



**UNIVERSIDAD
DE GRANADA**

**TRABAJO FIN DE GRADO
INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN**

**Implementación de un Beamformer
acústico mediante un array Ad-Hoc de
smartphones**

**Sincronización temporal de dispositivos móviles mediante
una excitación acústica.**

Autor

Sergio Zapata Caparrós

Directores

Antonio Miguel Peinado Herreros
Ángel Manuel Gómez García



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN**

—
Granada, Julio de 2022



Implementación de un Beamformer acústico mediante un array Ad-Hoc de smartphones.

Sincronización temporal de dispositivos móviles mediante
una excitación acústica.

Autor

Sergio Zapata Caparrós

Directores

Antonio Miguel Peinado Herreros
Ángel Manuel Gómez García



DEPARTAMENTO DE TEORÍA DE LA SEÑAL, TELEMÁTICA Y
COMUNICACIONES

Granada, Julio de 2022

Implementación de un Beamformer acústico mediante un array Ad-Hoc de smartphones: Sincronización temporal de dispositivos móviles mediante una excitación acústica.

Sergio Zapata Caparrós

Palabras clave: sincronización, beamforming, array, directividad, correlación, ruido, retardo, localización de fuentes sonoras

Resumen

Numerosas situaciones en las que se desea realizar un procesamiento de señales acústicas, requieren de una buena sincronización entre señales, con el objetivo de cometer el menor error posible en las aplicaciones que de ello dependan. Entre estas situaciones, se encuentran por ejemplo la localización de fuentes sonoras o el proceso de beamforming.

En este proyecto se ha propuesto un método de sincronización alternativo, analizado y estudiado en un array ad-hoc de smartphones, con el propósito de poder llegar a implementar correctamente un beamformer y obtener finalmente el patrón de directividad experimental resultante.

En primer lugar, en esta memoria se proporcionarán las nociones teóricas relacionadas con el procesamiento espacial de señales acústicas. El proyecto continúa con la creación de una aplicación, que actuará como cliente en cada dispositivo móvil, y un servidor. Siendo dos herramientas necesarias para la correcta comunicación de todos los elementos del array y las cuales posibilitan el posterior tratamiento de las señales grabadas por los smartphones.

Una vez establecida una comunicación app-servidor fiable, se procederá al desarrollo del método de sincronización y, para concluir, se comprobará el funcionamiento de la técnica desarrollada para una aplicación de beamforming Delay & Sum sobre un array con geometría lineal uniforme, compuesto por tres smartphones.

Implementation of an acoustic Beamformer using an Ad-Hoc array of smartphones: Temporal synchronization of mobile devices using acoustic excitation

Sergio Zapata Caparrós

Keywords: synchronization, beamforming, array, directivity, correlation, noise, delay, source localization

Abstract

Several situations in which an acoustic signal processing is desired, an accurate synchronization between signals is required, in order to make the least possible error in the applications that depend on it. These situations are, for instance, the location of sound sources or a beamforming process.

In this paper, an alternative synchronization method has been proposed, studied in an ad-hoc array of smartphones, with the purpose of being able to correctly implement a beamformer and finally obtain the experimental directivity pattern resulting from the array.

First of all, some theoretical notions related to acoustic signal processing will be provided. The project continues with the creation of an app, which will play the role of a client on each smartphone, and a server, being these tools necessary for the correct communication between all the elements of the array and which enable the subsequent processing of the signals recorded by the smartphones.

Once a reliable app-server communication is achieved, the synchronization method will be developed and, eventually, the operation of the technique developed for a beamforming Delay & Sum application on an array with uniform linear geometry, composed of three smartphones.

Yo, **Sergio Zapata Caparrós**, alumno de la titulación INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 77145800M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Sergio Zapata Caparrós

Granada a 8 de Julio de 2022.

D. **Antonio Miguel Peinado Herreros**, Profesor del Área de Teoría de la Señal del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

D. **Ángel Manuel Gómez García**, Profesor del Área de Teoría de la Señal del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Implementación de un Beamformer acústico mediante un array Ad-Hoc de smartphones*, ha sido realizado bajo su supervisión por **Sergio Zapata Caparrós**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de Julio de 2022.

Los directores:

Antonio Miguel Peinado Herreros Ángel Manuel Gómez García

Agradecimientos

Primeramente, me gustaría agradecer a mis tutores Antonio y Ángel, los cuales me han facilitado mucho la realización del proyecto, resolviéndome diversos tipos de dudas y proponiéndome un abanico de alternativas cuando algún problema surgía. Han sido unos directores de trabajo excelentes.

A mi familia, que siempre ha estado apoyándome en los cuatro años de carrera y en el desarrollo del trabajo, facilitándome algunos materiales necesarios para pruebas experimentales y animándome día a día.

También quisiera agradecer tanto a la universidad de Granada como a los profesores que han contribuido a mi formación y me han llenado de ganas y curiosidad por el ámbito de las telecomunicaciones.

Índice general

1. Introducción	1
1.1. Procesamiento de señales multicanal	1
1.1.1. Concepto de array Ad-Hoc	5
1.2. Concepto de Beamforming	5
1.2.1. Tipos de Beamformers	6
1.3. Impacto en la actualidad	9
1.3.1. Diarización	9
1.3.2. Localización de fuentes	10
1.3.3. Realce de Voz	12
2. Planteamiento del problema	15
2.1. Distribución del proyecto	15
2.1.1. Requisitos de realización	15
2.1.2. Seguimiento del proyecto	16
2.1.3. Entornos de trabajo	17
2.2. Justificación del proyecto	17
3. Desarrollo del servidor externo	19
3.1. Propósito principal	19
3.1.1. Requisitos funcionales	19
3.1.2. Requisitos no funcionales	20
3.2. Construcción del servidor	21
3.2.1. Protocolo de comunicación y noción de <i>socket</i>	21
3.2.2. Implementación del servidor	23
3.3. Comunicación	25
4. Desarrollo de la aplicación	31
4.1. Introducción a los fundamentos de la aplicación	31
4.2. Funcionalidades	32
5. Sincronización	35
5.1. Recursos empleados	36
5.1.1. Correlación cruzada	36
5.1.2. Señales de sincronización	38

5.2.	Sincronización mediante el servidor	41
5.2.1.	Análisis	42
5.2.2.	Procedimiento	44
5.3.	Sincronización mediante “auto-chirps”	46
5.3.1.	Fundamento	46
5.3.2.	Procedimiento y conclusiones	47
5.4.	Aplicación del método de sincronización	49
5.4.1.	Preparación del entorno adecuado	49
5.4.2.	Resultados y limitaciones	51
6.	Implementación del Beamforming en un array lineal	53
6.1.	Análisis experimental	53
6.2.	Algoritmo y representación	56
6.3.	Evaluación de los resultados	61
7.	Conclusiones y trabajo futuro	65
7.1.	Balance del proyecto	65
7.2.	Perspectiva de futuro	67
7.2.1.	Reconocimiento automático de voz	67
7.2.2.	Sincronización en casas inteligentes	68
8.	Presupuesto del trabajo	71
8.1.	Inversión en capital humano	71
8.2.	Inversión en material	72
9.	Apéndice	77
9.1.	Apéndice A	77
9.2.	Apéndice B	78
9.3.	Apéndice C	78
9.4.	Apéndice D	79

Capítulo 1

Introducción

En este capítulo se revisará el fundamento teórico del concepto de “Array Signal Processing” y el concepto de “Beamforming”, lo cual servirá para la posterior comprensión y análisis de los resultados experimentales. Además, se contextualizarán este tipo de técnicas, explicando brevemente su ámbito de aplicación en la actualidad.

1.1. Procesamiento de señales multicanal

El procesamiento de señales multicanal, *Array Signal Processing* en inglés, se interpreta como un procesamiento de las señales en el espacio y en el tiempo.

Se parte del concepto de apertura, entendiéndose como un elemento que captura un cierto campo en una región del espacio. La respuesta de una apertura es direccional, es decir, la señal proporcionada por la apertura depende de la dirección de llegada (DOA). El patrón de directividad de una apertura lineal uniforme de longitud ‘L’ se muestra en la *Figura 1.1*.

Se aproximará el comportamiento de una apertura mediante la conformación de una serie de aperturas puntuales las cuales capturan el campo en una serie de puntos. Suponiendo que los elementos del array son idénticos e isotrópicos, la expresión general para su patrón de directividad es:

$$D(f, \alpha) = \sum_{n=0}^{N-1} w_n^*(f) e^{-2j\pi\alpha r_n} \quad (1.1)$$

Donde $w_n^*(f)$ se corresponde con el peso o contribución de cada elemento de la apertura general, r_n la posición de cada elemento y α el ángulo de radiación. Si dicho array tiene una geometría lineal uniforme (los elementos del array están uniformemente distribuidos en el eje X), se simplifica la

ecuación (1.1), de la manera:

$$D(f, \alpha) = \frac{1}{N} \frac{\sin(\pi\alpha_x Nd)}{\sin(\pi\alpha_x d)} e^{-2j\pi\alpha_x \frac{N-1}{2}d} \quad (1.2)$$

Siendo N el número de elementos, y manteniéndose estos equiespaciados a una distancia d . Al calcular el módulo del patrón de directividad resultante:

$$|D(f, u_x)| = \frac{1}{N} \frac{\sin(\pi u_x Nd/\lambda)}{\sin(\pi u_x d/\lambda)} \quad (1.3)$$

Donde u_x se corresponde con la componente X del vector unitario. Sustituyendo $u_x \in [-1, 1]$ en la ecuación (1.3) resulta el diagrama de directividad de una apertura lineal uniforme e independiente de la frecuencia de la Figura 1.1, con lo cual, se puede afirmar que un array lineal uniforme de N elementos, simula el comportamiento de una apertura lineal uniforme de longitud $L = Nd$.

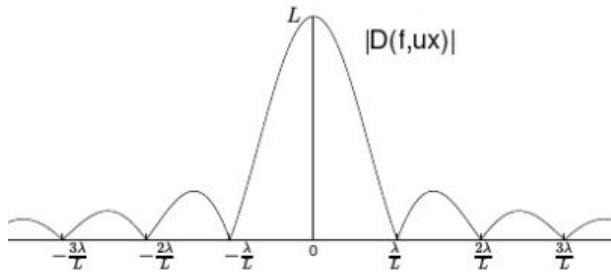


Figura 1.1: Directividad apertura

Se puede apreciar que la apertura actúa como un filtro espacial (fenómeno clave para hacer beamforming), diferenciándose claramente el lóbulo principal de los demás, menos directivos.

Un array lineal de sensores es, básicamente, una apertura conformada por un número sensores posicionados en una misma línea, simulando de esta manera el comportamiento de una apertura lineal de longitud la suma de las longitudes de los elementos del array más la separación entre ellos. En la Figura 1.2 se puede observar un array lineal uniforme orientado sobre el eje X . Variando la longitud del array con una longitud de onda fija, se pueden lograr diferentes patrones de directividad, como son el “endfire” (con el máximo de directividad en 0° y 180°) y el “broadside” (con el máximo de directividad en 90°).

Según la ecuación (1.3), el máximo valor posible del patrón de directividad es $D_{max} = L$. El ancho del lóbulo principal, de acuerdo con [1], del patrón de directividad se puede controlar ajustando el cociente de L/λ . Esto

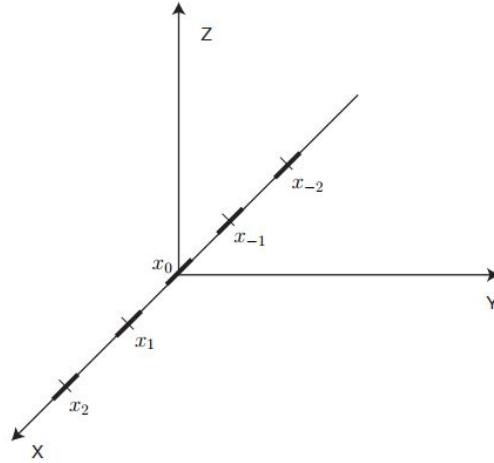


Figura 1.2: Agrupación de elementos

se justifica con la *Figura 1.3*, en la cual se muestra el patrón de directividad una apertura de longitud L . Además, en esta misma figura, se puede identificar una configuración del tipo “broadside”, ya que posee los máximos de directividad en $\pm 90\text{grados}$. A diferencia de la configuración del tipo endfire, que presenta los máximos de directividad en 0° y 180° .

El comportamiento de un array de sensores con una configuración tipo broadside tendría un comportamiento similar a los patrones mostrados en la *Figura 1.3*, por lo que, modificando d y λ se lograría modificar el ancho del lóbulo, pudiendo conseguir una recepción de señal lo más precisa posible. Un factor a tener en cuenta en el caso del array de sensores, es el *aliasing* espacial, el cual provoca la aparición de lóbulos secundarios. Los límites del aliasing espacial se basan en el teorema de Nyquist en el dominio de la frecuencia, de la forma:

$$f_{xs} = \frac{1}{d} \geq 2f_{x_{max}} \quad (1.4)$$

Siendo f_{xs} la frecuencia de muestreo espacial a lo largo del eje X, dada por:

$$f_{xs} = \frac{\sin\theta\cos\phi}{\lambda} \quad (1.5)$$

Maximizando la *ecuación (1.5)* y sustituyendo en la *ecuación (1.4)*, resulta el límite de aliasing espacial en:

$$d < \frac{\lambda_{min}}{2} \quad (1.6)$$

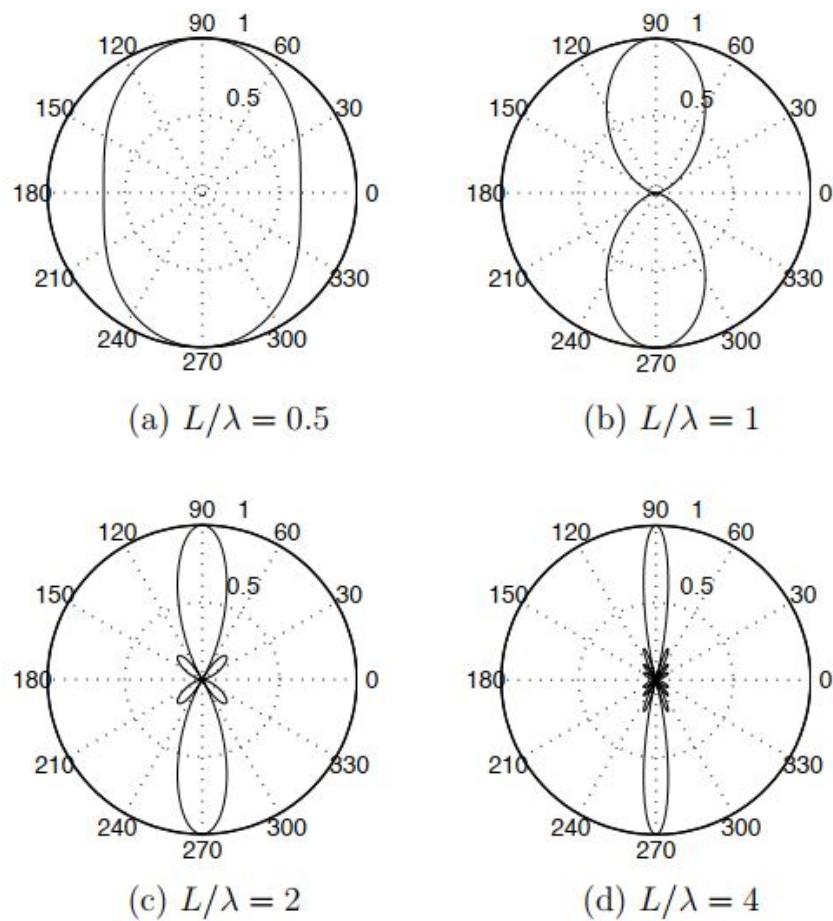


Figura 1.3: Ancho del lóbulo principal

Las agrupaciones de micrófonos consisten en un número de micrófonos que se combinan para filtrar espacialmente las ondas acústicas. La configuración geométrica del array permitirá filtrar las señales deseadas para diversas aplicaciones.

1.1.1. Concepto de array Ad-Hoc

Varios tipos de arrays de micrófonos son empleados para mejorar la calidad y la precisión temporal de la recepción de señales acústicas. Un tipo de array utilizado es el Ad-Hoc. Conforme aumenta el uso de los smartphones o tablets como herramientas de grabación, los arrays de micrófonos del tipo Ad-Hoc se hacen un hueco en conferencias y reuniones, ya que brindan una cobertura espacial amplia y flexible [3]. Al proporcionar una zona de cobertura amplia y flexible, los arrays Ad-Hoc son ideales para manejar varias fuentes de sonido ubicadas espontáneamente en un entorno ruidoso.

Un array Ad-Hoc de micrófonos consiste en una serie de dispositivos, por ejemplo, smartphones o tablets. Cada uno de estos se mantienen al mismo nivel en la red y no tienen posibilidad de comunicarse entre entre sí. Una configuración de array Ad-Hoc proporciona una gran escalabilidad, ya que se puede añadir a la red numerosos dispositivos, dependiendo únicamente de las limitaciones fijadas por el servidor al que se mantienen conectados. Un ejemplo explicativo de este tipo de array microfomal lo presenta [2], y sería la situación de varias tablets o móviles distribuidos espacialmente en una sala de una reunión, conectados a una red. El propósito de esta distribución es mejorar la calidad del sonido, dependiendo de la posición del interlocutor respecto a los elementos del array.

1.2. Concepto de Beamforming

El concepto de “Beamforming” agrupa las técnicas necesarias para la determinación de filtros específicos, que tienen como fin obtener un patrón de directividad con una forma y dirección determinadas.

Un “beamformer” se interpreta como un procesador, utilizado conjuntamente con una serie de sensores, para proporcionar una forma versátil de filtrado espacial. El objetivo es estimar la señal procedente de una dirección deseada, en un entorno ruidoso y en presencia de señales de interferencia. El “beamformer” es un filtro espacial que actúa separando las señales que han sufrido una superposición en frecuencia, pero que se han originado en diferentes localizaciones espaciales. Para determinar el patrón de directividad deseado se hace uso de filtros o, siguiendo la nomenclatura de la sección de procesamiento de señales multicanal, pesos. Estos filtros o pesos se com-

ponen de un módulo y una fase acordes al objetivo del filtro espacial.

$$w_n(f) = a_n e^{j\varphi_n(f)} \quad (1.7)$$

La *ecuación (1.7)* se corresponde con un esquema típico de amplitud (a_n) y fase ($e^{j\varphi_n(f)}$). Grossó modo, la amplitud determina la forma del patrón de directividad, mientras que la fase interviene en la orientación del lóbulo principal en un determinado ángulo. El caso que se planteará en este proyecto corresponde a un beamforming acústico.

1.2.1. Tipos de Beamformers

Antes de entrar en detalle en los distintos modelos, cabe diferenciar tres tipos de campos de ruido existentes para los cuales sería más adecuado el uso de un tipo de beamformer u otro y el criterio que se va a seguir para evaluarlos:

- Campo de ruido coherente: El ruido o interferencia se propaga en una determinada dirección con la misma energía.
- Campo de ruido incoherente: No existe correlación entre las señales de ruido captadas por los distintos sensores.
- Campo de ruido difuso: La propagación del ruido es en todas las direcciones con una misma energía.

Se define la ganancia del array como el cociente de la SNR a la salida y la SNR a la entrada del beamformer.

$$G(f) = \frac{SNR_o(f)}{SNR_i(f)} \quad (1.8)$$

Destacar que, tanto la SNR a la entrada y a la salida, como la ganancia, dependen de la frecuencia.

El factor de directividad, dado por la *ecuación (1.9)*, se define como el cociente del patrón de directividad en la dirección deseada y el patrón de directividad de todas las direcciones en general.

$$F_d(f, \phi_s, \theta_s) = \frac{|D(f, \phi_s, \theta_s)|^2}{\frac{1}{4\pi} \int_0^{2\pi} \int_0^\pi |D(f, \phi, \theta)|^2 \sin(\theta) d\theta d\phi} \quad (1.9)$$

A continuación se describirán las tres clases de beamforming más utilizadas: Beamforming Delay & Sum, Beamforming MVDR y Beamforming superdirectivo. Existen otras técnicas de beamforming, como el beamforming mediante subarrays o el beamforming adaptable, aunque no se tratarán en este documento. Se hará un mayor énfasis en la explicación del beamforming Delay & Sum, ya que será el empleado en la parte experimental del proyecto.

Delay & Sum

Este primer beamformer es el más comúnmente utilizado en procesos de beamforming debido a su simplicidad y a su robustez. El beamformer Delay & Sum o *DAS* ha sido utilizado en muchos estudios de localización de ruido en aeronáutica, como se verá posteriormente en la *sección 1.3.2*.

En el beamformer DAS, los pesos son representados mediante **retardos**. Siguiendo la estructura de la *ecuación (1.7)* resultaría:

$$w_n(f) = e^{j\omega t_n} / N \quad (1.10)$$

Con una contribución uniforme para cada elemento de $1/N$ y siendo $t_n = \varphi_n(f)/\omega$. Introduciendo el peso obtenido en la *ecuación (1.10)* operando con las exponenciales y sacando factor común, el patrón de directividad queda de la forma:

$$D(f, u - u_s) = \frac{1}{N} \sum_{n=0}^{N-1} e^{j2\pi f(\frac{ur_n}{c} - t_n)} \quad (1.11)$$

Donde r_n es el vector de posición del elemento n , u el vector unitario, f la frecuencia, c la velocidad de propagación y t_n un retardo aplicado. El objetivo es obtener el patrón de directividad en la dirección deseada u_s , por lo que fijando $t_n = \frac{u_s r_n}{c}$ conseguimos el direccionamiento comentado. Definimos instante de llegada de la señal justo en la dirección deseada ($u = u_s$) como $\tau_{n,s} = -t_n$.

Los pesos o filtros correspondientes a los N elementos del array del beamformer DAS son:

$$\mathbf{w}(f) = \frac{1}{N} (e^{j\omega t_0}, \dots, e^{j\omega t_{N-1}})^T = \frac{1}{N} \mathbf{d}(f, u_s) \quad (1.12)$$

Siendo $\mathbf{d}(f, u_s)$ el llamado *steering vector*.

Básicamente, el beamformer DAS hace la sumatoria de todas las versiones retardadas para cada elemento o micrófono, consiguiendo una superposición destructiva para aquellas señales que no lleguen en la dirección deseada y una superposición constructiva para las señales procedentes de la dirección deseada. El esquema básico de un beamformer delay & sum se ejemplifica en la *Figura 1.4*.

Se logra de esta manera un realzado de la señal procedente de la fuente de interés, mientras que se suprime la contribución de otras fuentes desde una dirección diferente a la determinada. Además de atenuar las señales procedentes de direcciones distintas a la de interés, este beamformer tiene la capacidad de suprimir la contaminación producida por ruido desde la señal de interés.

El beamformer DAS maximiza la *ecuación (1.8)* en un campo de ruido incoherente. Por otra parte, aunque el beamformer DAS man-

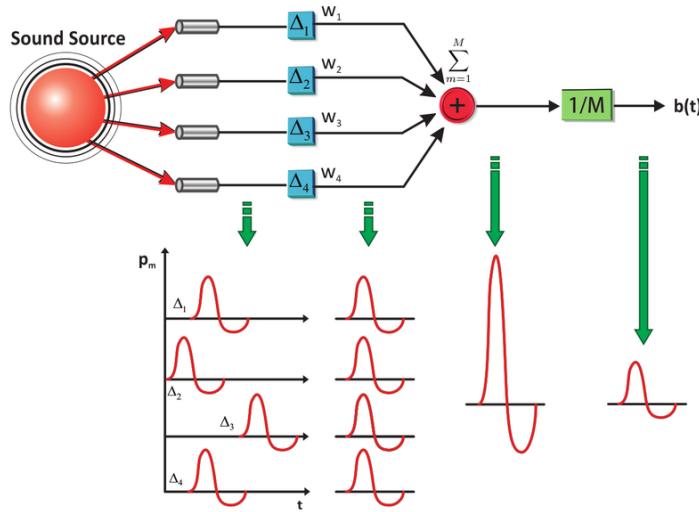


Figura 1.4: Esquema beamformer DAS

tiene un buen comportamiento ante ruido blanco, el factor de directividad que posee no es muy alto.

MVDR

El principal objetivo del beamformer MVDR (*Minimum Variance Distortionless Response*) es hacer mínima la contribución del ruido, como su propio nombre indica.

Considerando una señal deseada y el ruido capturado, en el dominio de la frecuencia se obtiene la señal:

$$\mathbf{X}(f) = S(f)\mathbf{d}_s(f) + \mathbf{V}(f) \quad (1.13)$$

Donde $S(f)$ es la fuente de señal, \mathbf{d}_s el steering vector y $\mathbf{V}(f)$ la contribución del ruido.

Se conoce que, la respuesta del array, apoyándose en la definición de patrón de directividad de la *ecuación (1.1)* es el siguiente:

$$Y(f) = D(f, u)S(f) = \mathbf{w}^H(f)\mathbf{X}(f) = \mathbf{w}^H(f)(\mathbf{d}_s(f)S(f) + \mathbf{V}(f)) \quad (1.14)$$

Siendo \mathbf{w} los pesos del beamformer.

El beamformer MVDR se centra en minimizar la contribución del ruido, por lo que se debe resolver un problema de optimización, imponiendo $w^H d_s(f)S(f) = S(f)$. El beamformer resultante, requiere el conocimiento

de la matriz de correlaciones espaciales (entre las señales de ruido captadas por los diferentes sensores) del campo de ruido.

En canales multirayectoria, el beamformer MVDR se emplea como un ecualizador, el cual filtra espacialmente las señales deseadas e intenta eliminar lo máximo posible las fuentes de ruido.

Teóricamente, el beamformer MVDR es capaz de mejorar siempre la SNR de una señal de interés captada por el sensor de referencia.

El beamformer MVDR maximiza la *ecuación (1.8)* en el campo de ruido cuya matriz de correlaciones es la empleada en el cómputo del beamformer.

Superdirective

Este tipo de beamformer es conocido por maximizar el factor de direccitividad. Sin embargo, es extremadamente sensible ante patrones de ruido no correlados, como puede ser el ruido blanco y ante pequeños errores de precisión y de ganancia en los elementos del array. Frente a ruido blanco, el beamformer superdirective además de realzar la señal de interés, amplificaría también el ruido blanco. Se puede entender como un filtro lineal. Más información sobre este tipo de beamformer se puede encontrar en [7].

El beamformer superdirective maximiza la *ecuación (1.8)* en un campo de ruido difuso.

1.3. Impacto en la actualidad

El procesamiento de señales multicanal tiene múltiples aplicaciones. Algunas de ellas son: radioastronomía, radar, sismología, tomografía, comunicaciones inalámbricas, sonar y tratamiento de audio.

Desde un punto de vista acústico y de manipulación de audio y voz, se va a exponer tres aplicaciones distintas. La primera será diarización de locutores, la segunda será la localización de fuentes sonoras y la tercera será realce de voz.

1.3.1. Diarización

La diarización entre locutores consiste en determinar los instantes de tiempo en los que cada locutor interviene, dada una señal de audio captada con array de micrófonos, como va a ser el caso. Se expone el ejemplo real de una sala de reunión con múltiples locutores y múltiples micrófonos distribuidos en la sala. Para aprovechar la capacidad de un array de múltiples

micrófonos, se pueden utilizar técnicas de beamforming, ya explicadas en una sección anterior para realzar la señal de interés.

El proceso de diarización planteado, se puede encontrar en [8]. Antes de ningún procedimiento que abarque a todos los micrófonos, se debe aplicar un filtro de Wiener a cada canal individual para eliminar el ruido aditivo. Teniendo en cuenta el tiempo de llegada de cada señal y el número total de micrófonos, se implementa un algoritmo que implica correlación cruzada entre señales, obteniendo de esta manera el canal que proporciona una mejor calidad de señal. Tras aplicar beamforming, computando los retrasos entre cada canal, se propone en [8] usar el algoritmo de Viterbi, el cual se encarga de hallar el camino o secuencia de estados con mayor probabilidad. El objetivo de este último paso es proporcionar una continuidad a la señal del locutor que está en ese momento hablando, estableciendo el retardo de la señal y filtrando la dirección del haz no deseada.

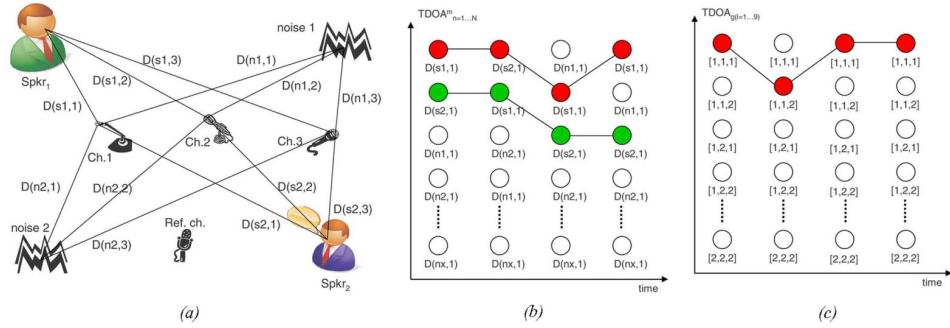


Figura 1.5: Decodificación de Viterbi. Las columnas de la prerepresentación trellis del algoritmo, corresponden a la diferencia de tiempos de llegada obtenidos con anterioridad.

En la *Figura 1.5* se esquematiza el algoritmo de Viterbi para un ejemplo de dos locutores, dos fuentes de ruido y micrófonos distribuidos. En el primer paso (b) para cada canal individual, se obtiene los dos mejores caminos en función del tiempo. En el segundo paso (c) se seleccionan los retardos apropiados, considerando todas las posibles combinaciones de todos los canales. Se acaba eligiendo el mejor camino según los datos de distancias y correlación.

1.3.2. Localización de fuentes

Las técnicas de beamforming acústico son usadas con el propósito de localizar fuentes sonoras. Diferentes aplicaciones de localización de fuentes sonoras se describen en [9]. Hay que destacar que las condiciones ambientales pueden afectar considerablemente a la fiabilidad del beamforming. La

reflexión y difracción de las ondas sonoras dan lugar a las llamadas “fuentes fantasma” o *ghost sources* en inglés. Estas “fuentes fantasma” aparecen cuando el proceso de beamforming no está perfectamente adecuado para los fenómenos de propagaciones reales del entorno.

- Una aplicación interesante de beamforming es la relacionada con la identificación de fuentes en movimiento. En este caso, se debe considerar el efecto Doppler, por lo que se tendrá que compensar este fenómeno. El beamforming aplicado a objetos en movimiento se suele aplicar en el dominio del tiempo, ya que es más rápido para un número grande de micrófonos. Con las coordenadas del móvil estimadas y la frecuencia Doppler compensada, se procede con un beamforming “delay and sum”, visto en secciones anteriores.
- Un campo en el que el beamforming ha supuesto una herramienta de mejora importante es en los experimentos llamados “Túneles de viento” o *Wind tunnels*. Estos experimentos se realizan con el propósito de analizar el efecto del aire incidente en objetos. Se simula una situación real, enfocando una estructura como puede ser un avión, una aeronave o incluso edificaciones.

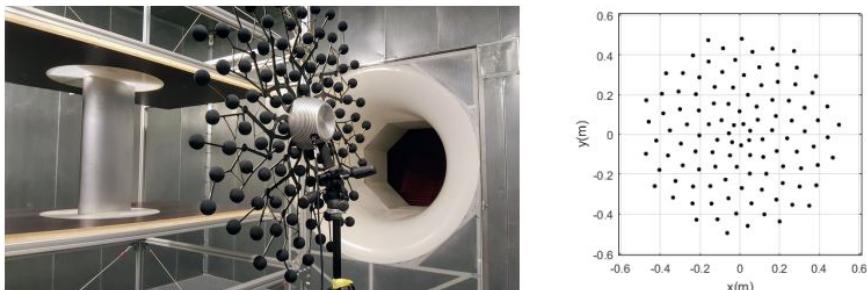


Figura 1.6: Configuración del array para un túnel de viento

Las ondas de sonido radiadas por las fuentes aero-acústicas ubicadas en la superficie del modelo, sufren refracción y scattering debido a entorno del experimento, por lo que deberá existir un algoritmo para compensar este fenómeno. La técnica de beamforming juega un papel importante al estimar las repercusiones acústicas que sufre el modelo y evitar posibles averías o incluso conseguir una mejoría en el comportamiento del modelo. En la *Figura 1.6* se muestra el experimento real (parte de la izquierda) y la disposición de los micrófonos (parte de la derecha).

- Por último, mencionar la aplicación de beamforming en interiores. Micrófonos empotrados son normalmente utilizados en entornos de in-

terios, así como cabinas de avión o interiores de automóviles. De esta manera, se consigue identificar la ubicación de las fuentes sonoras y actuar frente a estas.

1.3.3. Realce de Voz

Diversas reuniones y conversaciones son grabadas por diversos micrófonos distribuidos en el espacio, con el objetivo de conseguir una mejora notable en la calidad del audio. Este proceso es conocido como realce de voz o *speech enhancement* en inglés.

Cuando se utilizan diversos micrófonos distribuidos, por ejemplo, en una sala de reuniones, es un factor clave determinar qué micrófono del array es el que se encuentra más cercano al orador. Idealmente, se puede distribuir un micrófono para cada orador, si se conoce de antemano la posición de este, pero lo usual es que se desconozca este dato. Determinando el micrófono con mayor cercanía al orador, se puede llegar a realizar un reconocimiento del discurso con mayor precisión, ya que este micrófono suele poseer una mejor calidad de la señal y menos reverberación. La manera más simple de elegir el micrófono de interés en función del orador activo sería observando la mayor amplitud en las grabaciones de cada micrófono. Sin embargo, los micrófonos pueden no tener la misma sensibilidad e incluso tratarse de un array Ad-Hoc de smartphones los cuales pueden no tratar la señal de la misma forma. Además, en una reunión o conferencia suele existir bastante ruido, reverberaciones y voces de otros oradores, por lo que no es un camino fiable a la hora de determinar el micrófono de referencia.

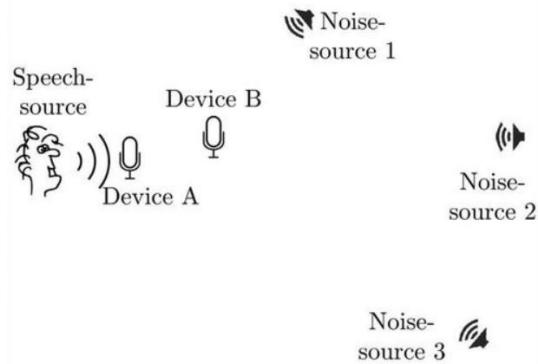


Figura 1.7: Esquema de realce de voz.

Suponiendo que el número de micrófonos es mayor al número de interlocutores, el objetivo es llegar a un método que seleccione el micrófono más cercano cuando uno de los interlocutores se encuentre hablando. En [10] se proponen algunas aproximaciones:

- Normalizar la potencia de cada micrófono en función de las potencias de todos los oradores y, a continuación, se selecciona el micrófono con una mayor potencia normalizada.
- Esta segunda aproximación selecciona el micrófono de referencia para cada orador de una manera independiente, de forma que la potencia total de los micrófonos seleccionados sea máxima.

En la *Figura 1.7* se identifica el orador (*Speech source*), los micrófonos componentes del array y distintas fuentes de ruido.

Capítulo 2

Planteamiento del problema

Establecidas unas bases teóricas y la perspectiva práctica de las agrupaciones de sensores y beamforming, se dispone a plantear el problema en el que se ha trabajado.

El objetivo del proyecto es conseguir una sincronización en el tiempo entre los elementos de un array Ad-Hoc de smartphones, para así lograr una correcta implementación de un beamformer acústico u otras aplicaciones. Los móviles se colocarán en una posición determinada, siendo cada uno de ellos un elemento del array lineal, manteniéndose conectados a un servidor externo. El servidor externo emitirá órdenes para que los dispositivos empiecen a grabar o, por el contrario, terminen la grabación. Las grabaciones obtenidas por los tres smartphones del array se corresponderán con las señales captadas por los elementos de una agrupación de micrófonos, las cuales se procesarán y llegarán como argumento, en este caso, al algoritmo de beamforming. Como última instancia, se comprobará el correcto funcionamiento del beamforming con las grabaciones obtenidas.

2.1. Distribución del proyecto

En esta sección, se explicará el procedimiento seguido para la correcta realización del proyecto, así como los requisitos de realización y los entornos en los que se ha trabajado.

2.1.1. Requisitos de realización

Se van a enumerar los requisitos para la realización del proyecto actual. Se diferenciará tres niveles: software, hardware o elementos físicos y conocimientos.

SOFTWARE

- Aplicación Android, instalada en cada uno de los smartphones, para una comunicación exitosa entre la agrupación de dispositivos y el servidor y para un correcto tratamiento de la grabación.
- Servidor externo, necesario para el control de los dispositivos a la hora de tomar mediciones.
- Algoritmo de sincronización, ya que es necesario mantener un sincronismo estricto entre las grabaciones para hacer beamforming.
- Algoritmo de beamforming, para el tratamiento de las señales acústicas según se desee.

HARDWARE Y ELEMENTOS FÍSICOS

- Ordenador. Máquina en la cual va a estar activo el servidor y la cual ejecutará los diferentes programas, así como el método de sincronización y el beamformer.
- 3 móviles Android. Al menos 3 dispositivos compatibles con la aplicación creada y con las opciones de grabación en funcionamiento.
- Altavoz monofónico, a poder ser con conexión bluetooth.
- Materiales para el montaje experimental, como cartón pluma, papel de dimensiones A2, ejes metálicos, cinta adhesiva.

CONOCIMIENTOS

- Programación. Conocimientos de programación en java y en python.
- Procesamiento de señales. Necesario para el método de beamforming y de sincronización.
- Telemática. Para el fundamento de la transmisión de datos entre los móviles y el servidor.

2.1.2. Seguimiento del proyecto

A continuación, se muestra una partición cronológica de las tareas realizadas:

1. Realización del servidor externo. Comprobando su funcionamiento mediante la conexión con otra máquina.
2. Desarrollo de la aplicación. Teniendo en cuenta los requerimientos del proyecto, comprobando el correcto funcionamiento mediante el conexiónado con el servidor.
3. Implementación del algoritmo de sincronización. Realizando diversas pruebas con los tres dispositivos y modificando el código para su mejora en cada prueba.
4. Algoritmo de beamforming. Programación del código correspondiente al beamformer y trazado del patrón de directividad.
5. Medición de datos experimentales. Realización de numerosas pruebas mediante el montaje experimental orientado a trazar el patrón de directividad experimental del beamformer.
6. Documentación y justificación de los resultados. Tras suficientes datos obtenidos, se extraen las conclusiones del experimento.

En la *Figura 2.1* se puede visualizar el diagrama de bloques correspondiente al seguimiento del proyecto. Se hace tres particiones: La parte de conexión y comunicación, la parte de tratamiento de la señal y la parte de recolecta y exposición de los resultados.

2.1.3. Entornos de trabajo

Como se ha comentado, el servidor se ha programado en lenguaje Java, específicamente en el entorno de NetBeans. La aplicación, como la gran mayoría de aplicaciones android, se ha desarrollado en el entorno de desarrollo AndroidStudio. El método de sincronización y el algoritmo de beamforming, se ha programado en lenguaje Python, en concreto se ha empleado el entorno de “Spyder”, proporcionado por la distribución Anaconda.

2.2. Justificación del proyecto

Se decidió el estudio del tema propuesto en el proyecto porque, en general, hace referencia a bastantes ámbitos dentro de las telecomunicaciones, como los conocimientos requeridos vistos en la sección anterior. Además, la investigación del tema elegido, ganó bastante atractivo al conocer todas las aplicaciones prácticas que pueden incorporar beamforming o sincronización acústica de señales. La perspectiva de futuro y la continua mejora en

este campo, como se verá posteriormente, ha jugado un papel importante también en la elección del proyecto.

La investigación realizada en este proyecto puede proporcionar un método válido de sincronismo entre dispositivos. Asimismo, se aporta un número de factores reales a tener en cuenta para la realización de este tipo de experimentos, tanto para el proceso de beamforming como el método de sincronización. Puede ser de ayuda para perfeccionar cualquier procedimiento relacionado con el que se expone.

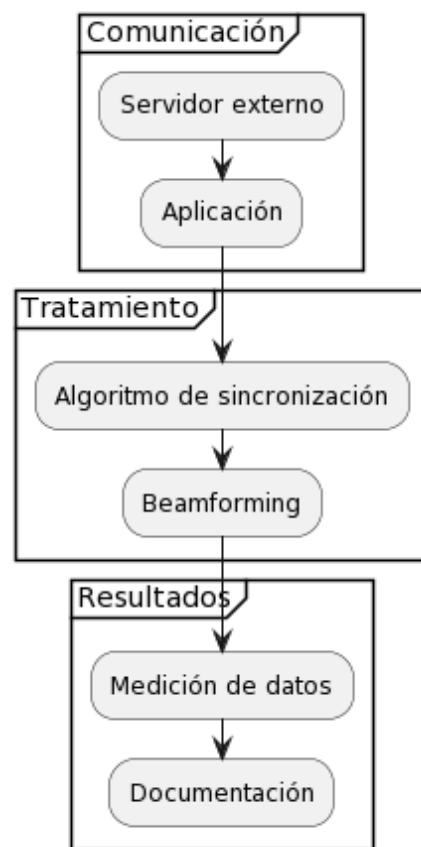


Figura 2.1: Diagrama de bloques del sistema

Capítulo 3

Desarrollo del servidor externo

Tal y como se ha comentado, para crear la comunicación con los móviles y manejar las distintas órdenes correspondientes al objetivo de grabar de forma coherente en cada dispositivo, es necesaria la incorporación de un servidor externo.

3.1. Propósito principal

Lo primero es aclarar la función que el servidor externo va a desempeñar. Ésta será dar órdenes a la aplicación, la cual se verá en el *capítulo 4*, de comienzo y finalización de la grabación. Además, será el encargado de, una vez recibidas las grabaciones procedentes de la aplicación, guardar los ficheros correspondientes. Por supuesto, será el encargado de habilitar un puerto disponible para cada dispositivo que se conecte.

3.1.1. Requisitos funcionales

A continuación se van a enumerar los parámetros funcionales a nivel de software que posee el servidor externo creado.

- Capacidad para manejar un número máximo de 10 clientes.
- La primera conexión de cada cliente se realiza en un mismo puerto común.
- Una vez realizada la primera conexión, el servidor maneja los puertos de conexión fija de los clientes, buscando puertos libres, de manera que tengan una asignación consecutiva (ej: nº puerto 5001, 5002, 5003...)

- Una vez conectados el número de clientes especificado, esperará la confirmación de la aplicación para enviar la orden de "comenzar a grabar"
- Se le podrá especificar la duración de la grabación en milisegundos.
- Una vez acabada la grabación por cada uno de los clientes, se guarda cada fichero de audio en formato de datos brutos (.raw) con la nomenclatura: "Device" + "nº cliente"
- El servidor muestra por pantalla las marcas de tiempo correspondientes al inicio y al final de la grabación de cada dispositivo o cliente.
- Posee una función la cual reproduce una señal de sincronización una vez hayan comenzado a grabar todos los dispositivos.

3.1.2. Requisitos no funcionales

En esta parte se va a exponer la calidad del software en sí mismo, con motivo de evaluar su funcionamiento.

- Es necesario especificar el número de conexiones o número de dispositivos que se van a conectar antes de iniciar el servidor.
- Una vez iniciado el servidor, espera indefinidamente (timeout) hasta que se realicen el número de conexiones especificadas.
- Los tiempos correspondientes a la comunicación entre los diversos dispositivos y el servidor son variables.
- Si a la hora de grabar se produce algún problema en uno de los dispositivos, los ficheros correspondientes a los demás dispositivos se guardarán correctamente, mientras que el fichero correspondiente al dispositivo erróneo, no se almacenará de forma correcta.
- Los tiempos de respuesta del servidor dentro de una zona de cobertura local rondan los 100 milisegundos.
- La comunicación del servidor con los distintos clientes no se produce de una forma simultánea, sino que genera y recibe información de los dispositivos de uno en uno.
- Las marcas temporales devueltas por el servidor son relativas, ya que dependen de la sincronización de los relojes de cada dispositivo en concreto.
- Si se desea un tiempo de grabación alto, se deberá aumentar el "buf-fer" correspondiente al almacenamiento de bytes de los archivos de grabación.

3.2. Construcción del servidor

Como ya se ha comentado anteriormente, para la implementación del servidor se ha utilizado el entorno de “NetBeans” debido a la facilidad que proporciona el lenguaje de programación “java” con los *sockets* para la comunicación cliente-servidor.

Se va a proporcionar una noción respecto al concepto de *socket* y la típica comunicación cliente-servidor. Acto seguido se verá en detalle cada funcionalidad implementada en el servidor .

3.2.1. Protocolo de comunicación y noción de *socket*

Para entender el manejo de los sockets, es necesario conocer la comunicación estándar cliente-servidor.

En una comunicación vía red, la información se desglosa en paquetes. La forma en la que están estructurados estos paquetes, la define el protocolo de transporte utilizado. En el caso que nos concierne, debido a que tenemos el propósito de establecer una comunicación orientada a conexión, se va a utilizar el protocolo TCP (Transfer Control Protocol), complementado con IP (Internet Protocol), formando TCP/IP. Este protocolo utiliza dos piezas claves de identificación: la dirección IP y un número de puerto.

Los términos que hacen referencia a los conceptos de *cliente* y de *servidor* son que el cliente debe iniciar la comunicación, mientras que el servidor espera pasivamente a este primer mensaje por parte del cliente; una vez recibe el servidor el mensaje del cliente, el servidor responde de forma coherente a la petición.

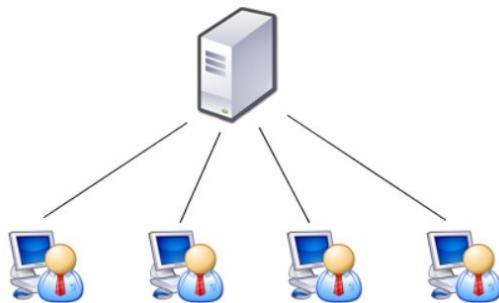


Figura 3.1: cliente-servidor

La distinción *cliente-servidor* es importante porque el cliente necesita conocer la dirección IP y el puerto del servidor, pero no al contrario.

En concepto de socket es abstracto. Se puede entender como un método

el cual permite la comunicación de aplicaciones en una misma red, pudiendo enviar y recibir datos a través del socket. El protocolo TCP/IP permite transmitir flujos de datos y, a través de un socket, una aplicación es capaz de alcanzar un puerto disponible de la red. En la *Figura 3.2.1* se esquematiza la posición de los sockets en el entorno de comunicación. Una información más extensa sobre sockets puede ser proporcionada por [16].

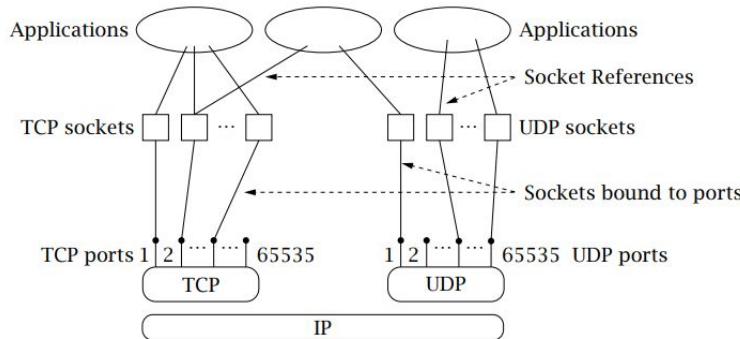


Figura 3.2: TCP sockets

El manejo de los sockets en Java lo podemos encontrar en su propia API [12].

Server Socket y Socket: Se deben inicializar tanto los sockets correspondientes al cliente como los correspondientes al servidor. El socket del servidor o “Server Socket” hace referencia al puerto al que debe conectar el cliente para mantener la comunicación con el servidor. El socket perteneciente al cliente o “Socket” es el que se le asigna al cliente por el servidor una vez el cliente haya enviado una request ó petición mediante el Server Socket. El socket del cliente es de utilidad al propio servidor para manejar al cliente correspondiente y para inicializar este último se necesita además del número de puerto, la dirección IP del servidor.

En java, el socket procedente al servidor se inicializa de la siguiente manera:

```
ServerSocket server = new ServerSocket(Port)
```

El socket correspondiente al cliente:

```
Socket client = new Socket(IP, Port)
```

InputStream y OutputStream: La comunicación se realizará mediante flujos de entrada y de salida o también llamados *InputStream* y *OutputStream*. En particular, para transmitir datos en forma de byte, más

en específico, se utilizan los flujos *DataOutputStream* y *DataInputStream*. Se corresponden al flujo de salida y al flujo de entrada respectivamente. Desde el punto de vista del servidor, el *DataInputStream* va del cliente al servidor y el *DataOutputStream* va del servidor al cliente.

La sintaxis en java para inicializar estos puentes es la siguiente:

```
DataStream output = new  
DataStream(client.getOutputStream())  
  
DataStream input = new  
DataStream(client.getInputStream())
```

En la salida (*output*) se escribirán bytes para ser enviados y en la entrada (*input*) se leerán los bytes entrantes.

Teniendo una idea general de lo que es un socket y cómo se utiliza, es trivial que la aplicación que se comunica con el servidor poseerá sockets relacionados con cada dispositivo o cliente en cuestión, para comunicarse correctamente con el servidor en el puerto correspondiente.

3.2.2. Implementación del servidor

Llegados a este punto, se procede a explicar la arquitectura del propio servidor creado para el proyecto.

Se espera manejar 10 clientes como máximo, por lo que el servidor podrá crear y dirigir 10 sockets de clientes distintos. En cualquier caso, se establecen dos conexiones por defecto. La primera conexión será común para todos los dispositivos, es decir, debido a esta primera conexión se ha creado un socket de servidor y otro socket de cliente adicionales, los cuales serán comunes para todos los clientes. Esto significa que cada dispositivo, por primera vez, se conectará a un mismo puerto. Una vez conectado el cliente a este puerto inicial, el servidor le proporcionará al cliente un nuevo puerto de conexión fija, en el cual se llevará a cabo el intercambio final de datos.

Para la selección de los números de puertos, hay que tener en cuenta cuáles están en uso y cuáles no, para que no haya ningún tipo de problema. Los puertos del 0 hasta el 1023 están reservados para el sistema operativo y los utilizan protocolos como FTP, DNS, SSH... Por lo que, para asegurar una correcta elección de número de puerto, se establece el puerto fijo de la primera conexión en el número de puerto 5000 y los puertos finales serán el 5001, 5002, 5003, ... 5010.

Además, para comprobar la disponibilidad del servidor y el número de puertos abiertos se ha hecho uso del comando *telnet* desde la terminal de Windows. Con este comando se comprueba fácilmente la conexión a un

puerto determinado conociendo la dirección IP de destino. La sintaxis del comando es la siguiente:

telnet[dominio ó dirección IP]/[puerto]

Si se llega a entablar la conexión, aparecerá una sección de pantalla vacía indicando que el puerto se encuentra disponible y si la conexión no se establece, se emitirá un mensaje de error lo cual indica que el puerto al que se referencia no está disponible o que, simplemente, el servidor no se encuentra en estado de escucha en ese puerto específico.

El servidor creado posee varias funciones, las cuales se enumeran y se explican a continuación:

- **Search4port:** Función encargada de devolver una lista de puertos disponibles para iniciar nuevas conexiones. Toma de argumentos de entrada un vector del tipo “booleano” donde se indica los puertos ocupados.
- **Recording:** Función encargada de enviar la orden de grabar a cada cliente. Una vez el servidor se encuentre en esta función, deberá recibir el “ACK” de todos los clientes para lanzar la orden de grabar. Toma de argumentos de entrada los sockets correspondientes al número de clientes.
- **Delay:** Esta función controla el tiempo de grabación. Simplemente manda a reposar el hilo principal durante un tiempo determinado, introducido en milisegundos.
- **stopRecording:** Como su propio nombre indica, es la función relacionada con el envío de la orden de parar la grabación. Similar a *Recording* pero con la diferencia de que se envía la orden opuesta y que el servidor no espera ningún ”ACK” por parte de los clientes para emitir la orden correspondiente.
- **getTimeStamp:** Función que lee la marca de tiempo devuelta por cada móvil. Como argumento de entrada toma el socket de cada cliente, ya que la marca temporal de cada móvil proviene de la aplicación.
- **Time:** Si fuese necesario devolver una marca temporal “absoluta”, tomada con el reloj del servidor, se puede llamar a esta función cuando se desee sin argumentos de entrada. Puede ser útil para comparar las demás marcas de tiempo “relativas” obtenidas para cada dispositivo.
- **timeView:** Se apoya en la función *Time()*. Únicamente muestra la marca de tiempo definida con el servidor por pantalla, para tener constancia del tiempo absoluto en que empieza cada grabación.

- **serverImpulse:** Esta función se encarga de emitir una señal acústica, justo cuando todos los dispositivos están grabando. De esta manera, con la señal inicial captada, se podrá llegar a un algoritmo de sincronización entre las grabaciones efectivo.
- **playChirp:** Para realizar un método de sincronización distinto al comentado en la función anterior, se puede llamar a esta función para emitir una orden de reproducir una señal “chirp” en cada uno de los móviles o clientes mientras se encuentran en estado activo de grabación, con el propósito obtener las señales grabadas sincronizadas.
- **saveFiles:** Por último, se llama a esta función. Su principal cometido es recibir los bytes procedentes de cada una de las grabaciones, almacenarlos en un buffer temporal, y guardarlos en la memoria de la máquina en la que se está ejecutando el servidor. Se debe ser coherente con el tiempo de la grabación y el tamaño del buffer temporal. Guarda las grabaciones como archivos de datos en bruto (.raw). Toma de argumentos de entrada el socket de cada cliente.

3.3. Comunicación

En esta sección se va a detallar la comunicación producida entre el servidor y cada uno de los dispositivos.

En primer lugar, se debe tener claro el modo de operación del servidor. Una vez iniciado el servidor, se presentan los siguientes flujos de comunicación:

1. El servidor se queda esperando hasta que recibe una petición de conexión de un cliente al puerto 5000.
2. Una vez aceptada esta primera conexión, el servidor busca un puerto libre, mediante la función *Search4port* y se lo devuelve al cliente para que realice una nueva conexión.
3. El cliente envía una nueva *request* al puerto indicado por el servidor. Al aceptar el servidor esta nueva conexión, el cliente finalmente se encuentra en el puerto adecuado para realizar la comunicación completa.
4. Ya conectados el número de clientes especificado, el dispositivo deberá enviar un mensaje de “READY”, para dar a entender al servidor que está listo para grabar.
5. El servidor emite la orden de grabar (“START”) y espera a que cada uno de los móviles le hagan llegar un “ACK” de confirmación por

empezar a grabar. Adicionalmente, los móviles transmiten al servidor la marca temporal correspondiente al inicio de la grabación y el servidor, por su parte, emite una marca temporal absoluta para cada confirmación recibida, la cual se utilizará más adelante para ayudar a sincronizar las señales.

6. Pasados los segundos de grabación establecidos, el servidor emite la orden de parar la grabación (“STOP”) y cada dispositivo devuelve de nuevo una marca temporal, relacionada con el final de la grabación.
7. Por último, cada smartphone envía al servidor el fichero de grabación resultante.

Con el propósito de comprobar este flujo de transporte enumerado, se hará uso de la herramienta *WireShark* ([14]). El principal objetivo de esta herramienta es analizar el tráfico de una red. Además resulta muy apropiada para estudiar cualquier tipo de comunicaciones y problemas de red.

Cabe señalar que, tras diversas pruebas, se han obtenido resultados más consistentes respecto a los tiempos de la red si se crea un punto de acceso inalámbrico en la máquina en la que corre el servidor. De esta forma, todos los smartphones que deseen interactuar con el servidor deberán conectarse a este punto de acceso creado. La comunicación mediante la Wi-Fi local presentaba muchos retardos inesperados, debido a que diversos dispositivos están conectados a la red, por lo que el tráfico de paquetes será mayor y, siendo esto lo más desfavorable, con más aleatoriedad. Además, disponiendo de un punto de acceso privado, el tráfico de la red se reducirá considerablemente a la hora de visualizar el flujo de transporte.

Se ha filtrado el tráfico de manera que solo se visualice el protocolo TCP en el puerto 5000 y en el puerto 5001, ya que se va a mostrar el flujo de un único dispositivo para simplificar el intercambio de mensajes. El tráfico resultante únicamente del conexiónado con el servidor se muestra en la *Figura 3.3*.

Se puede apreciar la dirección IP de la máquina donde corre el servidor (192.168.019) y la dirección IP del smartphone (192.168.137.184). Además, se observa enmarcado en rojo dos envíos de paquetes interesantes, ya que es el momento en el cual el servidor le ha enviado ya el siguiente puerto de conexión fija y realiza de nuevo el dispositivo una *request* al nuevo puerto (50001).

Para entender mejor todos los envíos de paquetes, se ejemplifica en el diagrama UML de la *Figura 3.4* el típico esquema de un flujo TCP. El protocolo TCP utiliza el llamado “*three-way handshake*” con el propósito de establecer una conexión fiable. Este método de conexión se basa en el envío y recepción de señales de sincronización y señales de acuse y recibo, o

tcp.port == 5000 tcp.port == 5001						
No.	Time	Source	Destination	Protocol	Length	Info
5	15.772499	192.168.137.184	192.168.0.19	TCP	74	45184 → 5000 [SYN] Seq=
6	15.772720	192.168.0.19	192.168.137.184	TCP	66	5000 → 45184 [SYN, ACK]
7	15.776004	192.168.137.184	192.168.0.19	TCP	54	45184 → 5000 [ACK] Seq=
8	15.784531	192.168.0.19	192.168.137.184	RSL	58	[Malformed Packet]
9	15.787373	192.168.137.184	192.168.0.19	TCP	54	45184 → 5000 [ACK] Seq=
10	15.787565	192.168.137.184	192.168.0.19	TCP	54	45184 → 5000 [FIN, ACK]
11	15.787657	192.168.0.19	192.168.137.184	TCP	54	5000 → 45184 [ACK] Seq=
12	15.789403	192.168.137.184	192.168.0.19	TCP	74	40310 → 5001 [SYN] Seq=
13	15.789606	192.168.0.19	192.168.137.184	TCP	66	5001 → 40310 [SYN, ACK]
14	15.795333	192.168.137.184	192.168.0.19	TCP	54	40310 → 5001 [ACK] Seq=
15	15.800724	192.168.0.19	192.168.137.184	TCP	54	5000 → 45184 [FIN, ACK]
16	15.848937	192.168.137.184	192.168.0.19	TCP	54	45184 → 5000 [ACK] Seq=

Figura 3.3: Tráfico “connect”

también llamadas *SYNC* y *ACK*. Concretamente, se distribuyen en su envío de la manera:

1. SYN
2. SYN/ACK
3. ACK

Una vez se haya completado el envío y recepción de esos tres mensajes, se podrá confirmar que existe una conexión fiable. El cliente mantiene un rol activo, enviando un segmento SYN, portador de su número de secuencia. El servidor, por su parte, juega un papel más pasivo.

Al recibir el servidor el segmento SYN, este último responde otro segmento SYN con su número de secuencia inicial y con un acuse y recibo (ACK). De esta manera, conociendo los números de secuencia de cada parte y con el ACK, ayuda a reconocer mensajes erróneos o perdidos. Cuando el cliente recibe el paquete SYN/ACK del servidor, sabe que se ha establecido exitosamente la conexión y devuelve un ACK al servidor para terminar.

En cada fase del proceso, es posible que se produzca una pérdida de paquetes o que no lleguen al destino correctamente, por lo que existe un tiempo límite o *timeout* con el cual se comprueba si se debe enviar de nuevo el segmento de sincronización. Todo este flujo de establecimiento de la conexión, lo podemos apreciar claramente en la *Figura 3.3*. El proceso de conexionado, en este caso, se realiza dos veces (una vez al puerto común 5000 y otra vez al puerto fijo 5001).

Una vez hecho el conexionado, el cliente hace saber al servidor que está listo para grabar. Cuando todos los dispositivos le hayan dado la confirmación para empezar a grabar, el servidor enviará la orden para empezar. Este flujo lo podemos ver en la *Figura 3.5*.

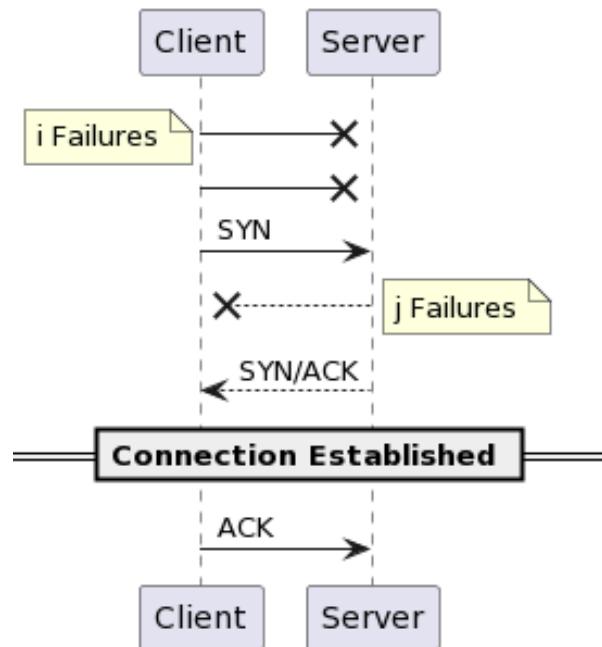


Figura 3.4: Conexión TCP

192.168.137.184	192.168.0.19	TCP	61 40310 → 5001 [PSH, ACK]
192.168.0.19	192.168.137.184	TCP	54 5001 → 40310 [ACK] Seq=
192.168.0.19	192.168.137.184	TCP	61 5001 → 40310 [PSH, ACK]

Figura 3.5: Tráfico "record"

Si observamos el contenido del primer paquete y del último paquete del flujo presentado, se pueden apreciar los correspondientes mensajes de “READY” por parte del dispositivo y de “START” por parte del servidor en la *Figura 3.6*. Los bytes anteriores a las palabras clave comentadas son correspondientes a la cabecera del paquete.

vp.....B...<..E.. ./@. @..... ...v....Yr/P. -W.....READY	B...<vp.....E.. ./T@...X..... ...vYr /.....P.START
---	---

Figura 3.6: Palabras clave comienzo

Una vez emitida la orden de grabar por el servidor, cada dispositivo envía un “ACK” adicional y empieza la grabación. En el período de tiempo de grabación no se intercambian paquetes, ya que el servidor se encuentra en reposo y los móviles se encuentran grabando.

Cuando la grabación finaliza, llega el momento de intercambiar el archivo de datos brutos almacenado. Este proceso se puede apreciar en la *Figura 3.7*.

192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=131462 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=132922 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=134382 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=135842 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=137302 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=138762 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=140222 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=141682 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=143142 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=144602 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=146062 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=147522 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=148982 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=150442 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=151902 Ack=14 Win=87808 Len=1460
192.168.137.184	192.168.0.19	TCP	1514 40310 → 5001 [ACK] Seq=153362 Ack=14 Win=87808 Len=1460
192.168.0.19	192.168.137.184	TCP	54 5001 → 40310 [ACK] Seq=132922 Win=262656 Len=0
192.168.0.19	192.168.137.184	TCP	54 5001 → 40310 [ACK] Seq=14 Ack=134382 Win=262656 Len=0
192.168.0.19	192.168.137.184	TCP	54 5001 → 40310 [ACK] Seq=14 Ack=135842 Win=262656 Len=0

Figura 3.7: Transferencia archivo

En la que se puede observar que el envío se divide en tramas de 1460 bytes y que llegados una serie de paquetes, el servidor responde con un ACK referenciando al número de secuencia del paquete el cual se está confirmando su correcta llegada.

Con los datos proporcionados por el tráfico de red, se traza un diagrama UML en la *Figura 3.8* para una mejor visualización de la comunicación entre el servidor y un cliente.

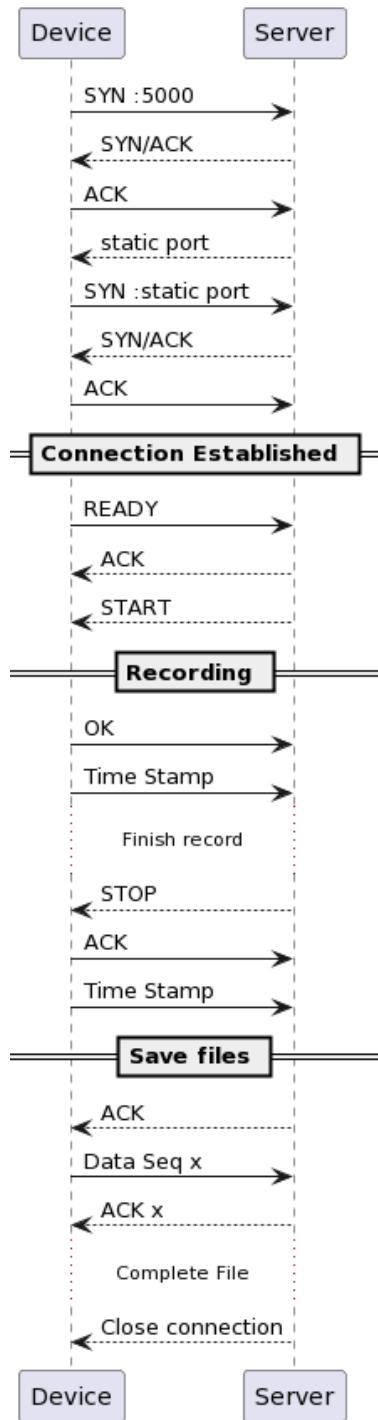


Figura 3.8: Tráfico completo.

Capítulo 4

Desarrollo de la aplicación

Tal y como se ha comentado anteriormente, para la comunicación entre el servidor y los respectivos smartphones y para que cada uno ejecute la grabación correctamente, es necesario la introducción de una aplicación Android en cada uno de los dispositivos. De esta manera se consigue la interconexión que se desea.

4.1. Introducción a los fundamentos de la aplicación

La aplicación ha sido desarrollada en *Android Studio*. Se le ha especificado el nombre de *BeamRec*.

El propósito de la aplicación es cumplir los requisitos necesarios, esto es la conexión con el servidor, la correcta grabación y la debida transmisión del fichero de grabación. La interfaz de la aplicación está pensada para que sea lo más intuitiva posible, para conseguir únicamente la meta del proyecto que se está tratando.

Nada más abrir la aplicación, la interfaz resultante se muestra en la *Figura 4.1a*. Como se puede apreciar, únicamente proporciona la opción al usuario de conectar el dispositivo con el servidor externo. Una vez se pulse el botón de “CONNECT”, el cliente enviará una petición de conexión al servidor especificado con una dirección IP fija.

Al pulsar el botón de “CONNECT” y habiéndose conectado el dispositivo exitosamente al servidor, la siguiente interfaz o *activity* que el usuario visualiza se observa en la *Figura 4.1b*.

En este caso, la aplicación únicamente proporciona la opción de grabar. Este botón se deberá pulsar una vez todos los móviles que se desean conectar

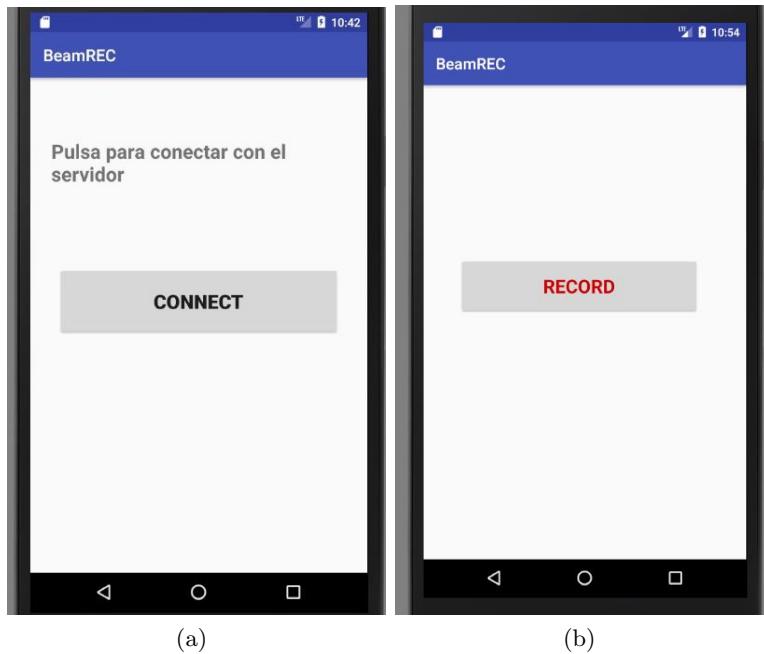


Figura 4.1: Interfaces de la aplicación

al servidor hayan conseguido una conexión exitosa. Cuando se desee comenzar a grabar, una vez pulsado el botón, el cliente envía el correspondiente mensaje de que está listo para grabar hacia el servidor. Este último, cuando recibe el mensaje del estado de “listo” por todos los clientes conectados, emite la orden de grabar. Al recibir la aplicación la orden de “grabar” por el servidor, se activa la grabadora en el dispositivo y se envía un ACK al servidor por parte de cada dispositivo conectado. Si se desea repetir la grabación, se deberá realizar desde el principio el proceso de conexión con el servidor externo. Será necesario que cada uno de los móviles estén conectados a la misma red a la que está conectado el servidor.

4.2. Funcionalidades

En esta parte se va a explicar las distintas funciones internas de la propia aplicación.

En total se han creado 3 clases java en Android Studio:

- **MainActivity:** Es la clase principal. En ella se encuentra el diseño de la primera interfaz mostrada en la *Figura 4.1a*. Al pulsar el usuario el botón, se hará una llamada a la siguiente clase con su nueva interfaz.

- **RecordClient:** Contiene el diseño de la segunda interfaz de la aplicación (*Figura 4.1b*). Esta clase es la encargada de mantener la conexión con el servidor dada una dirección IP específica y el puerto común, el cual se ha determinado como el puerto número 5000 en la sección del servidor. Realiza todo el proceso de doble conexión al servidor en segundo plano y, una vez conectado el dispositivo al servidor, se cargará la interfaz correspondiente a la acción de grabar.

Al pulsar el usuario el botón de “RECORD”, se mandará el mensaje de “READY” al servidor y se esperará a la orden de empezar la grabación emitida por el servidor. La comunicación con el servidor desde la aplicación se realiza de manera similar que en el servidor: con el uso de *Sockets* y flujos de datos *DataInputStream* y *DataOutputStream*. Cuando la aplicación recibe la orden de grabar por el servidor, se hace la llamada a la clase *Recorder* y empieza la grabación en un nuevo hilo. En el hilo principal se procede al envío del “ACK” y de la marca de tiempo correspondiente al comienzo de la grabación por parte del dispositivo al servidor y se mantiene en espera hasta recibir la orden correspondiente a parar la grabación.

Una vez recibida la orden de “STOP” procedente del servidor para parar la grabación, se hace la llamada correspondiente a la clase *Recorder* para dejar de grabar y se envía la marca de tiempo de finalización. Acto seguido, se procede a la manipulación del archivo. Mediante las clases *BufferedInputStream* y *BufferedOutputStream* se almacenará la grabación en un buffer momentáneo que será enviado directamente al servidor.

Cabe destacar que, en esta clase también se definen los permisos necesarios para la correcta ejecución de la aplicación en el dispositivo Android. Estos son los permisos de utilización del micrófono y de la grabadora del smartphone.

- **Recorder:** Esta clase, como se ha mencionado en *RecordClient*, es la responsable de la propia acción de grabar. Se ejecuta en segundo plano y es capaz de ejecutar una grabación con una frecuencia de muestreo de 44100 Hz, la cual es la que soporta la mayoría de dispositivos actualmente, en un canal monofónico y una codificación del tipo PCM (Modulación de Pulses Codificados) de 16 bits para que, de esta manera, no exista compresión alguna en el audio.

Además, se implementa la desactivación del control de ganancia automático ó *AGC* y la cancelación de eco ó *AEC*, lo cual podrían ser factores perjudiciales para el propósito del proyecto, ya que se modificaría la señal automáticamente interfiriendo procesos posteriores como es el de sincronización o beamforming.

La clase *Recorder* proporciona dos métodos, uno de comienzo de la grabación, en el cual se irán almacenando paquetes de bytes en un buffer temporal, y el método relacionado con terminar la grabación.

Capítulo 5

Sincronización

Para el correcto funcionamiento del algoritmo de beamforming, es necesario que las señales de entrada en el beamformer únicamente posean el retardo debido al tiempo de propagación desde la fuente hasta las posiciones de los móviles. Es decir, se debe eliminar cualquier retardo producido por la red de servidor-app creada así como cualquier desplazamiento en el reloj interno de cada móvil. La comunicación entre el servidor y la aplicación genera retardo de red. Este retardo será mayor o menor dependiendo del tráfico o utilización de la red. Mediante las marcas de tiempo proporcionadas por el servidor, se midió el retardo de la red Wi-Fi local, obteniendo unos retardos desde 10 a 900 ms aproximadamente. Sin embargo, debido a la aleatoriedad de los retardos en la Wi-Fi local por los distintos paquetes en cola, se ha optado por realizar el experimento en un punto de acceso inalámbrico, ganando de esta manera un retardo más constante, de unos 100 ms aproximadamente.

Las órdenes que el servidor transmite a cada dispositivo las transmite de forma secuencial, por lo que la orden determinada llegará antes al primer cliente que al último. Anotar que las marcas de tiempo producidas por los móviles no se encuentran sobre un eje temporal común. Esto es debido a que existe un cierto retardo entre los relojes de los propios smartphones el cual habría que compensar.

Los retardos mencionados se ilustran en la *Figura 5.1*, donde se ha ejemplificado el retardo correspondiente a los relojes y el retardo correspondiente a la red, para dos dispositivos. Estos retardos, se deben eliminar, manteniendo de esta manera un eje temporal absoluto para el cual el instante de comienzo (T_{START}) de la grabación coincide para cada uno de los dispositivos conectados.

Conociendo el retardo que hay que eliminar, se logrará una serie de señales con un eje temporal absoluto y únicamente con el retardo físico

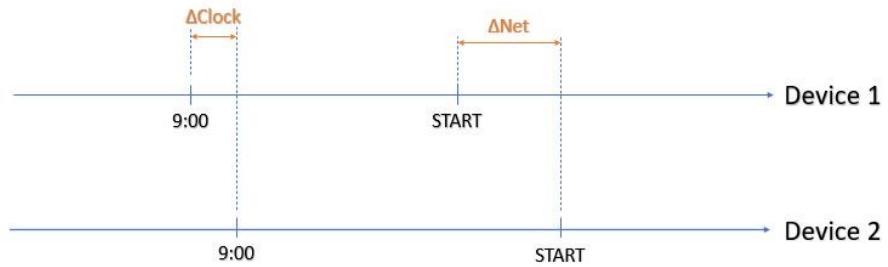


Figura 5.1: Retardos de red y reloj

producido por la propia transmisión de las señales desde el altavoz hasta los micrófonos de los dispositivos. El propósito de este capítulo es hallar y compensar, de la forma más exacta posible, estos retardos. Se propondrá dos alternativas de sincronización distintas.

5.1. Recursos empleados

Para el desarrollo de los métodos de sincronización, se va a hacer uso de información proporcionada por la correlación cruzada y el tratamiento de señales adecuadas para un buen sincronismo entre dispositivos.

5.1.1. Correlación cruzada

En procesado de señales, la correlación cruzada entre dos señales es de gran utilidad, ya que es una herramienta matemática que proporciona el grado de similitud entre las señales en función de un retardo considerado entre las mismas. Su cálculo se basa en la convolución de una de las señales con la otra invertida temporalmente. Y se expresa siguiendo la ecuación (5.1).

$$R_{x,y}[k] = \sum_{n=-\infty}^{\infty} x[n]y[n-k] \quad -\infty < k < \infty \quad (5.1)$$

La correlación cruzada de una señal con ella misma se denomina autocorrelación. Un ejemplo de autocorrelación de una señal contaminada con ruido de media la unidad se muestra en la *Figura 5.2*.

La correlación cruzada será máxima en el instante de llegada de la señal. Como en la *Figura 5.2* se está computando la misma señal, el máximo de la correlación se sitúa en 0 muestras, ya que no existe retardo alguno.

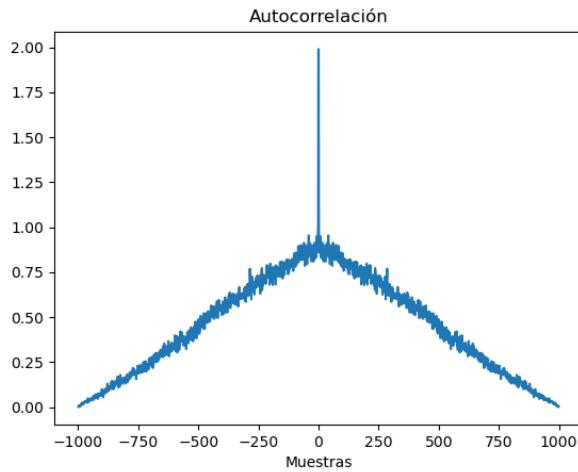


Figura 5.2: Autocorrelación

En el caso de dos señales idénticas, una retardada respecto a la otra, el valor proporcionado por la correlación cruzada señalará el retardo que se busca para la sincronización de las señales. La *Figura 5.3* muestra una simulación en la que se ha hecho la correlación cruzada de dos señales idénticas, una retardada 1000 muestras respecto a la otra.

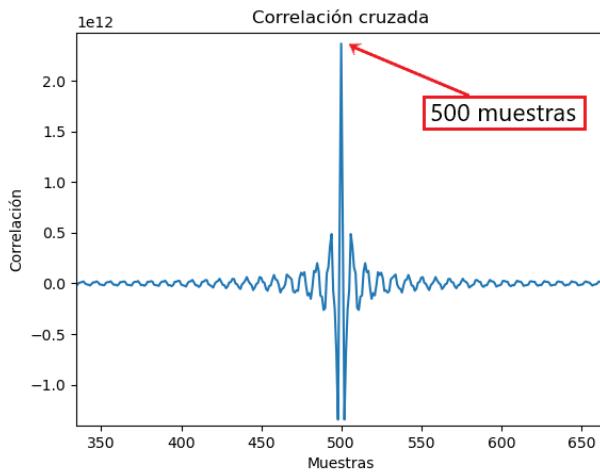


Figura 5.3: Correlación cruzada

Se ha representado la parte simétrica de la correlación, es decir, desde $k = -N$ hasta $k = N$, siendo N la longitud en muestras de la correlación. Se aprecia un claro máximo de la correlación cruzada, el cual denota el retardo entre las dos señales. Al representar en $-N < k < N$, el valor de la

correlación cruzada marca 500 muestras (sumando las muestras simétricas se obtiene un retardo de 1000 muestras).

5.1.2. Señales de sincronización

Se necesita conocer el desfase entre señales de una forma precisa. Para ello, se proponen varias señales ejemplo, con el fin de obtener un máximo de correlación coherente. Estas señales serán generadas por el servidor externo, reproducidas por un altavoz conectado al servidor y captadas por los dispositivos cliente.

La idea es introducir en los audios captados por los dispositivos, una primera parte formada por señales de sincronización, con el fin únicamente de hallar el desfase entre señales y, posteriormente, realizar el procesado correspondiente a beamforming. Se proponen diferentes señales ejemplo para implementar la sincronización comentada.

Impulso

Dentro de las señales discretas más utilizadas se encuentra el impulso unitario. El cual es utilizado para evaluar todo tipo de canales mediante la respuesta impulsiva. Se define un conjunto de muestras nulas excepto en 0 que toma un valor máximo de 1.

$$\delta(n) = \begin{cases} 1 & \text{si } n = 0 \\ 0 & \text{resto} \end{cases}$$

Para la transición que debe hacer la señal desde que el servidor manda reproducir la señal y esta se reproduce por el altavoz, el impulso unitario es demasiado ideal, por lo que se ha adaptado a un pulso desplazado a la derecha, con un total de 10 muestras de ancho, ya que si se hace más ideal el impulso no es reproducido correctamente.

La señal creada se muestra en la *Figura 5.4*.

Chirp

Un “chirp” es un tipo de señal con una variación determinada de la frecuencia instantánea. Un chirp hace un barrido entre dos frecuencias asignadas, lo cual facilita la aparición de un máximo claro de la correlación cruzada en el retardo con el que se consigue alinear los dos chirps.

Se ha creado un chirp el cual hace un barrido en frecuencia entre 5000 y 16000 Hz y una duración total de 0.1 segundos, la cual ha sido la duración mínima para su correcta reproducción y función requerida. Un “zoom” de

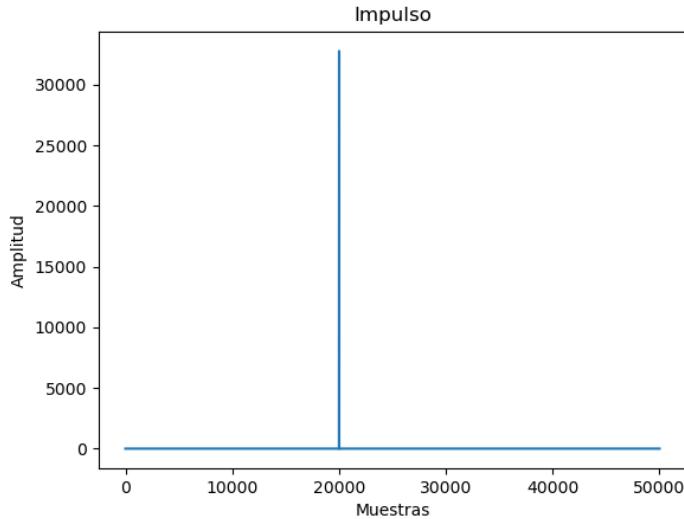


Figura 5.4: Señal impulsiva

la señal comentada se presenta en la *Figura 5.5*, donde se puede observar el cambio en la frecuencia instantánea.

Además de ser utilizado para sincronizar señales y obtener un valor de correlación óptimo, el chirp es también utilizado en comunicaciones ópticas. El método que se menciona es conocido como *prechirping* y consiste en una modificación de las características del pulso antes de ser introducido en una fibra óptica. En la *Figura 5.6* se aprecia un pulso gaussiano sin modificar y un pulso gaussiano con chirp.

Esta compensación previa puede hacer que el pulso se comprima y alcanzar longitudes de enlace de la fibra mayores e incluso transmisiones con mejor detección en el receptor, como justifica [19]. A lo que se desea llegar con la analogía del uso del chirp en comunicaciones ópticas, es la importancia en la variación de la frecuencia instantánea de la señal. Esta variación proporciona información muy útil a la hora de determinar el instante de llegada de una señal.

Tren de impulsos

Como última señal, se propone un tren de impulsos. Consta de la concatenación de varios impulsos seguidos. De este modo, se consigue que, al realizar la operación de correlación cruzada, tengan que coincidir cada impulso con su correspondiente mientras se realiza el desplazamiento de la señal invertida, tal y como define la *ecuación (5.1)*. Aparece, de esta forma, un máximo más claro de la correlación en el instante en que todos los

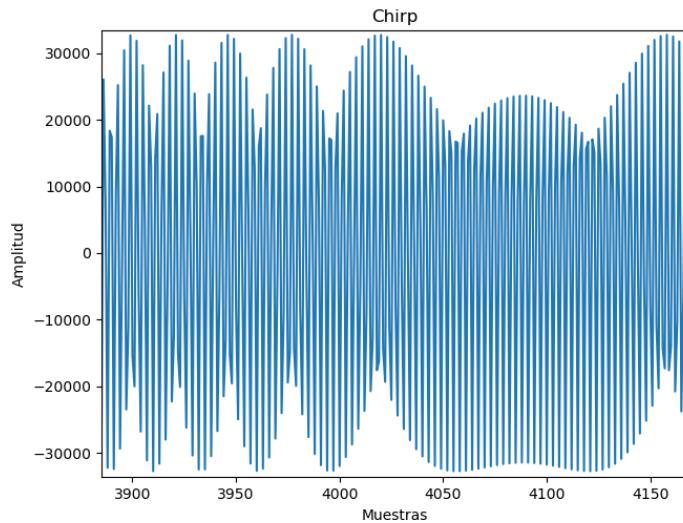


Figura 5.5: Chirp creado

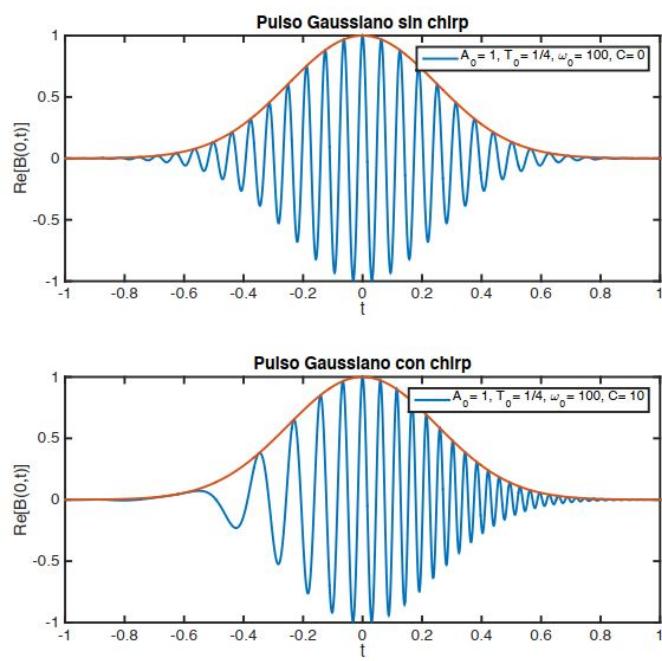


Figura 5.6: Prechirping

impulsos coincidan.

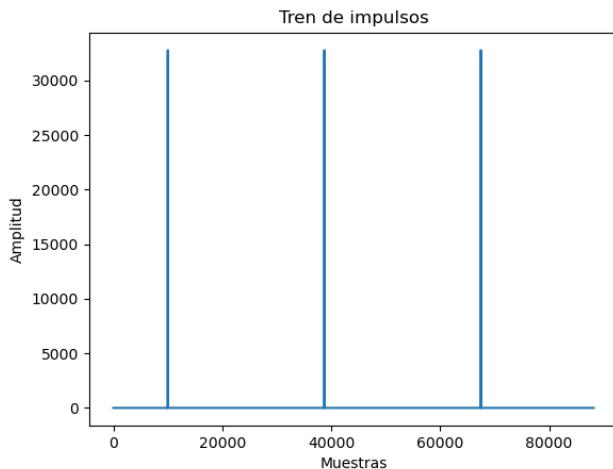


Figura 5.7: Tren de impulsos

Se ha creado un tren de un total de tres impulsos, siendo estos equidistantes y teniendo en cuenta un margen de muestras inicial y final para la correcta reproducción por el altavoz y captación por los móviles. La señal comentada se representa en la *Figura 5.7*.

5.2. Sincronización mediante el servidor

Habiendo analizado las tres variantes de señal de sincronización propuestas, se procede a explicar dos métodos de sincronización distintos.

El primer método de sincronización propuesto, es realizado mediante la ayuda externa del servidor. Dado que las marcas temporales de los móviles no son muy fiables, ya que arrastran el retardo debido a los relojes y a la red, se propuso la emisión de una señal de sincronización, por parte del servidor, cuando todos y cada uno de los móviles estén en estado activo de grabación.

Cuando el servidor recibe la confirmación de cada dispositivo de haber comenzado a grabar, procede a la reproducción, por un altavoz monofónico, de la señal de sincronización. La idea es que esta señal, sea grabada por todos los móviles y, acto seguido, sin detener la grabación, captar una segunda señal o señal principal. Los móviles deberán colocarse de forma equidistante al altavoz, determinándose así la posición inicial y, tras la captación de la señal de sincronización, la posición de los móviles podrá ser modificada según la geometría del array deseada, para la captación de la señal principal. Esta última señal, es la que se debe obtener sin retardo alguno de red y la que se

debe transmitir para realizar el proceso de beamforming.

De forma ideal, se deberán sincronizar todas las N señales de forma perfecta con un retardo nulo. Esto se pretende conseguir realizando un análisis mediante correlación cruzada de la señal acústica de sincronización captada por primera vez por uno de los dispositivos y esta misma señal grabada por los demás dispositivos. De esta manera, se supone el tiempo inicial 0 en el instante en que uno de los dispositivos capta primero la señal de sincronización. Como se ha comentado en la *sección 5.1.1*, el mayor pico resultante de la correlación cruzada corresponderá, en muestras, al retardo de la señal grabada respecto a la señal primera.

5.2.1. Análisis

Antes de hablar de retardos y modificaciones en las grabaciones, se debe elegir la señal de sincronización.

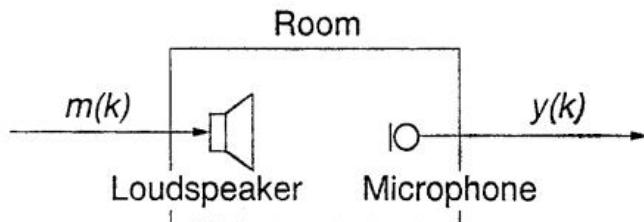


Figura 5.8: Bloque altavoz-micrófono

Se realizaron diversas pruebas para cada una de las señales propuestas en la sección anterior. Hay que tener en cuenta que, al usar un altavoz externo para emitir la señal de sincronización, como en la *Figura 5.8*, se pueden producir efectos no deseados, como modificaciones no controladas de la señal a la salida del altavoz. Se va a analizar la distorsión no lineal que hace que aparezcan alteraciones, no existentes en la señal de entrada, en la señal de salida y que pueden producir errores en las medidas. Haciendo referencia a los estudios de [22], el error causado por distorsión no lineal aparece como pulsos aleatoriamente distribuidos pero espaciados una distancia similar a la del período de muestreo (*Figura 5.9.a*). Este efecto no lineal se puede disminuir bajando el nivel de sonido del altavoz (*Figura 5.9.b*), pero si se disminuye demasiado, el ruido de fondo llega a tener un impacto considerable en la señal de interés y se incrementa el error aleatorio (*Figura 5.9.c*).

Realizando pruebas de sincronización con la señal del tipo “chirp”, no se llegaba a obtener una buena correlación, esto es, un máximo de correlación preciso y distinguible. Esta poca precisión desembocó en un retardo final inadmisible. Este resultado no tan bueno con la señal de sincronización

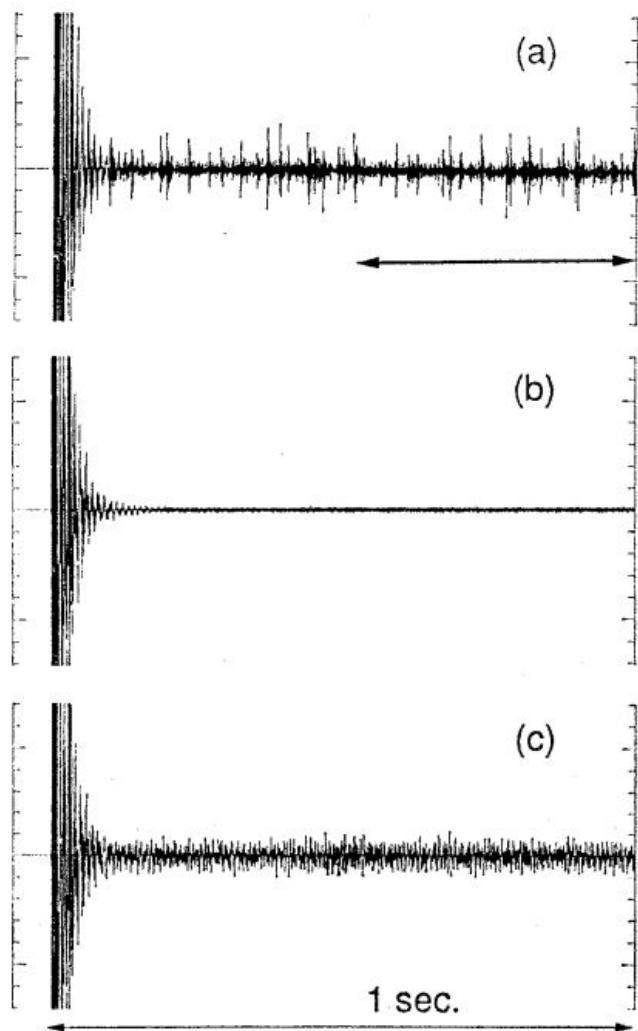


Figura 5.9: Efecto de la no-linealidad

chirp puede ser debido a los efectos no lineales producidos por el altavoz, ya que los chirps son muy utilizados para procesos de sincronización por los cambios en la frecuencia instantánea de la señal, los cuales son clave para correlar muestras. A pesar de esto, si no se capta bien estas modificaciones en frecuencia en el instante preciso, la correlación cruzada puede resultar errónea.

Habiendo tomado la decisión de no utilizar el chirp, se probó con la señal impulso. Se obtuvieron mejores resultados.

Por último, se experimentó la sincronización con el tren de impulsos, obteniendo la máxima precisión en el máximo de la correlación de las tres señales, por lo que se ha optado en este método por el uso del **tren de impulsos** como señal de sincronización acústica.

5.2.2. Procedimiento

Con el propósito de tomar de referencia el primer tiempo de llegada de la señal de sincronización, se debe determinar cuál es el dispositivo que capta primero esta señal. Con este dato, se fijará el instante cero en la grabación de este dispositivo y se buscará los retardos respecto a este instante.

Para determinar la primera captación de la señal de sincronización, se emplea la correlación cruzada del tren de impulsos creado y del tren de impulsos captado en la grabación de cada uno de los dispositivos. Una vez conocida la grabación de referencia, se realiza de nuevo un análisis de correlación cruzada, entre el tren de impulsos de la grabación de referencia y el tren de impulsos de cada uno de los dispositivos. El máximo valor de las N correlaciones cruzadas coincide con el retardo correspondiente en función del instante de referencia.

En la *Figura 5.10a* se muestra la autocorrelación de la señal tren de impulsos de la grabación de referencia tomada. Trivialmente, en este caso el retardo es cero exacto. Se pueden ver 4 máximos locales y un máximo absoluto en la representación de la correlación. Los máximos locales se corresponden a la superposición de los impulsos en el proceso del desplazamiento de la señal en la convolución con la señal invertida. Cuando se superponen todos y cada uno de los impulsos, las dos señales están perfectamente alineadas y se produce el máximo absoluto, el cual muestra el retardo real de una señal respecto a la otra.

La correlación cruzada entre el tren de impulsos del dispositivo tomado como referencia temporal y el tren de impulsos grabado por otro dispositivo diferente se presenta en la *Figura 5.10b*. Se observa que el máximo absoluto posee una cierta desviación respecto al 0 en el eje de abcisas. El valor de este máximo se corresponde con el retardo de la grabación de este smartphone.

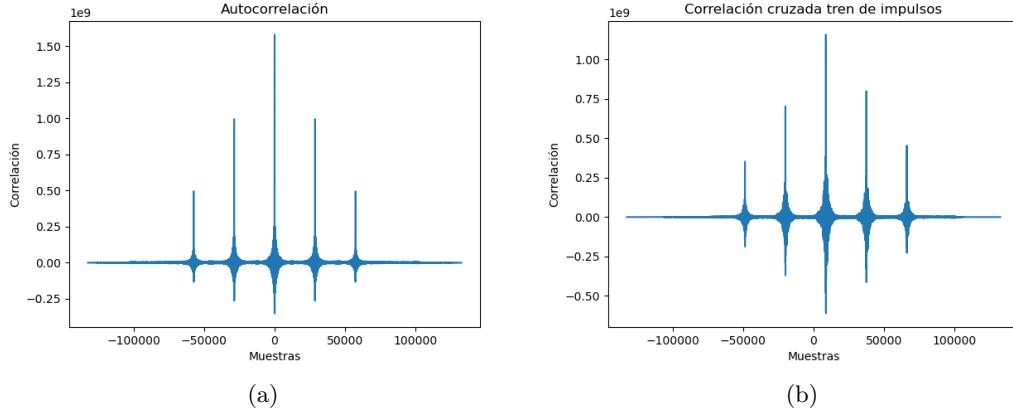


Figura 5.10: Correlaciones tren de impulsos

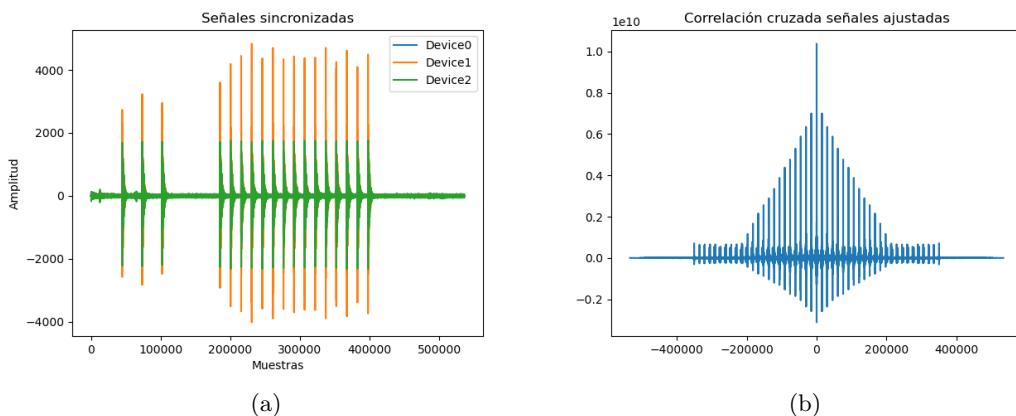


Figura 5.11: Señales sincronizadas

Conocido el retardo de cada grabación, se procede a eliminar las muestras correspondientes, acortando cada señal desde el principio. Debido a que la orden de parar la grabación no llega en el mismo instante para todos los móviles, las señales se deberán acortar también por el final, determinando el tamaño para todas las grabaciones como la longitud mínima de todas las señales una vez acortadas las muestras correspondientes al retardo inicial. Mediante este proceso de eliminación de muestras iniciales y finales, se obtiene una matriz con todas las grabaciones sincronizadas con un mismo número de muestras.

En la *Figura 5.11a* las señales perfectamente sincronizadas. Los tres primeros pulsos se corresponden con el tren de impulsos de sincronización y los demás pulsos se corresponden con la señal de interés, la que se necesita sin

retardo alguno. En el caso de la simulación, se ha realizado la sincronización con 3 dispositivos. Su comprobación se puede consultar en la *Figura 5.11b* donde se muestra una correlación cruzada entre la grabación de un dispositivo y la grabación de otro dispositivo distinto. Debe notarse que, la señal principal se corresponde con un tren de impulsos de 15 impulsos en total, por eso la correlación cruzada de la *Figura 5.11b* posee picos uniformemente separados.

5.3. Sincronización mediante “auto-chirps”

5.3.1. Fundamento

Durante el desarrollo del presente trabajo, se propuso adicionalmente una alternativa innovadora consistente en la adaptación de la sincronización espacial propuesta por Hennecke et al. [23] a una sincronización meramente temporal como la que aquí nos ocupa. En dicho artículo se describe un método de autolocalización de los smartphones mediante la reproducción de una señal “chirp” por cada uno de estos dispositivos. Teniendo en cuenta los tiempos de llegada de cada dispositivo, se realiza una optimización de una función criterio obteniendo finalmente una estimación de los tiempos de comienzo y de las posiciones de cada uno de ellos.

El método que se propone aquí es análogo al estudiado en [23], con la diferencia de que en lugar de calcular las posiciones relativas de cada móvil, se estimará el tiempo de vuelo de la señal “chirp” en cada caso. De esta manera se consigue una sincronización temporal, en vez de una sincronización espacial como se propone en [23].

En primer lugar, se deben adaptar las ecuaciones expuestas en [23] para los tiempos de propagación, estableciendo la siguiente definición de tiempo de vuelo,

$$t_{ki}^{flight} = \frac{\|s_k - m_i\|}{c} \quad (5.2)$$

correspondiéndose el numerador a la distancia euclídea entre el altavoz originario del chirp (dispositivo k) y el receptor (dispositivo i) y siendo c la velocidad de propagación del sonido.

La función que se debe minimizar en este caso es la siguiente:

$$(\hat{T}_{flight}, \hat{T}_{START}) = argmin \sum_{k=1}^N \sum_{(i,j)} ((\hat{\tau}_{ij}^k - \tau_{ij}^k) / \sigma_{ij}^k)^2 \quad (5.3)$$

Se basa en la estimación de la diferencia de los tiempos de llegada de cada dispositivo (TDOA), figurada como $\hat{\tau}_{ij}^k$ y el cálculo de su error cuadrático medio. La diferencia de los tiempos de llegada figura en la *ecuación (5.3)* como τ_{ij}^k . La desviación típica de las medidas se representa como σ_{ij}^k , siendo esta constante. La variable \hat{T}_{flight} se corresponde con los tiempos de vuelo estimados y \hat{T}_{START} con los tiempos de comienzo estimados para cada dispositivo, mencionados en el diagrama de la *Figura 5.1*. Una vez adaptada la función de "coste", los gradientes para el tiempo de propagación y para el tiempo de comienzo deberán ser los adecuados.

5.3.2. Procedimiento y conclusiones

A diferencia de la sincronización mediante el servidor, las señales de sincronización serán reproducidas por los propios móviles coordinadamente. Debido a esto, es de suma importancia la obtención de un eje temporal absoluto. Para ello, se extrae la marca temporal del servidor, la cual se considera absoluta, una vez empieza a grabar cada dispositivo. Modificaremos la longitud de cada una de las señales en función de las marcas temporales devueltas por el servidor, obteniendo finalmente un eje temporal absoluto.

El chirp creado en este caso tiene una duración de 0.1 segundos y realiza un barrido en frecuencia desde 5000 hasta 16000 Hz.

Cuando empieza la grabación para todos los dispositivos, se reproduce un chirp por el altavoz de cada dispositivo de forma ordenada. Este chirp es captado por todos los móviles incluido el que ha emitido la señal. Una vez concluida la reproducción de los N chirps, se grabará la señal objeto de sincronización.

En la parte superior de la *Figura 5.12* se puede ver una porción de la señal grabada por uno de los dispositivos en la cual se capta los chirps de cada uno de los dispositivos. La parte inferior de la *Figura 5.12* se corresponde con la correlación cruzada entre la señal chirp originalmente creada y la grabación de los tres chirps. Los 3 máximos obtenidos se corresponden con los tiempos de llegada (t_i^k) de cada señal, obteniendo la diferencia de los tiempos de llegada tal y como se muestra en la *ecuación (5.4)*.

$$\tau_{ij}^k = t_i^k - t_j^k \quad (5.4)$$

Sustituyendo posteriormente τ_{ij}^k en la *ecuación (5.3)*. Con el valor de la diferencia de los tiempos de llegada, se pueden estimar los tiempos de vuelo t_{ij}^{flight} y los instantes de comienzo t_i^{START} . El tiempo de llegada t_i^k se define como:

$$t_i^k = t_s^k + t_{ki}^{flight} - t_i^{START} \quad (5.5)$$

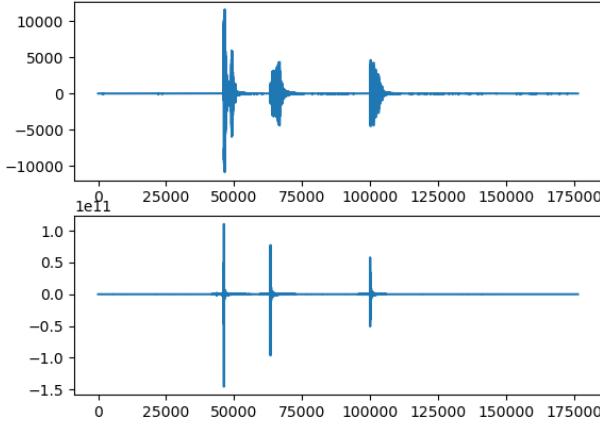


Figura 5.12: Chirps y TOA

Siendo t_s^k el instante de emisión de la fuente, t_{ki}^{flight} el tiempo de vuelo desde la fuente k al micro i y t_i^{START} el instante de comienzo de la grabación en el micro i .

La diferencia de tiempos de llegada solo depende de los tiempos de vuelo y de los tiempos de comienzo, tal que:

$$\tau_{ij}^k = t_{ki}^{flight} - t_{kj}^{flight} - t_i^{START} + t_j^{START} \quad (5.6)$$

Imponiendo $\tau_{ij}^i - \tau_{ij}^j$ se puede despejar el tiempo de vuelo, resultando:

$$t_{ij}^{flight} = \frac{\tau_{ij}^j - \tau_{ij}^i}{2} \quad (5.7)$$

Por último, para estimar los instantes de comienzo, se usa la siguiente ecuación:

$$t_i^{START} = \frac{t_{i-1}^{i-1} - t_i^{i-1} + t_{i-1}^i - t_i^i}{2} - t_{i-1}^{START} \quad (5.8)$$

Se toma el primer instante de comienzo como el de referencia, por tanto se le impone que sea cero ($t_0^{START} = 0$).

Con estas estimaciones realizadas a partir de las medidas, se construye la función criterio (*ecuación (5.3)*). El último paso de optimización se puede realizar mediante un descenso de gradiente o mediante un algoritmo de minimización directa como, por ejemplo, el algoritmo de *trust-region*.

Resultado

A pesar de la precisión que ha proporcionado la correlación cruzada de los chirps para el valor de los tiempos de llegada, los resultados proporcionados por el método no fueron los esperados.

De manera general, se impuso que el tiempo de referencia de comienzo para el primer dispositivo sea 0. A partir de este valor, se debe estimar los demás instantes de comienzo.

Se ha comprobado que la optimización requerida nos lleva a tener que resolver un sistema de ecuaciones mal condicionado (con matriz del sistema singular). El hecho de que la matriz de coeficientes del sistema sea singular, desemboca en que el orden de captación de los chirps esté prefijado, siendo el dispositivo que emite el primer chirp el que tiene que captar su propio chirp antes que el resto de los dispositivos. Sin embargo, en la práctica esto no es así, y los chirps no son captados en el orden ideal, por lo que el instante de comienzo tomado como referencia no sería el adecuado.

Como conclusión, no se ha podido continuar con el método en cuestión y se ha optado por la utilización del método de sincronización mediante el servidor.

5.4. Aplicación del método de sincronización

Explicados los dos métodos propuestos y habiendo comprobado que, la sincronización mediante “auto-chirps” no ha sido capaz de devolver resultados coherentes, se va a transportar la sincronización mediante el servidor al caso práctico.

5.4.1. Preparación del entorno adecuado

Es trivial que no cualquier lugar y/o entorno es conveniente para una sincronización acústica. Lo primero que se debe determinar es la distancia mínima a la que se puede encontrar el altavoz conectado al servidor de los micrófonos. Es un factor importante, ya que, el propósito de mantener un sincronismo entre dispositivos es el de poder tratar después las señales resultantes para diariación, localización de fuentes o beamforming, por ejemplo. Estas aplicaciones no se pueden permitir un espacio excesivo, por el contrario, para la mayoría de los casos, el límite de suele encontrar en la longitud de una habitación convencional, lo cual puede rondar los 4 o 5 metros.

En la determinación de las distancias de trabajo, también tendremos en cuenta que, para este trabajo, consideramos operar con ondas planas. Conociendo la ecuación de una onda plana:

$$x(t, \mathbf{r}) = A e^{j(wt - \mathbf{k}\mathbf{r})} \quad (5.9)$$

Donde \mathbf{k} se corresponde con el número de onda y \mathbf{r} el vector de posición. Si la distancia entre el array y el altavoz es lo suficientemente grande en relación a la distancia intermicrofónica, se considera zona de campo lejano y llega un frente de onda plano. La condición para que este fenómeno ocurra es, aproximadamente, la siguiente:

$$r > \frac{2\Delta d^2}{\lambda} \quad (5.10)$$

Dependiendo λ de la frecuencia, tomada como 1kHz, y de la velocidad de propagación del sonido, tomada como 342 m/s. La distancia entre micrófonos se ha aproximado como 10 cm (móviles juntos). Con los datos comentados, resulta un límite de $r > 0.0585m$, por lo que no habría problema en sincronizar móviles en un espacio relativamente pequeño.

Para la correcta ejecución del método de sincronización, los móviles se mantendrán pegados con una distancia intermicrofónica mínima, y sobre un soporte preferiblemente aislante. Este soporte se ha colocado debido a la posibilidad de que se propaguen las ondas por el suelo, ayudando a la mejora de la precisión del sincronismo.



Figura 5.13: Array micrófonos sincronización

Para el correcto funcionamiento del método, es necesario que los micrófonos de los dispositivos estén direccionados correctamente y que, si algún dispositivo posee más de una entrada de audio, desactivarla desde los ajustes del teléfono y forzar una única entrada de audio cuando se emplee la grabadora. De esta forma, se tienen localizados los micrófonos y, por tanto, los elementos del array. En la *Figura 5.13* se pueden identificar los micrófonos de cada móvil con un círculo rojo.¹

¹Es recomendable desactivar el control de ganancia automática y la disminución de eco inteligente desde los ajustes del teléfono si los poseen, ya que pueden estar tratando los datos grabados en formato bruto modificando la señal e interviniendo en el proceso de sincronización.

Es de suma importancia que el entorno en el que se vaya a realizar la prueba de sincronismo, no sea un entorno excesivamente reverberante, ya que posibles reflexiones de la señal podrían dar errores en la captación de la señal acústica de sincronización.

Con las pautas de separación y del entorno claras, se procederá a la realización del método. Los pasos son los siguientes:

1. Configurar la entrada monofónica de audio de cada dispositivo.
2. Conectar los móviles al punto de acceso creado para evitar tráfico y esperas de red no deseadas.
3. Colocación de los dispositivos de forma colindante en un soporte aislante.
4. Colocación del altavoz orientado hacia los dispositivos en línea recta, a una distancia aproximadamente de 1 o 2 metros.
5. Asegurarse de que la orientación de los micrófonos es la adecuada.
6. Establecimiento de la conexión con el servidor
7. Reproducción y captación de la señal tren de impulsos
8. Reproducción de cualquiera señal para comprobar la correcta sincronización.

Una vez completados los anteriores pasos, los ficheros de audio se envían al servidor que los procesa posteriormente afín a la *sección 5.2*.

5.4.2. Resultados y limitaciones

El método implementado es válido para cualquier número de dispositivos, por lo que la portabilidad hacia otros arrays con más elementos es factible.

El error en la precisión del método se establece en un máximo de 5 muestras. Aunque, habitualmente, se produce el sincronismo con una **precisión de 0 o 1 muestra**, por lo que se puede afirmar que es un método prácticamente exacto a nivel de muestra.

A pesar de esta buena precisión y posibilidad de portabilidad, posee algunas limitaciones:

- El altavoz de sincronía deberá ser monofónico, para evitar la superposición de señales y fallos en la captación coordinada de la señal de sincronización.

- La fuente sonora se deberá colocar en una posición fija, donde la distancia a los micrófonos de cada móvil sea la misma.
- La precisión en la sincronización depende del tipo de smartphone que se esté utilizando.
- Existencia de efectos no lineales producidos por el altavoz.
- La posición inicial de los móviles y la orientación de los micrófonos es esencial.

Cabe notar que, para unos mejores resultados de sincronización y sobre todo, consistentes, se debería componer el array lineal de un mismo tipo de smartphone similarmente configurados.

Identificados los factores a tener en cuenta del método de sincronización mediante el servidor, se procede a evaluarlo en el proceso de beamforming que se verá en el siguiente capítulo.

Capítulo 6

Implementación del Beamforming en un array lineal

En este capítulo se evaluará el funcionamiento del algoritmo de sincronización mediante la implementación de un beamformer DAS, sobre un array uniforme con una geometría lineal y una separación entre elementos de 10 centímetros. Se medirá el patrón de directividad de un beamformer Delay & Sum con una ponderación uniforme y se detallará el procedimiento realizado para la implementación de este. El patrón de directividad deberá estar orientado hacia la dirección perpendicular de la línea de sensores. Se podrá cotejar el patrón de directividad experimental conseguido a partir de la sincronización desarrollada con el patrón de directividad teórico del array y beamformer planteados. Por último se hará un resumen de los resultados.

6.1. Análisis experimental

Para obtener el patrón de directividad del beamformer, es necesario realizar un barrido en ángulos con un intervalo pequeño que permita realizar el trazo del patrón de directividad lo más exacto posible.

Con el propósito de tomar estas medidas, se ha hecho un montaje mediante el cual el array de smartphones se rota centrado en el eje de una circunferencia, realizando el barrido de ángulos, mientras la fuente sonora se mantiene fija. Cada móvil se coloca encima de un soporte rectangular, en una posición determinada d , la cual depende de la longitud de onda λ y la frecuencia de la señal f . Este soporte rectangular de 50 cm, se encuentra sobre una base de papel, entrelazados entre sí mediante una chincheta de

soportes moldeables para permitir el giro de la pieza rectangular. El montaje se ilustra en la *Figura 6.1*.



Figura 6.1: Montaje patrón directividad

La orientación y la posición de los dispositivos viene dada por la localización de sus respectivos micrófonos activos.

Además, se han trazado líneas haciendo referencia al ángulo en el que se está midiendo y se ha perforado en la pieza rectangular un orificio el cual permite visualizar el ángulo con el que se está realizando la medición.

La señal de prueba tratada es un seno de 2 kHz de frecuencia el cual se ha segmentado cada dos segundos. Es decir, la señal se encuentra en activo durante dos segundos y otros dos segundos en silencio. Con esto se logra que, el tiempo que la señal esté activa sea captada completamente por el array de micrófonos y cuando la señal está “en reposo” se ejecuta el cambio de orientación en función de ϕ . La señal comentada se muestra en la *Figura 6.2*.

Como es de esperar, antes de tratar la señal mediante beamforming, se necesita un sincronismo entre las señales. Se debe eliminar el retardo de red y de los relojes, existiendo únicamente el retardo procedente al tiempo de propagación. Debido a esto, se debe ejecutar un método de sincronización de los anteriormente comentados antes de realizar el barrido en ϕ . Una vez estén las señales sincronizadas, se deberán colocar los móviles en sus respectivas posiciones del array. El procedimiento del cambio de posición se hace justo después del sincronismo y se ha utilizado material externo

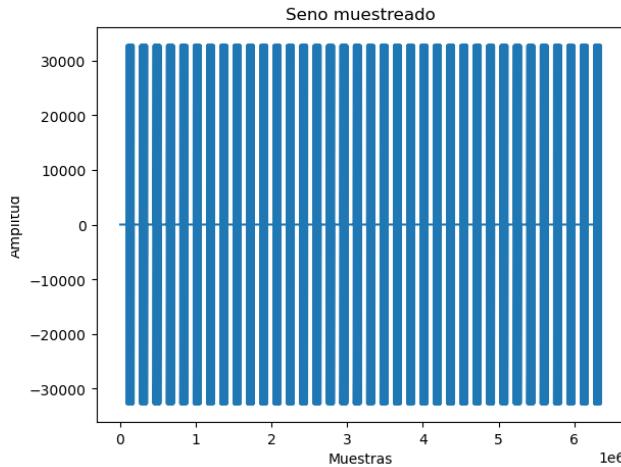


Figura 6.2: Seno 2kHz segmentado

fijador, como cinta, para que durante el barrido, los móviles no sufran ningún desplazamiento imprevisto.¹

Respecto a la distancia entre micrófonos, ésta está directamente relacionada con la longitud de onda, que se calcula como:

$$\lambda = \frac{c}{f} = \frac{342m/s}{2kHz} = 17.1cm \quad (6.1)$$

Según el teorema de muestreo espacial, mostrado en la *ecuación (1.6)*, el límite de la distancia entre los elementos del array se encuentra en:

$$d \leq \frac{17.1cm}{2} \quad (6.2)$$

Por lo que, teniendo en cuenta el máximo valor de distancia² dado por la *ecuación (6.2)* y la longitud de onda resultante, se ha elegido una distancia de 9 cm entre micrófonos y se ha realizado el barrido de ángulos con un paso de $\Delta\phi = 5$ grados.

En el caso expuesto, se realizó un barrido desde 270º hasta 90º. La fuente sonora o altavoz se ha colocado a una distancia de 1.5 metros, en dirección

¹ Las señales correspondientes al proceso de sincronismo y de colocación de los móviles se eliminarán a la hora de la implementación del beamformer.

² Aunque la distancia entre micrófonos sobrepasa ligeramente el teorema de muestreo espacial, se ha fijado este valor porque no se ha percibido un gran efecto de aliasing en el patrón de directividad teórico (Figura 6.4), el cual se verá más adelante. Además, tras numerosas pruebas experimentales modificando minuciosamente la distancia intermicrófónica, es la separación que ha brindado resultados más coherentes.

perpendicular al array. La distancia se ha intentado elegir muy cercana al límite de muestreo espacial, ya que, para $d \leq \frac{\lambda}{2}$ los elementos del array lineal se encuentran en fase y los lóbulos laterales resultantes en la directividad se atenuarán, dando más protagonismo al lóbulo principal, considerando los efectos producidos por aliasing despreciables. Sin embargo, teóricamente, lo ideal y lo que consigue un array longitudinalmente más largo y sin cometer errores por aliasing, es mantener una distancia de $d = \frac{\lambda}{2}$ o cercana, sin sobrepasar este límite.

Resumen del experimento

1. Se colocan todos los dispositivos juntos, con una distancia entre micrófonos mínima.
2. Establecimiento de la conexión con el servidor y puesta a punto para la grabación
3. Reproducción del tren de impulsos correspondiente al algoritmo de sincronización mediante el servidor.
4. Colocación de los móviles en la posición determinada por la d calculada con su debida sujeción al soporte rectangular giratorio.
5. Reproducción del seno muestreado cada dos segundos a 2kHz.
6. Modificación de la orientación del array lineal cuando la señal seno se encuentre en reposo.
7. Pasadas las 36 medidas, se corta la grabación y se guardan los archivos de audio para manipularlos.

6.2. Algoritmo y representación

La *ecuación (1.1)* se puede expresar alternativamente de la siguiente manera:

$$D(f, \phi) = \sum_{n=0}^{N-1} w_n^*(f) e^{\frac{2j\pi f}{c} nd \cos \phi} \quad (6.3)$$

Conociendo la dirección deseada o *target*, la cual es 90° , se calcularán los pesos correspondientes. Para trazar el patrón directivo, solo resta conocer el barrido angular, la distancia y la frecuencia de la señal como establece la *ecuación (6.3)*.

RETARDO o *DELAY*

Los retardos necesarios se computarán de la forma:

$$t_n = nd\cos\phi_s/c \quad (6.4)$$

Como la dirección target es 90° , el retardo resultante es 0 para todos los sensores, y según la *ecuación (1.2.1)*, el *steering vector* estará compuesto únicamente con valores de $1/N$, siendo N el número de dispositivos, que en el caso experimental es 3.

SUM

Una vez conocidos los pesos del beamformer, solo resta sumar las señales retardadas, para que se implemente la suma constructiva de la señal en la dirección deseada y la suma destructiva en las demás direcciones.

En el experimento presentado se trata simplemente de un promedio de las señales de entrada, debido a los valores del *steering vector*. La señal salida del beamformer delay & sum se muestra en la *Figura 6.3* para la secuencia de ángulos de llegada seleccionada.

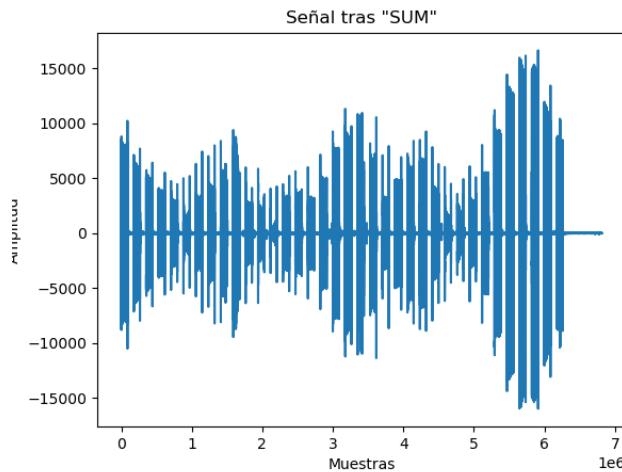


Figura 6.3: Señal a la salida del beamformer

Tal y como se ha comentado, la señal queda realizada en la parte de la dirección target. En este caso, al realizar el barrido desde 270° hasta 90° , la amplificación de la señal interviene en la parte final. Cabe notar que la parte restante de la circunferencia se completa con simetría.

Por último se ha evaluado la parte de la señal correspondiente al cambio de orientación en el montaje y el ruido de fondo. Para ello se ha seleccionado

un mismo segmento de señal en reposo, tanto para la señal de salida del beamformer como para la señal de entrada a este. Se ha calculado la varianza de la señal en los dos casos:

Medida del ruido	Antes del beamformer	Después del beamformer
Varianza	1161.773	209.204

Se comprueba que consigue reducirse la potencia del ruido en un 82 % aproximadamente. Esta conclusión confirma que, el beamformer DAS, además de direccionar y realzar la señal de interés, atenúa las contribuciones ruidosas.

DIRECTIVIDAD

Para la interpretación del trazo del patrón directivo, mediante la *ecuación (6.3)*, se ha representado el patrón de directividad teórico para una distancia $d = 9\text{cm}$ y una frecuencia de $f = 2\text{kHz}$ (la correspondiente a la señal seno). El resultado se muestra en la *Figura 6.4*.

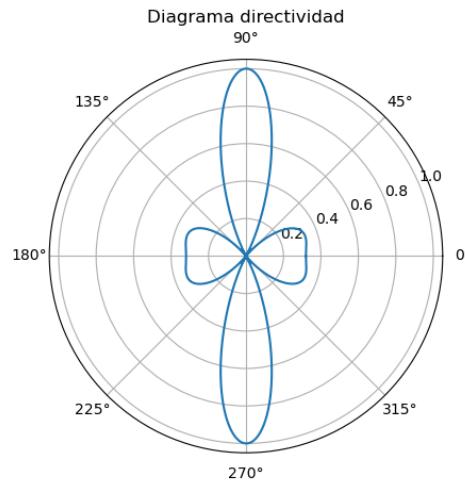


Figura 6.4: Patrón de directividad teórico para $d = 9\text{cm}$ y $f = 2\text{kHz}$

El máximo del lóbulo principal se encuentra en $\pm 90\text{grados}$, como era de esperar y los lóbulos laterales son relativamente pequeños en función al lóbulo principal. Se corresponde con una distribución del tipo *broadside*. Como se ha comentado anteriormente, no se perciben efectos notables de aliasing, debido al pequeño exceso de distancia respecto al límite del teorema espacial.

Conocida la forma aproximada que debería presentar el patrón experimental, se procede al cálculo de este. Con la señal a la salida del beamformer

(Figura 6.3), se segmenta la señal para aislar las 36 partes activas del seno. Cada uno de estos segmentos se corresponde con la señal medida en un valor del ángulo ϕ , por lo que se calcula los picos del segmento y se promedian. De este modo conseguimos un único valor de amplitud para poder realizar el trazo del patrón de directividad correctamente.

Con las 36 medidas de amplitud y con los ángulos del barrido, se presenta en coordenadas polares el patrón de directividad experimental en la Figura 6.5.

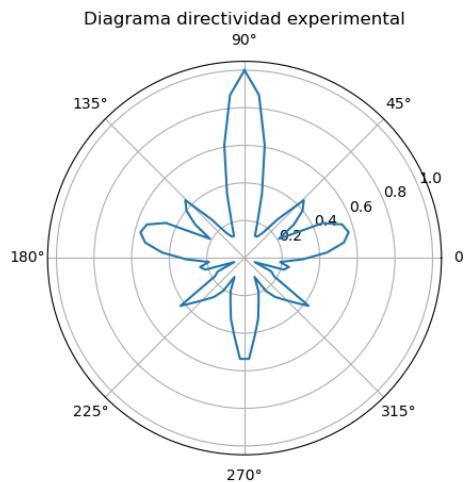


Figura 6.5: Directividad experimental

Se aprecia un claro máximo en la dirección de 90 grados, con una reducción considerable en la dirección de 270 grados. Esto puede ser debido a dos factores:

- Micrófono de tipo cardioide: Este tipo de micrófonos captan primariamente el sonido en su parte frontal, atenuando consecuentemente el sonido en sus laterales y en la parte posterior. El patrón polar de este tipo de micrófonos se muestra en la Figura 6.6.
- Sombra acústica: Este fenómeno se produce cuando, por ejemplo, el micrófono se encuentra en la parte inferior del móvil, todos los sonidos procedente de la dirección de la parte superior se encontrarán primero con la propia estructura del móvil, atenuando las ondas antes de llegar al micrófono.

Con el fin de comprobar la fiabilidad del patrón de directividad resultante, se ha calculado el error cuadrático medio de la forma:

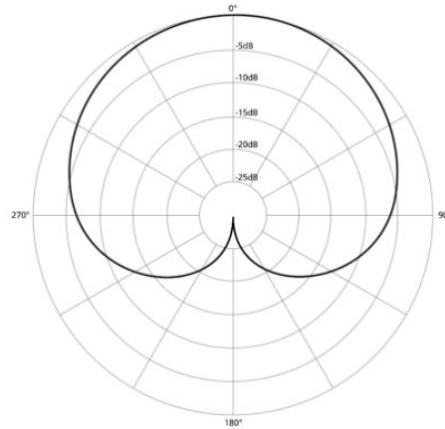


Figura 6.6: Patrón polar micrófono cardioide

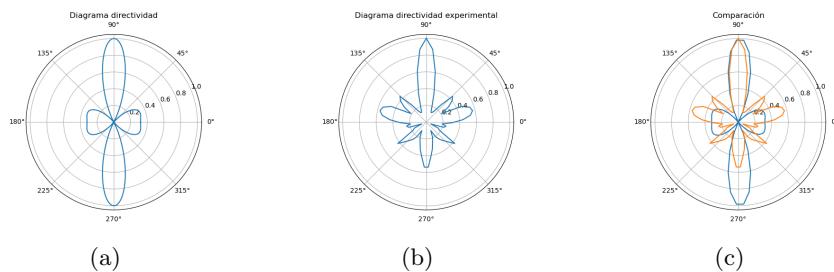


Figura 6.7: Comparación patrones de directividad teórico y experimental

$$MSE = \frac{1}{N_\phi} \sum_{\phi=0}^{\limBarrido} (D_{sim} - D_{exp})^2 \quad (6.5)$$

Siendo D_{sim} el módulo de la directividad teórica, expuesto en la *Figura 6.4* y D_{exp} el módulo de la directividad experimental. Resulta un error cuadrático medio de 0.08278.

En la *Figuras 6.7a y 6.7b* se muestran los patrones de directividad teórico y experimental respectivamente y en la *Figura 6.7c* una comparación del trazado de estos. Se comprueba que el lóbulo principal del patrón de directividad experimental se ajusta bastante a la forma del lóbulo del patrón de directividad teórico.

Otra medida para evaluar el patrón de directividad resultante es el *factor de directividad*. El factor de directividad (F_d) se define como el cociente entre el patrón de directividad en la dirección deseada (en este caso $\phi = \pi/2$) y el patrón de directividad promedio para todas las direcciones.

$$F_d(f, \phi_s) = \frac{|D(f, \phi_s)|^2}{\frac{1}{N_\phi} \sum_{\phi=0} |D(f, \phi)|^2} \quad (6.6)$$

Se ha obtenido un factor de directividad para el patrón teórico de 5.77 (7.61 dB) y un factor de directividad para el patrón experimental de 6.28 (7.97 dB). El factor de directividad del beamformer experimental es mayor, ya que existen efectos que aumentan la directividad de los micrófonos involucrados en el array como la sombra acústica o la existencia de un micrófono cardioide. Estos factores hacen que la parte opuesta de los elementos del array no consiga los mismos resultados (captación en menor medida del sonido) que la parte frontal, donde se encuentran los micrófonos.

6.3. Evaluación de los resultados

Los aspectos favorables del experimento según los resultados obtenidos son los siguientes:

- Se ha logrado medir la señal correctamente para cada ángulo del barrido realizado.
- El beamformer ha realizado la señal en la dirección de interés correctamente, atenuando las señales procedentes de otras direcciones.
- La sincronización de los móviles para la prueba realizada ha sido lo suficientemente precisa para la correcta implementación del beamformer.
- Se ha conseguido un patrón de directividad experimental muy semejante al teórico, obteniendo un error cuadrático medio bastante razonable considerando las limitaciones del mecanismo de sincronización.

Sin embargo, existen múltiples factores a tener en cuenta, los cuales han limitado considerablemente el experimento. Tras numerosas pruebas examinando las causas de resultados incorrectos, se han determinado las siguientes restricciones:

- El método de sincronización implementado puede cometer errores de varias muestras, siendo este error dependiente del tipo de smartphone en cuestión o de los efectos no lineales del altavoz.
- El procedimiento de beamforming es muy sensible a errores de sincronización, si todos los dispositivos no están perfectamente sincronizados

de antemano puede dar errores y no actuar como se espera. En la *Figura 6.8*, se puede apreciar una prueba realizada en la que los móviles no han estado perfectamente sincronizados. Para forzar el error de sincronización, simplemente se han espaciado los móviles unos centímetros respecto a su posición inicial de sincronización (en la cual deben estar lo más juntos posible). Al efectuar esta separación, el tiempo de propagación gana protagonismo en el proceso de transmisión de la señal de sincronización y no se obtienen unos instantes de comienzo precisos.

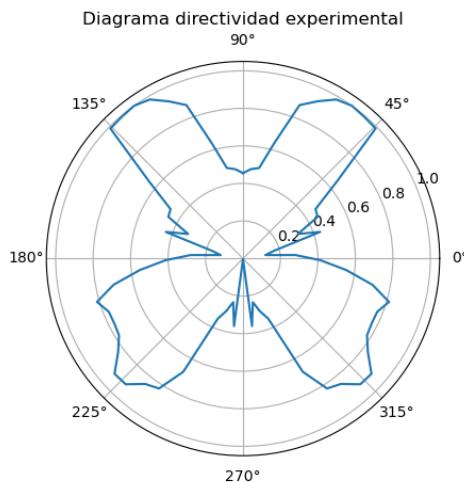


Figura 6.8: Fallo sincronismo

Tras la reproducción de la señal de sincronización, se hace con normalidad la prueba experimental del barrido, anteriormente explicada, y se representan los patrones de directividad de la misma manera. Como puede verse, no se produce la correcta amplificación en la dirección deseada. Además, se produce una subdivisión en el lóbulo principal del factor de directividad, debido al fallos de sincronismo. El error cuadrático medio (MSE) en este caso, calculado de manera similar a la *ecuación (6.5)*, resulta 6.27886, considerablemente superior al error cometido con una buena sincronización.

- Para el correcto funcionamiento del método de sincronismo del servidor, es necesario que todos los smartphones posean una única entrada de audio (modo monofónico). Esto limita mucho los máximos en la parte supuestamente simétrica del patrón de directividad. Al encontrarse el array en $\phi = 90$ los micrófonos activos están orientados hacia la fuente sonora. Sin embargo, en la orientación de $\phi = 270$ los micrófonos activos se encuentran en la parte opuesta. Se debe tener en cuenta los efectos comentados de sombra acústica y de la posibilidad de que el micrófono sea del tipo cardioide.

- Aunque se hayan tomado las distancias de una forma precisa y se hayan realizado las medidas con precaución, sigue siendo un experimento con un montaje hecho a mano y limitado por el error humano, por lo que las medidas de los ángulos dentro del barrido y las distancias entre micrófonos no son completamente exactas, lo que llevar hasta resultados ligeramente erróneos.

De forma análoga al procedimiento de sincronismo, estas limitaciones serían menos notables si se emplean un mismo modelo de smartphone como cada elemento del array, consiguiendo de esta manera eliminar posibles diferencias de la captación del audio, colocación de los micrófonos y sincronismo. De este modo, los resultados del beamforming con su consecuente patrón de directividad serían más precisos.

Capítulo 7

Conclusiones y trabajo futuro

Como síntesis, se hace un balance del proyecto al completo, resaltando algunas posibles mejoras y aplicaciones.

7.1. Balance del proyecto

Se comenzó proporcionando algunas nociones básicas sobre procesamiento de señales multicanal y beamforming, con el propósito de llegar al máximo entendimiento de la parte principal del trabajo realizado. También se dio una visión más amplia y práctica de las posibles aplicaciones de las técnicas expuestas en el fundamento teórico para hacer ver que, en la actualidad, son notablemente utilizadas.

Posteriormente, se presentó el problema y el tema en específico a desarrollar en el proyecto. Esto abarcó tanto la enunciación de los objetivos esperados como los materiales y entornos de trabajo necesarios. Una vez dado el fundamento teórico de interés y mencionado el problema a tratar, se procedió a la explicación de la parte principal del proyecto.

Se consiguió mantener una conexión robusta entre los móviles y un servidor externo, mediante la ayuda de una aplicación android creada, pudiendo analizar de una forma desglosada el tráfico de red correspondiente. Tanto la aplicación como el servidor son herramientas bastante portables, ya que el servidor puede abarcar simultáneamente hasta 10 clientes por lo que, probablemente, la conexión sea útil para agrupaciones de smartphones con un número mayor de elementos del array.

Con la aplicación y el servidor en correcto funcionamiento, se continuó con la implementación de un método de sincronización acústica. Al inicio del

estudio, el objetivo era presentar dos métodos de sincronización distintos, realizando la respectiva comparación entre ellos. Debido a factores prácticos y experimentales, no fue posible llevar a cabo uno de los dos métodos con éxito, por lo que se acabó implementando un método basado en una excitación acústica inicial proporcionada por un servidor externo. Para este último método de sincronismo comentado, se realizaron numerosas pruebas, analizando cuál podría ser la señal óptima de sincronización acústica y la preparación del entorno exacta para una mayor precisión. Favorablemente, se obtuvo una precisión de sincronismo bastante buena, por lo que se determinó una opción totalmente válida y efectiva para aplicaciones como beamforming o localización de fuentes sonoras.

Por último, se implementó el algoritmo de un beamformer *Delay and Sum* con el objetivo de trazar un patrón de directividad experimental y otro teórico, haciendo la correspondiente comparación entre ellos. Para el trazo del patrón de directividad experimental del beamformer, se hizo un montaje de forma que se lograse un barrido de al menos 180 grados de la agrupación de dispositivos en función de una fuente sonora fija. Los resultados experimentales primeramente, no fueron del todo coherentes, por lo que se mejoró la precisión del montaje y de la ejecución del experimento, obteniendo unos resultados bastante similares a los teóricos. La evaluación de este apartado es existosa ya que, a pesar del error humano y de la poca precisión del montaje, sumando la posible inexactitud del sincronismo entre señales, el patrón de directividad final es coherente al fundamento teórico, lo que indica el correcto funcionamiento del beamformer y del array de micrófonos. Como se ha comentado en anteriores secciones, una mejora de este proceso de beamforming sería que los elementos del array fuesen iguales, es decir, dispositivos del mismo modelo, aproximando de esta manera elementos isotrópicos. En general, para la concreta disponibilidad de material y de tiempo, se hace un balance altamente favorable tanto de los resultados como de las vías de investigación y desarrollo del experimento.

Existen algunos aspectos los cuales se pueden mejorar dependiendo del propósito del experimento:

- Por ejemplo, la adición de una transmisión de los datos del audio grabado por los móviles en *streaming*, simplificando de esta manera el almacenamiento de estos ficheros y proporcionando la posibilidad de tratar las señales de forma **online**.
- Además, la alternativa de sincronización propuesta es sensible ante factores cambiantes del entorno, que pueden ser sonidos inesperados, debidos a reflexiones y ruido, o efectos no lineales del altavoz emisor de la señal de excitación, por lo que se podría mejorar el entorno mediante aislantes acústicos, evitando reflexiones, y regular el volumen

del altavoz para evitar algunos efectos no lineales.

- No todos los smartphones del mismo tipo llegan a sincronizarse de forma óptima con este método de sincronización. A parte, existe un impacto notable en los resultados por el uso de micrófonos del tipo cardioide, los cuales son bastante comunes. La posible mejora en este caso sería utilizar para todos los elementos del array Ad-Hoc un mismo tipo de smartphone y asegurando el uso de micrófonos omnidireccionales (captan todos los sonidos de cualquier dirección) o bidireccionales.
- Para no depender de un elemento externo, como es el altavoz para la emisión de la señal de sincronización, se propone desarrollar, de una forma innovadora el método basado en “auto-chirps”, obteniendo resultados consistentes.

7.2. Perspectiva de futuro

Es de esperar de un proyecto o de una investigación que sea viable que mantenga un punto de vista hacia el futuro. En este capítulo se expondrá dos posibles vías hacia el futuro, empleando técnicas estudiadas en el proyecto actual. Estas dos vías, no tan lejanas del presente, son el *5G* y el concepto de *smart home*.

7.2.1. Reconocimiento automático de voz

El reconocimiento automático de voz o Automatic Speech Recognition (ASR) en inglés, es un procedimiento el cual incorporan actualmente varios sistemas de inteligencia artificial. Este tipo de tecnología permite al ser humano comunicarse con una máquina a través de la propia voz, pudiendo identificar la máquina los patrones característicos de la voz del interlocutor. El ASR es utilizado en dispositivos inteligentes como *Alexa*. El fundamento del reconocimiento de voz para la familia de dispositivos “Amazon Echo” se puede consultar en [28].

Los arrays de micrófonos son utilizados en ASR. La voz de la persona oradora es captada por una agrupación de M micrófonos. Se debe tener en cuenta que van a existir reflexiones del sonido y ruido en la estancia donde se sitúe el dispositivo inteligente. Además, los micrófonos captarán también la señal “respuesta” emitida por los L altavoces del dispositivo. La situación comentada se ilustra en la *Figura 7.1*.

En el bloque de “Audio Front End” se hace el procesamiento del audio. Esto son algoritmos de localización de fuentes sonoras, para determinar la dirección *target*, beamforming, para combinar todas las señales procedentes

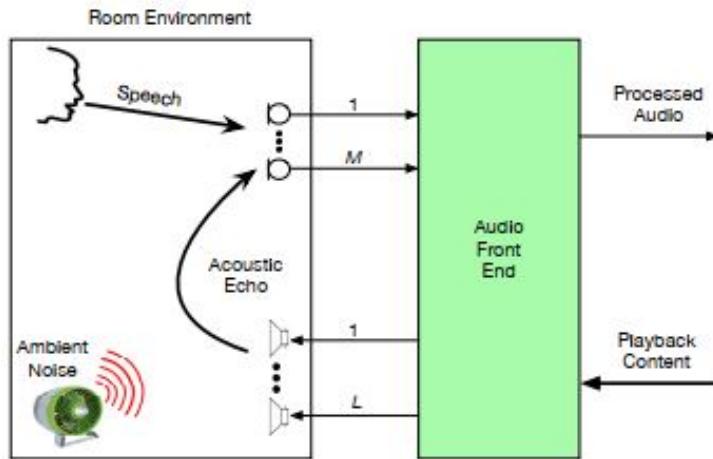


Figura 7.1: Entorno acústico ASR

de los elementos del array microfónico y realizar la señal de interés en la dirección dada por el algoritmo de localización de fuentes sonoras, y cancelación de eco.

Todo el procedimiento comentado se debe realizar a tiempo real, para simular un diálogo. El audio se procesa por tramas y los elementos del array deberán estar previamente sincronizados.

7.2.2. Sincronización en casas inteligentes

El internet de las cosas o *IoT* ha ido avanzando progresivamente. Con la incorporación del *Big Data*, la aplicación del internet de las cosas se ha generalizado hasta llegar a los hogares de muchas personas. Estos hogares mencionados, hacen uso de dispositivos electrónicos inteligentes, los cuales pueden llegar a proporcionar una mejor calidad de vida y bienestar, además de un mayor control de la energía utilizada.

La gran conectividad de estos dispositivos hace necesaria una precisa sincronización temporal entre dispositivos. Estos dispositivos electrónicos emplean la energía proporcionada según su ciclo de trabajo o *Duty Cycle*. El ciclo de trabajo define el tiempo que está en activo o en reposo cada dispositivo. En la *Figura 7.2* se puede ver distintos ciclos de trabajo, los cuales se elegirán según las necesidades requeridas.

Adaptando el ciclo de trabajo para todos los dispositivos, consiguiendo una sincronía entre ellos, se puede ahorrar considerablemente energía, tal y como sostiene [26]. De esta forma, al comunicarse un dispositivo con otro de la red, los datos enviados por uno de ellos serán procesados en el mismo instante que llegan al receptor, ya que este último se encontrará en estado

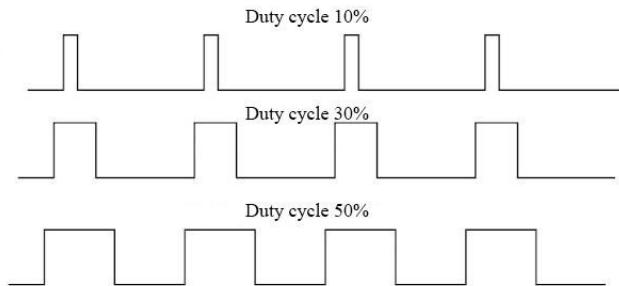


Figura 7.2: Duty Cycle

activo.

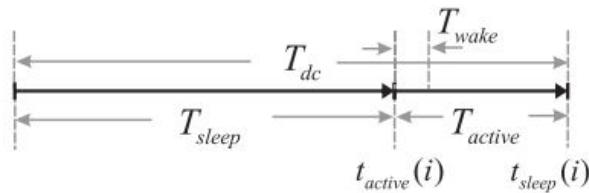


Figura 7.3: Tiempos en ciclo de trabajo

Sin embargo, si no existe una sincronización entre ciclos de trabajo, los datos pueden no llegar en el estado activo del dispositivo receptor, retrasando así el procesamiento de los datos y gastando más energía. Las marcas temporales del dispositivo receptor se pueden visualizar en la *Figura 7.3*, donde T_{sleep} se corresponde con el tiempo en reposo, T_{active} con el tiempo en activo y T_{wake} el instante en el que recibe el dispositivo receptor los datos enviados por otro dispositivo.

Una variante adaptada a la red de una casa inteligente se podría basar en el método de sincronización acústica planteado en el proyecto actual. La condición sería que todos los dispositivos estuvieran en una misma habitación. Además, el método de sincronización propuesto puede ser suficientemente preciso para adaptar el *Duty Cycle*, es simple de implementar, no malgasta energía y es barato, ya que el único hardware requerido para todos los dispositivos es un micrófono. El modo de operación sería maestro-esclavo, como se propone en [27], el cual sostiene un método de sincronización acústico para el mismo propósito de sincronizar los dispositivos inteligentes de una habitación.

Capítulo 8

Presupuesto del trabajo

Se va a realizar una aproximación del coste total del proyecto. Para ello, se ha tenido en cuenta el tiempo invertido por los contribuyentes al trabajo y todo el material utilizado.

8.1. Inversión en capital humano

Para determinar cuantitativamente el dinero relacionado con los costes de personal, se han contabilizado las reuniones mantenidas y el tiempo invertido en buscar alternativas a diversos problemas propuestos. Se han mantenido un total de 6 tutorías, con un tiempo cada una de 1 hora, aproximadamente. El tiempo de planteamiento de algún problema por los tutores, se ha determinado en 30 minutos por cada adversidad propuesta. Para determinar el tiempo invertido por el alumno, teniendo en cuenta el sistema universitario español, un crédito ECTS equivale a 25 horas de trabajo por el estudiante, teniendo en cuenta que el trabajo fin de grado consta de 12 créditos ECTS.

Adicionalmente, se ha diferenciado el coste de trabajo entre un ingeniero junior y un ingeniero senior.

Trabajador	€/ hora	Tiempo	Cuantía
Ingeniero Senior	60	9 horas	540 €
Ingeniero Senior	60	9 horas	540 €
Ingeniero Junior	30	300 horas	9000 €

Cuadro 8.1: Costes de personal

Obteniendo finalmente, una inversión total en capital humano de **10080 €**.

8.2. Inversión en material

El coste de materiales y hardware necesarios en el desarrollo del proyecto es el siguiente:

Tipo	Material	Precio
Smartphone	Huawei p30 pro	500 €
Smartphone	PocoPhone M3 Pro	150 €
Smartphone	Huawei P Smart Z	220 €
Ordenador	HP Pavilion 15 i7	969 €
Montaje	Cartón Pluma	20 €
Montaje	Chinchetas	2 €
Altavoz	JBL go 3	40 €

Cuadro 8.2: Costes materiales

Obteniendo finalmente, una inversión total en material de **1901 €.**

Debe notarse que el coste presentado en esta sección es relativo, ya que se podría haber desarrollado el proyecto con otros materiales. El precio depende de la disponibilidad en cada caso.

Bibliografía

- [1] McCowan, I. (2001). Microphone arrays: A tutorial. Queensland University, Australia, 1-38.
- [2] Liaquat, M.U.; Munawar, H.S.; Rahman, A.; Qadir, Z.; Kouzani, A.Z.; Mahmud, M.A.P. Sound Localization for Ad-Hoc Microphone Arrays. Energies 2021.
- [3] S. Pasha, C. Ritz and J. Lundgren, .^A Survey on Ad Hoc Signal Processing: Applications, Challenges and State-of-the-Art Techniques,” 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2019, pp. 1-6
- [4] B. D. Van Veen and K. M. Buckley, “Beamforming: a versatile approach to spatial filtering,” IEEE ASSP mag., vol. 5, no. 2, pp. 4–24, 1988.
- [5] C. Pan, J. Chen, and J. Benesty, “Performance study of the MVDR beamformer as a function of the source incidence angle,” IEEE ACM Trans. Audio Speech Lang. Process., vol. 22, no. 1, pp. 67–79, 2014.
- [6] T. Yardibi, C. Bahr, N. Zawodny, F. Liu, L. N. Cattafesta III, and J. Li, “Uncertainty analysis of the standard delay-and-sum beamformer and array calibration,” J. Sound Vib., vol. 329, no. 13, pp. 2654–2682, 2010.
- [7] R. Berkun, I. Cohen, and J. Benesty, “Combined Beamformers for Robust Broadband Regularized Superdirective Beamforming,” IEEE ACM Trans. Audio Speech Lang. Process., vol. 23, no. 5, pp. 1–1, 2015.
- [8] X. Anguera, C. Wooters, and J. Hernando, “Acoustic Beamforming for Speaker Diarization of Meetings,” IEEE Trans. Audio Speech Lang. Processing, vol. 15, no. 7, pp. 2011–2022, 2007.
- [9] P. Chiariotti, M. Martarelli, and P. Castellini, “Acoustic beamforming for noise source localization – Reviews, methodology and applications,” Mech. Syst. Signal Process., vol. 120, pp. 422–448, 2019.

- [10] S. Araki, N. Ono, K. Kinoshita and M. Delcroix, “Comparison of Reference Microphone Selection Algorithms for Distributed Microphone Array Based Speech Enhancement in Meeting Recognition Scenarios,” 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC), 2018, pp. 316–320.
- [11] N. Cardwell, S. Savage, and T. Anderson, “Modeling TCP latency,” in Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), 2002, vol. 3, pp. 1742–1751 vol.3.
- [12] “Java platform SE 7,” Oracle.com. [Online]. Available: <https://docs.oracle.com/javase/7/docs/api/>.
- [13] [23] E. Bisong, “NumPy,” in Building Machine Learning and Deep Learning Models on Google Cloud Platform, Berkeley, CA: Apress, 2019, pp. 91–113.
- [14] “Wireshark . Go Deep,” Wireshark.org. [Online]. Available: <https://www.wireshark.org/>.
- [15] “PlantUML Web Server,” Plantuml.com. [Online]. Available: <http://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000>.
- [16] Calvert et al., “9780123742551 - TCP/IP sockets in java, second edition,” Biblio.com, 22-Feb-2008. [Online]. Available: <https://www.biblio.com/9780123742551>.
- [17] M. L. Murphy, The busy coder’s guide to android development, 2nd ed. Commonsware, 2009.
- [18] “Android API reference,” Android Developers. [Online]. Available: <https://developer.android.com/reference>.
- [19] N. Henmi, T. Saito, and T. Ishida, “Prechirp technique as a linear dispersion compensation for ultrahigh-speed long-span intensity modulation directed detection optical communication systems,” J. Lightwave Technol., vol. 12, no. 10, pp. 1706–1719, 1994.
- [20] C. Chamorro-Zurita and E. Ovando-Shelley, “Análisis de señales: Aplicación de la correlación cruzada discreta para la determinación del primer arribo de la onda cortante,” XXIX Reunión de Ingeniería Geotécnica, 2018.
- [21] P. P. de la Torre, Á. M. G. García, J. M. M. Ramos, C. G. Ruiz, and M. del Carmen Benítez Ortúzar, Señales digitales.

- [22] Y. Kaneda, “A study of non-linear effect on acoustic impulse response measurement,” *J. Acoust. Soc. Jpn. (E)*, vol. 16, no. 3, pp. 193–195, 1995.
- [23] M. H. Hennecke and G. A. Fink, “Towards acoustic self-localization of ad hoc smartphone arrays,” in *2011 Joint Workshop on Hands-free Speech Communication and Microphone Arrays*, 2011.
- [24] A. Plinge, F. Jacob, R. Haeb-Umbach, and G. A. Fink, “Acoustic Microphone Geometry Calibration: An overview and experimental evaluation of state-of-the-art algorithms,” *IEEE Signal Process. Mag.*, vol. 33, no. 4, pp. 14–29, 2016.
- [25] A. C. Aznar, *Antenas*. Alfaomega Grupo Editor, 2000.
- [26] F. Li, Y. Yang, Z. Chi, L. Zhao, Y. Yang, and J. Luo, “Trinity: Enabling self-sustaining WSNs indoors with energy-free sensing and networking,” *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 2, pp. 1–27, 2018.
- [27] Z. Nakad, M. A. Sayed, A. Yaghi, H. Margossian, and W. Fawaz, “A novel offline indoor acoustic synchronization protocol: experimental analysis,” *Ann. Telecommun.*, vol. 77, no. 3–4, pp. 221–236, 2022.
- [28] A. Chhetri et al., ”Multichannel Audio Front-End for Far-Field Automatic Speech Recognition,” *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018

Capítulo 9

Apéndice

Se presenta en este capítulo algunas funciones y procedimientos de utilidad programados en Python específicamente para este proyecto.

9.1. Apéndice A

La creación del tren de impulsos utilizado para fines de sincronización:

```
def tren_impulsos(Nimp, len_tren):
    'Función que crea una señal acústica de sincronización'
    'correspondiente a un tren de un número de impulsos dado'

    Fs = 44100
    tren = np.zeros(len_tren)

    #Longitud en muestras de cada impulso
    len_impulso = 10
    #margen de muestras para que se reproduzca correctamente
    margen = 2000
    #Separación entre impulsos
    intervalo = int((len_tren - margen) / Nimp)
    #Posición primer impulso
    pos_ini = 10000

    for i in range(Nimp):
        pos = pos_ini + intervalo * i
        for j in range(len_impulso):
            tren[pos + j] = np.iinfo(np.int16).max

    return tren
```

Figura 9.1: Código creación tren de impulsos

9.2. Apéndice B

La correlación cruzada en el método de sincronización mediante el servidor entre la señal de sincronismo captada en el instante de referencia 0 y las demás señales captadas por los restantes dispositivos:

```
primera = np.argmin(toa)
print('Comienza Device ' + str(primera))
for i in range(Ndevices):

    rx_tren2 = scipy.signal.correlate(tren[:, i], tren[:, primera], mode = 'full', method='fft')
    N = len(rx_tren2)
    k = np.linspace(-(N/2) +1, (N/2)-1, N)

    plt.figure(i+1)
    plt.title('Correlación cruzada tren de impulsos')
    plt.xlabel('Muestras')
    plt.ylabel('Correlación')
    plt.plot(k,rx_tren2)

    toamed[i] = correlaMax(rx_tren2, len(rx_tren2))
    tam_postdelay[i] = tam[i] - toamed[i]
```

Figura 9.2: Código correlación cruzada tren de impulsos

9.3. Apéndice C

Procedimiento seguido para implementar el beamformer *Delay and Sum*. Cálculo del retardo y de los pesos y posterior suma de las señales:

```
##%% RETARDO (delay)

retardo = (n * d * np.cos(phi)) / c

#PESOS
for i in range (Ndevices):
    w[i] = np.exp(1j * 2 * np.pi * f * retardo[i])

w = w / Ndevices

##%% PROMEDIO (sum)

for j in range (Ndevices):
    y = y + (sincronizadas[recorte:, j] * w[j])
```

Figura 9.3: Código DAS

9.4. Apéndice D

Señal creada para medir la calidad del beamformer. Consta de una señal sinusoidal a una frecuencia f , con una duración total de t y un tiempo en reposo de t_{wait} .

```
def waiterSin(f, t, twait):
    'Función que genera un seno a una frecuencia f,
    'con una duración de t segundos y muestreado
    'cada twait segundos'

    Fs = 44100
    nwait = twait * Fs
    Ts = 1 / Fs
    n = t / Ts
    n = np.arange(n)
    t_vector = n * Ts
    A = np.iinfo(np.int16).max

    s = A * np.sin(2 * np.pi * f * t_vector)

    for i in range(0, len(s), 2*nwait):
        for j in range(nwait):
            s[i + j] = 0

    plt.figure(1)
    plt.plot(s)
    plt.title('Seno muestreado')
    plt.xlabel('Muestras')
    plt.ylabel('Amplitud')

    file_name1 = 'seno2k'
    wavfile.write('seno/' + file_name1 + '.wav', Fs, s.astype('int16'))

    pass
```

Figura 9.4: Señal seno creada

