UNIVERSITAT POLITÈCNICA DE VALÈNCIA
ESCOLA POLITÈCNICA SUPERIOR DE GANDIA
Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

ESCOLA POLITÈCNICA
SUPERIOR DE GANDIA

"Estudio, diseño y construcción de un levitador acústico para desplazamiento de partículas"

TRABAJO FINAL DE GRADO

Autor/a:
Alejandro Del Pozzo Escalera
Tutor/a:
Jesús Alba Fernández
Romina María Del Rey Tormos

GANDIA, 2021

Índice de contenido

Índice de tablas

Índice de ilustraciones

1. Resumen

En este trabajo se presenta tanto el estudio, diseño y construcción de un levitador acústico uniaxial no resonante cuya frecuencia de trabajo sea de 40kHz, como su funcionamiento y aplicaciones. Además, se pretende demostrar la levitación de objetos pequeños y livianos aplicando voltajes de entrada reducidos.

Se justificarán tanto los conceptos físicos que interactúan para hacer posible la levitación junto a los procesos de toma de medidas realizados, como el montaje y la programación aplicada junto a su electrónica. Además, como objetivo final, se determinará la densidad máxima levitable de un objeto cuyo radio no supere λ/2, que en el caso de 40kHz se concreta en 0,0043m.

Abstract

This work presents both the study, design and construction of a non-resonant uniaxial acoustic levitator whose working frequency is 40kHz, as well as its operation and applications. In addition, it is intended to demonstrate the levitation of small and light objects by applying reduced input voltages.

The physical concepts that interact to make levitation possible together with the measurement-taking process carried out, as well as the assembly and the programming applied together with its electronics, will be justified. Furthermore, as a final objective, the maximum levitable density of an object whose radius does not exceed λ/2 will be determined, which in the case of 40kHz is specified in 0,0043m.

2. Introducción

En física, una onda es la propagación de una perturbación a través de un medio, donde se provoca una transmisión de energía sin desplazamiento de materia. Un levitador acústico es un dispositivo que, mediante la energía que transmiten las ondas de sonido, es capaz de mantener partículas suspendidas en el aire de forma estable contrarrestando la fuerza gravitatoria.

El diseño general de un levitador acústico consta de un emisor y un reflector que, a una determinada distancia, son capaces de crear una onda estacionaria en su interior, de modo que las partículas pueden quedar atrapadas en el centro de los nodos propios de la onda estacionaria, es decir, en sus puntos "nulos" de presión acústica.

La levitación acústica es un concepto muy estudiado a nivel global debido a las posibilidades que aporta en el sector de la química o la electrónica, donde pueden verse involucrados procesos sin contenedor o sin contacto.

Destacan los siguientes tipos de levitación física:

- **Levitación electroestática:** Para que ocurra, es necesario que el objeto a levitar esté cargado eléctricamente, de modo que al aplicarse un campo eléctrico con el potencial adecuado, se produzca una fuerza que sea capaz de contrarrestar la gravedad. Los sistemas de control requeridos para efectuar este tipo de levitación son complejos y las muestras de materiales levitables son limitadas.

- **Levitación magnética:** Este tipo de levitación mantiene a flote los objetos por la acción única de un campo magnético. Este tipo de levitación puede sostener firmemente materiales en aire, pero aquellos objetos levitados deben estar dotados de propiedades ferromagnéticas.

- **Levitación acústica:** Este es el fenómeno que se tratará en este documento. La ventaja de la levitación acústica frente a las anteriores es que los objetos levitados pueden prescindir de carga o propiedades eléctricas.

También se pueden destacar otros tipos de levitación como la aerodinámica, la óptica y la antigravedad (no demostrada).

2.1. Historia del levitador acústico

A raíz de la demostración del movimiento de las partículas debido a la acción de fuerzas acústicas, se planteó la posibilidad de la levitación acústica. Esta demostración se realizó en 1866 con los experimentos del tubo de Kundt. En el proceso se aplicaba cierta presión acústica dentro de una cámara resonante con partículas en su interior con el objetivo de crear una onda estacionaria, de modo que, debido a la acción de esta presión, las partículas se reunieran en los nodos. Sin embargo, el origen del experimento consistía en la posibilidad de medir la longitud de onda para demostrar la variación de la velocidad de propagación del sonido en un gas y no se centraba en la posibilidad de levitar objetos mediante presión acústica.

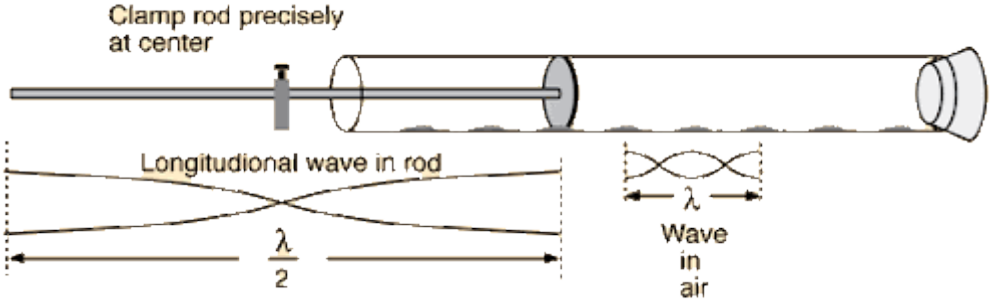


Ilustración 1: Tubo de Kundt[1]

La primera demostración de levitación fue ejecutada por los científicos Bücks y Muller en 1933, quienes consiguieron levitar gotas de alcohol entre un cristal de cuarzo y un reflector, gracias a las propiedades piezoeléctricas del cuarzo[2].

En 1962, el físico ruso Lev P. Gor'kov publicó el artículo "*On the forces acting on a small particle in an acoustic field in an ideal fluid*" donde se desarrollan formulas referentes al campo acústico y las fuerzas que actúan sobre una partícula que se encuentra en su interior[3].

En 1987, el doctor David Deak diseñó y construyó para la NASA una cámara de levitación acústica como un experimento para mostrar los efectos de la microgravedad que existe en el entorno de órbita de un transbordador espacial, pero reproducido en la tierra. La cámara consta de 3 altavoces y fue diseñada basándose en la cavidad resonante de Helmholtz. Mediante este experimento, fue la primera vez que se consiguió controlar el movimiento tridimensional de las partículas en el campo acústico debido a cambios de fase y de amplitud de la onda[4].

En 1992 fué posible demostrar la levitación acústica de seres vivos gracias al estudio realizado por Wen Jun Xie, en la *Northwestern Polytechnical University*. En este estudio se levitaron pequeños insectos como hormigas, mariquitas y pequeños peces para comprobar si este tipo de levitación podía tener efectos negativos en seres vivos. La vitalidad de los insectos probados no se vio influenciada notablemente, mientras que la de los peces jóvenes solo se vio afectada debido a la insuficiencia de agua[5].

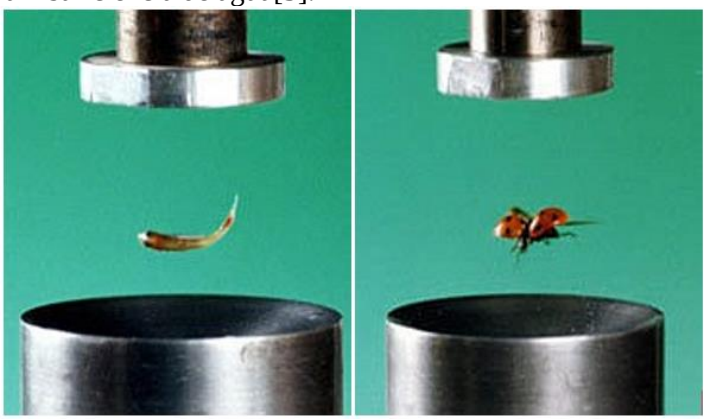


Ilustración 2: Levitación de seres vivos[6]

Más recientemente, en 2014 se publicó un artículo de investigación realizado por los científicos Yoichi Ochiai, Takayuki Hoshi y Jun Rekimoto en la universidad de Tokio titulado "*Three-Dimensional Mid-Air Acoustic Manipulation by Ultrasonic Phased Arrays".* En este artículo se propone la manipulación y control de las partículas mediante matrices de transductores que generan ondas estacionarias localizadas, a diferencia de los levitadores clásicos, que limitaban el movimiento de las partículas a un eje dimensional mediante cambios en la fase de la onda.

Con este método se consigue la manipulación de partículas de entre 0,3 y 1 mm de radio. Además, es posible ajustar la resolución espacial hasta 0,5mm cambiando la distribución de campo del potencial acústico.[7]

El ingeniero español Asier Marzo Pérez de la universidad pública de Navarra, junto a investigadores de la universidad de Sussex y Bristol, desarrollaron en 2015 el diseño del primer rayo tractor sónico. Este dispositivo permite, mediante ondas de sonido, atraer una partícula hacia la fuente, al contrario de los sistemas clásicos de levitación, donde los transductores repelen la partícula hacia el exterior para contrarrestar la gravedad. Este método presenta un tipo de levitación más práctica para ser usada en otros campos y más manipulable ya que prescinde de reflector[8].

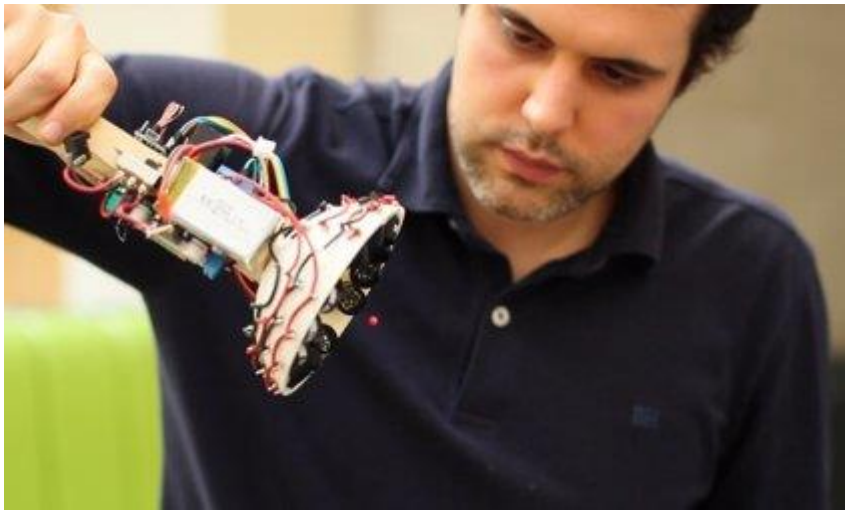


Ilustración 3: Rayo tractor sónico[8]

2.2. Aplicaciones de la levitación acústica

El desarrollo del levitador acústico ha sido impulsado mayoritariamente por las distintas aplicaciones que puede tener en ciertos campos científicos de investigación y producción.

La levitación acústica puede ser de utilidad en el campo de la química, ya que es posible la cristalización de elementos químicos sin que adapten la forma del contenedor, además de evitar la contaminación debida al contacto.

Se considera que este fenómeno físico tendrá mucha utilidad en el campo de la medicina, ya que se podrían adaptar los dispositivos en la realización de ecografías para poder mover piedras en el riñón o partículas dentro de los ojos. También se ha pensado en la posibilidad de trasportar elementos microquirúrgicos en el tejido humano o incluso capsulas de medicamentos.

En el campo del ensamblaje se podría implementar la levitación acústica para manipular y montar objetos delicados sin la necesidad de tocarlos. Además, es posible suspender pequeños componentes electrónicos que puedan ser dañados debido a la carga inducida por electricidad estática en la manipulación, o elementos microópticos que puedan ser rallados o dañados debido al contacto. También es posible la solidificación de objetos mediante suspensión, de modo que se podría aumentar la precisión en la manipulación para poder crear elementos electrónicos.

Estas son solo unas pocas de las múltiples aplicaciones que la levitación acústica puede alcanzar, pero según se desarrolle esta tecnología, aparecerán más aplicaciones y se crearán nuevos campos donde este fenómeno sea de utilidad[9-12].

2.3. Tipos de levitador

Debido a la evolución que ha sufrido esta tecnología, se han desarrollado distintos tipos de levitadores capaces de suspender objetos con finalidades distintas o simplemente con el objetivo de mejorar antiguos diseños funcionales.

La configuración más común en los levitadores acústicos es la de un solo eje, donde destacan dos tipos de diseño clásico de levitador acústico:

1. **Levitador acústico resonante:** este tipo de levitador se compone de un emisor compuesto por uno o varios transductores y un reflector cuya distancia de separación equivalga a un múltiplo de media longitud de onda, de modo que se crea una onda estacionaria en su interior debido a las múltiples reflexiones del sonido. El reflector debe ser un sólido de complexión plana o cóncava. Este tipo de configuración es más efectiva, pero es más sensible a cambios de temperatura y de posición.

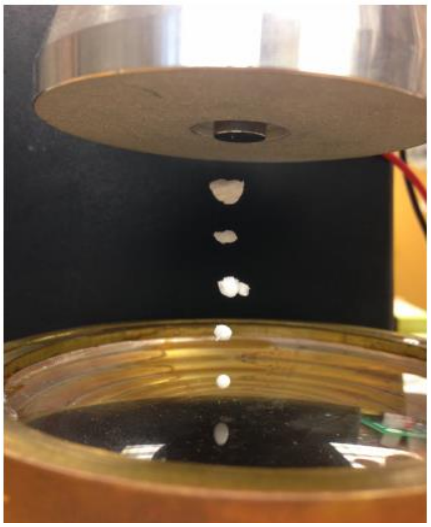


Ilustración 4:Levitador acústico resonante[13]

2. **Levitador acústico no resonante:** A diferencia del diseño anterior, este levitador se compone de dos emisores contrapuestos, de modo que emiten la misma señal en fase para crear una onda estacionaria. Este tipo de diseño es el utilizado en el desarrollo de este proyecto. Al igual que el anterior, puede componerse de uno o varios transductores y pueden estar posicionados de forma plana o cóncava. Además, también es necesario que la separación entre las caras emisoras sea un múltiplo de media longitud de onda.

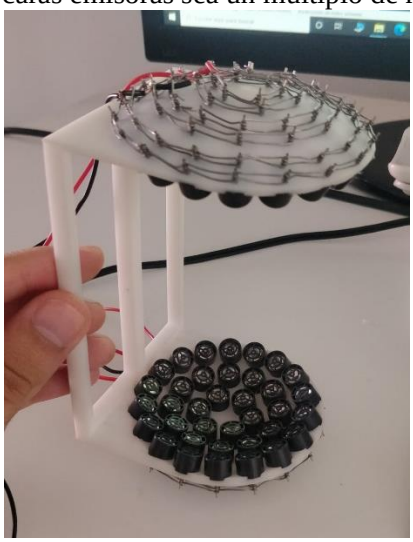


Ilustración 5:Levitador acústico no resonante(objeto de este trabajo)

La ambición por la mejora y la búsqueda de aplicaciones de los levitadores acústicos ha empujado a científicos a desarrollar modelos que van más allá de los diseños clásicos. Algunos ejemplos de diseños alternativos se muestran a continuación:

- **Levitador acústico de 3 ejes:** En los ejemplos anteriores, la levitación de las partículas está limitada a un solo eje, donde es posible efectuar un desplazamiento vertical mediante el cambio de fase de la onda. Este tipo de levitador aplica el mismo principio de cambio de fase para desplazar las partículas en los ejes OX, OY y OZ. Está compuesto por 4 placas planas de transductores ultrasónicos contrapuertas entre sí formando un cuadrilátero, en cuyo centro se crea un campo acústico capaz de levitar las partículas. Es posible manipular la posición de las partículas mediante cambios de fase a través de un proceso computacional[14].

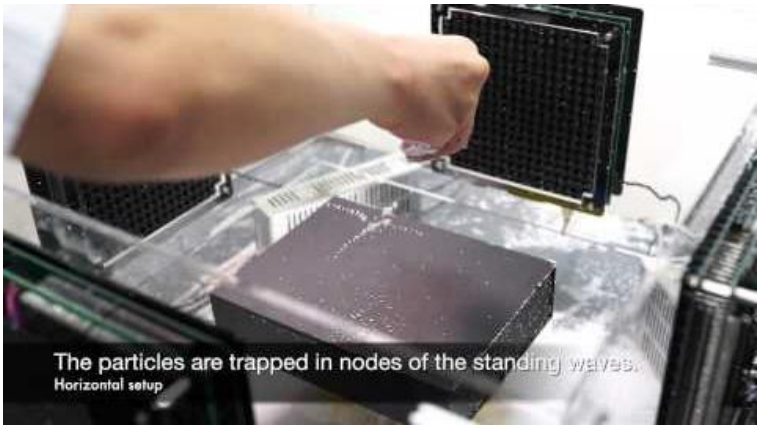


Ilustración 6: Levitador acústico de 3 ejes[14]

- **Pinza acústica holográfica:** Este modelo de levitador acústico desarrollado por el ya mencionado ingeniero Asier Marzo junto a Bruce W. Drinkwater, es capaz de levitar y manipular partículas en los 3 ejes OX, OY y OZ de forma independiente. Está estructurado en dos bases planas compuestas por múltiples transductores, paralelas entre sí y perpendiculares al eje vertical. En este modelo es posible efectuar el movimiento de cada partícula de forma independiente al resto, a diferencia de los modelos anteriores, donde las partículas se mueven conjuntamente ya que quedan atrapadas en los nodos de una misma onda estacionaria[15].

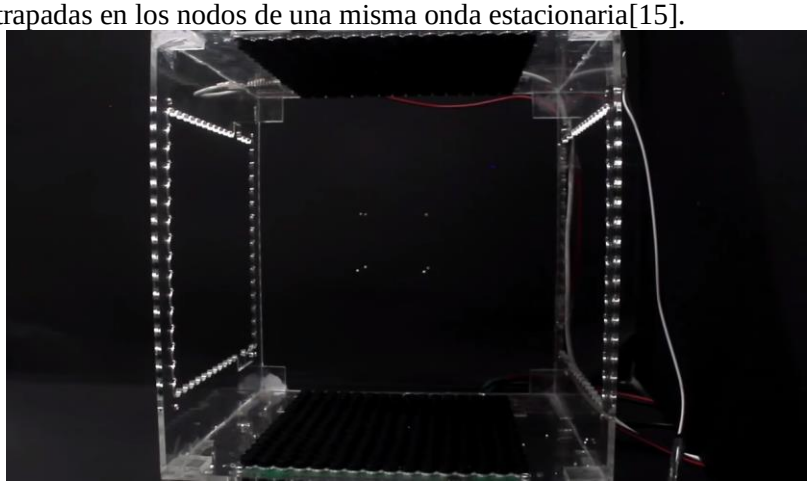


Ilustración 7: Pinza acústica holográfica[15]

- **Rayo tractor sónico:** Los modelos explicados anteriormente, son capaces de levitar objetos por la acción repulsiva de la fuerza acústica en los nodos de las ondas estacionarias. A diferencia de estos modelos, este dispositivo, desarrollado también por Asier Marzo, es capaz de atraer las partículas levitadas hacia el emisor, sin la necesidad de un reflector. El dispositivo está formado por una base cóncava compuesta por varios transductores que emiten a 40kHz con la misma amplitud, de modo que las interferencias de las ondas emitidas crean un campo acústico tridimensional capaz de atrapar y atraer las partículas hacia el emisor. [16]

Ilustración 8: Rayo tractor sónico(ampliado)[16]

3. Objetivos del proyecto

En este trabajo se pretende realizar un estudio sobre el concepto de levitación acústica mediante un dispositivo que efectúe este fenómeno físico. Los objetivos a cumplir en este proyecto son los siguientes:

- Montar un levitador acústico uniaxial no resonante mediante los componentes necesarios y de forma estable.
- Programar el microprocesador Arduino con un código funcional que sea capaz tanto de levitar objetos como desplazarlos en el eje OZ.
- Entender y explicar los fenómenos físicos involucrados en la levitación por medio de ondas ultrasónicas.
- Realizar las medidas necesarias para demostrar la teoría.
- Calcular la densidad máxima levitable de un objeto no superior a media longitud de la onda emitida por el dispositivo.

4. Montaje experimental de un levitador acústico no resonante

Una de las ventajas de los levitadores acústicos clásicos es que su montaje es relativamente sencillo una vez se tienen los materiales y las herramientas necesarias para ensamblarlos. A continuación, se muestran los materiales utilizados para el diseño del levitador objeto de este trabajo, así como el proceso de montaje.

4.1. Materiales utilizados y herramientas necesarias

- Estructura 3D del levitador:
  El diseño de esta estructura está disponible en internet para ser impresa mediante una impresora 3D.

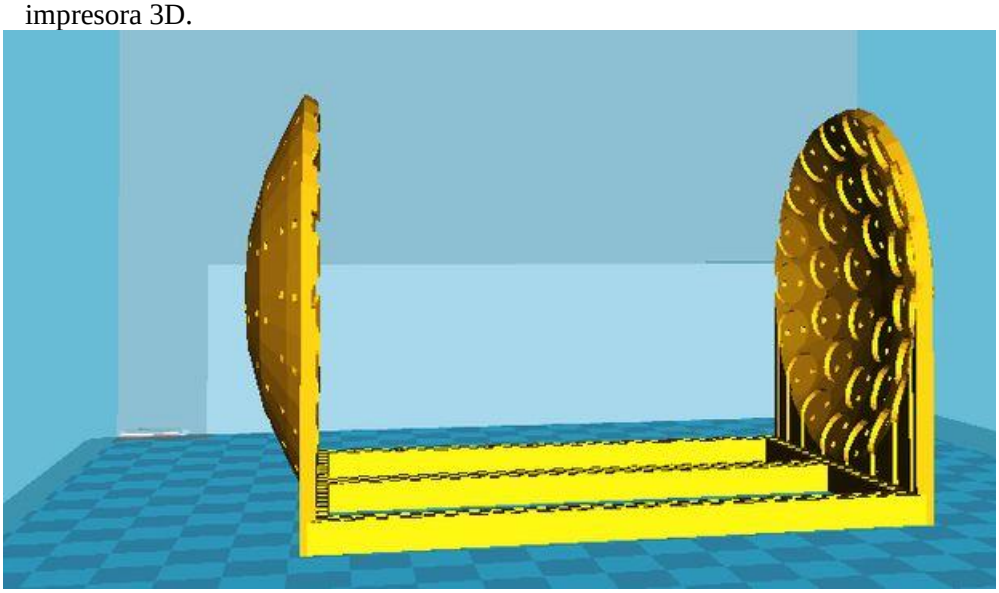


Ilustración 9: Diseño 3D levitador acústico

- 72 Transductores MCUSR10P40B07RO:
  Estos componentes son transductores ultrasónicos piezoeléctricos. Son los elementos que emitirán las ondas de ultrasonidos para hacer levitar los objetos.
  Sus características técnicas son las siguientes [17]:
  - Temperatura de trabajo: -35º a 85ºC
  - Directividad: 50º

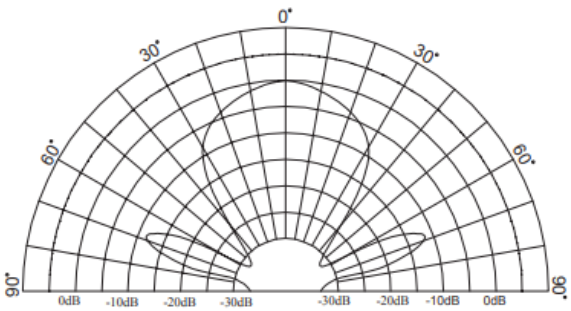


Ilustración 10: Diagrama de directividad del transductor

Ilustración 11:Transductores MCUSR10P40B07RO

- Dimensiones:

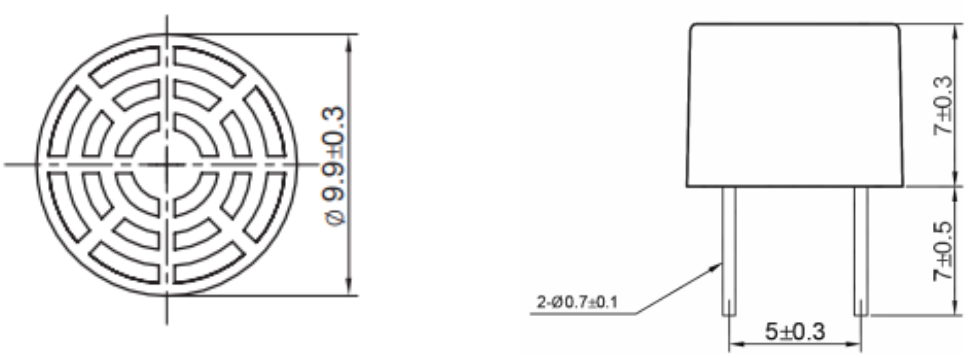


Ilustración 12: Planos y medidas del transductor

- Frecuencia central: 38,3 ± 1 kHz
  Sensibilidad a 40kHz: -70dB/V/µbar

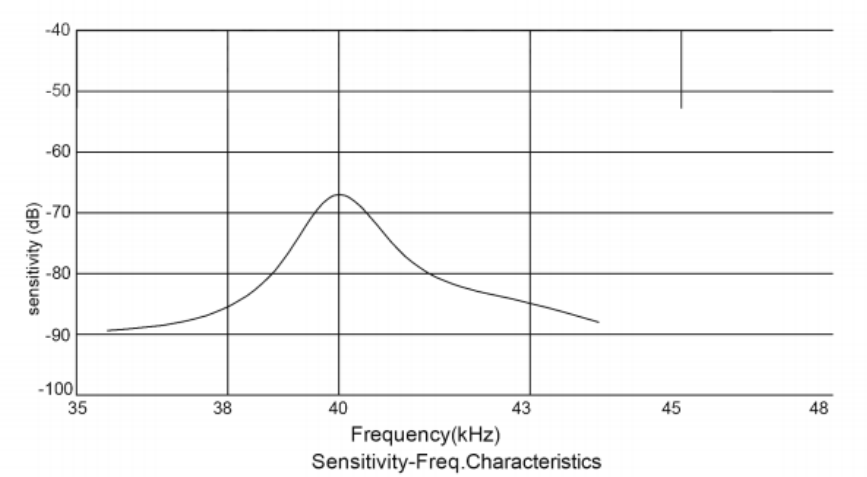


Ilustración 13:Respuesta en frecuencia del transductor

- Arduino Nano[18]:
  Debido al tamaño y a las funcionalidades, este microcontrolador Arduino es más que suficiente para realizar las funciones lógicas que se necesitan para generar una señal de 40kHz. Sus especificaciones técnicas son las siguientes:
  - Microcontrolador: Atmel ATmega328p
  - Frecuencia de reloj: 16MHz
  - Voltaje de trabajo: 5V
  - Voltaje de entrada:7-12V
  - Nº de entradas analógicas: 8
  - Nº de pines digitales: 14
  - Nº de pines PWM: 6
  - Intensidad máxima en pin: 40mA
  - Dimensiones: 48x18mm

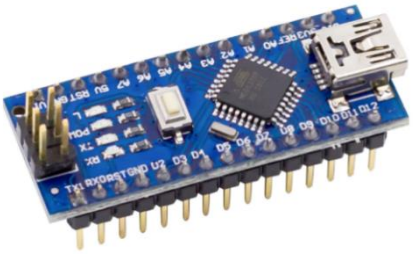


Ilustración 14: Arduino Nano

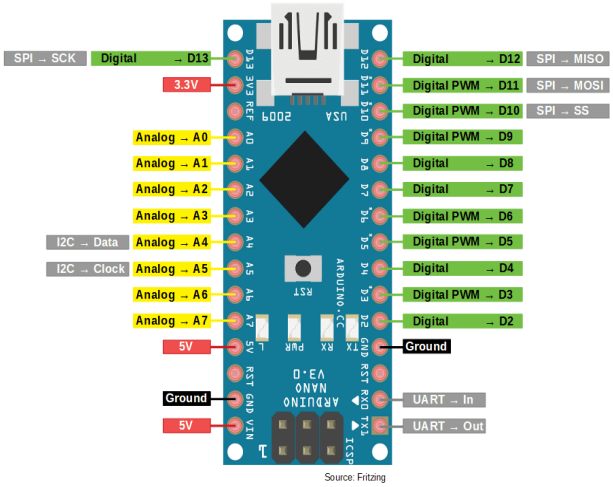


Ilustración 15: Arduino Nano Pinout [19]

- <u>Módulo Driver L298N[20]:</u>
  Este módulo consta de un circuito integrado L293N. En su interior contiene dos puentes en H que amplifican la señal con la que se alimentan los transductores.
  - Potencia máxima: 25 W
  - Capacidad de corriente: 2A
  - Consumo de corriente: 0 a 36mA
  - Voltaje de entrada: 5-12V
  - Canales: 2
  - Dimensiones: 43x43x27mm

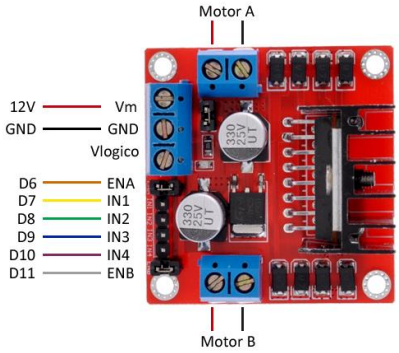


Ilustración 16: Puertos Módulo Driver L198N [20]

- <u>Diferentes tipos de cable e interruptor:</u>
  Cable negro y rojo, cables Dupont macho-macho, cables Dupont macho-hembra, cables Dupont hembra-hembra.

 *Ilustración 17: Cable negro y rojo*	 *Ilustración 18: Dupont macho-macho*	 *Ilustración 19: Dupont macho-hembra*	 *Ilustración 20: Dupont hembra-hembra*	 *Ilustración 21: Interruptor*

Tabla 1: Cables e interruptor

Las herramientas necesarias son:

- Impresora 3D (es posible utilizar servicios de impresión 3D online)
- Soldador con estaño
- Osciloscopio
- Fuente de alimentación regulable
- Destornillador
- Alicates
- Pistola de pegamento térmico
- Multímetro digital

4.2. Coste total del Proyecto

Materiales	**Precio**
Estructura 3D del levitador	Servicio Online: 4-10€ (+gastos de envío) Impresión personal: 2€ (aprox.)
Transductores MCUSR10P40B07RO	2,40€ * 72 = 172,8€
Arduino Nano	4,21€
Módulo Driver L298N	4,95€
Cables	7€
Interruptor	1,95€
Precio Total	**192,11 – 200,11**€ (+ gastos de envío Estructura 3D)

Tabla 2: Precio total del dispositivo

4.3. Proceso de montaje

1. Para empezar, se deben pegar los transductores a la estructura 3D mediante el pegamento térmico. Hay que tener en cuenta la polarización marcada en ellos, ya que deben colocarse con el polo positivo orientado al centro de la estructura.

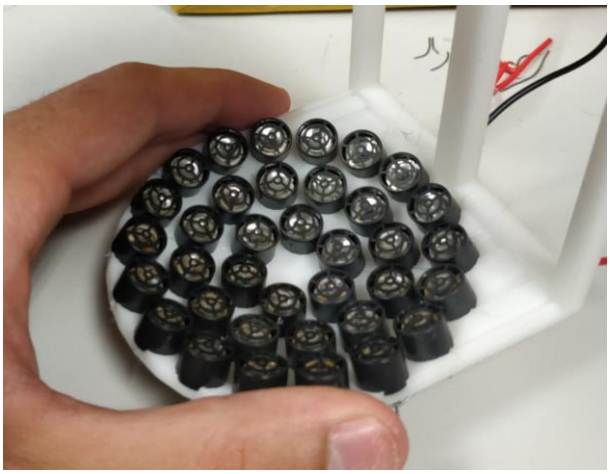


Ilustración 22: Instalación de los transductores en la estructura

2. Seguidamente, se deben unir los transductores en las dos bases mediante algún tipo de hilo conductor, en el caso de este proyecto, se ha utilizado estaño para

unir los bornes de los transductores. Se deben entrelazar por anillos de transductores, conectando entre sí los bornes de la misma polaridad, realizando una conexión en paralelo tal y como se muestra a continuación:

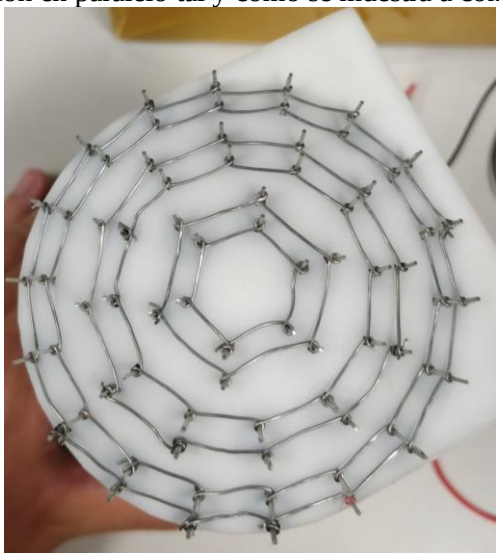


Ilustración 23: Interconexión de los bornes de cada capa

3. A continuación, se deben interconectar los anillos de transductores soldando los cables que irán conectados al módulo. El cable rojo se unirá a los anillos cuyos bornes sean positivos y el cable negro en aquellos negativos. Este conexionado alimentará todos los transductores por igual.

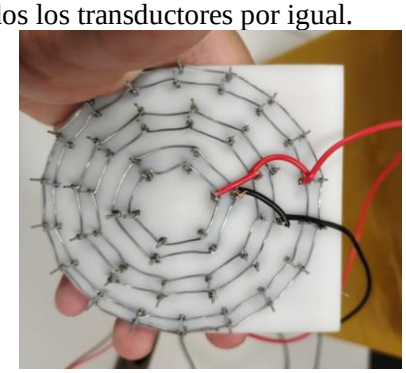


Ilustración 24: Interconexión entre capas

4. Seguidamente, se debe instalar el código en el controlador Arduino conectándolo por USB al ordenador.

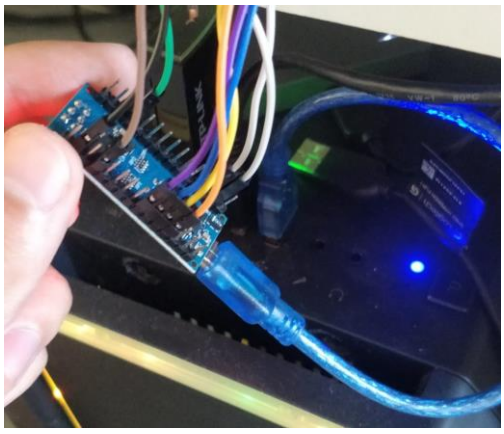


Ilustración 25: Flasheo del programa en Arduino Nano

5. El conexionado de cables para interconectar el módulo con el Arduino y la estructura 3D sigue el siguiente mapa:

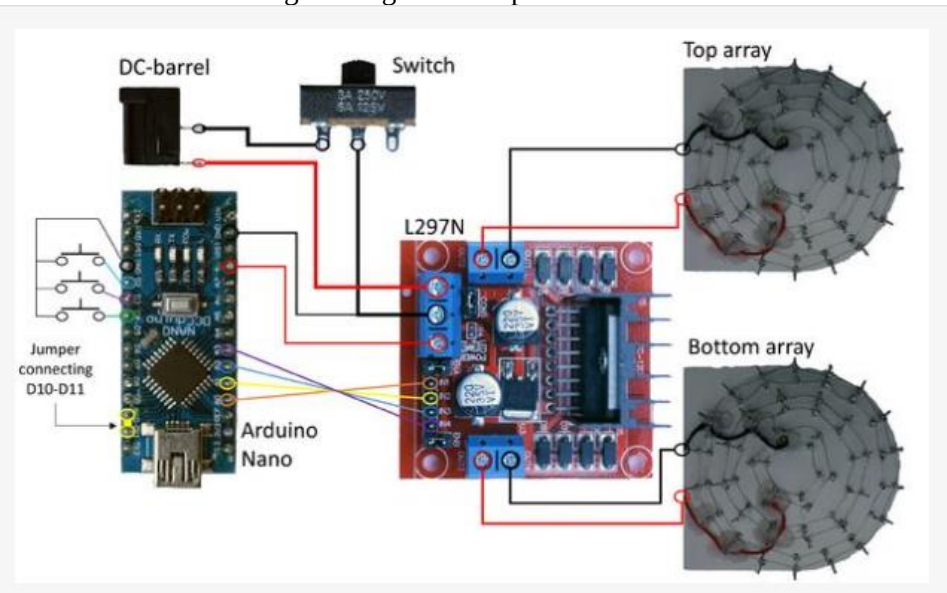


Ilustración 26: Mapa de conexiones del dispositivo[21]

- El Arduino es alimentado por el módulo conectándolo a la salida de tensión de 12V y al pin de tierra como aparece en el mapa.
- Los pines de A0 a A3 van conectados a los pines del módulo de A1 a A4 respectivamente.

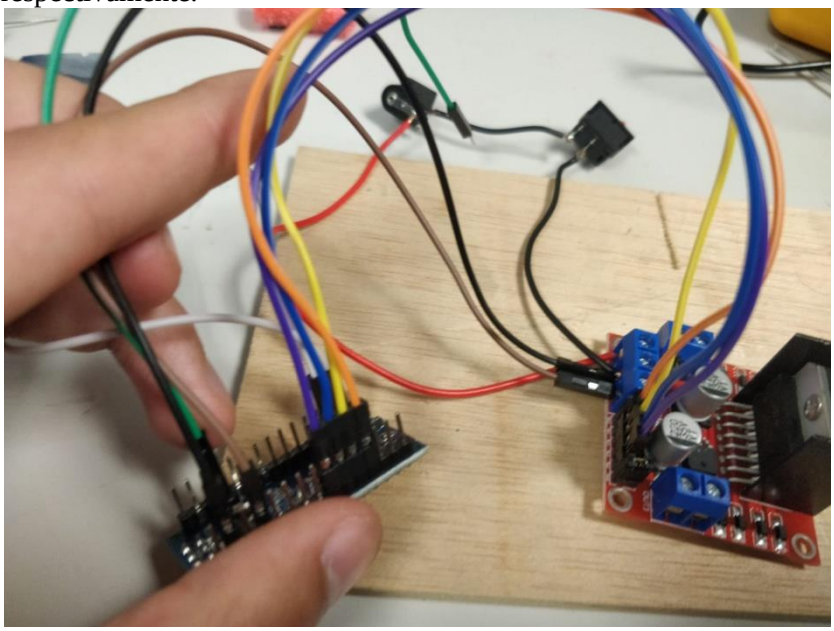


Ilustración 27: Pines conectados de A0-A3 a A1-A4

- Es necesario interconectar los Pines D10 y D11 mediante un Jumper o, en su defecto, un cable Dupont.
- En el mapa aparecen las salidas D2 a D4 conectadas al pin de tierra mediante pulsadores. En vez de utilizarlos, basta con conectar un cable Dupont hembra-macho al pin de tierra, de modo que se pueda establecer una conexión entre los pines mencionados haciendo contacto con el borne macho.
- El interruptor va conectado a la entrada de tierra en el módulo.

- Se debe conectar el cable positivo a la entrada de tensión del módulo y el cable negativo al interruptor. En este proyecto, no se utiliza el conector DC ya que los cables van conectados a una fuente de tensión regulable para ajustar la entrada de voltaje.
- Los cables soldados en la estructura se conectan a las salidas del módulo. Es importarte corroborar mediante un multímetro que la tensión sea positiva en las dos salidas para que los transductores estén en igualdad de fase.

6. Una vez realizados los pasos anteriores, es necesario probar que todos los transductores están emitiendo a la misma potencia y fase. Para ello se debe conectar un transductor a un osciloscopio y enfocarlo sobre cada uno de los transductores ya instalados mientras estén emitiendo. Cabe destacar que no existe la misma amplitud en las salidas del módulo. Esto es debido a que la salida que emita mayor tensión será aquella que actúe de emisor, por lo que la base que esté conectada a esta salida se situará en la parte inferior, ya que contrarrestará mejor la gravedad, mientras que la otra base actuará de reflector.

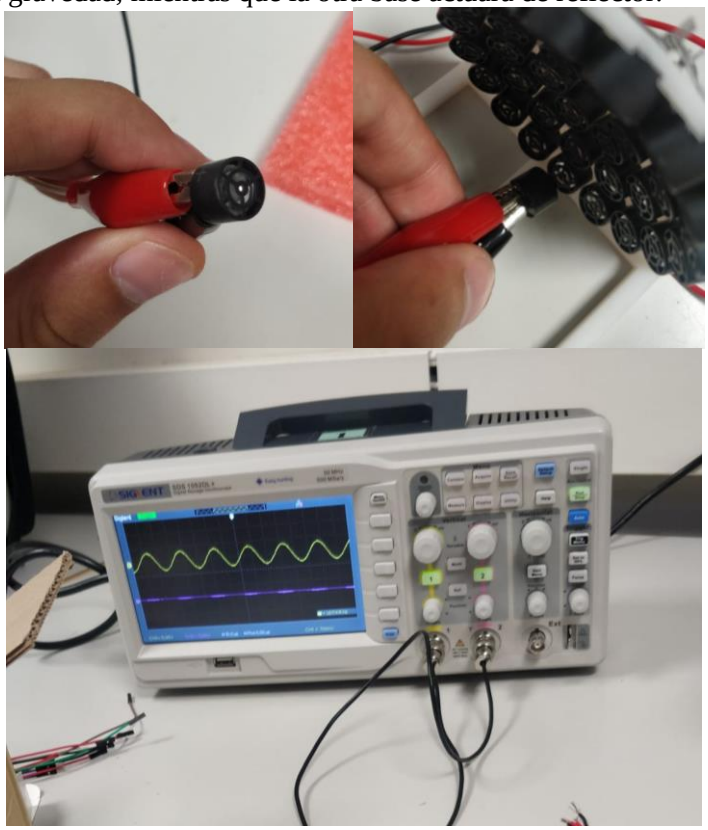


Ilustración 28: Test de funcionamiento y fase

7. Si algún transductor está emitiendo a una amplitud menor, puede ser debido a un problema en el conexionado. En ese caso es necesario revisar que no queden restos de pegamento en los bornes del transductor ya que pueden disminuir la conectividad. También es posible reforzar la conexión añadiendo un segundo cable en los bornes o atándolo con varias vueltas. Lo más recomendable es utilizar un hilo conductor que pueda soldarse sin fundirse, pero debido a la falta de este material, en el proyecto se utilizaron hilos de estaño como se ha mencionado anteriormente.

8. Si todo está implementado correctamente, a este punto el levitador debe funcionar. El voltaje aplicado no debe superar los 12V ya que es la tensión máxima que admite el Arduino.

5. Programación

Una vez montado el levitador, es necesario dotarlo de ciertas funciones para que emita ultrasonidos a una frecuencia concreta. Para ello se debe implementar un código en la placa Arduino Nano que cumpla dichas funciones.

Para comenzar a redactar el código, es necesario utilizar el propio IDE (Entorno de Desarrollo Integrado) de Arduino, ya que está capacitado para entender su propio lenguaje de programación y poder "flashearlo" en la placa.

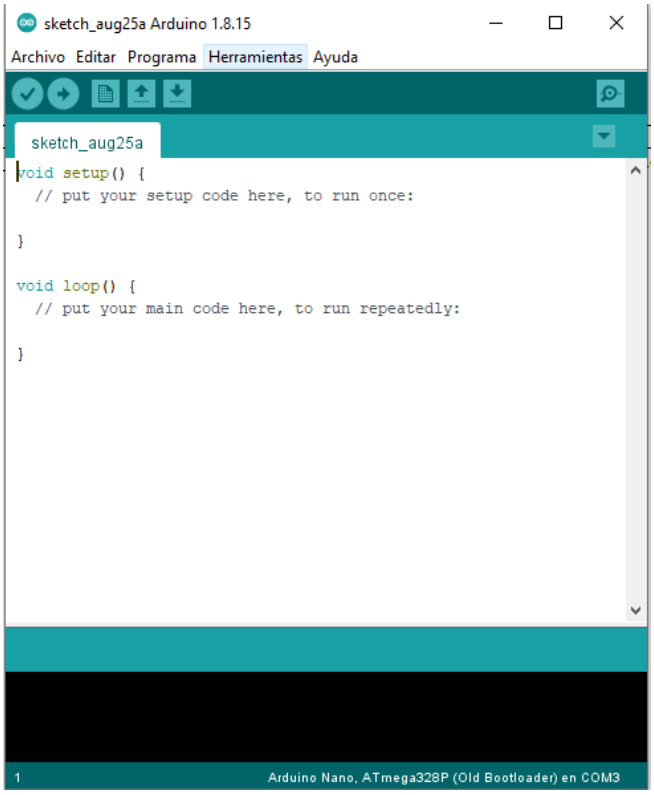


Ilustración 29: Sketch Arduino

Un código programado y diseñado para ejecutarse en Arduino se denomina "*sketch*". Un *sketch* se divide en 3 áreas de programación:

- **Declaración de variables globales**: Se ubica en el inicio del *sketch* y sirve para declarar variables comunes para las 3 secciones del *sketch*.
- **void setup()**: Aquellas instrucciones ubicadas en esta sección serán ejecutadas una única vez. En esta sección se realizan las configuraciones iniciales
- **void loop()**: Las instrucciones ubicadas en esta sección se ejecutarán continuamente hasta que la placa sea desconectada.

5.1. Área de declaración de variables

Para empezar se deben añadir las siguientes librerías al código:

- ***avr/sleep.h*** : esta librería da la opción de dejar la *CPU* en suspensión mientras los relojes periféricos funcionan, de modo que se reduce el consumo de energía considerablemente.
- ***avr/power.h*** : esta librería otorga un registro de reducción de energía que permite deshabilitar o habilitar varios periféricos integrados para reducir también el consumo de energía. Los macros para deshabilitar o habilitar estos periféricos se configuran en el *void setup()*.

Seguidamente, se declaran las constantes del proyecto mediante la función *#define*. Estas constantes no consumen memoria del programa en el controlador.

```
#define N_PORTS 1 //número de puertos con los que se trabaja
#define N_DIVS 24 //cantidad de divisiones en la matriz

#define WAIT_LOT(a) __asm__ __volatile__ ("nop"); __asm__ __volatile__ ("nop"); __asm__ __volat
#define WAIT_MID(a) __asm__ __volatile__ ("nop"); __asm__ __volatile__ ("nop"); __asm__ __volat
#define WAIT_LIT(a) __asm__ __volatile__ ("nop"); __asm__ __volatile__ ("nop"); __asm__ __volat
//variables que añaden retrasos en los ciclos de reloj: LOT (mucho retraso), MID (retraso medic
```

Las constantes declaradas son las siguientes:

- ***N_PORTS***: Esta constante define el número de puertos. Con esta constante se trabajará en la función macro más adelante.
- ***N_DIVS***: Esta constante define la cantidad de divisiones en la matriz, entrará en contexto más adelante cuando se expliquen las matrices del programa.
- ***WAIT_LOT***: Se define esta constante para generar tiempos de espera largos.
- ***WAIT_MID***: Se define esta constante para generar tiempos de espera medios.
- ***WAIT_LIT***: Se define esta constante para generar tiempos de espera cortos.

Las últimas 3 constantes están compuestas por varias palabras clave que generan retrasos en el sistema para mejorar el flujo de las instrucciones.

La palabra *__asm__*, además de indicar que el resto de la línea es una instrucción de lenguaje ensamblado, llama al ensamblador alineado, y permite insertar directamente en programas de lenguaje C (como Arduino) instrucciones de lenguaje ensamblado, prescindiendo de un ensamblador independiente y evitando realizar pasos adicionales de ensamblado. Dentro de la palabra *__asm__* se encuentra la instrucción *"nop"*. Esta instrucción, aparentemente no hace nada, simplemente genera un tiempo de demora realizando un ciclo sin operar.

Normalmente, cuando un programa ejecuta funciones pasivas como las mencionadas, el propio compilador, al tratar de optimizar el código, las elimina. Para evitar esto se utiliza la palabra clave *__volatile__* junto aquellas funciones que se quieran mantener, en este caso *__asm__* y *("nop")*.

El conjunto de estas instrucciones genera pequeñas demoras en el sistema equivalentes a un ciclo máquina, por tanto, cuantas más instrucciones formadas por el conjunto *__asm__ __volatile__ ("nop")*, más demora se aplicará en el sistema.

A continuación, se crea la siguiente función macro:

```
#define OUTPUT_WAVE(pointer, d)   PORTC = pointer[d*N_PORTS + 0]
```

La diferencia entre un macro común y una función macro es que la función permite operar con valores de retorno y parámetros. En este caso, los valores de retorno son *(pointer, d)*. Estos valores serán asignados en diferentes partes del código para ser procesados por la función macro.

La variable *PORT_C* entrará en contexto más adelante en el código.

Las siguientes constantes se definen a continuación en esta sección del código, pero tomaran contexto más adelante cuando se les dé uso.

```
#define N_BUTTONS 6
//half a second
#define STEP_SIZE 1
#define BUTTON_SENS 2500
#define N_FRAMES 24
```

Tras definir las constantes anteriores, se declara una variable "*static*". Esta configuración implica que la variable no desaparezca a lo largo de la ejecución del programa. Al ser de tipo byte, su rango de valor se extiende de 0 a 255 y ocupará un byte de memoria (8bits).

A continuación se define una matriz estática de *arrays*, que están compuestos por valores de tipo byte. Se observa que junto a la variable se están definiendo dos parámetros, *[N-FRAMES*] y *[N-DIV]*. El primero define el número de divisiones que tendrá la matriz, el segundo define el número de subdivisiones que tendrá cada división de la matriz. A raíz de esto se entiende que

dicha matriz en realidad es un array de *arrays*(*subarrays*). El array estará compuesto por 24 *subarrays* de 24 elementos cada uno, por lo que toma forma de matriz.

```
static byte frame = 0;
static byte animation[N_FRAMES][N_DIVS] =
{{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x5,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0xa},
{0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0x6,0x6,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6},
{0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x5,0x9,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0xa,0x6}};
```

Los valores contenidos en los *subarrays* están definidos mediante el formato hexadecimal, que sigue la estructura “0x_”.

5.2. void setup()

En esta sección del código se desarrolla todo el programa. El microcontrolador utilizado en este proyecto se denomina ATmega328p. Este microcontrolador dispone de registros vinculados con los puertos del Arduino.

Es necesario definir qué puertos se utilizarán y de qué tipo serán, si de entrada o de salida. Esto se realiza mediante el registro DDRX, donde X corresponde a la letra asociada al puerto que se desea configurar. En este proyecto, se utiliza el puerto C, por lo que el registro quedará como DDRC. Para indicar que un puerto es de salida, se añade un 1 en el pin deseado, por tanto, teniendo en cuenta que el puerto C dispone de 6 pines, la instrucción será la siguiente:

```
DDRC = 0b00001111;
```

Debido a que un registro consta de 8bits, es necesario añadir todos los valores a pesar de que el puerto conste solo de 6 pines, siempre y cuando esta instrucción se indique en formato bit (0b_). En el esquema siguiente se puede comprobar que los puertos configurados como salida son del A0 al A3, los cuales van conectados con el módulo driver para controlarlo.

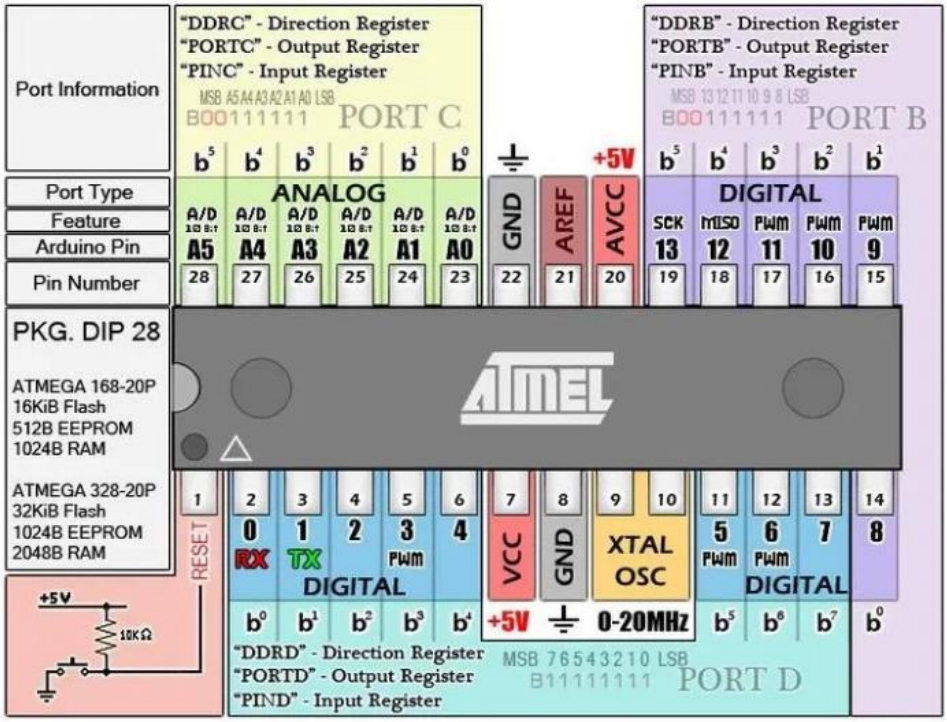


Ilustración 30: Mapa de puertos ATmega328p[22]

Seguidamente, se debe determinar si los pines del puerto C son de nivel alto o nivel bajo. Para ello se hace uso del registro *PORTC*.

Para indicar si un pin es de nivel alto, se le asigna un 1, para indicar si es de nivel bajo, se asigna un 0. En este caso, el nivel predeterminado de los pines del puerto C será de nivel bajo:

```
PORTC = 0b00000000;
```

Durante el montaje del dispositivo, se realiza un conexionado entre los pines 10 y 11 mediante un *jumper* (paso 5 del proceso de montaje). El pin 10 se configura como salida de la señal de sincronismo de 40kHz y el 11 como entrada. A diferencia de los formatos de configuración anteriores, se ha determinado la función de entrada y la salida de los pines haciendo uso de la instrucción *pinMode()*. Ha sido conveniente utilizar este formato ya que permite activar resistencias de *PULL-UP* internas de 20kΩ haciendo uso del modo *INPUT_PULLUP*.

```
pinMode(10, OUTPUT);
pinMode(11, INPUT_PULLUP);
```

Las siguientes líneas de código sirven también para configurar pines como entrada o como salida. Para ello se ha creado un bucle *for* que recorre los pines del número 2 al 7 mientras aplica la configuración. Estos pines servirán de entrada para los botones utilizados para efectuar un cambio de fase en el campo acústico del levitador y así poder desplazar la partícula en el eje OZ. Es necesario que las entradas se configuren con la resistencia de *PULL-UP* activada para evitar flancos producidos por ruido en el contacto, por eso se ha optado por usar *pinMode()* en vez de *DDRX*.

```
for (int i = 2; i < 8; ++i){
 pinMode(i, INPUT_PULLUP);
}
```

En este proyecto, los botones se han sustituido por un cable Dupon macho-hembra que haga contacto entre los pines de entrada y el de tierra.

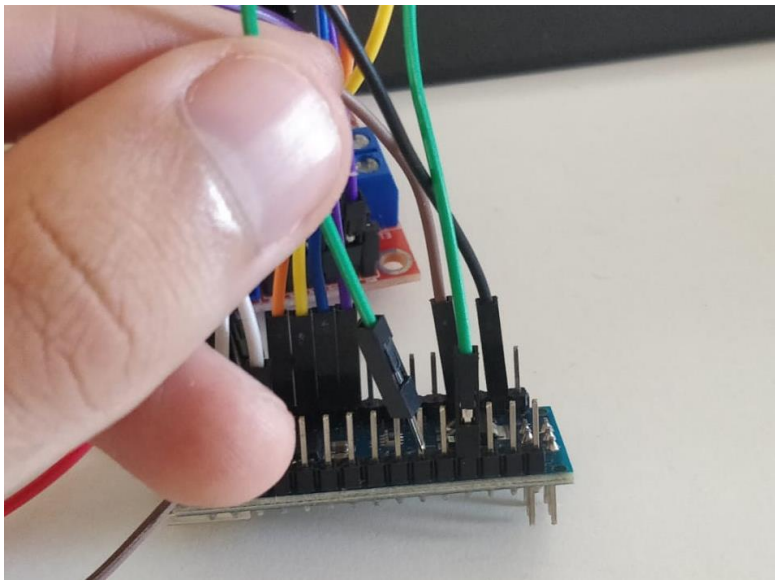


Ilustración 31: Conexión pines de entrada a tierra

A continuación, se presenta el tramo de código que genera la señal de sincronismo de 40kHz. Para comprender todas las instrucciones de este tramo, es necesario explicar algunos conceptos.

Para generar la señal de sincronismo se hace uso del *PWM* (*Pulse Width Modulation*), usado por los *timers* del microcontrolador. Este tipo de modulación se utiliza para ajustar el ciclo de trabajo de una señal periódica tanto cuadrada como sinusoidal.

El modo de trabajo de los *timers* estará configurado como "*Fast PWM*", de modo que se generará una señal *PWM* de esta forma:

1. Un registro hace de contador y cuenta de forma ascendente desde 0 hasta el valor más alto.
2. Otro registro hace de comparador, de modo que cuando el registro contador alcanza el umbral del comparador, este conmuta el pin de salida y queda en nivel alto.
3. Cuanto el contador alcanza el nivel más alto, vuelve a 0 y el pin de salida se conmuta de nuevo
4. El periodo de la señal viene definido en función del tiempo que tarde el registro del contador en ascender de 0 al valor más alto.

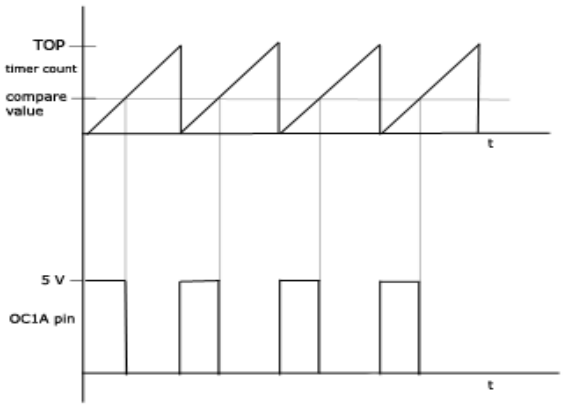


Ilustración 32: Funcionamiento PWM[23]

El fragmento de código para generar la señal de sincronismo de 40kHz es el siguiente:

```
noInterrupts();           // disable all interrupts
TCCR1A = bit (WGM10) | bit (WGM11) | bit (COM1B1);
TCCR1B = bit (WGM12) | bit (WGM13) | bit (CS10);
OCR1A =  (F_CPU / 40000L) - 1;
OCR1B = (F_CPU / 40000L) / 2;
interrupts();
```

Se hace uso de los 3 contadores que dispone el microprocesador para llevar a cabo la *PWM* (*timer0*, *timer1* y *timer2*). Para controlarlos, se utilizan los registros de control *TCCR1A* y *TCCR1B*, que disponen de varios grupos de bits utilizados para la configuración de estos registros.

- *WGM*: Estos grupos de bits son los encargados de generar la forma de onda controlando el modo general del contador.
- *COMxA* y *COMxB*: son los grupos de bits de comparación de coincidencia de las salidas A y B respectivamente, aunque en este proyecto solo se ha utilizado *COMxB*. Estos grupos de bits pueden activar, desactivar e invertir la salida en A o B.
- CS: Este grupo de bits permite seleccionar el tipo de *prescaler*.

Para establecer los umbrales de comparación, se hace uso de los registros *OCR1A* y *OCR1B*. Cuando el nivel de los contadores alcance el valor umbral de los registros *OCR*, cambiarán las salidas correspondientes.

Como se puede observar en el código, se está trabajando con el *timer1*. Esto es debido a que es el único de los 3 *timers* de 16bits de memoria, es decir, puede contar de 0 hasta un valor de 65535. Las tablas que determinan las funciones de los registros *TCCR1A* y *TCCR1B* son las siguientes:

Name: TCCR1A
Offset: 0x80
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM1	COM1	COM1	COM1			WGM11	WGM10
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Name: TCCR1B
Offset: 0x81
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Tabla 3: Registros TCCR1B y TCCR1A[24]

El timer1 da la opción de trabajar con los pines *PWM9* y *PWM10*. Para este caso se ha elegido trabajar con el pin *PWM10*. A continuación, se debe especificar el modo de trabajo mediante los bits *WGM10* y 11 en el registro A y *WGM12* y 13 en el registro B. Como se han activado los cuatro, ha sido seleccionado el modo de trabajo número 15, es decir, *Fast PWM*. Los tipos de configuración se muestran en la siguiente tabla:

Mode	WGM13	WGM12 (CTC1)[1]	WGM11 (PWM11)[1]	WGM10 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Tabla 4: modos de trabajo de PWM[24]

A continuación, se configura el bit *COM1B* para efectuar la comparación en el registro *TCCR1A* y finalmente, en el registro *TCCR1B*, se activa el bit *CS10* para configurar el contador sin *prescaler* como indica la tabla siguiente:

Table 20-7. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

Tabla 5: Descripción de bit seleccionador de reloj[24]

Después de configurar los registros de control de los *timer* se debe configurar el valor de desbordamiento, que simbolizará el valor máximo antes de que se reinicie la cuenta, y el valor de comparación que servirá de umbral para generar la señal a partir de *PWM*. Esto se calcula mediante las dos fórmulas siguientes respectivamente:

$$Reg.Comp = \frac{Frec.\ CPU}{Frec.Señal\ \cdot\ Prescaler} - 1$$

$$Reg.Comp = \frac{\frac{Frec.CPU}{Frec.Señal\ \cdot\ Prescaler}}{2}$$

Finalmente, en este fragmento de código aparecen dos instrucciones de interrupciones internas: *noInterrupts()* para deshabilitar las interrupciones e *interrupts()* para volver a habilitarlas. Esto es necesario ya que este fragmento de código debe realizarse sin ningún tipo de interrupción para evitar errores en el funcionamiento, de modo que se deshabilitan las interrupciones y una vez realizadas las instrucciones, se vuelven a activar.

Para ahorrar energía y mejorar la eficiencia del programa, se desactivan aquellos parámetros que no se vayan a utilizar. En este punto se utilizan los macros de la librería a*vr/power.h*. Estos parámetros son los siguientes:

```
ADCSRA = 0;  // ADC
power_adc_disable ();
power_spi_disable();
power_twi_disable();
power_timer0_disable();
//power_usart0_disable();
Serial.begin(115200);
```

- **ADCSRA=0**:Se deshabilita el conversor analógico/digital.
- **power_adc_disable**: Se desactiva el módulo del conversor analógico/digital.
- **power_spi_disable**: Se desactiva la interfaz periférica en serie.
- **power_twi_disable**: Se desactiva la interfaz de dos cables (*Two Wire Interface*)
- **power_timer_disable**: Se desactiva el módulo del timer0.

Seguidamente, se inicia la comunicación con el puerto serie mediante la instrucción *Serial.begin()* a la mayor velocidad que permite Arduino, es decir, 115200 baudios.

En la siguiente línea de código tenemos una variable de tipo puntero. Este tipo de variable almacena la dirección de memoria de las variables que se le ha asignado. Para ello, se declara el puntero con el tipo de variable cuya dirección se desea asignar y se le nombra añadiendo un asterisco, tal y como se muestra en el código. Seguidamente, se obtiene el número de dirección de memoria añadiendo el símbolo & delante del nombre de la variable.

```
byte* emittingPointer = &animation[frame][0];
```

En esta línea se está obteniendo la dirección de memoria del primer valor almacenado en el primer array de la matriz "*animation*" creada previamente.

A continuación, se declara una variable de tipo byte llamada "*buttonsPort*" y se le asignara el valor 0. También se declaran dos variables de tipo booleano (1 byte cada una) y una de tipo short (2 bytes). La variable *buttonPressed[N_BUTTONS]* consiste en un array booleano que contiene 6 casillas, ya que es el valor asignado previamente a N_BUTTONS. Todas estas variables serán utilizadas más adelante en el código.

```
byte buttonsPort = 0;
```

A pesar de seguir en la sección *void setup()*, se realizará un tramo de código que se repetirá en bucle mediante la sentencia *goto*. Esta sentencia transfiere el flujo del programa a un punto etiquetado en el programa[25]. Para crear este bucle, se debe redactar el código entre dos sentencias: la etiqueta y el *goto* con el que se hace referencia a la etiqueta. Un ejemplo de uso es el siguiente:

```
LOOP:                  //etiqueta
//código en la etiqueta
goto LOOP;             //referencia a la etiqueta
```

Dentro de la etiqueta *LOOP* se encuentran varias líneas de código que se repetirán de forma indefinida. Primero, se utiliza un bucle *while* que terminará cuando deje de cumplirse la condición que se ha determinado en los paréntesis. Se observa que en el bucle *while* no se ha indicado ninguna instrucción que deba cumplir mientras su condición sea verdadera, por tanto el flujo del programa quedará en "espera" en esa línea hasta que deje de cumplirse la condición.

```
while(PINB & 0b00001000); //wait for pin 11 (B3) to go low
```

La condición establece que el bucle se mantendrá siempre y cuando *PINB* sea igual que 0b00001000. Esta condición viene dada por el símbolo *&*, que representa una función lógica de tipo *AND*.

PIN es un registro que permite leer el estado de los pines. Si se declara la instrucción *PINB*, se están leyendo todos los pines del puerto B, es decir, del pin 8 al pin 13, como se puede

comprobar en el mapa de puertos mostrado anteriormente. Con todo esto, se entiende que mientras el pin número 11 del puerto B esté a nivel alto, el bucle *while* seguirá activo.

Después del bucle *while* aparece un tramo de código compuesto por varias funciones macro *OUTPUT_WAVE()*, que tomarán contexto a continuación.

```
OUTPUT_WAVE(emittingPointer, 0); buttonsPort = PIND; WAIT_LIT();
OUTPUT_WAVE(emittingPointer, 1); anyButtonPressed = (buttonsPort & 0b11111100) != 0b11111100; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 2); buttonPressed[0] = buttonsPort & 0b00000100; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 3); buttonPressed[1] = buttonsPort & 0b00001000; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 4); buttonPressed[2] = buttonsPort & 0b00010000; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 5); buttonPressed[3] = buttonsPort & 0b00100000; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 6); buttonPressed[4] = buttonsPort & 0b01000000; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 7); buttonPressed[5] = buttonsPort & 0b10000000; WAIT_MID();
OUTPUT_WAVE(emittingPointer, 8); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 9); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 10); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 11); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 12); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 13); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 14); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 15); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 16); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 17); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 18); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 19); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 20); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 21); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 22); WAIT_LOT();
OUTPUT_WAVE(emittingPointer, 23);
```

Se tiene una cantidad de 24 funciones *OUTPUT_WAVE()* que recorren todos los *subarrays* de la matriz *animation*, barriendo por completo los valores de la matriz. En cada una de las funciones se introduce la variable *emmitingPointer*, que apunta a la dirección de memoria de la matriz *animation* que se determine, y un número, que indica el *subarray* que se está recorriendo en la matriz (como son 24 *subarrays*, ese parámetro irá de 0 a 23).

En la primera línea de código, al lado de del primer *OUTPUT_WAVE()*, aparece una instrucción que almacena la información de los pines del puerto D en la variable *buttonsPort*, creada anteriormente, mediante el registro *PIND*. Los pines de D2 a D7 se habían configurado como entradas para este fin. Finalmente, en la propia línea se ha colocado una función de espera *WAIT_LIT* de corta duración.

A continuación, tras el siguiente *OUTPUT_WAVE()*, aparece la variable *anyButtonPressed* , de tipo booleana. Esta variable servirá para determinar si existe algún botón pulsado. Para ello se realizan dos operaciones:

1. se compara la variable *buttonsPort* y el valor binario 0b11111100 bit a bit mediante una función lógica *AND* (&).
2. El resultado de la operación anterior se compara de nuevo con el valor binario 0b11111100, pero esta vez con la función "!=", que devuelve *TRUE* cuando la comparación no se cumpla.

La finalidad de esta función es obligar al flujo del programa a entrar en la sección de sentencias condicionales que se verá más adelante. Como se puede observar, los dos bits menos significativos están a 0. Como estos bits corresponden a los pines encargados de la transmisión y recepción de datos de Arduino (*D1→TX*, *D0→RX*), no serán utilizados como entradas de botones ya que pueden originarse errores en el código.

Tras esta línea, se tienen 6 líneas más que cumplen la función de determinar qué botón se está pulsando. Como se tiene la posibilidad de usar hasta 6 botones, se redacta la misma función 6 veces con distintas condiciones. Esta función lógica compara, mediante una función *AND*, los pines activos del puerto D con el valor binario que represente aquel botón que esté activado. Es decir, si se pulsa el botón 0, se activará el pin 0b00000100 del puerto D, por tanto, la función será verdadera en aquella línea donde se esté comparando este valor de *buttonsPort* con el valor binario 0b00000100.

A partir de aquí comienza la sección de sentencias condicionales mencionada anteriormente.

```
if( anyButtonPressed ){
   ++buttonCounter;
   if (buttonCounter > BUTTON_SENS){
    buttonCounter = 0;

    if (! buttonPressed[0] ) {
      if( frame < STEP_SIZE ) {
        frame = N_FRAMES-1;
     }else{
        frame-=STEP_SIZE;
     }
    }
    else if (! buttonPressed[1] ) {
      if ( frame >= N_FRAMES-STEP_SIZE ) {
        frame = 0;
      }else {
        frame+=STEP_SIZE;
      }
   }else if (! buttonPressed[2] ) {
      frame = 0;
   }
    emittingPointer = & animation[frame][0];
   }
 }else {
   buttonCounter = 0;
 }

goto LOOP;
```

Para empezar, se tiene un *if* condicionado por la variable *anyButtonsPressed*. Mientras sea verdadera, se ejecutarán las líneas del interior del *if*, en caso contrario, se ejecutará el bloque *else*. Este condicional se ha creado con el fin de ejecutar ciertas funciones cuando se detecte algún botón pulsado, pero de la forma en la que ha sido programado anteriormente, esta condición siempre se cumplirá. La intención detrás de esto es evitar errores producidos al pulsar todos los botones a la vez.

```
anyButtonPressed = (buttonsPort & 0bllllllll00) != 0bllllllll00; WAIT_MID();
```

Dentro de la condición del *if*, se encuentra una instrucción que suma el valor de 1 a la variable *buttonCounter* mediante el operador "++" (*buttonCounter=buttonCounter + 1*). Además, aparece otro *if* cuya condición es *buttonCounter > BUTTON_SENS*. Teniendo en cuenta que el valor que se le asignó a BUTTON_SENS es 2500, la condición no se cumplirá hasta que *buttonCounter* no alcance ese valor. Una vez lo alcance, se reseteará el valor de *buttonCounter* al del interior del último *if*.

```
++buttonCounter;
if (buttonCounter > BUTTON_SENS){
 buttonCounter = 0;
```

A lo largo del programa, se han configurado 6 pines como entrada para los botones, pero para este proyecto se utilizan solo 3. Estos botones se han configurado para ejecutar funciones capaces de mover hacia arriba, hacia abajo o resetear la posición de la partícula a levitar. El resto de las entradas no tendrán ninguna configuración aparente, pero pueden ser utilizadas en el futuro para añadir alguna función adicional al programa. A continuación, se tiene 3 sentencias condicionales (1 *if* y 2 *elseif*) correspondientes a los botones mencionados.

El primer *if* corresponde al botón 0, por tanto, la condición dicta que cuando la variable booleana *buttonPressed[0]* sea falsa, se ejecutarán las funciones de su interior. Dentro de esta condición se tiene otro if que se cumplirá cuando *frame<STEP_SIZE*. Como a la variable *STEP_SIZE* se le ha asignado un valor de 1, esta condición se cumplirá cuando el valor de *frame* sea menor. En el interior de este *if*, se tiene la función *frame=N_FRAMES-1*, por lo tanto, cuando el valor de *frame* alcance 0, su valor será igualado al valor de *N_FRAMES(24) - 1*, es decir, tomará un valor de 23. En el caso de que la variable *frame* no sea menor que 1, se ejecutará la función

else, en la que se restará el valor de *STEP_SIZE* a *frame* mediante el operador "-=" (*frame = frame – STEP_SIZE*)

```
if (! buttonPressed[0] ) {
  if( frame < STEP_SIZE ) {
    frame = N_FRAMES-1;
 }else{
    frame-=STEP_SIZE;
 }
}
```

Lo que se consigue con esto es que la variable *frame* realice un descuento desde 23 a 0 y se reinicie a 23. Este valor será asignado al puntero de la matriz, que hará que cambie su correspondiente array. De este modo la onda cambiará de fase y la partícula se desplazará hacia abajo.

El siguiente *elseif* corresponde al botón 1 y se utiliza para desplazar la partícula hacia arriba. Para ello se hace uso de la variable *buttonPressed[1] y* se sigue la misma lógica que en el *if* anterior pero de forma invertida. Por tanto, la variable *frame* se reseteará a 0 cuando su valor alcance *N_FRAMES – STEP_SIZE (24-1)*, de modo que se crea una cuenta ascendente. En la función *else* se utiliza el operador "+=" (*frame = frame + STEP_SIZE*) para efectuar dicha cuenta ascendente.

```
else if (! buttonPressed[1] ) {
  if ( frame >= N_FRAMES-STEP_SIZE ) {
    frame = 0;
  }else {
    frame+=STEP_SIZE;
  }
```

Por último, se tiene el *elseif* correspondiente al botón 2, que se utilizará para resetear la configuración inicial de fase de la onda emitida. La condición a cumplir es la misma que en los botones anteriores pero para el *buttonPressed[2]*. Dentro de esta condición se encuentra una sola función que asigna el valor 0 a la variable *frame*.

```
else if (! buttonPressed[2] ) {
    frame = 0;
}
```

Mediante esta función, se consigue llevar la onda a un estado inicial sin la necesidad de reiniciar el dispositivo entero o ajustar la fase de la onda manualmente.

Tras haber desarrollado el funcionamiento de las sentencias condicionales de los 3 botones, se pasa a analizar la última instrucción contenida en el *if* de condición *buttonCounter>BUTTON_SENS*.

```
emittingPointer = & animation[frame][0];
```

El valor que tome la matriz *animation* será determinado por la variable *frame* obtenida de las sentencias condicionales anteriores, por tanto, mediante esta instrucción se almacenará la dirección de memoria de este valor en el puntero *emittingPoint*. Este valor será modificado en función del botón que esté siendo pulsado.

Para finalizar, se tiene el *else* correspondiente al *if* principal, el cual será ejecutado cuando no se cumpla dicho *if*. Básicamente, la instrucción que contiene establece la variable *buttonCounter* en su valor inicial (0).

```
else {
  buttonCounter = 0;
}
```

A continuación se encuentra la sentencia de *goto LOOP;* que cierra el bucle:

```
goto LOOP;
```

5.3. void loop()

Como se ha utilizado la sentencia *goto*, no es necesario hacer uso de esta sección.

6. Caracterización del levitador

6.1. Modelo teórico

Para poder levitar un objeto, se debe generar un campo acústico que ejerza una fuerza sobre él y que además quede atrapado en un punto en concreto. Las fuerzas producidas por la acción de ondas sonoras suelen ser de intensidad débil, por lo que es necesario crear un campo acústico de alta intensidad que provoque efectos no lineales. Para este fin es necesario crear una onda estacionaria. Una onda estacionaria se forma debido a la acción de dos ondas de igual amplitud, frecuencia y fase que interfieren entre sí cuando viajan en sentido opuesto.

Cuando el sonido se propaga a través del aire, se crea un desplazamiento de partículas por la acción de la onda sonora. Se forman puntos de máximos y mínimos de presión acústica (antinodos) y puntos de presión nula (nodos).

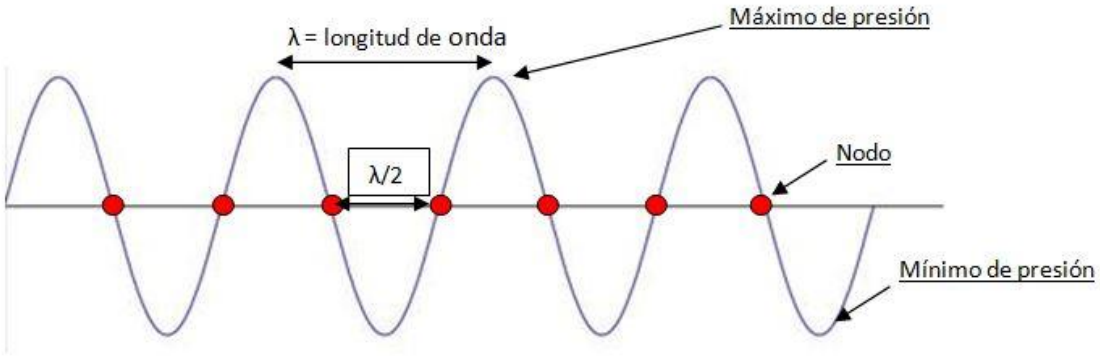


Ilustración 33: Descripción de onda[12]

Aplicando este fundamento al diseño clásico de un levitador acústico, se obtiene una configuración compuesta por un emisor que genera la onda y un reflector cuya distancia de separación corresponda a un múltiplo de media longitud de la onda emitida, de modo que se crea una onda estacionaria debido a las reflexiones que se superponen sobre la onda emitida.

Para este fin, se suelen utilizar ultrasonidos a frecuencias relativamente altas y niveles de presión sonora elevados (alrededor de 150dB SPL), de modo que los niveles de presión sonora de los antinodos son suficientemente altos como para mantener una partícula atrapada en una posición concreta. Esta posición que albergará la partícula será el nodo estable más próximo a ella cuando se encuentre en el interior de un campo acústico.

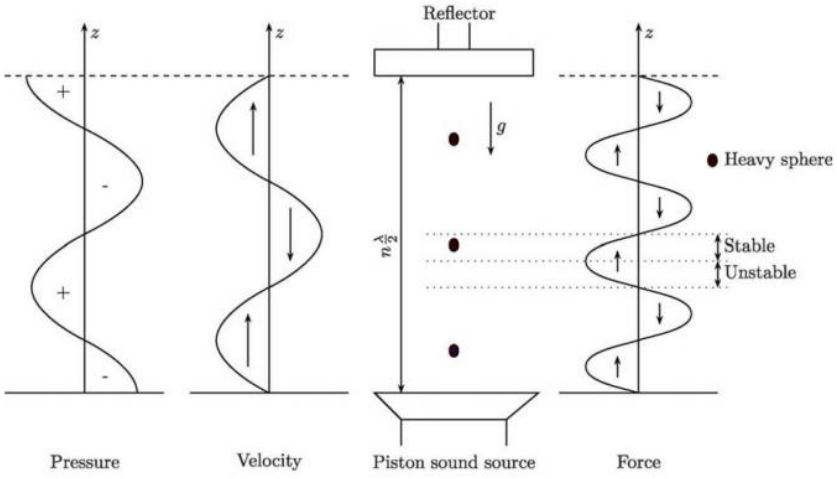


Ilustración 34: Zona de estabilidad de la onda estacionaria en el levitador acústico[26]

Para el modelo clásico de levitador acústico, es necesario que el diámetro de la partícula sea $d \leq \frac{\lambda}{2}$. En caso de ser superior, el objeto levitado supera el rango estable del interior del nodo,

pierde estabilidad debido a la presión de los antinodos y cae. Teniendo esto en cuenta, aumentar la frecuencia provoca que los objetos a levitar deban ser más pequeños y viceversa.

Es fácil pensar que reducir la frecuencia del campo acústico permite elevar objetos de mayor diámetro, pero tiene sus riesgos, ya que la directividad también disminuye, de modo que la presión acústica es más dispersa. Además, hay que tener en cuenta la frecuencia de resonancia del emisor para poder aprovechar toda la energía. Si la frecuencia emitida no está centrada con la frecuencia de resonancia, no se estará alcanzando la presión máxima emisible, por lo que se obtiene un sistema ineficiente.[12]

La fuerza de radiación acústica es un término de segundo orden, por tanto no plantea el uso del análisis de la teoría acústica linealizada. Muchas de las investigaciones sobre la levitación acústica se apoyan en simulaciones numéricas. El software más común utilizado para este fin es *Comsol Multiphysics*. [13]

El estudio analítico de la levitación acústica radica en el potencial de Gor'kov. En 1961, Gor'kov derivó la fuerza de más bajo orden sobre una partícula esférica en un campo acústico. Este potencial queda definido por la siguiente ecuación:

$$\widetilde{U}_a = 2\pi R_s^3 \left(\frac{p_{rms}^2}{3\rho_0 c^2} - \frac{\rho_0 v_{rms}^2}{2} \right)$$

Ecuación 1

donde p_{rms} y v_{rms} son los valores obtenidos de la media cuadrática de la presión acústica y la velocidad de partícula, ρ_0 y c es la densidad y velocidad del sonido del fluido respectivamente y R_s es el radio de la muestra a levitar. Esta ecuación es válida para partículas cuyo radio es pequeño en comparación con la longitud de onda del campo acústico y cuya densidad sea mucho mayor que la densidad del medio que la rodea.

Para la obtención de la fuerza total que actúa sobre la partícula atrapada en el campo acústico, se requiere además la acción del potencial gravitatorio, que viene dado por:

$$\widetilde{U}_g = \frac{4}{3}\pi R_s^3 \rho_s g z$$

Ecuación 2

donde ρ_s es la densidad de la muestra, $g = 9.81 m/s^2$ es la aceleración gravitacional y z la coordenada vertical del centro de masa. El potencial de radiación acústica total viene dado por la suma de estos dos componentes[27]:

$$\widetilde{U}_{tot} = \widetilde{U}_a + \widetilde{U}_g$$

Ecuación 3

Esta igualdad se cumple, ya que se ha realizado un análisis dimensional para comprobar que es correcta y entender mejor este fenómeno físico:

- $\boldsymbol{\widetilde{U}_a = 2\pi R_s^3 \left(\frac{p_{rms}^2}{3\rho_0 c^2} - \frac{\rho_0 v_{rms}^2}{2} \right)}$
  - $R_s^3 \equiv m^3 \equiv L^3$
  - $p_{rms}^2 \equiv \left(\frac{N}{m^2}\right)^2 = \left(\frac{kg * \frac{m}{s^2}}{m^2}\right) \equiv M^2 \cdot L^{-2} \cdot T^{-4}$
  - $\rho_0 \equiv \frac{kg}{m^3} \equiv M \cdot L^{-3}$

$$\widetilde{U}_a \equiv L^3 \cdot \left(\frac{M^2 \cdot L^{-2} \cdot T^{-4}}{M \cdot L^{-3} \cdot L^2 \cdot T^{-2}} - M \cdot L^{-3} \cdot L^{-2} \cdot T^{-2} \right)$$
$$= L^3 \cdot (M \cdot L^{-1} \cdot T^{-2} - M \cdot L^{-1} \cdot T^{-2}) = M \cdot L^2 \cdot T^{-2}$$

- $\boldsymbol{\widetilde{U}_g = \frac{4}{3}\pi R_s^3 \rho_s g z}$

$$\widetilde{U}_g \equiv L^3 \cdot M \cdot L^{-3} \cdot L \cdot T^{-2} = M \cdot L^2 \cdot T^{-2}$$

- $\widetilde{\boldsymbol{U}}_{\boldsymbol{tot}} = \widetilde{\boldsymbol{U}}_{\boldsymbol{a}} + \widetilde{\boldsymbol{U}}_{\boldsymbol{g}}$

$$\widetilde{U}_{tot} \equiv M \cdot L^2 \cdot T^{-2} + M \cdot L^2 \cdot T^{-2} \equiv kg\frac{m^2}{s^2}$$

El campo acústico total es proporcional al volumen de la muestra y es conveniente normalizarlo por un factor similar, dando como resultado unidades de presión

$$U_{tot} = \frac{\widetilde{U}_{tot}}{2\pi R_s^3}$$

Ecuación 4

La fuerza por unidad de volumen que actúa sobre la esfera se calcula mediante el gradiente del potencial total:

$$\vec{F} = -\nabla U_{tot}$$

Ecuación 5

6.2. Toma de medidas

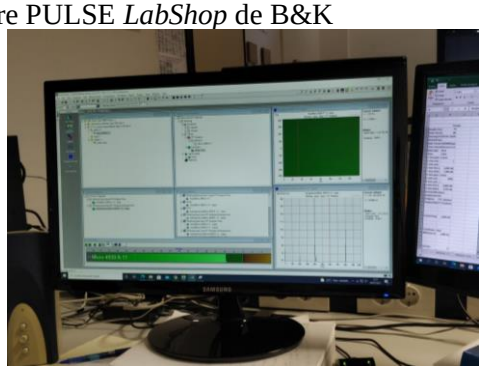
Una vez explicados los fenómenos físicos involucrados en la levitación de los objetos mediante ultrasonidos, se realizan una serie de medidas para comprobar cómo se desarrollan en un medio no ideal.

Se ha realizado un barrido acústico desde la parte más baja del levitador hasta unos milímetros más por encima de la mitad y varias medidas de presión sonora equivalente a distintas tensiones en el centro del levitador, con una tensión de entrada de 5 a 12V en el módulo del circuito. Para la toma de estas medidas se han utilizado los siguientes materiales:

- Micrófono de ultrasonidos B&K 4939 A11

- Módulo B&K 3110

- Software PULSE *LabShop* de B&K



- Trípode

- Papel milimetrado
- Fuente de Alimentación
- Multímetro

Barrido acústico

Para la toma de esta medida se ha colocado el micrófono sobre el trípode con una pinza y gomaespuma para mantenerlo sujeto. En el trípode se ha pegado una hoja de papel milimetrado para ajustar con exactitud la elevación.

Ilustración 35: Ajuste de medición en el barrido acústico

Se han realizado un total de 22 medidas cada 2 milímetros, desde la parte más baja teniendo en cuenta el grosor del micrófono, hasta pocos milímetros más arriba del centro, como se ha explicado anteriormente. Debido a las reflexiones del sonido junto a la emisión simultánea de ultrasonidos en la base superior, se considera que los nodos que aparecen en la mitad superior del levitador son equivalentes a los de la mitad inferior.

Para esta medida se ha aplicado un voltaje de 10.5V, ya que es un valor de tensión medio y suficiente para obtener un mapeo correcto de los nodos, sin riesgos de sobrecalentamiento del Arduino. La gráfica obtenida es la siguiente:

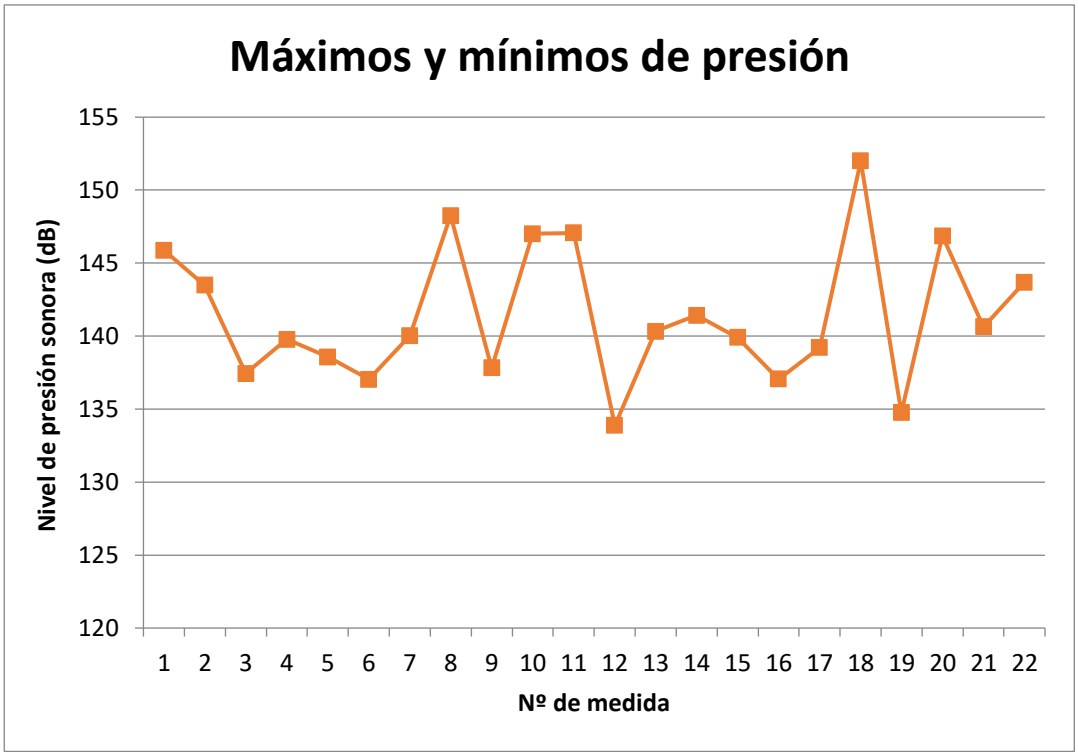


Ilustración 36: Máximos y mínimos de presión

Se puede observar que existe una variación de presión en cada medida realizada. Esto demuestra que en el interior del levitador existe un campo acústico compuesto por puntos de mayor y menor presión.

Los picos de la gráfica representan el módulo de medidas de presiones tanto máximas como mínimas, ya que la presión medida en el micrófono será siempre positiva. Los valles representan "nulos" de presión, que son los puntos de la onda estacionaria donde las partículas quedan atrapadas de forma estable.

Se observa que los niveles de presión en las primeras medidas toman valores irregulares debido a diferentes factores. Por un lado, al tomar medidas demasiado próximas al emisor, se han captado cancelaciones debido a las reflexiones de los altavoces. Por otro lado, el hecho de aproximar tanto el micrófono al emisor provoca que no se capte todo el conjunto de presión acústica que emiten los transductores debido a su directividad.

A medida que el micrófono se posiciona en el centro, se definen mejor estos picos y valles de presión. A partir de la medida número 7 se observan variaciones de presión acústica más claras. La medida con mayor nivel de presión acústica es la número 18, debido a que se ha tomado con el micrófono a la altura central del levitador.

Si se tiene en cuenta que la frecuencia a la que emiten los transductores es de 40kHz, se obtiene que la longitud de onda es de 0.00865m (8.65mm), considerando la velocidad del sonido como c=346m/s a una temperatura de 25ºC. De modo que la longitud máxima entre las medidas correspondería a 1/4 de la longitud de onda para obtener una resolución aceptable, es decir, menos de 2.16mm.

Para obtener mejores resultados en la toma de medidas, sería recomendable reducir la distancia entre ellas, de modo que aumentaría la resolución del gráfico y se observarían mejor los máximos y mínimos, pero se ha decidido tomar medidas de 2mm de distancia entre sí para reducir considerablemente el número de tomas, además que los trípodes disponibles en el laboratorio no están previstos para realizar desplazamientos de precisión.

Nivel de presión acústica en el centro del levitador a distintos valores de tensión

En esta toma de medidas se ha situado el micrófono en el centro del levitador acústico, donde se halla el punto de mayor presión acústica. Mediante la fuente de tensión regulable, se ha alimentado el circuito del levitador de 12 a 5V en periodos de 0.5V entre medidas. Para cada valor de tensión, se han tomado 3 medidas de presión acústica para obtener un valor medio, ya que el nivel de presión acústica radiada no es constante. Tras realizar las medidas, se ha vuelto a medir la presión en 12V para demostrar que la temperatura de los transductores afecta a la presión acústica emitida.

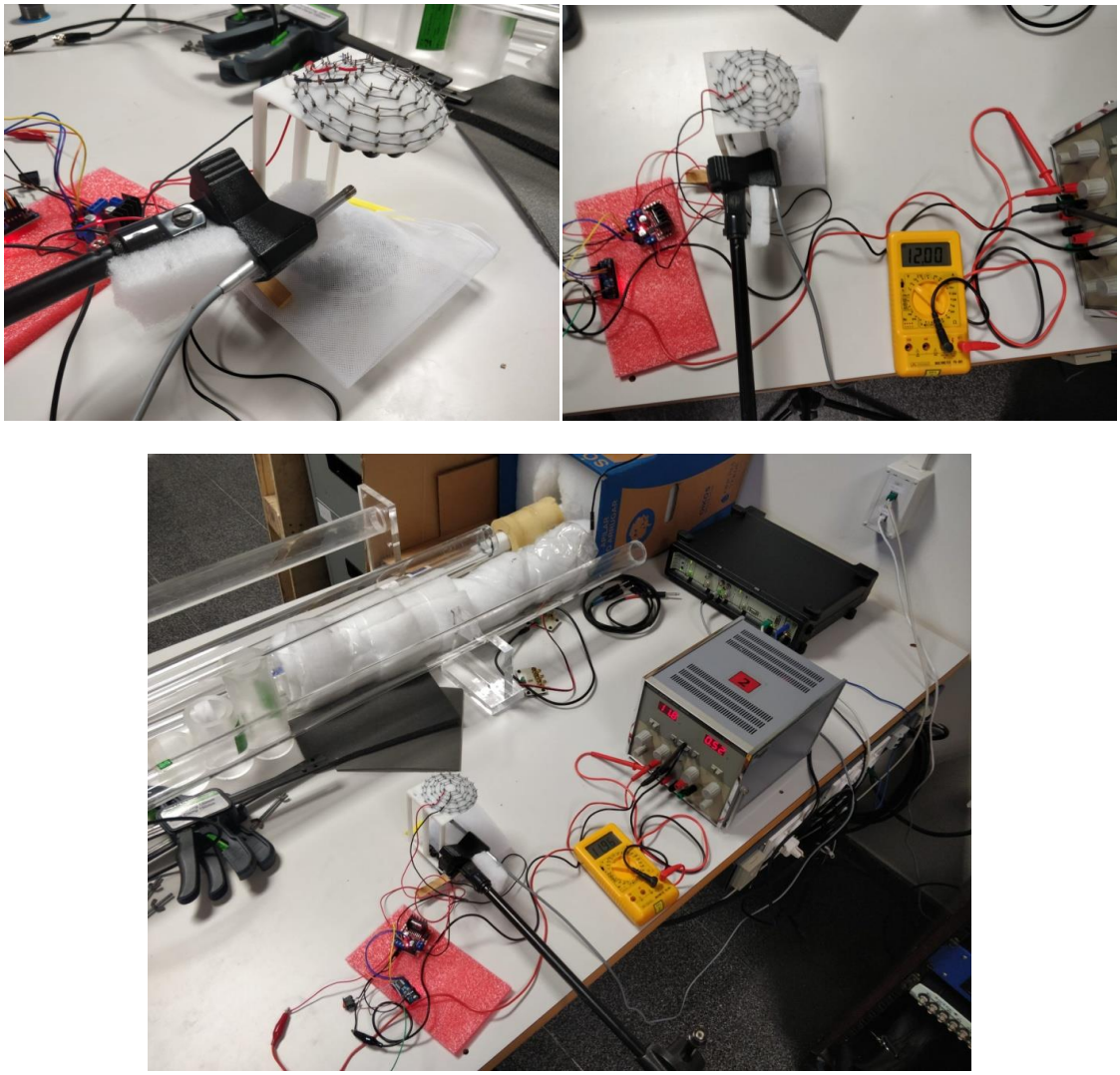


Ilustración 37: Toma de medidas del nivel de presión acústica

A continuación, se expresa en una gráfica la variación del nivel de presión acústica en el centro del levitador en función del voltaje de entrada.

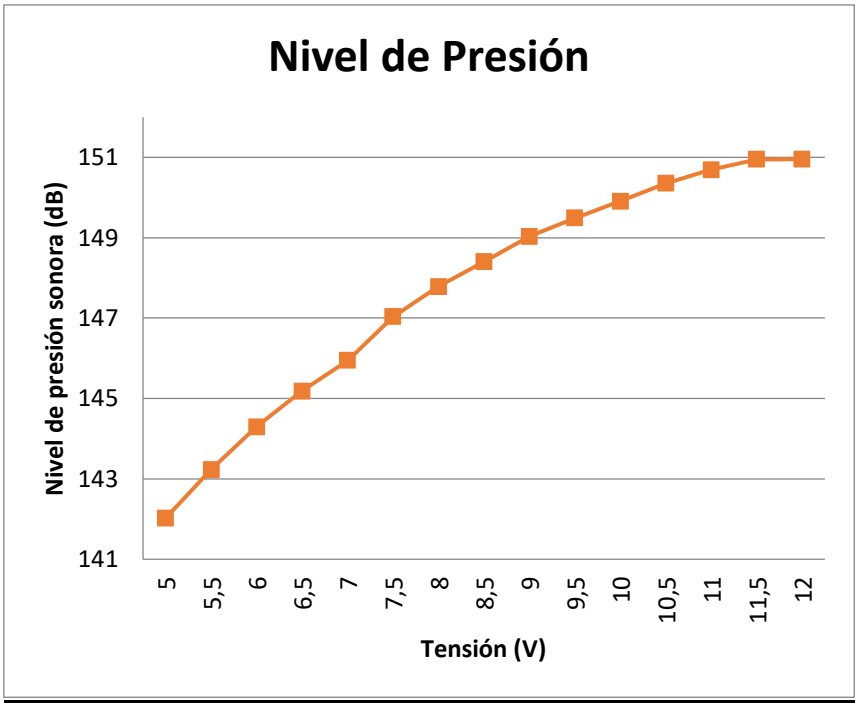


Ilustración 38: Nivel de presión en función de la tensión

Tensión (V)	Presión (dB)
5	142,0
5,5	143,2
6	144,3
6,5	145,2
7	145,9
7,5	147,0
8	147,8
8,5	148,4
9	149,0
9,5	149,5
10	149,9
10,5	150,4
11	150,7
11,5	151,0
12	151,0

Tabla 6: Presión en el centro del levitador en función de la tensión de entrada.

Los valores de presión acústica producen una curva logarítmica en la gráfica debido a que las medidas están expresadas en decibelios. El valor máximo alcanzado por el levitador acústico en su centro es de 150.953dB (≈151dB), que equivale a un valor de presión de 705.81Pa. Los valores de presión entre 11.5V y 12V son similares, ya que el circuito no está capacitado para admitir más de 12V, por lo que la potencia se limita al acercarse a ese umbral.

Las medidas se han comenzado aplicando 12V en frio y descendiendo hasta los 5V. Tras tomar las medidas, se ha regulado la fuente de tensión a 12Vde nuevo para comprobar cómo el calentamiento de los transductores y del sistema en conjunto afecta a la radiación de sonido. En la siguiente tabla aparece una comparativa de los valores en frio y en caliente.

Presión a 12V en frio	151,0 dB
Presión a 12V en caliente	150,6 dB

Tabla 7: Nivel de presión en función de la temperatura.

Efectivamente, los niveles de presión sonora han disminuido debido al uso continuo del dispositivo, por lo que queda demostrado que la temperatura afecta a los niveles de presión.

6.3. Limitaciones del levitador

El modelo de levitador acústico de este proyecto es capaz de levitar objetos pequeños que cumplan ciertas características y poder desplazarlos en el eje OZ mediante cambios de fase. Principalmente, este modelo está capacitado para suspender elementos cuyo tamaño sea menor que la distancia de media longitud de onda emitida, como se ha explicado anteriormente. Actualmente existen versiones de levitador acústico que se han desarrollado específicamente para levitar objetos cuyo tamaño sea $d > \frac{\lambda}{2}$, pero su construcción es más compleja y están limitados a levitar objetos concretos[26], [28].

Para este modelo se han podido levitar distintos objetos que cumplen con la condición de tamaño anterior y con diversas formas. Aquellos objetos esféricos o planos presentaban mayor estabilidad, en cambio, algunos objetos de forma irregular comenzaban a rotar debido al torque producido por la presión acústica.

Dependiendo el voltaje aplicado en el dispositivo, es posible levitar objetos de mayor peso, respetando siempre el tamaño y voltaje límite.

Además, está capacitado para efectuar un movimiento vertical de las partículas levitadas efectuando cambios progresivos en la fase de la onda estacionaria. Existen tres funciones programadas para efectuar este movimiento: desplazamiento ascendente, desplazamiento descendente y reseteo de la onda en fase 0.

Este levitador está capacitado para suspender algunos tipos de líquidos como alcohol. El comportamiento de los líquidos en el levitador es diferente al de objetos sólidos, ya que un sólido puede exponerse a altos niveles de presión acústica sin que se vea afectada su estabilidad. Para levitar líquidos, el voltaje aplicado en el dispositivo debe estar comprendido entre 8 y 10,5 V aproximadamente, ya que con voltajes inferiores a este rango, el líquido es incapaz de mantenerse suspendido y cae, en cambio si el valor de tensión supera los 10,5V, el líquido no puede soportar los niveles de presión acústica y estalla. Para prevenir daños en los transductores por la acción de líquidos, es necesario cubrirlos con algún material que evite su contacto, pero que no absorba o difracte el sonido. En este proyecto se ha utilizado un fragmento de tela de tutú plegada (ver *Ilustración 40*).

Teniendo esto en cuenta, es posible saber cuál es la densidad máxima levitable en el dispositivo si se introduce un objeto cuyo tamaño sea el máximo permitido por el levitador, es decir, $\lambda/2$.

Para la obtención de este valor es necesario medir el nivel de presión acústica en el centro del dispositivo a la máxima tensión permitida, es decir, 12V. Este valor es de:

$$LP = 151{,}0dB$$

obtenido previamente en la toma de medidas.

El potencial de Gro'kov, como se ha explicado anteriormente, describe la radiación de presión en una esfera de radio "R" y densidad "p_s" en un campo acústico.

El campo acústico queda descrito por los parámetros "p" (presión) y "u" (velocidad de partícula). La desviación cuadrática media de estos términos es independiente del tiempo en un campo armónico y las dos son funciones de posición.

La velocidad de partícula máxima se calcula mediante la siguiente expresión:

$$v_0 = \sqrt{\frac{2P_{eff}}{A\ c\ \rho_0}}$$

Ecuación 6[13]

donde "A" es el área de la superficie del transductor, "*Peff*" es la potencia efectiva, "c" es la velocidad de propagación del sonido en el medio y "ρ_0" es la densidad del medio en el que se levita. Los nodos donde la partícula levita de forma estable están dispuestos a lo largo de $r = 0$, y a lo largo de este eje, las desviaciones cuadráticas medias son:

$$\langle u^2 \rangle = \frac{P_{eff}}{A\ c\ \rho_0} sin^2\left(\frac{\pi\ f\ z}{c}\right)$$

Ecuación 7[13]

$$\langle p^2 \rangle = \frac{4\ c\ \rho_0\ P_{eff}}{A} cos^2\left(\frac{\pi\ f\ z}{c}\right)$$

Ecuación 8[13]

Donde "f" es la frecuencia de la onda emitida y "z" la distancia vertical de los transductores al objeto levitado. Como la fuerza está relacionada con el gradiente del potencial $F = -\nabla U$, el componente "Z" de la fuerza por volumen a lo largo del eje $r = 0$ es:

$$F_z = \left[\frac{\pi\, f\, P_{eff}(\rho_0 + 11\rho_s)}{2\, A\, c^2(\rho_0 + 2p_s)}\right] \sin\left(\frac{2\pi\, f\, z}{c}\right)$$

Ecuación 9[13]

El valor máximo de la fuerza es el termino contenido en los corchetes, por tanto, el valor de densidad máximo a levitar (p_s) está implícito en la ecuación:

$$g\, p_s = \frac{\pi\, f\, P_{eff}(\rho_0 + 11\rho_s)}{2\, A\, c^2(\rho_0 + 2p_s)}$$

Ecuación 10[13]

Resolviendo solamente para p_s y asumiendo que *f* es un valor grande, se obtiene un límite superior para el cálculo de la densidad de la partícula a levitar:

$$p_s \leq \frac{11\, \pi\, f\, P_{eff}}{4\, A\, c^2\, g} - \frac{9\rho_0}{22}$$

Ecuación 11[13]

Además, se puede obtener la potencia acústica necesaria para levitar un objeto de una densidad determinada:

$$P_{eff} \geq \frac{2\, A\, c^2 g(9\rho_0 + 22p_s)}{121\, \pi\, f}$$

Ecuación 12[13]

Con todo esto, se pasa a calcular la densidad máxima levitable. Será necesario obtener un valor de potencia acústica efectiva en lineal, de modo que se calculará a partir de los valores decibélicos de presión acústica y se realizará una conversión decibelio-lineal.

Teniendo en cuenta que el nivel de presión sonora máxima en el centro del levitador es de 151,0dB (709,6Pa), es posible obtener la potencia efectiva necesaria despejando la variable en la *Ecuación 8*[13]:

$$P_{eff} = \frac{p^2 A}{4c\rho_0 cos^2\left(\frac{\pi f z}{c}\right)}$$

Si se considera que de 72 transductores sólo 36 hacen de emisor mientras que la otra mitad actúa como reflector, el área es la de un pistón equivalente al espacio que ocupan los 36 transductores. Si el radio de uno es $r = 0.0099m$, su área será:

$$A_u = r^2\pi = 0.0099^2\pi = 7{,}7 \cdot 10^{-5} m^2$$

Y el área total será:

$$A = 7{,}7 \cdot 10^{-5} \cdot 36 = 0{,}002772 m^2$$

Una vez calculada el área, se aplica la *Ecuación 8*[13] para obtener la potencia efectiva. Para este caso se considera $c = 343\frac{m}{s}$, $\rho_0 = 1{,}22\frac{kg}{m^3}$, $f = 40000Hz$ y $z = 0{,}0495m$:

$$P_{eff} = \frac{709{,}6^2 \cdot 0{,}002772}{4 \cdot 343 \cdot 1{,}22 \cdot cos^2\left(\frac{\pi \cdot 40000 \cdot 0{,}0495}{343}\right)} = 1{,}46W$$

La obtención de la densidad máxima se realiza mediante la *Ecuación 11*[13]. Como todos los parámetros son el máximo valor obtenido de las medidas, la inecuación se puede calcular como una ecuación:

$$p_s = \frac{11\,\pi\,f\,P_{eff}}{4\,A\,c^2\,g} - \frac{9\rho_0}{22} = \frac{11 \cdot \pi \cdot 40000 \cdot 1{,}46}{4 \cdot 0{,}002772 \cdot 343^2 \cdot 9{,}81} - \frac{9 \cdot 1{,}22}{22} = 157{,}21\frac{kg}{m^3}$$

Este valor de densidad máxima obtenido es lógico para la levitación de algunos materiales como papel o poliestireno. En cambio, el elemento más denso levitado en el dispositivo a nivel experimental ha sido el alcohol. La densidad del alcohol es de $789\frac{kg}{m^3}$, por lo que es mucho mayor que el valor calculado.

Ilustración 39: Levitación de papel y poliestireno (objeto de este trabajo)

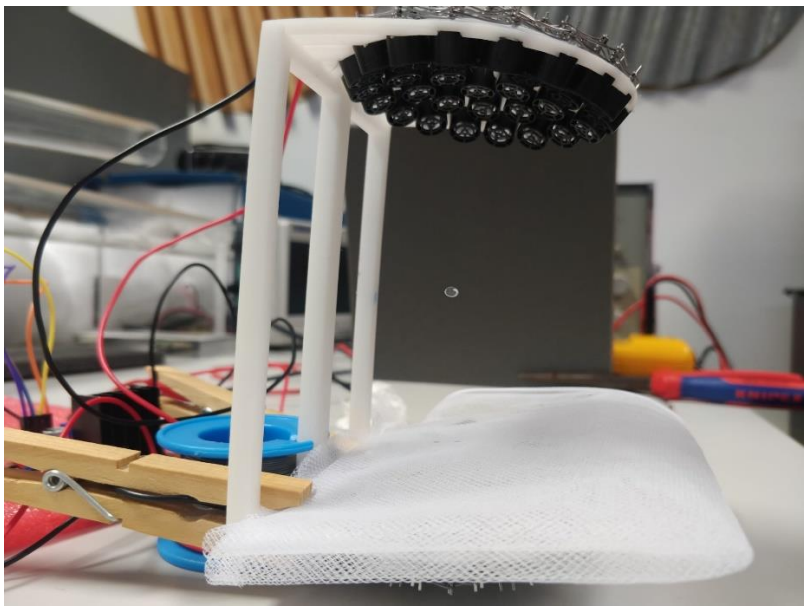


Ilustración 40: Levitación de una partícula de alcohol en el centro del dispositivo (objeto de este trabajo)

Este error es debido a que la *Ecuación 8*[13] es muy sensible al valor "z". Si se tiene en cuenta que la cápsula del micrófono utilizado para medir la presión es de ¼" (0,00635m), se comete un error de posición de 0,0127m en *z*.

Por tanto, realizando un gráfico en Excel para valores de "z" en el rango de 0,0495 ±0,00635 m tomando valores con una resolución de 0,001m, se obtiene la siguiente relación de z con la densidad:

Ilustración 41: Valor de densidad en función de z

z (m)	Peff (W)	Densidad (kg/m^3)
0,0430	0,84	89,72
0,0435	0,88	94,42
0,0440	0,99	106,74
0,0445	1,22	130,83
0,0450	1,65	177,12
0,0455	2,55	274,55
0,0460	4,89	526,95
0,0465	14,44	1558,05
0,0470	235,55	25429,91
0,0475	54,82	5918,03
0,0480	9,14	986,35
0,0485	3,76	405,95
0,0490	2,15	231,09
0,0495	1,46	157,24

z (m)	Peff (W)	Densidad (kg/m^3)
0,0500	1,12	120,61
0,0505	0,94	101,38
0,0510	0,86	92,02
0,0515	0,83	89,56
0,0520	0,87	93,29
0,0525	0,97	104,29
0,0530	1,17	126,17
0,0535	1,56	167,99
0,0540	2,36	254,28
0,0545	4,35	468,77
0,0550	11,70	1263,12
0,0555	110,42	11920,59
0,0560	90,44	9763,50

Tabla 8: valores de densidad

Se observa que la mayor densidad obtenida se encuentra en $z = 0{,}0470m$. Como estos valores nacen de un cálculo teórico y la resolución entre valores de "z" es pequeña, no es posible levitar objetos de esta densidad a nivel experimental. Hay que tener en cuenta además que se está considerando el mayor tamaño de partícula levitable ($\lambda = \frac{c}{f} = 0.0086m$; $\frac{\lambda}{2} = 0.0043m$), por lo que esta ocupa varias posiciones de "z". En este gráfico se está determinando el rango de posición donde una partícula podría levitar de forma estable (zona de estabilidad).

Mediante el gráfico obtenido de las simulaciones realizadas por Asier Marzo en el documento "*TinyLev: A multi-emitter single-axis acoustic levitator*", se tiene que para un valor de 12V, la densidad que es capaz de levitar el dispositivo es de 1200-1300kg/m^3 [29]:

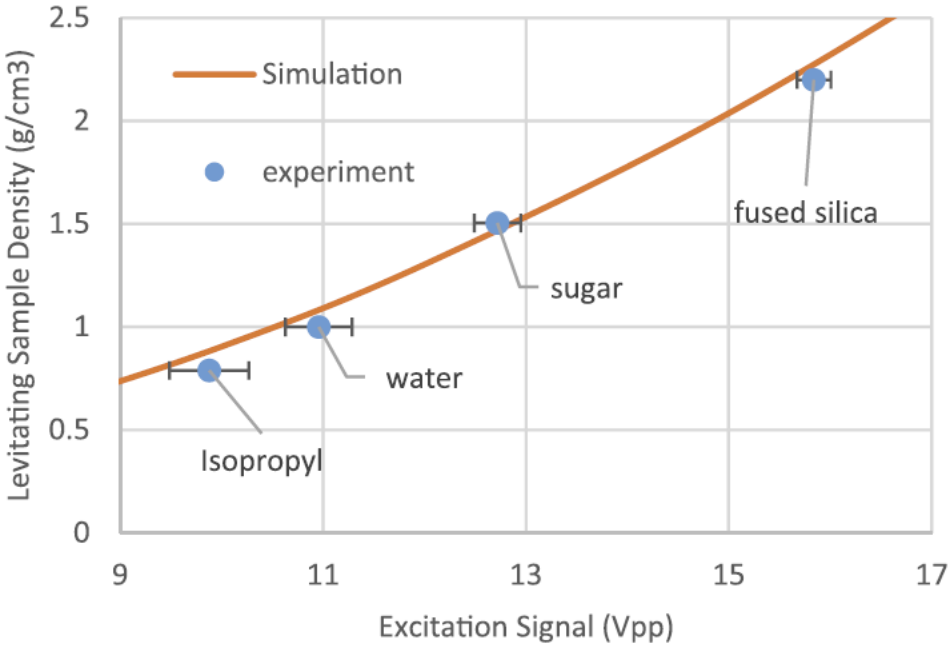


Ilustración 42: Simulación de levitación de densidad[29]

Por tanto, si se considera un valor de densidad de $\rho = 1250kg/m3$ se obtiene que:

$$p_{eff} = \left(\rho_s + \frac{9 \cdot \rho_0}{22}\right)\frac{4\,A\,c^2\,g}{11\,\pi\,f\,P_{eff}} = \left(1250 + \frac{9 \cdot 1{,}22}{22}\right)\frac{4 \cdot 0{,}002772 \cdot 343^2 \cdot 9{,}81}{11 \cdot \pi \cdot 40000} = 11{,}58W$$

$$z = \frac{c\cos^{-1}\left(\sqrt{\frac{p^2 A}{4c\rho_0 P_{eff}}}\right)}{\pi f} = 0{,}0479\text{m}$$

Este resultado está redondeado para obtener una solución realista, ya que como la fórmula es tan sensible a "z", un mínimo cambio decimal haría variar por completo la densidad resultante, por tanto, lo interesante es hallar una zona de estabilidad en la que se pueda levitar una densidad determinada.

7. Conclusión

En este proyecto se ha construido con éxito un levitador acústico no resonante uniaxial. Se han explicado los diferentes tipos de levitadores que existen hasta el momento, sus características a grandes rasgos y las aplicaciones que tiene la levitación acústica en distintos sectores, justificando porque es el mejor tipo de levitación para dichas aplicaciones frente a otros tipos.

Se ha programado el microprocesador Arduino con un código funcional que ha permitido realizar las pruebas y experimentos con éxito. Además se ha explicado el código empleado al detalle para lograr comprender cómo se generan las ondas que producen la levitación dentro del dispositivo.

Los fenómenos físicos que ocurren en la levitación acústica presentan una parte fundamental en el proyecto, por lo que se han explicado todos aquellos conceptos físicos en torno a la levitación acústica.

La realización de medidas ha ayudado a comprender mejor el funcionamiento del dispositivo y de la física involucrada y ha permitido obtener valores concretos como el cálculo de la densidad máxima que es capaz de levitar si se considera un volumen límite de media longitud de onda.

8. Futuras líneas de mejora

La levitación acústica es un campo muy reciente, esto permite que se desarrollen nuevos modelos de levitadores con propiedades mejoradas y mayor potencial. En el caso de este proyecto, podrían modificarse algunas características de este modelo para poder levitar elementos de mayor densidad. Para ello sería necesario aumentar la potencia. Esto podría ser posible si se utilizara un microprocesador y un driver que admitiera mayor voltaje junto con transductores de mayor tolerancia o tamaño. También podría colocarse un anillo exterior más de transductores en cada base de la estructura. Estos procedimientos se pueden realizar de forma independiente entre ellos, pero si se unen se podría obtener una potencia mayor.

Por otro lado, se podría aumentar el tamaño máximo que el dispositivo es capaz de levitar reduciendo la frecuencia de la onda estacionaria, pero para ello habría que tener en cuenta las dificultades planteadas en el punto **6.1**. Además, si se desea reducir la frecuencia, la separación entre las bases deberá ser mayor, de modo que para evitar que la onda emitida sufra una atenuación excesiva, se debería aumentar la potencia y el radio de orientación de los transductores para que estén siempre enfocados al centro del dispositivo[29]. Además, también se debería aumentar la cantidad de transductores para evitar que la fuerza lateral decaiga tal y como se muestra a continuación:

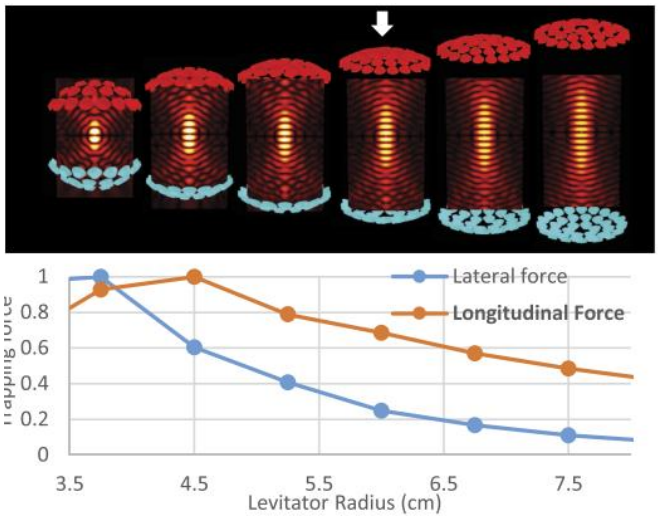


Ilustración 43: Efectos de la variación de distancia entre transductores[29]

Debido a que en el código se configuraron 6 botones de entrada de los cuales solo han sido utilizados 3, se podrían añadir nuevas funcionalidades al dispositivo que pudiesen ser útiles para casos concretos. Teniendo esto en cuenta, modificando la estructura y el código del levitador, estas funcionalidades podrían servir para efectuar un movimiento lateral de las partículas, de modo que el desplazamiento se pudiese realizar en 2 ejes. Por tanto, otra futura línea de mejora es poder generar movimiento en distintos ejes y no solo en vertical.

Bibliografía

[1] «Longitudinal Waves - Kundt's Tube». http://hyperphysics.phy-astr.gsu.edu/hbasees/Waves/kundtosc.html (accedido ago. 26, 2021).

[2] K. Bücks y H. Müller, «Über einige Beobachtungen an schwingenden Piezoquarzen und ihrem Schallfeld», *Z. Physik*, vol. 84, n.º 1, pp. 75-86, ene. 1933, doi: 10.1007/BF01330275.

[3] L. P. Gor'kov, «On the forces acting on a small particle in an acoustical field in an ideal fluid», en *Selected Papers of Lev P. Gor'kov*, 2014, pp. 315-317. doi: 10.1142/9789814366960_0008.

[4] «food for design: ACOUSTIC LEVITATION CHAMBER - DR DAVID DEAK». http://foodfordesign.blogspot.com/2007/02/acoustic-levitation-chamber-dr-david.html?m=0 (accedido ago. 26, 2021).

[5] W. J. Xie, C. D. Cao, Y. J. Lü, Z. Y. Hong, y B. Wei, «Acoustic method for levitation of small living animals», *Applied Physics Letters*, vol. 89, n.º 21, p. 214102, nov. 2006, doi: 10.1063/1.2396893.

[6] Onetnec, « La Levitación Acústica », *Pasele, pasele si hay.*, ago. 21, 2011. http://blogdetijuana.blogspot.com/2011/08/la-levitacion-acustica.html (accedido ago. 26, 2021).

[7] Y. Ochiai, T. Hoshi, y J. Rekimoto, «Three-Dimensional Mid-Air Acoustic Manipulation by Ultrasonic Phased Arrays», *PLOS ONE*, vol. 9, n.º 5, p. e97590, may 2014, doi: 10.1371/journal.pone.0097590.

[8] E. Press, «Cómo imprimir en 3-D un rayo tractor sónico para levitar objetos», ene. 03, 2017. https://www.europapress.es/ciencia/laboratorio/noticia-imprimir-rayo-tractor-sonico-levitar-objetos-20170103173214.html (accedido ago. 26, 2021).

[9] «¿Qué es el "rayo tractor sónico" que hace que los objetos leviten como en Star Trek?», *BBC News Mundo*. Accedido: ago. 26, 2021. [En línea]. Disponible en: https://www.bbc.com/mundo/noticias-42837561

[10] «La levitación acústica evoluciona y podría abrir puertas a nuevas aplicaciones - Olhar Digital». https://olhardigital.com.br/es/2020/11/20/videos/levitacao-acustica-evolui-e-pode-abrir-portas-para-novas-aplicacoes-3/ (accedido ago. 26, 2021).

[11] «Más cerca de hacer 'flotar' personas: nuevo método de levitación cuadruplica en potencia a los anteriores». https://nmas1.org//2018/01/24/levitacion-acustica (accedido ago. 26, 2021).

[12] masterDbplus, «Ingeniería acústica: levitación de objetos», *dBplus, ingeniería acústica*, jun. 08, 2016. https://www.dbplusacoustics.com/levitacion_acustica/ (accedido ago. 26, 2021).

[13] L. Wortsmann, «Stability of a Particle Levitated in an Acoustic Field», p. 4.

[14] Yoichi Ochiai, *Three-Dimensional Mid-Air Acoustic Manipulation [Acoustic Levitation] (2014-)*. Accedido: ago. 26, 2021. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=odJxJRAxdFU

[15] A. Marzo y B. W. Drinkwater, «Holographic acoustic tweezers», *PNAS*, vol. 116, n.º 1, pp. 84-89, ene. 2019, doi: 10.1073/pnas.1813047115.

[16] upna, *El rayo tractor sónico*. Accedido: ago. 26, 2021. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=Qlq5clzE9qQ

[17] «1759976.pdf». Accedido: sep. 01, 2021. [En línea]. Disponible en: https://www.farnell.com/datasheets/1759976.pdf

[18] «Arduino Nano», *Components101*. https://components101.com/microcontrollers/arduino-nano (accedido sep. 02, 2021).

[19] «▷ Tutorial de Arduino Nano [Pinout] (2021)». https://elosciloscopio.com/tutorial-arduino-nano-pinout/ (accedido sep. 02, 2021).

[20] «Controlar motores de corriente continua con Arduino y L298N», *Luis Llamas*. https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/ (accedido sep. 02, 2021).

[21] UpnaLab, «Acoustic Levitator», *Instructables*. https://www.instructables.com/Acoustic-Levitator/ (accedido sep. 02, 2021).
[22] jecrespom, «Entradas y Salidas Arduino», *Aprendiendo Arduino*, nov. 07, 2016. https://aprendiendoarduino.wordpress.com/2016/11/08/entradas-y-salidas-arduino/ (accedido ago. 26, 2021).
[23] «PWM in AVR», *[Curiosity,Experimentation]*, sep. 29, 2010. https://appusajeev.wordpress.com/2010/09/30/pwm-in-avr/ (accedido ago. 26, 2021).
[24] «datasheet.pdf».
[25] «Arduino en español: goto», *Arduino en español*. http://manueldelgadocrespo.blogspot.com/p/goto.html (accedido ago. 26, 2021).
[26] S. Zhao y J. Wallaschek, «A standing wave acoustic levitation system for large planar objects», *Arch Appl Mech*, vol. 81, n.º 2, pp. 123-139, feb. 2011, doi: 10.1007/s00419-009-0401-3.
[27] «srep20023-s1.pdf».
[28] M. A. B. Andrade, F. T. A. Okina, A. L. Bernassau, y J. C. Adamowski, «Acoustic levitation of an object larger than the acoustic wavelength», *The Journal of the Acoustical Society of America*, vol. 141, n.º 6, pp. 4148-4154, jun. 2017, doi: 10.1121/1.4984286.
[29] A. Marzo, A. Barnes, y B. W. Drinkwater, «TinyLev: A multi-emitter single-axis acoustic levitator», *Review of Scientific Instruments*, vol. 88, n.º 8, p. 085105, ago. 2017, doi: 10.1063/1.4989995.