Universidad Zaragoza
1542

Trabajo Fin de Grado

Proyección de Figuras de Lissajous a Partir de Análisis Espectral.

Projecting Lissajous Figures from Spectral Analysis

Autor

David Martín-Sacristán Avilés

Director

David Asiain Ansorera

Escuela Universitaria Politécnica La Almunia

Febrero 2025

Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Proyección de Figuras de Lissajous a Partir de Análisis Espectral.

Projecting Lissajous Figures from Spectral Analysis

Identificador

Autor: David Martín-Sacristán Avilés

Director: David Asiain Ansorera

Fecha: 11/2024

Página intencionadamente en blanco.

INDICE DE CONTENIDO BREVE

INDICE DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE ECUACIONES

1. RESUMEN

El propósito de este proyecto de final de grado (TFG) es crear un sistema para regular motores que se basa en la identificación de frecuencias predominantes de una señal musical en tiempo real. A través de LabVIEW, se lleva a cabo un análisis de las propiedades espectrales de una pieza musical, logrando identificar las frecuencias más significativas, que se utilizarán para controlar la velocidad de los motores mediante señales PWM generadas por un microcontrolador.

Este método tiene como objetivo investigar cómo el sonido se relaciona con el movimiento mecánico, ofreciendo una solución novedosa en el ámbito del control de sistemas electromecánicos. Se han realizado diversas pruebas para asegurar una correcta identificación de frecuencias, el procesamiento de la señal y la transformación de los datos en órdenes de control para los motores. Sin embargo, el sistema enfrenta problemas técnicos como la fricción mecánica y la necesidad de mejorar el arranque de los motores, los cuales son examinados en el documento.

Además, el análisis incluye una evaluación comparativa entre motores convencionales y motores brushless para identificar cuál de estos proporciona un rendimiento superior en esta aplicación particular.

1.1. PALABRAS CLAVE

Control de motores, PWM, LabVIEW, análisis espectral, Transformada de Fourier Rápida (FFT), frecuencia dominante, señales musicales, fricción mecánica, filtro de Hilbert, figuras de Lissajous, motores brushless, sincronización, modulación de velocidad, control electromecánico.

2. ABSTRACT

The purpose of this final degree project is to create a system for regulating motors based on the identification of the predominant frequencies of a musical signal in real time. Using LabVIEW, an analysis of the spectral properties of a piece of music is carried out, identifying the most significant frequencies, which will be used to control the speed of the motors by means of PWM signals generated by a microcontroller.

This method aims to investigate how sound is related to mechanical movement, offering a novel solution in the field of electromechanical systems control. Several tests have been carried out to ensure correct frequency identification, signal processing and transformation of the data into control commands for the motors. However, the system faces technical problems such as mechanical friction and the need to improve motor starting, which are examined in the paper.

In addition, the analysis includes a comparative evaluation between conventional and brushless motors to identify which of these provides superior performance in this particular application.

2.1. KEY WORDS

Motor control, PWM, LabVIEW, spectral analysis, Fast Fourier Transform (FFT), dominant frequency, musical signals, mechanical friction, Hilbert filter, Lissajous figures, brushless motors, synchronisation, speed modulation, electromechanical control.

3. INTRODUCCIÓN

En este proyecto se elabora un sistema de gestión de motores que se basa en la modulación de señales PWM según las frecuencias predominantes obtenidas de una canción en tiempo real. A través de LabVIEW, se lleva a cabo un análisis espectral de la señal musical usando la Transformada Rápida de Fourier (FFT), generando información esencial que se envía a un microcontrolador para ajustar la velocidad de los motores.

Se llevaron a cabo múltiples experimentos para medir el rendimiento del sistema, abarcando la estabilidad de la señal PWM, la respuesta de los motores y el efecto que tiene la fricción sobre el movimiento. También se incorporaron métodos avanzados de análisis de señales, como la envolvente utilizándola transformada de Hilbert y la creación de figuras de Lissajous, con el fin de mejorar la sincronización y la estabilidad del control.

Los hallazgos indicaron que aunque el sistema funciona, la fricción mecánica tiene un impacto en el movimiento, lo que implica la necesidad de estrategias de optimización, incluyendo la reducción de áreas de contacto. Además, se examinó la posibilidad de utilizar motores brushless como alternativa a los motores tradicionales.

Esta investigación muestra el potencial del uso de señales musicales para dirigir el movimiento de motores en contextos creativos y audiovisuales.

4. DESARROLLO.

4.1. ANTECEDENTES.

Para entender bien este proyecto empezaremos explicando el principio básico de la creación de las figuras de Lissajous y como mediante tecnologías adyacentes podremos introducirlo en el videomapping.

Las figuras de lissajous son patrones que resultan por la superposición de dos oscilaciones perpendiculares con diferentes frecuencias o fases. Estas figuras fueron estudiadas inicialmente por Nathaniel Bowditch en 1815 y más tarde por Jules Antoine Lissajous en 1857. Estas curvas tienen aplicaciones en múltiples disciplinas, como la física, las matemáticas, la electrónica y el arte, debido a su capacidad para representar gráficamente relaciones entre frecuencias y fases de señales periódicas.

Los avances en tecnología, como los sistemas de proyección láser y los microcontroladores, nos han permitido explorar nuevas formas de mostrar y manipular estas figuras en tiempo real. Los espejos móviles accionados por motor, a menudo llamados galvanómetros o servomotores, son componentes clave para la proyección dinámica de patrones geométricos mediante láser. Estos sistemas se utilizan ampliamente en espectáculos de luz y sonido y en experimentos científicos que estudian la interacción de las ondas.

Por otro lado, el análisis de señales musicales ha cobrado relevancia en la última década debido al desarrollo de herramientas como en mi caso, LabVIEW (un entorno de programación gráfica que facilita el procesamiento y análisis de señales en tiempo real). Extraer las frecuencias dominantes de una señal musical puede crear una correspondencia entre las propiedades sonoras y las propiedades visuales proyectadas, lo que da como resultado una experiencia audiovisual inmersiva.

En este caso, el uso de microcontroladores como Arduino ha demostrado ser una solución eficaz y fácilmente disponible para implementar un sistema integrado de hardware y software. Arduino permite la precisión y sincronización de dispositivos electrónicos como motores y láseres, abriendo posibilidades creativas para proyectos.

Este trabajo investiga el uso de un sistema de proyección láser controlado por dos espejos móviles colocados en un plano vertical para generar figuras de Lissajous. Se utilizó LabVIEW para obtener y analizar la frecuencia dominante de la canción, la cual se utilizó para generar la señal que controla el motor responsable del movimiento del espejo. De

esta forma, el sistema transforma cualidades musicales en animaciones visuales, creando una conexión entre el sonido y la luz con la ayuda de la tecnología.

4.2. MARCO TEÓRICO.

En este apartado se explicará teóricamente todas las definiciones, principios, leyes o teorías necesarias para comprender el proyecto.

4.2.1. Curvas de Lissajous.

Este fenómeno se podría definir matemáticamente como la representación de un sistema de ecuaciones paramétricas cuyas ecuaciones correspondientes a los ejes x e y, son las ecuaciones de la superposición de dos movimientos armónicos simples, por lo que podríamos decir:

$$x = A * \sin(\omega_x * t + \alpha), \qquad y = N * \sin(\omega_y * t + \beta), \qquad \delta = \alpha - \beta$$

Ecuación 1 Ecuacion Onda Sinusoidal

Siendo x e y las ecuaciones correspondientes a los ejes x e y, ω_x y ω_y la frecuencia de cada movimiento armónico, $\alpha\, y\, \beta$ es la fase de cada movimiento, y δ es la diferencia de fase.

La figura que queramos dibujar va a estar definida por la relación que hay entre las frecuencias, eso quiere decir que si la relación entre ω_x / ω_y va a condicionar la imagen obtenida, y la diferencia de fase.

Si la relación entre las frecuencias es uno la figura que obtendremos será una elipse, pero si ahora modificamos la fase de cualquiera de las dos haciendo que la diferencia de fase sea 90o la figura que obtendremos será un círculo, y cuando su diferencia de fase sea 0 o 180 o la figura es una línea, si su relación es un dos obtenemos un infinito, pero si es ½ es un 8. Si su relación es 3 obtenemos la figura de un caramelo.

Mediante este experimento se ve que una forma muy sencilla de voltear el dibujo 90 o es cambiando la relación de las frecuencias y si es 2 entonces con la relación ½ el infinito se levanta y se convierte en un ocho, por lo que se observa una relación entre las figuras, si la relación es 2/3 significara que se cortara 3 veces en el eje x y 2 en el eje y.

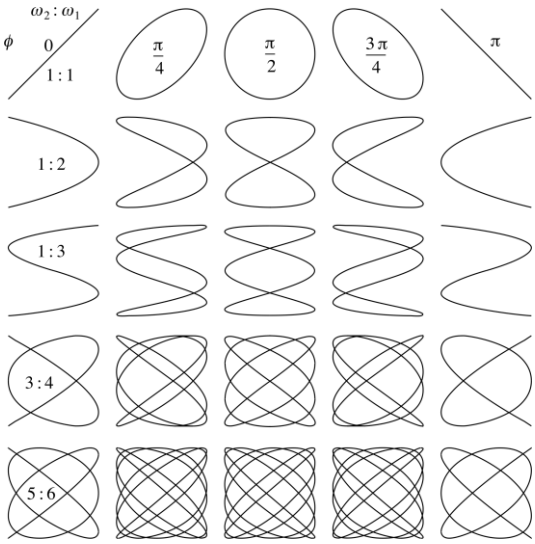


Ilustración 1 Figuras Representativas en función a la relación de frecuencias

4.2.2. Digitalización de Señal.

Para la digitalización de una señal analógica, es necesario tener en cuenta los siguientes conceptos:

4.2.2.1. Ancho de Banda.

El ancho de banda se define como la capacidad del digitalizador para obtener una señal analógica y convertirla a digital perdiendo la mínima amplitud posible. El ancho de banda describe el rango de frecuencias que el digitalizador puede medir con precisión.

La frecuencia quedaría definida por el 70.7% o a -3dB de su amplitud original.

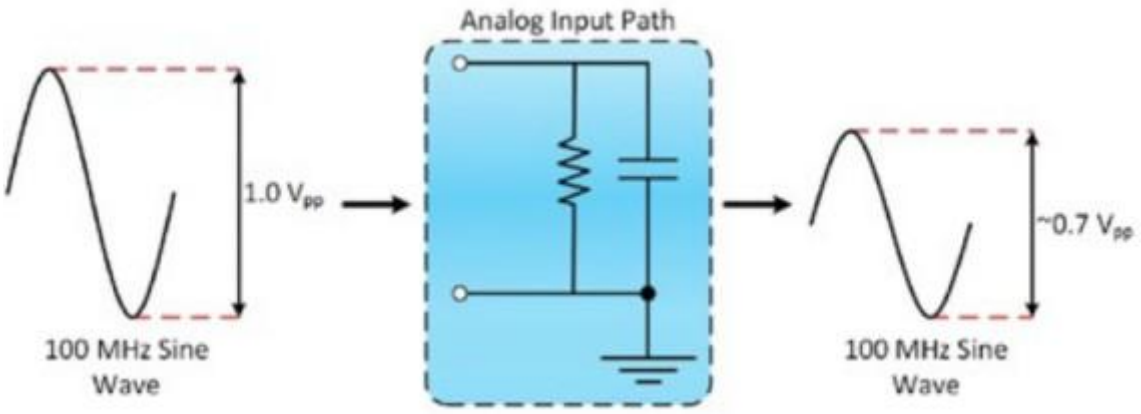


Ilustración 2 Atenuación de una señal al 70.7% o -3dB

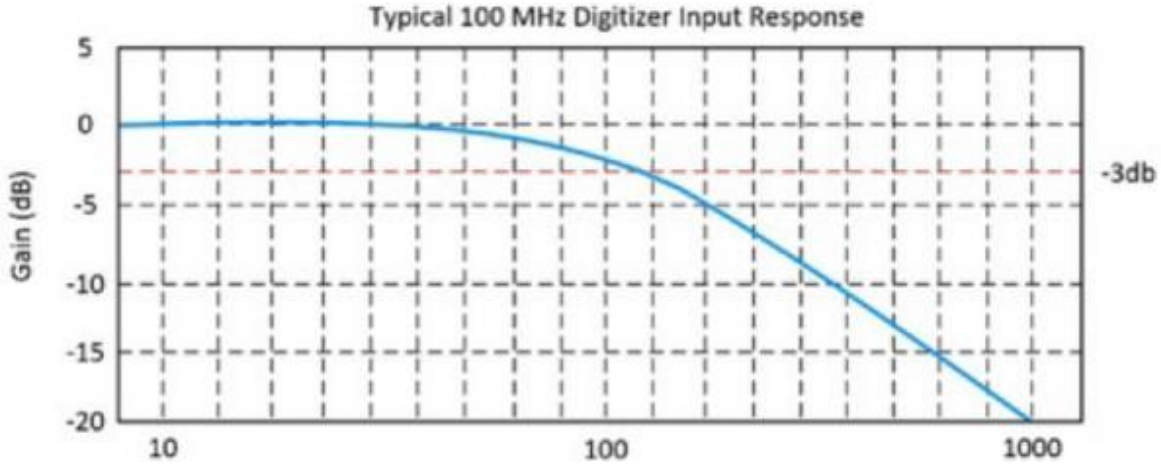


Ilustración 3 Señal de entrada a 100MHz llega a -3dB

El ancho de banda se mide entre los puntos de frecuencia inferior y superior donde la amplitud de la señal cae a -3 dB por debajo de la frecuencia de pasa banda.

Primero calculamos el valor de -3dB

$$-3dB = 20 * \log \frac{V_{out.pp}}{V_{in.pp}}$$

Ecuación 2 Calcular -3dB

$V_{in.pp}$ es el voltaje pico a pico de la señal de entrada, $V_{out.pp}$ Es el voltaje pico a pico de la señal de salida.

Dado que la señal de entrada es sinusoidal, hay dos frecuencias (f_1 y f_2) en las que la señal alcanza ese valor, denominadas frecuencias de corte, mientras que la frecuencia central f_0 es la media geométrica de dichas frecuencias de corte.

$$f_0 = \sqrt{f_1 * f_2}$$

Ecuación 3 Frecuencia Central

$$BW = f_2 - f_1$$

Ecuación 4 Ancho de Banda

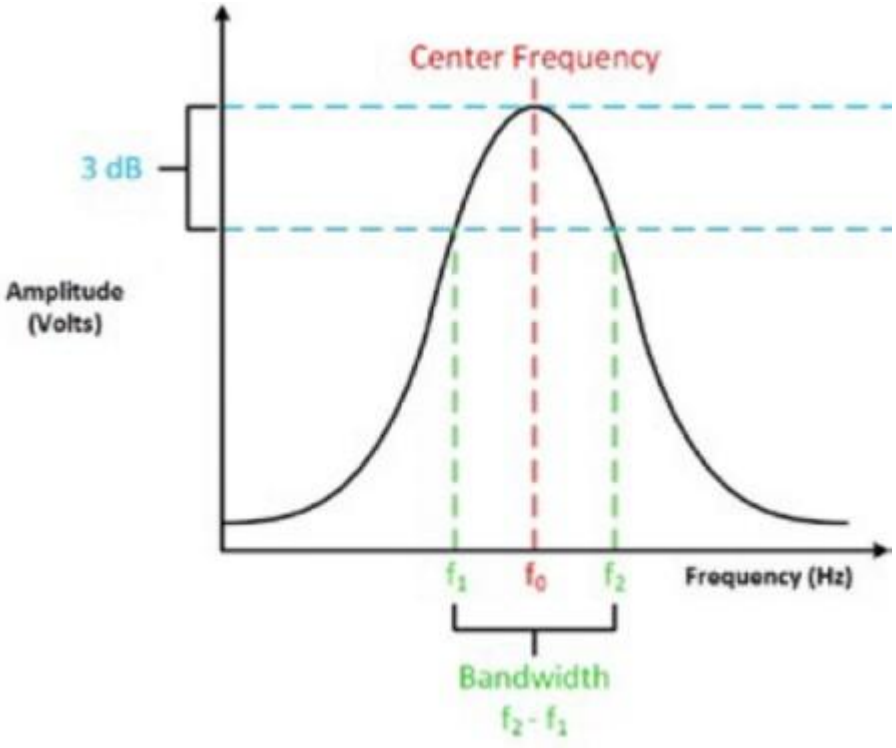


Ilustración 4 Ancho de banda, Frecuencias de corte, Frecuencia central y punto -3dB

4.2.2.1.1. Error de Amplitud.

Se expresa como un porcentaje.

$$Error\ de\ Amplitud = 100 * \left(1 - \frac{R}{\sqrt{1+R^2}}\right), Para\ R = \frac{BW}{f_{in}}$$

Ecuación 5 Error relativo de Amplitud

R es la relación entre el ancho de banda y la frecuencia de la señal de entrada f_{in}.

Se recomienda que el ancho de banda del digitalizador sea de tres a cinco veces superior que la máxima frecuencia de la señal medida. Ahora calcularemos el error para una señal de 1V a 100MHz usando un digitalizador de 100MHz, de 300MHz y 500MHz.

$$Error\ de\ Amplitud\ 100MHz = 100 * \left(1 - \frac{1}{\sqrt{2}}\right) = 29.28\%$$

$$Error\ de\ Amplitud\ 300MHz = 100 * \left(1 - \frac{3}{\sqrt{1+3^2}}\right) = 5.13\%$$

$$Error\ de\ Amplitud\ 500MHz = 100 * \left(1 - \frac{5}{\sqrt{1+25^2}}\right) = 1.94\%$$

4.2.2.1.2. Tiempo de Incremento.

Como hemos visto para digitalizar la señal se debe tener más ancho de banda que la frecuencia que se desea medir, pero también es necesario tener suficiente tiempo para capturar los detalles de las transiciones rápidas.

El tiempo de incremento de una señal de entrada es el tiempo que tarda una señal en pasar del 10% al 90% de la amplitud máxima de la señal.

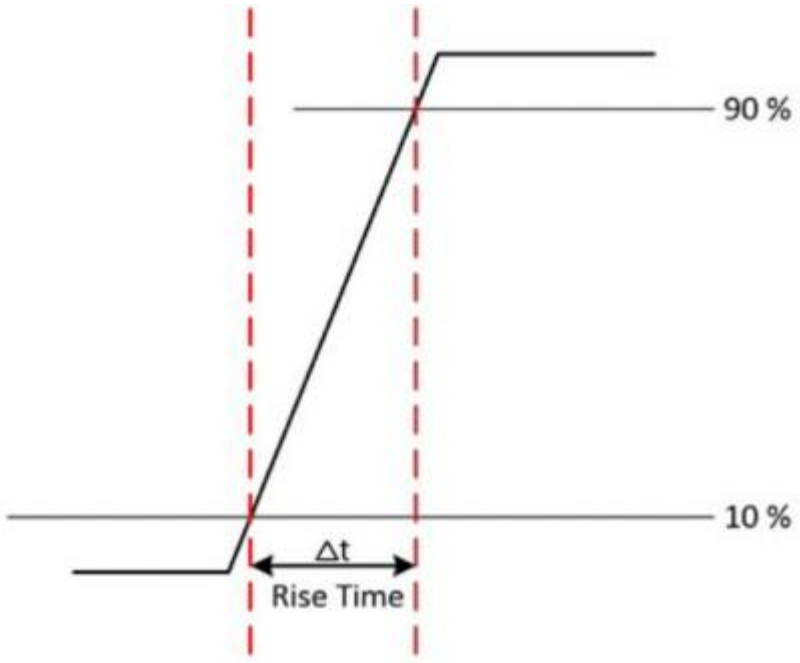


Ilustración 5 Tiempo que tarda una señal en pasar del 10% al 90% de la amplitud máxima

$$T_r = \frac{k}{BW}$$

Ecuación 6 Tiempo de Incremento

La constante k depende del digitalizador. En la mayoría de los casos con un ancho de banda superior a 1GHz suele ser k=0.4 ó 0.45, mientras que para los inferiores k=0.35.

El tiempo teórico t_{rm}, se puede calcular a partir del tiempo del digitalizador $t_{r\text{-}d}$ y la el de la señal de entrada $t_{r\text{-}in}$.

$$t_{rm} = \sqrt{{t_{r-d}}^2 + {t_{r-in}}^2}$$

Ecuación 7 Tiempo Teórico

Se recomienda que el tiempo de incremento del digitalizador será de un tercio a un quinto del tiempo de incremento de la señal medida.

4.2.2.2. Velocidad de Muestreo.

La velocidad de muestreo es la frecuencia a la que el ADC convierte la forma de onda de entrada analógica en datos digitales. El digitalizador muestrea la señal después que se haya aplicado cualquier modificación (atenuaciones, filtros ganancias, etc.) y convierte la señal analógica en digital, por lo tanto, cuanto más rápido muestree el digitalizador, mayor será la resolución y el detalle que se puede observar en la onda.

4.2.2.2.1. Teorema de Muestreo de Nyquist.

Este teorema explica la relación entre la velocidad de muestreo y la frecuencia de la señal medida. Dice que la velocidad de muestreo f_s debe ser mayor que el doble de la frecuencia más alta en la señal medida. Esta frecuencia se suele conocer como la frecuencia de Nyquist, f_N.

$$f_s > 2 * f_N$$

Ecuación 8 Teorema de Muestreo

Para entenderlo observaremos una onda sinusoidal en 3 casos diferentes:

- A: frecuencia de señal = frecuencia de muestreo.
- B: frecuencia de señal = 2 * frecuencia de muestreo.
- C: frecuencia de señal = 4/3 frecuencia de muestreo.

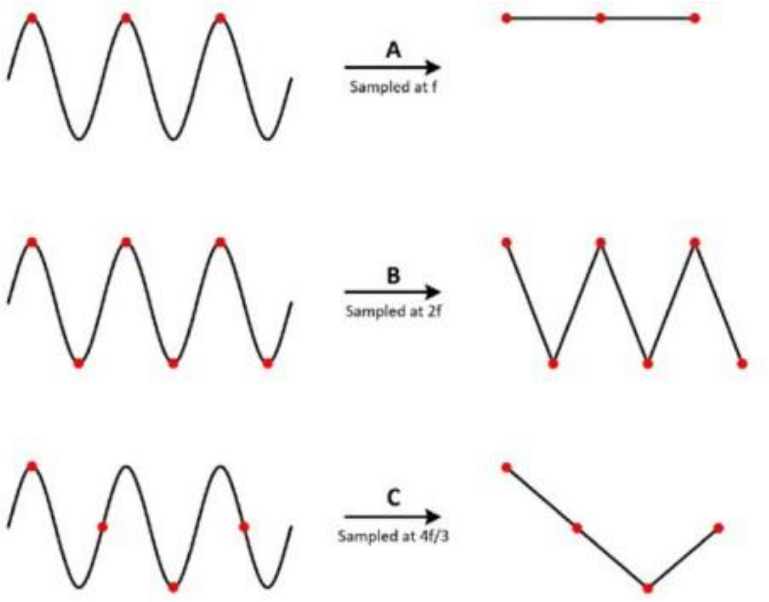


Ilustración 6 Diferentes casos de velocidades de muestreo.

Como podemos observar para tener una mínima similitud a la onda por lo menos en frecuencia la frecuencia de muestreo debe ser el doble que la de la señal.

4.2.2.2.2. Aliasing.

Si muestreamos a una velocidad inferior a la velocidad de Nyquist, aparecen componentes falsos de frecuencias más bajas en los datos muestreados. Este fenómeno se conoce como Aliasing. En la siguiente figura podemos observar como una onda sinusoidal de 800KHz muestreada a 1Ms/s, la línea de puntos muestra la señal con aliasing que es la que se observaría, apareciendo con una frecuencia de 200KHz.

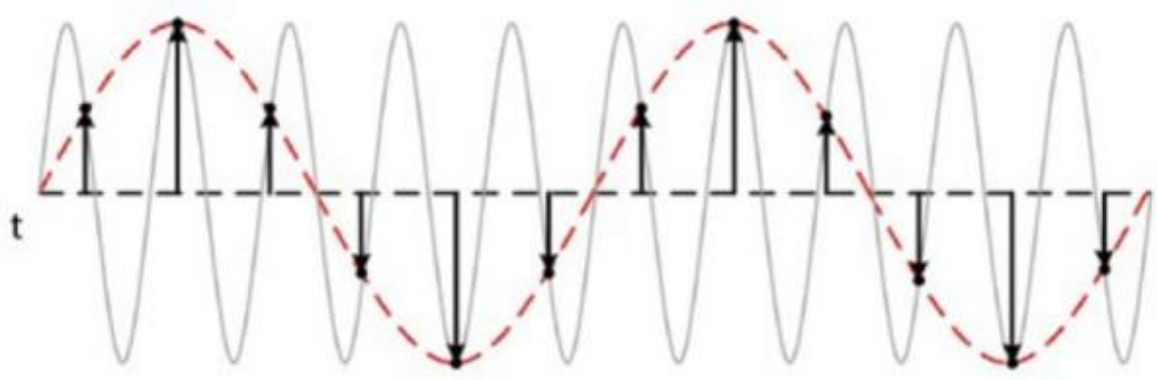


Ilustración 7 Aliasing de 800KHz a 200KHz

4.2.2.2.3. Resolución.

Los bits de resolución se refieren al número de niveles verticales únicos que un digitalizador puede usar para representar una señal.

La resolución de un ADC es una función de en cuántas partes se puede dividir la señal máxima. La resolución de amplitud está limitada por el número de niveles de salida discreta que tiene un ADC. Un código binario representa cada división, como tal, el número de niveles se puede calcular de la siguiente manera:

$$numero\ de\ niveles = 2^{Resolución}$$

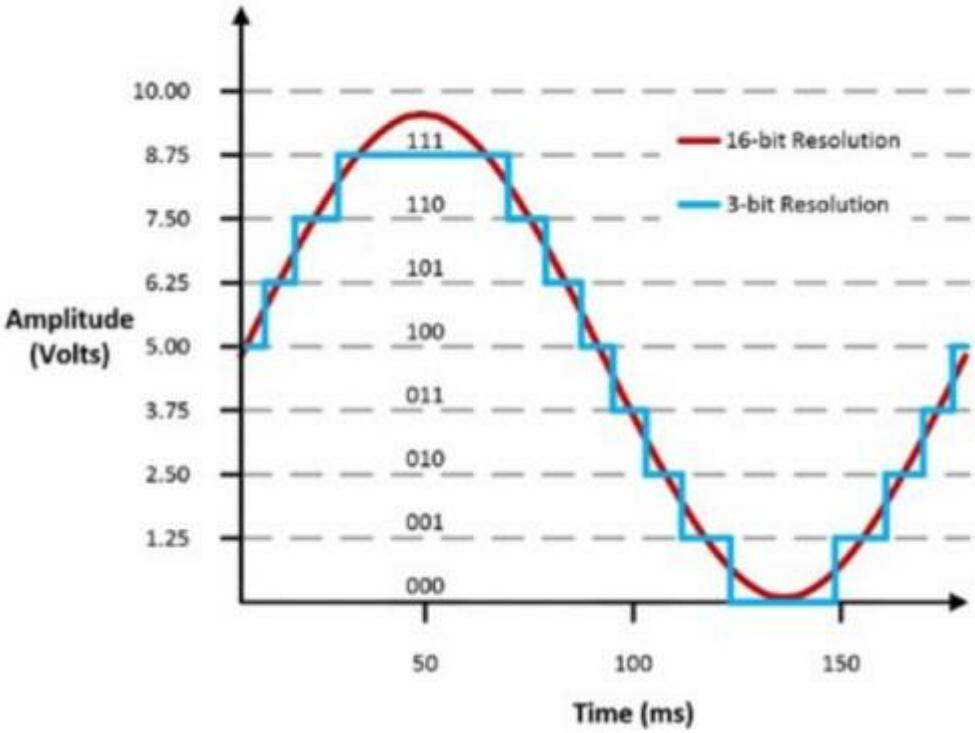


Ilustración 8 Diferencia entre 3 bit y 16 bit de resolución

4.2.3. Análisis espectral.

El análisis espectral es una disciplina que estudia las propiedades de las señales en función de su frecuencia. Permite descomponer una señal compleja en sus componentes frecuenciales, proporcionando información sobre su contenido energético, patrones repetitivos y posibles ruidos.

El análisis espectral se basa en las matemáticas de las series de Fourier y la transformada de Fourier, las cuales permiten representar una señal en el dominio del tiempo como una combinación de senos y cosenos en el dominio de la frecuencia. Otros métodos avanzados, como las transformadas de Laplace y de Wavelet, también se utilizan dependiendo de la aplicación específica.

4.2.4. Transformada de Fourier.

El principio básico de esta transformada radica en que cualquier función de tiempo, siempre que cumpla ciertas condiciones, puede expresarse como una suma (o integral) de funciones sinusoidales con diferentes frecuencias, amplitudes y fases.

Las señales pueden clasificarse según diferentes criterios:

- Señales continuas y discretas: Una señal continua se define para todo instante de tiempo, mientras que una señal discreta está definida solo en ciertos intervalos de tiempo.
- Señales periódicas y aperiódicas: Las señales periódicas se repiten en intervalos regulares de tiempo, mientras que las señales aperiódicas no tienen un patrón de repetición.

En el dominio del tiempo, una señal describe cómo varía una magnitud en función del tiempo.

En el dominio de la frecuencia, la señal se descompone en sus componentes de frecuencia (espectro), lo que permite analizar cómo se distribuyen las diferentes frecuencias que componen la señal.

La Transformada de Fourier de una función x(t) se define como:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt$$

Ecuación 9 Transformada de Fourier

- X(f): Representación de la señal en el dominio de la frecuencia.
- x(t): Señal en el dominio del tiempo.
- f: Frecuencia en hercios (Hz).
- j: Unidad imaginaria ($j = \sqrt{-1}$).

La transformada inversa, que permite reconstruir la señal original se define como:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft}\, dt$$

Ecuación 10 Transformada Inversa de Fourier

A su vez han aparecido variantes de la transformada de Fourier, como podrían ser:

- **Transformada Discreta de Fourier (DFT)**

  La DFT se aplica a señales discretas y tiene la siguiente definición:

$$x(k) = \sum_{n=0}^{N-1} x[n] * e^{-j*\frac{2\pi kn}{n}}, k = 0,1, \dots N-1$$

Ecuación 11 Transformada directa de Fourier

  Donde N es el número de muestras en la señal.

- **Transformada Rápida de Fourier (FFT)**

  La FFT es un algoritmo eficiente para calcular la DFT, reduciendo el costo computacional de $O(N^2)$ a O (N*log (N)). Esto la hace ideal para aplicaciones prácticas como procesamiento de audio, video y datos en tiempo real.

- **Transformada de Fourier de Tiempo Corto (STFT)**

  La STFT permite analizar señales no estacionarias dividiendo la señal en segmentos temporales pequeños y aplicando la Transformada de Fourier a cada segmento:

$$x(t,f) = \int_{-\infty}^{\infty} x(\tau) * \omega(t-\tau) e^{-j2\pi f\tau}\, d\tau$$

Ecuación 12 Transformada de Fourier de Tiempo Corto

  Donde w(t) es una ventana que define el intervalo de análisis.

4.2.5. Transformada de Hilbert.

Los resultados de la Transformada de Fourier se presentan como una gráfica de los valores absolutos de los coeficientes frente a sus respectivas frecuencias, esta grafica se le denomina espectro de Fourier. Sin embargo, la distribución en el tiempo del contenido de frecuencias no se puede determinar. Este es uno de los mayores problemas que presenta el análisis de Fourier. Si la señal es lineal y estacionaria este inconveniente no es muy grave, pero muchas señales no lo son, por lo que será necesario realizar un análisis en ambos dominios, tiempo y frecuencia, simultáneamente.

La transformada de Hilbert tiene el inconveniente de solo poder usarse con señales mono componentes, es decir con un solo tono de frecuencia. Pero gracias al desarrollo del método de la descomposición modal empírica que permite descomponer la señal en monocomponentes, facilita así el estudio de la señal en ambos dominios.

La transformada de Hilbert (HT) de una función real *x(t)* se define como:

$$H[x(t) = \frac{1}{\pi} * \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau$$

Ecuación 13 Hahn (1996)

La HT es la convolución entre la función y el inverso del tiempo. La HT se utiliza sobre todo para reconstruir la señal analítica *z(t)*.

$$z(t) = x(t) + iy(t) = x(t) + iH[x(t)]$$

Ecuación 14 Gabor (1946)

La señal analítica es una señal compleja cuyo espectro en frecuencias negativas es nulo y en las reales es igual a la señal original.

$$z(t) = a(t) * e^{i\theta(t)}$$

Ecuación 15 Otra forma de expresar Gabor

Donde:

$$a(t) = \sqrt{x^2 + y^2} \qquad y \qquad \theta(t) = \arctan\left(\frac{y}{x}\right)$$

Ecuación 16 Definición Variables

Es decir, la señal analítica permite separar una señal en sus componentes de amplitud *a(t)* y fase instantánea $\theta(t)$. Y gracias a estas ecuaciones se propuso el concepto de frecuencia instantánea *(IF)* como la derivada del tiempo de la fase.

$$IF(t) = \frac{1}{2\pi}\frac{d}{dt}\theta(t)$$

Ecuación 17 Frecuencia Instantanea

La siguiente figura muestra el resultado del análisis usando HT de una señal sinusoidal de amplitud constante donde la frecuencia crece linealmente en el tiempo desde 0.5 Hz hasta 8 Hz (señal "chirp"). El apartado a) muestra la amplitud instantánea y podemos apreciar que la amplitud que se obtiene es la envolvente de la señal en el tiempo. En el apartado b) muestra los resultados obtenidos para la dase instantánea. Las discontinuidades que se observan son debidas al uso de la función arctan de la Ecuación 16 Definición Variables, la cual siempre va a presentar saltos entre π y −π. En el apartado c) la frecuencia instantánea es calculada directamente usando diferenciación numérica de los valores de fase instantánea, los resultados obtenidos van a exhibir discontinuidades en los mismos instantes donde la fase instantánea presenta discontinuidades. A parte la evolución de las frecuencias en el tiempo es identificada satisfactoriamente. Para resolver el problema de las discontinuidades en la fase existen diferentes técnicas. La más simple seria sumarle 2π a la fase cada vez que se completa un ciclo para suavizar las discontinuidades antes de tomar la derivada. Otra opción es calcular las fases de las diferencias en lugar de las diferencias de las fases.

$$IF_n = \frac{1}{2\bar{\pi}} * \arctan\left(z_n * conj(z_{n+1})\right)$$

Ecuación 18 Feldman (2011)

Las frecuencias instantáneas que se muestran en el apartado d) fueron obtenidas usando la Ecuación 18 Feldman (2011). Se puede apreciar que, sin necesidad de previamente desenrollar la fase, la variación de las frecuencias en el tiempo es identificado satisfactoriamente y sin discontinuidades.

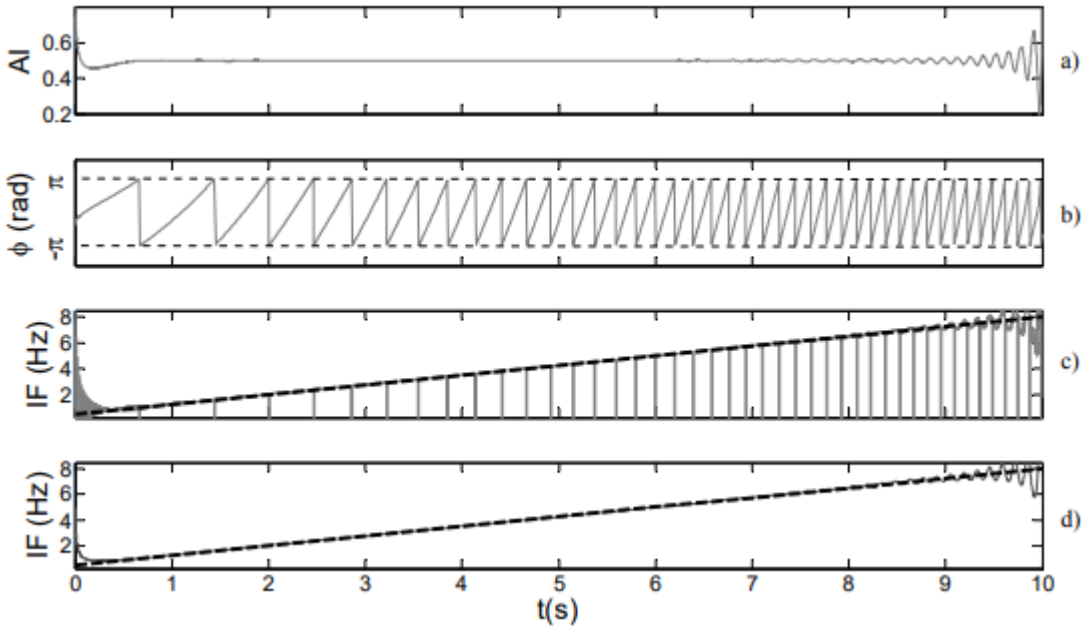


Ilustración 9 Análisis de la señal a través de la HT

La determinación de frecuencias instantáneas usando la señal analítica obtenida a partir de la transformada de Hilbert solo arrojará resultados satisfactorios si la señal es mono componente. Mientras que cuando la señal presenta varias frecuencias simultáneamente, el resultado obtenido para la frecuencia instantánea será un solo valor por instante de tiempo, el valor promedio. Por ejemplo, una señal generada a partir de la suma de dos sinusoides de amplitud constante y frecuencias de 2 Hz y 6 Hz. Si la frecuencia instantánea de esta señal es analizada, el resultado que se obtiene es una frecuencia de 4 Hz. Para aplicar exitosamente la HT a señales con múltiples tonos, la señal original debe ser preprocesada para descomponerla en monocomponentes, por ejemplo, a través del uso de filtros, descomposición modal empírica (Huang, 1998), descomposición vibracional Hilbert (Feldman, 2006) o la transformada wavelet "synchrosqueezed" (Daubuchies et al., 2011).

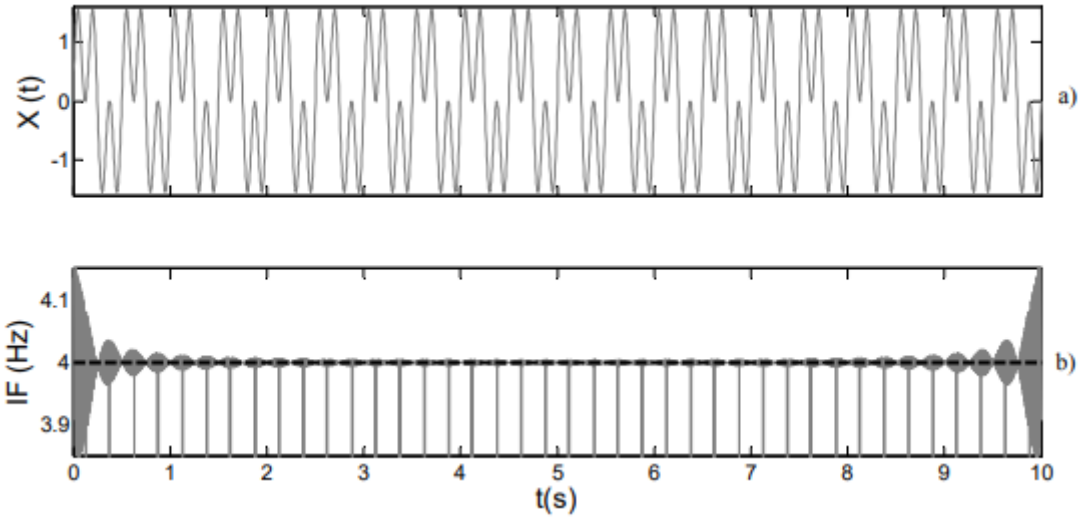


Ilustración 10 a) Sinusoidal compuesto por frecuencias de 2 y 6 Hz; b) Frecuencia instantánea usando la transformada de Hilbert.

4.2.6. Filtros

Los filtros son dispositivos o algoritmos diseñados para modificar ampliar o reducir características de una señal. Dependiendo de su implementación, los filtros pueden clasificarse en dos grandes categorías: digitales y analógicos.

4.2.6.1. Filtros analógicos.

Los filtros analógicos operan directamente sobre señales continuas en el tiempo, utilizando componentes pasivos (resistencias, condensadores y bobinas) y activos (amplificadores operacionales). Son comunes en aplicaciones donde la señal no ha sido digitalizada.

Hay cuatro tipos de filtros típicos, Pasa bajos, Pasa altos, Pasa banda y, rechazo de banda. Hay que destacar dos tipos específicos de filtros analógicos:

4.2.6.2. Filtros Butterworth.

Están diseñados para tener una respuesta en frecuencia lo más plana posible dentro de su banda de paso. Este diseño minimiza las distorsiones en la amplitud de las señales filtradas, haciéndolos adecuados para aplicaciones donde lo que buscamos es la fidelidad en amplitud.

Estos filtros pueden ser digitalizados, manteniendo su respuesta plana y permitiendo ajustar fácilmente los parámetros en aplicaciones digitales, como el procesamiento de audio.

4.2.6.3. Filtros Exponenciales.

Son comunes en aplicaciones de sistemas de control y modelado, especialmente cuando se busca suavizar rápidamente los datos o atenuar componentes no deseados.

Al digitalizarse, los filtros exponenciales se convierten en algoritmos eficientes para suavizar datos en tiempo real.

El filtro EMA implica calcular un valor filtrado usando la expresión siguiente a partir de una medición.

$$A_n = \alpha M + (1 - \alpha) A_{n-1}$$

Ecuación 19 Filtro Exponencial

Con A_n representando el valor filtrado, A_{n-1} el valor filtrado previo, M como el valor de la señal a filtrar, y Alpha como un factor en el rango de 0 a 1.

Así, el filtro EMA proporciona información adicional a través de la medición M y también suaviza los datos utilizando la memoria del valor filtrado anterior An-1. Posteriormente, analizaremos cómo el factor Alpha afecta la cantidad de suavizado en la señal resultante de un filtro exponencial EMA.

Es evidente la ventaja en términos de facilidad y eficiencia computacional. La computación necesita una sola instrucción simple. En lo que respecta a los requisitos de memoria, solo necesitamos guardar el valor filtrado previo.

4.2.6.4. Filtros Digitales.

Los filtros digitales operan sobre señales discretas en el tiempo, manipulando datos numéricos mediante algoritmos implementados en microprocesadores, DSPs (Procesadores Digitales de Señales) o software.

Permiten ajustes precisos y adaptativos mediante cambios en el código o en los parámetros del filtro. Habiendo dos tipos de filtros digitales principalmente:

4.2.6.5. Filtros FIR (Finite Impulse Response).

Tienen respuesta finita en el tiempo, por lo que su salida depende únicamente de un número limitado de valores pasados de la entrada.

Son estables, ya que no tienen polos. Permiten un control preciso de la respuesta en frecuencia gracias a los cálculos matemáticos. Y no introducen distorsión de fase si se diseñan como lineales.

Se suelen utilizar en procesamiento de imágenes, telecomunicaciones, y eliminación de ruido.

4.2.6.6. Filtros IIR (Infinite Impulse Response).

Tienen una respuesta infinita en el tiempo debido a la retroalimentación.

Tienen una mayor eficiencia computacional, ya que pueden lograr la respuesta en frecuencia con un orden más bajo que los FIR, por lo que los hace idóneos para aplicaciones donde los recursos computacionales son limitados.

Estos filtros introducen distorsión en fase, por lo que dependiendo de la aplicación podría suponer un problema.

Son usados en sistemas de audio, ecualización y compresión de datos.

4.2.7. Motores Eléctricos.

Un motor es un dispositivo que convierte la energía eléctrica en mecánica y viceversa. Para su correcto funcionamiento requieres drivers o controladores para regular su velocidad, sentido y torque.

Los motores eléctricos pueden clasificarse en dos grandes apartados:

4.2.8. Motores de Corriente Continua (DC).

Se alimentan con corriente continua como su nombre indica, y a su vez estos se clasifican en tres grandes grupos:

4.2.8.1. Motores con escobillas (Brushed DC Motor).

Convierten la energía eléctrica en energía mecánica a través de la interacción de un campo magnético generado por el estator y un rotor con devanados eléctricos. Utilizan un conmutador mecánico y escobillas de carbón para invertir la polaridad del rotor de manera continua, asegurando así un movimiento rotativo constante.

Cuando se aplica voltaje a las escobillas, la corriente fluye a través de los devanados del rotor, generando un campo magnético que interactúa con el campo del estator. Esta interacción produce un par motor que hace girar el rotor. A medida que el rotor gira, el conmutador mecánico cambia la dirección de la corriente en los devanados,

asegurando que el campo magnético siga impulsando el movimiento en la misma dirección.

Se puede regular la velocidad y el par regulando la tensión de alimentación o usando señales de modulación por pulso (PWM), son motores económicos y fácil de implementar. Pero tienen un desgaste continuo entre el conmutador y las escobillas lo que requiere un mayor mantenimiento.

4.2.8.2. Motores DC sin escobillas (Brusheless DC Motor).

Los motores sin escobillas utilizan un controlador electrónico para conmutar las bobinas del estator en lugar de emplear un conmutador mecánico.

Estos motores poseen un rotor con imanes permanentes y un estator con bobinas. Un sistema de sensores (como sensores Hall) detecta la posición del rotor y un controlador conmuta las bobinas del estator en la secuencia correcta, generando un campo magnético giratorio que hace rotar el rotor.

Son mas eficientes que los Brushed ya que no tienen escobillas ni fricción, no hay desgaste mecánico por lo que requieren menos mantenimiento, y se pueden controlar con mayor precisión, pero requieren un Driver más complejo y no pueden operar sin este.

4.2.8.3. Motores paso a paso (Stepper Motor).

Los motores paso a paso son un tipo especial de motor DC diseñado para moverse en pasos. Se controlan mediante pulsos eléctricos enviados a las bobinas del estator, lo que permite una gran precisión en el posicionamiento.

Un motor paso a paso tiene múltiples bobinas distribuidas en fases. Al inducir corriente en un orden específico, el rotor se mueve en pequeños pasos angulares. Existen diferentes tipos:

4.2.8.3.1. Motores de reluctancia variable:

Este tipo de motor no usa imanes en el rotor. En su lugar, el rotor está compuesto por dientes de hierro que se alinean con los polos magnéticos del estator a medida que las bobinas se inducen en una secuencia determinada.

4.2.8.3.2. Motores de imán permanente:

Estos motores utilizan un rotor con imanes permanentes y un estator con bobinas que se activan en una secuencia específica para mover el rotor en pasos

4.2.8.3.3. Motores híbridos:

Combinan características de los motores de reluctancia variable y de imán permanente, ofreciendo mayor precisión y torque. Cuentan con un rotor dentado con imanes permanentes y un estator diseñado para mejorar la alineación magnética.

4.2.9. Motores de corriente Alterna.

Los motores de AC se alimentan con corriente alterna y son ampliamente usados en aplicaciones industriales.

4.2.9.1. Motores de inducción

Estos motores funcionan mediante la inducción electromagnética en el rotor. El estator genera un campo magnético rotatorio que induce una corriente en el rotor, generando un campo opuesto que produce el movimiento.

- Jaula de ardilla: Simples y robustos.
- Rotor bobinado: Permiten mayor control del par.

4.2.9.2. Motores sincrónicos

El rotor gira a la misma velocidad que el campo magnético del estator, lo que permite una operación estable en aplicaciones de velocidad constante.

4.2.10. Driver o Controlador Motores DC.

Los drivers de motores son circuitos electrónicos diseñados para controlar el voltaje, corriente y frecuencia de los motores, haciendolos eficientes y protegiéndolos contra sobrecargas y otros problemas eléctricos. Estos pueden estar formados por transistores, MOSFET, o circuitos integrados.

Existen diferentes tipos de Drivers para los motores de DC:

- Drivers Lineales: Funcionan mediante transistores en la región activa pero no son nada eficientes ya que disipan mucha energía en forma de calor.

- Drivers con modulación de Ancho de Pulso (PWM): Permiten un control eficiente de la velocidad variando el ciclo de trabajo de una señal cuadrada.
- Puente en H: El puente en H es una configuración electrónica de conmutación compuesta por cuatro interruptores, que pueden ser transistores bipolares, MOSFETs o incluso relés y permite invertir la polaridad aplicada a los terminales del motor, lo que posibilita su rotación en ambos sentidos.

  Su principio de funcionamiento se basa en la activación de distintos pares de interruptores:

  - Movimiento en un sentido: Se activan los interruptores en diagonal (por ejemplo, Q1 y Q4), permitiendo el flujo de corriente en una dirección.
  - Movimiento en sentido contrario: Se activan los otros dos interruptores (Q2 y Q3), invirtiendo la dirección del flujo de corriente.
  - Frenado: Se activan ambos interruptores superiores o inferiores simultáneamente, generando un cortocircuito entre los terminales del motor y deteniéndolo rápidamente.
  - Paro sin freno: Se desactivan todos los interruptores, dejando el motor en estado de alta impedancia.

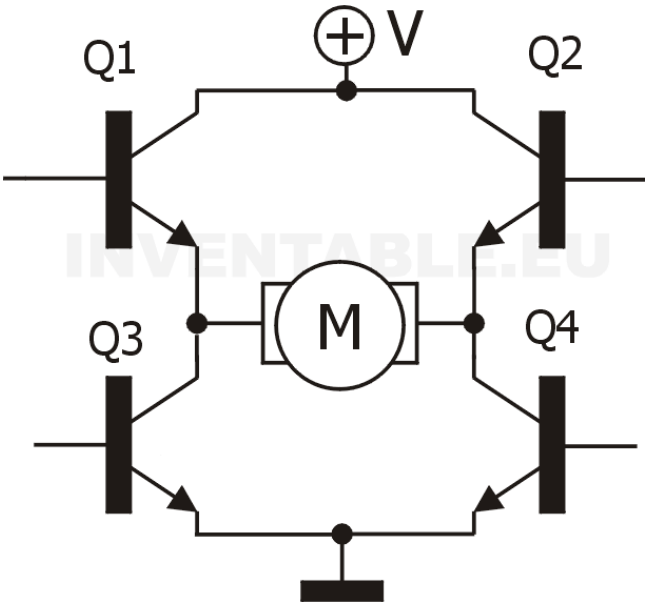


Ilustración 11 Puente en H

- Drivers de microstepping: Para accionar el motor paso a paso, existen varios modos de accionamiento, o modos de excitación:
  - "Modo de accionamiento por ondas, en este modo activamos sólo una bobina del estator a la vez, luego la

siguiente, y así sucesivamente. En este modo, sólo se activa una bobina, y para mover el rotor al siguiente paso, encendemos las bobinas una a una consecutivamente. Cuando encendemos la segunda bobina, se apaga la primera, etc.

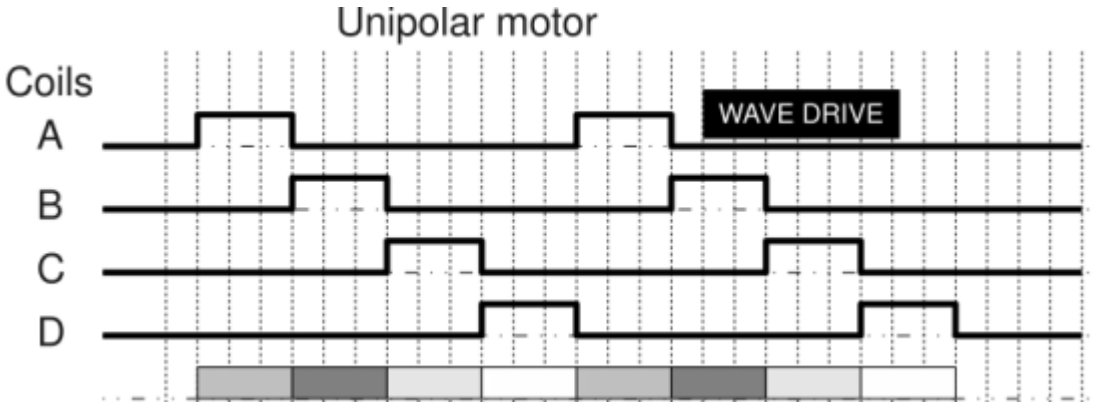


Ilustración 12 Accionamiento por ondas

- "El modo de accionamiento de paso completo proporciona una salida de par mucho mayor porque siempre tenemos dos bobinas activas en un momento dado.

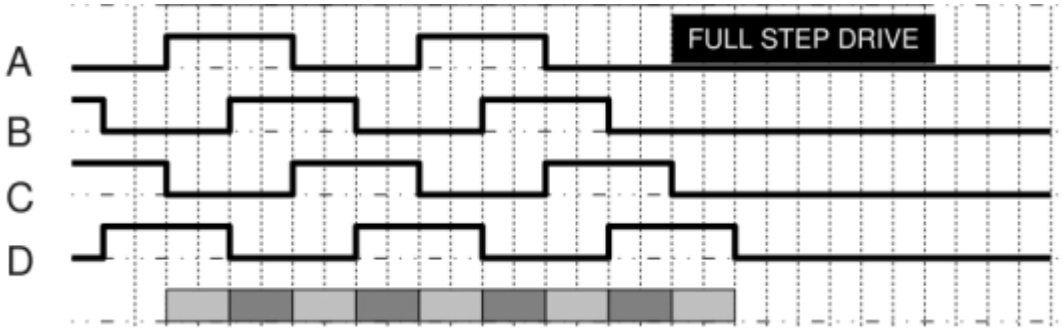


Ilustración 13 Accionamiento Paso completo

- "El modo de accionamiento de medio paso se utiliza para aumentar la resolución del motor paso a paso. Este modo surgió de la combinación de los dos modos anteriores. Aquí activamos una bobina, y cerca del final del estado activo de esa bobina, activamos la siguiente bobina. Cuando se activa la segunda bobina apagamos la primera, y así sucesivamente. Con este modo conseguimos el doble de resolución con la misma construcción de motor.

Ilustración 15 Accionamiento medio paso

- " El modo Microstepping es el método más común de controlar motores paso a paso hoy en día. En este modo proporcionamos corriente variable a las bobinas en forma de ondas sinusoidales. Esto proporcionará un movimiento suave del rotor, disminuirá la tensión de las piezas y aumentará la precisión del motor paso a paso.

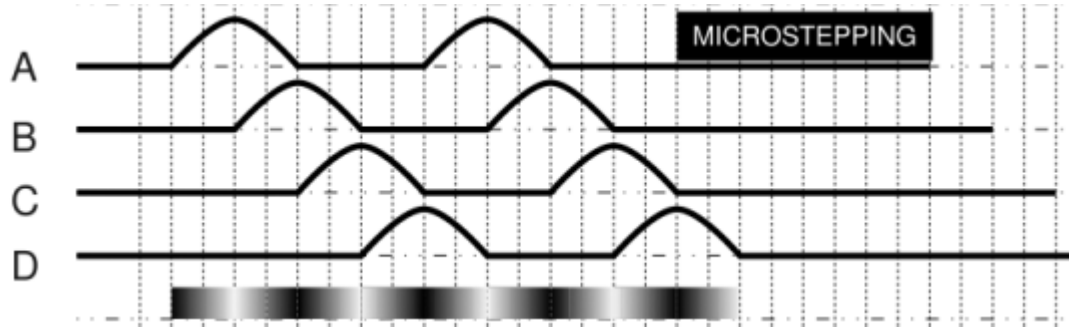


Ilustración 14 Accionamiento 1/4 de paso

4.3. Elección de Componentes.

Para el control de los motores se ha elegido Arduino Mega, el cual será programado en Visual Studio Code.

Los motores que deseamos controlar son dos motores paso a paso de cuatro pines de la marca TWO TREE los cuales son Nema 17 y soportan 1.5 A por fase soportando un par de 42N*cm^2.

Por otra parte el Driver del motor que se ha elegido para el trabajo es el A4988 el cual es un driver de microstepping con el siguiente PinOut:

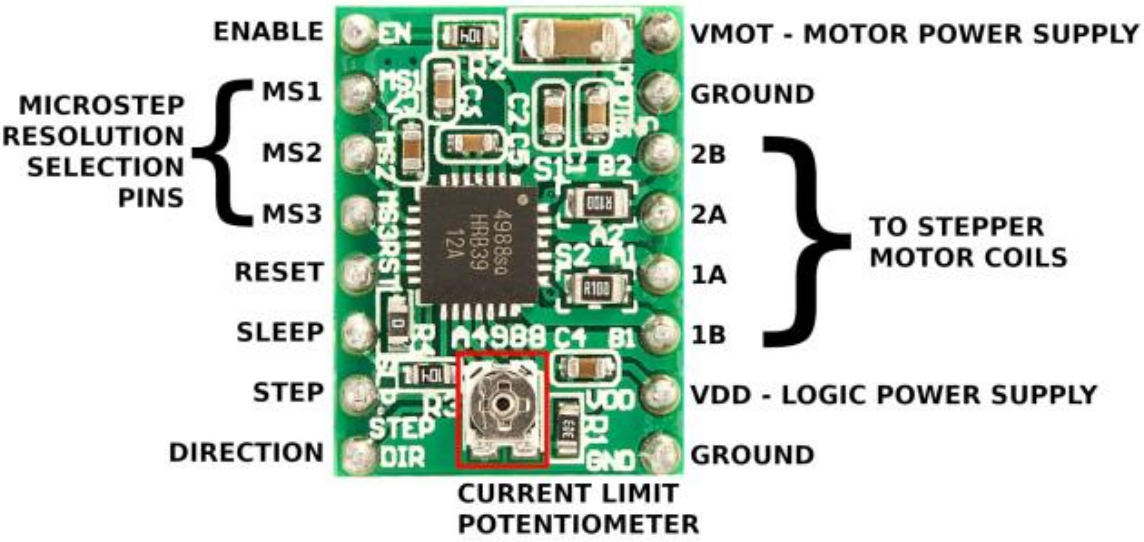


Ilustración 16 A4988 PinOut

El controlador lleva incorporado un traductor para facilitar su manejo. Esto reduce el número de pines de control a sólo dos, uno para controlar los pasos y otro para controlar la dirección de rotación. El excitador ofrece cinco resoluciones de paso diferentes, o modos de excitación: paso completo, y cuatro modos de micropaso: medio paso, cuarto de paso, octavo de paso y decimosexto de paso.

El A4988 requiere dos conexiones de alimentación. Una para los pines lógicos y otra para la alimentación del motor:

VDD y GND se utilizan para la conducción de la lógica interna del controlador, (de 3V a 5,5V), lo que significa que podemos utilizar Arduino como unidad de control para este dispositivo.

VMOT y GND se utilizan para alimentar el motor, de 8V a 35V. Según la hoja de datos, la alimentación del motor requiere un condensador de desacoplamiento adecuado cerca de la placa, capaz de soportar una corriente de 4ª, de 100µF (o al menos 47µF) a través de las patillas de alimentación del motor

El controlador A4988 tiene tres pines de entrada de selección de resolución de micropasos, MS1, MS2, y MS3.

Ajustando los niveles lógicos apropiados a estos pines podemos ajustar el modo de accionamiento del motor a uno de estos cinco modos:

EM1	MS2	MS3	Resolución de micropasos
BAJA	BAJA	BAJA	Paso completo
ALTO	BAJA	BAJA	Medio paso
BAJA	ALTO	BAJA	Cuarto de paso
ALTO	ALTO	BAJA	Octavo paso
ALTO	ALTO	ALTO	Decimosexto paso

Ilustración 17 Resolución de Pasos

Estos tres pines de selección de micropasos se ponen en LOW mediante resistencias pull-down internas, por lo que si los dejamos todos desconectados, el motor funcionará en modo paso completo.

Los modos de excitación de micropasos son todos los modos en los que el eje del motor se mueve entre los pasos de hardware. Estos modos posicionan el eje del motor entre los pasos, creando más pasos y un movimiento suave del eje.

El modo de excitación de medio paso es una combinación de paso completo y accionamiento de onda. El resultado es la mitad del ángulo de paso básico. Este ángulo de paso más pequeño proporciona un funcionamiento más suave debido a la mayor resolución del ángulo. El medio paso produce aproximadamente un 15% menos de par que el paso completo; sin embargo, el medio paso modificado elimina la disminución de par al aumentar la corriente aplicada al motor cuando se excita una sola bobina.

El micropaso puede dividir el paso básico de un motor hasta 256 veces, haciendo más pequeños los pasos pequeños. Un accionamiento de microdetección utiliza dos ondas sinusoidales de corriente variable separadas 90°, lo que resulta perfecto para permitir un funcionamiento suave del motor. Se observará que el motor funciona de forma silenciosa y sin que se detecte ninguna acción de paso a paso.Al controlar la dirección y la amplitud del flujo de corriente en cada bobina del estator, aumenta la resolución y mejoran las características del motor, lo que se traduce en menos vibraciones y un funcionamiento más suave. Como las ondas sinusoidales trabajan juntas, se produce una transición suave de una bobina a otra. Cuando aumenta la corriente en una bobina,

disminuye la corriente en la bobina siguiente, lo que se traduce en una progresión suave de los pasos y una salida de par mantenida.

El pin de entrada STEP controla los pasos del motor. Cada pulso ALTO enviado a este pin, hace girar el motor en el número de micropasos establecido por los pines de selección de micropasos. Cuanto más rápidos sean los pulsos, más rápido girará el motor.

El pin de entrada DIR controla la dirección de rotación del eje del motor. Si lo conectas en ALTO, el eje girará en el sentido de las agujas del reloj, y si lo conectas en LOW, el eje girará en sentido contrario. Si sólo desea que el eje gire en una sola dirección, puede conectar el pin DIR directamente a VCC o GND.

EN pin, cuando esta en LOW el controlador está habilitado. Por defecto esta patilla se pone BAJA para que el driver esté siempre habilitado, a menos que se ponga explícitamente ALTA para deshabilitar el driver.

SLP pin, conectando este pin a LOW pone el controlador en modo de reposo, minimizando el consumo de energía.

El pin RST, cuando esta a LOW, todos los pines STEP son ignorados, hasta que cambia a ALTO.

4.4. MÉTODO OPERATIVO:

En el desarrollo del Trabajo de Fin de Grado (TFG), se ha implementado un sistema basado en motores y láseres, cuyo correcto funcionamiento requiere una gestión de señales y potencia. Para comprender mejor la operación del sistema, se ha diseñado un diagrama de bloques que representa las etapas clave del proceso.

Ilustración 18 Diagrama Etapas Clave

En el diagrama podemos observar como desde LabVIEW reproducimos una pista de audio, de la cual analizándola se obtienen las frecuencias dominantes que serán enviadas al microcontrolador.

El microcontrolador, será el encargado de recibir las frecuencias enviadas por el LabVIEW y será el encargado del control de los motores, mediante el controlador correspondiente.

Los motores son los encargados de mover los espejos que gracias a un mecanismo diseñado son capaces de mover los espejos a las posiciones deseadas, y así mediante la proyección del láser sobre estos, se conseguirá mover el haz del láser a la posición deseada pudiendo generar así las figuras de Lissajous.

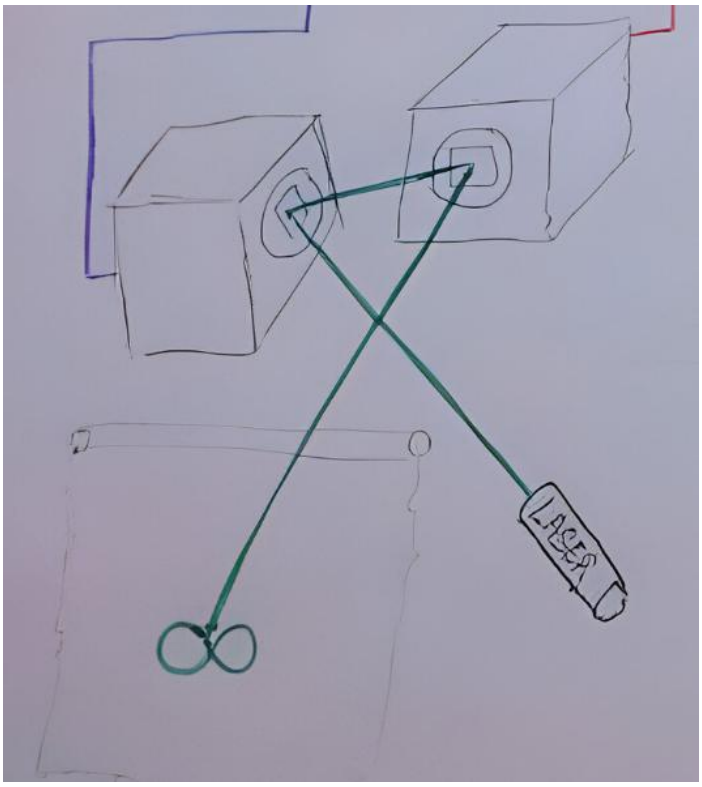


Ilustración 19 Boceto Funcionamiento.

Como podemos observar hay tres grandes bloques, uno sería el de adquisición de audio y tratamiento de la señal para obtener las frecuencias dominantes (LabVIEW), otro sería el control de los motores en tiempo real mediante el microcontrolador (Arduino Mega) y el ultimo seria el diseño del mecanismo para mover los espejos a la posición deseada (Diseño 3D).

4.4.1. Diseño Gráfico.

Como hemos explicado anteriormente para poder representar las figuras de lissajous necesitamos la interseccion de dos movimientos armónicos simples.

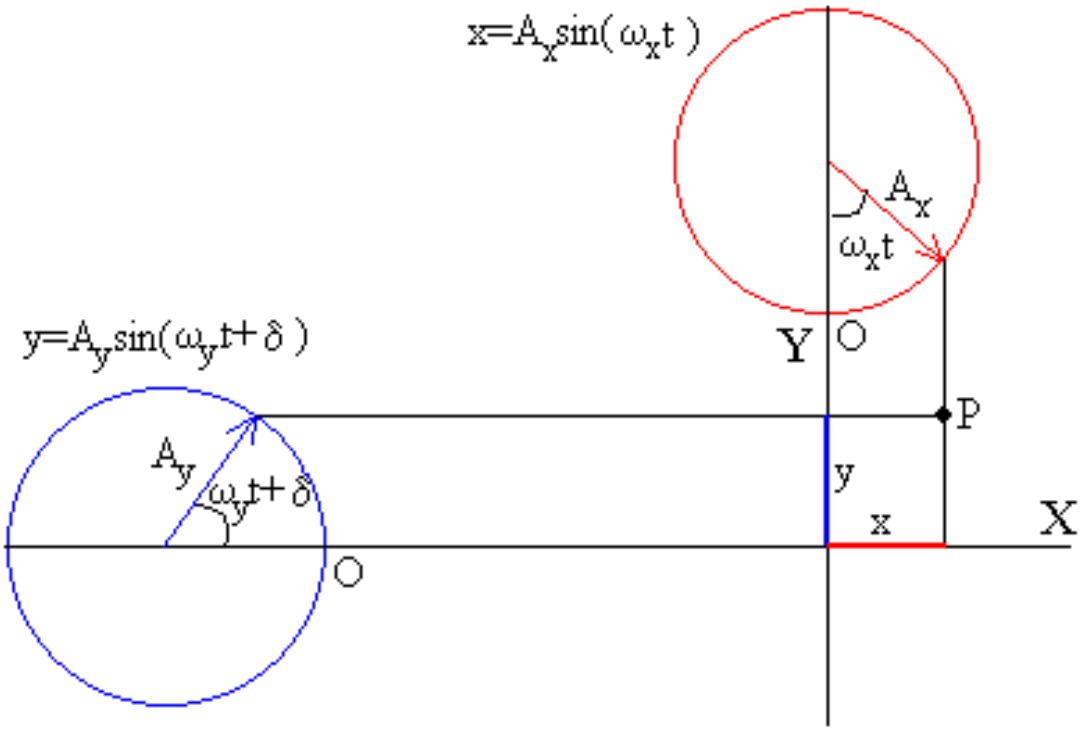


Ilustración 20 Representacion de un punto de las Figuras de Lissajous

Como podemos observar cada movimiento armónico representa el movimiento en cada eje del punto que queremos representar, por lo que deberemos representar dichos movimientos en planos perpendiculares para así tener una proyección ortogonal.

Para resolver el movimiento recurriremos al clásico movimiento Biela-manivela, pero con una modificaciones adaptándolo al proyecto.

Dado que una limitación que tenemos al usar motores paso a paso es la velocidad, deberemos ampliar el número de veces que se realiza el periodo del movimiento armónico por vuelta. Para ello se ha diseñado un oscilador para que en cada vuelta se realicen cuatro ciclos del movimiento.

En el orificio central le conectaremos con la ayuda de calor un piñón para poder unirlo al motor con la ayuda de un tornillo Allen.

Ilustración 22 Piñon unión motor

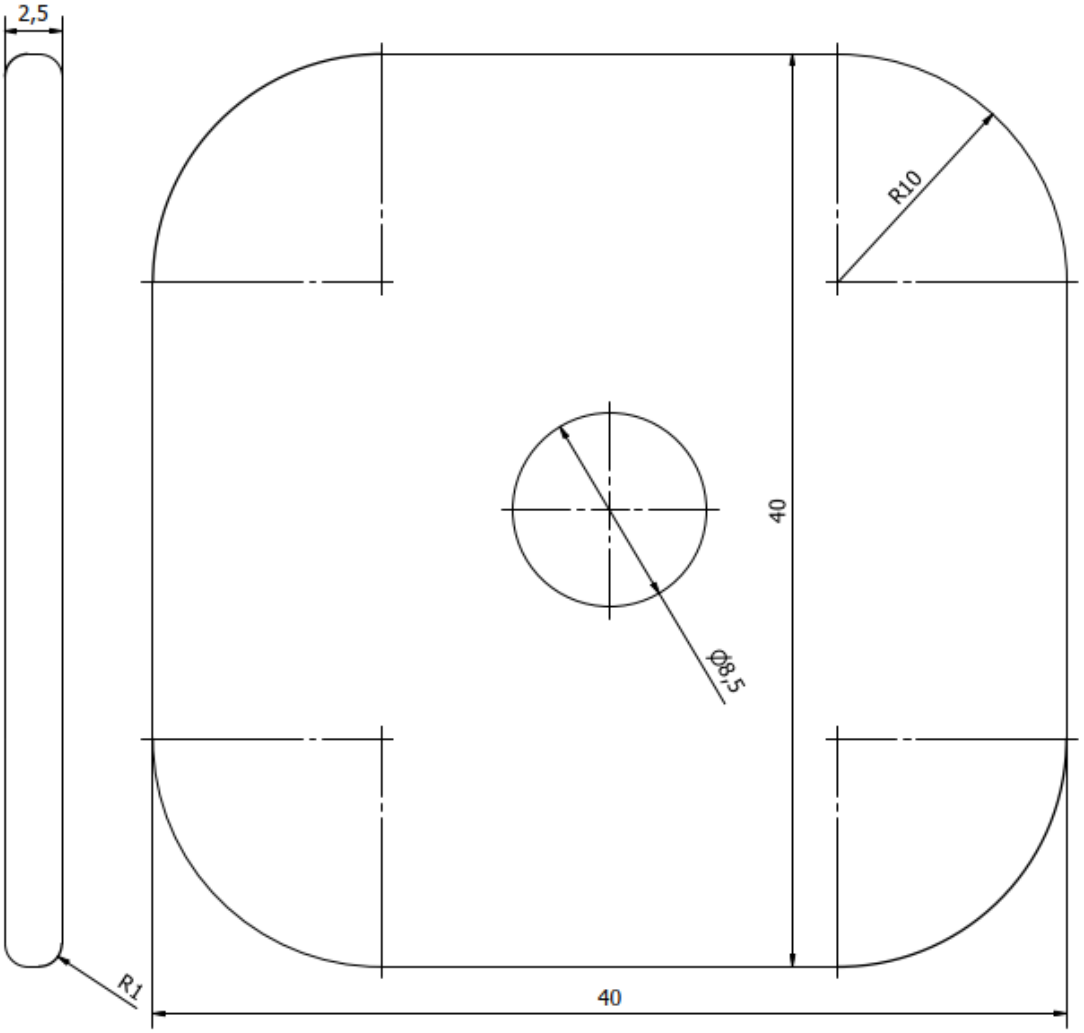


Ilustración 21 Oscilador

Quedando de la siguiente forma:

Ilustración 23 Piñon y Oscilador unidos

Y conectándolo con el motor:

Ilustración 24 Motor y Oscilador

Por otra parte se ha diseñado la base de uno de los ejes de movimiento en el que ira conectado el motor con el oscilador, y a su vez el soporte del espejo que tendrá que realizar el movimiento positivo y negativo de X o Y.

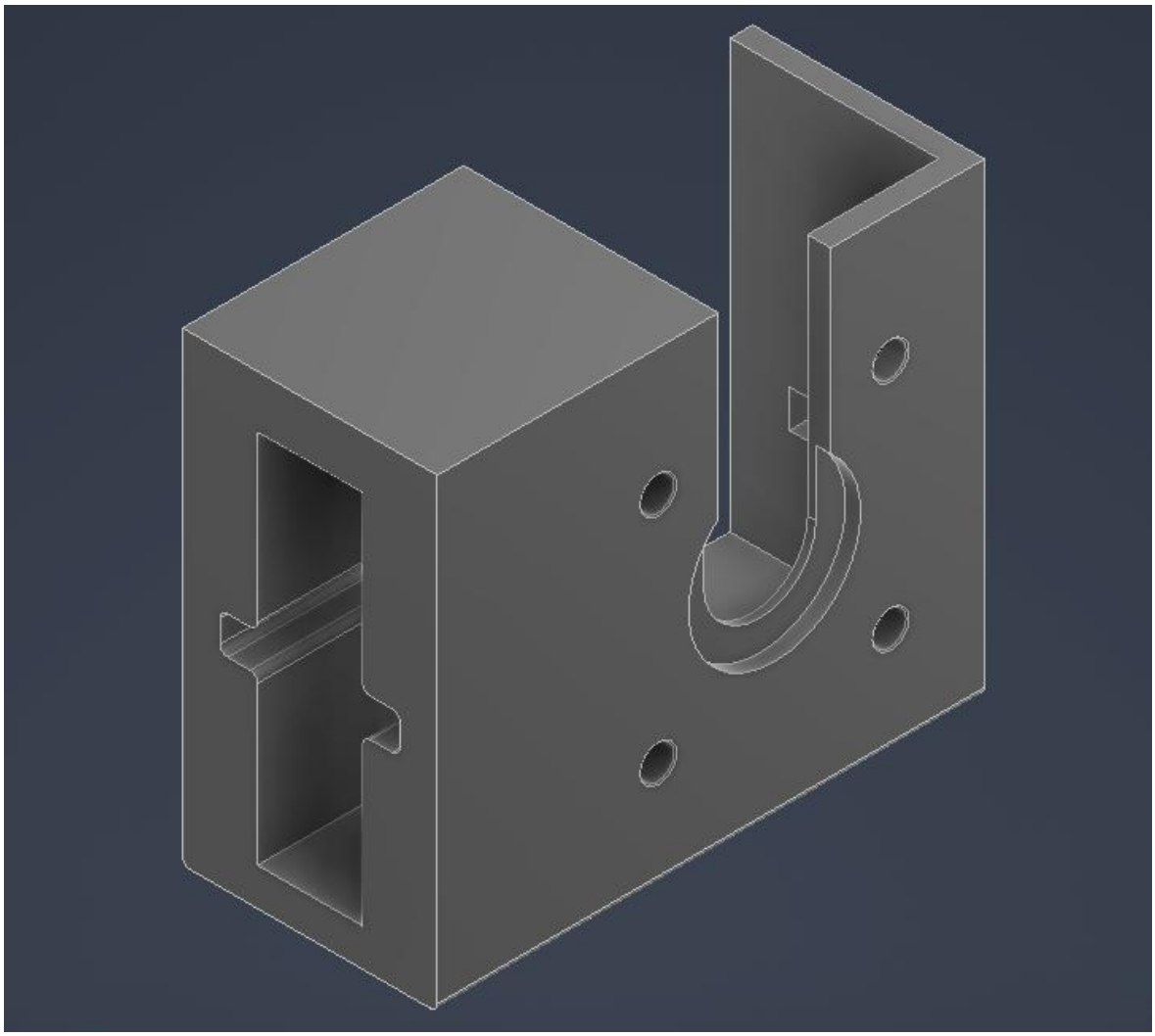


Ilustración 26 Base

La base esta diseñada para alojar en el lateral derecho el motor con sus cuatro agujeros para los tornillos y que posteriormente se le añada el oscilador.

En la parte delantera hay dos ranuras para alojar el soporte del espejo que mediante una goma en la parte trasera de este se realizara el retorno del soporte, quedando de la siguiente forma conectada:

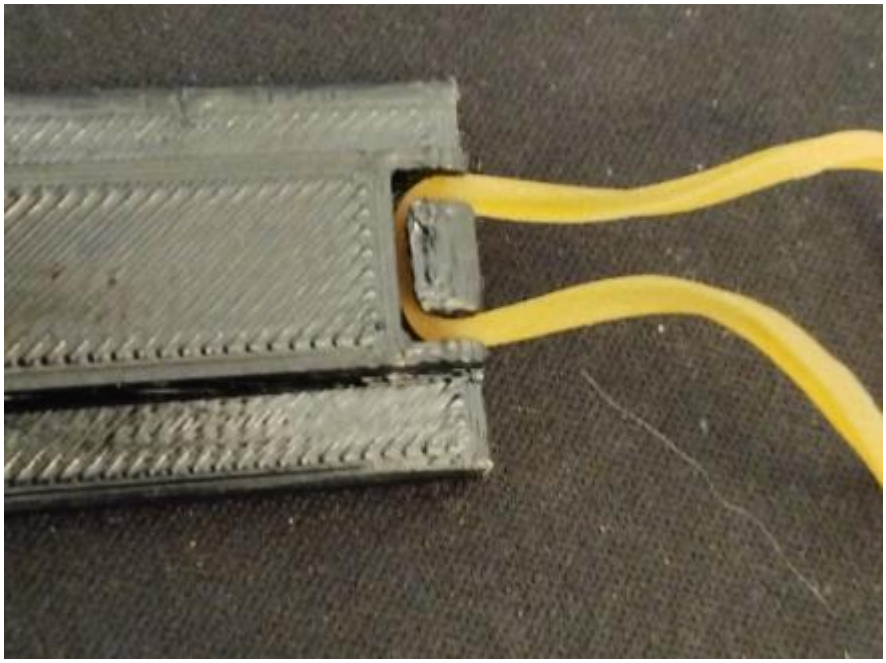


Ilustración 27 Retorno del Soporte del Espejo.

Que uniendo todo el conjunto quedaria de la siguiente forma:

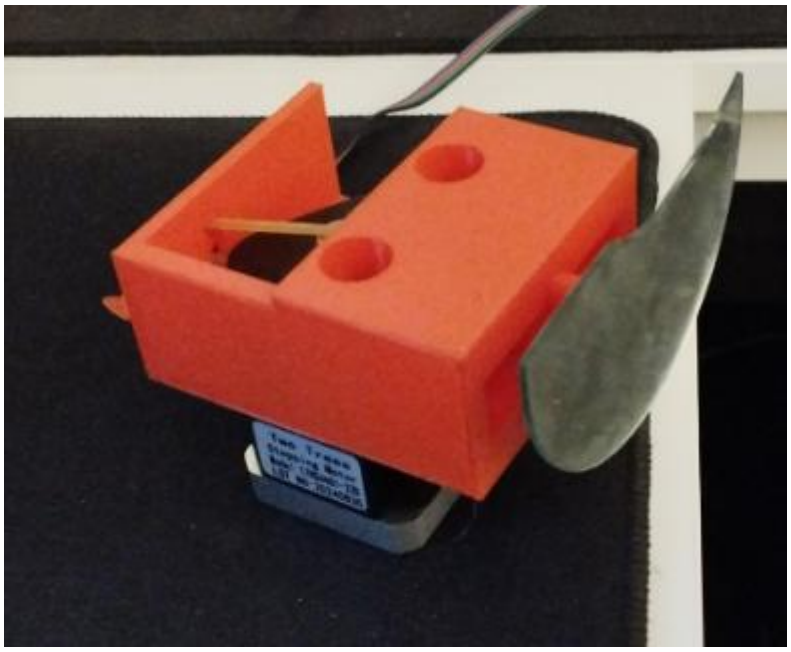


Ilustración 28 Movimiento espejo mediante motor

De esta forma ya tendriamos el movimiento de un eje montado, ahora solo tendriamos que montar otro sistema exactamente igual y colocarlo en un plano ortogonal al primero, para asi generar el eje X e Y.

Quedando de la siguiente forma:

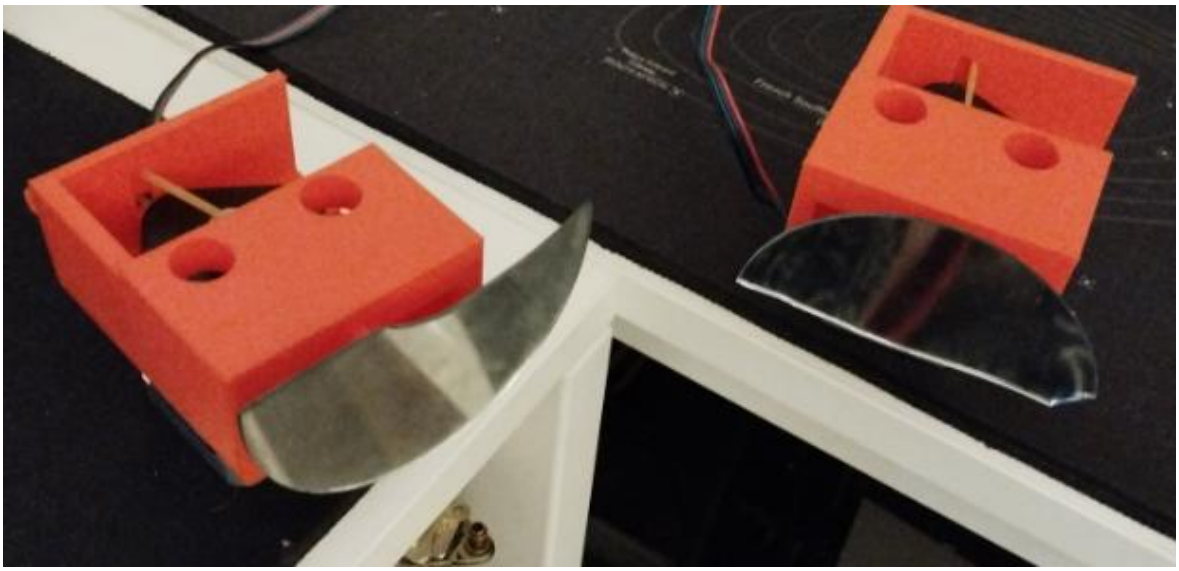


Ilustración 29 Sistema con los dos motores

Para la construcción de estos bloques se ha optado por la impresión 3D. Por lo que una vez diseñadas las piezas se han exportado en formato .stl para poder abrirlos desde el laminador.

El laminador que se ha usado es PrusaSlicer que permite una amplia configuración de la impresión y la opción de generar soportes para zonas donde el material estaría suspendido en el aire perjudicando así las medidas de como por ejemplo los orificios donde se aloja el soporte del espejo, y la impresora que se ha utilizado es una CR-10 con una base de impresión de 30cmX30cm y una altura de 40cm.

Una vez introducidos los objetos en el laminador:

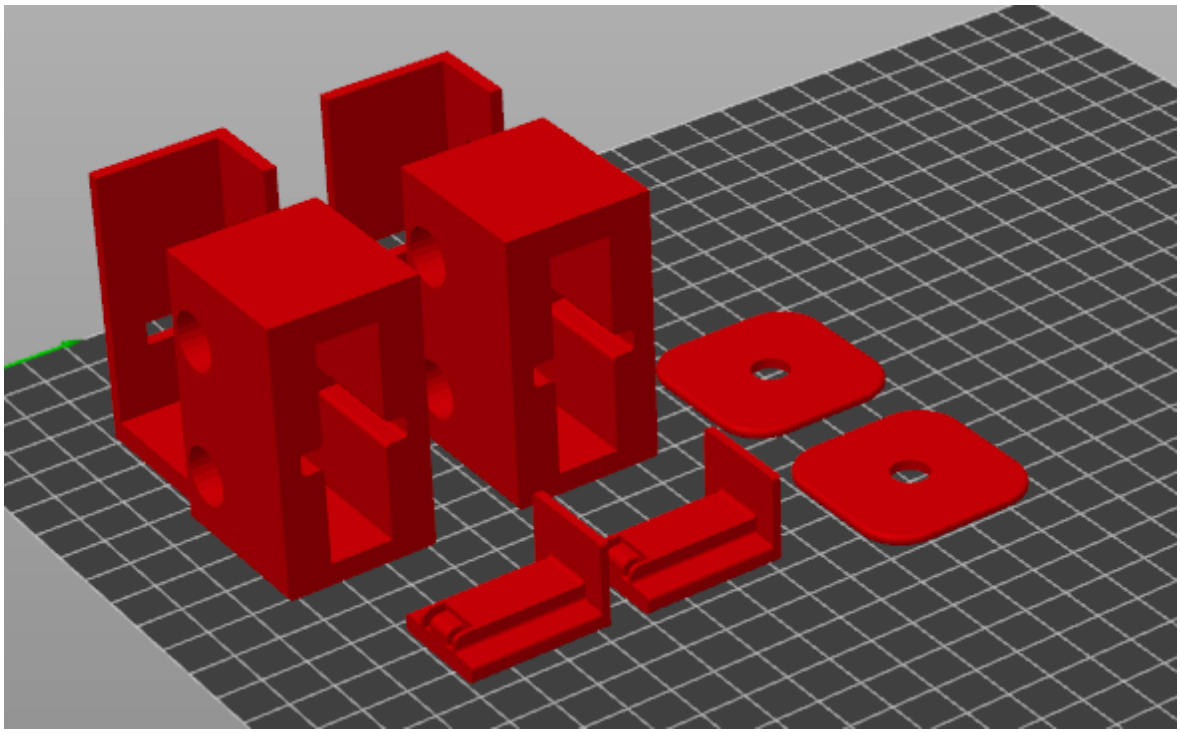


Ilustración 30 Base de impresión

Para dibujar los soportes elegiremos donde queremos que estén mediante la ayuda de la herramienta para dibujar los soportes.

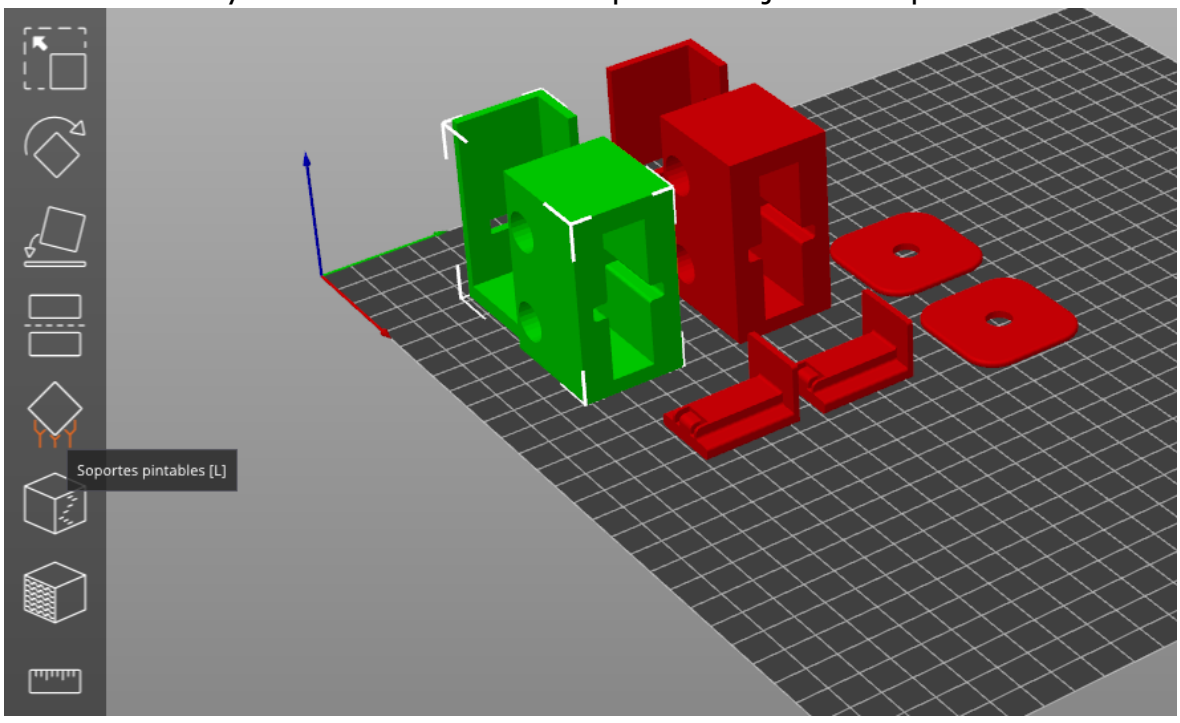


Ilustración 31 Seleccion de soportes

Y seleccionaremos las zonas donde queramos que se generen los soportes, quedando al final de la siguiente forma:

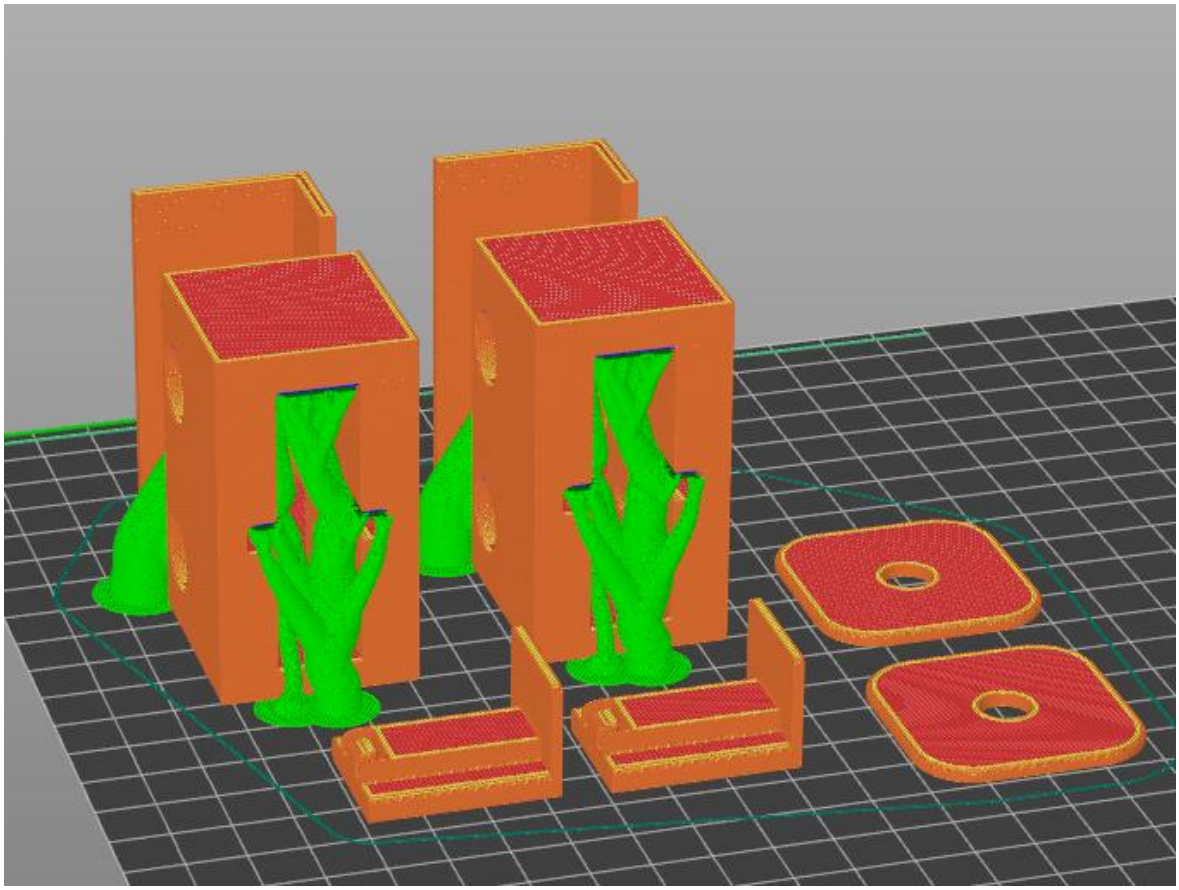


Ilustración 33 Base de Impresión con soportes.

Como podemos observar en la siguiente imagen se han generado soportes en todos los voladizos permitiendo que el material se apoye haciendo que se puedan respetar las tolerancias establecidas.

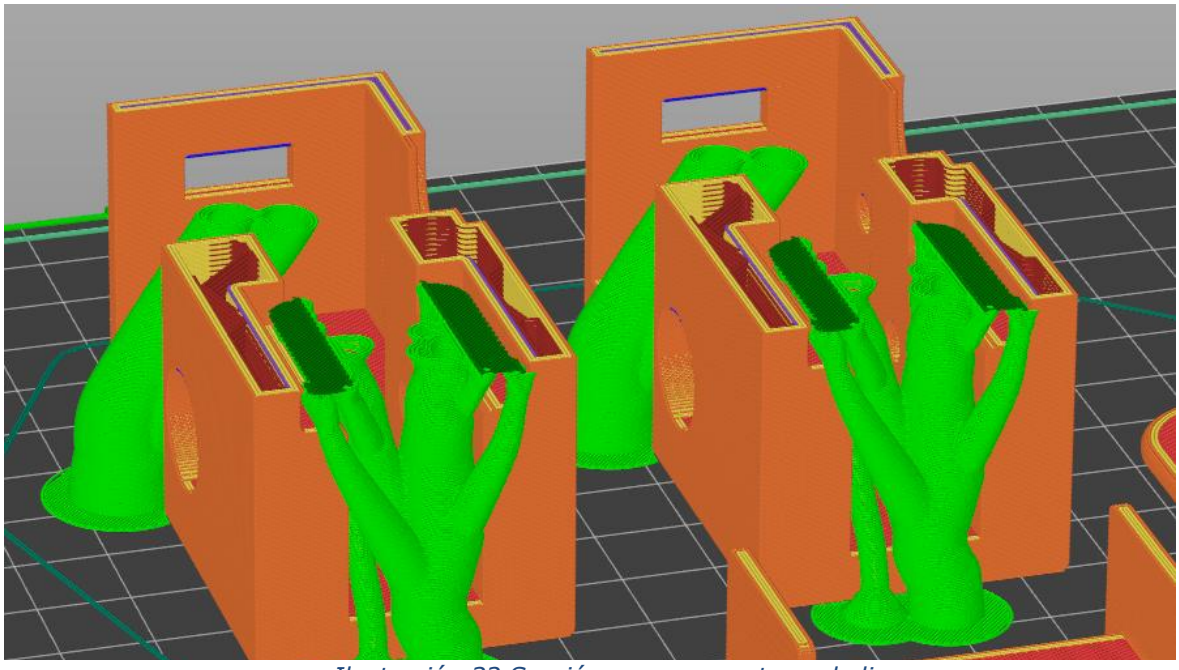


Ilustración 32 Sección apoyo soportes voladizos.

Por otra parte el material usado es PETG que es un plástico mucho mas resistente que el PLA, ya que el oscilador va a estar en constante rozamiento con el soporte del espejo.

4.4.2. Segundo Bloque: LabVIEW.

Para realizar todo el tratamiento de la señal, la adquisición de audio y la elección de la información que nos es útil para mover los motores la hemos realizado con LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un entorno de desarrollo gráfico creado por National Instruments que se utiliza principalmente para adquisición de datos, control de dispositivos electrónicos y automatización de pruebas.

4.4.2.1. Adquisición de Audio y Tratamiento de la señal.

Para adquirir el audio usaremos un bloque en LabVIEW el cual se llama Sound File Read Simple.VI que mediante un path le indicamos la ruta de la pista de audio que queremos analizar (LabVIEW solo trabaja con archivos .wav por lo que habría que convertir cualquier archivo a este formato antes), este bloque nos devuelve una matriz con un elemento tipo señal, el cual, sacaremos de la matriz para conectarlo con otro modulo que nos reproduce la pista que hemos seleccionado. A parte esa señal la conectaremos a dos módulos para obtener la duración de la pista y el número de muestras que hemos obtenido para saber la frecuencia de muestreo que hemos usado, y ser enviado a otro bloque que nos separará la canción en segmentos o Samples para hacer un análisis con menos cantidad de datos, poder optimizar los recursos y realizar el análisis más rápido.

Una vez Sampleada la canción procederemos a normalizarla para trabajar al mismo nivel de ganancia, poder realizar un análisis espectral, y poder poner puertas para detectar picos y llevar el análisis en tiempo.

Una vez normalizada filtraremos la señal para descartar ruidos y frecuencias elevadas, obteniendo una señal más limpia, para más tarde obtener la envolvente mediante la discretización de Hilbert, pero aun esta envolvente tiene mucho ruido y para suavizarla aplicaremos un filtro EMA para dejar una señal más plana y poder llevarla al bloque de detector de picos pudiendo obtener los beats por minuto (BPM) y el tiempo exacto entre la transición de cada pegada, por lo que aquí tenemos la información para sincronizar los motores y el cambio de las figuras con la música.

Por otra parte, se han graficado varias partes de las señales para comprobar que tanto la normalización, como los filtros y la discretización

se han realizado de manera correcta, quedando en el panel frontal con las siguientes graficas: "sample", "sample normalizado"," la envolvente de la normalización", y "la envolvente suavizada", por otra parte, también enviaremos toda la información del sample con su espectro amplitud, mediante una tarea al siguiente bloque.

Quedando así el primer bloque:

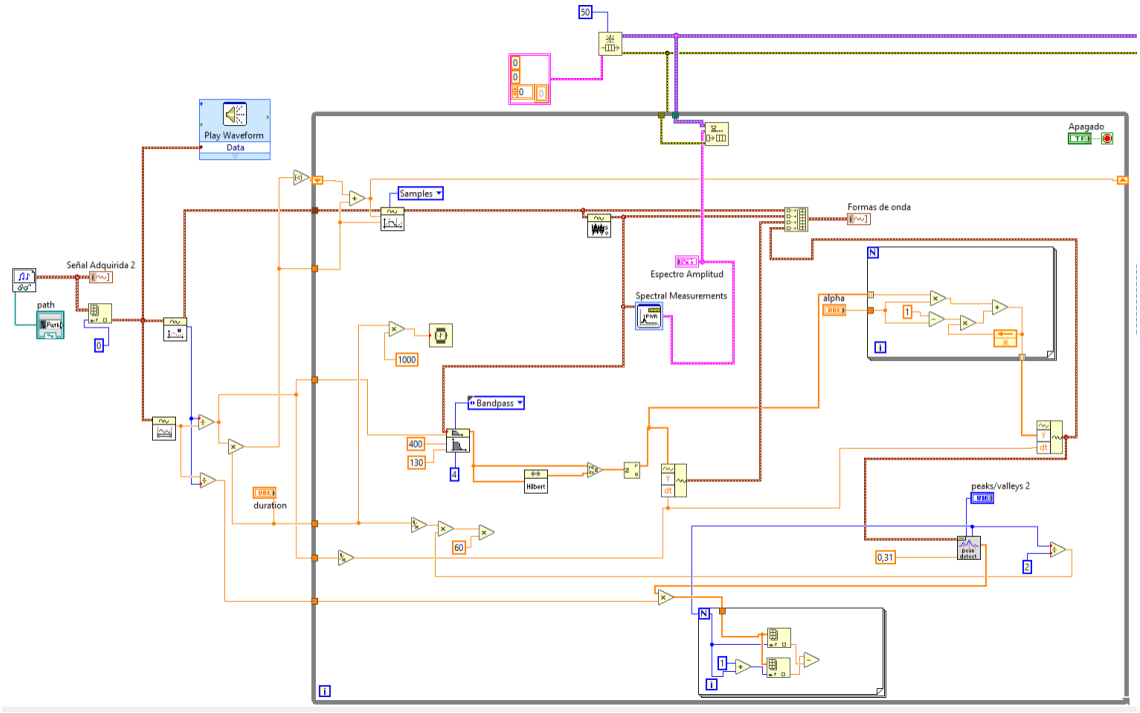


Ilustración 34 Adquisición de Audio y Tratamiento de la Señal

4.4.2.2. Elección de la Información.

Una vez enviada la información del espectro a la tarea, esta llegara al bloque para ser descompuesta en dos rangos, alta y baja frecuencia, guardándose todos los valores en una matriz. Dicho rango de frecuencias lo elegiremos nosotros y después analizaremos la matriz, que obteniendo los valores del espectro podremos ver cual tiene más energía, y obtener la frecuencia dominante de cada rango.

Una vez creada la matriz con la frecuencia dominante de cada rango, procederemos a filtrarla con un filtro EMA ya que la transición de la frecuencia dominante de un sample a otro puede ser muy escalonada produciendo interrupciones en el movimiento de la figura, por lo que al usar el filtro EMA esa transición está mucho más suavizada.

Una vez que tengamos las frecuencias dominantes de cada rango procederemos a meter en una matriz las frecuencias dominantes filtradas y sin filtrar para poder comprobar la diferencia entre una selección de datos u otra.

Con la matriz creada procederemos a enviarla con una tarea al siguiente bloque, quedando configurado este apartado de la siguiente forma:

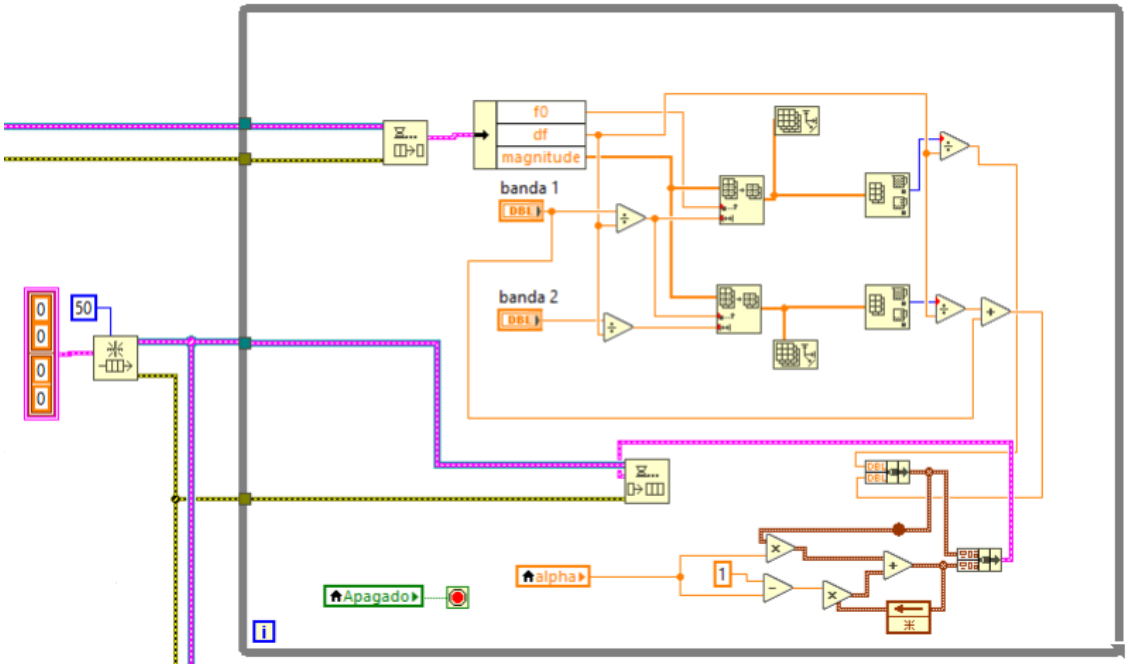


Ilustración 35 Elección de la Información

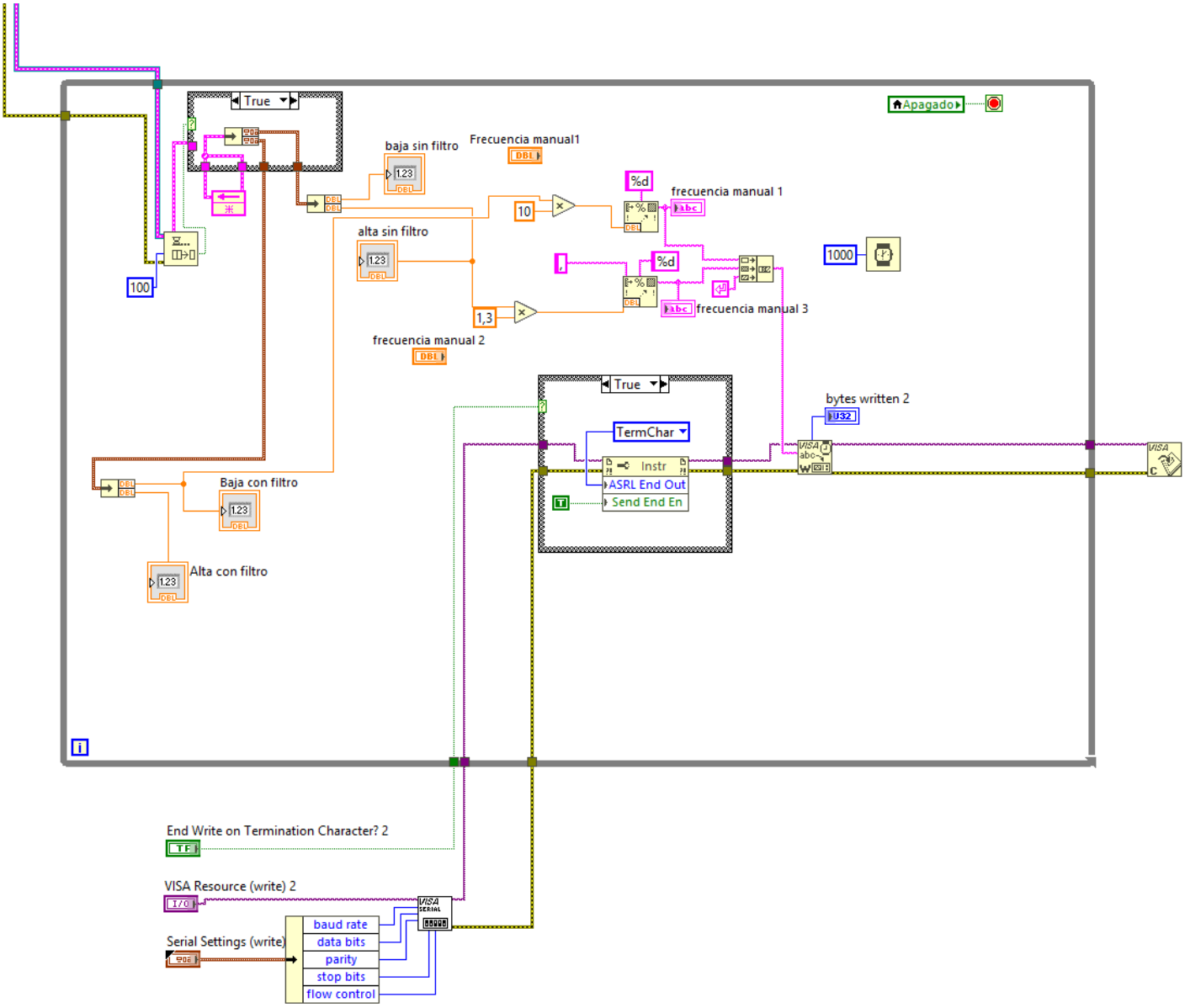
4.4.2.3. Envío de Datos al Microcontrolador.

Una vez recibida la información en forma de tarea nos dispondremos a separar la matriz para obtener la frecuencias dominates, tanto las que estamos filtrando como las que no.

Una vez tengamos los valores los convertiremos a string ya que LabVIEW solo puede enviar datos string por el puerto serie, y después concatenaremos los string, separados por una "," y terminando la concatenación con \n\r.

Con el dato ya preparado nos dispondremos a mandárselo al bloque de VISA que realiza la comunicación por la UART con el microcontrolador. Y le colocaremos un delay de 1000 ms para enviar los datos cada segundo y evitar problemas de concurrencia con Arduino.

Por otra parte se han dejado dos selectores de valor por si se desea seleccionar la velocidad manualmente simplemente seria desconectar la matriz de entrada al conversor de decimal a string y conectar el selector.


True
Apagado
baja sin filtro
Frecuencia manual1
10
%d
frecuencia manual 1
alta sin filtro
1000
100
1,3
frecuencia manual 3
frecuencia manual 2
True
bytes written 2
TermChar
Instr
ASRL End Out
Send End En
Baja con filtro
Alta con filtro
End Write on Termination Character? 2
VISA Resource (write) 2
Serial Settings (write)
baud rate
data bits
parity
stop bits
flow control

4.4.3. Segundo Bloque: Microcontrolador.

En este bloque se explicará el control de los motores tanto con sus drivers como la conexión con el microcontrolador, como el estudio eléctrico del sistema:

4.4.3.1. Estudio eléctrico:

Este sistema está compuesto por los siguientes elementos eléctricos:

- Arduino Mega: Controlador principal que gestiona la comunicación con los drivers y coordina el movimiento de los motores.
- Drivers A4988: Son los controladores de los motores paso a paso NEMA 17. Permiten controlar la corriente que pasa por cada fase del motor para hacer que el motor gire con precisión.
- Motores NEMA 17: Motores paso a paso que permiten el control preciso del movimiento. Cada motor tiene una corriente nominal de 1.5A por fase.
- Fuente de alimentación: Una fuente de 20V y 4.5A, que suministra energía al sistema.
- Cables Dupont: Cables finos utilizados para las conexiones entre los componentes, típicos en protoboards.

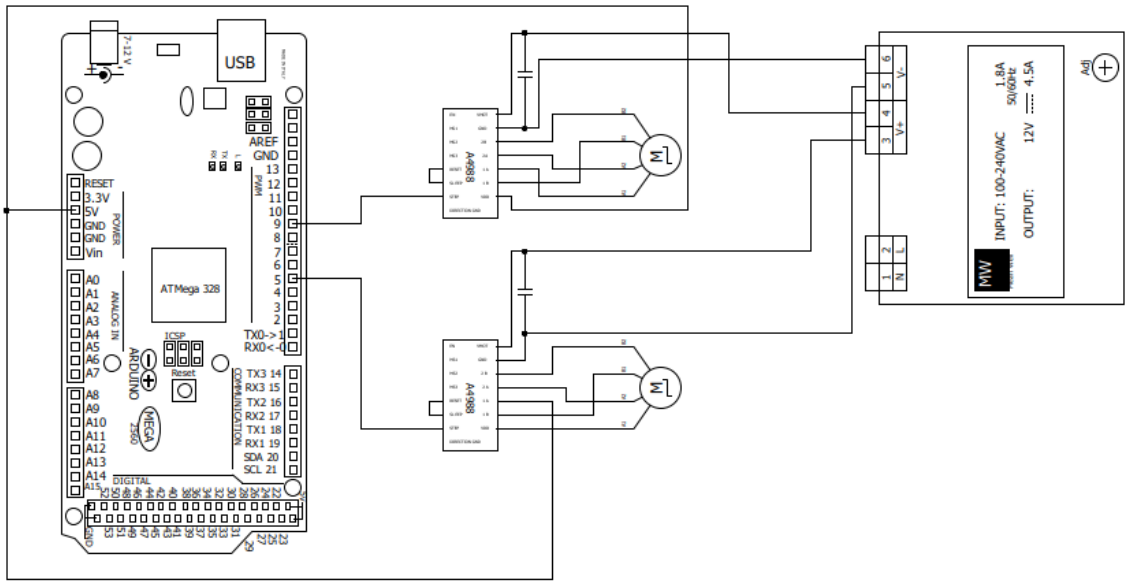


Ilustración 36 Esquema Eléctrico, Microcontrolador, Drivers y Motores.

Los motores NEMA 17 están diseñados para consumir 1.5A por fase, lo que significa que cada motor puede consumir hasta 3A, ya que cada

motor tiene dos fases. Como el sistema tiene dos motores, la corriente total consumida por los motores sería.

$$Imotor = 1{,}5A * 2fases = 3A$$

Ecuación 20 Corriente motor

Por lo que al tener dos motores la intensidad consumida por los motores seria de 6 A.

El Arduino mega por otra parte tendría un consumo de unos 100-250 mA de corriente.

Y según el datasheet del driver A4988 [1] el circuito lógico en modo operación con señales pwm de menos de 50KHz es de 8mA y la corriente consumida para alimentar a los motores en las mismas condiciones es de 4mA. Por lo que el consumo total del driver es de 12mA y al ser 2 serian 24mA.

Dispositivo	Motor	Microcontrolador	Driver
Cantidad	2	1	2
Consumo	6A	100-250mA	24mA
Total	6.3A		

Tabla 1 Consumos

Por otra parte la potencia Consumida en forma de disipación Térmica en el driver seria:

$$Pdisipada = I^2 \cdot Rds(on)$$

Ecuación 21 Potencia disipada transistor[2]

Como en cada fase del motor hay dos mosfets activos simultáneamente la potencia disipada seria el doble, y como son dos fases por motor:

$$Pdisipada = I^2 \cdot Rds(on) * 2 * 2 = I^2 \cdot Rds(on) * 4$$

Ecuación 22 Potencia disipada

El datasheet del A4988 proporciona los valores necesarios para calcular la disipación. Los valores relevantes son:

Rds(on) (resistencia en conducción): Típico 320 mΩ, Máximo 430 mΩ por MOSFET.

Temperatura máxima de unión (Tj max): 150°C.

Resistencia térmica (RθJA): 32 °C/W (sin disipador en PCB de 4 capas).

Protección por sobrecalentamiento: Activa a 165°C.

Por lo que calcularemos la potencia disipada por el driver y veremos si supera el valor de 1.5W que el fabricante recomienda no superar:

$$Pdisipada = 1.5^2 * 0.32 * 4 = 2.88W$$

Como podemos observar la potencia disipada es mayor a la recomendada por el fabricante, pero debemos tener en cuenta que este cálculo es para estar funcionando a máxima potencia constantemente, por lo que mediante el análisis de temperatura se pudo observar que el driver no superaba los 50º ya que esta al aire libre y la disipación con el medio es mayor que si estuviera en una caja. En ese caso deberíamos ponerle un disipador con ventilación para que disipe más temperatura con el medio.

Con esto tendríamos el consumo total del sistema, que serían 120W de los cuales 2.88, se irían en forma de calor, teniendo en cuenta que nuestra fuente puede entregar 90W tendríamos un problema con la potencia, pero al no estar a máxima potencia constantemente ese consumo desciende drásticamente permitiendo a la fuente de alimentación operar sin ningún problema sin que se active la protección de esta.

Para calcular la sección adecuada del cable, usaremos la fórmula de la ley de Ohm aplicada a la caída de tensión:

$$R = \rho \cdot \frac{A}{L}$$

Ecuación 23 Resistencia de un conductor

Donde:

- R es la resistencia del cable.
- ρ es la resistividad del material (para el cobre, ρ=1.68×10^{-8}Ω·m)
- L es la longitud del cable.
- A es el área de la sección transversal del cable (m^2).

Teniendo en cuenta que la caída máxima de tensión no exceda el 5%

$$Vcaida = I * R; 12V * 0.05 = I * R$$

Que despejando:

$$A = \frac{\rho \cdot L \cdot I}{Vcaida} =$$

$$= \frac{1.68 \times 10^{-8} \Omega \cdot m \times 1m \times 1.5\ A}{0.6\ V} = \ 4.2 \times 10^{-8} m^2 = 0.042 mm^2$$

Por lo que usando un cable dupont de 30 AWG, que es el que menos sección tiene (0.05mm^2), cumpliríamos con el Area exigida.

4.4.3.2. Control Motores.

Para que el microcontrolador pueda manejar los motores a las frecuencias seleccionadas desde LabVIEW deberemos conectar el microcontrolador con el driver y los motores según indica el siguiente esquema eléctrico:

Una vez conectado el circuito como en la imagen anterior nos dispondremos a escribir el código para Arduino Mega.

Ya que para el control del driver se necesita cambiar la frecuencia con la que una señal digital cambia de estado de HIGH a LOW, se ha decidido generar una señal cuadrada en la salida digital 9 y 5, una para cada driver.

Para generar esas señales cuadradas, se ha programado una interrupción en el timer1 y el timer3 en modo CTC, lo que significa que el temporizador cuenta desde 0 hasta un valor llamado OCRnA (Output Compare Register A), y cuando lo alcanza, reinicia la cuenta y genera una interrupción, en el que se cambia el estado del pin, creando el estado alto y bajo cada 2 interrupciones, y lo que se modifica es el tiempo que tarda esta interrupción en ejecutarse. Al estar creadas en dos timers diferentes la frecuencia de cada señal cuadrada es diferente, por lo que tenemos control sobre cada onda de forma independiente, pudiendo controlar así la velocidad de cada motor.

Por otra parte, para recibir las frecuencias desde LabVIEW se ha realizado mediante UART, por lo que como desde LabVIEW se envía un string de aproximadamente 10bytes de los cuales los 4 primeros bytes representan la frecuencia de movimiento del motor 1, el siguiente byte es una “,” usada para separar las dos frecuencias, los siguientes cuatro bytes es la segunda frecuencia y el ultimo carácter es el fin de cadena “\n”. por lo que separaremos el string recibido por la UART y lo convertiremos a entero para poder operar con el.

La frecuencia de la señal que sale por el pin está determinada por la ecuación:

$$Fsalida = \frac{Ftimer}{2 * (OCRnA + 1)}$$

Ecuación 24 Frecuencia de Salida.

$$OCRnA = \frac{Ftimer}{2 * Fsalida} - 1$$

Ecuación 25 Tiempo Interrupcción

Siendo Ftimer la frecuencia de trabajo del Arduino 16MHz que en este caso serian 2MHz porque usamos un preescaler de 8 quedando así

definido el tiempo de la interrupción, que será actualizado en cada interrupción.

El código main.cpp seria el siguiente:

```
#include <Arduino.h>
#include <avr/io.h>
#include <avr/interrupt.h>

// Función para leer frecuencias desde LabVIEW
void leerFrecuencias(volatile int* timeFreq1, volatile int* timeFreq2);

// Definición de pines para los motores
#define MOTOR1 9 // Pin de salida para motor 1 (Timer1)
#define MOTOR2 5 // Pin de salida para motor 2 (Timer3)
#define MicroStep 3

// Rango de frecuencias permitidas en Hz
const int freqMin = 150;    // Frecuencia mínima en Hz
const int freqMax = 2000;  // Frecuencia máxima en Hz

// Variables globales de frecuencia (se almacenan en microsegundos)
volatile int time1 = 500;
volatile int time2 = 500;
volatile bool state1 = false; // Estado de la señal para motor 1
volatile bool state2 = false; // Estado de la señal para motor 2

String str1, str2; // Variables para almacenar los datos recibidos por
Serial


void setup() {
    Serial.begin(115200);
    pinMode(MOTOR1, OUTPUT);
    pinMode(MOTOR2, OUTPUT);
    pinMode(MicroStep, OUTPUT);

    digitalWrite(MicroStep, HIGH); //Ponemos el driver en modo micropaso de
1/16

    cli(); // Desactivar interrupciones mientras configuramos

    // Configurar Timer1 (16 bits)
    TCCR1A = 0;  // Modo normal
    TCCR1B = (1 << WGM12) | (1 << CS11); // CTC, Prescaler de 8
    OCR1A = 500; // Valor inicial de comparación
    TIMSK1 |= (1 << OCIE1A); // TIMSK1 (Timer Interrupt Mask Register 1) y
OCIE1A (Output Compare Interrupt Enable 1A)
```

```
    // Configurar Timer3 (16 bits)
    TCCR3A = 0;
    TCCR3B = (1 << WGM32) | (1 << CS31); // CTC, Prescaler de 8
    OCR3A = 500;
    TIMSK3 |= (1 << OCIE3A); // Habilitar interrupción por comparación
OCIE3A

    sei(); // Habilitar interrupciones
}

// Interrupción Timer1 (Motor 1)
ISR(TIMER1_COMPA_vect) {
    state1 = !state1;
    digitalWrite(MOTOR1, state1);
    OCR1A = time1; // Actualizar intervalo
}

// Interrupción Timer3 (Motor 2)
ISR(TIMER3_COMPA_vect) {
    state2 = !state2;
    digitalWrite(MOTOR2, state2);
    OCR3A = time2;
}


void loop() {
    leerFrecuencias(&time1, &time2);
}

void leerFrecuencias(volatile int* timeFreq1, volatile int* timeFreq2) {
  if (Serial.available() > 0) {
        String input = Serial.readStringUntil('\n');  // Lee hasta encontrar
'\n'
        int separatorIndex = input.indexOf(',');      // Encuentra la coma
(delimitador)

        if (separatorIndex != -1) {
            str1 = input.substring(0, separatorIndex);   // Extrae la
primera frecuencia
            str2 = input.substring(separatorIndex + 1);  // Extrae la
segunda freciemcia
        }
      int f1 = str1.toInt();
      int f2 = str2.toInt();

      // Validar rangos y actualizar punteros
```

```
      if (f1 >= freqMin && f1 <= freqMax) {
        noInterrupts();
          *timeFreq1 = (2000000 / (2*f1)) - 1;
        interrupts();
      }
      if (f2 >= freqMin && f2 <= freqMax) {
        noInterrupts();
          *timeFreq2 = (2000000 / (2*f2)) - 1;
          interrupts();
      }
  }
}
```

Y para indicar que estamos trabajando a 115200 Baudios deberemos modificar el archivo platformio.ini debiendo quedar de la siguiente manera:

```
[env:megaatmega2560]
platform = atmelavr
board = megaatmega2560
framework = arduino
monitor_speed = 115200
```

De esta forma estaríamos mandando una señal cuadrada por el pin 5 y 9 del Arduino cuya frecuencia es la que nos manda LabVIEW por la UART, quedando así establecido el control de los motores a la frecuencia que mandamos desde LabVIEW después de hacer todo el análisis del audio.

5. RESULTADOS

Durante el desarrollo del proyecto, se realizaron diversas pruebas para evaluar el desempeño del sistema, enfocándose en la generación de la señal PWM, la implementación del código en diferentes plataformas y el movimiento de los motores. A continuación, se presentan los resultados obtenidos junto con un análisis de los problemas detectados y sus posibles soluciones.

5.1. PRUEBAS CON LABVIEW:

Se realizaron diversas pruebas en LabVIEW para analizar el comportamiento de la señal generada y su interacción con el sistema. Entre las pruebas destacadas, se incluyen:

Análisis del espectro de la señal: Se utilizó la Transformada de Fourier Rápida (FFT) para examinar la composición frecuencial de la pista de audio y poder obtener las frecuencias dominantes para sincronizarlas con los motores.

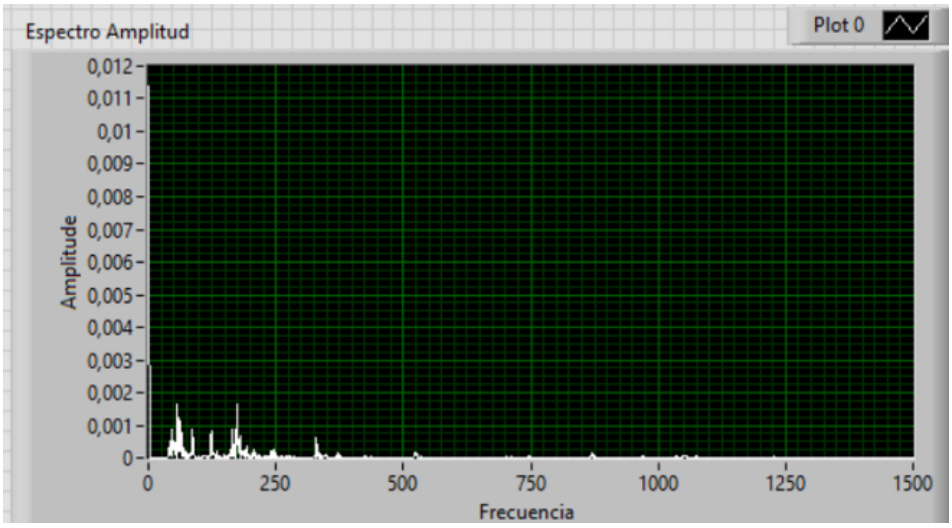


Ilustración 37 Analisis Espectral

Cálculo de la envolvente con el filtro de Hilbert:

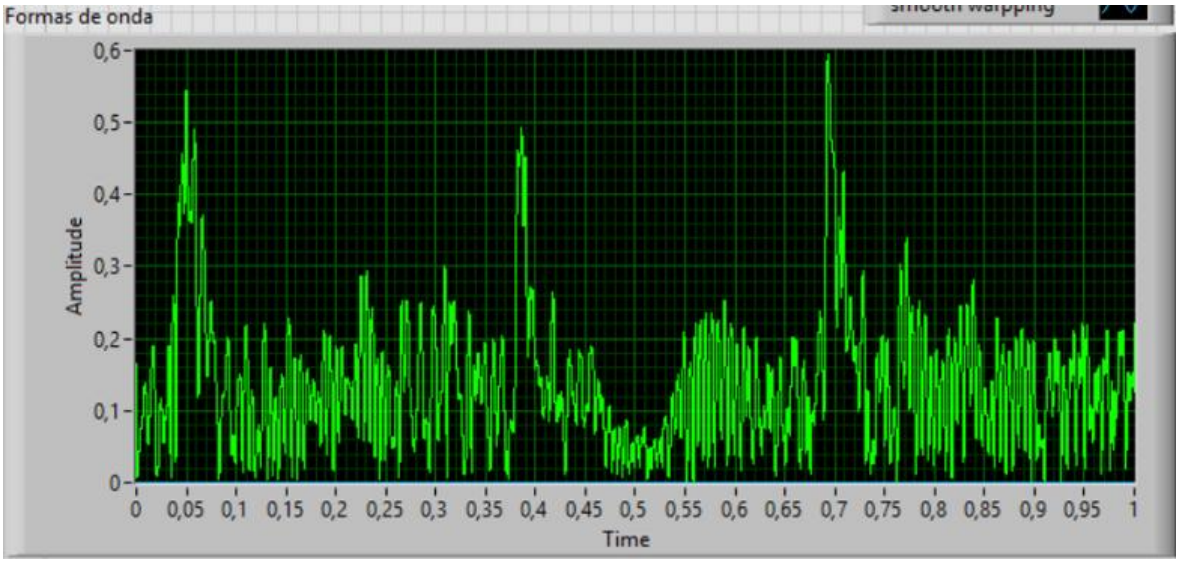


Ilustración 38 Envolvente de la señal

Se aplicó este método para analizar la variación de amplitud en la señal y detectar los BPM de la canción sincronizando el análisis con la pista de audio.

Envío de datos al microcontrolador: Se estableció una comunicación entre LabVIEW y el Arduino Mega para ajustar parámetros en tiempo real, optimizando el control del motor y la respuesta del sistema. Pero se ha observado que cuando el motor es controlado por las frecuencias obtenidas de la pista de audio, como muchas veces la transformacion equivale a velocidades altas, el motor no puede empezar sin una pequeña ayuda manual.

5.2. Pruebas para la generación de PWM.

Para comprobar el correcto funcionamiento de los motores los drivers y el microcontrolador, se ha desarrollado un código en el que con la ayuda de potenciómetros podemos regular la frecuencia de la señal PWM, y comprobar así que somos capaces de modificar la velocidad del motor.

```
#include <Arduino.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#define POT1 A0  // Potenciómetro 1
#define POT2 A1  // Potenciómetro 2
#define MOTOR1 9 // Pin de salida para motor 1 (Timer1)
#define MOTOR2 5 // Pin de salida para motor 2 (Timer3)
#define MicroStep 3
```

```
const int freqMin = 10;    // Frecuencia mínima en Hz
const int freqMax = 5000;  // Frecuencia máxima en Hz

volatile uint16_t time1 = 500;  // Período en microsegundos
volatile uint16_t time2 = 500;
volatile bool state1 = false; // Estado de la señal
volatile bool state2 = false;

void setup() {
    Serial.begin(115200);

    pinMode(MOTOR1, OUTPUT);
    pinMode(MOTOR2, OUTPUT);
    pinMode(MicroStep, OUTPUT);

    digitalWrite(MicroStep, HIGH); //Ponemos el driver en modo micropaso de
1/16

    cli(); // Desactivar interrupciones mientras configuramos

    // Configurar Timer1 (16 bits)
    TCCR1A = 0;  // Modo normal
    TCCR1B = (1 << WGM12) | (1 << CS11); // CTC, Prescaler de 8
    OCR1A = 500; // Valor inicial de comparación
    TIMSK1 |= (1 << OCIE1A); // TIMSK1 (Timer Interrupt Mask Register 1) y
OCIE1A (Output Compare Interrupt Enable 1A)

    // Configurar Timer3 (16 bits)
    TCCR3A = 0;
    TCCR3B = (1 << WGM32) | (1 << CS31); // CTC, Prescaler de 8
    OCR3A = 500;
    TIMSK3 |= (1 << OCIE3A); // Habilitar interrupción por comparación
OCIE3A

    sei(); // Habilitar interrupciones
}

// Interrupción Timer1 (Motor 1)
ISR(TIMER1_COMPA_vect) {
    state1 = !state1;
    digitalWrite(MOTOR1, state1);
    OCR1A = time1; // Actualizar intervalo
}

// Interrupción Timer3 (Motor 2)
ISR(TIMER3_COMPA_vect) {
    state2 = !state2;
```

```
    digitalWrite(MOTOR2, state2);
    OCR3A = time2;
}

void loop() {
    // Leer potenciómetros
    int potValue1 = analogRead(POT1);
    int potValue2 = analogRead(POT2);

    // Mapear a frecuencias deseadas
    int freq1 = map(potValue1, 0, 1023, freqMin, freqMax);
    int freq2 = map(potValue2, 0, 1023, freqMin, freqMax);

    // Calcular intervalo en ticks (1 tick = 0.5us con prescaler de 8)
    time1 = (1000000 / (2*freq1)) -1;
    time2 = (1000000 / (2*freq2)) -1;

    Serial.print("Freq1: "); Serial.print(freq1); Serial.print(" Hz, ");
    Serial.print("Freq2: "); Serial.print(freq2); Serial.println(" Hz");
  }
```

Para ello analizaremos mediante la ayuda de un osciloscopio las señales generadas por el microcontrolador conectadas al pin step del driver.

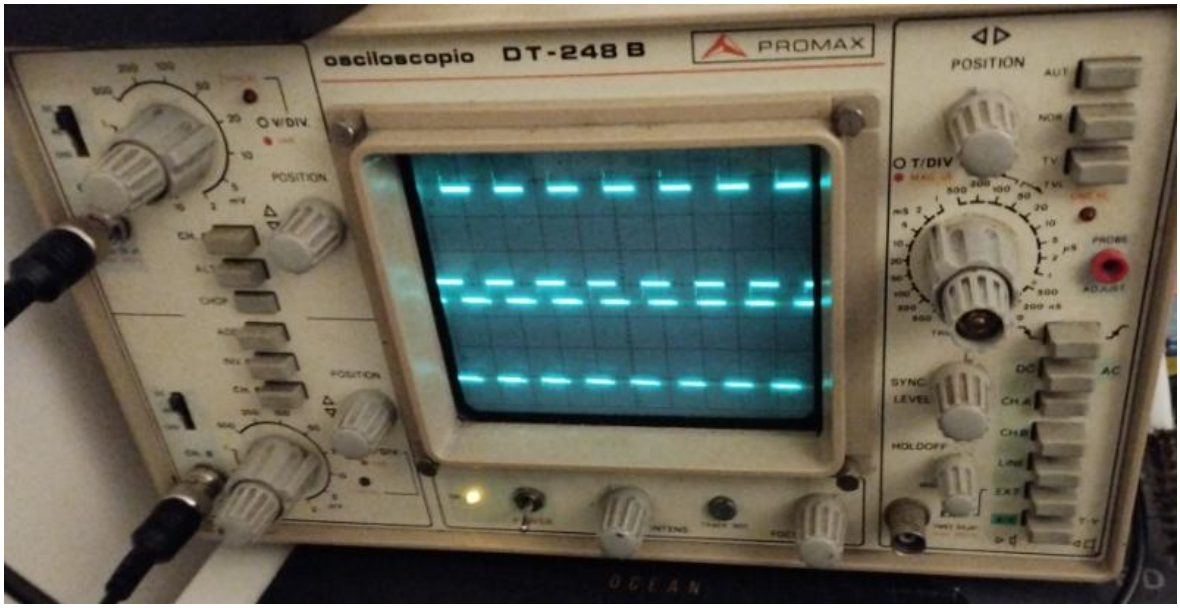


Ilustración 39 Comprobación PWM Osciloscopio

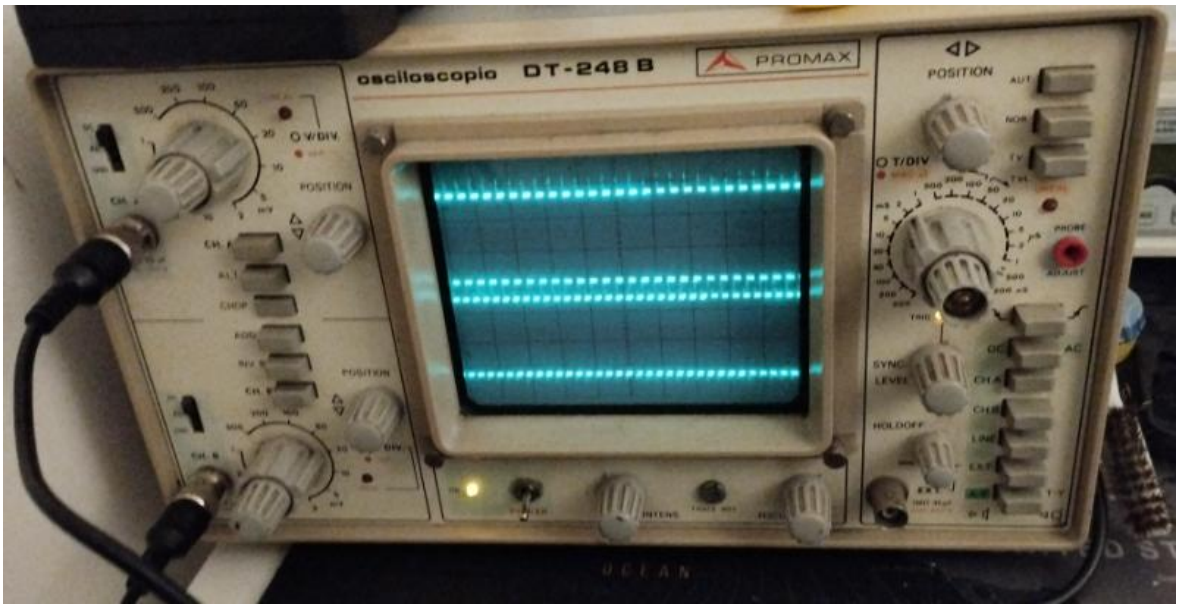


Ilustración 40 Frecuencia PWM modificada

Como podemos observar la frecuencia puede ser modificada variando a su vez la velocidad del motor paso a paso, y generando un pulso estable y sin distorsiones.

Por otra parte se ha observado que si empezamos a mover el motor y ascendemos poco a poco, podemos llegar a frecuencias altas como uno o dos, o incluso 3 KHz, pero luego cuando hemos querido arrancar el motor a un KHz o hacer el tránsito de 2 a 3KHz de una forma más rápida, el motor no ha sido capaz de alcanzar dichas frecuencias bloqueándose. Por lo que es considerable implementar una rampa de aceleración para que cuando este recibiendo las frecuencias desde LabVIEW y el motor no se quede bloqueado si el tránsito de una frecuencia a otra es muy grande, o si el motor de las frecuencias altas está trabajando con frecuencias superiores a 1KHz.

Se ha observado que esta rampa de aceleración es fundamental si queremos hacer un buen control del motor, ya que está directamente relacionada con la aceleración angular del motor $(\alpha = \frac{rad}{s^2})$ y esta con el torque $(T = N * m)$ generado. Por lo que para comprobar que no superemos el torque máximo del motor, deberemos relacionar la aceleración del motor con la fuerza generada por el torque, para ello deberemos conocer el momento de inercia del sistema $(I = Kg * m^2)$, y la aceleración angular de este.

$$T = I * \alpha$$

Ecuación 26 Segunda Ley de Newton en forma rotacional

Para obtener la inercia del oscilador recurriremos a inventor que nos proporciona el valor del momento de inercia del objeto que hemos diseñado en los 3 ejes de movimiento, y eligiendo el momento de inercia que es igual al del movimiento del motor obtendremos su valor.

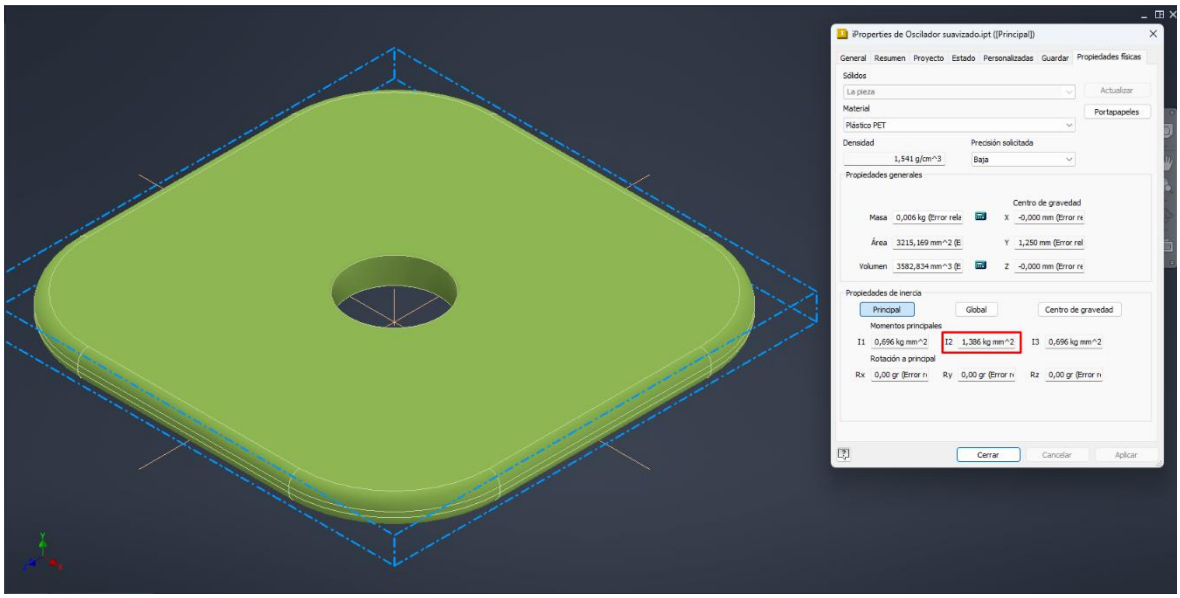


Ilustración 41 Momento de Inercia Oscilador

Como podemos observar su momento de inercia es 1.386Kg*mm^2 o 1.386*10^{-6}Kg*m, por lo que ahora solo nos queda relacionar la aceleración angular con la rampa de aceleración para asegurar que nunca superemos el torque máximo del motor ofrecido por el fabricante (43N*cm), por lo que nos dispondremos a sacar cual sería la aceleración máxima que permite el sistema. Para ello estudiaremos dos casos, partiendo desde el reposo, y ya con el motor en movimiento.

$$\alpha = \frac{T_{max}}{I} = \frac{0.43Nm}{1.386 * 10^{-6}Kg * m^2} = 0.31024531 * 10^6 \frac{rad}{s^2}$$

Este es el valor límite teórico de aceleración que, al no ser superado, garantiza que el torque aplicado se mantenga por debajo de 0.43 N·m indicados por el fabricante.

- Reposo: Si usamos una rampa de aceleración lineal, la velocidad angular viene dada por la siguiente ecuación:

$$w = \alpha * t$$

Ecuación 27 Velocidad Angular desde el reposo

Por lo que sabiendo la velocidad angular deseada y cuál es la aceleración máxima, podremos saber cuál el tiempo que tardara como mínimo en alcanzar esa velocidad.

- Movimiento: si el motor se encuentra en movimiento el tránsito de una frecuencia a otra quedaría definida por:

$\Delta w = \alpha * t$*Ecuación 28 Diferencia de velocidades angulares*

Y para no exceder el torque máximo:

$$\frac{\Delta w}{t} \leq \alpha_{max}$$

De esta forma podremos calcular teoriacamente cual es el tiempo necesario para ir de una frecuencia a otra sin superar el torque máximo ofrecido por el fabricante y asegurando el correcto funcionamiento del sistema.

5.3. CONTROL POSICIÓN MOTORES.

Para calcular la posición del motor en grados (°) en función del tiempo (t) y la frecuencia (f) de la señal, necesitamos conocer: la frecuencia de la señal (en Hz), el número de pasos por revolución del motor y el tiempo que están en movimiento.

$$Posicion\ (en\ °) = \left(\frac{f * t}{Pasos\ por\ revolucion}\right) * 360$$

Donde:

- f = frecuencia de la señal en Hz (pulsos por segundo).
- t = tiempo en segundos.
- Pasos por revolución = el número de pasos que el motor realiza para dar una vuelta completa. Para un motor de 1.8° por paso, este valor sería 200 pasos (360°/1.8°).
- 360° es el ángulo de una vuelta completa.

Por lo que creando un programa en Arduino exactamente igual que estamos para modificar las frecuencias con LabVIEW, pero estableciendo nosotros manualmente las frecuencias y haciendo que este un tiempo exacto podremos calcular su posición final y ver si alguno de los motores pierde pasos, para ello haremos varias pruebas con los dos motores cambiando la frecuencia de cada uno y dejándolos encendidos durante 5s.

El programa seria el siguiente:

```
#include <Arduino.h>
#include <avr/io.h>
#include <avr/interrupt.h>

// Definición de pines para los motores
#define MOTOR1 9 // Pin de salida para motor 1 (Timer1)
#define MOTOR2 5 // Pin de salida para motor 2 (Timer3)
#define MicroStep 3

// Rango de frecuencias permitidas en Hz
const int freqMin = 150;    // Frecuencia mínima en Hz
const int freqMax = 2000;   // Frecuencia máxima en Hz

// Variables globales de frecuencia (se almacenan en microsegundos)
volatile int time1 = 600;  // Frecuencia para motor 1
volatile int time2 = 650;  // Frecuencia para motor 2

// Frecuencias deseadas para los motores
int frecuenciaMotor1 = 600;  // Frecuencia para motor 1 (Hz)
```

```
int frecuenciaMotor2 = 650;  // Frecuencia para motor 2 (Hz)

volatile bool state1 = false; // Estado de la señal para motor 1
volatile bool state2 = false; // Estado de la señal para motor 2

unsigned long startMillis = 0; // Variable para almacenar el tiempo de
inicio
const unsigned long duration = 5000; // Duración en milisegundos (5
segundos)

void setup() {
    Serial.begin(115200);
    pinMode(MOTOR1, OUTPUT);
    pinMode(MOTOR2, OUTPUT);
    pinMode(MicroStep, OUTPUT);

    digitalWrite(MicroStep, HIGH); // Ponemos el driver en modo micropaso de
1/16

    cli(); // Desactivar interrupciones mientras configuramos

    // Configurar Timer1 (16 bits) para Motor 1
    TCCR1A = 0;  // Modo normal
    TCCR1B = (1 << WGM12) | (1 << CS11); // CTC, Prescaler de 8
    OCR1A = 500; // Valor inicial de comparación
    TIMSK1 |= (1 << OCIE1A); // Habilitar interrupción Timer1

    // Configurar Timer3 (16 bits) para Motor 2
    TCCR3A = 0;
    TCCR3B = (1 << WGM32) | (1 << CS31); // CTC, Prescaler de 8
    OCR3A = 500;
    TIMSK3 |= (1 << OCIE3A); // Habilitar interrupción Timer3

    sei(); // Habilitar interrupciones

    startMillis = millis(); // Guardar el tiempo de inicio
}

ISR(TIMER1_COMPA_vect) {
    state1 = !state1;
    digitalWrite(MOTOR1, state1);
    OCR1A = time1; // Actualizar intervalo para motor 1
}

ISR(TIMER3_COMPA_vect) {
    state2 = !state2;
    digitalWrite(MOTOR2, state2);
```

```
    OCR3A = time2; // Actualizar intervalo para motor 2
}

void loop() {
    // Verificar si han pasado 5 segundos
    if (millis() - startMillis < duration) {
        // Calcular el intervalo de tiempo para el motor 1 (en
microsegundos)
        if (frecuenciaMotor1 >= freqMin && frecuenciaMotor1 <= freqMax) {
            noInterrupts();
            time1 = (2000000 / (2 * frecuenciaMotor1)) - 1; // Formula para
calcular el tiempo entre pulsos
            interrupts();
        }

        // Calcular el intervalo de tiempo para el motor 2 (en
microsegundos)
        if (frecuenciaMotor2 >= freqMin && frecuenciaMotor2 <= freqMax) {
            noInterrupts();
            time2 = (2000000 / (2 * frecuenciaMotor2)) - 1; // Formula para
calcular el tiempo entre pulsos
            interrupts();
        }
    } else {
        // Detener los motores después de 5 segundos
        TIMSK1 &= ~(1 << OCIE1A);  // Deshabilita la interrupción de
comparación A del Timer1
        TIMSK3 &= ~(1 << OCIE3A);  // Deshabilita la interrupción de
comparación A del Timer3
    }
}
```

Una vez tenemos el código listo nos dispondremos a preparar los motores para poder medir el ángulo con el que termina después de 5s.

Para ello se ha imprimido y pegado sobre un cartón para darle más rigidez un circulo con todos los grados, que se ha unido al motor mediante dos tornillos evitando su descentre, y permitiendo así medir el ángulo de inicio el de finalizar y el número de vueltas que da, pudiendo comprobar si el valor teórico y el real coinciden o hay perdida de pasos, que si hubiera algún descuadre simplemente midiendo la diferencia en grados y relacionándolo con que un paso son 1.8º tendríamos el número de pasos perdidos.

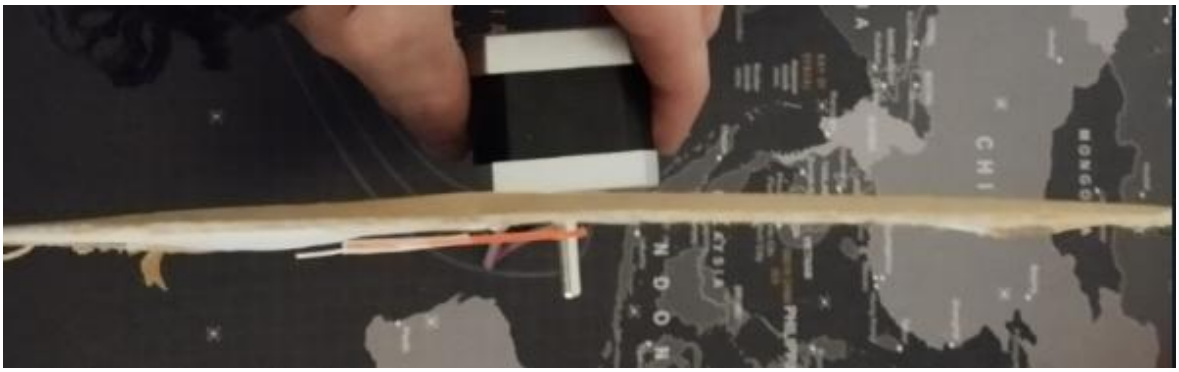


Ilustración 43 Unión al motor

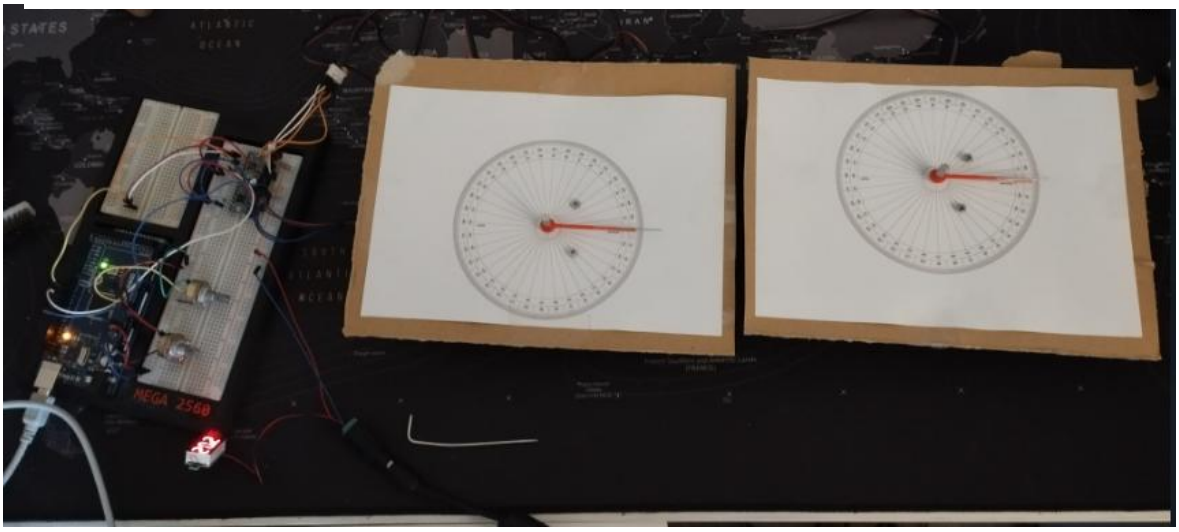


Ilustración 42 Control de posicion motores paso a paso

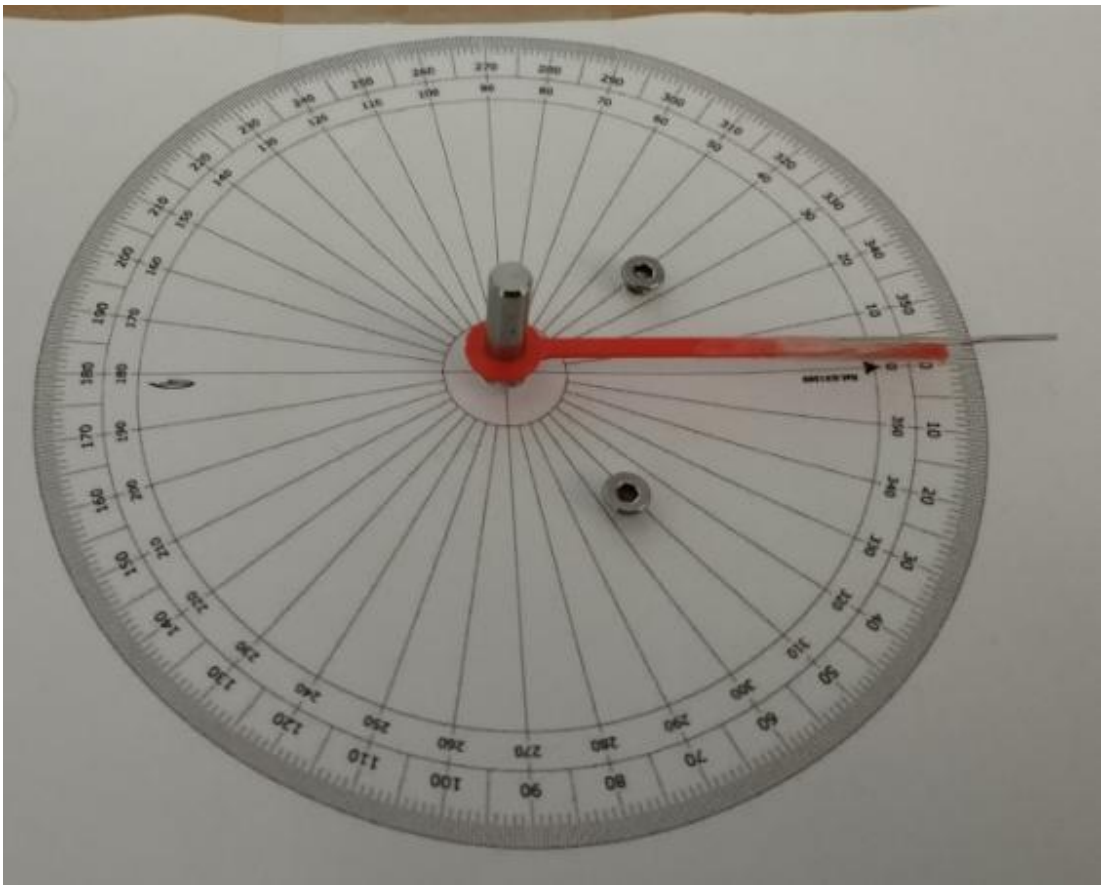


Ilustración 44 Medición de Posición

Los resultados obtenidos son los siguientes:

	Frecuencia	Pos Teórica	Pos Real	Diferencia
Motor 1	500	4500	4500	0
Motor 2	550	4950	4950	0
Motor 1	600	5400	5400	0
Motor 2	650	5850	5850	0
Motor 1	700	6300	6300	0
Motor 2	750	6750	6750	0

5.4. Creación de figuras de Lissajous con láser y espejos

Una vez probado que el análisis del audio, la comunicación con el microcontrolador, y el control de los motores es correcto procederemos a realizar una prueba y generar un círculo moviendo los dos motores a la misma velocidad.

Para ello montaremos el sistema diseñado.

Ilustración 45 Montaje plano ortogonal

Y en la pared se podrán observar la figura generada:

Ilustración 46 Proyección

Como se puede observar en la imagen anterior, se genera un patrón, pero no el esperado, eso se debe a que el movimiento del soporte del espejo no realiza un movimiento lineal, ya que al tener holgura para que pueda deslizar sin ofrecer mucha resistencia transmite vibraciones y movimientos no deseados.

Otro problema que se ha observado es que a velocidades altas el motor pierde par y no es capaz de vencer la fuerza ejercidad por la goma de retorno y por las fricciones ejercidas entre la base y el soporte del espejo, impidiendo así un control a frecuencias altas.

6. CONCLUSIONES

Para el desarrollo de este TFG se ha implementado un sistema de proyección de Figuras de Lissajous mediante el análisis espectral de pistas de audio. El estudio ha permitido explorar la relación entre el sonido y el movimiento mecánico generando una representación dinámica de una pista de audio.

Los principales objetivos del proyecto se han cumplido satisfactoriamente obteniendo resultados positivos en:

- Implementación de un sistema de análisis espectral en tiempo real.
- Control de motores modificando la frecuencia de la señal PWM. Se ha conseguido modular la velocidad y dirección de los motores paso a paso en función de las frecuencias detectadas, facilitando la representación visual de patrones de Lissajous con láseres y espejos.
- Se ha desarrollado e impreso en 3D una estructura que permite el movimiento adecuado de los espejos para reflejar el haz de láser en función de las señales de control generadas.

A pesar de ser exitosas casi todas las pruebas se han detectado varios problemas al desarrollar el proyecto:

- Problema con la velocidad de los motores paso a paso: la necesidad de diseñar un olcilador para aumentar la velocidad con la que se realizaba el ciclo del movimiento armónico simple podría haberse resuelto habiendo usado motores Brusheless los cuales tienen un movimiento mucho mas rápido llegando a alcanzar las 9000rpm.
- Problema con la fricción: Aunque el movimiento de los motores a velocidades bajas era adecuado, cuando subimos la velocidad aumenta la fricción y se para el motor. Este problema podría haberse solucionado usando guias lineales que permiten un movimiento rápido sin fricciones ni vibraciones solcionando así el problema de la fricción y el problema de la proyección, que al disminuir el ajuste entre las piezas están tienen mas holgura, lo que significa que ya ese movimiento no es lineal provocando que las figuras no correspondan con su verdadera representación.
- Optimización del procesamiento de datos: Dado que Arduino esta trabajando con string y funciones que requieren bastante capacidad de calculo se ha tenido que actualizar las frecuencias con el microcontrolador cada segundo, este tiempo podría

disminuir si la lectura de la UART en vez de con string se realizara con buffer de cadenas de caracteres.

En conclusión, este proyecto ha demostrado la viabilidad de utilizar el análisis espectral para la generación de patrones visuales dinámicos mediante motores y láseres, sentando las bases para futuras mejoras y aplicaciones en espectáculos audiovisuales, educación y arte digital. La integración de técnicas de procesamiento de señales con sistemas electromecánicos abre un campo de investigación y desarrollo con amplias posibilidades de innovación.

7. OBJETIVOS DE DESARROLLO SOSTENIBLE

Los objetivos de este Trabajo Fin de Grado están alineados con los siguientes Objetivos de Desarrollo Sostenible (ODS) y metas, de la Agenda 2030:

- Objetivo 4 - Garantizar una educación inclusiva y equitativa de calidad y promover oportunidades de aprendizaje permanente para todos
- Meta 4.4 De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento

- Objetivo 8 - Promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos
- Meta 8.2 Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra

8. BIBLIOGRAFÍA

Relación de documentos

(X) Memoria NN páginas

(_) Anexos NN páginas

La Almunia, a dd de mm de 202x

Firmado: David Martín-Sacristán Avilés

--------INDICACIONES PARA LA ENTREGA FINAL--------

Actualizar el número de páginas de los documentos a entregar, actualizar la fecha y después de imprimir firmar sobre el nombre de autor. Esta hoja deberá incluirse al final del documento.