra `loop`.**

Una vez comprendido por qué necesitamos utilizar una hebra en el código podemos continuar con el análisis: tras la definición de la función `loop` se encuentra la función `setup` de inicio. En ella se comienza inicializando el sensor BNO055, se ordena el uso del cristal externo, se arranca la hebra y se inicializan las variables globales a cero.

Por último, en la función render se visualizan los datos leídos por consola, utilizando para ello la función `rt_printf`, propia de Bela.

Gracias a este apartado se han completado los requisitos 1 y 3.1.

## 3.4. Software para el renderizado de audio

El siguiente paso lógico que se debe dar para cumplir con nuestro objetivo es el de utilizar la librería 3D Tune-In Toolkit del grupo DIANA para renderizar el audio. Es aquí donde este documento y el TFG asociado confluyen [1]: mientras ahí ha quedado reflejado cómo utilizar y optimizar la librería, en este documento se refleja el componente hardware y cómo se mide la latencia.

Un aspecto importante dentro de la comunicación entre el sensor y la librería es que ésta necesita recibir la información de orientación en **cuaterniones**.

Los cuaterniones son una unidad de medida de orientación y rotación en el espacio bastante compleja, utilizados por primera vez por el físico y matemático irlandés *Sir* William Rowan Hamilton (1805 – 1865, Dublín).

Se consideran una extensión de los números reales más amplia que los números complejos: mientras que en los últimos añadimos la unidad imaginaria $i$, en los cuaterniones utilizamos tres unidades: $i$, $j$ y $k$, tal que:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{1}$$

$$ij = -ij = k; \quad jk = -kj = i; \quad ki = -ik = j \tag{2}$$

(2) demuestra que la multiplicación no es conmutativa. Por otro lado, la forma de un cuaternión es

$$q = a + bi + cj + dk \tag{3}$$
$$\text{donde } a, b, c, d \in \mathbb{R}.$$

En honor a Hamilton, estos números son englobados por un conjunto denominado $\mathbb{H}$, tal que

$$\mathbb{H} = \{q = a + bi + cj + dk : a, b, c, d \in \mathbb{R}\} \subset \mathbb{R}^4. \tag{4}$$

Algebraicamente, si estudiamos el anillo $(\mathbb{H}, +, \cdot)$ se concluye que es de tipo no conmutativo (no abeliano) con elemento neutro.

Además, la representación vectorial de un cuaternión es

$$\vec{x} = (a, b, c, d), \tag{5}$$

siendo $\{1, i, j, k\}$ su base [17].

En nuestra aplicación utilizamos los cuaterniones como indicadores de rotación. Para ello utilizaremos cuatro variables concretas: $x$, $y$, $z$ y $w$. Dichas variables determinan unas coordenadas (las tres primeras) y el denominado operador cuaterniónico de rotación (la última).

En resumen, $q = xi + yj + zk$ representa un eje en el espacio y $w$ representa la cantidad de rotación alrededor de dicho eje.

Volviendo a nuestra aplicación, el flujo de funcionamiento del software será el mostrado en la Figura 9.

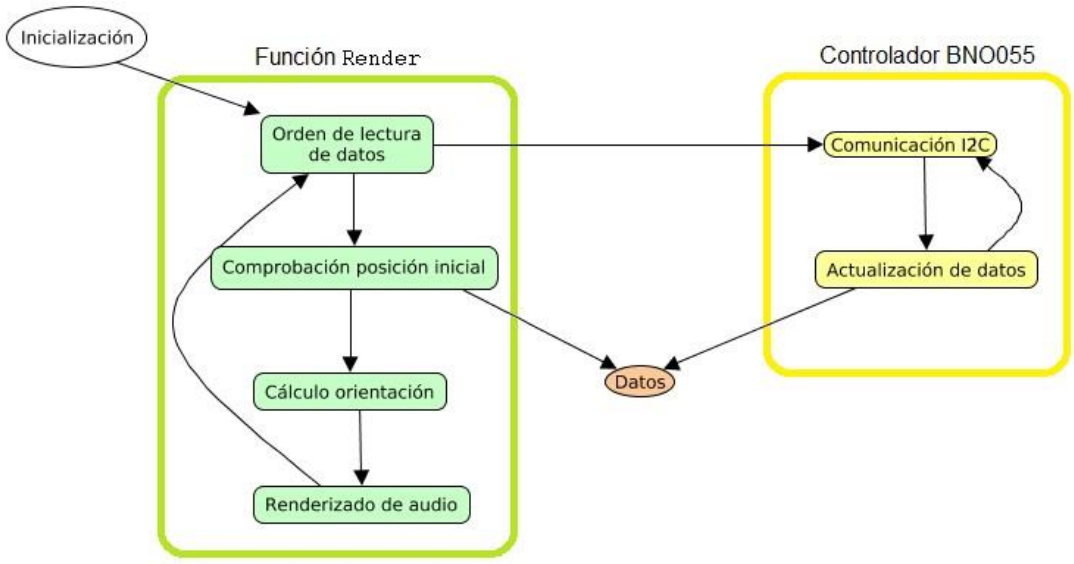


**Figura 9. Diagrama de flujo.**

Nada más encenderse, el código tardará unos segundos en cargar el sistema de Bela y la 3D Tune-In Toolkit. De forma nativa se han incluido una serie de pistas de audio en formato *wav* (por ejemplo, el sonido de una hoguera, un discurso o un arroyo) y el HRTF facilitado por los tutores. Además, el sistema tomará como punto de referencia la posición inicial del sensor.

Dentro del código, en cada iteración de la función `render` se va a hacer una llamada a la hebra encargada de gestionar el BNO055. Una vez activada, la hebra se comunicará por I2C con el sensor y actualizará los datos almacenados en la variable global que contiene la información de los cuaterniones.

A su vez, estos datos serán utilizados en la función `render` para calcular cuál debería ser la posición del audio según el punto inicial. Dicha información se transmite, junto con el HRTF, a la 3D Tune-In Toolkit, la cual devolverá la pista de audio modificada según sea conveniente.

Por último, el audio se emitirá a través de la salida de audio analógica, a la cual estarán conectados los auriculares de diadema.

En este punto se han cumplido los requisitos 2 y 2.3 (de nuevo, consúltese el TFG asociado [1] para más información sobre este tema).

## 3.5. Hardware de la aplicación

Tras haber estudiado la naturaleza de la placa de desarrollo que se va a emplear en nuestra aplicación, elegido el sensor que la complementará y diseñado el software que los comunicará y gestionará el renderizado del audio, podemos continuar con el siguiente punto: diseñar el hardware que acompañará a todo el conjunto. A partir de este momento se ha trabajado de forma exclusiva con la Bella en su versión mini ya que, debido a su tamaño, es más fácil de manipular.

Uno de los requisitos principales que definen nuestro sistema reside en la necesidad de controlar de alguna manera ciertas variables del software. En concreto, se requiere el control de la amplitud de la pista de audio (o volumen), la pista de audio en sí y la configuración del punto de referencia. A su vez, será necesario también monitorizar el estado del sistema. Para ello se ha decidido incluir los elementos electrónicos recogidos en la lista que se muestra a continuación:

- Control de amplitud: potenciómetro de 10 kΩ. Respecto al software, dicho potenciómetro controlará la amplitud multiplicando la señal de salida por un factor entre 0 y 1. Esto es así porque Bela representa los datos obtenidos a través de sus puertos de entrada analógicos con valores comprendidos entre el 0 y el 1.

- Control de pista: potenciómetro de 10 kΩ. El software que lo controla se ha diseñado de manera que se generen una cantidad de rangos equidistantes entre 0 y 1, según el número de pistas incluidas. De esta manera, las pistas tendrán asignados diferentes rangos, y el valor que detecte el puerto analógico de entrada será el que estipule qué pista debe reproducirse. Se permiten un máximo de siete pistas (dato obtenido durante la práctica) en formato *wav*. Cada vez que se seleccione una pista ésta comenzará desde el principio.
- Control del punto de referencia: para dicho control se ha utilizado un botón de pulsador el cual, al ser pulsado, sustituirá el cuaternión almacenado en la variable que contiene el punto de referencia por el valor que detecte el sensor en el momento en que se produzca la pulsación. Esta característica permite que, si al estar mirando en una dirección se pulsa el botón (por ejemplo, cuando se mira hacia la izquierda), esta orientación será tomada como nuevo *front* o punto de referencia frente al oyente.
- Indicación del estado del sistema: la inclusión de un diodo LED se ha pensado como ayuda para diagnosticar problemas o fallos del sistema, pero se espera una integración más profunda en implementaciones futuras. En la aplicación actual cuando el dispositivo se inicia, el diodo LED parpadea muy rápidamente. En situaciones en las que el sistema falla y no responde ni reproduce el audio, el diodo LED se encuentra completamente encendido. En cambio, si no hay fallos, el diodo parpadea o se encuentra apagado.

Para unir todos los componentes es necesario el diseño de un circuito que los ordene y conecte. Este circuito debe ser soldado sobre una placa perforada que incluya conexiones macho para el GPIO de la Bela: todo el conjunto permitirá una conexión física estable de los diferentes puertos, lo que evitará posibles errores a la hora de la comunicación sensor – Bela, así como movimientos indeseados.

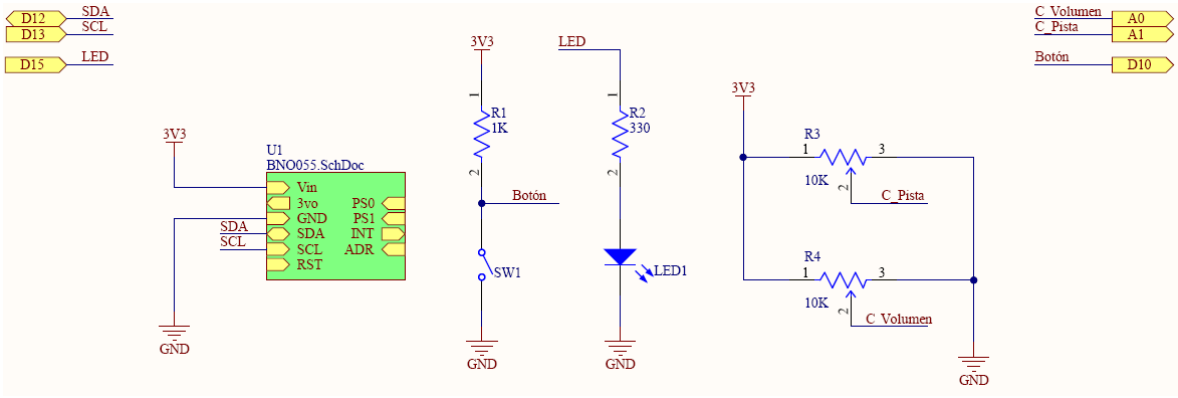


**Figura 10. Esquema eléctrico de la placa accesoria. Software utilizado: Altium Designer.**

El esquema del circuito eléctrico puede observarse en la Figura 10[6]: se emplean dos puertos de entrada analógicos, el A0 para el potenciómetro que controlará el volumen y el A1 para el que controlará la pista. Además, los puertos digitales E/S D10 y D15 conectarán al botón y al diodo LED respectivamente, el D13 se corresponde con la pista SCL y el D12 con la pista SDA, ambos necesarios para la comunicación a través del protocolo I2C. Por último, se utiliza una pista general de alimentación de 3.3 voltios, y una señal de tierra. Se ha colocado una resistencia de 1 kΩ para el botón y otra de 330 Ω para proteger al diodo LED.

Se ha optado por una placa perforada estañada por ambas caras con dimensiones de 6x4 centímetros a la cual se le ha eliminado un trozo de 18x8 milímetros que descansaba sobre las conexiones de salida de audio de Bela, utilizando tijeras de electricista y un cúter.

En consonancia con el poco espacio disponible, la colocación de los diferentes componentes y sus conexiones ha requerido de un ejercicio de ordenación y estrategia más intenso. Debido a la naturaleza y dimensiones del circuito, no se han encontrado grandes problemas de compatibilidad electromagnética.

[6] También disponible en el Apéndice B: Circuitos electrónicos.

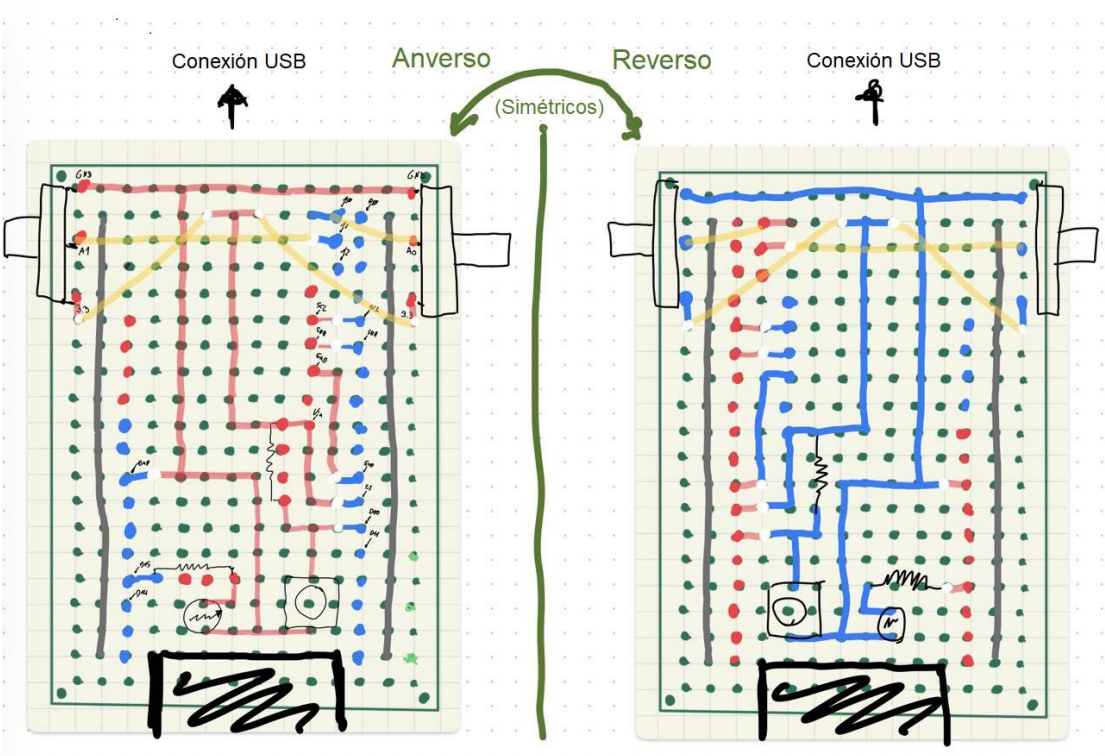


**Figura 11. Boceto de colocación de las pistas.**

La Figura 11 muestra el boceto de planificación para la soldadura de los componentes en la placa taladrada:

- Las líneas de color azul representan las pistas que se deben soldar en la cara en cuestión.
- Las líneas de color rojo huecos que hay que evitar (pues por detrás están conectadas).
- Las líneas grises representan huecos que se prefieren evitar, para poder usar en caso de necesidad.
- Las líneas amarillas representan cables enfundados (utilizados para garantizar el aislamiento eléctrico en pistas que se cruzan).
- Los puntos blancos de algunos huecos representan los puentes entre ambas caras.

En la parte más baja se ha tachado el trozo de placa que no será utilizada. El potenciómetro izquierdo corresponde al cambio de pista y el derecho al volumen (en el anverso).

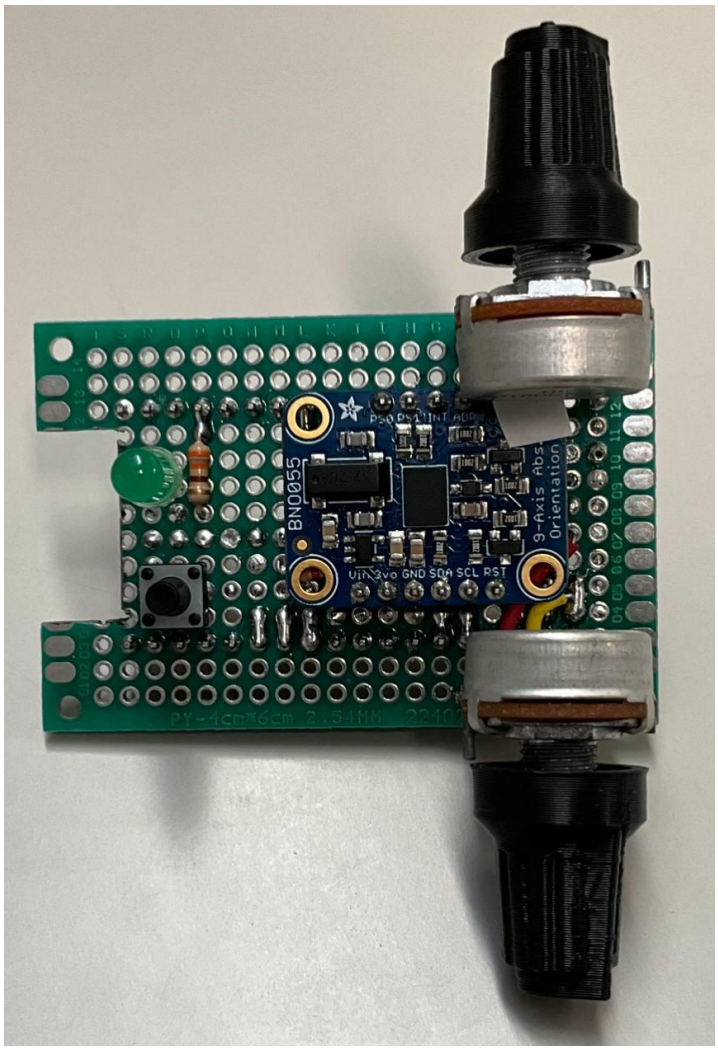


**Figura 12. Detalle del anverso del resultado final.**

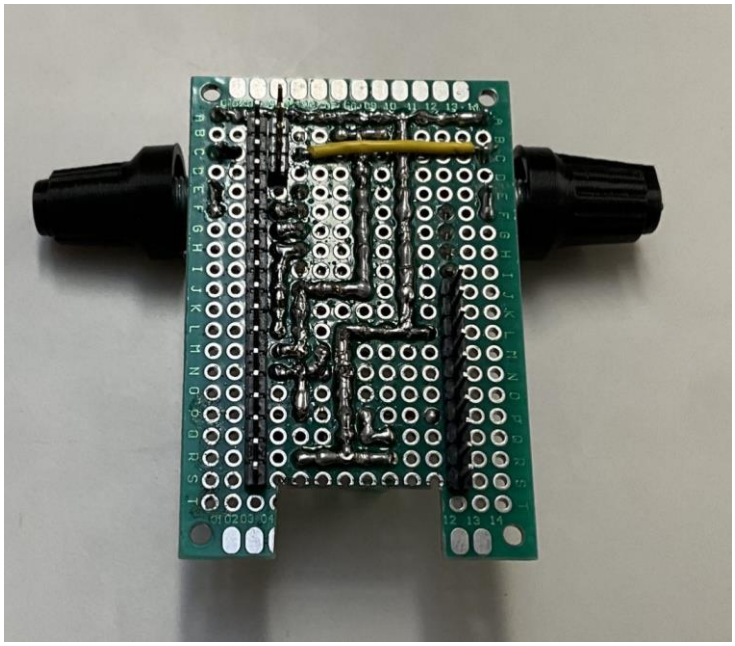

**Figura 13. Detalle del reverso del resultado final.**

El resultado obtenido tras soldar los diferentes componentes se puede observar en la Figura 12 y la Figura 13. Se ha utilizado estaño libre de plomo para todas las soldaduras. El diodo LED ha sido soldado con una holgura de 18.85 mm de altura para facilitar la visualización y acceso al mismo. El botón de pulsador tiene un accionador elevado (concretamente, de 5.5 mm) a propósito: esto permite un mejor agarre a la hora de colocar un extensor de botón (comentado en el capítulo 3.6. Problema mecánico). Sin éste, la pulsación del botón sería innecesariamente complicada.

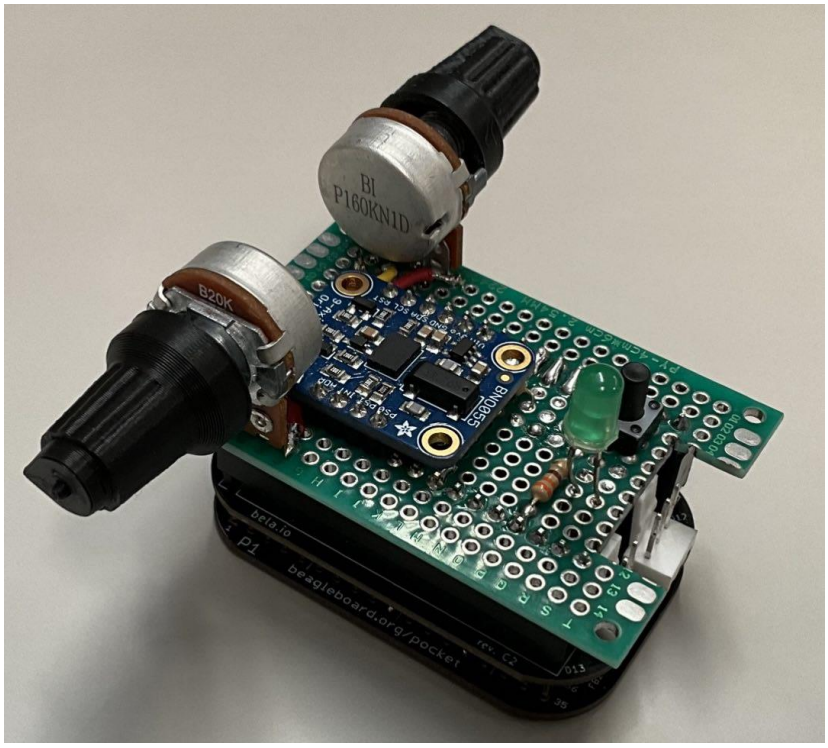


**Figura 14. Resultado al apilar la placa accesoria sobre Bela mini.**

En esta última imagen (Figura 14) se puede contemplar la placa de extensión apilada sobre Bela. Dicho sistema se corresponde a la versión número dos del diseño: la primera versión utilizaba una placa más pequeña al no contemplarse el uso de los potenciómetros o del diodo LED. También incluía dos líneas de puertos hembra de conexión para poder quitar y poner el sensor a voluntad (la versión actual no dispone de espacio físico para dicho propósito).

El diseño y montaje del circuito cumple con los requisitos 3, 3.2, 4 y 6.

## 3.6. Problema mecánico

Llegados a este punto poseemos un sistema funcional capaz de detectar orientación y renderizar audio en consecuencia, pero resta un último paso para completar la funcionalidad del dispositivo. Es necesario diseñar una caja que asegure la protección de éste y un ofrezca un soporte adecuado: el dispositivo debe encajar dentro de la carcasa sin holguras, evitándose así movimientos no intencionados que añadirían ruido al resultado final.

El objetivo final es el de crear una carcasa a medida que permita albergar el dispositivo, que tenga las aberturas necesarias para realizar las diferentes conexiones, aberturas que permitan la ventilación, una tapa para acceder al interior y un sistema para poder engancharla a la diadema del auricular sobre el que se quiera colocar. Dicho diseño se ha realizado utilizando la herramienta gratuita TinkerCAD, a la cual se puede acceder a través de internet.

Una vez obtenidas las diferentes medidas del dispositivo (usando regla y escalímetro) de la forma más precisa posible, se ha creado un modelo 3D del conjunto Bela, placa y sensor, junto con los diferentes componentes, la cual se puede observar en la Figura 15 (la figura de la izquierda se corresponde con la primera versión, mientras que la de la derecha se corresponde con la segunda).

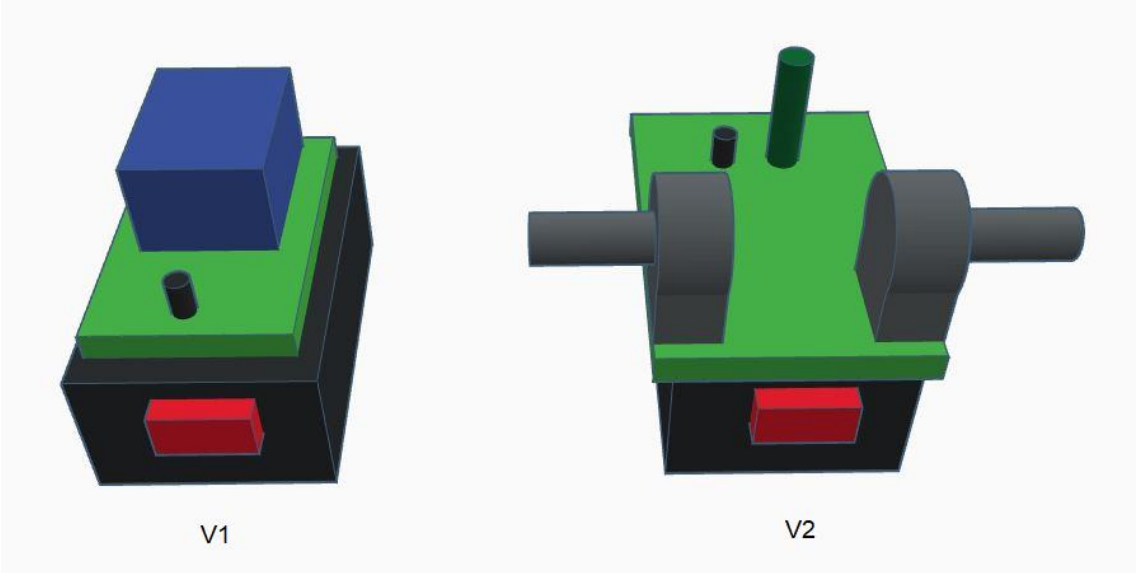


**Figura 15. Modelo de referencia 3D del sistema. La caja roja representa la ubicación del puerto USB, la caja negra representa la Bela, la verde la placa accesoria y la azul el BNO055. El cilindro negro representa el botón y el verde el diodo LED. Las dos estructuras grises de la imagen de la derecha representan los potenciómetros.**

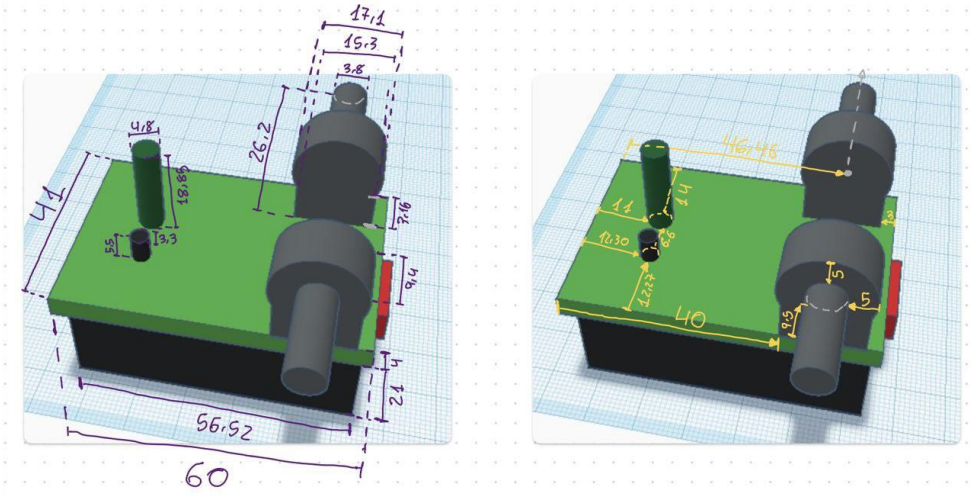


**Figura 16. Detalle de las dimensiones y posiciones en milímetros de los diferentes componentes.**

Tras la toma de medidas inicial y la creación del modelo de referencia, se continuó con el diseño de la carcasa. Dicho modelo se ha dividido en dos partes: el cuerpo de la carcasa y la tapa. En todo momento se ha añadido un margen de 0.2 milímetros extra con el objetivo de prevenir el error añadido por la boquilla de extrusión a la hora de imprimir la pieza.

Como se puede deducir de la Figura 17 y la Figura 18, la carcasa ha tenido unas dimensiones totales de 44.75 milímetros de ancho, 53 milímetros de alto y 64.75 milímetros de profundidad. La abertura inferior permite pasar una brida de cuatro milímetros de ancho para sujetar el dispositivo a la diadema del auricular. Se ha añadido un bisel interior de dos milímetros de alto para favorecer la unión de la caja y la tapa.

Debido a que el conjunto Bela más PocketBeagle tiene unas dimensiones menores que la placa taladrada, es necesario crear un interior más estrecho para favorecer un mejor agarre.

Se ha añadido un sistema de anclaje para poder fijar la tapa en su posición modificando un diseño de S. Bembouz en el que se usaba para cerrar la carcasa de otra aplicación. El diseño original se puede encontrar en [18] bajo licencia Creative Commons: CC BY.

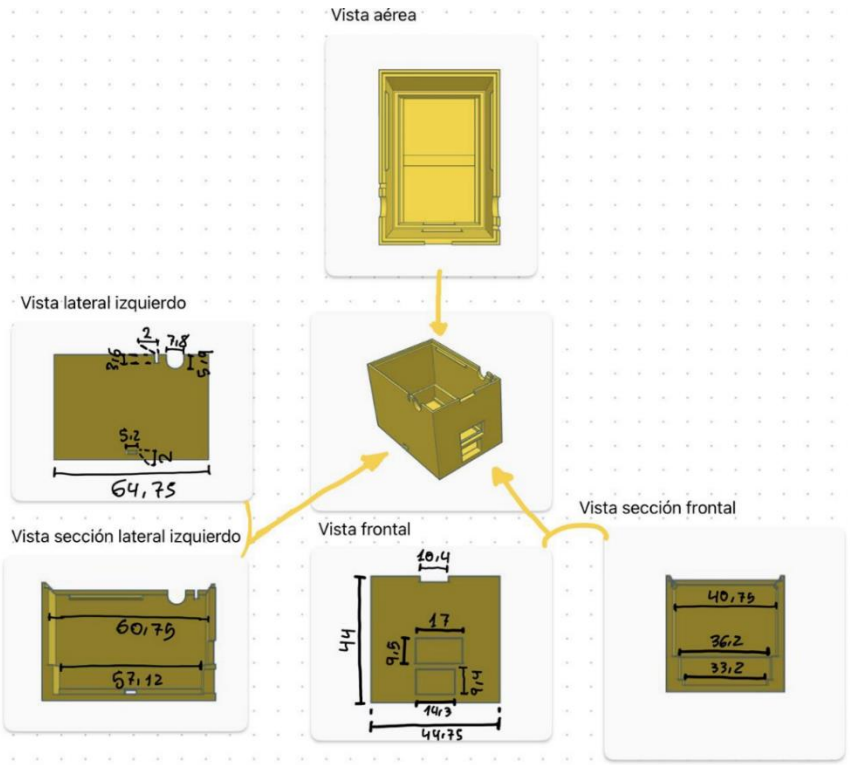


**Figura 17. Dimensiones en milímetros del cuerpo de la carcasa.**

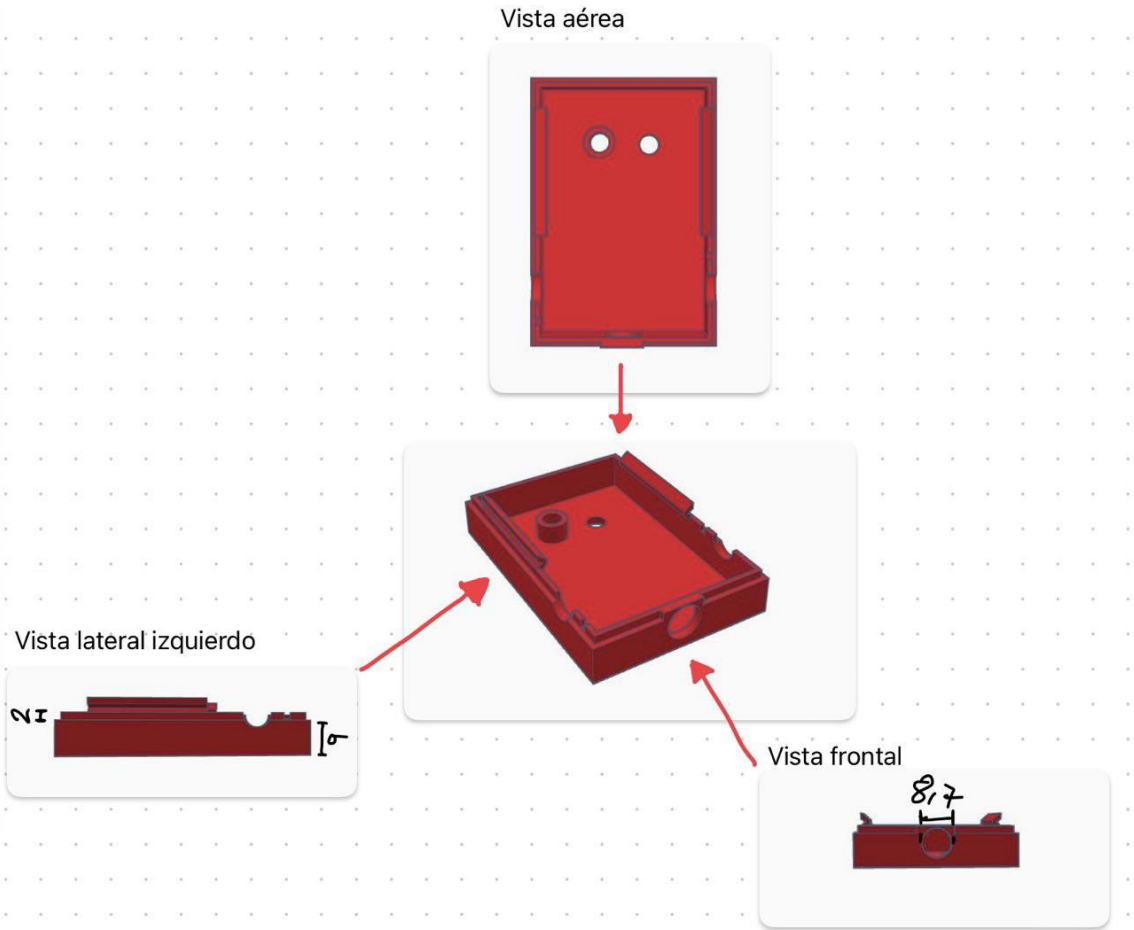


**Figura 18. Dimensiones en milímetros de la tapa de la carcasa.**

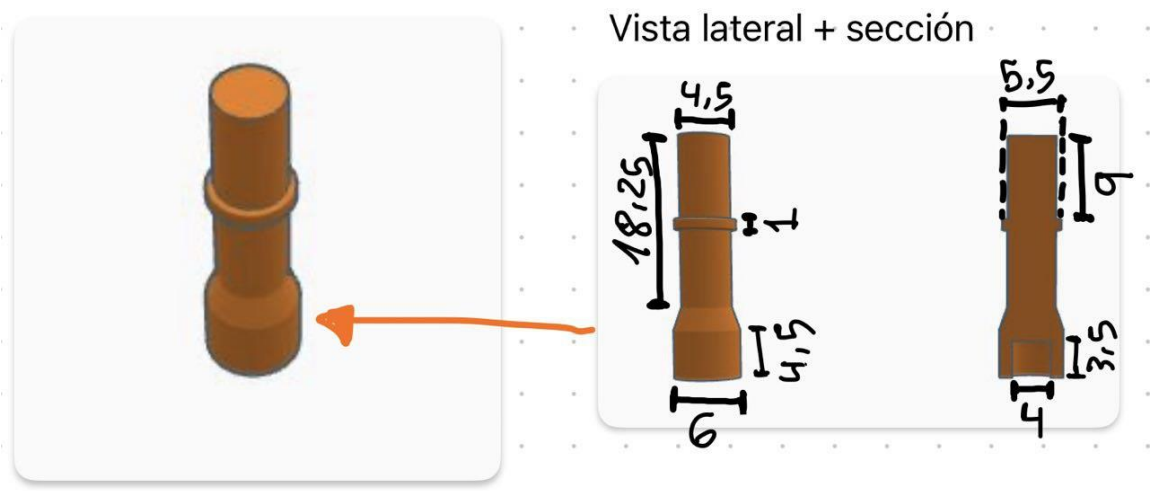


**Figura 19. Dimensiones en milímetros del extensor del botón (coloquialmente denominado 'linterna').**

La Figura 19 detalla el extensor de botón diseñado para facilitar su pulsación. El anillo que se añade a 12.75 milímetros de la base evita que al ser utilizado se salga de su posición.

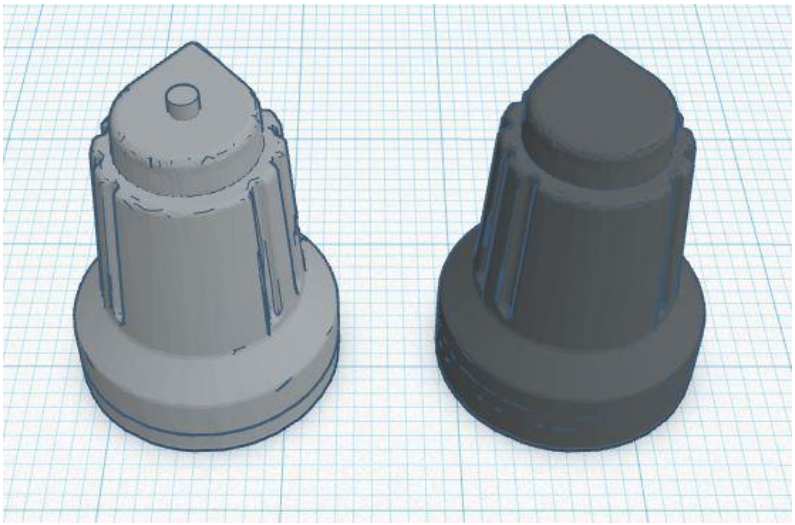

**Figura 20. Detalle de los embellecedores.**

Para el diseño del embellecedor se ha modificado el modelo facilitado por L. Fotr. El diseño original se puede encontrar en [19] bajo licencia Creative Commons: CC BY. Se ha colocado un punto palpable en el embellecedor que controla la pista, diferenciándose así del embellecedor que controla el volumen.

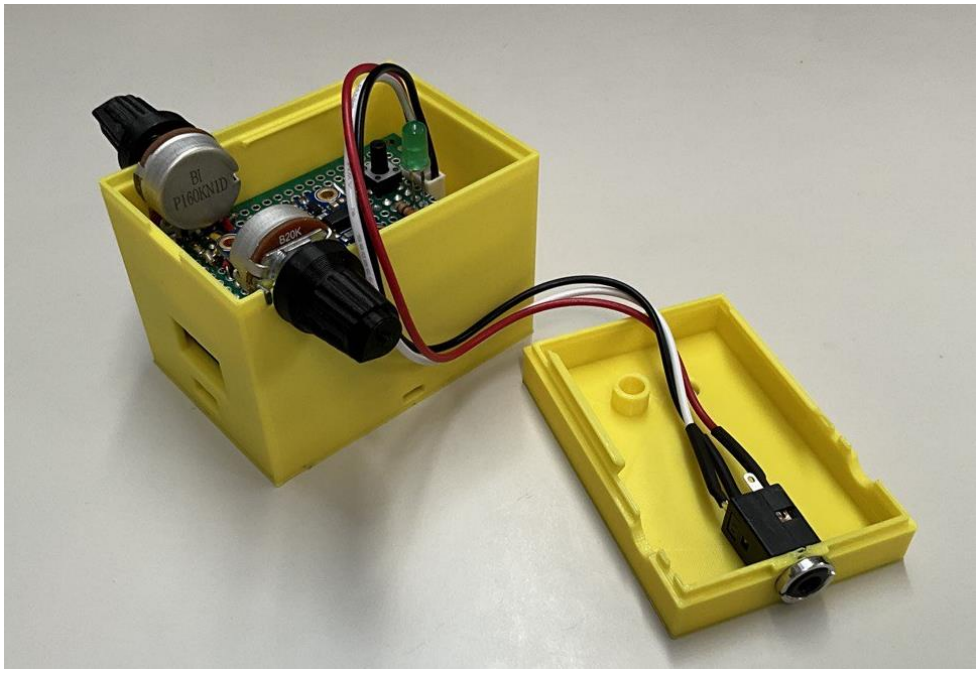


**Figura 21. Resultado final de la carcasa (abierta).**

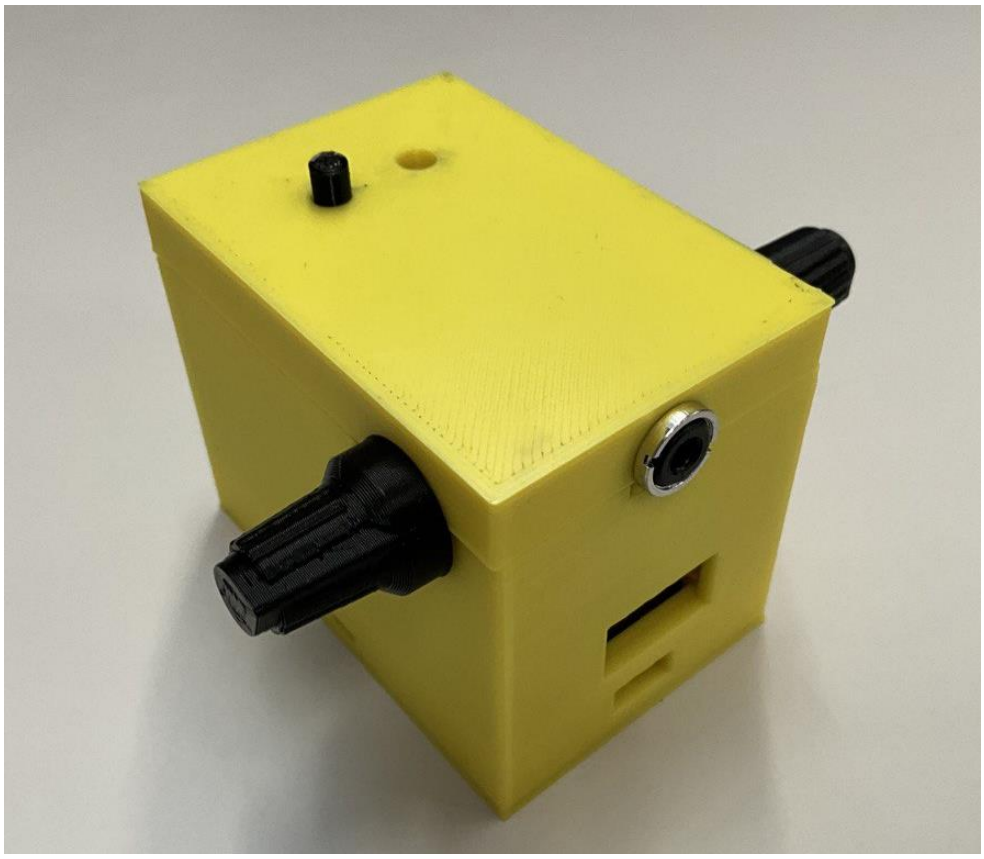

**Figura 22. Resultado final de la carcasa (cerrada).**

La Figura 21 y la Figura 22 muestran el resultado final tras la impresión. Para dicho cometido se ha utilizado la impresora Prusa i3 MK3 ubicada en el laboratorio del grupo DIANA. Se ha utilizado el programa CURA para generar el fichero *gcode* del diseño, configurando la velocidad de impresión a 50 mm/s, la temperatura de la cama de impresión a 60 ºC y la del extrusor a 195 ºC.

El plástico utilizado ha sido PLA reciclado producido por la empresa jienense Smart Materials 3D, en colores amarillo y negro. Los tiempos necesarios y los gramos de plástico utilizados se muestran en la siguiente tabla:

| Pieza | Tiempo requerido | Cantidad de material |
|---|---|---|
| Caja | 2 h 52 ‘ | 27 g (9.02 m) |
| Tapa | 1 h 11 ‘ | 11 g (3.6 m) |
| Extensor de botón | 5 ’ | 0.32 g (0.13 m) |
| Embellecedor (x2) | 12 ’ | 1 g (0.41 m) |
| **TOTAL** | 4 h 32 ‘ | 40.32 g (13.57 m) |

**Tabla 2. Cálculo de tiempos y materiales.**

La fijación a la diadema de los auriculares se ha realizado, como se ha comentado, utilizando bridas de 4 milímetros de ancho. Debido a la propia naturaleza de los auriculares utilizados en el laboratorio (concretamente, unos ATH-R70x de audio-technica) solo ha sido necesario colocar una brida en posición paralela a la diadema. En caso de utilizar unos auriculares diferentes, se debería utilizar tres bridas en total: una para la carcasa y dos para unirlas a diadema, colocadas en posición perpendicular.

La abertura para la conexión de un cable USB (o micro USB) permite la conexión con otros dispositivos. En concreto, dicha abertura permite la utilización de una batería externa para alimentar al sistema. Esto permite su uso sin necesidad de conectarse a un host, pero para ello es necesario iniciar el software de la aplicación como servicio [1].

Llegados a este punto hemos cumplido con los requisitos 5 y derivados (R5.1 a R5.4) y 7 y derivados (R7.1 y R7.2). No se ha cumplido en su totalidad el requisito R5.2, pues se ha comprobado a posteriori que las aberturas no son lo suficientemente amplias como para ventilar óptimamente el dispositivo.

**Figura 23. Sistema completo.**

## 3.7. Protocolo OSC

Una vez finalizado el diseño del dispositivo y tras haberlo probado con diferentes usuarios debemos proceder con el siguiente punto: las pruebas. Éstas serán explicadas en mayor detalle en el Capítulo 4. Verificación y Pruebas, pero antes comentaremos el uso de comunicación mediante mensajes OSC, protagonista en la realización de las mismas.

El protocolo OSC (*Open Sound Control*) fue creado para crear una red de dispositivos relacionados con audio (tales como sintetizadores u ordenadores), con la idea de crear una comunicación flexible y precisa. Se utiliza como alternativa al protocolo MIDI (*Musical Instrument Digital Interface*) y tiene presencia en una gran cantidad de proyectos de código abierto de audio.

Este protocolo funciona utilizando UDP/IP cuando se requiere transmitir mensajes a través de Internet (requieren puertos específicos para escribir/escuchar) o mediante Ethernet cuando se va a utilizar en redes locales (dentro de un mismo host utilizará la IP local 172.0.0.1). Los mensajes suelen contener un bloque de dirección acompañado por una serie de argumentos con información.

En nuestro caso, utilizaremos este tipo de mensajes para comunicarnos con el host, con la idea de transmitir mensajes que contengan la información de orientación. En el ordenador estará escuchando el programa creado por el grupo DIANA para probar la 3D Tune-In Toolkit: BiTa (*Binaural Test aplicaction*). BiTa contiene dicha librería y permite renderizar audio con más calidad, aprovechando la potencia de cálculo del host. Para realizar dicho renderizado necesita conocer la orientación del oyente, que puede ser enviada mediante mensajes OSC.

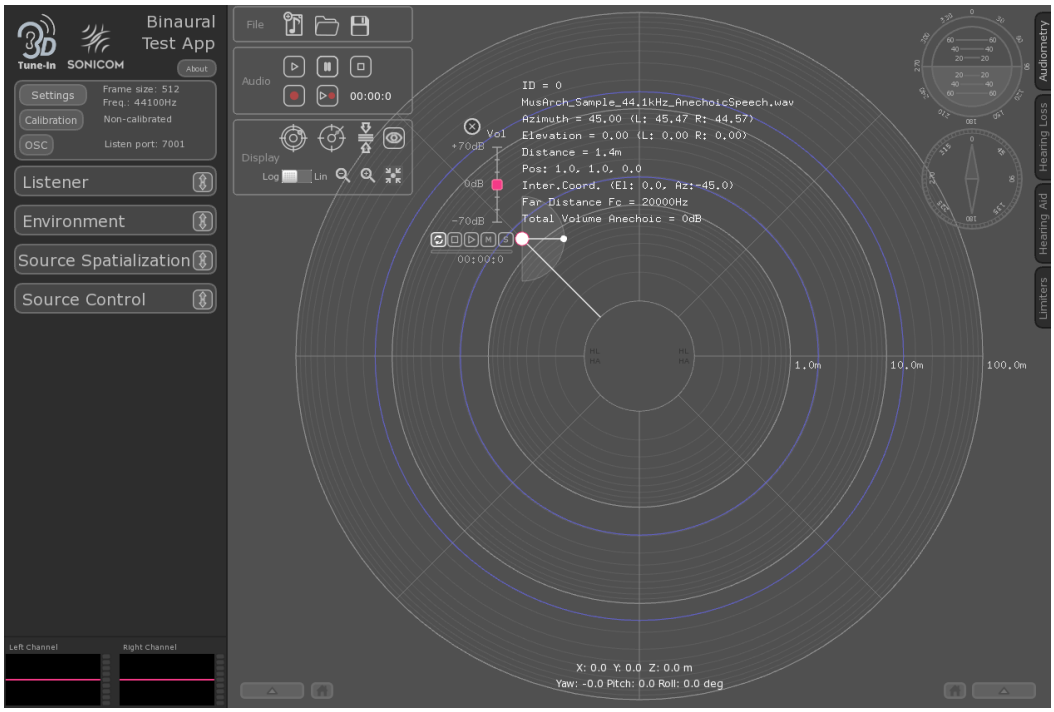


**Figura 24. Detalle de BiTa.**

Para la inclusión de esta característica se ha modificado el software utilizado en el subapartado 3.3.2. Descripción del código de Bela, al cual se le ha añadido los siguientes datos:

- Cabecera: `#include <libraries/OscSender/OscSender.h>`
- Objeto de comunicación: `OscSender osc_Tx_BiTa;` el cual será inicializado en la función `setup`.
- Puerto de comunicación: `int BiTa_port = 7001;`
- IP remota (Windows): `const char* remoteIP = "192.168.6.1";`

Además, se ha deshabilitado el volcado de datos que se realizaba por consola (en la función `render`, utilizando `rt_printf`) y se ha modificado la hebra `loop` para que utilice las funciones propias del protocolo OSC (Figura 25).

```
60 void loop(void*) {
61     while(!Bela_stopRequested()) {
62         imu::Vector<3> euler = bno.getVector(I2C_BNO055::VECTOR_EULER);
63         H_R_P[0] = euler.x() * M_PI / 180;       // Head
64         H_R_P[1] = euler.y() * M_PI / 180;       // Roll
65         H_R_P[2] = euler.z() * M_PI / 180;       // Pitch
66 
67 
68         // BITA reads P-H-R in radians
69         osc_Tx_BiTA.newMessage("/3DTI-OSC/v2/listener/orientation").add(H_R_P[2]).add(H_R_P[1]).add(H_R_P[0]).send();
70 
71 
72         // BeRTA reads listener-H-P-R in radians
73         osc_Tx_BeRTA.newMessage("/listener/-
   orientation").add("DefaultListener").add(H_R_P[0]).add(H_R_P[2]).add(H_R_P[1]).send();
74 
75         usleep(gTaskSleepTime);
76     }
77 }
```

**Figura 25. Función `loop` tras añadir el uso de mensajes OSC.**

BiTa lee los datos de orientación en vectores de Euler, en formato *pitch* (cabeceo o elevación respecto al eje x), *roll* (ángulo de alabeo respecto del eje z), *yaw* o *head* (guiñada o dirección dada respecto al eje y), en radianes.

El BNO055 ofrece los datos en orden *Head*, *Roll*, *Pitch*, en grados. Aunque el sensor ofrece la posibilidad de cambiar de unidad, no se ha conseguido que funcione dicha característica, por lo que se decidió implementar un cambio manual de grados a radianes apoyándonos en la librería `c_math`.

Por último, realiza el envío del mensaje estipulando la dirección (`"/3DTI-OSC/v2/listener/orientation"`) y añadiendo los argumentos en el orden correcto. Es importante destacar que la dirección a la que se envían los mensajes OSC es arbitraria y se acuerda entre transmisor y receptor. En este caso, BiTa espera recibir los mensajes de orientación en la dirección expuesta.

Queremos destacar que en la implementación final se realizó la comunicación simultánea con BiTa y con BeRTA (versión más moderna de la aplicación de testeo), enviando mensajes a la vez por puertos diferentes. BeRTA fue configurado para escuchar en el puerto 10017 y utilizaba un objeto de comunicación específico: `OscSender osc_Tx_BeRTA;`. Aunque fue un éxito, todo el código extra se eliminó para realizar las pruebas de latencia.

Se utilizaron dos hosts diferentes para comprobar la comunicación: Windows y Mac. Con Mac no hubo problema, pero para posibilitar la comunicación con Windows se necesitó configurar el firewall con dos pasos:

1. Crear una regla de entrada UDP para el puerto elegido (7001 en BiTa).
2. Autorizar la aplicación para que pueda utilizar redes privadas y públicas.

Por último, es importante destacar que en todo momento se ha realizado esta conexión a través del puerto USB de Bela, usando el servidor virtual que se crea al conectarse (explicado en el apartado 3.1 Complemento ‘Bela’).

Con este apartado cumplimos con el objetivo número 8.

# Capítulo 4. Verificación y pruebas

Llegados a este punto del documento ya contamos con un sistema completamente funcional que realiza el objetivo planteado: tenemos un dispositivo que detecta su orientación y genera un audio tridimensional en consonancia. Además, se han definido una serie de controles y se ha imprimido una carcasa para el dispositivo.

Pero un aspecto prioritario es poder caracterizar nuestro sistema, concretamente debemos estudiar qué niveles de latencia maneja. A priori, tras realizar pruebas de uso subjetivas en varios usuarios, no se ha detectado ningún tipo de error ni discontinuidad a la hora de reproducir el audio. Aun así, es necesario realizar un análisis riguroso del sistema que nos otorgue unos valores objetivos.

## 4.1. Punto de partida

Para poder crear un sistema de medidas adecuado debemos ampliar nuestros conocimientos en dicho ámbito. Es fundamental estudiar la bibliografía relacionada con medidas de latencia, concretamente en audio 3D. Siguiendo esta línea hemos desgranado una serie de artículos en los cuales se describe distintos sistemas de pruebas, pero todos tenían un punto en común: para realizar la medida de la latencia se diseña un sistema que genera un evento real que provoca un evento sonoro, y luego se mide la diferencia de tiempo entre dicho evento y la aparición del sonido deseado.

Una latencia demasiado elevada va a traducirse en un rendimiento pobre en cuanto a renderizado de audio se refiere (incluido el efecto de mareos en el usuario).

En [20] se utilizan dos VAEs (*Virtual Acoustic Environments* o entornos acústico/virtuales): el primero utiliza un HRTF con interpolación y sonido anecoico de entrada. El segundo utiliza la librería *libaave*, la cual usa varias fuentes para general un sistema más inmersivo. En ambos se utiliza un sistema de audio 3D, el cual necesita datos de posición y orientación del usuario.

El sistema de medida de latencia se ha divido en dos partes:

1. La primera parte se utiliza para caracterizar las latencias internas del sistema (desde la generación del evento real hasta la captación del evento generado). Para ello se utilizó un LED NIR (*Near Infra-Red* o infrarrojo cercano), un botón disparador, un fotodiodo y un sistema de cámaras de captura *Vicon*. Tanto el fotodiodo como el LED se conectaron a un DAQ (*Data AQquisition* o sistema de adquisición de datos), conectando el LED, además, al botón disparador. El fotodiodo se encontraba apuntando a una pantalla que mostraba la salida de las cámaras. Cuando se pulsaba el botón se generaba un evento real que era captado por el DAQ y pasados unos instantes el fotodiodo generaba una señal que simbolizaba la detección del evento por parte de las cámaras. Se midió la diferencia de tiempo entre ambas señales.
2. El segundo método fue creado para calcular la latencia en todo el sistema. Para ello se utilizaba un torso de maniquí colocado sobre una base giratoria, el cual incluía un generador de evento real. Además, en la cabeza del maniquí se añadieron unos marcadores, los cuales activarían las cámaras. La prueba consistió en girar el maniquí varias veces para comparar la diferencia de tiempo de la señal de las cámaras y la del evento real.

El sistema descrito en este artículo nos ha servido para distinguir entre los diferentes conceptos clave que forman parte de la medida de la latencia, pero no es adecuado para nuestra aplicación: los componentes que utiliza son específicos y poco accesibles.

En [21] se esboza otro método para medir la latencia del audio en un entorno tridimensional basado en movimiento. Para ello se comparó la señal de salida de unas gafas de realidad virtual con una señal de referencia sintética creada a partir de un sensor que monitorizaba los movimientos de la cabeza. Dichas señales se analizaron mediante una serie de procesos matemáticos que permitían la correlación cruzada entre ambas.

Para realizar las medidas siguieron el siguiente procedimiento: generaron la señal de referencia basada en los movimientos de la cabeza de un usuario, buscando los cero grados de azimut. Luego pidieron al usuario que realizara movimientos estrictamente horizontales de forma continua entre el azimut 0 ° y 90 °.

El estudio justifica la necesidad de crear un sistema de medida de latencia en la dificultad de conocer con exactitud los parámetros que caracterizan los diferentes productos comerciales. Además, a la hora se estudiar los resultados identificaron diversas derivas y buscaron sus posibles orígenes: una de las justificaciones sobre la existencia de derivas fue debido a que, a la hora de encontrar el ángulo cero de azimut, lo hacían mediante órdenes verbales (de un espectador al usuario que vestía el dispositivo).

En [22] se vuelve a crear un sistema de medición de latencia extremo a extremo utilizando Windows 2000.

En dicho sistema se estudiaban eventos externos e internos: los externos eran medidos mediante osciloscopio y los internos mediante la marca de tiempo de las funciones utilizadas, externalizando los eventos generados mediante un puerto serie. Para la generación del evento externo se utilizó una plataforma giratoria que contaba con un sensor electromagnético *Polhemus Fastrak*. Para la síntesis del evento generado, se utilizó un programa propio de análisis latencia que generaba señales por software.

En dicho estudio se acepta como latencia máxima de un sistema de audio 70 milisegundos (para una velocidad de 180 °/s), pues es el punto en el que se supera el umbral de percepción.

Tras el análisis de los diferentes artículos se han extraído los siguientes conceptos:

- Señal de evento real: la cual se utiliza como punto de referencia.
- Señal de evento generado: simboliza el evento que queremos medir.
- Medida extremo a extremo: la que se realiza desde la generación de la señal de evento real hasta la de evento generado, siendo este último el que se produciría cuando se han utilizado todos los componentes del sistema.
- Latencia Total del Sistema (LTS): simboliza la latencia total generada por todos los componentes de nuestro sistema. Se obtendría tras estudiar estadísticamente un conjunto de medidas extremo a extremo en las cuales se ha utilizado el sistema al completo.

En nuestro caso, se va a diseñar un sistema de pruebas para medir la latencia total del sistema usando una medida extremo a extremo. Para ello utilizaremos una plataforma giratoria que favorezca el movimiento de una cabeza de maniquí, la cual vestirá los auriculares de diadema encima: la salida de este sistema será la utilizada como señal de evento generado.

Como disparador de la señal de evento real utilizaremos una solución hardware localizada en la base de la plataforma. Todos los datos serán recogidos en un osciloscopio, donde se analizará la diferencia de tiempo entre ambos pulsos.

## 4.2. Diseño del sistema de pruebas

Para exponer cómo se ha diseñado el sistema vamos a dividir este apartado en tres partes: descripción de la plataforma giratoria, descripción del hardware y descripción del sistema de medida.

### 4.2.1. Descripción de la plataforma para pruebas

Podemos observar en la mayoría de los artículos que versan sobre este tema un denominador común: todos incluyen una parte móvil (generalmente, giratoria) que permite el movimiento del sistema.

**Figura 26. Plataforma de pruebas.**

En nuestro caso se ha creado una plataforma de madera (chapón marino de 15 milímetros) de 35 x 34 centímetros sobre la cual descansará una cabeza de maniquí.

Dicha plataforma incluye un eje central creado a partir de un manguito de latón de electricista, el cual cuenta con rosca y tuerca. Ambos objetos han sido modificados con un torno industrial para reducir el diámetro exterior (el resultado ha sido de 2.4 centímetros) y suavizar los bordes. Como se pude observar en la Figura 27, el eje sobresale 3.5 centímetros sobre la plataforma. Se han añadido cuatro tacos deslizantes con el objetivo de elevar la plataforma sobre la superficie donde se pose.

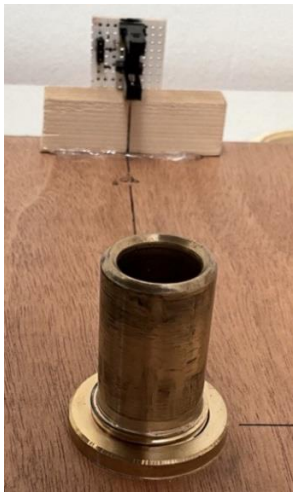

**Figura 27. Eje de latón.**

Para obtener una cabeza de maniquí de poliestireno expandido (coloquialmente denominado corcho blanco) se revisaron varias tiendas de arte y bazares, del entorno local. Llegó a nuestro conocimiento que dicho tipo de objeto es un artículo habitual en tiendas de disfraces, lo que nos permitió, en última instancia, adquirir uno de estos productos.

En concreto, se adquirió una cabeza de maniquí de poliestireno expandido que constaba de dos partes: trasera (que incluye la nuca y la parte posterior de la cabeza) y delantera (que incluye la cara y la parte anterior del cuello). Sus dimensiones aproximadas son: 20 centímetros de nariz a nuca, 18 centímetros de oreja a oreja y 33.5 centímetros de alto.

**Figura 28. Interior del maniquí.**

La parte trasera del maniquí se encuentra fija a una base de madera hecha a medida. Se ha utilizado un tubo de acero inoxidable de 2.5 centímetros de diámetro interior, fijándolo al maniquí con dos tornillos (uno en el occipucio del maniquí y otro en la parte posterior del cuello). El tubo metálico se fija a la plataforma de madera a través de un cilindro de aluminio (también torneado para este propósito) que cuenta con un tornillo para ajustar la presión (Figura 29).

**Figura 29. Detalle del cilindro de aluminio.**

La parte frontal del maniquí se ha unido a la trasera utilizando cinta aislante adhesiva de color blanco, facilitándose así su despiece en caso de ser necesario. Además, se ha añadido un palo de madera de 3 milímetros de diámetro a la base (coincidiendo con el punto más frontal del maniquí), utilizando silicona térmica. Dicho palo sobresale 4.5 centímetros de la base y forma parte del sistema disparador explicado en el siguiente aparado[7].

La base con la cabeza de maniquí se coloca sobre el eje que sobresale de la plataforma. Debido a que la velocidad angular del sistema no necesita ser elevada, se optó por utilizar el sistema explicado, el cual se basa en el rozamiento directo entre el eje de la base y el tubo de acero inoxidable de la cabeza. El eje entra dentro del tubo, lo que proporciona estabilidad. El cilindro de aluminio proporciona una elevación de 1.5 centímetros sobre la plataforma, lo que evita el rozamiento de la base al completo. Además, para favorecer el movimiento se ha aplicado grasa de litio entre ambos componentes metálicos.

En un acercamiento inicial se pensó en utilizar un sistema motorizado y un cojinete, pero debido a su complejidad se desechó la idea.

[7] La inclusión de este elemento genera una leve inclinación en la parte frontal del maniquí que no afecta a la hora de tomar medidas.

La altura total es de 37.1 centímetros. Todos los componentes de madera han sido tratados con imprimación selladora. La Figura 30 muestra el sistema de medida completo con el dispositivo montado encima.

**Figura 30. Sistema completo.**

### 4.2.2. Descripción del elemento medidor

Para medir las diferentes señales hemos optado por utilizar un osciloscopio, el Tektronix TBS 1052B-EDU presente en la mayoría de los laboratorios de nuestra escuela.

Las sondas utilizadas son las RIGOL RP2200 Passive Oscilloscope Probe, las cuales también son comúnmente utilizadas en nuestra escuela. Se utilizaron sin atenuación.

Ambos elementos aparecen en la Figura 31.

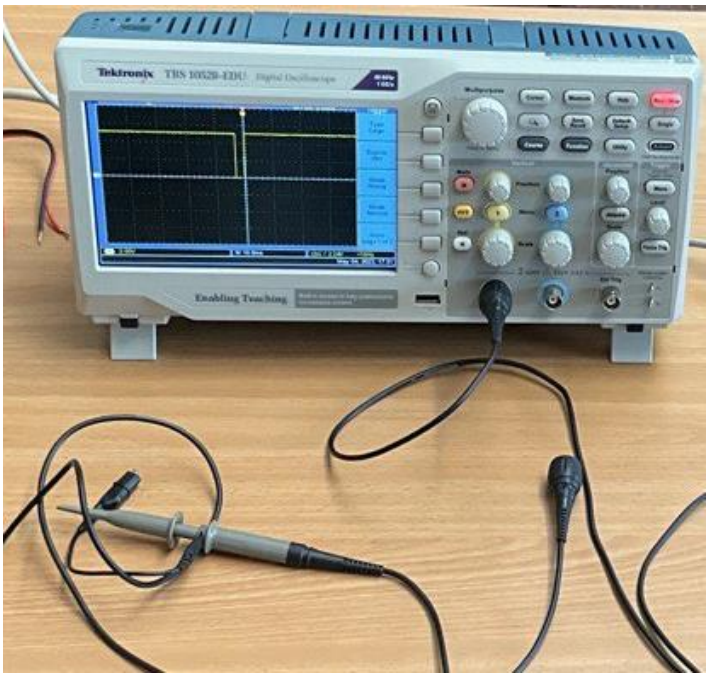


**Figura 31. Detalle del osciloscopio y las sondas.**

### 3.2.3 Descripción del disparador de evento real

La plataforma giratoria debe incluir en su estructura el disparador que generará el evento real, de manera que detecte cuándo pasa la parte delantera del maniquí por el punto en el que se encuentre situado.

Tras hacer una sesión de *brainstorming* se decidió elegir entre tres sistemas diferentes: una fotorresistencia, un sensor de efecto Hall y un fotointerruptor. Todos ellos fueron sometidos a pruebas para caracterizar su tiempo de respuesta ante el estímulo de entrada:

1. Uso de una fotorresistencia (LDR) modelo GM5539. El uso de una fotorresistencia fue la primera idea del proyecto, pero se dudaba sobre el tiempo de reacción de ésta. Para comprobar su velocidad se colocó un diodo láser en la plataforma giratoria y se fijó el LDR en uno de los laterales, conectando sus patillas a la alimentación y al sistema de medida previamente expuesto. En la Figura 32 se puede observar la curva resultante: utilizando una escala de tiempo de 10 microsegundos por división se obtiene un tiempo de 26 microsegundos.

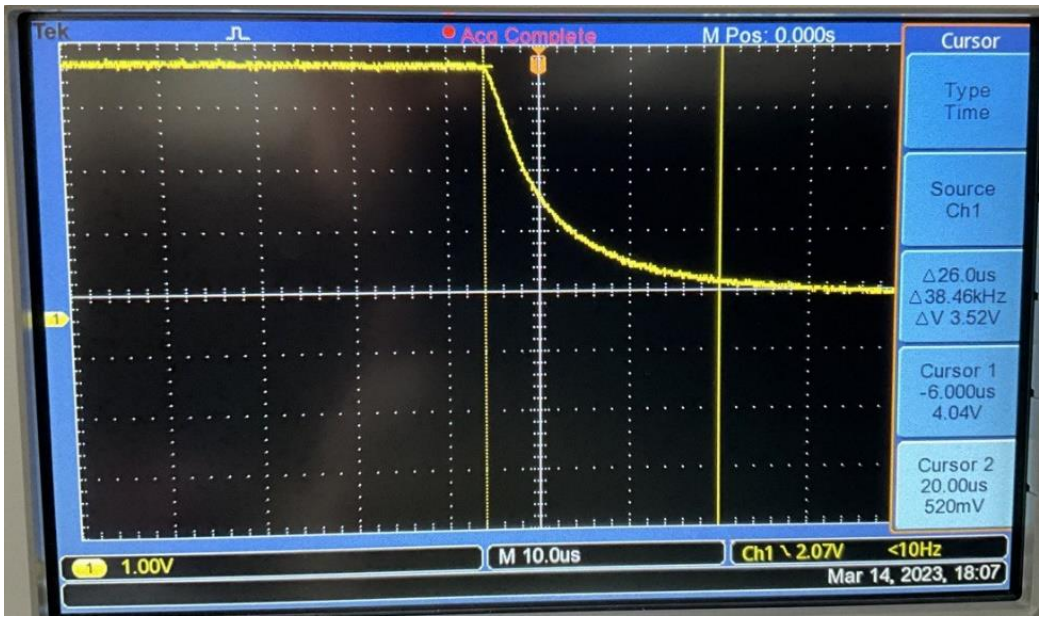


**Figura 32. Curva de salida del LDR.**

2. Uso de un sensor de efecto Hall. Ésta fue otra de las ideas a probar, la cual parecía la más rápida en primera instancia. Para medir la respuesta de este sensor se siguió el mismo procedimiento que con el LDR, pero sustituyendo el diodo láser por un imán de neodimio de cinco milímetros de diámetro. La Figura 33 muestra la curva de salida: utilizando una escala de tiempo de 50 nanosegundos por división obtuvimos un resultado de 90 nanosegundos.

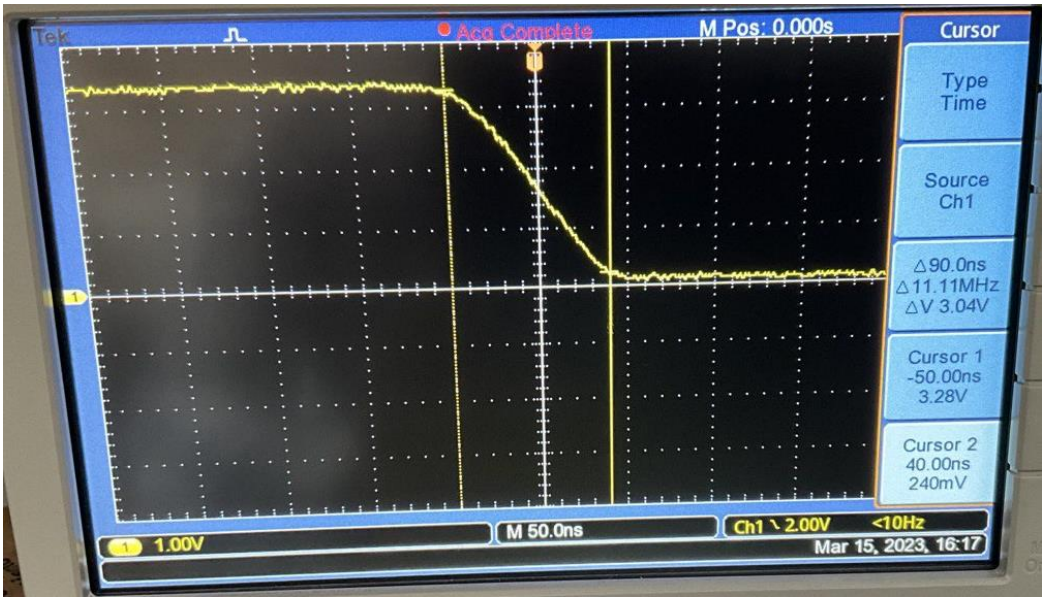


**Figura 33. Curva de salida del sensor de efecto Hall.**

3. Uso de un fotointerruptor modelo GP1A57HR (tiene forma de U cuadrada). Última de las ideas a probar. Para probar su funcionamiento se interrumpió la comunicación entre el emisor y el receptor utilizando un bolígrafo con el cuerpo negro, pasándolo entre ambos. La Figura 34 muestra su curva de salida: utilizando una escala de 10 nanosegundos por división obtuvimos un resultado de 40 nanosegundos.

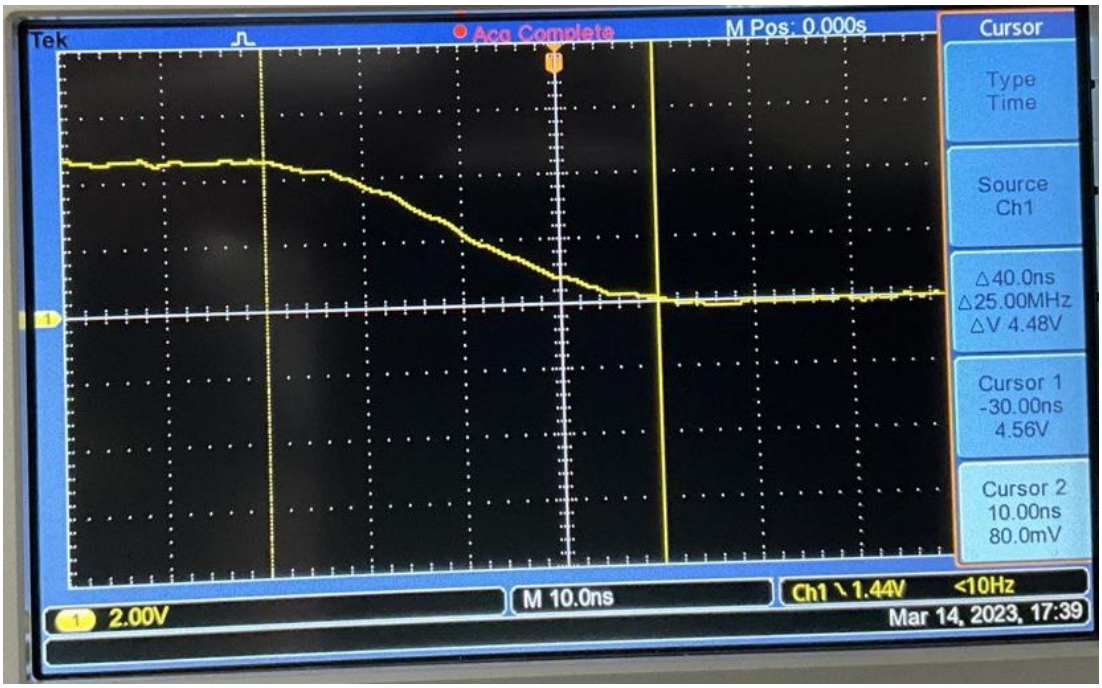


**Figura 34. Curva de salida del fotointerruptor.**

Hemos resumido los valores obtenidos dentro de la Tabla 3, mostrada a continuación:

| Sensor | Tiempo de reacción |
|---|---|
| LDR | 20 µs |
| Hall | 40 ns |
| Fotointerruptor | 10 ns |

**Tabla 3. Resumen de tiempos.**

Todos los métodos estudiados tienen un tiempo de respuesta mucho menor a los que se obtendrán a la hora de medir las latencias. Aun así, aunque todos los métodos son muy rápidos, al final se optó por utilizar el fotointerruptor como disparador de evento real.

Para ello se creó un circuito con una resistencia de 220 Ω de protección, y tres pines de conexión. Esto se puede observar en la Figura 35.

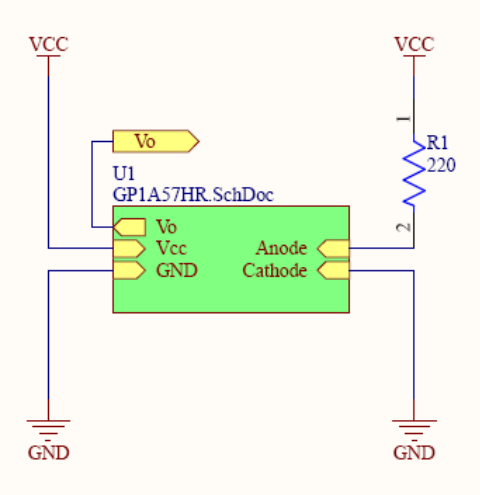


**Figura 35. Esquema del disparador.**

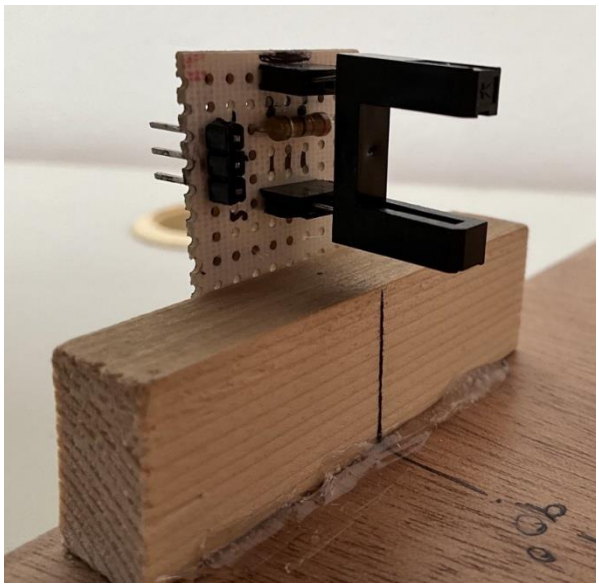

**Figura 36. Detalle del sensor colocado en la plataforma.**

El circuito se soldó sobre una placa perforada de 3x2.5 centímetros. Luego se colocó en la plataforma pegado con silicona térmica, apoyándose en un pequeño listón de madera. El circuito sobresale dos centímetros sobre dicho listón.

A la base giratoria del maniquí se le añadió el palo anteriormente comentado, y se pintó su extremo distal de color negro para mejorar la captación del evento. Dicho elemento puede atravesar el sensor sin ningún problema, tal y como se ve en la Figura 37.

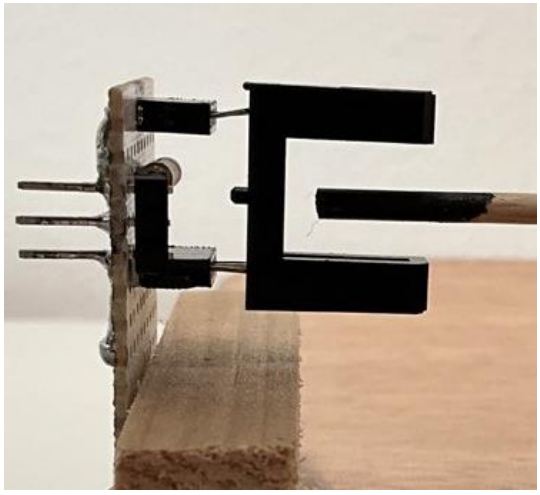

**Figura 37. Detalle del paso a través del sensor.**

Además, para mejorar la captación de la señal, se diseñó y agregó una cubierta impresa en PLA negro (misma configuración que en el apartado anterior). Dicha cubierta crea una sombra sobre el sensor para protegerlo en ambientes muy iluminados. La impresión de dicha pieza tarda 46 minutos en imprimir y consume siete gramos de plástico (2.45 metros). En la Figura 38 se muestran las medidas en milímetros de la pieza (cuenta con una pared de dos milímetros de grosor).

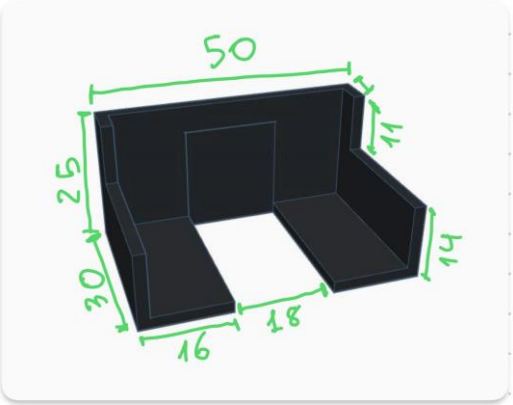


**Figura 38. Medidas del modelo 3D.**

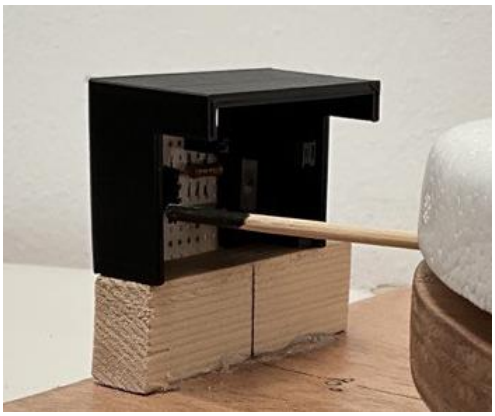

**Figura 39. Detalle del sensor con la cubierta.**

## 4.3. Diseño de las pruebas

A la hora de sintetizar cómo se iban a realizar las medidas divisamos la necesidad de comparar los resultados de nuestro sistema con otras configuraciones. En concreto se pensaron cuatro pruebas distintas: dos usando un mismo sistema de reproducción, pero con parámetros diferentes, y otras dos con sistemas dispares. Todas ellas se han creado con la misma premisa en cuanto a evento generado se refiere: configurar el dispositivo en cuestión para que reproduzca audio a partir de cierto ángulo.

Complementando las pruebas, los tutores crearon un HRTF específico de tipo heminegligente (esto es, que el audio se reproducirá desde los 0 a 180 grados de azimut, o lo que es lo mismo, cuando la fuente esté situada en el hemisferio izquierdo, pero no desde los 180 a 360 grados, esto es, cuando la fuente se encuentre en el hemisferio derecho) compuesto por 256 muestras. Además, se desactivó la interpolación del software de la aplicación (más información en [1]).

Como complemento del HRTF, se generó una pista de audio utilizando el programa de código abierto Audacity. Dicha pista contenía un tono cuadrado de 440 Hz, el cual oscilaba entre 1 y -1 voltios, empezando siempre en el valor positivo. Esta decisión fue tomada de forma arbitraria, con la intención de que la señal fuese más clara a la hora de estudiarla en el osciloscopio.

Por otro lado, el sensor ha sido alimentado y referenciado en todo momento con una placa accesoria: en nuestro caso, un Arduino, que servía única y exclusivamente para aprovechar sus conexiones de tierra y cinco voltios. Es posible utilizar alimentaciones de 3.3 voltios. Al ser alimentado, el sensor presenta una salida a nivel alto mientras no sea accionado. Cuando se active emitirá una señal en nivel bajo, que permanecerá hasta que se retire el objeto que esté interrumpiendo la comunicación.

Respecto al osciloscopio, en el canal 1 se colocó la sonda que lee la señal del disparador, y en el canal 2 la sonda del evento generado, ambos sin atenuación.

Es primordial el uso combinado del *trigger* y del modo *single* para la toma de medidas: esto permite congelar la imagen y colocar cómodamente los cursos en los flancos de las señales para medir la diferencia de tiempo producida desde que se generó el evento real hasta que el dispositivo comenzó a reproducir.

El *trigger* se colocó a media distancia entre la línea de cero voltios y el máximo del canal 1 (con una alimentación de cinco voltios, el *trigger* se colocaría a 2.5 voltios). El flanco que se precise detectar dependerá de la medida en cuestión.

En la mayoría de las aplicaciones se ha configurado la escala de amplitud del canal 1 en dos voltios y la del canal dos en 500 milivoltios. La escala de tiempo ha sido, en la mayoría de las pruebas, de 10 milisegundos.

El proceso de toma de muestras es prácticamente el mismo en todas las pruebas: se comienza con una fase inicial de colocación, luego se toman 10 medidas en sentido horario y 10 en sentido antihorario.

*Fase inicial*

Utilizada para colocar el sistema en la posición cero:

1. Colocación del disparador: en este primer paso se rota la cabeza en sentido horario hasta el instante en que la señal del canal 1 (disparador) cambia a nivel bajo.
2. Colocación de la cabeza: una vez hecho, se mueve toda la plataforma hasta conseguir un efecto similar en el canal 2. Si la señal del canal 2 es de cero voltios, tenemos que seguir moviendo todo el sistema de medida (la cabeza debe permanecer en posición fija) hasta el instante en el que comienza a reproducirse el audio.

   En caso de que estuviese reproduciendo audio al terminar el paso uno, movemos todo el conjunto en sentido antihorario hasta perder la señal, y procedemos como se ha explicado en el punto dos.

Tras cumplir con estos pasos, la pantalla del osciloscopio debería mostrar una señal de nivel bajo (0 voltios) en el canal 1 y una señal cuadrada en el canal 2.

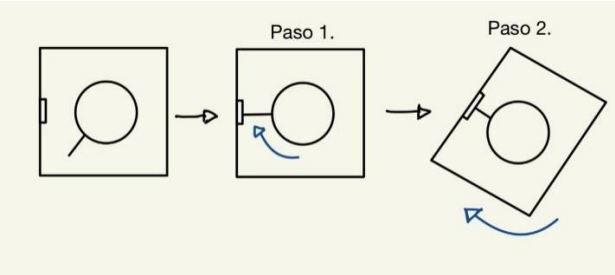


**Figura 40. Esquema explicativo de la fase inicial.**

*Fase de medición*

Una vez realizada la colocación inicial, se puede comenzar con la toma de las 40 medidas (20 en sentido horario, 20 en sentido antihorario). Todas las medidas se toman consecutivamente, encadenando los movimientos por pares, pasando por el sensor utilizando un ángulo de unos 90 grados de amplitud (realmente, sólo necesitamos pasar por el sensor, no hace falta moverse dentro de un ángulo mayor). Es muy importante mantener fija la plataforma en todo momento.

Antes de comenzar con la toma de medidas, es necesario girar la cabeza en sentido antihorario para sacarla de la posición inicial y colocarla dentro del semicírculo sin producción de audio.

1. Toma de muestra en sentido horario: primero configuramos el *trigger* para que detecte un flanco de bajada y activamos el modo *single*. Rotamos la cabeza en sentido horario rápidamente.

   La pantalla del osciloscopio se congelará, momento en el que utilizamos los cursores para calcular los tiempos: se coloca el cursor 1 sobre el *trigger* (en pantalla, el cursor 1 debería tener una medida de 0.000 segundos) y el cursor 2 sobre el primer flanco de la señal de audio. La diferencia de tiempo es el dato que buscamos.

2. Toma de muestra en sentido antihorario: se configura el *trigger* para que se detecte el flanco de subida y activamos el modo *single*. Procedemos igual que en el punto 1, colocando el cursor 2 en el último flanco visible de la señal de audio.

Es importante cerciorarse en ejercer la misma fuerza en ambos movimientos cada par de muestras, para asegurarse que la velocidad es la misma. Este proceso de toma de muestras se repite hasta completar la correspondiente tabla de mediciones.

El cambio de flanco entre medida horaria y antihoraria se introduce para eliminar las posibles derivas e imprecisiones al ajustar el cero. En este caso, tendremos un sesgo positivo en una dirección y un sesgo negativo en la otra. Suponiendo que ambos sesgos sean iguales, al realizar el promediado de las medidas éstos se sumarán y eliminarán.

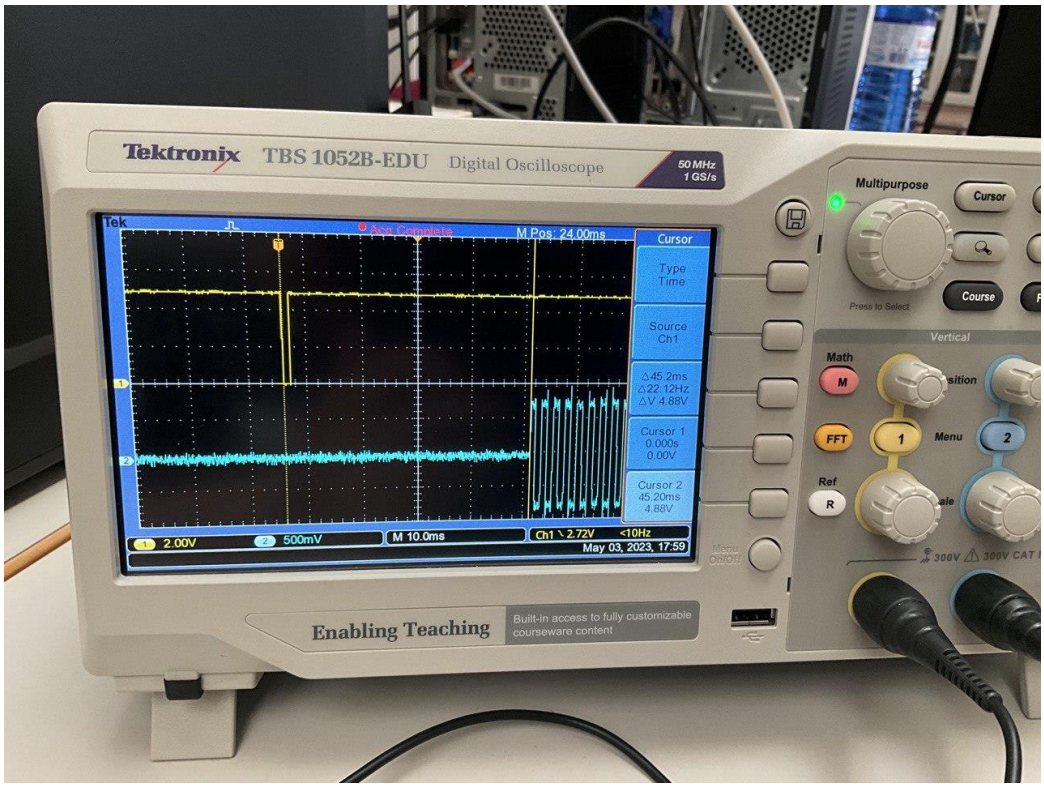


**Figura 41. Detalle de una de las medidas.**

Las pruebas elegidas son cuatro:

- Prueba 1.A – Bela detecta movimiento, Bela renderiza audio. Tamaño de búfer: 256.

  En esta primera prueba utilizamos el conjunto completo diseñado para nuestra aplicación. Estudiaremos la latencia extremo a extremo acorde al siguiente flujo: BNO055 capta movimiento → comunicación I2C → 3DTI Toolkit renderiza audio → salida audio.

- Prueba 1.B - Bela detecta movimiento, Bela renderiza audio. Tamaño de buffer: 512.

  Igual que caso anterior, pero utilizando un tamaño de buffer de 512 muestras.

- Prueba 2 – Bela detecta movimiento – host renderiza audio. Tamaño de búfer: 256.

  En este caso Bela será la encargada de utilizar el sensor BNO055 para capturar el movimiento.

Luego se comunicará mediante mensajes OSC con un host, el cual renderizará el audio utilizando BiTa. Tamaño de buffer de 256 muestras.

El flujo es: BNO055 capta movimiento → comunicación I2C → comunicación OSC → host renderiza audio → salida audio.

- Prueba 3 – Gafas de realidad virtual detectan movimiento – host renderiza audio. Tamaño de búfer: 256.

  La última prueba consiste en la utilización de un sistema comercial: unas Oculus Rift modelo HM-A. Precisan el uso de dos cámaras para detectar el movimiento.

  En este sistema, la comunicación se realizará dentro del mismo ordenador, de forma local, entre un programa de Unity y BiTa. El flujo sería: Oculus detectan movimiento → Programa en Unity calcula orientación → Comunicación Unity – BiTa dentro del mismo host → BiTa renderiza audio → salida de audio.

Como se ha mostrado, se realizarán dos pruebas con la misma configuración, pero diferentes parámetros: la prueba número uno. Es importante destacar que se han elegido tamaños de búfer de 256 y 512 muestras, pero no de 128. Esto es así porque en el TFG asociado [1], se demostró que dicho tamaño de búfer en Bela no es compatible con un HRTF de 256 muestras por cuestiones de rendimiento.

## 4.4. Resultados

A continuación, se adjuntan las tablas que recogen los datos recogidos en cada prueba. Cada tabla estará compuesta por cuatro columnas: *número* de muestra, *d*iferencia de tiempo en sentido *horario*, *d*iferencia de muestra en sentido *antihorario*, *media* del par de muestras.

A su vez, cada tabla irá acompañada con una gráfica de nube de puntos, en la cual se representarán las tres columnas a la vez.

Tanto el cálculo de las medias como la generación de las gráficas, así como los cálculos que se mostrarán a posteriori han sido realizados con Matlab (R2021b).

*Prueba 1.A* – Bela detecta movimiento, Bela renderiza audio. Tamaño de búfer: 256.

| Número | D. Horario (ms) | D. Antihorario (ms) | Media (ms) |
|---|---|---|---|
| 1 | 45,2 | 46,4 | 45.8 |
| 2 | 42 | 39,6 | 40.8 |
| 3 | 58,4 | 48 | 53.2 |
| 4 | 46,8 | 51,6 | 49.2 |
| 5 | 49,2 | 50 | 49.6 |
| 6 | 48 | 38,8 | 43.4 |
| 7 | 44,4 | 48 | 46.2 |
| 8 | 51,6 | 45,6 | 48.6 |
| 9 | 46,9 | 48 | 47.45 |
| 10 | 47,6 | 53,6 | 50.6 |
| 11 | 44,4 | 43,2 | 43.8 |
| 12 | 50,8 | 47,2 | 49 |
| 13 | 48 | 52,4 | 50.2 |
| 14 | 61,6 | 41,2 | 51.4 |
| 15 | 47,2 | 47,6 | 47.4 |
| 16 | 46,4 | 50,8 | 48.6 |
| 17 | 59,6 | 50,8 | 55.2 |
| 18 | 54,8 | 30,4 | 42.6 |
| 19 | 58,8 | 40,8 | 49.8 |
| 20 | 54 | 47,2 | 50.6 |

**Tabla 4. Resultados de la prueba 1.A.**

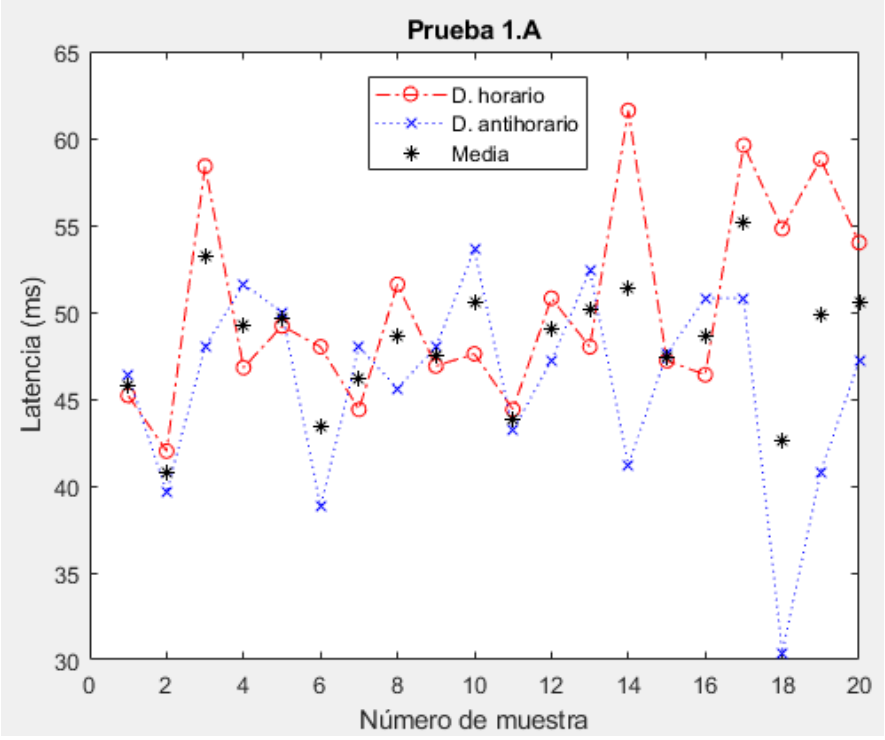


**Figura 42. Gráfica de la prueba 1.A.**

*Prueba 1.B* - Bela detecta movimiento, Bela renderiza audio. Tamaño de buffer: 512.

| Número | D. Horario (ms) | D. Antihorario (ms) | Media (ms) |
|---|---|---|---|
| 1 | 46 | 45,6 | 45.8 |
| 2 | 60,4 | 41,2 | 50.8 |
| 3 | 54 | 44,8 | 49.4 |
| 4 | 44 | 36,8 | 40.4 |
| 5 | 49,2 | 46,8 | 48.0 |
| 6 | 49,6 | 44,4 | 47.0 |
| 7 | 64,8 | 50,8 | 57.8 |
| 8 | 47,6 | 49,6 | 48.6 |
| 9 | 50 | 52 | 51.0 |
| 10 | 49,2 | 45,6 | 47.4 |
| 11 | 53,6 | 40,4 | 47.0 |
| 12 | 53,2 | 48,4 | 50.8 |
| 13 | 60,8 | 44,4 | 52.6 |
| 14 | 57,2 | 45,6 | 51.4 |
| 15 | 56 | 41,6 | 48.8 |
| 16 | 58 | 38,8 | 48.4 |
| 17 | 52,4 | 48 | 50.2 |
| 18 | 49,6 | 41,6 | 45.6 |
| 19 | 44,4 | 40 | 42.2 |
| 20 | 60,8 | 43,6 | 52.2 |

**Tabla 5. Resultados de la prueba 1.B.**

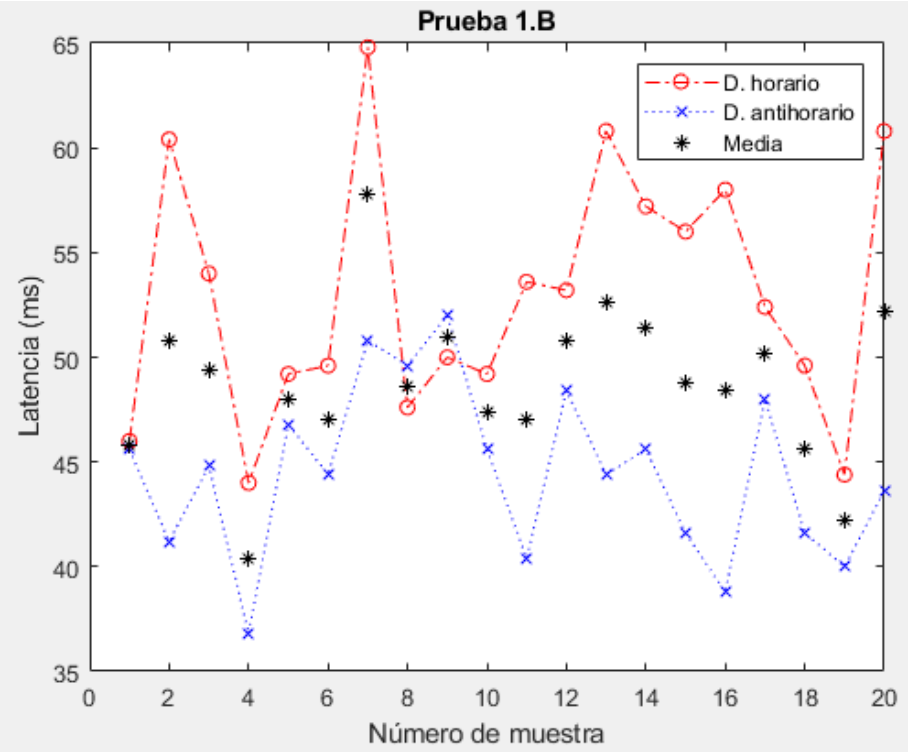


**Figura 43. Gráfica de la prueba 1.B.**

*Prueba 2* – Bela detecta movimiento – host renderiza audio. Tamaño de búfer: 256.

| Número | D. Horario (ms) | D. Antihorario (ms) | Media (ms) |
|---|---|---|---|
| 1 | 59,6 | 55,6 | 57.6 |
| 2 | 46,4 | 55,2 | 50.8 |
| 3 | 44,8 | 74,8 | 59.8 |
| 4 | 43,6 | 55,2 | 49.4 |
| 5 | 33,2 | 69,2 | 51.2 |
| 6 | 60 | 54,8 | 57.4 |
| 7 | 58,8 | 42,4 | 50.6 |
| 8 | 58,8 | 48 | 53.4 |
| 9 | 54 | 49,6 | 51.8 |
| 10 | 55,6 | 58,8 | 57.2 |
| 11 | 56 | 54 | 55 |
| 12 | 54 | 47,2 | 50.6 |
| 13 | 57,6 | 57,6 | 57.6 |
| 14 | 52 | 66,8 | 59.4 |
| 15 | 46,8 | 62,4 | 54.6 |
| 16 | 43,2 | 59,2 | 51.2 |
| 17 | 57,6 | 56,4 | 57 |
| 18 | 45,6 | 73,6 | 59.6 |
| 19 | 39,6 | 54 | 46.8 |
| 20 | 35,6 | 62,4 | 49 |

**Tabla 6. Resultados de la prueba 2.**

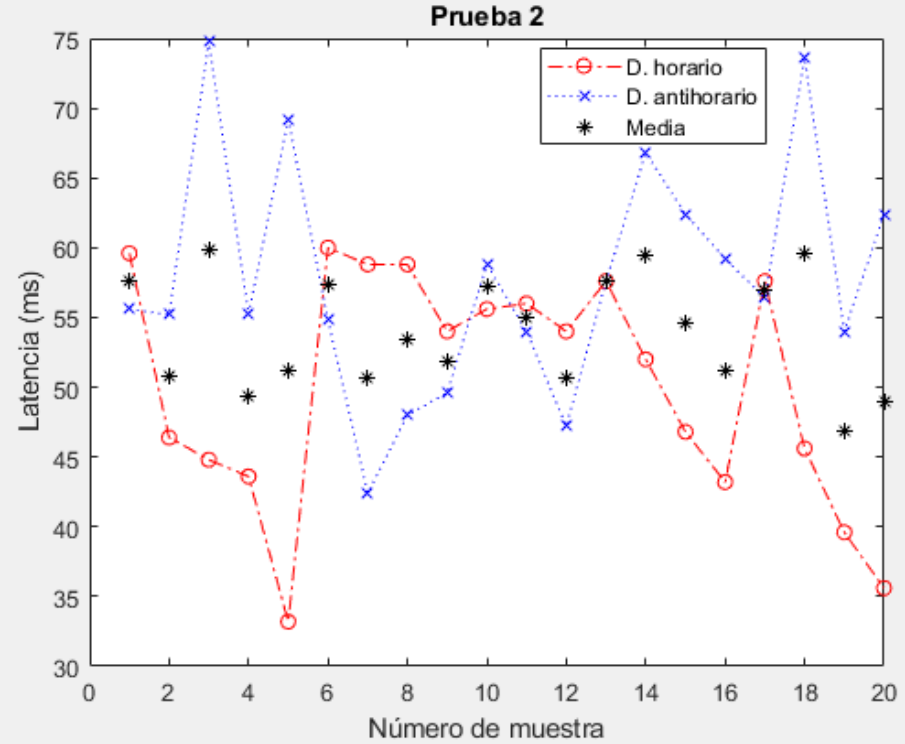


**Figura 44. Gráfica de la prueba 2.**

*Prueba 3* – Gafas de realidad virtual detectan movimiento – host renderiza audio. Tamaño de búfer: 256. [8]

| Número | D. Horario (ms) | D. Antihorario (ms) | Media (ms) |
|---|---|---|---|
| 1 | 79 | 58 | 68.5 |
| 2 | 67 | 64 | 65.5 |
| 3 | 59 | 61 | 60 |
| 4 | 82 | 51 | 66.5 |
| 5 | 90 | 76 | 83 |
| 6 | 79 | 60 | 69.5 |
| 7 | 82 | 66 | 74 |
| 8 | 68 | 62 | 65 |
| 9 | 65 | 61 | 63 |
| 10 | 74 | 46 | 60 |
| 11 | 77 | 68 | 72.5 |
| 12 | 79 | 63 | 71 |
| 13 | 79 | 69 | 74 |
| 14 | 62 | 56 | 59 |
| 15 | 64 | 68 | 66 |
| 16 | 76 | 60 | 68 |
| 17 | 73 | 66 | 69.5 |
| 18 | 57 | 55 | 56 |
| 19 | 69 | 73 | 71 |
| 20 | 86 | 57 | 71.5 |

**Tabla 7. Resultados de la prueba 3.**

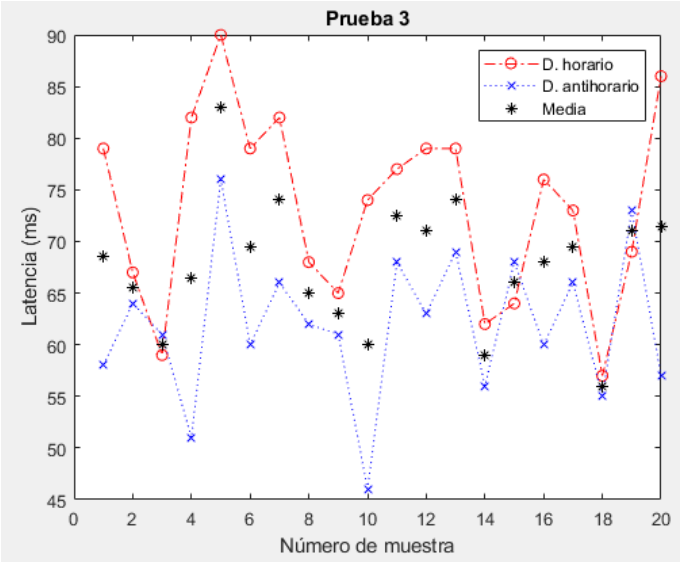


**Figura 45. Gráfica de la prueba 3.**

[8] Estos datos tienen una precisión menor que los recogidos anteriormente debido a que los tiempos de respuesta son mayores, lo que obliga a aumentar la escala de tiempo (25 ms).

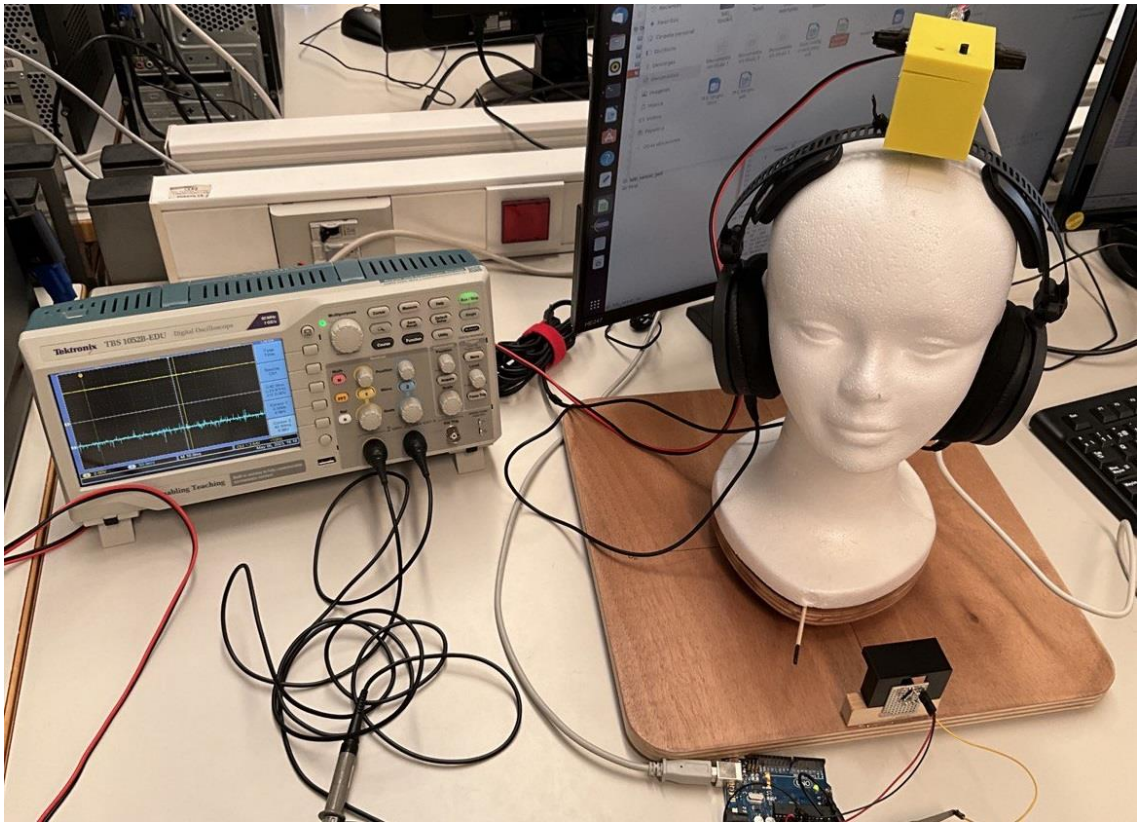


**Figura 46. Sistema de medidas, prueba 1.**

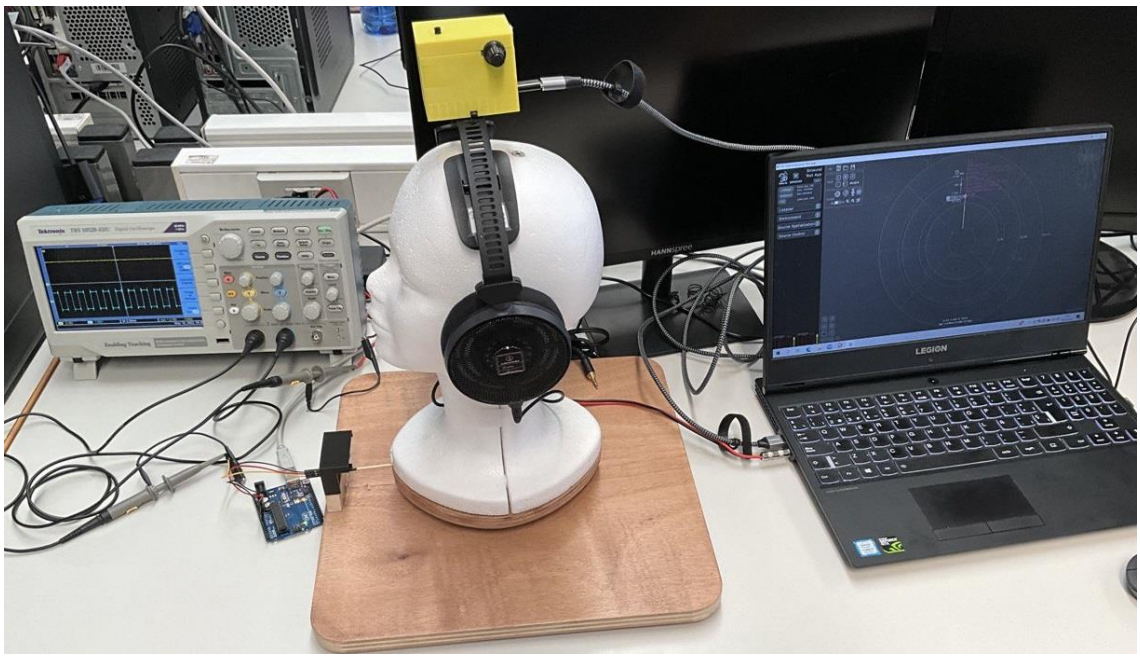


**Figura 47. Sistema de medidas, prueba 2.**

**Figura 48. Sistema de medidas, prueba 3.**

En la siguiente tabla se muestra un desglose de diferentes valores estadísticos obtenidos tras analizar los datos:

| | Medida | Media (ms) | Mediana (ms) | Varianza (ms) | Desviación típica |
|---|---|---|---|---|---|
| P. 1.A | Horario | 50.285 | 48 | 32.6392 | 5.7131 |
| | Antihorario | 46.06 | 47.4 | 31.7857 | 5.6379 |
| | Media par | 48.1725 | 48.8 | 12.9320 | 3.5961 |
| P. 1.B | Horario | 53.04 | 52.8 | 34.9036 | 5.9079 |
| | Antihorario | 44.5 | 44.6 | 16.6211 | 4.0769 |
| | Media par | 48.77 | 48.7 | 14.2917 | 3.7804 |
| P. 2 | Horario | 50.14 | 53 | 69.4783 | 8.3354 |
| | Antihorario | 57.86 | 56 | 71.8867 | 8.4786 |
| | Media par | 54 | 54 | 16.0253 | 4.0032 |
| P. 3 | Horario | 73.35 | 75 | 83.2921 | 9.1265 |
| | Antihorario | 62 | 61.5 | 52 | 7.2111 |
| | Media par | 67.675 | 68.25 | 39.4546 | 6.2813 |

**Tabla 8. Desglose de medidas estadísticas**

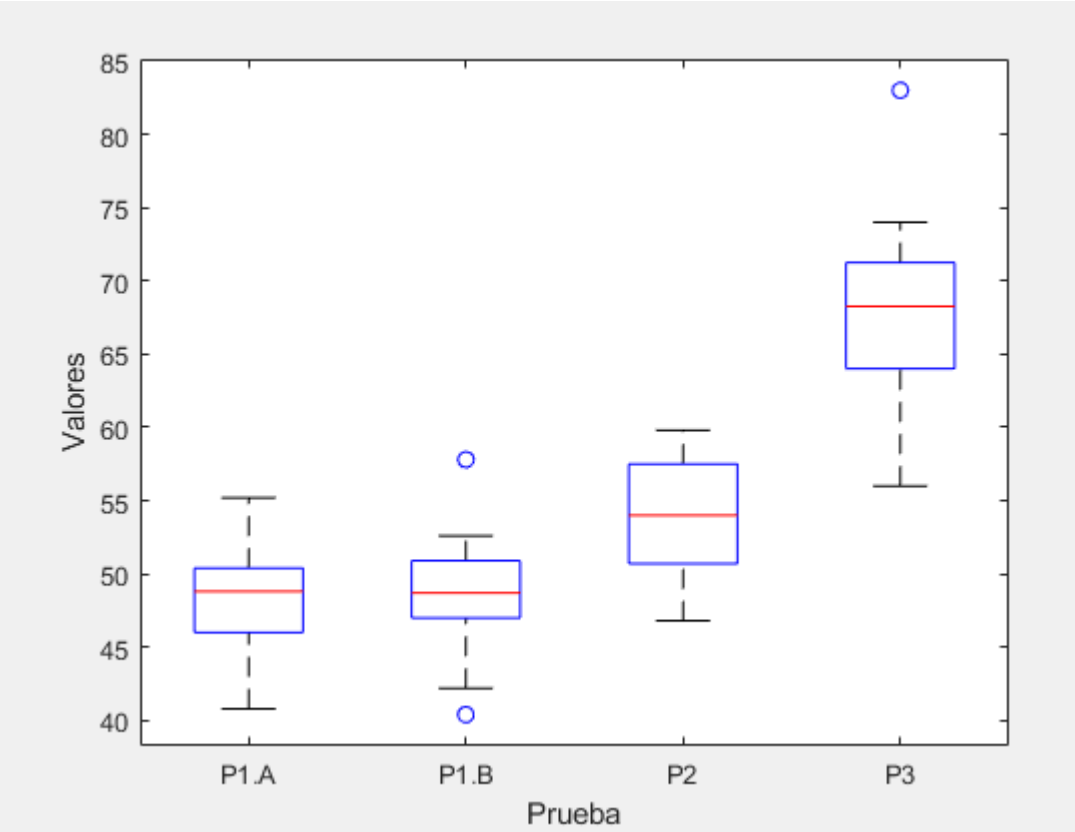


**Figura 49. Diagrama de cajas con los valores de las pruebas.**

## 4.5. Discusión de los resultados

Gracias a la Tabla 8 podemos tener una idea de cómo se comportan las medidas obtenidas: lo primero que se puede observar es la gran dispersión en todas ellas (también se puede apreciar observando cada gráfica individual).

Como los valores de varianza y desviación típica han resultado ser bastante elevados en muchas ocasiones, se ha decidido incluir la mediana al set de medidas estadísticas: se ha observado que en todos los casos (menos en la prueba 1.A.), la diferencia entre media y mediana ha sido menor a dos unidades. Esto nos indica que, aunque los valores sean dispersos, la media parece ser un indicador aceptable del centro de los valores.

En definitiva, estos valores no nos permiten caracterizar con precisión nuestro sistema, pero sí que nos ayuda a conocer cuál es el orden de magnitud al que se acerca, y tener una idea sobre cuál debería ser su valor medio aproximado.

A la hora de realizar las medidas detectamos un problema relacionado con el sensor, el cual se solventa con la toma de medidas por pares: descubrimos que el propio sensor tiene derivas, es decir, su orientación no es siempre exacta:

- Por un lado, a la hora de colocar la plataforma buscando el inicio de la señal de audio, si se realizaba demasiado lento (moviendo la plataforma con movimientos muy pequeños), el BNO055 iba autoajustándose. Esto le provocaba al usuario una sensación de que nunca se llegaba a entrar en la zona de reproducción de audio.
- Por otro lado, a veces tras la toma de las 40 medidas, si se rotaba la cabeza para dejarla en posición de cero con respecto al sensor, el audio estaba apagado o había empezado a sonar desde antes. Aunque pensamos que el motivo principal de esto son las derivas, deberíamos colocar unos tacos antideslizantes a la plataforma para evitar posibles errores extra.

En definitiva, somos conscientes de que las medidas se han tomado en unas condiciones que no son estrictamente ‘seguras’. Asumimos que ha habido errores en las mismas, pero pensamos que el promediado entre sentido horario y antihorario ha sido capaz de eliminar los sesgos sistemáticos, los cuales son los errores más importantes en este caso.

# Capítulo 5. Conclusiones y trabajo futuro

Tras cerca de tres meses realizando este proyecto hemos llegado al capítulo donde se esbozan las conclusiones. Ha sido, sin lugar a duda, un camino divertido y desafiante, en el cual se han aprendido muchas cosas y donde se han utilizado numerosos conocimientos aportados por el grado. Ha habido ciertos momentos difíciles en los que parecía que el aparato no terminaba de funcionar, así como momentos muy felices: exactamente cuando probamos por primera vez el audio renderizado por Bela.

Los tutores han ido ofreciéndonos desafíos retadores conforme íbamos superando los que estaban propuestos, lo que ha permitido enriquecer en gran medida este documento. Además, este trabajo nos ha brindado la posibilidad, a mi compañero Sergio y a mí, de realizar un último trabajo en pareja: hemos estado trabajando juntos durante la mayoría de las asignaturas de la carrera, pero debido al COVID la última vez que así lo hicimos fue hace más de tres años.

A la vista de los resultados hemos podido llegar a la conclusión de que el dispositivo creado cumple con las expectativas: estamos ante un sistema que funciona bien, que no tiene fallos y que es divertido y útil. El principal problema que hemos divisado es su falta de potencia: aunque el sistema funciona bien y es rápido de por sí, no es capaz de funcionar con más de una fuente de audio.

Las pruebas han sido especialmente delicadas: la toma de medidas es un procedimiento arduo propenso al sesgo constante por fallos humanos. El montaje del set de pruebas lleva su tiempo y la búsqueda del 'cero' requería cierta agilidad. También hay que tener cuidado con la fuerza que se le aplica a al maniquí, asegurándose que la velocidad angular a la hora de tomar un par de muestras sea lo más parecida posible (en un entorno en que se realizan las medidas 'a mano' esto se antoja bastante complicado). Sin olvidarnos que tenemos que evitar mover la plataforma de su posición inicial en todo momento.

Estudiando los resultados observamos que no hay una diferencia sustancial entre la media de tiempos de la prueba 1, en lo que a tamaño de búfer se refiere: 48.17 milisegundos con 256 muestras y 48.77 milisegundos usando 512. Si bien es cierto que se tarda más con un búfer más amplio, la diferencia es muy pequeña (este dato concuerda con el obtenido en el TFG asociado [1]).

Por otro lado, la diferencia de tiempos entre Bela 'lo hace todo' (46.77 milisegundos), Bela detecta orientación y se comunica por OSC (54 milisegundos) y el uso de un sistema de Realidad Virtual (67.68 milisegundos) demuestra que nuestro sistema es más que válido. Un punto interesante aparece a la hora de comparar los resultados de la prueba 1.A. y la prueba 2: aunque en los tres casos se ha utilizado la 3D Tune-In Toolkit, en estos dos se ha utilizado Bela con el BNO055. Viendo los resultados podemos observar que parte de ese tiempo está siendo usado por la librería y parte por la comunicación OSC, pero hay que tener en cuenta que ambas medidas comparten un tiempo: el que tarda el sensor en detectar la orientación. Sospechamos que cambiar dicho sensor por uno más rápido y potente podría mejorar en gran medida los datos obtenidos.

Por otro lado, la latencia medida con el sistema comercial es esperable, pues su hardware y software nativo realiza muchas más tareas que simplemente detectar orientación. Aun así, también es muy rápido y funciona muy bien.

En cuanto a los requisitos, se han cumplido todos los propuestos excepto una parte de un sub-requisito: no se han diseñado unas aberturas en la carcasa para la ventilación del dispositivo. Aunque a priori parecía suficiente con las aberturas que ya tenía de por sí la carcasa, en los últimos días abrimos la caja para tomar unas fotografías tras una sesión de medidas y notamos que la temperatura en su interior era elevada.

## 5.1. Posibles líneas futuras

Para acabar, a la hora de realizar este proyecto hemos descubierto o pensado una serie de puntos que sería interesante realizar en algún momento:

1. Mejora de la plataforma de pruebas: sin duda sería un gran avance motorizar el sistema para emplear exactamente la velocidad angular que necesitemos. Se podría ampliar este apartado añadiendo una pantalla que permitiese controlar dicha velocidad. También, recordamos, hay que añadir tacos antideslizantes a la plataforma.
2. Mejora de la plataforma giratoria: en caso de que no se contemple realizar la mejora anteriormente expuesta, resultaría muy útil añadir un asa o agarradera a la plataforma para favorecer un mejor agarre y control. También buscar una cabeza que no esté levemente inclinada.
3. Mejora del sistema de medida: otra gran mejora aplicable al sistema de pruebas tiene como objetivo el osciloscopio utilizado. En la asignatura de Instrumentación Virtual hemos aprendido a programar diferentes instrumentos de medida. Se podría diseñar un código (utilizando LabView o programándolo en C y luego añadiendo una interfaz gráfica con Qt) que realizase de forma automática la toma de muestras que se le pidan, guardándose información relevante de las mismas (como una instantánea de las curvas en cada medida). Esto agilizaría enormemente el proceso.
4. Ampliación de la usabilidad del dispositivo: una de las veces que utilizamos el dispositivo para ver cómo se comportaba nos percatamos de lo útil que sería añadir una pantalla que mostrase la información que queremos seleccionar o el estado del sistema. Se podría diseñar un mando (conectado por cable o inalámbrico) que incluya dicha pantalla.
5. Optimización del software: explorar una mejor optimización de las hebras.
6. Ampliación del software del dispositivo: siguiendo con el punto anterior, una vez creado dicho mando podría crearse por software algún tipo de videojuego en el que se tengan que buscar los sonidos en el ambiente.
7. Alternativas del modo de comunicación: también se contempló el uso de Bluetooth o Wifi para transmitir la orientación al host o archivos de audio a Bela, pero al final no se avanzó por esta rama.

# Apéndice A. Lista de materiales

En la siguiente tabla recopilamos todos los materiales necesarios para el montaje del dispositivo. Incluimos también el precio de los componentes electrónicos del disparador del sistema de pruebas:

| Objeto | Cantidad | Precio unidad (€) | Precio total (5) |
|---|---|---|---|
| Bela Mini (incluye PocketBeagle) | 1 | 153.79 | 153.79 |
| BNO055 | 1 | 31.80 | 31.80 |
| Potenciómetro 10 kΩ | 2 | 0.59 | 1.18 |
| Resistencia 1 kΩ ¼ W | 1 | 0.15 | 0.15 |
| Resistencia 330 Ω ¼ W | 1 | 0.15 | 0.15 |
| Resistencia 220 Ω ¼ W | 1 | 0.15 | 0.15 |
| Diodo LED | 1 | 0.24 | 0.24 |
| Botón de pulsador | 1 | 0.80 | 0.80 |
| Fotointerruptor GP1A57HR | 1 | 4.76 | 4.76 |
| Placa perforada doble cara estañada | 2 | 1.72 | 3.44 |
| Pines de conexión hembra | 1 tira | 1.00 | 1.00 |
| Pines de conexión macho (6 mm) | 1 tira | 0.80 | 0.80 |
| Pines de conexión macho (12 mm) | 1 tira | 1.11 | 1.11 |
| Plástico PLA | 46.3 g (45.61 m) | 0.0266 | 1.23 |
| | | TOTAL | 200.6 |

**Tabla en Apéndice A. 1. Lista de materiales (también conocido por BOM o *Bill of Materials*).**

Al precio total falta añadirle mínimo 60 euros en transporte (Bela y Bno055) y los posibles impuestos de importación.

# Apéndice B. Circuitos electrónicos

Recogemos en este apéndice los circuitos diseñados durante el desarrollo del documento en formato CAD. Todos han sido creados utilizando la herramienta Altium Designer.

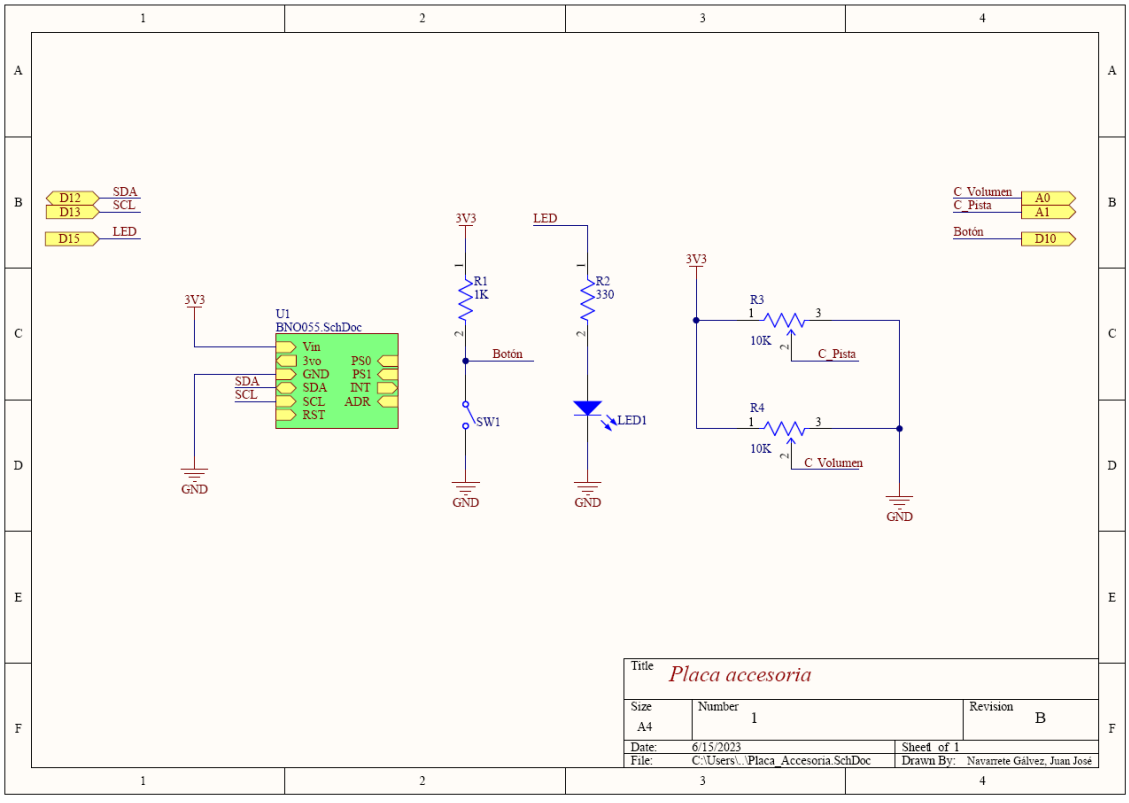


**Figura en Apéndice B. 1. Esquema eléctrico CAD de la placa accesoria.**

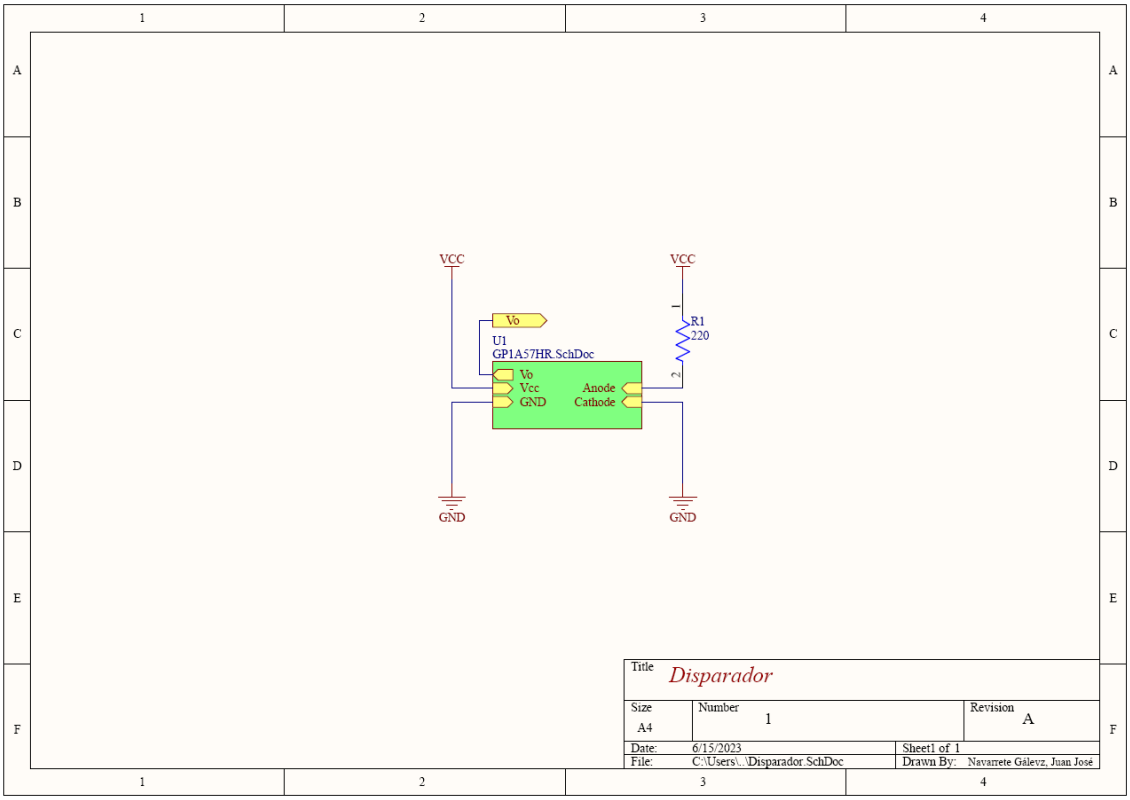


**Figura en Apéndice B. 2. Esquema eléctrico CAD del disparador.**

# Apéndice C. Tabla de requisitos

A continuación, se adjunta la tabla de requisitos que acompaña al Capítulo 2. Se ha decidido una presentación en estilo apaisado para facilitar la lectura de la misma.

Las columnas que componen la tabla son:

- ID o etiqueta de identificación. Permite la designación técnica del requisito.
- Palabra corta. Definición que permite definir al requisito de forma rápida y clara.
- Descripción. Descripción técnica del requisito, como aparece en el propio Capítulo 2 pero con ciertas modificaciones.
- Motivación. Resulta primordial explicar por qué se ha decidido incluir los diferentes requisitos.
- Prueba. Prueba pensada para comprobar si el requisito en cuestión se ha cumplido. Se ha intentado ser lo más específicos posible.
- ¿C? Para indicar si el requisito en cuestión ha sido cumplido. Leyenda: Sí → cumplido, No → no cumplido.

| ID | Palabra corta | Descripción | Motivación | Prueba | ¿C? |
|---|---|---|---|---|---|
| **R1** | Detección de orientación | El dispositivo debe detectar su orientación, utilizando un sensor inercial para su implementación. | Este requisito comprende uno de los pilares fundamentales del presente documento. La elección del sensor es un paso muy importante en el desarrollo del dispositivo, pues va a ser el encargado de captar los eventos del mundo exterior. | Una vez elegido el sensor e implementada la aplicación, se podrá probar este requisito. Para ello se configurará un sonido que esté virtualmente en frente del usuario, se vestirá el dispositivo y se girará la cabeza a ambos lados, observando si se escucha adecuadamente el sonido (al girar a la izquierda el sonido tiene mayor amplitud en el oído derecho). Además, el sonido debe escucharse fluido sin interrupciones. | Sí |
| **R2** | Diseño del Software | El dispositivo debe, utilizando la información de orientación del sensor, calcular y sintetizar la información de audio en consecuencia. | Este requisito comprende el otro pilar fundamental de la aplicación, donde el software tiene un gran protagonismo. El procesado del audio debe ser lo suficientemente rápido como para que se escuche adecuadamente. | Misma prueba que requisito 1. | Sí |
| **R2.1** | Uso BeagleBone Black | El sistema debe utilizar la placa BeagleBone Black (en formato normal o mini) como plataforma sobre la que se desarrollará el sistema. | El principal motivo por el que elegimos dicha placa como plataforma de desarrollo es porque va ligada a Bela. | Se ha utilizado dicho elemento en la solución final. | Sí |
| **R2.2** | Uso Bela | La placa BeagleBone Black debe ser complementada con la extensión Bela (en formato normal o mini). | Bela es un complemento diseñado para procesar datos de audio y sensores con ultra baja latencia, usando un sistema empotrado. Debido a esto es el candidato ideal para nuestra aplicación. | Se ha utilizado dicho elemento en la solución final. | Sí |
| **R2.3** | Uso 3Dti Toolkit | El sistema debe utilizar el 3D Tune--In Audio ToolKit diseñado por el grupo DIANA. | La 3D Tune-In Toolkit es el software diseñado por el grupo DIANA de nuestra escuela. Dicho software permite el procesamiento necesario para posicionar el audio en tres dimensiones. | Las especificaciones sobre este requisito se encuentran en [1]. | Sí |

**Tabla en Apéndice C. 1 de 5. Tabla de requisitos.**

| ID | Palabra corta | Descripción | Motivación | Prueba | ¿C? |
|---|---|---|---|---|---|
| **R3** | Comunicación Bela - Sensor | Debe haber una comunicación software estable entre el sensor y Bela. Además, el sensor se debe fijar adecuadamente a la placa. | Para garantizar una comunicación estable se debe diseñar un controlador que permita conectar el sensor con un dispositivo ajeno a Arduino. Por otro lado, existe la necesidad de fijar adecuadamente el sensor a Bela para evitar movimientos indeseados y para garantizar una conexión física adecuada. | Se ha creado un circuito con conectores hembra que se colocará sobre el complemento Bela. Una vez cargado el controlador, se creará un software que permitirá observar por consola los datos leídos del sensor: no se debe observar ningún tipo de error. | Sí |
| **R3.1** | Controlador Software | Debe existir una comunicación software implementada a través de un controlador. | Debido a la gran variedad de plataformas existentes en el mercado, es necesario crear un software específico para Bela. | En la solución final se hace uso de un controlador que comunique el sensor con la placa. | |
| **R3.2** | Placa Hardware | El sistema debe incluir una conexión física estable que fije el sensor a la placa y evite movimientos indeseados. | El sensor debe acoplarse de forma firme a Bela, de manera que se establezcan las conexiones físicas necesarias para la comunicación. | El sistema final incluye una placa hardware donde descansa el sensor. Dicha placa se adapta a Bela y se conecta firmemente. | |
| **R4** | Interfaz del dispositivo | El usuario debe poder controlar ciertas variables del sistema, tales como la amplitud de la señal, la pista de audio y la posibilidad de fijar un punto de referencia como cero. Para ello se deberá diseñar un hardware específico. | En la mayoría de los aparatos electrónicos es necesaria la implementación de un interfaz que permita la interacción usuario - máquina. En el ámbito de nuestra aplicación buscamos un control de tipo hardware sobre tres variables básicas del sistema: amplitud del sonido, pista de audio, punto de referencia. El apartado software será tratado en [1]. | Cuando el usuario se coloque el dispositivo y éste se encienda, el usuario debe ser capaz de cambiar el volumen según el sistema contemplado, aumentándolo y disminuyéndolo según se accione. Posteriormente se probará el cambio de pista, el cual debe funcionar acorde a la implementación, asegurándose que se puede acceder a todas las pistas. Por último, para comprobar el punto de referencia se debe mirar a la derecha y accionar el control: cuando el usuario vuelva a mirar hacia el frente debería escuchar el audio desplazado. Repetir con el lado izquierdo. | Sí |
| **R5** | Carcasa | El dispositivo debe ir contenido en una carcasa diseñada para tal propósito. | Es fundamental el diseño de una carcasa que proteja y guarde al dispositivo. | Se ha diseñado una carcasa para guardar el dispositivo. | Sí |

**Tabla en Apéndice C. 2 de 5. Tabla de requisitos.**

| ID | Palabra corta | Descripción | Motivación | Prueba | ¿C? |
|---|---|---|---|---|---|
| **R5.1** | Carcasa: Seguridad | Se debe garantizar la seguridad del usuario en todo momento. | La carcasa debe guardar adecuadamente todos los componentes electrónicos que contiene el dispositivo, evitándose así el acceso directo a los mismos por parte del usuario. | La carcasa no permite el acceso a ningún componente electrónico exceptuando las conexiones USB, mini USB y conector Jack. | Sí |
| **R5.2** | Carcasa: Ventilación y Agarre | La carcasa debe permitir un funcionamiento óptimo del dispositivo, asegurando la ventilación del mismo y ofreciendo un agarre adecuado que evite la captación de movimientos indeseados. | La ventilación es un punto fundamental en el diseño de la carcasa: todo circuito electrónico debe tener una ventilación adecuada para evitar sobrecalentamientos. Por otro lado, un agarre firme al lugar donde se quiera utilizar el dispositivo es fundamental para evitar movimientos que generen ruido. | La carcasa cuenta con un sistema de ventilación visible. La carcasa tiene una apertura que permite fijar el dispositivo donde se desee, sin holguras. | **No** |
| **R5.3** | Carcasa: Conexiones | La carcasa debe contar con aberturas para posibles conexiones. | En concordancia con lo estipulado en el requisito 5.1., es necesaria la existencia de estas aberturas para poder acceder libremente a las conexiones de la placa. | La carcasa cuenta con aberturas que permitan la conexión/desconexión cómoda de cables a los puertos USB, mini USB y Jack. | Sí |
| **R5.4** | Carcasa: Controles | La carcasa debe contar con elementos que permitan utilizar adecuadamente los controles. | Se deben incluir elementos que permitan acceder cómodamente a los diferentes controles del dispositivo. | Se han utilizado embellecedores sobre los potenciómetros y botoneras sobre los botones. Dichos elementos se mantienen fijos en su sitio, incluso tras un uso prolongado. | Sí |
| **R6** | Diodo LED | El dispositivo debe incluir un diodo LED que indique el estado del sistema. | Otro punto importante a la hora de diseñar el interfaz usuario - máquina es la inclusión de algún mecanismo que nos permita observar el estado de nuestro dispositivo. En nuestro caso hemos optado por un diodo LED. | Al encender el dispositivo el diodo se mantendrá encendido un tiempo (alrededor de un minuto). Si el dispositivo falla, el LED se quedará en estado encendido. Si no falla, parpadeará. | Sí |

**Tabla en Apéndice C. 3 de 5. Tabla de requisitos**

| ID | Palabra corta | Descripción | Motivación | Prueba | ¿C? |
|---|---|---|---|---|---|
| **R7** | Vestible | El dispositivo debe considerarse vestible. Para ello, el dispositivo debe poder colocarse sin mayor dificultad sobre la diadema de los auriculares que esté utilizando el usuario. | Nuestro enfoque en este proyecto es el de crear un dispositivo que pueda ser usado cómodamente, sin necesidad de ser manipulado (exceptuando los controles). Se busca que el dispositivo descanse sobre la diadema de un auricular. | El dispositivo se fija a la diadema de cualquier auricular. El enganche no es permanente. | Sí |
| **R7.1** | Cómodo | El dispositivo debe ser lo más cómodo posible. | Nos referimos a tamaño y peso adecuados, es decir, que no sea demasiado grande o pesado como para que llevarlo sobre la cabeza sea molesto. | El dispositivo apenas se nota cuando se viste. | Sí |
| **R7.2** | Autónomo | El dispositivo debe poder alimentarse mediante una batería, ya sea interna o externa. | Este requisito forma parte de la esencia del proyecto: la intención es poder utilizarlo libremente, en cualquier lugar. | Se puede encender el dispositivo sin necesidad de estar conectados a un host o a una toma de corriente. Uso de batería externa o interna. | Sí |
| **R8** | Comunicación OSC | el dispositivo debe poder mantener una comunicación con un host utilizando mensajes OSC, los cuales contendrán información sobre orientación. | Dicha comunicación nos permite acceder a ciertas características del dispositivo: corrección de errores y medidas de latencia. | Se pueden leer mensajes de entrada OSC en el host: un ordenador con Windows o Mac. | Sí |
| **R9** | Latencia | El dispositivo debe tener una latencia conocida, la cual debe estar por debajo de un máximo aceptable para procesado de audio. | Todo sistema que trabaje con audio debe ser lo suficientemente rápido como para que el usuario no detecte ningún tipo de fallo (como una reproducción no fluida). Como estamos realizando una toma y lectura de datos en una plataforma de desarrollo desconocida, necesitamos medir la latencia para constatar que el resultado es válido. | Se ha realizado un estudio de latencia y ha dado un valor por debajo de 70 ms (según [22]). No se han detectado errores en la reproducción. | Sí |

**Tabla en Apéndice C. 4 de 5. Tabla de requisitos.**

| ID | Palabra corta | Descripción | Motivación | Prueba | ¿C? |
|---|---|---|---|---|---|
| **R10** | SP (Sistema de Pruebas) | Para cumplir con el requisito 10, se debe diseñar un sistema que permita medir los valores de latencia del dispositivo. | Este sistema será fundamental para poder medir la latencia de nuestro dispositivo, y de otros que queramos probar. | Se ha creado un sistema o aparato que permite realizar las medidas de latencia. | Sí |
| **R10.1** | SP: soporte | El sistema de pruebas debe sostener adecuadamente el dispositivo a testear. | Como se va a utilizar unos auriculares con diadema, es necesario que el sistema de medida permita un uso cómo y fiable de los mismos. | El sistema utilizado sostiene los auriculares de diadema con el dispositivo (u otro aparato a probar) de manera prolongada. Lo que vaya a colocarse encima no debe caerse o inclinarse. | Sí |
| **R10.2** | SP: evento real | El sistema de pruebas debe contar con un disparador de evento real. | Es necesario generar un evento real pues éste nos servirá como punto de referencia para medir la diferencia de tiempo con el evento generado. Además, es necesario que la generación de evento real sea más rápida que el procesado del audio (en caso contrario, las medidas serían erróneas). | Se ha elegido un dispositivo que tarde menos de un milisegundo en generar el evento real. | Sí |
| **R10.3** | SP: medida | El sistema de pruebas debe incluir un dispositivo que nos permita comparar el evento real con la respuesta del dispositivo medido. | Para realizar la prueba es necesario la inclusión de un elemento medidor que nos permita visualizar y calcular la diferencia de tiempo entre los pulsos. | Se ha utilizado un dispositivo que permite diferenciar las señales de forma clara. | Sí |
| **R11** | Comparación | Para tener más información del dispositivo, se deben comparar los resultados obtenidos con los de otros sistemas. | Este punto es clave para comprobar en qué medida nuestro dispositivo es más rápido y elegible frente a otros sistemas, ya sean propios o comerciales. | Se realizarán medidas y se calcularán valores estadísticos de tres sistemas: nuestro dispositivo captando movimiento y renderizando audio; nuestro dispositivo captando movimiento y comunicándose con un host por mensajes OSC para que renderice el audio; usar unas gafas de realidad virtual para detectar el movimiento y que un host realice el renderizado de audio. Para considerarse elegible, nuestro sistema debe ser el más rápido de todos, es decir, que tenga la media de latencia total más baja. | Sí |

**Tabla en Apéndice C. 5 de 5. Tabla de requisitos.**

# Referencias


[1] S. Florido-Becerra, “Audio binaural 3D en Bela”, Universidad de Málaga, Málaga, junio, 2023.

[2] M. Cuevas-Rodríguez, L. Picinali, D. González-Toledo, C. Garre, E. de la Rubia-Cuestas, L. Molina-Tranco, A. Reyes-Lecuona, “3D Tune-In Toolkit: An open-source library for real-time binaural spatialisation”, PLoS One, vol. 14, no. 3, p. e0211899, marzo, 2019, DOI: 10.1371/journal.pone.0211899.

[3] A. Reyes-Lecuona, “Audio inmersivo, personalizado e interactivo para RV/RA e investigación en psicoacústica, proyectos SONICOM y SAVLab”, actas de *INTERACCIÓN 2022*, Teruel (España), septiembre, 2022.

[4] M. Cuevas-Rodríguez, “3D Binaural Spatialisation for Virtual Reality and Psychoacoustics”, Universidad de Málaga, Málaga, julio, 2022.

[5] L. Baiheng, Z. Wen, “Research on China’s AR/VR Industry Transformation Strategies in the Context of Domestic and Foreign Market Changes”, actas de *E3S Web Conf.*, Vol.235, Nº 03037, Dali (China), febrero, 2021. DOI: 10.1051/e3sconf/202123503037

[6] P. Milgram, H. Takemura, A. Utsumi, F. Kishino, “Augmented reality: A class of displays on the reality-virtuality continuum”, actas de *SPIE – The International Society for Optical Engineering*, 2351, Kyoto (Japón), enero, 1994. DOI: 10.1117/12.197321.

[7] E. Aguilera, J. Lopez, P. Gutierrez, M. Cobos, "An Immersive Multi-Party Conferencing System for Mobile Devices Using Binaural Audio", actas de *55th International Conference: Spatial Audio*, Paper 4-3, Valencia (España), agosto, 2014.

[8] R. Fernández, “Tamaño del mercado de la realidad virtual (RV), realidad aumentada (RA) y realidad mixta (RM) a nivel mundial de 2021 a 2024”, *es.statista.com*, agosto, 2022. [Online]. Disponible en: https://es.statista.com/estadisticas/662028/tamano-de-mercado-mundial-de-la-realidad-virtual-aumentada-extendida/ [Última visita: 13 de diciembre, 2022].

[9] M. Romanov, P. Berghold, D. Rudrich, M. Zaunschirm, “Implementation and Evaluation of a Low-cost Head- tracker for Binaural Synthesis”, actas de *142nd Convention Audio Engineering Society*, pp. 1–6, Berlín (Alemania), 2017.

[10] A. Reyes-Lecuona, L. Picinali, M. Cuevas-Rodríguez, D. González-Toledo, L. Molina-Tranco, C. Garre, E. de la Rubia-Cuestas, A. Rodríguez-Rivero, “3dti AudioToolkit”, *github.com*, octubre, 2022. Disponible en: https://github.com/3DTune-In/3dti_AudioToolkit [Última visita: 13 de diciembre, 2022].

[11] C. Long, J. Kridner, “Meet Beagle™: Open Source Computing”, *beagleboard.org*, noviembre, 2021. Disponible en: https://beagleboard.org/ [Última visita: 13 de diciembre, 2022].

[12] A. McPherson, V. Zappi, “An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black”, actas de *Audio Engineering Society Convention 138*, Paper 9331, Varsovia (Polonia), mayo, 2015.

[13] Bosch Sensortec GmbH, “BNO055 Intelligent 9-axis absolute orientation sensor”, BST-BNO055-DS000-14 v1.4, Reutlingen (Alemania), junio, 2016.

[14] K. Townsend, “Adafruit_BNO055”, Adafruit Industries, *github.com*, mayo, 2022. Disponible en: https://github.com/adafruit/Adafruit_BNO055 [Última visita: 22 de junio, 2023].

[15] B. Stewart, “Put a Bela On Your Head”, actas de *Soundstack 2018*, 2018, Londres. Disponible en: https://github.com/theleadingzero/belaonurhead. [Última visita: 12 de junio, 2023].

[16] S. Cowen, “imumaths”, *github.com*, septiembre, 2021. Disponible en: https://github.com/supercamel/imumaths [Última visita: 22 de junio, 2023].

[17] V. Rodríguez Bouza, “Sobre los cuaterniones, álgebras de Lie, y matrices de Pauli. Teoría básica y aplicaciones físicas.”, Repositorio Institucional de la Universidad de Oviedo, Oviedo (España), 2012.

[18] S. Benbouz, “DIY Mechanical Macro Keypad”, *thingiverse.com*, septiembre, 2022. Disponible en: https://www.thingiverse.com/thing:5535019 [Última visita: 15 de junio, 2023].

[19] L. Fotr, “Potenciometer-KNOB”, *thingiverse.com*, febrero, 2020. Disponible en: https://www.thingiverse.com/thing:4188050 [Última visita: 15 de junio, 2023].

[20] J. Lamas, C. C. L. Silva, M. Silva, S. Mouta, J. C. Campos, J. A. Santos, “Measuting end-to-end delay in real-time auralisation systems”, actas de *Euronoise 2015*, Maastricht (Países Bajos), junio, 2015.

[21] J. Estrella, J. Plogsties, “Motion-to-Sound Latency Procedure for VR Sound Reproduction”, actas de *Audio Engineering Society Convention 142*, número 345, Berlín (Alemania), mayo, 2017.

[22] J. D. Miller, M. R. Anderson, E. M. Wenzel, B. U. McClain, “Latency measurement of a real-time virtual acoustic environment rendering system”, actas de *2003 International Conference on Auditory Display*, Boston, EEUU, julio, 2003.