UNIVERSIDAD DE VALLADOLID
E.T.S.I DE TELECOMUNICACIÓN

TRABAJO FINAL DE GRADO
GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS
DE TELECOMUNICACIÓN

Influencia del número de receptores en un radar pasivo acústico basado en datos de amplitud

Autor: Agustín José Pallarés García
Tutor: Juan Blas Prieto

26 de julio de 2018

TÍTULO:	Influencia del número de receptores en un radar pasivo acústico basado en datos de amplitud
AUTOR:	Agustín José Pallarés García
TUTOR:	Juan Blas Prieto
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Dr. Evaristo José Abril Domingo
VOCAL:	Dr. Juan Carlos Aguado Manzano
SECRETARIO:	Dr. Juan Blas Prieto
SUPLENTE:	Dr. Jaime Gómez Gil
SUPLENTE:	Dr. Ramón de la Rosa Steinz
FECHA:	3 Septiembre de 2018
CALIFICACIÓN:	

Resumen

El número y la posición de los elementos del array, componente fundamental del radar acústico pasivo, es vital para caracterizar este tipo de sistemas. En este trabajo dicho array se sitúa en el interior de una cavidad resonante en forma de cubo con paredes de fibrocemento en la que penetra el frente de onda a través de una apertura. Tras automatizar el desplazamiento de la fuente externa para conseguir la capacidad de adquisición de datos necesaria, se ha realizado una campaña de medidas a una frecuencia de 5 kHz. A modo de conclusión, los resultados de referencia obtenidos usando 400 elementos empeoran en general al reducir el número de elementos, aunque depende de la posición de la fuente externa. Por tanto es necesario intentar optimizar el algoritmo de procesado de datos antes de reducir el número de elementos.

Abstract

The number and position of the array elements, a fundamental component of passive acoustic radar, is vital to characterize this type of system. In this work, this array is placed inside a resonant cube-shaped cavity with fibrocement walls into which the wavefront penetrates through an aperture. After automating the movement of the external source to achieve the necessary data acquisition capacity. A measurement campaign was carried out at a frequency of 5 kHz. To conclude, the reference results obtained using 400 elements generally get worse by reducing the number of elements, although it depends on the position of the external source. It is therefore necessary to try to optimize the data processing algorithm before reducing the number of elements.

«Para mi familia»

—

Agradecimientos

Con estas líneas querría expresar mi agradecimiento:
A mi tutor Juan Blas, por todo el tiempo que ha aportado y la paciencia que ha tenido conmigo a la hora de llevar a cabo este trabajo. Gracias por la oportunidad de poder desarrollar este trabajo fin de grado con usted.
A mis padres, por haberme dado los medios para poder recibir la mejor educación que se podía tener. Por todo el apoyo que me han dado día a día y las lecciones de vida que me han ayudado a realizar todo lo que he realizado hasta el día de hoy.
A mi padre Agustín, por demostrarme que una persona puede conseguir todo lo que se propone a base de esfuerzo, lucha, sangre y sudor.
A mi madre María Jesús, por enseñarme que el valor de una persona se encuentra en el corazón y que hay que ayudar a todo aquel que te rodea.
A mi hermana Lorena, por apoyarme en todas las decisiones que he realizado a lo largo de mi vida.
A mi hermano José María, ese granuja que me ha obligado a intentar ser el mejor ejemplo para él (sin llegar a conseguirlo) y que me ha convertido en mejor persona desde que nació.
A esa persona especial que ha vivido tantos momentos conmigo. Siempre me ha sacado una sonrisa en los peores momentos.
A mis amigos de toda la vida, estos tíos con quienes vives los mejores momentos de tu vida.
A mis amigos que esta preciosa carrera me ha traído. En tan pocos años os habéis convertido en un pilar muy importante de mi vida.
A todos los profesores y personal de la escuela que han convertido esta preciosa facultad en mi segunda casa.

A todas aquellas personas que he conocido a lo largo de mi vida que han hecho que me convierta en la persona que hoy soy.

Al igual que mío, esto también es vuestro. **Gracias**.

Índice general

Capítulo 1.

Introducción

En el presente capítulo se contextualiza el proyecto llevado a cabo. Se empieza presentando el principio de Huygens-Fresnel como explicación cualitativa de la propagación de ondas acústicas. Se continúa explicando los fundamentos físicos de la difracción de ondas acústicas sobre los que se basa este trabajo y por último se explica el Principio de Equivalencia, que tiene un papel central en la realización de las simulaciones.

Contenido

1.1. Contexto

Se dispone de una cavidad de fibrocemento en cuyo interior se encuentra un micrófono controlado por varios motores paso a paso para la captación de ondas acústicas emitidas desde el exterior de esta cavidad. El emisor es una bocina acústica alimentada por un generador de ondas. El objetivo es analizar la influencia de la densidad y localización de las muestras de intensidad acústica tomadas en el interior de la cavidad de cara a identificar la posición del emisor externo. De modo que el sistema funcione como un radar acústico pasivo, que sea capaz de extraer la posición de la fuente externa utilizando las medidas en el interior de la cavidad.

Debido a que este proyecto se basa en el estudio de la influencia del número de receptores dentro de la cavidad, es interesante acudir al principio de Huygens para dar una explicación cualitativa en primera aproximación del comportamiento de las acústicas en un problema complicado como es la iluminación de una cavidad a través de una apertura.

Las ondas atraviesan la apertura para poder acceder al interior de la cavidad, por lo que se produce un fenómeno de difracción. La difracción fue investigada por Huygens, posteriormente Fresnel, Kirchhoff y Sommerfeld continuaron su trabajo. Desde un punto de vista más general y en términos más modernos, el teorema de equivalencia explica cómo sustituir una distribución de fuentes dada por unas fuentes equivalentes que ejercen el mismo efecto que las originales.

En este trabajo se emplean ondas acústicas, ya que tienen un comportamiento parecido al de las ondas electromagnéticas en ciertos aspectos. De hecho las ecuaciones de Maxwell se inspiraron en las de las ondas acústicas. De modo que lo desarrollado en este trabajo es aplicable hasta cierto punto también a las ondas electromagnéticas, si bien es necesario incluir fenómenos adicionales como por ejemplo la polarización de las ondas. El motivo por el que se ha trabajado con ondas acústicas es la disponibilidad de la instrumentación necesaria en el laboratorio.

Para entender la propagación de las ondas acústicas es interesante remontarnos al principio de Huygens propuesto en 1680.

1.1.1. Principio de Huygens

El principio de Hyugengs propone un método gráfico que nos permite obtener los frentes de onda sucesivos de una onda que se propaga. Este principio puede enunciarse así:

"Todo punto de un frente de onda inicial puede considerarse como una fuente de ondas esféricas secundarias que se extienden en todas las direcciones con la misma velocidad, frecuencia y longitud de onda que el frente de onda del que proceden."

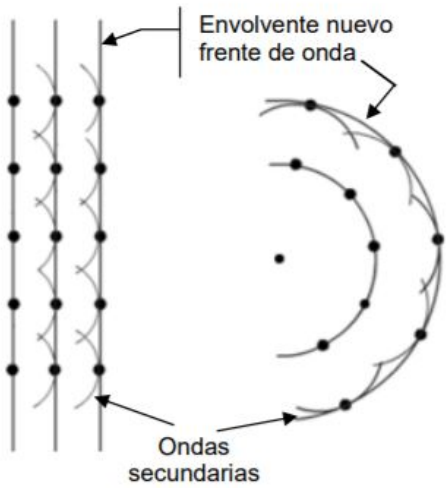

Figura 1.1.: Principio de Huygens

El principio supone que cada punto del frente de ondas primario da origen a una fuente puntual de ondas secundarias que produce ondas esféricas con la misma frecuencia y que se propagan en todas las direcciones con la misma velocidad que la onda primaria.
El nuevo frente de ondas, en un instante dado, es la envolvente de todas las ondas secundarias tal como se muestra en la figura 1.1. En el modelo de Huygens hay que ignorar *ad hoc* las ondas emitidas en sentido contrario al de propagación, por lo que está claro que la teoría de Huygens está incompleta. Fresnel en el año 1826 (por ello se denomina frecuentemente "principio de Huygens-Fresnel"), propuso que las fuentes secundarias en el caso del sonido tenían que tener un efecto debido al desplazamiento y velocidad de las partículas de un elemento de superficie del frente de ondas.

Según Fresnel el frente de onda en sentido contrario al de propagación desaparece porque la superposición de los efectos de las ondas secundarias debe hacerse mediante el principio de interferencia y los efectos de las diferentes ondas elementales se destruyen mutuamente entre sí en este sentido de propagación. Fue Kirchhoff el que presentó una expresión analítica de esta idea de Fresnel. Posteriormente Sommerfeld eliminó ciertas inconsistencias matemáticas en la teoría de Kirchhoff.

1.1.2. Fenómeno de difracción

La difracción es la desviación que sufren los rayos cuando un frente de onda es obstruido por algún obstáculo, como ocurre en este trabajo al incidir la onda acústica sobre la apertura de la cavidad. La distribución espacial de la onda resultante puede calcularse considerando cada punto del frente de la onda original como una fuente puntual de acuerdo con el principio de Huygens, anteriormente citado, y calculando el diagrama de interferencia que resulta de considerar todas las fuentes según la visión de Fresnel del problema.

El principio de Huygens por sí solo, no puede explicar el proceso de difracción. Las ondas de sonido de frecuencias audibles graves se curvan fácilmente alrededor de objetos grandes como los postes de teléfono y los árboles. Por el contrario esos mismos obstáculos forman sombras muy definidas cuando se iluminan con luz. Sin embargo el principio de Huygens es independiente de cualquier consideración de longitud de onda y predecirá las mismas configuraciones de onda en ambas situaciones. Esta dificultad fue resuelta por Fresnel con su adición del concepto de interferencia. El principio de Huygens-Fresnel establece que cada punto sin obstrucción de un frente de onda, en un instante de tiempo dado, sirve como una fuente de ondas secundarias esféricas (de la misma frecuencia de la onda primaria). La amplitud del campo óptico en cualquier punto es la superposición de todas estas ondas considerando sus amplitudes y fases relativas. El límite a la resolución angular impuesto por la difracción se suele aproximar mediante la fórmula:

$$\lambda / D \tag{1.1.1}$$

Donde D es por ejemplo el diámetro de la apertura (en caso de una apertura circular) y el resultado se mide en radianes.

En nuestro caso emitimos a 5 kHz, la velocidad del sonido en condiciones normales es de 343,2 m/s. Conociendo esto puedo hallar la longitud de onda de las onda acústica emitida:

$$\lambda = \frac{v}{f} = \frac{343,2}{5\,10^3} = 0{,}06864\text{m} \tag{1.1.2}$$

La dimensión característica de nuestra apertura es de 0,4m y por tanto mide unas 7 longitudes de onda. El efecto que produce en general la difracción de la onda que incide sobre la apertura es una desviación de los rayos que permite a la energía rodear los obstáculos, como se puede observar en la figura 1.2.

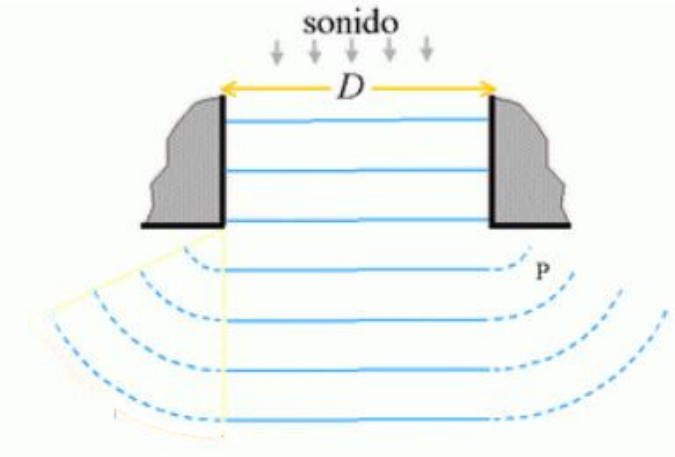

Figura 1.2.: Fenómeno de difracción ondas acústicas.

1.1.3. Región de Fraunhofer

Trabajamos en el entorno de campo cercano, pero para que resulte más sencillo de analizar lo que ocurre en el interior de la cavidad subdividimos la abertura en varias subaberturas lo suficientemente pequeñas respecto a la distancia de los receptores del interior de la cavidad para encontrarnos en campo lejano o región de Fraunhofer. Podríamos decir que se puede aproximar que los rayos que llegan al receptor desde una fuente extensa se pueden aproximar mediante rayos paralelos que parten de fuentes puntuales distribuidas espacialmente para formar la fuente extensa bajo consideración, que en este caso sería cada una de las subaberturas. Todo lo anterior se tiene en cuenta para calcular el comportamiento de las ondas acústicas y así poder entender los datos captados por el micrófono en el interior de la cavidad.

1.1.4. Principio de Equivalencia

Una de las finalidades de este proyecto es conocer la influencia de las ondas que se emiten dentro de la cavidad en el exterior de esta. Se capturan todos los datos posibles del comportamiento de las ondas acústicas en el interior de la cavidad, para poder realizar simulaciones de como se ven las ondas acústicas del interior de la cavidad desde el exterior. Todo ello con el fin de deducir la posición original del emisor que está alimentando desde fuera la cavidad.

Para conocer dicha influencia, acudimos al Principio de Equivalencia el cual nos permiten sustituir, a efectos de calcular los campos en el semiespacio z>0 (en el exterior de la cavidad), los campos en la apertura de la cavidad. De esta forma sustituir las fuente del interior de la cavidad por una línea de fuentes en la posición de la apertura que generen exactamente las misma ondas acústicas.

1.2. Objetivos

El primer objetivo de este trabajo es automatizar la labor de muestreo, ya que la toma de datos para caracterizar la variación espacial de las señales debida al cambio de posición de la fuente externa es muy costosa y repetitiva. Otro objetivo es averiguar qué número de muestras en el interior de la cavidad es suficiente para caracterizar la posición de la fuente externa. Para ello es necesario poder obtener suficientes muestras en el interior de la cavidad para distintas posiciones de la fuente externa. Sería deseable reducir al máximo el número de paradas en el interior de la cavidad a la hora de realizar el muestro y que dicho número retenga una cantidad de información suficiente en comparación con las cuatrocientas muestras empleadas inicialmente, ahorrando de esta forma tiempo en la realización de la toma de medidas. El último objetivo es poder analizar la capacidad de localización de la fuente externa considerando las tandas de decenas de miles de medidas que obtendremos una vez que se automatice el muestreo interno para una fuente externa que pueda cambiar de posición de forma automática, en contraposición con el trabajo preexistente en el que el cambio de posición de la fuente externa era manual.

1.3. Fases

Para poder llevar a cabo todos los objetivos primarios anteriormente planteados y los objetivos secundarios que han aparecido durante el desarrollo de este trabajo, se han llevado acabo diferentes fases:

1. Análisis de los principios teóricos de propagación de ondas, fundamentos de la difracción en ondas acústicas y del Principio de Equivalencia, para de esta forma poder entender lo que ocurre en el interior y exterior de la cavidad.
2. Desarrollo de un control de posición para la fuente externa, que permita ir desplazando la fuente a lo largo de un segmento para realizar un muestreo espacial de un segmento ideal, acelerando enormemente el proceso de toma de medidas.
3. Empleo del control de posición anterior para tomar medidas dentro de la cavidad ubicando el emisor en distintas posiciones, consiguiendo una gran variedad de datos para poder desarrollar el trabajo.
4. Implementación de una red distribuida para poder realizar las simulaciones pertinentes, a partir de los datos obtenidos, de la forma más rápida posible.
5. Realización de simulaciones con distintas posiciones y distintos números de receptores.
6. Obtención de los gráficos de comportamiento en el exterior de la cavidad de las ondas reflejadas en el interior de la misma
7. Comparación del gráfico con el máximo número de receptores con los distintos gráficos obtenidos, variando la posición y el número de receptores.

1.4. Medios empleados

Primeramente decir que todo el proyecto se ha estado documentado en el software emacs. Se han utilizado dos tipos de lenguajes de programación:

- Python: Para el control de parte de la instrumentación y para el modelado de propagación.
- Arduino: Para la programación de los dos arduinos utilizados.

Básicamente tenemos un micrófono que sirve para capturar la intensidad acústica en el interior de la cavidad y que realiza un muestreo espacial al ser desplazado mediante motores paso a paso controlados por un arduino MEGA 2560. Hay otro micrófono externo a la cavidad para poder tomar una referencia. Imaginemos que hay un ruido externo, podría ser confundido con una variación espacial de la señal por el micrófono en el interior de la cavidad, pero gracias al micrófono externo se puede detectar que es un ruido transitorio. También es necesario un analizador dinámico de señal para cuantificar la intensidad acústica captada por estos dos micrófonos. Por último un ordenador con sistema operativo Linux controla el arduino MEGA 2560 y el analizador dinámico de señal.

En la parte de emisor de ondas acústicas se ha utilizado una estructura con una bocina, un generador de ondas conectado a dicha bocina y un arduino UNO. El arduino hace el control del servo de la estructura que permite apuntar en todo momento la bocina hacia el centro de la abertura a pesar de los desplazamientos a los que se ve sometida. Además el arduino MEGA 2560 mencionado antes también controla el desplazamiento espacial de la bocina externa. La fuente que alimenta la bocina también es controlada por el PC principal que controla toda la instrumentación.

Por último, en la parte de simulación, se ha empleado una red distribuida de agentes de cálculo para poder realizar las simulaciones de forma rápida y un ordenador personal con sistema operativo Linux para poder realizar todo el trabajo.

Capítulo 2.

Posicionamiento automático del transmisor

En este capítulo se explicará la mejora introducida en el posicionamiento del transmisor. Se detallarán las diferencias entre la técnica anteriormente utilizada y la actualmente implantada, tanto a nivel de hardware como de software. También se explicará el por qué se decidió realizar esta implantación y qué mejoras ha traído a la investigación.

Contenido

2.1. Hardware implantado

El transmisor anteriormente utilizado sólo constaba de un generador de ondas conectado a través de un cable coaxial a una bocina, lo cual hacía que fuese imposible su movimiento de forma automática. En este proyecto se ha sustituido dicha arquitectura por un estructura de unos 1,5 metros de largo sobre la cual se mueve una base en la cual se encuentra la bocina conectada por cable coaxial al generador de ondas. Dicho generador de ondas se encuentra situado fuera de este entorno para que no entorpezca el movimiento del transmisor. Todo esto hace posible que se mueva de forma automática tanto la bocina a lo largo de la estructura como la orientación de dicha bocina. El hardware desarrollado consta de:

- Arduino UNO: se encarga del control en la orientación de la bocina a través de la comunicación vía radio con un arduino MEGA 2560.
- Carrito: con un servo en el eje de giro de la bocina y una estructura de rodamientos para el movimiento sobre raíles del transmisor.
- Arduino MEGA 2560: se encarga de comunicar al arduino UNO qué orientación debe tomar la bocina para cada una de las posiciones del transmisor, de controlar el motor paso a paso de la estructura para la colocación del transmisor en la posición deseada y del movimiento del micrófono dentro de la cavidad.

2.1.1. Diferencias con la arquitectura anterior del transmisor

Como se puede ver en la imagen 2.1, para mover el transmisor de posición era necesario mover de forma manual la caja sobre la que está situado. Además como se puede ver en la imagen 2.2, era necesario orientar la bocina también de forma manual. Todo esto dificultaba en gran medida el posicionamiento del transmisor ya que no solo era necesario el movimiento manual del transmisor sino también el de la orientación de la bocina. Todo esto hacia que no fuera de gran precisión la posición donde se colocaba y se orientaba, ya que podía existir un error humano. Además al realizarse la toma de datos , solo se podía realizar una toma por noche ya que la posición era fija durante toda esa noche. Debido a todos estos inconvenientes se decidió automatizar tanto el movimiento del transmisor como el movimiento de orientación de la bocina.

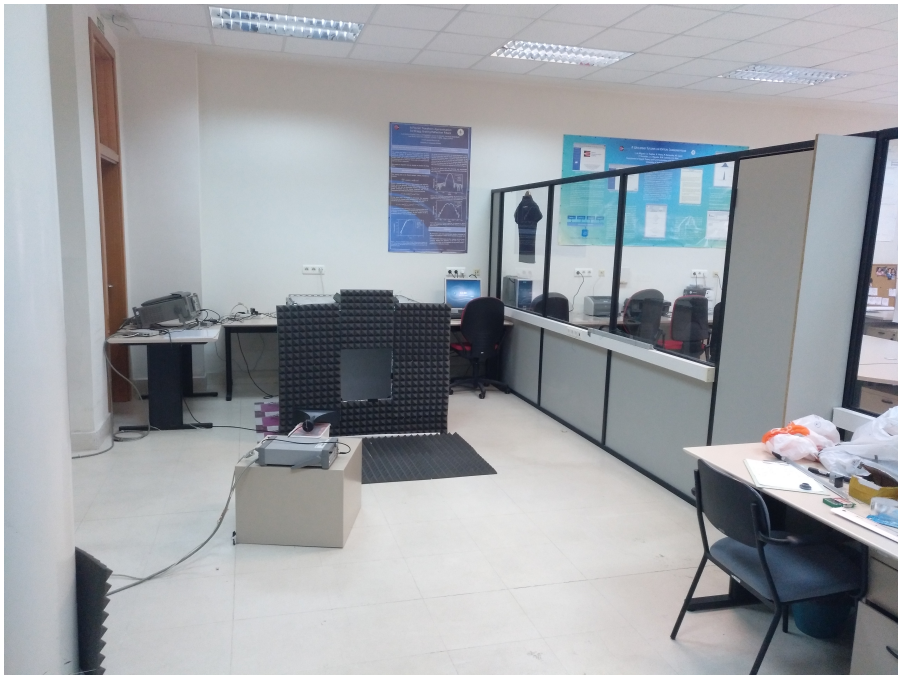

Figura 2.1.: Imagen panorámica del emisor antiguo

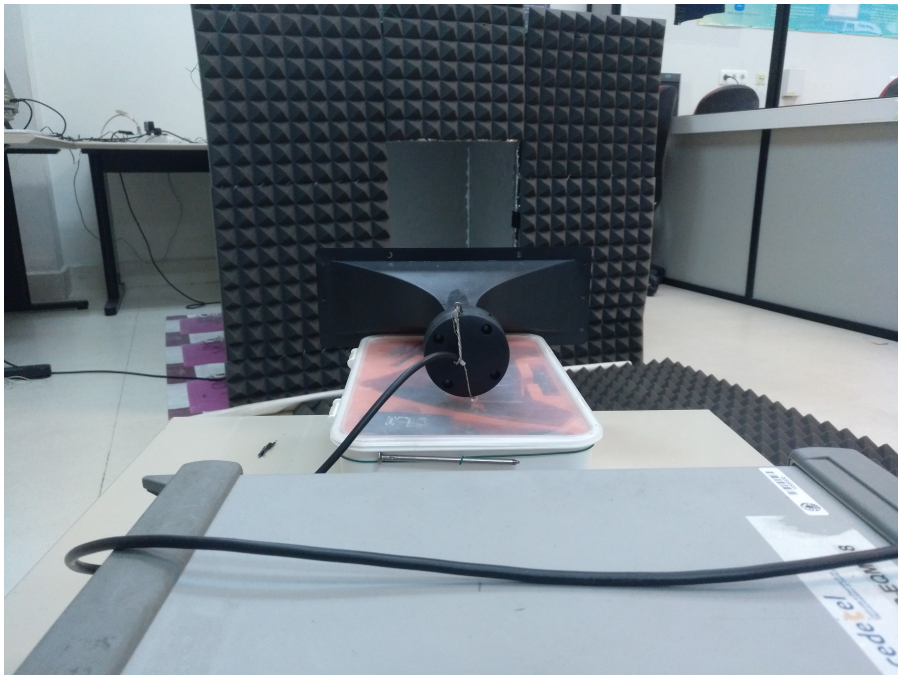

Figura 2.2.: Imagen del sistema de apuntamiento del emisor antiguo

Figura 2.3.: Imagen panorámica del nuevo emisor

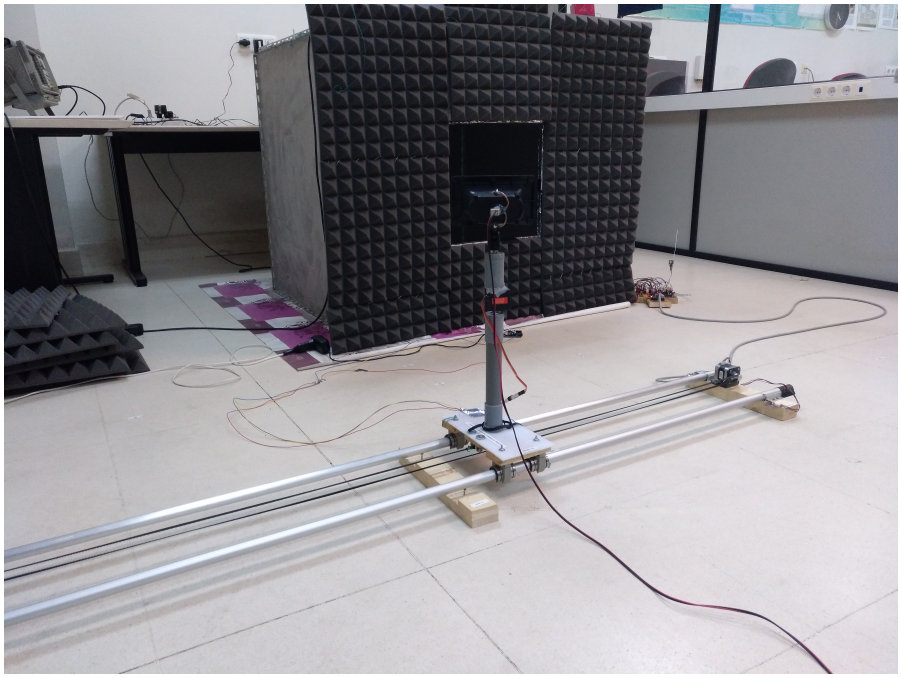

Figura 2.4.: Imagen del nuevo sistema de apuntamiento del emisor

Como se puede ver en la imagen 2.3, el movimiento del transmisor es mucho mas eficiente ya que es controlado remotamente. Además la orientación para cada posición del transmisor es exacta ya que se desarrolló un algoritmo para la orientación perfecta de la bocina dependiendo de la posición en la que se encontrara el transmisor.

Como se puede contrastar en las figuras anteriores, hemos pasado de tener que mover manualmente la posición y la orientación del transmisor, a que se realice de forma automática a través del control de un servo, un motor paso a paso y a través de la comunicación vía radio entre dos arduinos. Todo esto nos permitió poder realizar en cada noche la toma de datos de varias posiciones distintas del transmisor, permitiendo realizar un análisis comparado del efecto del cambio de posición basado en un gran número de posiciones externas.

2.1.2. Estructura del carro de medidas

La principal estructura sobre la que se basa el robot tiene una longitud de 1 metro y está constituida por dos tubos de aluminio que actúan a modo de raíles y sobre las cuales se soporta y se mueve el transmisor utilizando rodamientos.

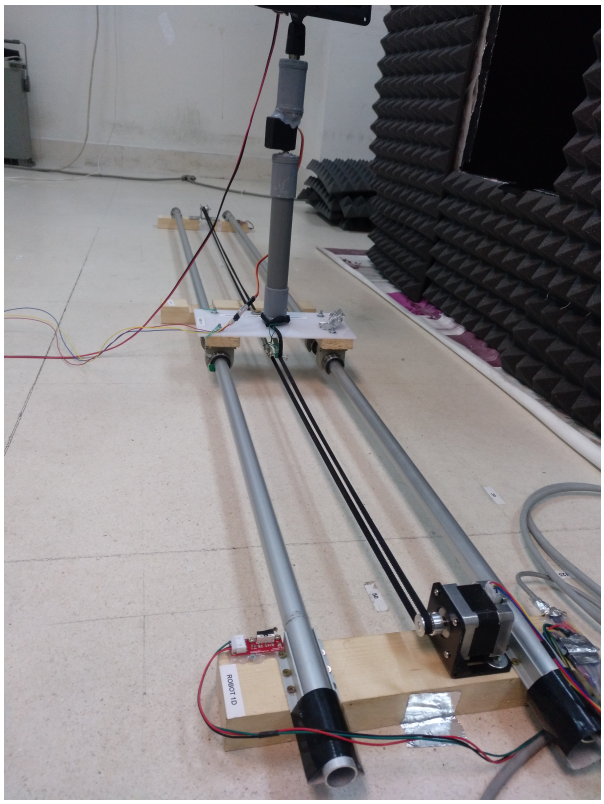

Figura 2.5.: Estructura sobre la que se transporta el transmisor

En los extremos de los raíles se encuentran dos finales de carrera basados en contacto por presión, como se puede ver en la figura 2.6, para detectar el momento en el que la base de la bocina llega al final de cualquiera de los dos extremos de la estructura.

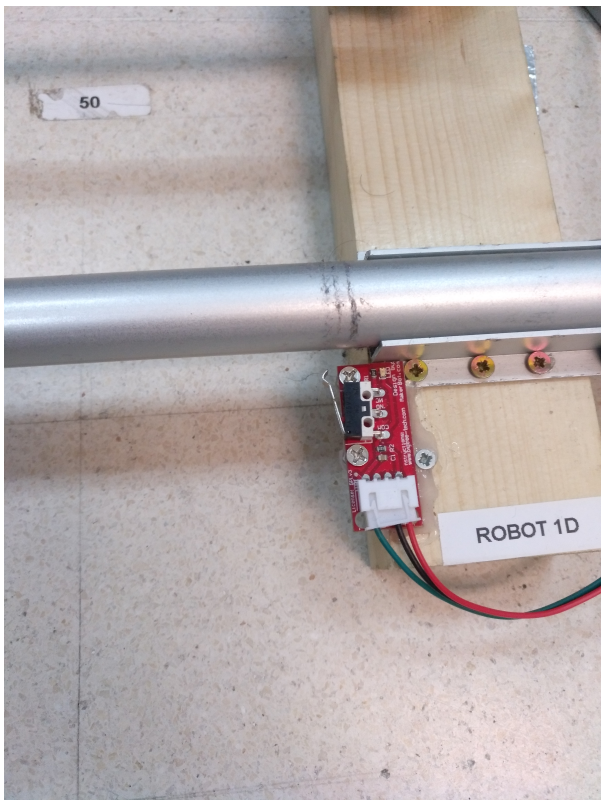

Figura 2.6.: Imagen del final de carrera utilizado

En uno de los dos lados del carrito hay situado un motor paso a paso (el cual está controlado por el arduino MEGA2560) , como se puede ver en la figura 2.7. El cable que aparece en la figura 2.8 corresponde al final de carrera del extremo opuesto al motor. Como se puede ver en la imagen 2.9, el motor paso a paso y el final de la estructura están conectados a través de una correa de transmisión, fijada en la parte inferior de la base del carrito permitiendo el movimiento de nuestra bocina sobre los raíles a lo largo de toda esta estructura. El motor paso a paso recibe la información pertinente del arduino MEGA2560, para saber cuantos milímetros tiene que avanzar en cada una de las paradas para ir realizando el desplazamiento externo de la fuente acústica.

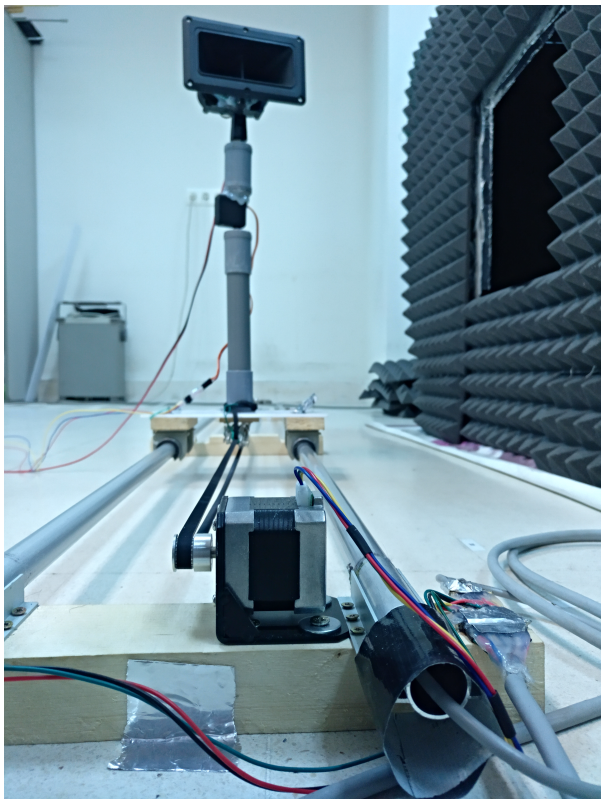

Figura 2.7.: Situación del motor paso a paso que controla el movimiento de la base del transmisor

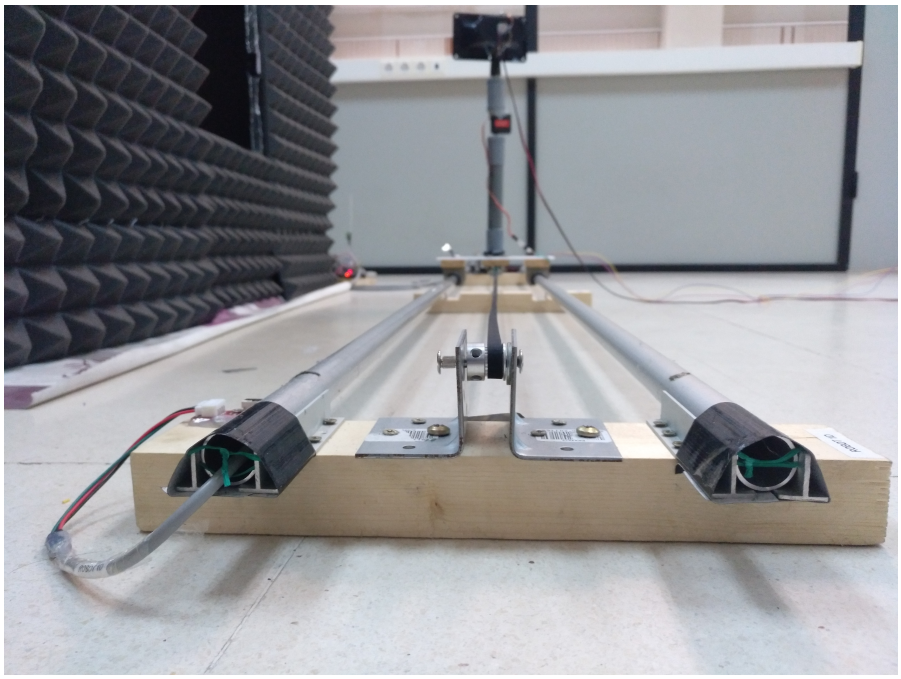

Figura 2.8.: Extremo final de la correa de transmisión

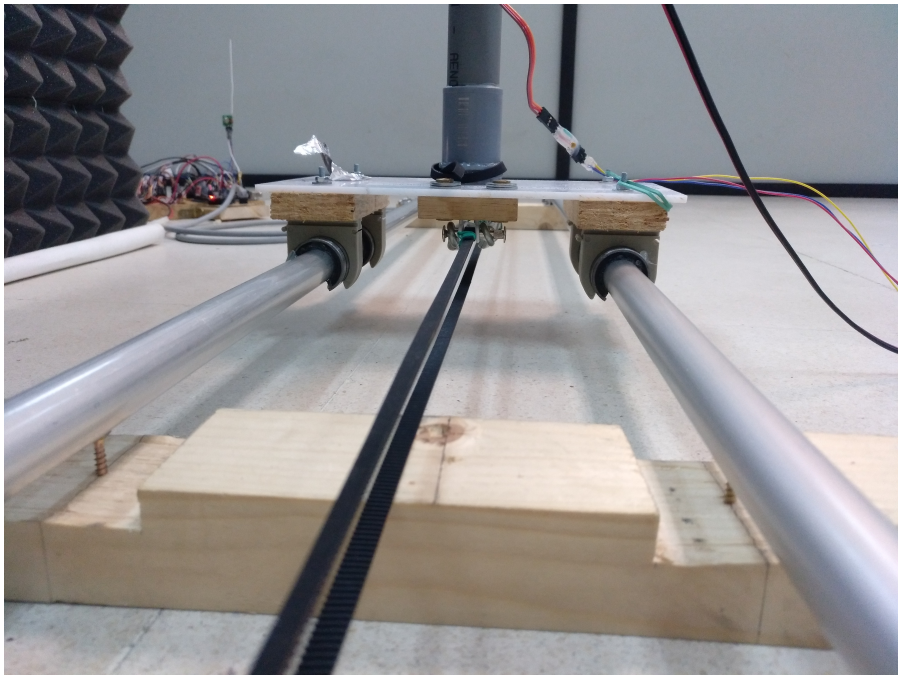

Figura 2.9.: Correa de transmisión reforzada con fibra de vidrio que permite el movimiento de la base a través de toda la estructura

2.1.3. Base de la bocina

Como se puede ver en la figura 2.10, la base de la bocina consta de un rectángulo de 12.5 cm de ancho y 24 cm de largo, en cuyo centro se sitúa un tubo. Entre dos secciones de tubo se encuentra el servo encargado de la orientación de la bocina. Dicha bocina se sitúa al final del tubo haciendo que su altura coincida con la del micrófono situado dentro de la cavidad. De esta forma se consigue que los datos recogidos por el micrófono dentro de la cavidad sean de aún mas interés.

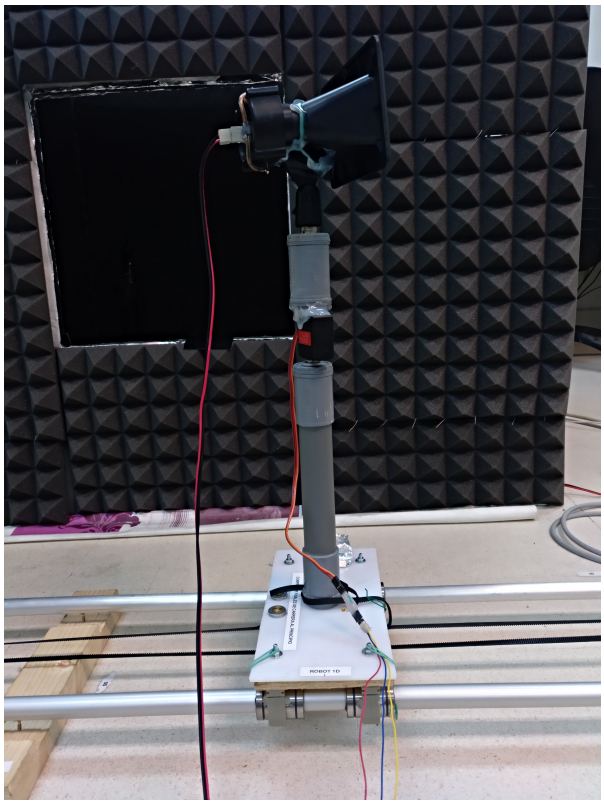

Figura 2.10.: Estructura sobre la que se sitúa la bocina controlada por el servo

Tanto el micrófono dentro de la cavidad como la bocina, se encuentran a una altura de 54,5 cm.

Se busca en todo momento que la bocina esté apuntando al centro de la cavidad en cada uno de los movimietnos del carrito para que la señal transmitida no tenga variaciones debidas al diagrama de radiación de la fuente. Para conseguir esto, el PC de control realiza los cálculos y luego transmite al MEGA 2560 el ángulo de giro del servo para cada una de las posiciones. El arduino MEGA 2560 le envía el ángulo en el que se debe encontrar la bocina al arduino UNO vía radio, y el arduino UNO se lo comunica al servo situado en la base.

2.1.4. Arduino UNO

El arduino UNO está conectado por un cable de cobre al servo para controlar el ángulo de giro de la bocina, los otros dos cables proporcionan alimentación al servo. El arduino Uno también está conectado vía radio con el arduino MEGA 2560 para conocer el valor del ángulo. La comunicación con el servo se realiza por bus SPI (*Serial Peripheral Interface*), y la señal que se le envía es una señal modulada por ancho de pulso. En resumen, el servo tiene tres cables, uno de alimentación a 3.3 Voltios, uno de referencia (GND) y el último de datos. Tanto el de alimentación como el de referencia se conectaron al Arduino UNO a los pines de 3,3 Voltios y GND respectivamente. El de datos se conecto al pin 9 de PWM (*Pulse-Width Modulation*).

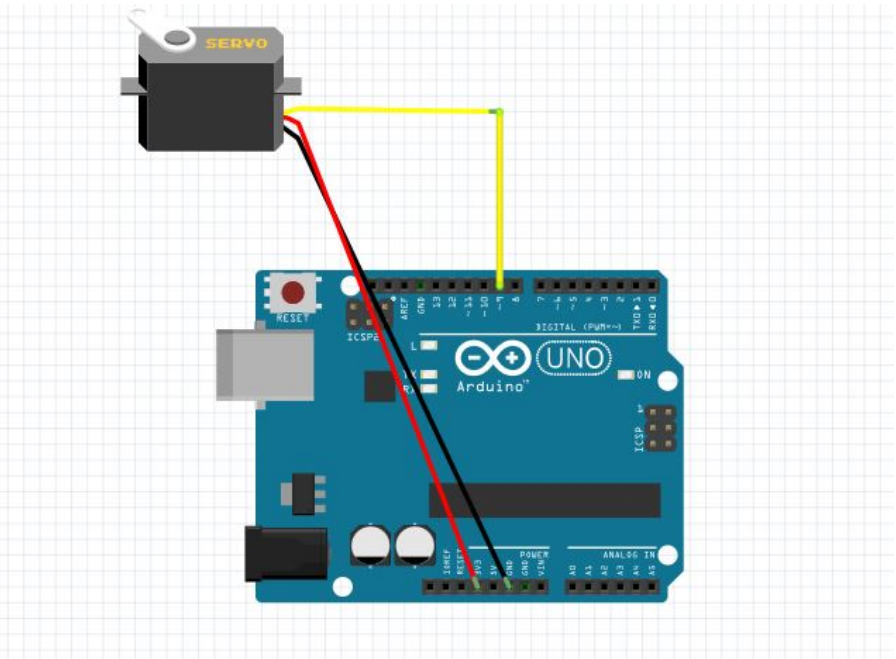

Figura 2.11.: Esquema de la conexión entre el arduino UNO y el servo

A través del pin de datos se envía la señal modulada por ancho de pulso. Es decir, según el ciclo de trabajo que tenga la señal mandada por el arduino UNO, el servo girará un tanto por ciento de la capacidad de giro que tiene. Teniendo 2500 microsegundos de ciclo de trabajo 180 grados y 500 microsegundos 0 grados. Hemos colocado el servo de tal manera que cuando reciba un ciclo de trabajo de duración 500 microsegundos (cero grados) la bocina tenga una orientación perpendicular a la estructura del carro y mirando a la derecha de esta. Todo esto se ha realizado con un mapeo en el software del arduino UNO.

A la hora de la conexión del servo con el arduino UNO, surgió un problema con el movimiento de orientación del servo. En reposo rebotaba unos pocos milímetros debido al ruido que se encontraba en su alimentación. Para combatir este ruido se desarrolló un filtro paso bajo constituido por un condensador de 10nF y la propia resistencia del cable de conexión. De esta forma se consiguió que la banda de trabajo del filtro dejara pasar la señal de alimentación totalmente limpia y no existiera ningún rizado en dicha señal.

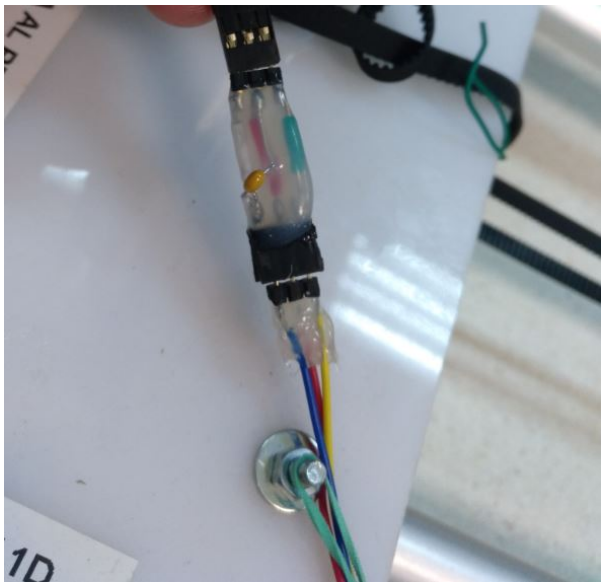

Figura 2.12.: Implementación filtro paso bajo para la eliminación del ruido

El cable rojo corresponde a la alimentación 3.3 V del servo, el cable blanco a referencia del arduino y el cable verde a la señal de datos PWM.

La comunicación radio con el arduino MEGA 2560 se realiza a través del módulo de arduino NRF24L01.

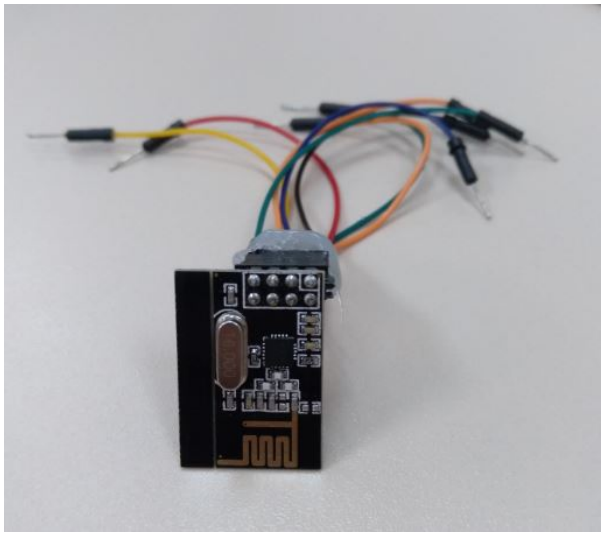

Figura 2.13.: Módulo NRF2401 utilizado para la comunicación por radio frecuencia

El módulo NRF24L01 es un chip de comunicación inalámbrica fabricado por Nordic Semiconductor que integra un transceptor de radio frecuencia (transmisor + receptor) a una frecuencia entre 2.4GHz a 2.5GHz, una banda libre para uso gratuito. El NRF24L01 también incorpora la lógica necesaria para que la comunicación sea robusta, como corrección de errores y reenvío de datos si es necesario, liberando de esta tarea al procesador.

Esto es de gran importancia por si surge el caso en el que el arduino UNO no recibe correctamente la información, poder saber si ha habido un fallo en el envío de datos y reenviarle dicha información. El control del módulo lo realizamos a través de un bus SPI. En la figura 2.14 podemos ver el esquema de conexión de dicho módulo con el arduino.

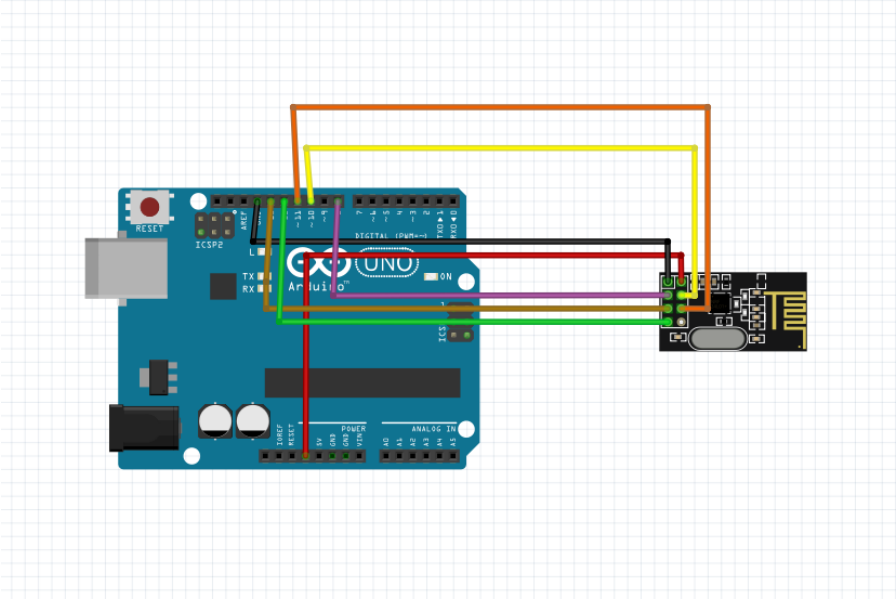

Figura 2.14.: Esquema hardware de la conexión del arduino UNO con el módulo NRF2401

Con la conexión del servo y del módulo de radio frecuencia nos queda el esquema electrónico de la figura 2.15

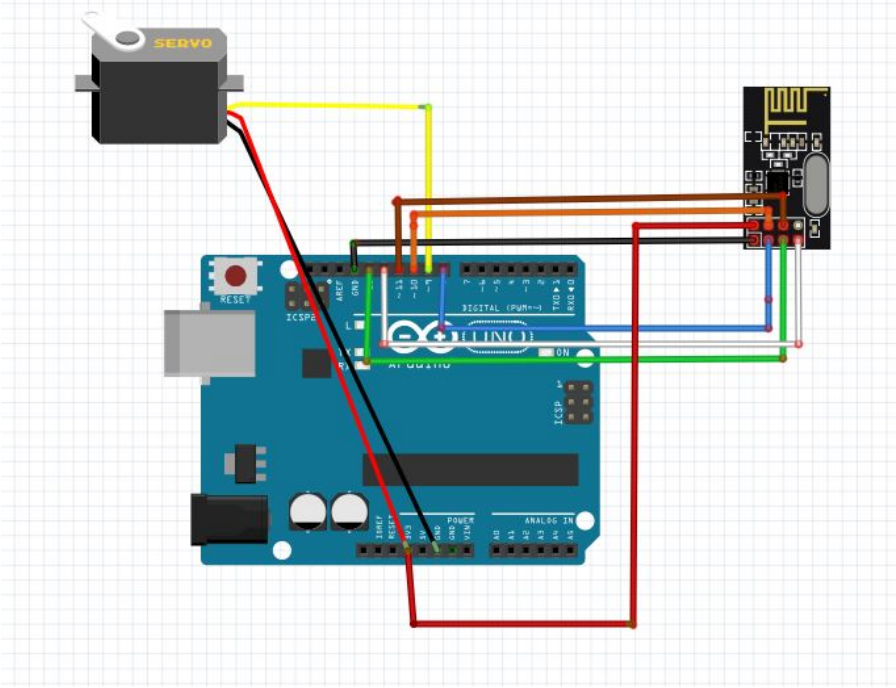

Figura 2.15.: Esquema hardware de toda la conexión del arduino UNO

2.1.5. Arduino MEGA 2560

El arduino MEGA 2560 al poseer mayor capacidad de procesado que el arduino UNO se encarga de realizar todos los cálculos para el giro del servo. También se encarga de enviar la señal PWM al motor paso a paso de la estructura y a los motores paso a paso del interior de la cavidad para que el micrófono del interior siga las pautas requeridas. Además se encarga de la toma de la humedad y de la temperatura para cada una de las muestras a través de un sensor de temperatura y otro de humedad. Para poder comunicarse de forma dúplex con el arduino UNO es necesario otro módulo de radio frecuencia NRF2401 conectado al arduino MEGA2560.

En este arduino se tuvo problemas a la hora de buscar patillas disponibles para su conexión ya que la mayoría estaban ocupadas por el sensor de humedad, el sensor de temperatura y los motores paso a paso. Finalmente se consiguió encontrar una solución y se obtuvo el esquema hardware de la conexión con el módulo de radio frecuencia de la figura 2.16.

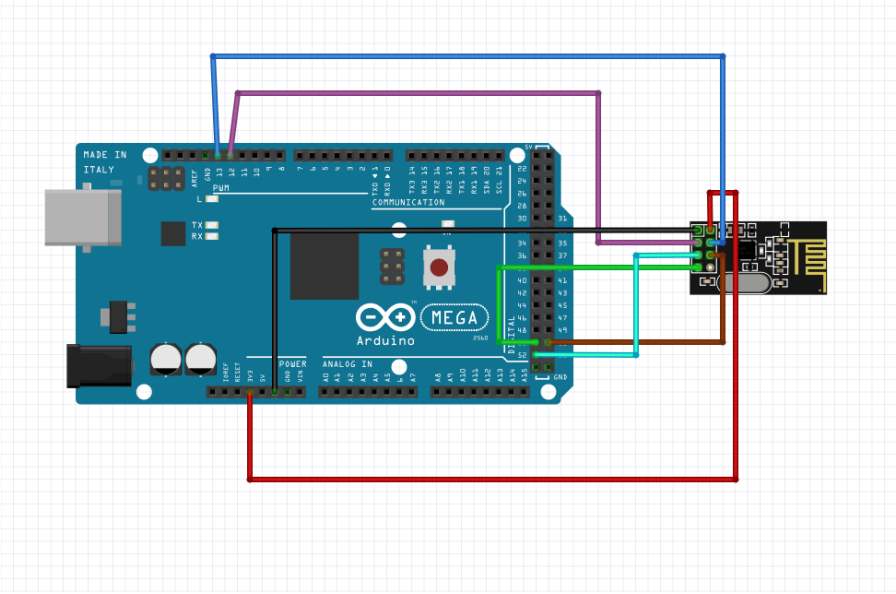

Figura 2.16.: Esquema hardware de toda la conexión del arduino MEGA2560 con el módulo NRF2401

También se encarga del control del movimiento del motor paso a paso de la estructura. Nuestro motor paso a paso es bipolar, es decir, que nos permite avanzar y retrasar la base de la bocina. Un motor paso a paso es un dispositivo electromagnético que convierte impulsos eléctricos en movimientos mecánicos de rotación. La principal característica de estos motores es que se mueven un paso por cada impulso que reciben. Nuestro motor paso a paso avanza dos grados en cada impulso. El motor paso a paso está formado por dos partes:

- El estator: es la parte fija del motor donde sus cavidades van depositadas las bobinas.
- El rotor: es la parte móvil del motor construido por un imán permanente.

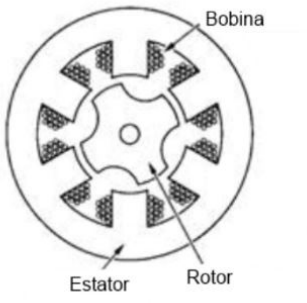

Figura 2.17.: Partes de un motor paso a paso

Estas dos partes van montadas sobre un eje.

Figura 2.18.: Motor paso a paso y su interior

El funcionamiento del motor paso a paso para obtener el movimiento de la base de la bocina se basa en que cuando circula corriente por una o más bobinas del estator se crea un campo magnético creando los polos Norte-Sur. Luego el rotor se equilibrará magnéticamente orientando sus polos Norte-Sur hacia los polos Sur-Norte del estator. Cuando el estator vuelva a cambiar la orientación de sus polos a través de un nuevo impulso recibido hacia sus bobinas, el rotor volverá a moverse para equilibrarse magnéticamente. Si se mantiene esta situación, obtendremos un movimiento giratorio permanente del eje. El ángulo de paso depende de la relación entre el nombre de polos magnéticos del estator y el nombre de polos magnéticos del rotor.

2.2. Principio de funcionamiento

Como ya hemos comentado en la sección 2.1.1, en las investigaciones anteriores se utilizaba un generador de ondas directamente conectado a la bocina y con posición fija. Solo se podía obtener medidas en el interior de la cavidad para una única posición de la fuente externa por noche dado que no se podía mover la posición de dicha fuente.

Para poder conseguir un número suficientemente significativo de datos decidimos desarrollar un robot de medida que desplazara la bocina a lo largo de unos raíles de longitud en torno al 1,5 m. El principio de funcionamiento de la nueva implementación, como ya hemos comentado a lo largo de todas las subsecciones anteriores, se basa en la conexión por radio frecuencia entre el arduino UNO y el arduino MEGA 2560, la conexión por cable del arduino UNO con el servo de la bocina y la conexión también

por cable del arduino MEGA 2560 con el motor paso a paso de la estructura y de los motores paso a paso del interior de la caja.

Primeramente se coloca la estructura a una determinada distancia del cajón sobre el que se encuentra el generador. Se configura cuántos milímetros queremos que avance la base del transmisor sobre la estructura (nosotros configuramos que avanzara 7mm en cada paso) y cuántas paradas queremos que haga tanto el motor paso a paso de la estructura como los motores paso a paso en el interior de la caja. No interesa que haga paradas innecesarias porque solo podemos poner a trabajar el robot en torno a unas 10 horas en un día laboral (duración entre finalización del horario laboral a las 10:00 PM y comienzo de este a las 8:00 AM). Cuando ya tenemos todo correctamente configurado comienza la etapa de toma de medidas.

En esta etapa lo primero que ocurre es que el PC de control calcula la posición que debe alcanzar el transmisor para esa toma de medidas y se la envía al arduino MEGA 2560. El arduino MEGA 2560 la comunica a su vez al motor paso a paso de la base de estructura a través de una señal PWM. Después, el arduino MEGA 2560 envía la orientación que debe de tomar la bocina para dicha posición. Finalmente el arduino MEGA 2560 transmite los datos vía radio al arduino UNO. El cálculo del ángulo de giro del servo se comentará en la sección de desarrollo software.

Si el arduino UNO ha recibido la señal correctamente, responde con una señal ACK para que su par sepa que la información se ha recibido correctamente. Si se ha recibido mal o si el arduino UNO no recibe ninguna información, no se produce envío de la señal ACK. Por su parte, el arduino MEGA espera unos milisegundos la recepción de la señal ACK, si no la recibe reenvía la información hasta que se reciba la señal ACK enviada por el arduino UNO. Toda la generación de estas señales se comentará también en la sección de desarrollo software. Cuando recibe dicho ACK, el arduino MEGA comienza la toma de medidas dentro de la cavidad moviendo el micrófono en el interior de la cavidad tal y como se haya configurado para esa toma de medidas. Una vez haya terminado la toma de medidas, el arduino MEGA empuja los milímetros pertinentes el transmisor sobre la estructura y comienza la misma secuencia otra vez hasta que se llegue a la posición final para la bocina.

En todo este proceso se busca que la bocina del transmisor apunte siempre al centro de la abertura, para que la señal entre con la mayor potencia posible para cada una de las posiciones y no haya variaciones debidas al diagrama de radiación de la bocina.

Aquí mostramos una secuencia de 5 figuras con distintas posiciones y la pertinente orientación de la bocina para cada una de ellas.

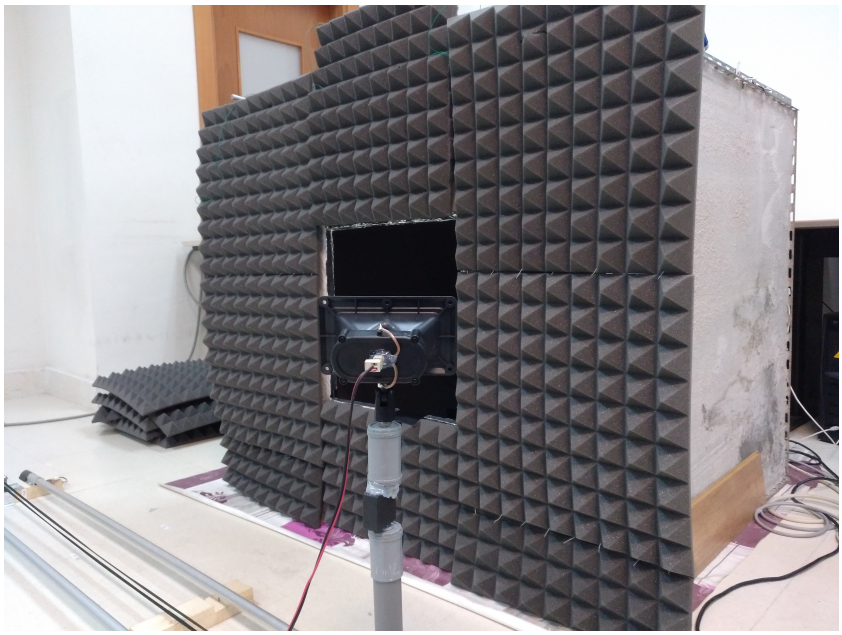

Figura 2.19.: Posición 1 de la secuencia de imágenes

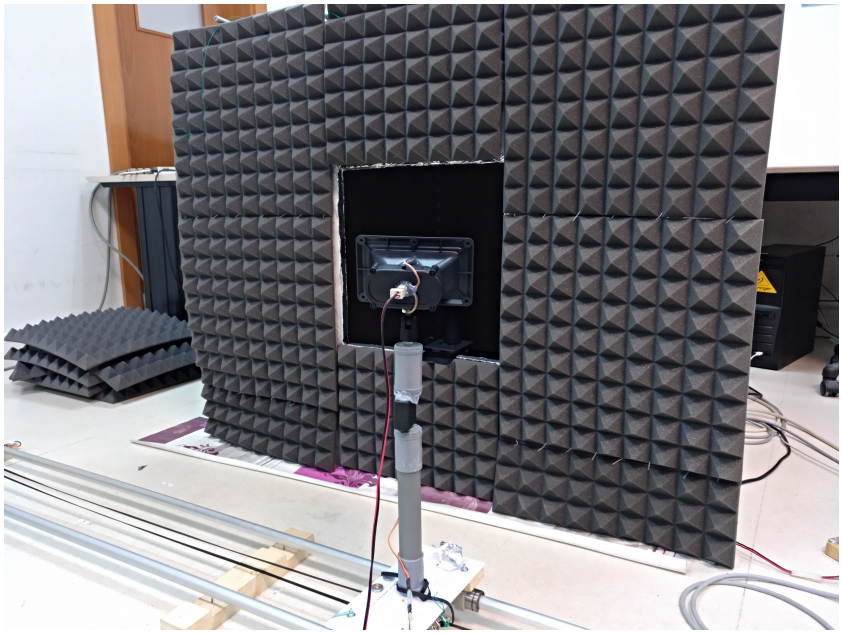

Figura 2.20.: Posición 2 de la secuencia de imágenes

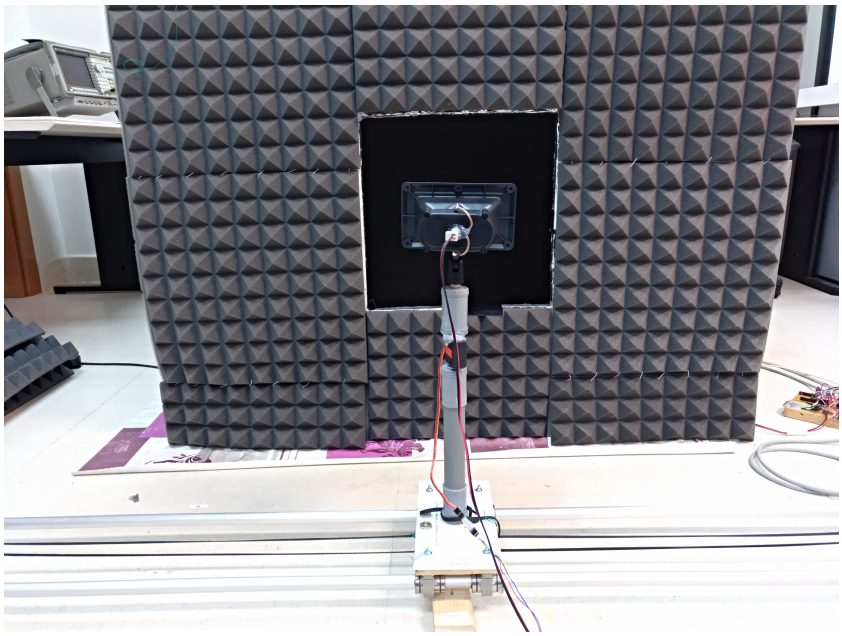

Figura 2.21.: Posición 3 de la secuencia de imágenes

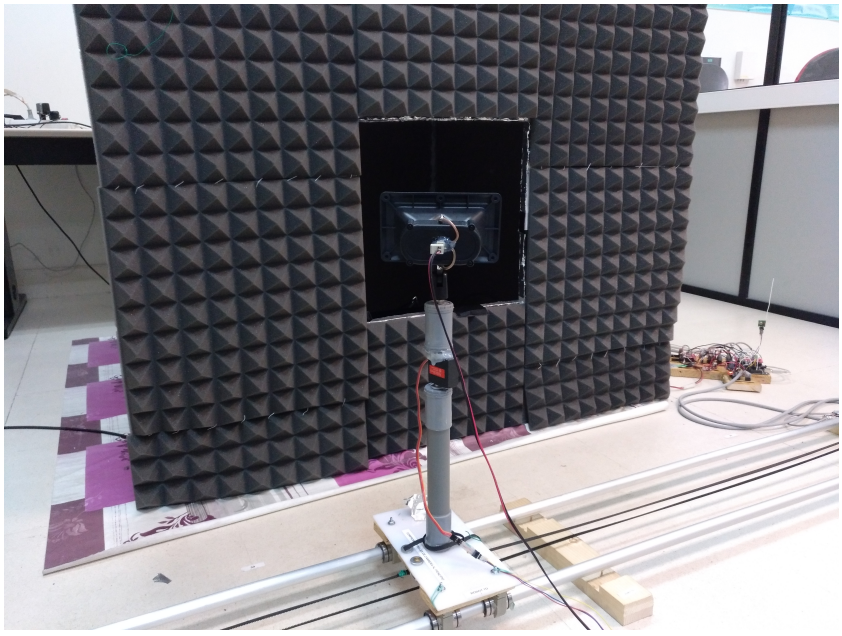

Figura 2.22.: Posición 4 de la secuencia de imágenes

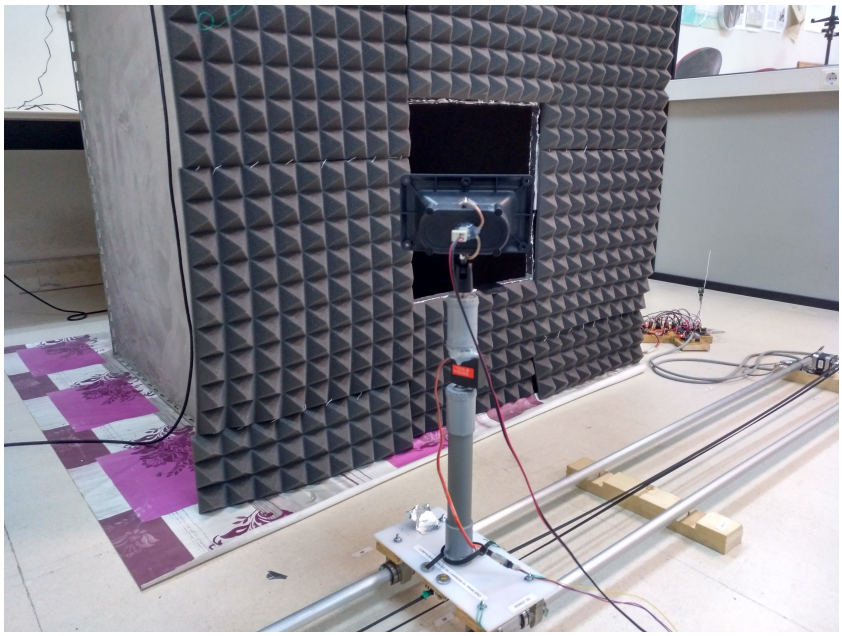

Figura 2.23.: Posición 5 de la secuencia de imágenes

2.3. Software desarrollado

En esta sección voy a explicar el software desarrollado explícitamente para el transmisor automático. El código en su totalidad está incluido en el Anexo.
El software desarrollado para la automatización del transmisor se ha realizado en arduino, ya que era mucho mas fácil de implementar en este entorno. A pesar de implementar la comunicación entre los arduinos, la comunicación arduino-servo y arduino-motor paso a paso en el entorno de arduino, el lenguaje principal en el que se basa el proyecto a la hora de toma de las medidas, obtención de las gráficas pertinentes a dichas medidas y la ejecución del código fuente, ha sido el lenguaje python.

2.3.1. Arduino UNO

En esta subsección voy a explicar algunas partes importantes del código desarrollado para el arduino UNO, encargado del uso del servo de la bocina y del módulo de radio frecuencia.
Lo primero que hacemos es escoger los dos canales que vamos a usar como receptor y como emisor:

```
const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };
```

En la función de setup comenzamos la comunicación radio y establecemos el número de reintentos a la hora de inicializar la conexión:

```
radio.begin();
radio.setRetries(15,15);
```

Abrimos el canal de emisión en el canal configurado anteriormente:

```
radio.openWritingPipe(pipes[1]);
```

y el canal de recepción:

```
radio.openReadingPipe(1,pipes[0]);
```

Comienza el bucle de ejecución del programa, al ser arduino es un bucle infinito. Si hay datos disponibles en el canal de recepción los grabamos en la variable ServoCharMSG. Nótese que debemos especificar el tamaño del mensaje esperado:

```
if ( radio.available() )
 {
radio.read(&ServoCharMsg, sizeof(ServoCharMsg));
```

Tras recibir un mensaje, dejamos de escuchar el canal, y pasamos a ser emisores:

```
radio.stopListening();
```

Como hemos leído los datos enviados por el arduino MEGA2560, respondemos con un ACK escribiendo en el canal de emisión:

```
radio.write(&ack, sizeof(ack) );
```

Convertimos los datos recibidos en un integer para establecer el ángulo de giro:

```
 pos = atol(ServoCharMsg);
```

Evaluamos los dos posibles valores de extremos que puede tomar el ángulo para evitar usar valores que no tienen sentido:

```
if(pos < 0) pos = 0;
if(pos > 180*1000000) pos = 180;
```

Ahora nos quedarían los ángulos intermedios. Realizamos un mapeo del valor recibido, convirtiendo el menor valor que puede tomar la variable pos (0) en 500 (duración del pulso en microsegundos) y convirtiendo el valor máximo (180 grados) en 2500 microsegundos de duración del pulso que se le envía al servo. Se emplea una interpolación lineal para los ángulos intermedios:

```
pos = map(pos, 0, 180*1000000, 500, 2500);
```

Se le envía dicha señal PWM al servo y se obtiene la orientación deseada, es decir, que la bocina apunte al centro de la abertura de la cavidad:

```
Serial.print(pos);
myservo.attach(9);
myservo.write(pos);
```

2.3.2. Arduino MEGA2560

En esta subsección voy a explicar algunas partes importantes del código desarrollado para el arduino MEGA2560, el cual se encarga del control del motor paso a paso de la estructura y del envío por radio frecuencia del ángulo de apuntamiento de la bocina.

Como con el arduino UNO, lo primero que hacemos es escoger los dos canales que vamos a usar como receptor y como emisor. Van a ser los recíprocos que en el arduino UNO para que donde emita uno el otro escuche y viceversa:

```
const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };
```

En su función setup también comenzamos la comunicación radio y establecemos el número de reintentos a la hora de inicializar la conexión:

```
radio.begin();
radio.setRetries(15,15);
```

Abrimos el canal de emisión en el canal configurado anteriormente:

```
radio.openWritingPipe(pipes[0]);
```

y el canal de recepción:

```
radio.openReadingPipe(1,pipes[1]);
```

En este bucle infinito se va a ejecutar la llamada a una función que se encarga de realizar la comunicación serie entre el arduino MEGA2560 y el PC de control, que al final es el que sincroniza a todos los equipos:

```
sCmd.readSerial();
```

En este trabajo fin de grado se ha desarrollado la función de envío de datos por radio frecuencia. Se pasa como argumento de la función un puntero al mensaje de datos que queremos enviar al arduino UNO:

```
void send_rf (char *message)
```

A continuación se realiza una comparación para saber si el arduino UNO envió su asentimiento y se recibió bien. Si este asentimiento no se ha recibido bien, se repetiría el bucle while indefinidamente y se volverían a enviar constantemente los datos, ya que se han perdido en el canal o el arduino UNO no los recibió correctamente:

```
while (ok==false)
```

Envíamos los datos. Una vez los enviamos, pasamos a escucha para recibir el asentimiento:

```
radio.write(message, sizeof(String));
radio.startListening();
```

Si hay datos disponibles en el canal, nos disponemos a leerlos. Como tiene que ser el asentimiento del arduino que habíamos definido en el código del arduino UNO como una variable string que contenía la cadena de caracteres ACK, comparamos el mensaje recibido con dicha cadena para saber si hemos recibido correctamente el asentimiento

```
if ( radio.available() )//If data are available
        {
            char recibido[3] = " ";
            radio.read(&recibido, sizeof(recibido));
            if (strstr(recibido, "ACK")) ok=true;
            else ok=false;
        }
```

Adicionalmente se ha realizado el código para el motor paso a paso que mueve la fuente acústica. Este corresponde con la función:

```
void processMotorE()
```

El código siguiente es para procesar los argumentos de la orden en formato ascii:

```
arg = sCmd.next();
```

Si el argumento no está vacío, entonces se calcula el número de pasos que debe de dar el motor. La función atol convierte una variable string correspondiente al mensaje en ascii en un integer.

```
steps=atol(arg);
```

Evaluamos que no sea un valor negativo, porque si no, no nos valdría:

```
if (steps<=0){
```

Posteriormente se evalúa si los pasos se van a dar hacia delante o hacia atrás. El motor solo puede girar en esos dos sentidos:

```
  if ((forward!=0)&&(forward!=1)){
```

Finalmente tras descartar valores no válidos, se envía al motor la orden de avanzar los pasos calculados anteriormente:

```
moveMotorE(steps,(forward==1));
```

2.4. Ventajas obtenidas

Con la nueva implementación se ha ganado mucho tiempo a la hora de la toma de medidas. Con el mecanismo antiguo, solo se podían realizar las medidas correspondientes a una posición de la fuente externa por noche. De esta forma hemos podido realizar en un día laboral la toma de medidas correspondiente a hasta 7 posiciones distintas de la fuente externa en una sola noche. En un fin de semana hemos podido realizar a la largo de todos los días disponibles, la toma de medidas de toda las posiciones de la estructura usando una diferencia de 7mm entre posiciones.

De esta forma ha sido factible pasar a la etapa de análisis de resultados, ya que el cambio de posición de la fuente hecho a mano hubiera requerido mucho más tiempo del disponible para la toma de medidas.

Capítulo 3.

Medidas realizadas

En este capítulo se explicará el procedimiento llevado a cabo para la toma de medidas, el por qué se decidió realizar así cada una de ellas y las hojas de datos obtenidas como resultado del muestreo de las señales acústicas.

Contenido

3.1. Toma de medidas

Para saber si es posible recuperar la posición de la fuente externa basándonos en las medidas de intensidad acústica tomadas con un micrófono en el interior de la cavidad, es necesario contar con un número importante de experimentos que pongan de relieve la influencia de la posición de la fuente externa en la distribución de la intensidad dentro de la cavidad. Cuando la luz del sol atraviesa una ventana, la posición de las zonas más intensamente iluminadas permite deducir la posición del sol respecto a la ventana. La idea es intentar hacer algo análogo con la intensidad que atraviesa la apertura de nuestra cavidad. Sin embargo, al estar próximos al régimen de baja frecuencia la presencia de modos dificulta extraer dicha información. Debemos tener en cuenta que los modos presentan poca dependencia respecto a la posición de la fuente externa. Por tanto es necesario realizar un muestreo de distintas posiciones del emisor, para poder recoger el mayor número de datos posibles y así tratar de averiguar qué cambia dentro de la cavidad al ir cambiando la posición de la fuente. Si únicamente tenemos unas pocas posiciones de la fuente externa, es prácticamente imposible comprender el fenómeno dada la enorme complejidad del mismo. El generador de funciones se configuró para emitir una frecuencia de 5kHzy así encontrarnos en la región de frecuencia intermedia y de esta forma obtener una longitud de onda lo suficientemente pequeña para poder aumentar los grados de libertad de la distribución espacial del campo dentro de la cavidad. Tampoco se podría emitir con una longitud de onda muy pequeña, ya que el tamaño del micrófono sería muy grande en comparación con esta y no se podría obtener mucha resolución espacial.

La toma de medidas se realizó de dos maneras:

- Muestreo denso: el micrófono realizaba una parada cada 3 milímetros.
- Muestreo ligero: el micrófono realizaba una parada cada 1 centímetro.

El objetivo de la realización del muestreo denso era poder determinar con detalle la distribución espacial de la intensidad acústica en el interior de la cavidad. Este muestreo tenía una duración de 60 horas. Una vez que obtuvimos varias muestras densas, realizamos para esas mismas posiciones de la fuente externa un muestreo ligero (se realiza entorno a 5 horas y media). Se observó que los decibelios captados por el micrófono no se diferenciaban en más de 2dB, observando el ahorro de tiempo que se obtenía se decidió realizar diariamente un muestreo de 1x1 centímetro.

El muestreo denso pasó solo a realizarse cuando se pudiera realizar la toma de muestras durante varios días y en una posición en la que se encontrara algún patrón interesante.

3.2. Colocación del transmisor

Se realizaron tomas de medidas con 3 tipos distintos de colocación del emisor. Referencia de coordenadas de la cavidad:

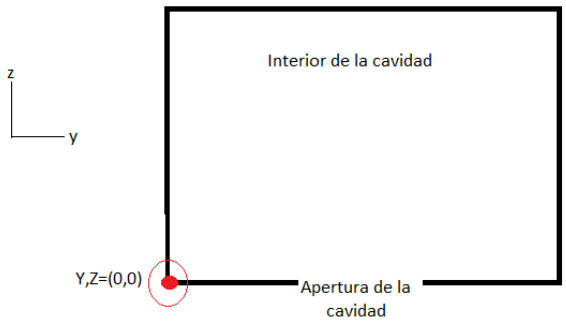

Figura 3.1.: Referencia de coordenadas de la cavidad

3.2.1. Arco radial

El objetivo de la colocación en forma radial de los raíles sobre los que se desplaza la fuente es poder ver el comportamiento a grandes rasgos de las ondas acústicas y de esta forma seleccionar sitios de interés para el transmisor. Se comenzó a realizar la medición en el punto y= -0.245m , z=-1.137 y se terminó en la posiciones y=0, z=-1.273, realizando 5 paradas entre estos dos puntos. En las figuras 3.2 y 3.3 se puede ver una secuencia de colocación de esta posición.

Figura 3.2.: Posición 1 del transmisor en toma de pruebas con arco radial

Figura 3.3.: Posición 2 del transmisor en toma de pruebas con arco radial

3.2.2. Fuente acercándose hacia la apertura de la cavidad

Se colocó el carrito de tal forma que en cada paso del emisor se acercara cada vez más a la apertura de la cavidad. De esta manera la bocina está constantemente apuntando a la esquina de la cavidad ya que cuando se realizaron simulaciones en estas posiciones, se observó que en esta esquina existía un punto de singularidad. En las figuras 3.4 y 3.5 se puede ver un ejemplo de esta forma de colocar la fuente.

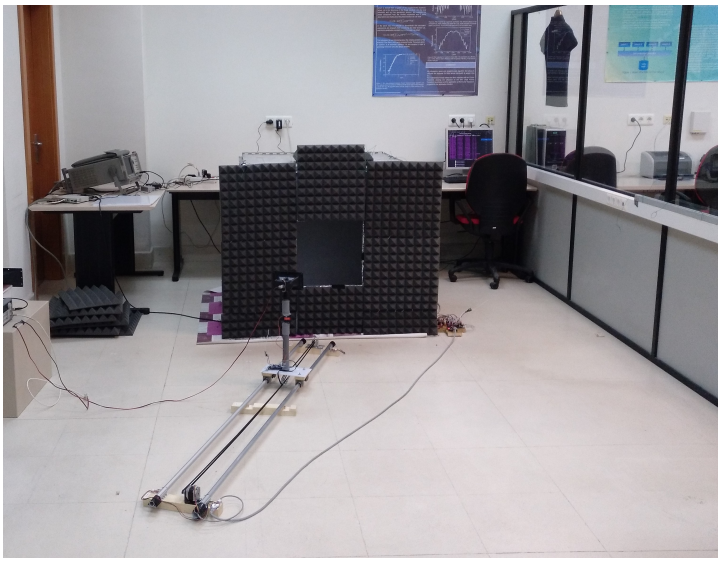

Figura 3.4.: Posición 1 del transmisor acercándose a la boca

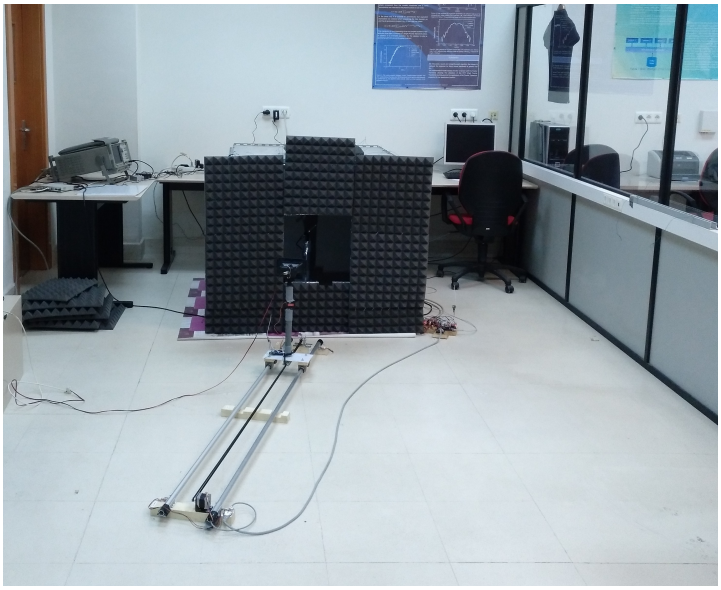

Figura 3.5.: Posición 2 del transmisor acercándose a la boca

3.2.3. Fuente con raíles paralelos a la pared frontal de la cavidad

Con esta forma de colocar la fuente era necesario utilizar el servo para que la bocina apuntara constantemente al centro de la apertura. Primeramente, desde una distancia lejana (z=-1m) como se ve en la figura 3.6, se observó que la señal perdía potencia, así que también se tomaron medidas con la fuente más cerca de la apertura para que la potencia de la señal dentro de la cavidad fuera lo mayor posible y los micrófonos midieran con mejor relación señal a ruido.

En la figura 3.7 se puede ver un ejemplo de esta configuración.

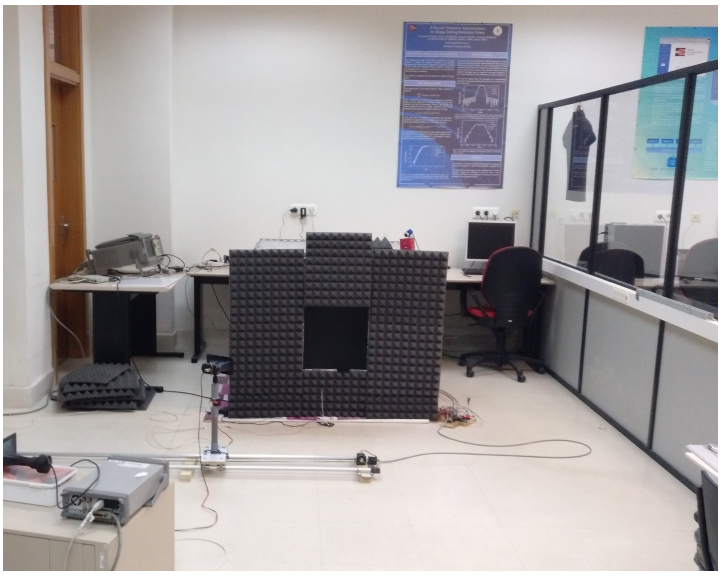

Figura 3.6.: Posición en paralelo a la caja

Figura 3.7.: Posición en paralelo a la caja, cerca de la boca

3.3. Archivo de datos capturados

En esta sección se va a explicar la estructura del archivo de datos que contiene las medidas capturadas para cada posición del transmisor. En la figura 3.8, se puede ver un ejemplo de este archivo.

#x=545[mm],y=-179.6[mm],z=-1422.4[mm] -> Source Location

x[mm]	y[mm]	z[mm]	REFERENCE_MIC[dBVrms]	SAMPLING_MIC[dBVrms]	TEMPERATURE[℃]	HUMIDITY[%]	ELAPSED_TIME[s]
545	885	695	-55.4265785217285	-51.9658393859863	21.01	27.19	4.414560079574585
545	855	695	-55.5919532775879	-57.5687866210937	21.05	27.17	9.422550678253174
545	825	695	-55.4279823303223	-56.1737899780273	21.06	27.16	14.432541847229004
545	795	695	-55.7821006774902	-56.3742065429688	21.07	27.1	19.43255877494812
545	765	695	-55.5728607177734	-58.7131958007812	21.07	27.08	24.44069218635559
545	735	695	-55.8125305175781	-51.2959938049316	21.08	27.06	29.449565410614014
545	705	695	-55.7903518676758	-61.0960540771484	21.06	27.07	34.450567960739136
545	675	695	-55.4816398620605	-51.0580101013184	21.08	27.07	39.450555086135864
545	645	695	-55.6758918762207	-52.6922454833984	21.07	27.07	44.448580503463745
545	615	695	-55.5843391418457	-60.2832527160645	21.08	27.07	49.4375581741333
545	585	695	-55.348762512207	-59.727409362793	21.08	27.05	54.43719267845154
545	555	695	-55.6064910888672	-55.3895606994629	21.08	27.04	59.44669222831726
545	525	695	-55.2568626403809	-55.8508644104004	21.09	27.02	64.45458102226257
545	495	695	-55.3116912841797	-79.5959930419922	21.11	27.01	69.42796397209167
545	465	695	-55.1520690917969	-61.5874099731445	21.14	26.96	74.43356943130493
545	435	695	-55.6137275695801	-58.1977272033691	21.11	26.95	79.43357110023499
545	405	695	-55.4435806274414	-61.6638565063477	21.09	26.97	84.4336953163147
545	375	695	-55.5852508544922	-63.5906791687012	21.13	26.99	89.41557478904724
545	345	695	-55.4339981079102	-53.0263633728027	21.13	26.98	94.44169807434082
545	315	695	-55.3282890319824	-55.9631576538086	21.11	26.98	99.44057941436768
545	285	695	-55.2677154541016	-56.4982414245605	21.11	26.98	104.4495759010315
545	255	695	-55.4179992675781	-60.924732208252	21.11	26.97	109.45869946479797
545	885	665	-55.1021423339844	-50.6857147216797	21.16	26.91	129.9975790977478
545	855	665	-55.0902099609375	-51.0801429748535	21.16	26.88	135.00845503807068
545	825	665	-55.2495269775391	-60.5685081481934	21.15	26.9	140.0185830593109
545	795	665	-55.2614593505859	-53.5793914794922	21.15	26.95	145.0267059803009
545	765	665	-55.252685546875	-62.9556083679199	21.16	26.91	149.98858261108398
545	735	665	-55.4148597717285	-52.9921035766602	21.15	26.92	154.9857108592987
545	705	665	-55.7174491882324	-61.5362968444824	21.16	26.91	159.9945855140686
545	675	665	-55.3361396789551	-56.6289901733398	21.17	26.9	164.99358701705933
545	645	665	-55.6328887939453	-60.668342590332	21.17	26.94	170.00170874595642
545	615	665	-55.6650848388672	-56.1062126159668	21.16	26.94	175.00056910514832
545	585	665	-55.3994216918945	-60.1777229309082	21.17	26.94	180.0116093158722
545	555	665	-55.1028938293457	-50.3138618469238	21.17	26.95	185.01958632469177
545	525	665	-54.9955978393555	-56.0384559631348	21.19	26.87	190.01958918571472
545	495	665	-55.2765502929688	-55.8960151672363	21.2	26.82	195.02958750724792
545	465	665	-55.3472099304199	-57.6383552551269	21.2	26.79	200.0285906791687
545	435	665	-55.1636085510254	-59.9684753417969	21.21	26.78	205.03784227371216
545	405	665	-55.2611694335937	-63.1848106384277	21.21	26.78	210.03757071495056
545	375	665	-55.0906105041504	-63.8075675964355	21.21	26.79	215.02658772468567
545	345	665	-55.1771125793457	-59.2816848754883	21.21	26.78	220.03471994400024
545	315	665	-55.7196235656738	-53.1427192687988	21.22	26.77	225.04572200775146
545	285	665	-55.6676177978516	-59.389705657959	21.23	26.75	230.06471610069275

Figura 3.8.: Ejemplo de una hoja de medidas

En la primera línea están escritas las coordenadas (en milímetros) en las que estaba situado el emisor para esta hoja de datos, decir que la coordenada x corresponde a la altura de la bocina. Tiene una almohadilla al comienzo para que a la hora de realizar las simulaciones de esta hoja de datos, tome esta línea como un comentario. En la segunda línea se encuentran los nombres de las columnas del archivo. Las primeras tres columnas hacen referencia a la posición del micrófono receptor en el interior de la cavidad, en milímetros.

La cuarta columna recoge los decibelios del valor eficaz captados por el micrófono de referencia conectado al analizador dinámico de señal. Esto se hace para poder diferenciar entre los decibelios generados por las ondas acústicas y los generados por el ruido en el ambiente. En la quinta columna se encuentran los decibelios del valor eficaz captados por el micrófono en el interior de la cavidad. Estos son los datos que realmente nos interesan para este trabajo. La sexta y séptima columna recogen los valores de humedad y temperatura para cada una de las posiciones del micrófono del interior de la cavidad. De esta forma es posible analizar el efecto tanto de la humedad como de la temperatura en la toma de medidas y saber si interfieren a la hora de realizar el muestreo. Por último, el *elapsed time*, recoge el tiempo transcurrido en cada posición de toma de medidas del micrófono en el interior de la cavidad. A partir de la tercera fila, esta inclusive, aparecen los resultados para cada una de las posiciones del micrófono en el interior de la cavidad.

Capítulo 4.

Simulación a partir de los datos muestreados

En esta sección se va a explicar cómo se han realizado las simulaciones variando el número de micrófonos y su posición en el interior de la cavidad. También se explicarán los gráficos obtenidos a partir de estas simulaciones en las cuales se observa la influencia de estos micrófonos en el exterior de la cavidad. Finalmente se realizará una comparación entre todas las simulaciones realizadas analizando la influencia del número y posición de los micrófonos.

Contenido

4.1. Base de la simulación

La simulaciones realizadas se basan en los datos tomados en cada posición del micrófono del interior de la cavidad, con un total de cuatrocientos pasos. De esta forma se conoce lo que cada micrófono en cada posición es capaz de captar al entrar las ondas acústicas en la cavidad. Sabiendo esto, se puede llegar a conocer la influencia en el exterior de la cavidad de estas ondas acústicas. Cada micrófono pasará a ser emisor de lo que capta, ya que si capta una amplitud de la señal en su posición significa que ese punto alcanzado por la perturbación se convierte en una fuente secundaria. La cavidad se convierte en una fuente constituida por distintos emisores. Para reducir los cálculo de estos emisores en el exterior se acude al Principio de Equivalencia para sustituir todas fuentes del interior de la cavidad por una distribución de emisores en la apertura. Debido a que únicamente contamos con datos de intensidad acústica es necesario emplear un algoritmo de extrapolación modificado que ha sido proporcionado por el tutor del proyecto. La distribución equivalente generará la misma influencia que el conjunto de emisores del interior de la cavidad.

También se ha subdivido la apertura en varias subaperturas más pequeñas, para poder encontrarnos en el entorno de campo lejano desde el punto de vista del tamaño de la subapertura y la distancia hasta cada uno de los micrófonos del interior de la cavidad y de los puntos de observación en el exterior de la cavidad. De esta forma se realiza una aproximación por rayos paralelos de las ondas acústicas. Las ondas acústicas son reflejadas en el interior de la cavidad. Así se puede llegar a describir el comportamiento de las ondas acústicas en el interior de la cavidad y tratar de averiguar la posición de la fuente externa. Para modelar las paredes de la cavidad se utiliza el método de las imágenes.

4.2. Realización de simulaciones

La realización de la simulaciones se controla mediante un archivo de configuración. A modo de muestra se ha incluido un ejemplo del mismo en el anexo del trabajo. En este fichero de configuración se pueden realizar varias modificaciones como puede ser variar el número de subaperturas o realizar operaciones de diezmado en el muestreo espacial de la intensidad acústica en el interior de la cavidad.

El número máximo de muestras que se se tomó a modo de referencia eran cuatrocientos micrófonos (tantos como paradas realizamos al realizar el muestreo espacial), si se escoge dividir entre dos, se realiza una simulación con doscientos micrófonos, descartando la mitad de las medidas.

De la forma en la que está configurada la simulación, estos doscientos micrófonos corresponden con las posiciones pares. Si se escoge dividir entre cuatro, la simulación toma un micrófono (partiendo desde el punto inicial) cada cuatro micrófonos. Para la realización de simulaciones generales no es un desarrollo malo, pero cuando se quieren analizar configuraciones más complicadas para los puntos de muestreo en el interior de la cavidad con muestreos no uniformes la operación para descartar medidas no es tan inmediata. Por ello se decidió no diezmar en la etapa de simulación, y posteriormente diezmar las medidas pasando a la simulación únicamente la información de los micrófonos seleccionados. En los archivos con las medidas se escogían las posiciones de los micrófonos que queríamos simular y el resto se borraba. De modo que se podía conseguir de una manera muy flexible cualquier distribución espacial para los micrófonos que posteriormente se tienen en cuenta en la simulación.

Para lanzar las simulaciones se utilizó una red de agentes de cómputo distribuidos en la que se disponía de varios servidores. Estos servidores se repartían los cálculos para realizar la simulación ya que existían simulaciones con mucha carga de procesamiento.

Una vez realizada la simulación deseada, se dispone de un archivo generado por la simulación. Con este archivo se crea una imagen en la que se puede observar la influencia de las fuentes secundarias correspondientes a los micrófonos escogidos en el exterior de la cavidad.

4.3. Imagen de las simulaciones

En esta sección voy a explicar cómo es una imagen calculada a partir de una simulación, ya que en la sección 4.4 se va a realizar comparaciones entre ellas. Un ejemplo de imagen de simulación es la que se ve en la figura 4.1.

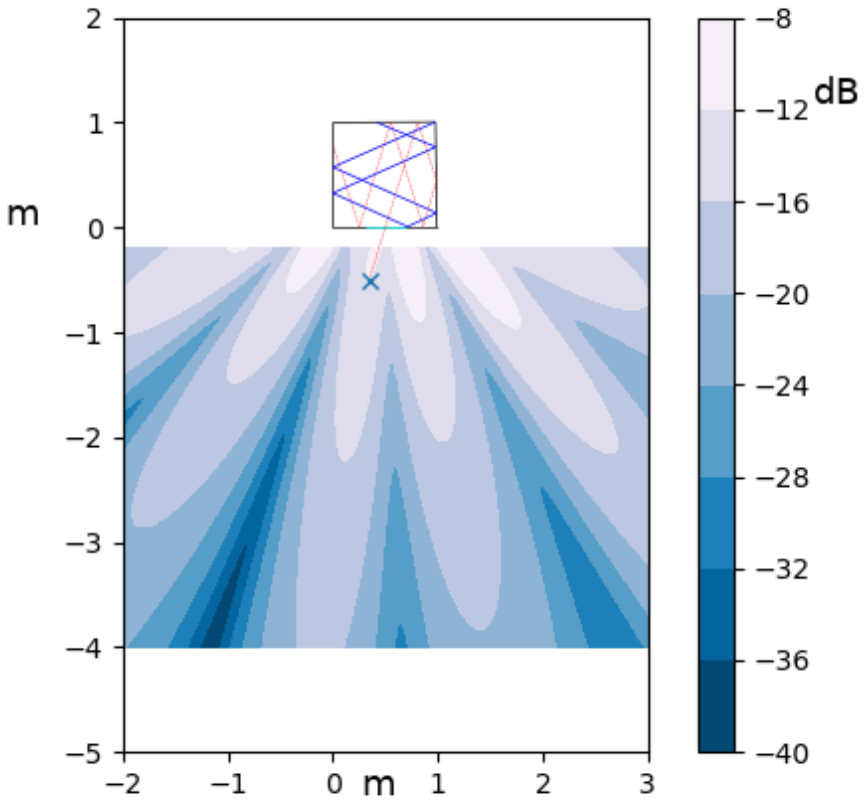

Figura 4.1.: Ejemplo de una imagen obtenida a partir de la simulación de 400 fuentes secundarias caracterizadas mediante 400 muestras espaciales de intensidad acústica en el interior de la cavidad. La "X"señala la posición del la fuente acústica que estaba emitiendo hacia la cavidad cuando se tomaron las muestras en el interior de la misma.

La figura 4.1 corresponde a una de las simulaciones realizada empleando la distribución espacial de intensidad acústica en cuatrocientas posiciones del micrófono en el interior de la cavidad. La escala situada a la derecha de la figura corresponde a la amplitud de la señal generada en el exterior en decibelios, y la escala situada a la izquierda y debajo de la figura corresponden a ejes de coordenadas en metros. El marcador "X", señala la posición en la que se encontraba el emisor cuando se tomaron las medidas en el interior de la cavidad, los lóbulos de color azul-blanquecino corresponden a la amplitud de la señal generada. Como se ve en la escala de amplitud, el color azul más oscuro corresponde a señales más débiles (por lo que serán mínimos) y el color blanco a la mayor amplitud (por lo que serán máximos).

En el interior de la cavidad las rectas de color rojo corresponden a las sucesivas reflexiones de la onda acústica. Lo que se busca con estas imágenes es averiguar cuántas muestras espaciales en el interior de la cavidad hacen falta para mantener los máximos y mínimos en las mismas posiciones que se obtienen cuando se emplean cuatrocientas posiciones del micrófono interno. Ya que si se mantienen la posiciones de dichos máximos y mínimos significa que se capta casi la misma información y por tanto se puede reducir el tiempo necesario para obtener los datos necesarios.

4.4. Tipos de simulaciones realizadas

Para poder caracterizar la información que se puede obtener a partir de una determinada distribución espacial de muestras en el interior de la cavidad es necesario realizar una gran variedad de simulaciones. Las simulaciones se realizan utilizando distintas posiciones del transmisor, en cada una de ellas se calcula el efecto de las fuentes secundarias en el interior de la cavidad para poder tener la posibilidad de encontrar las cosas que dependen de la posición de la fuente externa. Una vez se tiene la imagen generada con cuatrocientos micrófonos, se realizan las simulaciones y las imágenes con la selección deseada de las muestras disponibles en el interior de la cavidad y se observa la variación obtenida respecto a la simulación de referencia. De esta forma se puede ver si varía mucho la información captada. Voy a incluir un número pequeño de las simulaciones realizadas ya que se realizaron una gran cantidad.

4.4.1. A distancia de dos metros y modificando el valor decimation

Inicialmente se realizaron simulaciones a una distancia de dos metros de la boca de la cavidad y después se fue modificando el valor decimation del archivo de configuración de la simulación para ir mermando progresivamente el número de muestras considerado en las simulaciones. La siguientes figuras corresponden a un valor decimation=2 (4.2), 3 (4.3), 4 (4.4) y 5 (4.5). Es decir, 200 muestras tomadas en posiciones pares, 199 muestras situados en posiciones impares, luego 100 muestras y finalmente 80 muestras.

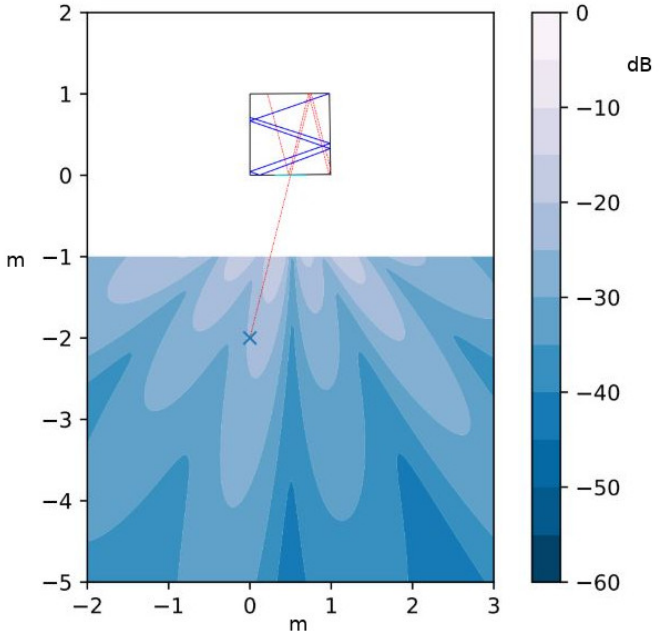

Figura 4.2.: Emisor situado a dos metros y el número decimation a 2

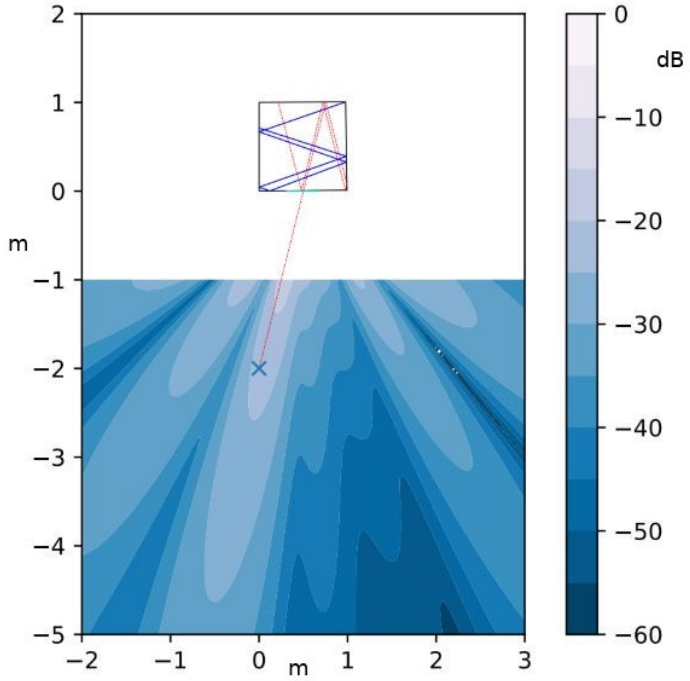

Figura 4.3.: Emisor situado a dos metros y el número decimation a 3

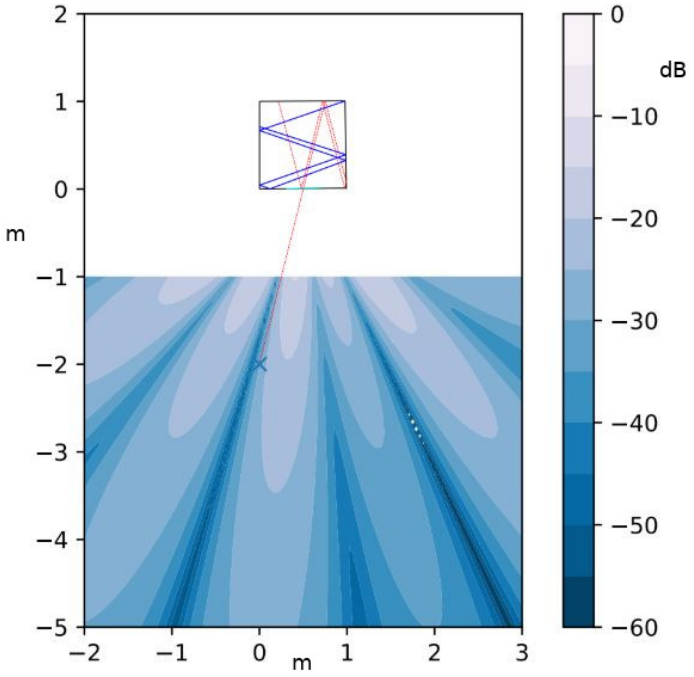

Figura 4.4.: Emisor situado a dos metros y el número decimation a 4

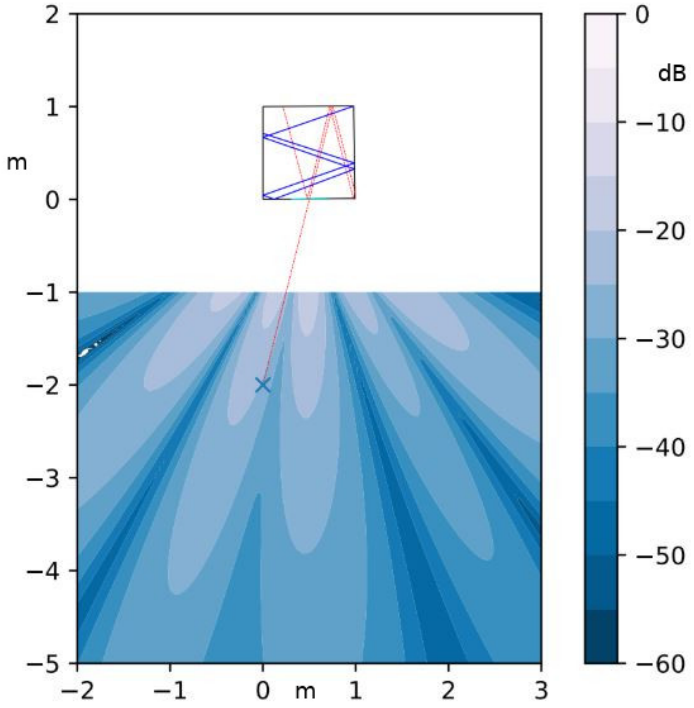

Figura 4.5.: Emisor situado a dos metros y el número decimation a 5

4.4.2. A distancia de cincuenta centímetros y formando una recta

Se decidió situar la fuente más cerca de la cavidad para que la señal acústica entrara con más potencia y así mejora la relación señal a ruido. Se realizó una simulación con una fila de micrófono en el lado derecho paralela a la pared de la cavidad 4.6, otra simulación con una fila en el lado izquierdo también paralela ala pared izquierda de la cavidad 4.7 y finalmente con las filas en ambos lados también paralelos a sus respectivas paredes 4.8.

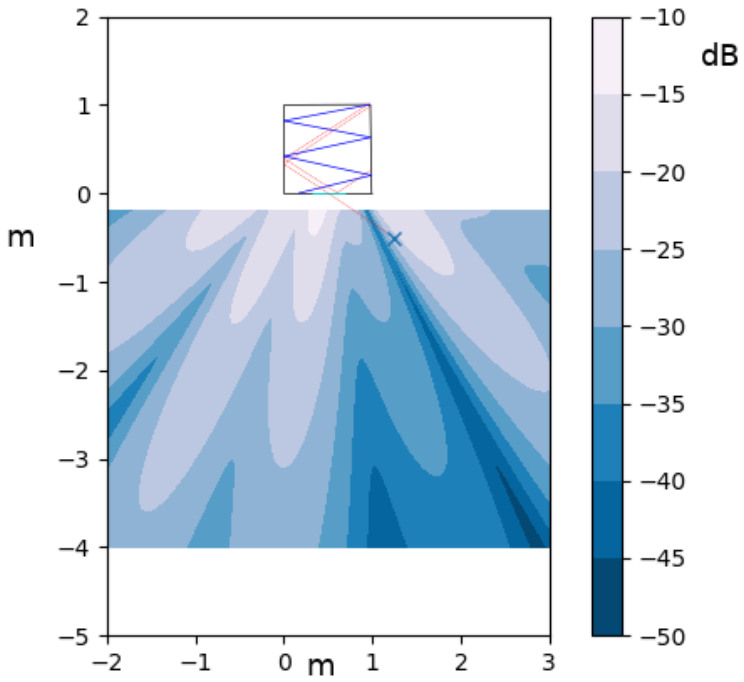

Figura 4.6.: Emisor situado a 50 cm y micrófonos formando una fila en el lado derecho

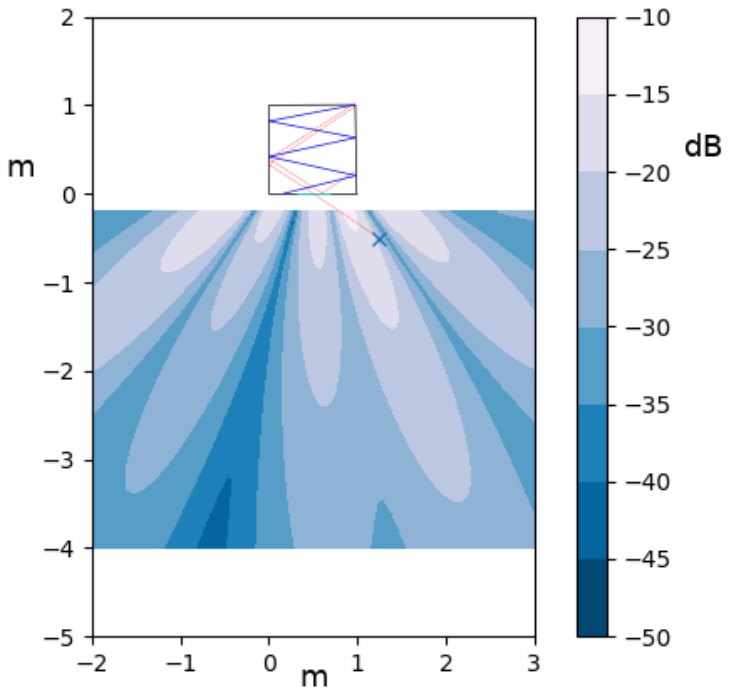

Figura 4.7.: Emisor situado a 50 cm y micrófonos formando una fila en el lado izquierdo

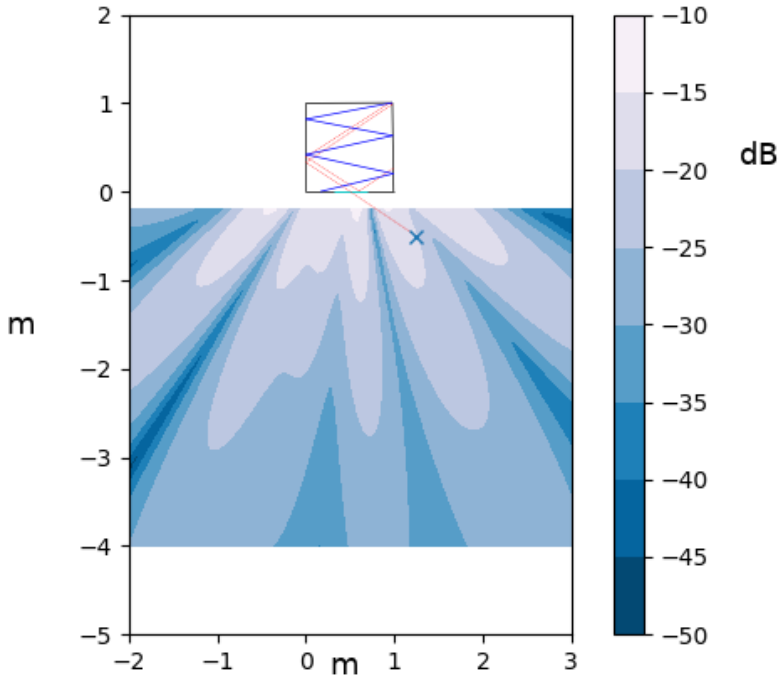

Figura 4.8.: Emisor situado a 50 cm y micrófonos formando una fila en el lado derecho e izquierdo

4.4.3. A distancia de cincuenta centímetros y realizando distintas configuraciones con los micrófonos

Después de haber realizado todas las anteriores simulaciones y observar que no se realizaba ninguna mejora, se decidió formar distintas estructuras para poder encontrar una forma mas eficiente. Primeramente se realizó un disposición circular formada por 10 micrófonos (4.9), la siguiente estructura fue un cuadrado formado por 40 micrófonos (4.10) y finalmente una escalera con comienzo en la esquina derecha superior de la caja y con final en la esquina inferior izquierda formado por 20 micrófonos (4.11).

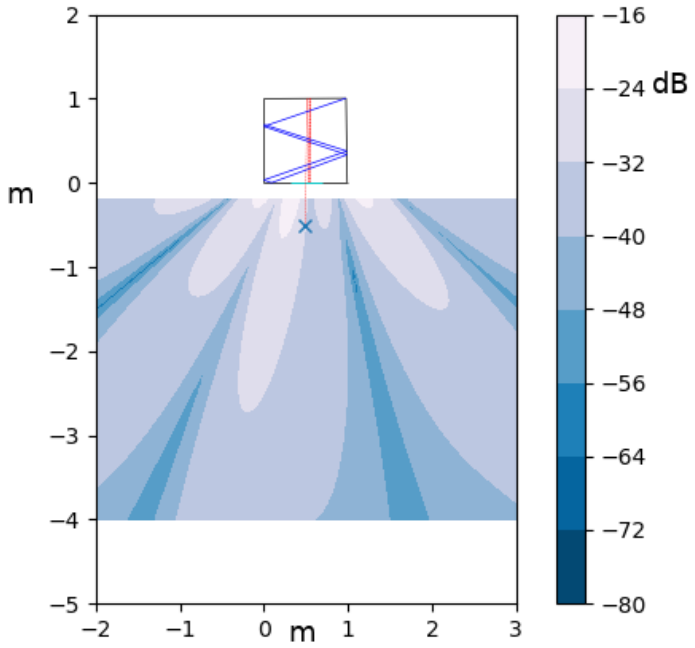

Figura 4.9.: Emisor situado a 50 cm y micrófonos formando una circunferencia

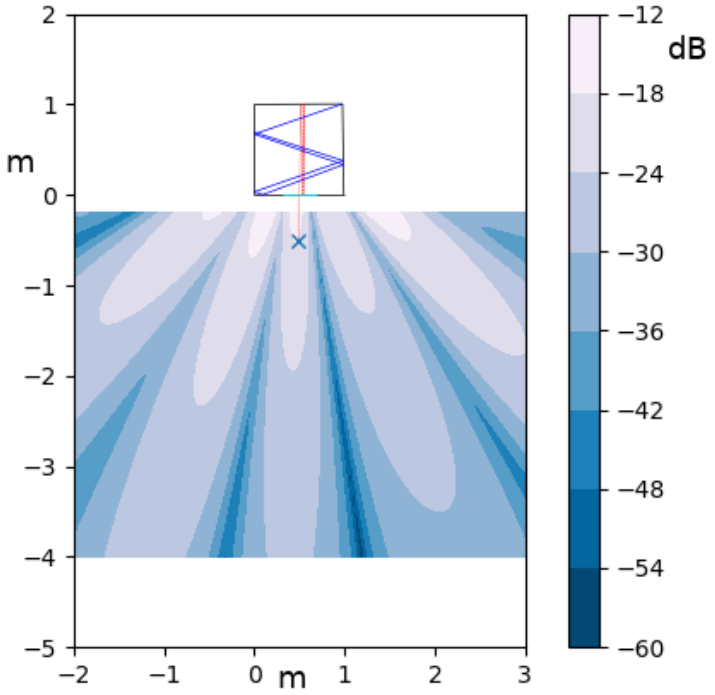

Figura 4.10.: Emisor situado a 50 cm y micrófonos formando una cuadrado

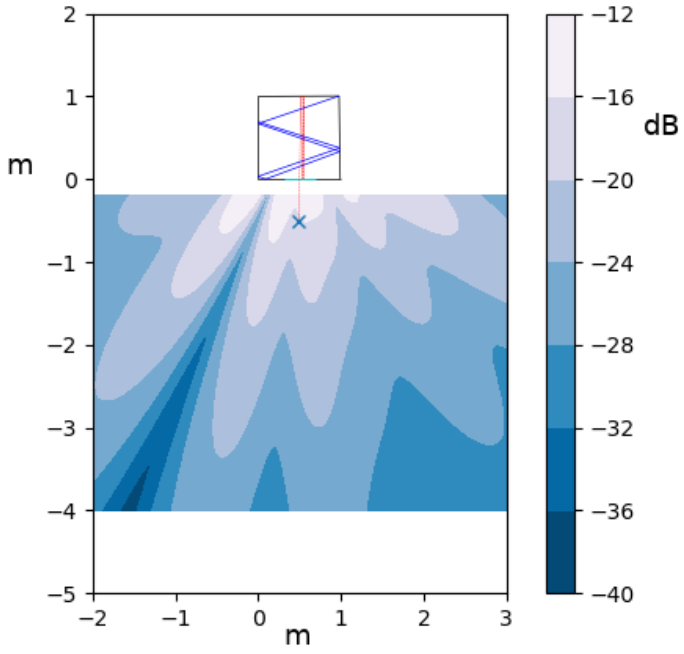

Figura 4.11.: Emisor situado a 50 cm y micrófonos formando una escalera

4.5. Comparación con cuatrocientos micrófonos

Se realizó una comparación de cada una de las simulaciones anteriormente presentadas con la correspondiente de cuatrocientos micrófonos. Se observaba que tanto los máximo como los mínimos se movían de posición y se inclinaban respecto a la de cuatrocientos micrófonos. Incluso había alguna simulación que no sacaba el mismo número de máximos y de mínimos. Para concluir este capítulo se puede decir que para conocer la total influencia de los micrófonos del interior de la cavidad sería necesario la utilización de los cuatrocientos micrófonos ya que la información captada por cada uno de ellos es de gran importancia.

Capítulo 5.

Conclusión y líneas futuras

Este capítulo recoge las conclusiones extraídas durante el desarrollo de la investigación. Como punto final se menciona una forma alternativa de realizar el estudio de la influencia del número de muestras espaciales en el prototipo de radar pasivo acústico e investigaciones futuras en las que sería de gran ayuda lo llevado acabo en este trabajo.

Contenido

5.1. Conclusión

Este trabajo forma parte de una investigación más amplia que busca desarrollar un prototipo de radar acústico pasivo que permita identificar la posición de un emisor externo. Para identificar el origen del frente de onda que proviene del emisor, no se utilizan lo retardos de llegada en los elementos de un array como es habitual. En nuestro caso el frente de onda entra dentro de una cavidad cúbica resonante de un metro de lado y con paredes de fibrocemento a través de una apertura cuadrada en la pared frontal de la cavidad que permite el paso hacia el interior de la misma. El objetivo final es deducir la posición de la fuente externa mediante el análisis de la distribución de intensidad acústica en el interior de la cavidad. Esta distribución interna se mide utilizando un micrófono que es desplazado mediante motores paso a paso generando una malla de muestreo bidimensional en un plano paralelo al suelo de la cavidad.

La distribución de los campos dentro de la cavidad es muy complicada y es necesario obtener medidas para muchas posiciones externas de la fuente antes de poder obtener conclusiones fiables. Antes de este trabajo la fuente externa tenía que se desplazada manualmente, con el consiguiente retraso en la obtención de los datos experimentales necesarios. A ese ritmo se tardaría muchos meses en obtener las medidas necesarias. Para solucionar este problema se ha desarrollado un automatismo para desplazar el emisor de forma automática y controlada por ordenador, de modo que lo que antes tardaría varios meses ahora se puede conseguir en un par de semanas.

Una vez reducido el tiempo necesario para cambiar el emisor de sitio, resulta muy importante averiguar hasta qué punto es posible reducir el número de medidas que se toman dentro de la cavidad. Por una parte de esta manera se acelera el algoritmo y por otra parte las medidas tomadas son más fiables, porque al extenderse mucho en el tiempo la toma de muestras, las condiciones de temperatura y humedad cambian provocando que los resultados de las medidas sean menos consistentes. Idealmente todas las medidas deberían estar hechas con la misma temperatura y humedad porque el micrófono cambia de características al cambiar estos parámetros.

Después de haber analizado todos los datos tomados y haber realizado distintas simulaciones con distintas posiciones y número de receptores, se puede observar en los gráficos desarrollados en el capítulo anterior, que en cuanto se realiza la simulación empleando menos de 400 muestras (para una frecuencia de emisión de 5 kHz) la variación en el resultado debida al número de micrófonos es importante. Sin embargo hay posiciones y situaciones específicas en las que esta variación es menor, pero son solo casos aislados. Al cambiar la posición del emisor hay cambios radicales. Por lo que se puede llegar a la conclusión de que es necesario tomar al menos 400 muestras para poder captar la toda la información. A la vista de los resultados obtenidos todas las muestras analizadas tienen su importancia en al menos alguna posición de la fuente externa.

5.2. Líneas futuras

Una vez que se pueda aislar mejor la información que depende de la posición de la fuente externa se podría volver a realizar el análisis actual con posibilidad de llegar a una respuesta distinta.

Apéndice A.

Anexo : Código de archivos utilizados

A.1. Arduinouno.ino

```
#include <SPI.h>
#include <ServoTimer2.h>
#include <RF24L01.h>
#include <RF24.h>
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created
ServoTimer2 myservo;
RF24 radio(8,10);//I'm setting the pins for the radio broadcast.

/*const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };
this indicates the pipe to use, which we can imagine as
one of the different channels the radio can operate on.
The pipe is a 64-bit number, which we indicate with the type uint64_t
, integer without 64-bit sign,  and in case it wasn't clear enough,
we indicate to the compiler that it is an LL at the end of the number
( LongLong = 64 bits).
We're going to need two channels or pipes, the first
one will speak and the second one will listen,
and the second one will work the other way around.*/
```

```
long pos = 0;    // variable to store the servo position
char ServoCharMsg[4];

void setup()
{

    pinMode(10, OUTPUT);//Pin 10 as a output
    Serial.begin(9600); // Debugging only
    radio.begin();//Inicialize the radio channel
    Serial.println("setup");
    radio.setRetries(15,15);//We set 15 attempts at initializing
    //Since it is the receiver it starts by listening to the channel
    radio.startListening();
    radio.openWritingPipe(pipes[1]);// We choose channel to send data
    radio.openReadingPipe(1,pipes[0]);//Channel to receive data
}

void loop()
{
    // If tha channel has data go into the if
    if ( radio.available() )
    {
        int i=0;
        char ack[]="ACK";
        // Flash a light to show received good message
        digitalWrite(13, true);
        // Message with a good checksum received, dump it.
        Serial.print("Got: ");
        //We reiceve the data
        radio.read(&ServoCharMsg, sizeof(ServoCharMsg));

        radio.stopListening();//Stop listening the channel

        radio.write(&ack, sizeof(ack) );//we send the message ACK
        Serial.println("Enviando Asentimiento");
```

```
        ServoCharMsg[10] = '\0';
        // Convert Sensor1CharMsg Char array to integer

        pos = atol(ServoCharMsg);

        if(pos < 0) pos = 0;
        if(pos > 180*1000000) pos = 180;
        // long map(long x, long in_min, long in_max, long out_min, long
        pos = map(pos, 0, 180*1000000, 500, 2500);
        Serial.print(pos);
        myservo.attach(9);
        // tell servo to go to position in variable 'pos'
        myservo.write(pos);
        // waits 1000 ms for the servo to reach the position
        delay(1000);
        myservo.detach();
        Serial.println("");
        digitalWrite(13, false);
         // To take more data, we start listening one more time
         radio.startListening();
         delay(500);
      }
}
```

A.2. ArduinoMega.ino

```
/*
   Uses the HTU21D library to display the current humidity
   and temperature.
   Open serial monitor at 9600 baud to see readings.
   Errors 998 if not sensor is detected. Error 999 if CRC is bad.

   HTU21D Hardware Connections (Breakoutboard to Arduino):
   -VCC = 3.3V
   -GND = GND
   -SDA = A4 (use inline 330 ohm resistor if your board is 5V)
   -SCL = A5 (use inline 330 ohm resistor if your board is 5V)
,*/
/*
   RF24 Radio transceptor
   Uses the module RF24 to do a duplex comunication
   between an arduino MEGA 2560  and an arduino UNO
*/
#include "Wire.h"
#include <string.h>
#include "SparkFunHTU21D.h"
#include "SerialCommand.h"
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
//SerialComand object
SerialCommand sCmd;

//Create an instance of the object
HTU21D myHumidity;
RF24 radio(12,13); // We initialize the control pins of the module

const uint64_t pipes[2] = { 0xF0F0F0F0E1LL, 0xF0F0F0F0D2LL };

// steps per mm = steps_per_revolution*microstepping_factor
```

```
//(pitch*number of teeth of the pulley)
int spm =200*16/(2*20);

const byte S1_NoAmp= 0x15;
const byte S2_NoAmp= 0x16;
const byte S3_NoAmp= 0x18;

const byte S1_SiAmp= 0x25;
const byte S2_SiAmp= 0x26;
const byte S3_SiAmp= 0x28;

// Function prototypes
void unrecognized(const char *);
void sendVersion();
void processProbe();
void processMotorY();
void processMotorZ();
void processMotorE();
void selectProbe(int, boolean);
void moveMotorY(long, boolean);
void moveMotorZ(long, boolean);
void moveMotorZ1(long, boolean);
void moveMotorZ2(long, boolean);
void moveMotorE(long, boolean);
void restartMotorY();
void restartMotorZ();
long restartMotorZ1();
long restartMotorZ2();
void restartMotorE();
void finalizeMotorY();
void finalizeMotorE();
void finalizeMotorZ();
void getTemperature();
void getHumidity();
void setServo();
void enableMotors();
```

```
// Limit switches
    int endEPin = 10;
    int begEPin = 11;
    int endYPin = 6;
    int begYPin = 5;
    int endZ1Pin = 15;
    int begZ1Pin = 14;
    int endZ2Pin = 16;
    int begZ2Pin = 17;
// Pins to enable the motors
    int enaYPin=2;
    int enaZ1Pin=4;
    int enaZ2Pin=3;
    int enaEPin= 7;
// Direction and step pins for the external motor (not for PORTC)
    int dirEPin=8;
    int stepEPin=9;
```

```
void setup()
{

 //On my case on the 10th but he's busy so I'll put him on the 12th.
 pinMode(13, OUTPUT);

  DDRA = 0xff; // PORT A initialization as output
  PORTA = 0x00; // PORT A starts in LOW state

  // Configuramos pines de dir y step para Y,Z1,Z2 (PORTC)
  DDRC = 0xff; // PORT C initialization as output
  PORTC = 0x00; // PORT C starts in LOW state
  // Configuramos pines de dir y step para motor externo E
  pinMode(dirEPin, OUTPUT);
  pinMode(stepEPin, OUTPUT);
  digitalWrite(dirEPin,LOW);
  digitalWrite(stepEPin,LOW);

  Serial.begin(9600);
  radio.begin(); //We begin radio communication
  radio.setRetries(15,15);// Maximum retries
  myHumidity.begin();

  //We'll have to open the communication channels or pipes,
  //one to broadcast and the other to receive
 radio.openWritingPipe(pipes[0]);
 radio.openReadingPipe(1,pipes[1]);
  //We configure limit switch pins
  pinMode(endEPin, INPUT);
  pinMode(begEPin, INPUT);

  pinMode(endYPin, INPUT);
  pinMode(begYPin, INPUT);
```

```
pinMode(endZ1Pin, INPUT);
pinMode(begZ1Pin, INPUT);

pinMode(endZ2Pin, INPUT);
pinMode(begZ2Pin, INPUT);

//We configure pins to enable motors
pinMode(enaYPin, OUTPUT);
pinMode(enaZ1Pin, OUTPUT);
pinMode(enaZ2Pin, OUTPUT);
pinMode(enaEPin, OUTPUT);

// Motors enabled. Motor coils with current.
//They must be enabled during the
// sampling process to avoit lost steps,
//now they are disabled until restart or
// finalize operations
digitalWrite(enaYPin,HIGH);
digitalWrite(enaZ1Pin,HIGH);
digitalWrite(enaZ2Pin,HIGH);
digitalWrite(enaEPin,HIGH); // disabled now: enabled after restart

// Setup callbacks for SerialCommand commands
// Process the probe selection command
sCmd.addCommand("PRB",processProbe);
// Answer with the version of arduino in use
sCmd.addCommand("VER",sendVersion);
sCmd.addCommand("MTY",processMotorY);
sCmd.addCommand("MTZ",processMotorZ);
sCmd.addCommand("MTE",processMotorE);
sCmd.addCommand("RSY",restartMotorY);
sCmd.addCommand("RSZ",restartMotorZ);
sCmd.addCommand("RSE",restartMotorE);
sCmd.addCommand("FNY",finalizeMotorY);
sCmd.addCommand("FNE",finalizeMotorE);
```

```
  sCmd.addCommand("FNZ",finalizeMotorZ);
  sCmd.addCommand("ENA",enableMotors);
  sCmd.addCommand("TMP",getTemperature);
  sCmd.addCommand("HMD",getHumidity);
  sCmd.addCommand("ANG",setServo);
  // Handler for command that isn't matched  (says "What?")
  sCmd.setDefaultHandler(unrecognized);

}

void loop()
{
    // We don't do much, just process serial commands
    sCmd.readSerial();
}

//Function for sending and receiving assent
void send_rf (char *message)
{
  bool ok=false;

  while (ok==false)
  {
    radio.write(message, sizeof(String));
    radio.startListening();//We hear again
    delay(500);
    if ( radio.available() )//If data are available
    {
        char recibido[3] = " ";
        radio.read(&recibido, sizeof(recibido));
         //To check that the assent arrives correctly
        if  (strstr(recibido, "ACK"))  ok=true;

        else ok=false;

       // delay(500);
```

```
    }
    //If there is no data available, it has not been sent.
    else ok= false;
  radio.stopListening();
  delay(500);
  }
  Serial.println("Asentimiento recibido");
}

void setServo(){
  char *arg;
  long angle; // angle in microdegrees (servo rotation)
  boolean error=false;

  arg = sCmd.next();

  if (arg != NULL) {
    angle = atol(arg);  // Converts a char string to an axis number
    if ((angle<0)||(angle>180*1000000)){
      Serial.println("Angle must be between 0.0 and 180.0 10^6 ");
      error=true;
    }
  }
  else {
    Serial.println("No arguments. Nothing done.");
    error=true;
  }

  if (error != true){
    char mensaje[10];
    sprintf(mensaje,"%lu",angle);
    mensaje[9]='\0';
    send_rf(mensaje);
```

```
    char respuesta[50];
    sprintf(respuesta,"ANG %lu.",angle);
    Serial.println(respuesta);
  }

}

/* Name: void processProbe() ****************************

   Description: Parse serial command to obtain the axis number of
   the triaxial probe (1/2/3) and if we use amplifier or not
   (1/0->On/Off). It calls function selectProbe to send signals to
   pins.

   Args:
   ,*arg: (char *) command with arguments

,*****************************************************************/

void processProbe() {
  char *arg;
  int probe; // probe ordinal number
  int amp; // 1/0 amplifier on/off
  boolean error=false;

  arg = sCmd.next();

  if (arg != NULL) {
    probe = atoi(arg);  // Converts a char string to an axis number
    if ((probe!=1)&&(probe!=2)&(probe!=3)){
      Serial.println("Probe can be 1/2/3. Nothing done.");
      error=true;
    }
  }
  else {
    Serial.println("No arguments. Nothing done.");
```

```
    error=true;
  }

  arg = sCmd.next();
  if (arg != NULL) {
    amp = atoi(arg);
    if ((amp!=0)&&(amp!=1)){
      Serial.println("amp can be 0/1. Nothing done.");
      error=true;
    }
  }
  else {
    Serial.println("No amplifier argument. Nothing done.");
    error=true;
  }

  if (error != true){
    selectProbe(probe,(amp==1));
    char respuesta[50];
    sprintf(respuesta,"Sonda %d, amplificador %d.",probe,amp);
    Serial.println(respuesta);
  }
}

/* Name: void selectProbe(probe, boolean) ***************

   Description:
   Sets digital outputs to control the insite switch box.
   Arduino pins: 22-29  PORTA

   Args:
   probe: (int) number of the probe
   amp: (boolean ) determines if we use amplifier or not

,*****************************************************/
```

```
void selectProbe(int probe, boolean amp)
{
  byte data_amp_off[4]={0x0,S1_NoAmp,S2_NoAmp,S3_NoAmp};
  byte data_amp_on[4]={0x0,S1_SiAmp,S2_SiAmp,S3_SiAmp};

  if (amp){
    PORTA = data_amp_on[probe];
  }else{
    PORTA = data_amp_off[probe];
  }

  delay(15); // milliseconds
  PORTA = B00000000; // insite box has latch registers

}



void unrecognized(const char *command) {
  Serial.println("What?");
}

void sendVersion() {
  char respuesta[50];
  sprintf(respuesta,"Arduino mega 2560.");
  Serial.println(respuesta);
}

void getHumidity(){
  float humd = myHumidity.readHumidity();
  char hmd[10];
  dtostrf(humd,1,2,hmd);
  Serial.println(hmd);
}
```

```
void getTemperature(){

float temp = myHumidity.readTemperature();
  char tmp[10];
dtostrf(temp,1,2,tmp);
  Serial.println(tmp);
}

/* Name: void processMotorY() ****************************

   Description:
   Parse serial command to obtain the number of steps to
   move the motor Y, and the directions of movement

   Args:
   ,*arg: (char *) command with arguments

,*************************************************************/

void processMotorY() {
  char *arg;
  long steps; // number of steps
  int forward; // 0/1 forward false/true
  boolean error=false;

  arg = sCmd.next();

  if (arg != NULL) {
        steps=atol(arg);
    if (steps<=0){
      char respuesta[50];
      sprintf(respuesta,"Negative steps: %lu cadena.ERROR");
      Serial.println(respuesta);
      error=true;
    }
```

```
  }
  else {
    Serial.println("No arguments. Nothing done.");
    error=true;
  }

  arg = sCmd.next();
  if (arg != NULL) {
    forward = atoi(arg);
    if ((forward!=0)&&(forward!=1)){
      Serial.println("forward can be 0/1. Nothing done.");
      error=true;
    }
  }
  else {
    Serial.println("No direction argument. Nothing done.");
    error=true;
  }

  if (error != true){
    moveMotorY(steps,(forward==1));
    char respuesta[50];
    sprintf(respuesta,"Pasos en eje y: %lu",steps);
    Serial.println(respuesta);
  }
}

/* Name: void processMotorE() ****************************

   Description:
   Parse serial command to obtain the number of steps to
   move the motor E (exterior), and the directions of movement

   Args:
   ,*arg: (char *) command with arguments
```

```
,*********************************************************/

void processMotorE() {
  char *arg;
  long steps; // number of steps
  int forward; // 0/1 forward false/true
  boolean error=false;

  arg = sCmd.next();

  if (arg != NULL) {
        steps=atol(arg);
    if (steps<=0){
      char respuesta[50];
      sprintf(respuesta,"Negative steps:%lu cadena.ERROR");
      Serial.println(respuesta);
      error=true;
    }
  }
  else {
    Serial.println("No arguments. Nothing done.");
    error=true;
  }

  arg = sCmd.next();
  if (arg != NULL) {
    forward = atoi(arg);
    if ((forward!=0)&&(forward!=1)){
      Serial.println("forward can be 0/1. Nothing done.");
      error=true;
    }
  }
  else {
    Serial.println("No direction argument. Nothing done.");
    error=true;
  }
```

```
  if (error != true){
    moveMotorE(steps,(forward==1));
    char respuesta[50];
    sprintf(respuesta,"Pasos en eje E: %u",steps);
    Serial.println(respuesta);
  }
}

/* Name: void processMotorZ() ****************************

   Description:
   Parse serial command to obtain the number of steps to
   move the motors in Z, and the directions of movement

   Args:
   ,*arg: (char *) command with arguments

,*****************************************************/

void processMotorZ() {
  char *arg;
  long steps; // number of steps
  int forward; // 0/1 forward false/true
  boolean error=false;
  arg = sCmd.next();

  if (arg != NULL) {
      steps=atol(arg);
    if (steps<=0){
      char respuesta[50];
      sprintf(respuesta,"Negative steps: %u. ERROR",steps);
      Serial.println(respuesta);
      error=true;
    }
  }
```

```
  else {
    Serial.println("No arguments. Nothing done.");
    error=true;
  }

  arg = sCmd.next();
  if (arg != NULL) {
    forward = atoi(arg);
    if ((forward!=0)&&(forward!=1)){
      Serial.println("forward can be 0/1. Nothing done.");
      error=true;
    }
  }
  else {
    Serial.println("No direction argument. Nothing done.");
    error=true;
  }

  if (error != true){
    moveMotorZ(steps,(forward==1));
    char respuesta[50];
    sprintf(respuesta,"Pasos en eje z: %u",steps);
    Serial.println(respuesta);
  }
}


void moveMotorY(long steps, boolean forward){
  // motors enabled to avoid lost steps
  // This motor uses bits 5 (direcc.) and 6 (step) del puerto C
  byte Puerto=B00000000;

  if(forward==true){
    Puerto=B00000000; //Pull direction pin low to move "forward"
  }else{
    Puerto=B00100000; //Pull direction pin high to move "backward"
```

```
  }

  for(long x=0; x<steps; x++)
  {
    PORTC=Puerto | B00010000; //Trigger one step forward
    delayMicroseconds(100);
    PORTC=Puerto & B00100000;
    delayMicroseconds(100);
  }

}

void moveMotorE(long steps, boolean forward){
  // motors enabled to avoid lost steps
  if(forward==true){
    digitalWrite(dirEPin,LOW); //Pull direction to move "forward"
  }else{
    digitalWrite(dirEPin,HIGH); //Pull direction to move "backward"
  }

  for(long x=0; x<steps; x++)
  {
    digitalWrite(stepEPin,HIGH); //Trigger one step forward
    delayMicroseconds(100);
    digitalWrite(stepEPin,LOW);
    delayMicroseconds(100);
  }

}


void moveMotorZ(long steps, boolean forward){
  byte Puerto=B00000000;
  // Motor Z1 uses bits 2 (direcc.) and 1 (step) del puerto C
  // Motor Z2 uses bits 4 (direcc.) and 3 (step) del puerto C
```

```
  if(forward==true){
    Puerto=B00001000; //Pull direction pin low to move "forward"
  }else{
    Puerto=B00000010; //Pull direction pin high to move "backward"
  }

  for(long x=0; x<steps; x++)
  {
    PORTC=Puerto | B00000101; //Trigger one step forward
    delayMicroseconds(1000);
    PORTC=Puerto & B00001010;
    delayMicroseconds(1000);
  }

}

// Motor z1 (close to y=0)
void moveMotorZ1(long steps, boolean forward){
  byte Puerto=B00000000;
  // Motor Z1 uses bits 2 (direcc.) and 1 (step) del puerto C
  if(forward==true){
    Puerto=B00000000; //Pull direction to move "forward"
  }else{
    Puerto=B00000010; //Pull directionto move "backward"
  }

  for(long x=0; x<steps; x++)
  {
    PORTC=Puerto | B00000001; //Trigger one step forward
    delay(1);
    PORTC=Puerto & B00000010;
    delay(1);
  }

}
```

```
    // motor z2 (far from y=0)
    void moveMotorZ2(long steps, boolean forward){
      byte Puerto=B00000000;
      // Motor Z2 uses bits 4 (direcc.) and 3 (step) del puerto C
      if(forward==true){
        Puerto=B00001000; //Pull direction pin low to move "forward"
      }else{
        Puerto=B00000000; //Pull direction pin high to move "backward"
      }

      for(long x=0; x<steps; x++)
      {
        PORTC=Puerto | B00000100; //Trigger one step forward
        delay(1);
        PORTC=Puerto & B00001000;
        delay(1);
      }

    }

void enableMotors(){
          // Motor enabled to move the cart
          digitalWrite(enaYPin, LOW);
          digitalWrite(enaEPin, LOW);
          digitalWrite(enaZ1Pin,LOW);
          digitalWrite(enaZ2Pin,LOW);
    }


    void restartMotorY(){
      long steps=0;
      if (digitalRead(begYPin)== HIGH){

        digitalWrite(enaYPin, HIGH);
        // while limit switches do not detect the cart presence
        while((digitalRead(begYPin)== HIGH)
```

```
      && (digitalRead(endYPin)== HIGH))
    { //&& steps<32000
      // Motor enabled to move the cart
      digitalWrite(enaYPin, LOW);
      // Small movement towards the beginning
      moveMotorY(16, 1);
      // Motor disabled again to read the limit switches
      digitalWrite(enaYPin, HIGH);
      delay(1);
      steps=steps+16;
    }
  }
  Serial.println("Reset hacia el motor en el eje Y");
  // Motor enabled to begin other measurement cycle
  digitalWrite(enaYPin, LOW);

}

void restartMotorE(){
  long steps=0;
  if (digitalRead(begEPin)== HIGH){

    digitalWrite(enaEPin, HIGH);
    // while limit switches do not detect the cart presence
    while(digitalRead(begEPin)== HIGH){ //&& steps<32000
      // Motor enabled to move the cart
      digitalWrite(enaEPin, LOW);
      // Small movement towards the beginning
      moveMotorE(4, 1);
      // Motor disabled again to read the limit switches
      digitalWrite(enaEPin, HIGH);
      delay(1);
      steps=steps+4;
    }
  }
  Serial.println("Reset hacia el motor en el eje E (Externo)");
```

```
  // Motor enabled to begin other measurement cycle
  digitalWrite(enaEPin, LOW);

}


void restartMotorZ(){
  long steps=0;
  long steps1=0;
  long steps2=0;

  /
  digitalWrite(enaZ1Pin, HIGH);
  digitalWrite(enaZ2Pin, HIGH);

  // While limit switches do not detect the y-bridge presence
  while((digitalRead(begZ1Pin)== HIGH)
     && (digitalRead(begZ2Pin)== HIGH)){

    // Enable motor to move
    digitalWrite(enaZ1Pin,LOW);
    digitalWrite(enaZ2Pin,LOW);

    // Small movement towards the beginning
    moveMotorZ(4, 1);


    digitalWrite(enaZ1Pin, HIGH);
    digitalWrite(enaZ2Pin, HIGH);
    delayMicroseconds(100);

    steps=steps+4;
  }

  steps1=restartMotorZ1();
  steps2=restartMotorZ2();
```

```
  // Motor enabled to begin other measurement cycle
  digitalWrite(enaZ1Pin,LOW);
  digitalWrite(enaZ2Pin,LOW);

  char respuesta[50];
  sprintf(respuesta,"%lu %lu %lu RSZ",steps,steps1,steps2);
  Serial.println(respuesta);
}

long restartMotorZ1(){
  long steps=0;


  digitalWrite(enaZ1Pin, HIGH);
  digitalWrite(enaZ2Pin, HIGH);

  // While limit switches do not detect the y-bridge presence
  while((digitalRead(begZ1Pin)== HIGH)
     && (digitalRead(endZ1Pin)== HIGH)){

    // Enable motor to move
    digitalWrite(enaZ1Pin,LOW);
    digitalWrite(enaZ2Pin,LOW);

    // Small movement towards the beginning
    moveMotorZ1(6, 1);


    digitalWrite(enaZ1Pin, HIGH);
    digitalWrite(enaZ2Pin, HIGH);
    delay(1);

    steps=steps+6;
  }
```

```
  // Motor enabled to begin other measurement cycle
  digitalWrite(enaZ1Pin,LOW);
  digitalWrite(enaZ2Pin,LOW);
  return(steps);

}

long restartMotorZ2(){
  long steps=0;


  digitalWrite(enaZ1Pin, HIGH);
  digitalWrite(enaZ2Pin, HIGH);

  // While limit switches do not detect the y-bridge presence
  while((digitalRead(begZ2Pin)== HIGH)
    && (digitalRead(endZ2Pin)== HIGH)){

    // Enable motor to move
    digitalWrite(enaZ1Pin,LOW);
    digitalWrite(enaZ2Pin,LOW);

    // Small movement towards the beginning
    moveMotorZ2(6, 1);


    digitalWrite(enaZ1Pin, HIGH);
    digitalWrite(enaZ2Pin, HIGH);
    delay(1);

    steps=steps+6;
  }

  // Motor enabled to begin other measurement cycle
  digitalWrite(enaZ1Pin,LOW);
  digitalWrite(enaZ2Pin,LOW);
```

```
  return(steps);
}


void finalizeMotorY(){
  long steps=0;


  digitalWrite(enaYPin, HIGH);
  delay(1);

  while((digitalRead(endYPin)== HIGH)){
    // Enable motor to move
    digitalWrite(enaYPin, LOW);

    // Small movement towards the end
    moveMotorY(4,0);


    digitalWrite(enaYPin, HIGH);
    delay(1);

    steps=steps+4;
  }

  digitalWrite(enaYPin, HIGH);
  delay(1);

  // If the cart is at the end
  if((digitalRead(begYPin)== HIGH)
    && (digitalRead(endYPin)== LOW)){
    // Enable motor to move
    digitalWrite(enaYPin, LOW);
    /
    moveMotorY(800,1);
```

```
    char respuesta[50];
    sprintf(respuesta,"%lu pasos hasta el final en Y",steps);
    Serial.println(respuesta);
  }

  // Motor enabled to begin other measurement cycle
  digitalWrite(enaYPin, LOW);
}

void finalizeMotorZ(){
  long steps=0;

  digitalWrite(enaZ1Pin, HIGH);
  digitalWrite(enaZ2Pin, HIGH);

  delay(1);

  // While limit switches do not detect the y-bridge presence
  while((digitalRead(endZ1Pin)== HIGH)
    && (digitalRead(endZ2Pin)== HIGH)){

    // Enable motor to move
    digitalWrite(enaZ1Pin,LOW);
    digitalWrite(enaZ2Pin,LOW);

    // Small movement towards the end
    moveMotorZ(4, 0);


    digitalWrite(enaZ1Pin, HIGH);
    digitalWrite(enaZ2Pin, HIGH);

    delay(1);

    steps=steps+4;
  }
```

```
  // Enable motor to move
  digitalWrite(enaZ1Pin,LOW);
  digitalWrite(enaZ2Pin,LOW);


  char respuesta[50];
  sprintf(respuesta,
    "%lu pasos hasta el final en Z (FNZ)",steps);
  Serial.println(respuesta);

}

void finalizeMotorE(){
  long steps=0;


  digitalWrite(enaEPin, HIGH);
  delay(1);

  while((digitalRead(endEPin)== HIGH)){
    // Enable motor to move
    digitalWrite(enaEPin, LOW);

    // Small movement towards the end
    moveMotorE(4,0);


    digitalWrite(enaEPin, HIGH);
    delay(1);

    steps=steps+4;
  }

  digitalWrite(enaEPin, HIGH);
  delay(1);
```

```
    // If the cart is at the end
    if((digitalRead(begEPin)== HIGH)
       && (digitalRead(endEPin)== LOW)){
      // Enable motor to move
      digitalWrite(enaEPin, LOW);

      moveMotorE(800,1);

      char respuesta[50];
      sprintf(respuesta,
      "%lu pasos hasta el final en E (eje externo)",steps);
      Serial.println(respuesta);
    }

    // Motor enabled to begin other measurement cycle
    digitalWrite(enaEPin, LOW);
  }
```

A.3. Archivo configuración de simulaciones Acoustic.cfg

```
[Images Recursion Depth]
# 4
N=4

[Physical parameters]
# Al calcular la fase de referencia
#, si empleo subaperturas con z>0, vale 1
subw_with_z_gt_0=1
# Al calcular la fase de referencia,
# si empleo subaperturas con z<=0, vale 1
subw_with_z_leq_0=0
# Exponente de pérdidas con la distancia
# desde el micro hasta la subapertura
```

```
exp_tx_subw=1
# Exponente de pérdidas con la distancia
#desde la subapertura hasta la RUT
exp_subw_rx=1
# Si incluyo el micro real como
# transmisor esto vale 1
include_real_tx=1

[Microphone data]
# Path of data file with mic measurements
input_data=./input_data/2016_07_21-22_32_26_5000Hz.csv
# Downsample the filtered signal by M; that is, keep only
#every Mth sample
mic_decimation=1
# Min z[m] value for the mic position (distance from the main wall)
z_mic_min=0.165

[Simulation results]
output_data=./output_data/2018_05_21-22_32_26_5000Hz_smr_v2.1.6.csv

[Source]
# Hertzs
frequency=5000.0
velocity=340.0
# Source position
# Apertures coordinates
Xs=0.55
Ys=-0.75
Zs=-1.85
#src_description=Apollada sobre la mesilla
#(ligeramente inclinada hacia arriba)

[Region Under Test]
# Number of sample points in the y direction (paralell to the aperture)
#250
Ny=125
```

```
# Number of sample points in the z direction
#(perpendicular to the aperture)
#200
Nz=100
# Limits of the sampling region
Yri=-2
Yrf=3
Zri=-0.2
Zrf=-4
Xr=0.55


[Cavity Facets]
# We need three points to define a plane
# Right facet
Ra=0,0,0
Rb=1,0,0
Rc=0,0,1
# Left facet
La=0,0.996,0.012
Lb=1,0.986,0
Lc=0,0.983,1
# Ceiling facet
Ua=1,0,0
Ub=1,1,0
Uc=1,0,1
# Ground facet
Da=0,0,0
Db=0,0,1
Dc=0,1,0
# Back facet
Oa=0,0,1
Ob=1,0,1
Oc=0,1,1.005
# Aperture facet
Ma=0,0,0
```

```
Mb=1,0,0
Mc=0,1,0

[Aperture Description]
# Anchura
w=0.39
# Altura
h=0.39
# Y coordinate of the right side (distance from the right corner
# to the frame, looking out from the inner part of the cavity aperture)
R=0.305
# X coordinate of the down side (heigth from the ground to the frame)
D=0.308
# Number of subapertures in the horizontal direction
Hor=3
# Number of subapertures in the vertical direction
Ver=3
```

Bibliografía

[1] ROLLER, D.E.;BLUM, R. *Física: volumen 1 (Mecánica, ondas y termodinámica)*, Ed. Reverté, Barcelona, 1983, cap. 18.

[2] ISALGUÉ, A., *Física de la Iluminación y el sonido*, Edicions UPC, Barcelona, 1995, cap.2.

[3] BEVAN,B.BAKER.; COPSON,E.T, *The mathematical theory Huygens principle*, Clarendon Press, Oxford, 1939.

[4] PYOTR YA UFIMTSEV , *Fundamentals of the Physical Theory of Diffraction*, 1st edition, Editor Wiley-Blackwell, 2007.