UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

DISEÑO Y CONSTRUCCIÓN DE UN SIMULADOR DE RESONANCIA EN ESTRUCTURAS

PRESENTADO POR:

ADRIÁN ANTONIO MIRA TRIGUEROS

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, FEBRERO DE 2022.

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL:

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO:

PhD. EDGAR ARMANDO PEÑA FIGUEROA

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR:

ING. ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título:

DISEÑO Y CONSTRUCCIÓN DE UN SIMULADOR DE RESONANCIA EN ESTRUCTURAS

Presentado por:

ADRIÁN ANTONIO MIRA TRIGUEROS

Trabajo de Graduación Aprobado por:

Docente Asesor:

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, febrero de 2022.

Trabajo de Graduación Aprobado por:

Docente Asesor:

DR. CARLOS EUGENIO MARTÍNEZ CRUZ

NOTA Y DEFENSA FINAL

En esta fecha, lunes 4 de noviembre de 2021, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 3:30 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director

   Firma

2. MSc. José Wilber Calderón Urrutia
   Secretario

   Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- DR CARLOS EUGENIO MARTINEZ CRUZ
  (Docente Asesor)

  Firma

- DR. CARLOS OSMIN POCASANGRE JIMENEZ

  Firma

- ING. JOSE MIGUEL HERNANDEZ

  Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

DISEÑO Y CONSTRUCCIÓN DE UN SIMULADOR DE RESONANCIA EN ESTRUCTURAS

A cargo del Bachiller:

- MIRA TRIGUEROS ADRIAN ANTONIO

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 9.2

(nueve punto dos)

Agradecimientos

He de agradecer a Dios por brindarme sabiduría, salud y las herramientas necesarias para culminar la carrera y este trabajo de graduación. A mi madre quien me ha brindado todo su apoyo y lo necesario para desarrollar mi carrera universitaria. A Gloria Regina, José Rafael Arvizu y sus hijos, por su apoyo a lo largo de la carrera. Al Dr. Carlos Eugenio Martínez, por ofrecer el tema, haber facilitado su realización y su paciencia en cada dificultad presentada. A Reina Vides que ha sido parte fundamental en el proceso que conlleva este trabajo de graduación. Por último, a mis compañeros que durante la carrera brindaron su apoyo en cada materia.

Adrián Antonio Mira Trigueros

1. Índice general

Índice de figuras.

Índice de tablas.

Capítulo I: Introducción.

En el estudio de la carrera de ingeniería es poco común que fenómenos muy importantes, como el de la resonancia, se puedan presentar de manera practica usualmente se presentan sus teorías y formas de análisis pero difícilmente su experimentación, estudiantes que han llevado cursos de mecánica, de circuitos y de electromagnetismo, tienen todas las bases para poder describir el fenómeno de resonancia pero se tiene un conocimiento muy deficiente de lo que es el fenómeno, así como del amplio campo de sus aplicaciones. Es por esta razón que construye un dispositivo llamado "simulador de resonancia en estructuras" cuyo diseño se basa en una mesa vibratoria y permite el estudio de la resonancia de una manera práctica.

Se presenta en esta investigación un dispositivo capaz de demostrar el fenómeno de la resonancia, permitiendo el análisis y comprensión del fenómeno. El desarrollo de este simulador de resonancia tiene como objetivo demostrar como las vibraciones causadas por un agente externo afectan a una estructura, demostrar que para cada estructura existe una frecuencia natural con la que la estructura entra en resonancia.

Con el objetivo de conocer la frecuencia de resonancia de una estructura el dispositivo es capaz de generar un barrido de frecuencias de una onda senoidal, para conocer la frecuencia a la que opera el simulador se implementa un sistema de medición y análisis de datos con una interfaz gráfica para presentar esta frecuencia.

2.1 Interés de la investigación

La investigación surge del interés por desarrollar un simulador capaz de hacer entrar en resonancia estructuras a escala. Fue en la materia de Aplicaciones de Tratamiento Digital de la Señal (ATD115) en la cual se reproduce la idea de una mesa simple que al moverla provoca que una estructura entre en resonancia, este fenómeno se presenta en el análisis estructural y la obtención de las señales involucradas en el sistema presenta una oportunidad para entender este fenómeno.

Luego de un diseño preliminar se continúa desarrollando esta idea en proyecto de ingeniería para lograr un control automático de este dispositivo, teniendo como resultado un simulador de resonancia para estructuras a escala. Este simulador se desarrolla en proyecto de ingeniería I y II, presentando el resultado de este desarrollo en el presente trabajo de graduación, se logran los objetivos en estas investigaciones previas, en el presente trabajo de graduación se perfecciona y se implementa un control con un microcontrolador de doble núcleo permitiendo una mejor implementación del dispositivo. El desarrollo de este dispositivo quedará como base para futuras investigaciones de temas relacionados, mejoras que siempre pueden realizarse y a la curiosidad que pueda generar en posibles usuarios del dispositivo.

2.2 Antecedentes

Este trabajo de graduación tuvo sus orígenes en las asignaturas de proyecto de ingeniería I y II, durante estas investigaciones se presentan dos mecanismos de movimiento diferentes uno utilizando un mecanismo de biela-manivela y en la siguiente investigación se presenta el diseño utilizando un tornillo trapezoidal con guías lineales.

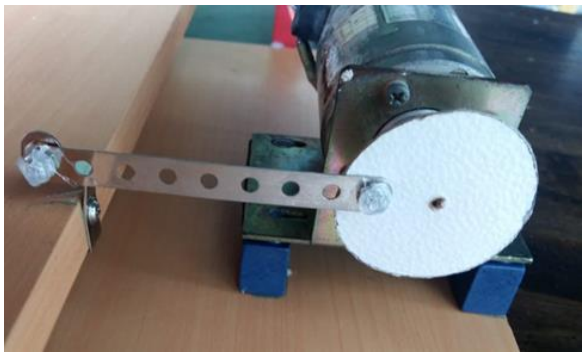


FIGURA 2 Simulador de resonancia biela manivela

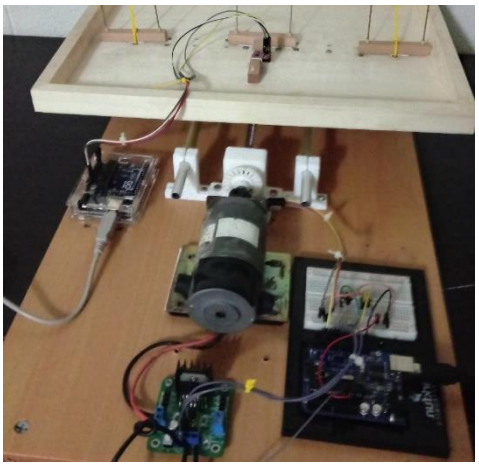


FIGURA 1 Simulador de resonancia ejes lineales

2.3 Motivación para realizar el proyecto

El estudio teórico de los métodos de análisis de señales se presenta a lo largo del estudio de la Ingeniería Eléctrica, siendo un conocimiento que puede reforzarse de manera práctica o visual. Teniendo en cuenta que el fenómeno de la resonancia tiene la ventaja de una fácil demostración se presenta un dispositivo capaz de someter estructuras a escala a este fenómeno. Para un público con conocimientos básicos en el tema le resulta fácil explicar el porqué del fenómeno al observar el comportamiento de diferentes estructuras sometidas a diferentes frecuencias que hacen que estas entren en resonancia. Además, se facilita su comprensión al observar el fenómeno.

Este fenómeno permite analizar el comportamiento de estructuras sometidas a diferentes frecuencias y su reacción al someterlas a su frecuencia natural de oscilación. El tener una herramienta capaz de reproducir el efecto de la resonancia y que no sea destructivo para las estructuras motiva para realizar un dispositivo capaz de verificar el efecto de la resonancia sobre estructuras.

Para esto se desarrolla un simulador de resonancia en estructuras a escala, dispositivo que genera aceleraciones en un eje a cualquier objeto que se encuentre asegurado sobre su superficie.

2.4 Objetivos

2.4.1 Objetivo general

Desarrollar un dispositivo capaz de demostrar el fenómeno de la resonancia en estructuras a escala.

2.4.2 Objetivos específicos

- Determinar una implementación adecuada para el control del dispositivo que permita una operación adecuada del mismo.

- Diseñar los circuitos de control y medición del dispositivo.

- Integrar el software de control y medición en un único microcontrolador.

- Diseñar un sistema de sujeción de las estructuras a la mesa, teniendo en cuenta que estará sometida a cambios repentinos de aceleración.

- Presentar en tiempo real el análisis en frecuencia de los datos de aceleraciones utilizando la transformada rápida de Fourier.

2.5 Organización

El trabajo presenta las etapas en el proceso de desarrollo y construcción de un simulador de resonancia en estructuras, basado en una mesa vibratoria. Se describen los resultados obtenidos del desarrollo del dispositivo y se desarrolla un experimento el cual permite observar una serie de estructuras sometidas por el simulador a un movimiento periódico, el dispositivo es capaz de generar resonancia en las estructuras cuando su frecuencia se iguala a la frecuencia natural de estas. Por último, se ofrecen orientaciones para su uso y una guía para utilizar el simulador.

Capítulo 1 describe una introducción al dispositivo, además plantea que se quiere lograr en el desarrollo del simulador y se ofrecen algunos antecedentes de la investigación.

Capítulo 2 ofrece una revisión de los principios del fenómeno, así como la teoría del dispositivo, el control, desempeño y uso de las mesas vibratorias.

Capítulo 3 se cada componente del sistema y como interactúan en el diseño, es decir, como el diseño mecánico afecta el diseño eléctrico de manera que se tenga un resultado favorable al final de la implementación. Se plantea como será controlado el dispositivo y como se obtendrán datos para su posterior análisis y presentación.

Capítulo 4 examina con más detalle el simulador en este capítulo se presentan los detalles de construcción del dispositivo, dando una guía de cómo se implementan el mecanismo de movimiento y el control del dispositivo. Además, se explica la medición y análisis de las señales obtenidas del simulador.

Capítulo 5 se presentan los resultados obtenidos tanto del diseño, como del experimento realizado para demostrar su funcionamiento.

Capítulo II: Marco teórico.

Este capítulo presenta los conceptos generales en los que se basa el simulador, presentando las bases teóricas de la implementación para posteriormente demostrar los conceptos aquí presentados. Además, en esta sección se presentan los diferentes tipos de mesas vibratorias que existen, y esboza sus características mecánicas básicas y los métodos comunes para controlarlas.

3.1 Resonancia

En física, la resonancia es el fenómeno del incremento de amplitud de un sistema u objeto con una frecuencia natural, que se ve sometido a una fuerza externa periódicamente aplicada, y la frecuencia de esta fuerza externa es igual o cercano a la frecuencia natural del sistema en el cual se aplica [1]. Para cada sistema existe una frecuencia en la cual la amplitud del movimiento es máxima, a esta frecuencia se le denomina frecuencia de resonancia. El fenómeno de resonancia se presenta cotidianamente en la vida del ser humano, es por ello que se plantea la importancia de su conocimiento.

Un ejemplo de los efectos destructivos que pueden producirse en caso de resonancia, se presenta cuando una ciudad es afectada por un sismo; la ciudad está llena de estructuras elásticas de gran escala, tales como edificios y puentes; la frecuencia de los sismos, es decir, la frecuencia con que se mueve el suelo, está ante todo en el rango de los 0.5 -2 Hz, son frecuencias relativamente bajas, pero las grandes masas de los edificios de más de 5 pisos de altura por su propia inercia tienden a tener frecuencias bajas y propician por tanto la ocurrencia del fenómeno de resonancia. En este caso la amplitud de las oscilaciones mecánicas de los edificios tiende a crecer tanto en cada ciclo que pueden llegar al punto de ruptura, tal como sucedió con muchos edificios en el terremoto de la ciudad de México en 1985.

3.2 Teoría del simulador

Para presentar los resultados obtenidos en el desarrollo del simulador es importante conocer los principios teóricos en los que se basa. Para lograr el objetivo de hacer entrar en resonancia estructuras a escala el simulador se desarrolla tomando una mesa vibratoria como base para el diseño, una mesa vibratoria es una herramienta utilizada para el análisis estructural y mayormente para el análisis de estructuras frente a sismos.

Para simplificar y facilitar la comprensión del fenómeno el simulador de resonancia se construye únicamente con un grado de libertad, es decir, únicamente genera aceleraciones en un eje, además se diseña únicamente para generar movimientos en forma senoidal permitiendo realizar un barrido en frecuencia para conocer la frecuencia natural de una estructura. Las consideraciones descritas anteriormente se toman para mantener el experimento lo más simple posible.

3.2.1 Mesa vibratoria

Una mesa vibratoria es una plataforma móvil que simula los movimientos de un sismo (o movimientos periódicos dependiendo de su configuración de movimiento) sobre un modelo estructural [2], tal como se observa en la FIGURA 3. Las mesas vibratorias pueden ser clasificadas según el tamaño, el tipo de actuador que genera el movimiento o los grados de libertad [3]. Los ensayos en mesa vibratoria generalmente involucran modelos a escala reducida.

En general, en el diseño de la mesa vibratoria se deben llevar a cabo cinco tipos de diseño: mecánico, eléctrico, estructural (cimentación), del sistema de control y del sistema de adquisición de datos [4].

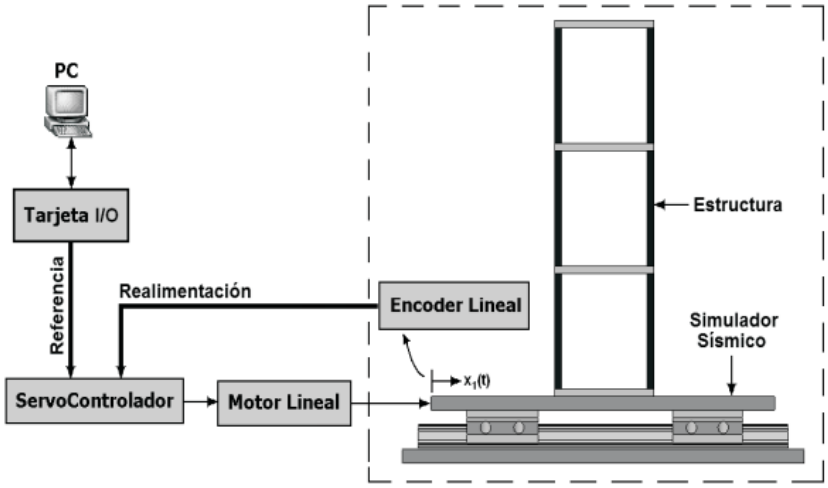


FIGURA 3 Mesa vibratoria. [4]

3.2.2 Mesa vibratoria de un eje

Las mesas vibratorias de un eje son la forma más sencilla de mesa vibratoria, en la que una plataforma montada sobre cojinetes es sacudida por uno o más actuadores. Estas mesas están normalmente orientadas para hacer vibrar una muestra horizontalmente, aunque algunas pueden ajustarse para que sea posible el movimiento vertical (únicamente). Para muchos ensayos sólo es deseable excitar la muestra en un eje, ya que esto simplifica la interpretación posterior de los resultados. En estos casos, las mesas de un solo eje pueden ser la mejor solución para realizar el ensayo si tienen una capacidad dinámica suficiente. Las mesas de un eje también son más sencillas de controlar. Los problemas mecánicos que experimentan se deben a los rodamientos lineales que se suelen utilizar para restringir el movimiento en un solo eje y evitar el problema de los movimientos horizontales y de cabeceo, ya que los cojinetes son normalmente muy rígidos en todos los ejes excepto en el eje libre [2].

3.3 Simulador como herramienta de investigación

Aunque el simulador es solo una herramienta demostrativa del fenómeno de la resonancia es posible analizar las señales que se adquieren con los sensores implementados y de esta forma realizar un análisis de los datos utilizando las diferentes herramientas disponibles. En esta investigación se desarrolla un experimento para demostrar el fenómeno de resonancia en estructuras simples a escala, es posible con algunas modificaciones desarrollar experimentos más complejos, inclusive para el análisis de sismos.

Una vez que se ha decidido el tipo de ensayo a realizar con la mesa de vibraciones se desarrollan los pasos apropiados para un programa de investigación, hay dos maneras de seleccionar la metodología que se utilizará para realizar las pruebas. En el primer caso, se puede planificar la prueba y, a continuación, elegir una mesa con una capacidad y un rendimiento adecuados para realizarla. La elección de la mesa de vibración puede ser difícil y dependerá de las necesidades específicas del programa de investigación. Además, puede haber limitaciones en las características de funcionamiento del simulador desarrollado. Las pruebas con mesas de vibración son caras, y cuanto más grande sea la instalación que se vaya a utilizar o la prueba que se planifique, más costará un experimento concreto. Esto suele restringir el uso de las mesas muy grandes a los grandes programas de investigación en los que se justifica el gasto de los ensayos a grandes modelos. [2].

La segunda opción, más común, para los investigadores es que una mesa concreta y que, por lo tanto, el ensayo debe diseñarse en función de la capacidad y el rendimiento de esa mesa específica. En cualquiera de los dos casos, los investigadores deben conocer de los tipos y características de varias o determinadas mesas de vibración.

3.4 Características mecánicas de las mesas de vibración

Las características de funcionamiento de las mesas vibratorias pueden variar principalmente por los grados de libertad. Además, hay diferentes materiales y disposiciones que pueden utilizarse para los componentes individuales que forman una mesa de vibración. A continuación, se describen los diferentes tipos de componentes mecánicos principales de una mesa vibratoria, seguidos de detalles sobre el comportamiento de cada uno de estos componentes y su efecto en el comportamiento dinámico global de una mesa de vibración.

3.4.1 Componentes mecánicos, hardware

En esta sección se describen todos los componentes mecánicos principales y los transductores asociados en un sistema de mesa de vibración. También se debe hacer referencia a la FIGURA 3 donde todos estos componentes se muestran en un diagrama de una mesa de vibración típica. Hay que tener en cuenta que no todos estos componentes se utilizan en todas las mesas.

- **Acelerómetros:** Se utilizan para proporcionar información al sistema de control sobre la aceleración de las diferentes partes de la plataforma.
- **Actuadores:** Todas las mesas, excepto las más pequeñas, utilizan actuadores servo-hidráulicos. Los actuadores electrodinámicos no se utilizan normalmente en la investigación sísmica debido a su restringida capacidad de fuerza. Sin embargo, los ingenieros mecánicos suelen utilizar pequeñas mesas vibratorias con este tipo de actuadores para los ensayos de fatiga de pequeños componentes.
- **Lugares de anclaje:** Alguna disposición de lugares de anclaje en la superficie de la plataforma de la mesa vibratoria para permitir la fijación de las muestras.
- **Rodamientos:** Algún tipo de cojinete esférico hidráulico o mecánico que permita la libre rotación de los extremos de los actuadores.
- **Plataforma de la mesa de vibración:** Normalmente es una estructura muy rígida de aluminio o acero.
- **Amortiguadores:** Dispositivos para amortiguar cualquier movimiento generado en la masa de reacción durante las sacudidas.
- **Masa de reacción:** Proporciona una masa muy rígida contra la que pueden empujar los actuadores.
- **Muestra de ensayo:** La estructura o modelo real que se somete a ensayo bajo carga vibratoria.

3.5 Comportamiento de los componentes de una mesa de vibración

Se puede decir que una mesa de vibración ideal es aquella en la que el número necesario de ejes puede controlarse con precisión para cualquier movimiento necesario y donde todos los movimientos en los otros ejes pueden mantenerse en cero. Además, lo ideal es que no haya ninguna interacción entre la mesa vibratoria y la muestra que se está ensayando. Lamentablemente, esto no es completamente posible, porque siempre habrá algo de flexibilidad y no linealidad en el sistema de la mesa vibratoria que tendrá que ser compensado por el hardware y el software que se utiliza para controlar los actuadores individuales en el sistema de la mesa de vibración [4].

El comportamiento de los componentes de una mesa vibratoria presenta una característica común, la flexibilidad de sus componentes, esto puede causar problemas para el control del sistema, estos problemas se listan a continuación.

1. La flexibilidad de la masa de reacción en el sistema de suspensión / amortiguación.
2. La flexibilidad interna en la masa de reacción.
3. La flexibilidad local de los soportes o de la masa de reacción en la conexión con los rodamientos de los actuadores.
4. La flexibilidad y cualquier holgura en los cojinetes de los actuadores (tanto en la plataforma y de la masa de reacción de los actuadores).
5. La rigidez axial, de torsión y flexión lateral de cualquier tubo de torsión u otro sistema de sujeción conectado a la plataforma.
6. La flexibilidad de la propia plataforma.

Estas resonancias son especialmente importantes porque si el modelo que se está probando tiene una frecuencia natural cercana a una de las resonancias de la mesa, es posible que el sistema de control de la mesa vibratoria no sea capaz de compensar una interacción significativa entre el comportamiento de la mesa y la muestra.

Cada una de las resonancias mecánicas señaladas anteriormente y otras cuestiones que afectan al rendimiento de una mesa de más detalladamente a continuación:

3.5.1 Masa de la plataforma

La última consideración principal es la masa de la plataforma, y aquí hay dos cuestiones conflictivas. En primer lugar, una plataforma muy ligera requerirá actuadores más pequeños para moverla, lo que reduce el costo inicial de la instalación. Sin embargo, una plataforma muy ligera es mucho más probable que se vea afectada por una interacción significativa entre la mesa y la muestra, lo que requerirá un mejor sistema de hardware y software para controlarla eficazmente. Por lo tanto, cuanto mayor sea la plataforma es mejor desde el punto de vista del espécimen, pero esto requiere actuadores más grandes, aumenta los costos de funcionamiento y puede limitar las aceleraciones máximas que se pueden alcanzar. Una solución a este conflicto es el uso de una plataforma ligera con actuadores de gran capacidad. Para muestras pequeñas y ligeras, la plataforma se utiliza tal cual, pero cuando una muestra más grande que pueda interactuar significativamente con la mesa, se puede añadir una masa estática adicional, hasta la capacidad de la mesa.

Esta masa adicional ayudará a reducir la interacción entre la mesa y la probeta al aumentar la masa de la plataforma que tiene que ser excitada por el espécimen. Las mejoras en el rendimiento de la mesa que se pueden conseguir mediante la adición de una masa estática.

3.6 Métodos de control de las mesas vibratorias

Inicialmente podría parecer que las mesas de un solo eje, al ser las más sencillas mecánicamente, serían controladas de forma sencilla. Sin embargo, estas mesas siguen teniendo todos los problemas descritos anteriormente.

Sin embargo, antes de considerar los métodos que se pueden utilizar para controlar las mesas de vibración, vale la pena examinar los tipos de movimiento que se utilizan normalmente en las pruebas con este tipo de aparatos [2].

3.6.1 Tipos de movimiento que se utilizan normalmente

Hay muchos tipos diferentes de movimiento de la mesa vibratoria que se utilizan normalmente para la investigación, el seleccionado para esta investigación es el de ondas senoidales. Estos son:

Ondas sinusoidales: Pueden ser muy eficaces en el estudio de sistemas estructurales muy complicados en los que se desea mantener la respuesta de la estructura lo más sencilla como sea posible.

Al mantener el movimiento es mucho más fácil comparar los resultados de cualquier trabajo experimental con los estudios teóricos para este tipo de sistemas. Con este tipo de movimiento también es posible transferir toda la energía de la mesa vibratoria directamente a la estructura en su frecuencia natural, y esto puede producir una prueba mucho más severa para el espécimen. De este modo incluso una mesa de pequeña capacidad puede someter a un espécimen a una prueba muy severa. Sin embargo, como este tipo de movimiento no tiene una amplia gama de frecuencias, normalmente sólo excita un modo de vibración en la estructura que, aunque es muy útil para comprender el comportamiento básico de la estructura, puede ser engañoso al considerar el comportamiento de la estructura bajo una situación real.

Pruebas de impulsos: Estos tipos de entrada pueden utilizarse para caracterizar el comportamiento dinámico (frecuencias naturales y coeficientes de amortiguación) de una estructura en una mesa de vibración. Sin embargo, rara vez se utilizan en los ensayos de mesas vibratorias, ya que pueden crear tensiones muy elevadas en la propia mesa al funcionar los actuadores a su máxima capacidad de fuerza.

El uso de una señal de ruido aleatorio (véase más adelante) producirá una caracterización estructural igual de buena, si no mejor, y este tipo de movimiento es el más adecuado para los ensayos con mesas vibratorias.

Barrido sinusoidal: Este tipo de entrada se utiliza generalmente para determinar las frecuencias naturales y valores de amortiguación de una estructura de prueba. Sin embargo, las estructuras con poca amortiguación pueden dañarse fácilmente cuando la frecuencia de la señal alcanza una de las frecuencias naturales del modelo. La realización de un ensayo con este tipo de señal también puede requerir bastante tiempo, por lo que se prefiere el uso de una señal de ruido aleatorio para caracterizar el modelo.

Ruido aleatorio: Este tipo de entrada se genera combinando una amplia banda de frecuencias con fases aleatorias en una sola señal. Este tipo de movimiento de entrada puede ser muy útil para determinar las frecuencias naturales y la amortiguación de la estructura que se está probando. Dado que la estructura no está siendo continuamente excitada a una frecuencia natural, es menos probable que sea sobreexcitada y dañada por la prueba.

3.6.2 Técnicas de control

Todas las mesas vibratorias incorporan dos tipos distintos de sistemas que se utilizan para controlar el movimiento de la plataforma. Estos sistemas se denominan a menudo de diferentes maneras, pero para simplicidad se denominarán el sistema de control de hardware y el sistema de control de software. A continuación, se describen cada una de estas técnicas de control en las secciones siguientes. Esto se presentará a detalle en un capítulo posterior.

3.6.3 Sistema de control, hardware

Este es el hardware electrónico que controla los actuadores individuales en la mesa de vibración e incorpora los bucles de retroalimentación. Estos bucles de retroalimentación son básicamente versiones mejoradas y ampliadas de un simple bucle de retroalimentación proporcional. Los bucles de retroalimentación simples como este funcionan restando la posición real del actuador de la posición deseada y utilizando esta nueva señal para accionar el actuador. Este caso sencillo se denomina "retroalimentación proporcional" porque la magnitud de la señal que controla el valor del motor es proporcional a la diferencia entre el lugar en el que debería estar el actuador y el lugar en el que se encuentra realmente. La velocidad con la que el actuador responde entonces puede variarse ajustando la "ganancia" o "amplificación" que se aplicada a la señal de control. Los bucles de retroalimentación más complicados de algunos sistemas de mesas de vibración de vibratorias utilizan términos de retroalimentación adicionales, cada uno con su propia ganancia, para mejorar la velocidad y precisión de la respuesta del actuador en una amplia gama de frecuencias. El proceso de ajustar las distintas ganancias para optimizar la respuesta de un actuador se denomina "sintonización".

El sistema idealmente sintonizado tendría una ganancia unitaria y ninguna diferencia de fase entre la señal de accionamiento y la respuesta de la plataforma en cada eje, en toda la gama de frecuencias de funcionamiento, tanto con o sin una muestra en la plataforma (es decir, la respuesta de la plataforma sigue exactamente la señal de entrada deseada).

Es importante tener en cuenta que incluso si el sistema de control de hardware pudiera ajustarse para que la respuesta en frecuencia de la plataforma de la mesa vibratoria fuera unitaria en todas las frecuencias en todos los ejes excitados y nula en todos los demás ejes, seguiría siendo necesario un sistema de control de software adicional para compensar cualquier respuesta no lineal de la muestra.

3.6.4 Sistema de control, software

Es el software que se utiliza para compensar cualquier inexactitud en el ajuste del sistema de control de hardware. En teoría, este software permite que el movimiento deseado se pueda reproducir con precisión en la plataforma de la mesa vibratoria. El software actual para controlar de las mesas vibratorias lo hace mediante un proceso iterativo de registro del movimiento de la plataforma logrado durante una prueba y, a continuación, corrigiendo la señal de accionamiento para que el movimiento de la plataforma en la siguiente prueba se acerque más al movimiento requerido [10]. Este procedimiento compensará cualquier error en el ajuste del hardware y también puede compensar cualquier interacción lineal entre la mesa de vibración y el espécimen, pero al ser un procedimiento iterativo que no reacciona en tiempo real, no puede compensar cualquier movimiento no lineal de la mesa o de la muestra.

3.7 Componentes del sistema, Hardware y Software.

Para el correcto funcionamiento del simulador se tienen una serie de sistemas que lo integran y permiten el adecuado funcionamiento del dispositivo, se describen cada uno de estos sistemas para luego en la fase de diseño e implementación tener claro para sirven cada uno de estos y su función principal.

3.7.1 Unidad de potencia

Este sistema se encarga de otorgar la potencia necesaria, aceleración y frecuencia requeridas por el simulador, entre los posibles tipos de unidades se encuentran unidades mecánicas, eléctricas o hidráulicas. La opción que se adapta al diseño propuesto para el simulador se tiene la unidad de potencia eléctrica en este caso utilizando un motor de corriente directa DC que ofrece características de control, facilidad de uso, costo accesible y rendimiento adecuado para el simulador.

3.7.2 Unidad de control

Este es el sistema que se encargara de transmitir el movimiento, aceleración, potencia y frecuencia del simulador, dentro de los posibles tipos podemos encontrar actuadores neumáticos, hidráulicos, levas, resortes, eléctricos o mecánicos.

En este caso se utiliza un sistema de potencia eléctrica de tal manera que permite reducir costo y facilitar su uso, teniendo una variedad de componentes fácilmente accesibles en el mercado.

Los competentes para este sistema de control principalmente son un motor DC y su respectivo controlador, para poder facilitar el control del motor se cuenta con un encoder de esta forma podemos saber la posición del eje del motor y un microcontrolador encargado de procesar los datos y enviar los parámetros de control del motor.

3.7.3 Sistema de rodamientos lineales

Provee una superficie de deslizamiento para la plataforma del simulador, facilita el movimiento con baja fricción.

Los rodamientos lineales son elementos de rodadura para movimientos de traslación. Igual que en el caso de los rodamientos rotativos, se distingue si las fuerzas que se producen son transmitidas por elementos rotativos o por elementos de fricción.

Están compuestos por una jaula de plástico que se combina con segmentos de rodadura de acero endurecido para el guiado de conjuntos de bolas.

Existen diferentes tipos de rodamientos dependiendo de su aplicación ofrecen precisión y velocidad o precisión y rigidez, en el caso del simulador se busca precisión y rigidez [5].

3.7.4 Mecanismo husillo y guías

Este mecanismo se compone de un husillo el cual gira sobre su eje por la acción de un motor, el cual ambos se encuentran unidos mediante acoplamiento. El giro del tornillo desplaza la tuerca y esta a su vez desplaza la plataforma la cual desliza sobre unos rodamientos lineales. La tuerca se encuentra unida a la plataforma por medio de una camisa.

Un husillo es un tipo de tornillo largo y de diferentes diámetros, utilizado para accionar dispositivos de apriete, así como prensas o mordazas, así como también para producir desplazamiento lineal [7]. Un husillo tradicional está compuesto por un eje de acero con rosca trapezoidal y una tuerca de bronce, metal o plástico.

3.7.5 Instrumentación para medición

Se montan sobre la plataforma del simulador algunos sensores, con el fin de medir variables durante la ejecución del sistema, principalmente se montan acelerómetros. Para este tipo de simulador es posible montar sensores para medir la distancia recorrida por la plataforma y celdas de carga para medir la fuerza que la plataforma ejerce sobre la estructura de prueba.

Capítulo III: Diseño del hardware y software.

En este capítulo se presenta la metodología desarrollada para obtener el diseño y la construcción del simulador y los pasos seguidos para lograr el objetivo de hacer entrar en resonancia estructuras a escala.

4.1 Diagrama conceptual del simulador

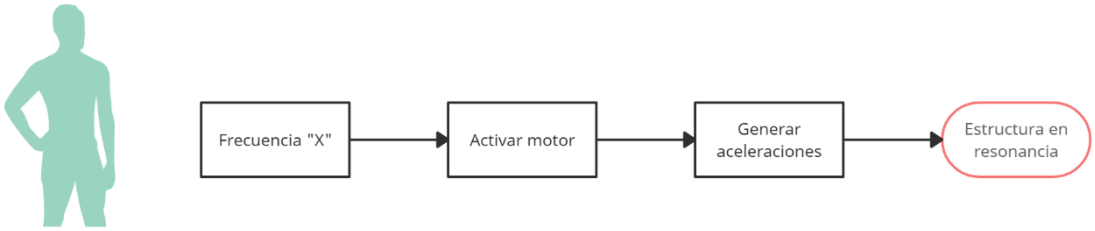


FIGURA 4 Diagrama del simulador [Elaboración propia]

A grandes rasgos el comportamiento del simulador se muestra en la FIGURA 4. El dispositivo funciona a una frecuencia preestablecida en la cual una de las estructuras entrará en resonancia, al presionar uno de los botones se cambiará la frecuencia de funcionamiento del dispositivo, provocando que otra de las estructuras entre en resonancia.

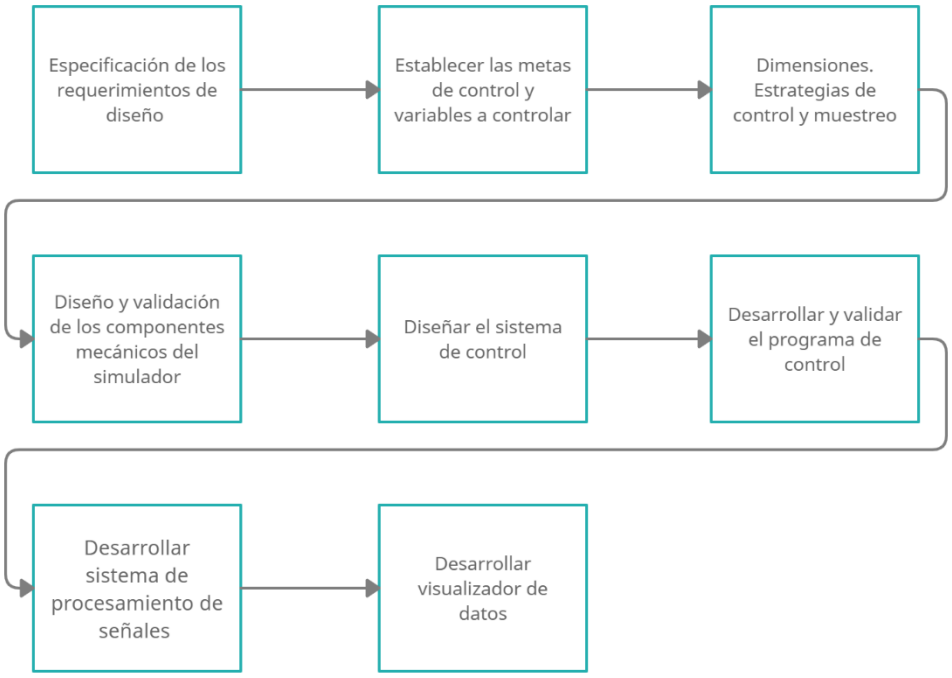


FIGURA 5 Pasos de diseño [Elaboración propia]

Para obtener un resultado favorable en el diseño y construcción del simulador se tiene que seguir una serie de pasos que permiten validar y corregir cada uno de los sistemas necesarios del simulador. En la FIGURA 5 se observa cada uno estos pasos a grandes rasgos, se tiene primero una idea la de demostrar el fenómeno de resonancia para lo cual se plantea una solución la de diseñar y construir un dispositivo capaz de hacer entrar en resonancia estructuras y con el cual se pueden desarrollar una serie de experimentos para el análisis del fenómeno.

4.2 Diseño de componentes del hardware

El mecanismo de movimiento para la implementación presenta múltiples opciones de las cuales se evalúan las más sencillas y factibles de implementar de tal forma que pueda ser barato y repetible, en consideración se tiene la necesidad de reducir ruidos externos, es decir, posibles golpes o fricción entre las piezas móviles.

Se presenta el sistema de husillo y guías lineales, explicando las partes que lo componen y sus ventajas, para posteriormente presentar los resultados obtenidos. Este sistema es ampliamente utilizado en la industria para maquinas que necesitan desplazarse en un eje con precisión y sin vibraciones. Por lo tanto, este sistema presenta ventajas para ser utilizado en el desarrollo del simulador.

4.2.1 Especificaciones técnicas

La base del diseño lo hacen las especificaciones deseadas del simulador, los movimientos a realizar, la capacidad de carga y su desplazamiento, velocidad y aceleración máximas. Las características de la Tabla 1 le permiten a este simulador realizar ensayos de vibración periódica a estructuras pequeñas y fueron definidas para estar en un rango bajo cercanas entre sí para observar cómo varia la frecuencia de resonancia con relación a la altura de las estructuras.

Tabla 1 Especificaciones técnicas del simulador [Elaboración propia]

Especificaciones	**Valor**	**Unidades**
Dimensiones de la base	280 x 216	cm
Máxima capacidad de carga	200	g
Recorrido máximo	10	cm
Ancho de banda	15	Hz
Aceleración máxima	1	g’s

4.2.2 Sistema de rodamiento lineal

La selección del sistema de rodamiento lineal para el simulador se fundamenta en un sistema que supere los requerimientos de carga y desplazamiento del simulador, con muy baja fricción y que soporte cambios bruscos de velocidad. Las guías lineales de perfil redondo resultaron ser una buena opción, principalmente por su bajo costo, recorrido suave, bajo valor del coeficiente de fricción. Dado lo anterior, se selecciona un par de guías lineales de perfil redondo con dos cojinetes en cada guía. Cada cojinete viene con un sistema de lubricación para producir la mínima fricción entre la guía lineal y el cojinete.

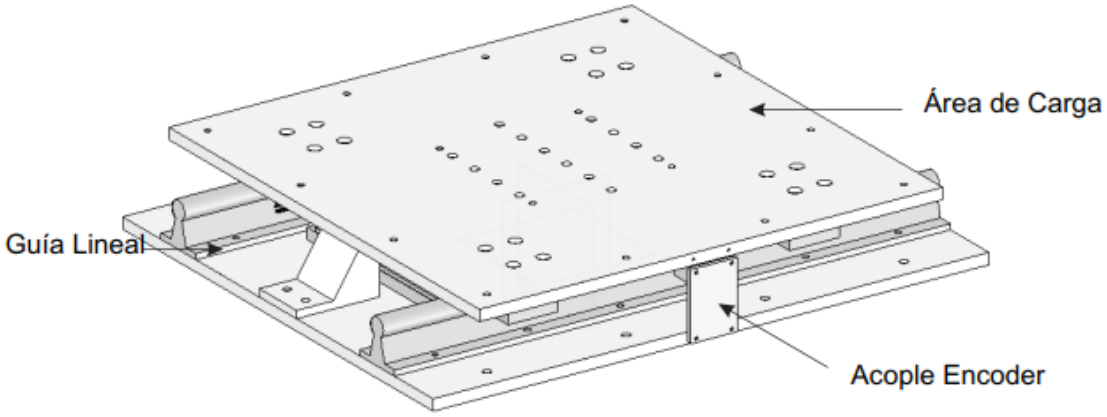


FIGURA 6 Sistema de rodamientos lineales [4]

4.2.3 Mecanismo husillo y guía

Como se describió anteriormente este mecanismo se encarga de transmitir el movimiento del motor hacia la plataforma de la mesa vibratoria, las guías sirven para mantener estable la plataforma durante el recorrido, de este sistema depende en gran parte el surgimiento de fricciones que generan ruido en las mediciones de aceleración.

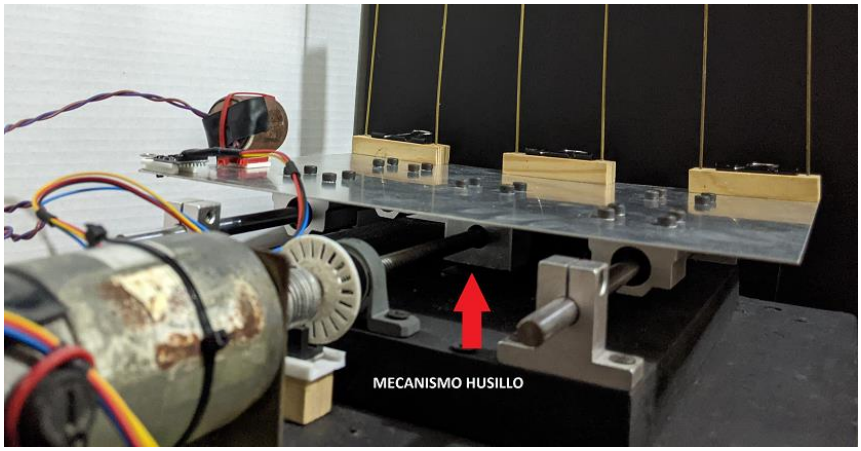


FIGURA 7 Mecanismo husillo y guías [Elaboración propia]

Para la implementación de este sistema en el simulador, los componentes fueros adquiridos en línea, se compraron en Amazon ya que ofrecen un resultado óptimo para la aplicación en la mesa vibratoria, en el mercado nacional se dificulto la obtención de este tipo de componentes, razón por la cual se adquieren fuera del país.

4.2.4 Área de carga, base y acoples

Con la especificación de las guías lineales, los requerimientos del área de carga (ver Tabla 1) y las dimensiones, se diseña la pieza donde se van a instalar las estructuras de prueba del simulador sísmico. el material seleccionado es aluminio por ser un material liviano y de rigidez suficiente para esta aplicación.

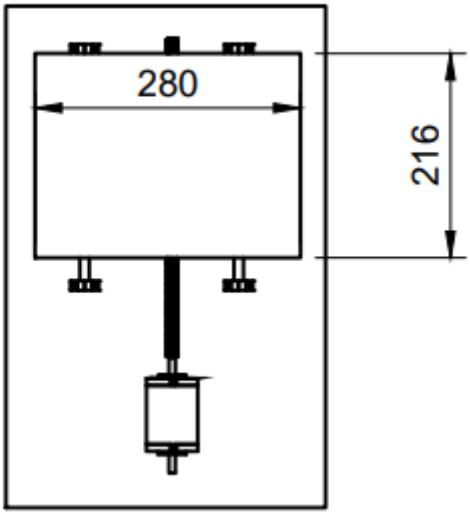


FIGURA 8 Dimensiones del área de carga en mm [Elaboración propia]

Una vez cortadas las placas de la mesa vibratoria, es necesaria una forma de acople para instalar el encoder de forma que sea posible tener el valor de giro del motor.

Una vez identificadas todas las piezas del simulador, se buscan en el mercado local y en tiendas en línea, luego se procede a validar el diseño mecánico armando cada una de las piezas con el fin de comprobar que ningún elemento esté en conflicto con otro o si existe alguna colisión entre piezas. luego de una detallada verificación del conjunto de la mesa vibratoria se confirma que no existen conflictos, agujeros errados o colisiones en el diseño. La FIGURA 9 muestra el diseño del simulador sísmico y sus partes internas.

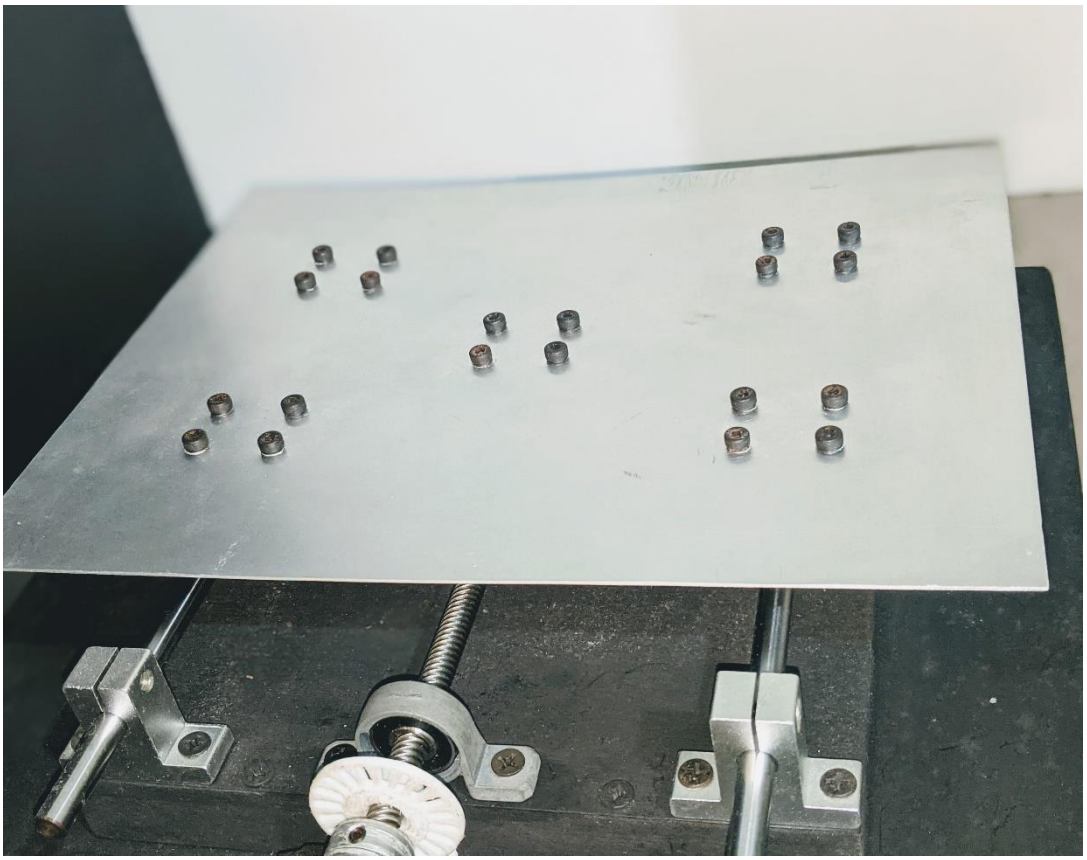


FIGURA 9 Fotografía del simulador [Elaboración propia]

4.2.5 Método de sujeción de estructuras.

Para colocar las estructuras a escala en la base del simulador es necesario considerar que la energía del movimiento de las estructuras puede afectar el efecto buscado en las estructuras, por esta razón el dispositivo de sujeción debe ser capaz de mantenerlas sin afectar este movimiento. Es decir mantener la fijeza sin importar la fuerza que pueda ser ejercida por el simulador, si forma de sujetar las estructuras no es lo suficiente podría actuar como un amortiguador al momento de entrar en resonancia si es que se consigue.

Se considera que debe ser de fácil montaje y desmontaje permitiendo tener la facilidad de cambiar lo que se desea colocar sobre el simulador, aparte es necesario que sea de bajo costo.

Para esto se utilizan clips de presión como el de la figura 10 siendo de fácil acceso y de bajo costo, además se obtienen los resultados deseados en las pruebas realizadas, el método de sujeción de sensores es el mismo, teniendo resultados bastante favorables.

Se evaluaron otros métodos de sujeción siendo este el de mejor resultado y menor costo, se evalúa la impresión de soportes en 3D siendo menos accesible, por esta razón se decide utilizar los clips.

FIGURA 10 Clip de presión

4.3 Diseño del sistema de control, Software

El control del simulador se implementa utilizando un microcontrolador ESP32, este microcontrolador cuenta con la característica de tener doble núcleo lo que permite realizar dos tareas en simultáneo. La primera tarea que realiza es el control del motor que a su vez transmite el movimiento al dispositivo, la segunda tarea se encarga de leer los datos de los acelerómetros y enviarlos utilizando el protocolo serial para su posterior análisis y presentación.

4.4 Componentes del control.

Los componentes del sistema de control son: un microcontrolador, un driver para el motor DC, un sensor óptico con un disco ranurado (encoder) y un motor DC. En la FIGURA 11 se observa como cada componente se integra en el sistema, esto se explicará más adelante.

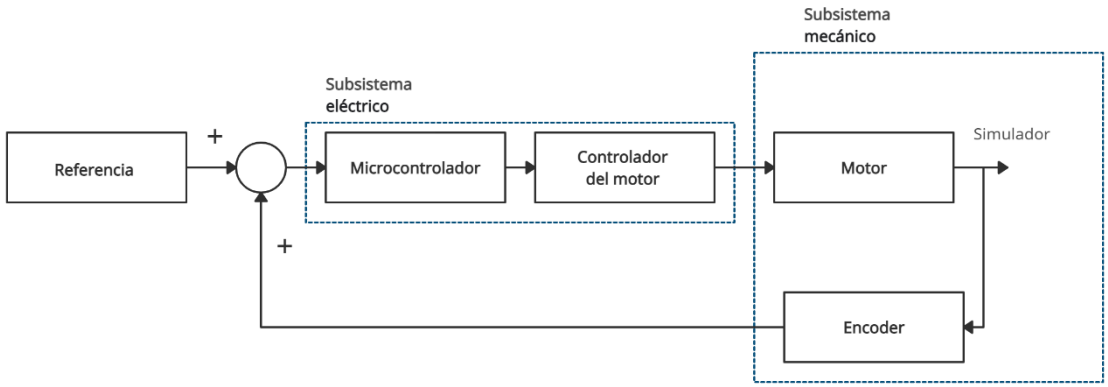


FIGURA 11 Diagrama del control del simulador [Elaboración propia]

4.4.1 Microcontrolador

La función del microcontrolador, en el programa de control, es de comparar las mediciones del encoder, y por medio de estas mediciones se obtiene un valor de desplazamiento del motor que es necesario controlar para invertir el giro del motor a una frecuencia determinada.

El microcontrolador ejecuta un programa para controlar el giro del motor, el control se logra utilizando un driver para el motor DC encargado de convertir las señales del microcontrolador en señal de voltaje para poner en funcionamiento el motor y de esta forma controlar el valor de frecuencia a la que se encuentra trabajando la mesa.

4.4.2 Controlador del motor DC

Se encarga de convertir las señales del microcontrolador a un voltaje aceptable para hacer funcionar el motor,

El sensor se encarga de retroalimentar el sistema para poder obtener el valor de desplazamiento del motor que puede traducirse al número de ranuras del disco.

El driver del motor es el encargado de traducir la información del microcontrolador y enviarle el valor de desplazamiento al motor, con esto se logra hacer girar el motor.

El motor DC es el encargado de producir las aceleraciones en las estructuras colocadas sobre el simulador.

4.4.3 Encoder rotativo

Este sensor se encarga de convertir el giro del motor en un valor manipulable para el programa y de esta forma controlar el giro del motor, debido a la dificultad de acceso en el mercado local se propone utilizar un sensor óptico con un disco ranurado permitiendo conseguir la misma función con un costo mucho menor.

En la FIGURA 12 se aprecia la implementación de este sensor, se trata de un sensor óptico formado por un fotodiodo y un fototransistor, la idea es que cada vez que se interrumpe el haz de luz entre el fotodiodo y el fototransistor se genera un pulso que el microcontrolador almacena para saber la posición del eje del motor, para generar las interrupciones de manera repetida y que pueda ser fácilmente interpretable, se usa un disco ranurado con 20 espacios es decir, se necesitan 20 cuentas para un giro completo del motor.

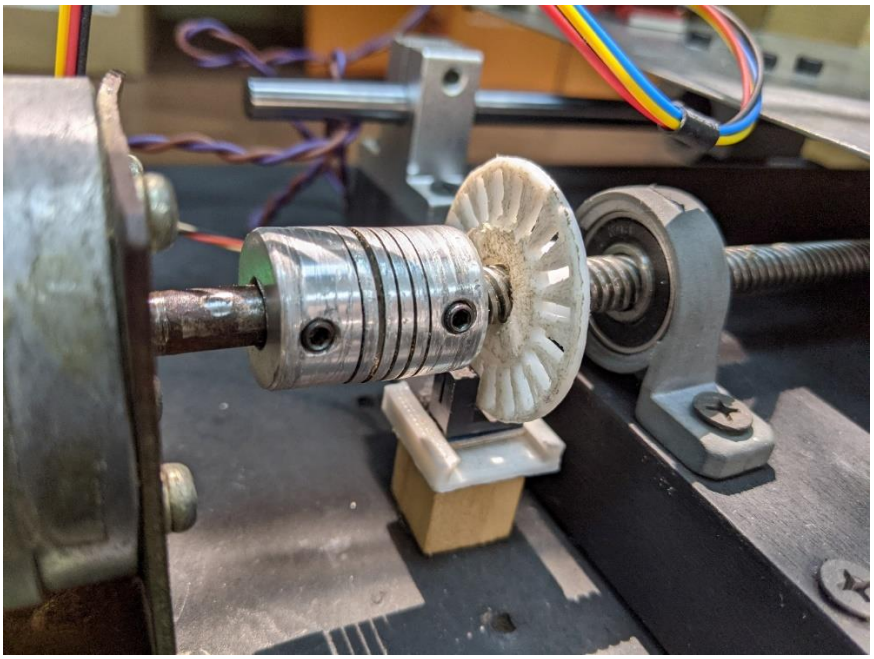


FIGURA 12 Encoder implementado en el simulador [Elaboración propia]

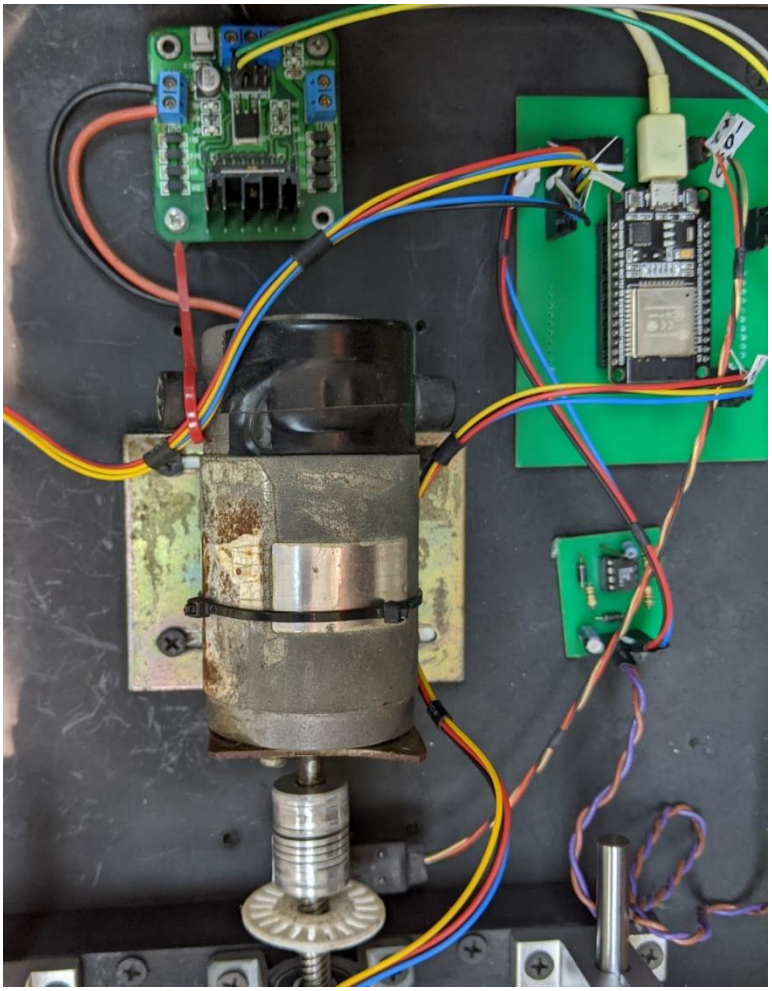


FIGURA 13 Componentes del control [Elaboración propia]

4.4.1 Motor DC

El motor DC se encarga de generar el movimiento de la mesa, se trata de un motor de 91 VDC, que se tenía con fácil acceso, es decir, se consigue usado, este componente puede ser sustituido por un motor DC de 12 o 24 VDC, se tiene que tener en cuenta que el acople del motor con el eje de la mesa es de 5 mm, por lo tanto, el eje del motor debe ser se esté diámetro.

En el simulador, el motor se encarga de generar el movimiento y el efecto de resonancia en las estructuras, en la FIGURA 13 se muestra el motor utilizado junto con los sistemas de control mencionados anteriormente.

4.5 Parámetros para el control del dispositivo

La implementación del control se basa en un lazo cerrado con los componentes antes mencionados, el microcontrolador se encarga de comparar los datos obtenidos de encoder elemento encargado de la retroalimentación.

El programa es un ciclo infinito principalmente encargado de comparar el número de giros que el motor ejecuta, este valor se obtiene del encoder por medio de una interrupción que el microcontrolador se encarga de atender. Como ejemplo se muestra el diagrama de flujo para el numero de cuentas igual a 10 este valor equivale a una frecuencia de aproximadamente 7.5 Hz.

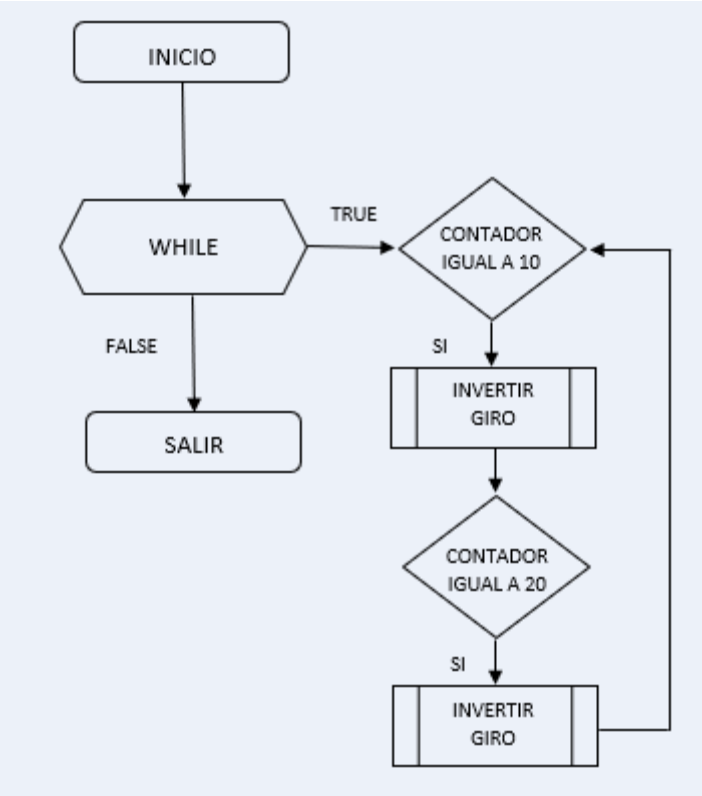


FIGURA 14. Diagrama de flujo para 10 cuentas. [Elaboración propia]

El valor de 10 equivale a que el motor de media vuelta, luego de contar a 10 se invierte el giro se espera a que el valor de cuentas sea el doble para llegar al valor de 10 cuentas en el sentido inverso generando así una onda senoidal aproximada que se puede observar en la gráfica de las aceleraciones obtenidas del geófono.

En el diagrama de la FIGURA 14 se observa la inicialización del microcontrolador es decir inicializar puertos de entra y salida, puertos de comunicación en este caso serial e I2C para las lecturas de los acelerómetros. Luego se corren en paralelo los programas para controlar el motor y tomar las capturas de los datos cada uno en su respectivo núcleo, estos programas son bastantes simples, pero pueden tener sus inconvenientes cuando se tiene que atender una interrupción, en general es desempeño de este microcontrolador es el adecuado para el dispositivo.

4.5.1 Sistema de lazo cerrado

Los sistemas de control realimentados se denominan también sistemas de control en lazo cerrado. En la práctica, los términos control realimentado y control en lazo cerrado se usan indistintamente. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación (que puede ser la propia señal de salida o una función de la señal de salida y sus derivadas y/o integrales), con el fin de reducir el error y llevar la salida del sistema a un valor deseado. El término control en lazo cerrado siempre implica el uso de una acción de control realimentado para reducir el error del sistema [10].

4.5.2 Procedimientos de construcción y desarrollo del dispositivo.

En la aproximación de prueba y error para el diseño de un sistema, se parte de un modelo matemático del sistema de control y se ajustan los parámetros de un compensador. La parte de este proceso que requiere más tiempo es la verificación del comportamiento del sistema mediante pruebas, después de cada ajuste de los parámetros. Una vez obtenido un modelo satisfactorio, el desarrollador debe construir un prototipo y probar el sistema. Si se asegura la estabilidad absoluta en lazo abierto, el desarrollador cierra el lazo y prueba el comportamiento del sistema en lazo cerrado. Debido a los efectos de carga no considerados entre los componentes, la falta de linealidad, los parámetros distribuidos, etc., que no se han tenido en cuenta en el diseño original, es probable que el comportamiento real del prototipo del sistema difiera de las predicciones teóricas. Por tanto, tal vez el primer diseño no satisfaga todos los requisitos de comportamiento. Mediante el método de prueba y error, el desarrollador debe cambiar el prototipo hasta que el sistema cumpla las especificaciones. Debe analizar cada prueba e incorporar los resultados de este análisis en la prueba siguiente. El desarrollador debe conseguir que el sistema final cumpla las especificaciones de comportamiento y, al mismo tiempo, sea fiable y económico [11].

4.5.3 Descripción de la implementación.

El sistema de control está basado en un sistema de lazo cerrado, el cual se encarga de controlar el motor, para su implementación se obtiene un valor de desplazamiento del eje del motor para esto se cuenta con un sensor capaz de registrar este dato, se trata de un sensor óptico y un disco ranurado el cual divide el valor de una vuelta entre 20. Es necesario registrar este valor de cuentas para esto se cuenta con una interrupción en el programa del microcontrolador, teniendo el número de veces que ha girado el eje se puede calcular el desplazamiento de la mesa en la que se encuentran las estructuras.

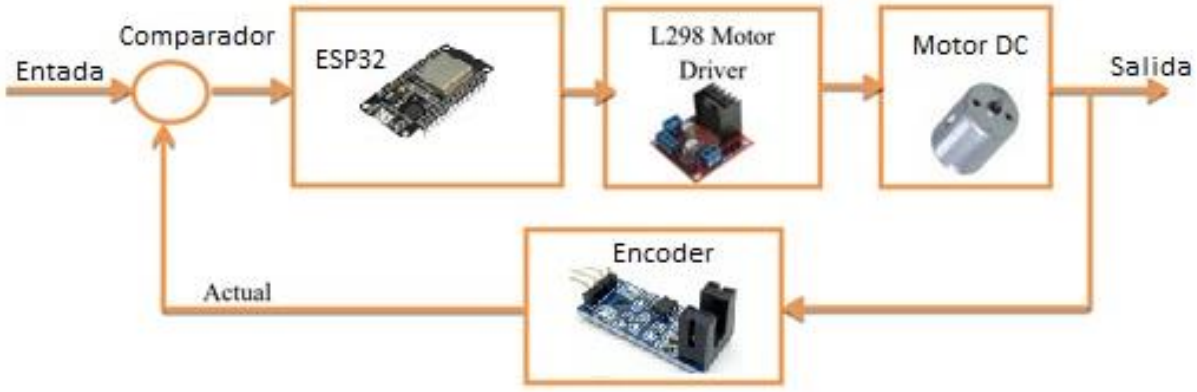


FIGURA 15 Control de lazo cerrado implementado. [Elaboración propia]

El control es un programa sencillo el cual se encarga de contar el número de giros en una dirección, luego invertir el giro y contar nuevamente, repitiendo esta acción hasta que se cambie la frecuencia del conteo.

Para obtener el sistema de lazo cerrado se tiene como variable de entrada el número de cuentas requeridas que equivale a una frecuencia específica, como comparador se tiene el programa que el microcontrolador ejecuta, como elemento de control se tiene el motor que es controlado por medio de un driver por el microcontrolador, como elemento de retro alimentación se tiene un sensor encargado de transmitir el número de veces que el eje del motor pasa por este, como salida se tiene el movimiento de la mesa que somete a aceleraciones las estructuras a escala.

4.5.4 Selección del microcontrolador

El microcontrolador ESP32 del fabricante chino ESPRESSIF que cuenta con dos núcleos Xtensa single-/dual-core 32-bit LX6 [11], es seleccionado para ejecutar el control y medición del simulador, la selección de este microcontrolador se debe a que posee dos núcleos independientes, realizando así cada tarea sin inconvenientes.

Aparte de poseer dos núcleos estos pueden operar a una velocidad de hasta 240MHz permitiendo una operación del dispositivo adecuada. Para el simulador se tiene el microcontrolador integrado en una placa de desarrollo es decir cuenta con algunos periféricos para comunicación USB integrados en un circuito para el desarrollo de aplicaciones en este caso se trata del DEV kit. 1 FIGURA 16.

El sistema operativo ESP-IDF admite la asignación de tareas a los núcleos, lo que significa que se asigna uno de los núcleos para ejecutar una tarea concreta. También admite tareas "sin afinidad", lo que significa que la tarea puede ejecutarse en cualquier núcleo.

FIGURA 16. ESP 32 DEV kit 1.

Los núcleos del ESP32 se denominan "Core 0" y "Core 1". El núcleo 0 se conoce como "núcleo de protocolo" o "CPU PRO". En las aplicaciones predeterminadas del ESP32, las tareas relacionadas con el protocolo, como el WiFi y el Bluetooth, se asignan a este núcleo. El núcleo 1 se conoce como "Núcleo de aplicación" o "CPU APP", que se encarga de ejecutar las aplicaciones de usuario.

Cada núcleo mantiene su propia lista de interrupciones, temporizadores y, si se ejecuta ESP-IDF. La RAM es compartida entre los núcleos, pero una sección de la RAM se reserva para la caché del núcleo 0, y otra sección se reserva para la caché del núcleo.

Cada vez que se ejecuta el programador en cualquiera de los núcleos, mira una lista compartida de tareas para ver qué tarea está en el estado listo. Elegirá cualquier tarea que esté lista y tenga la mayor prioridad. Sin embargo, sólo ejecutará una tarea si el campo xCoreID para esa tarea es "sin afinidad" o coincide con el número de núcleo del procesador que llama.

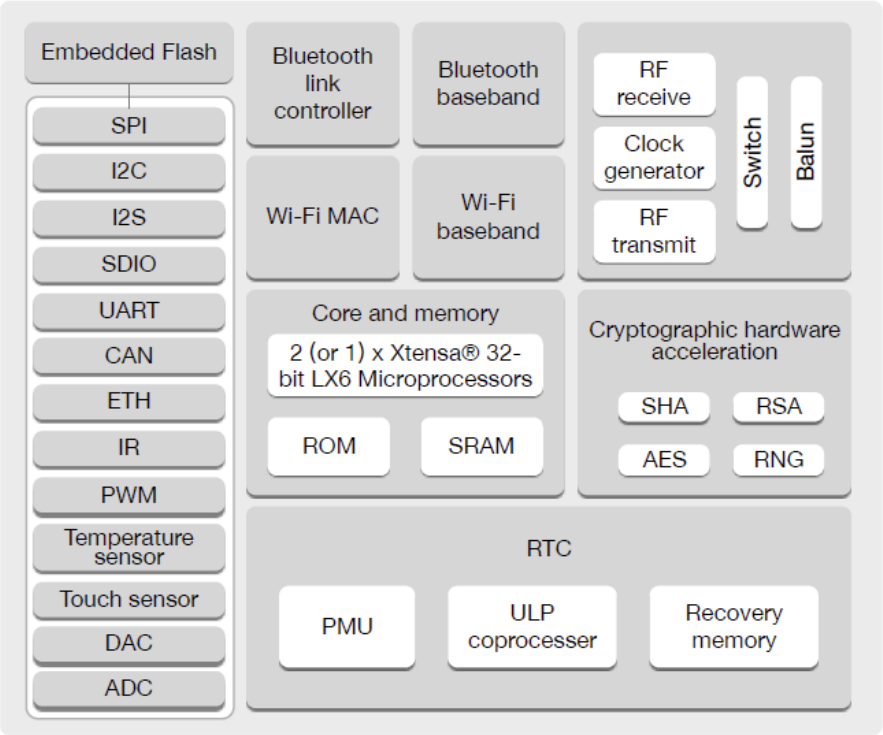


FIGURA 17 Diagrama de bloques ESP32 [11].

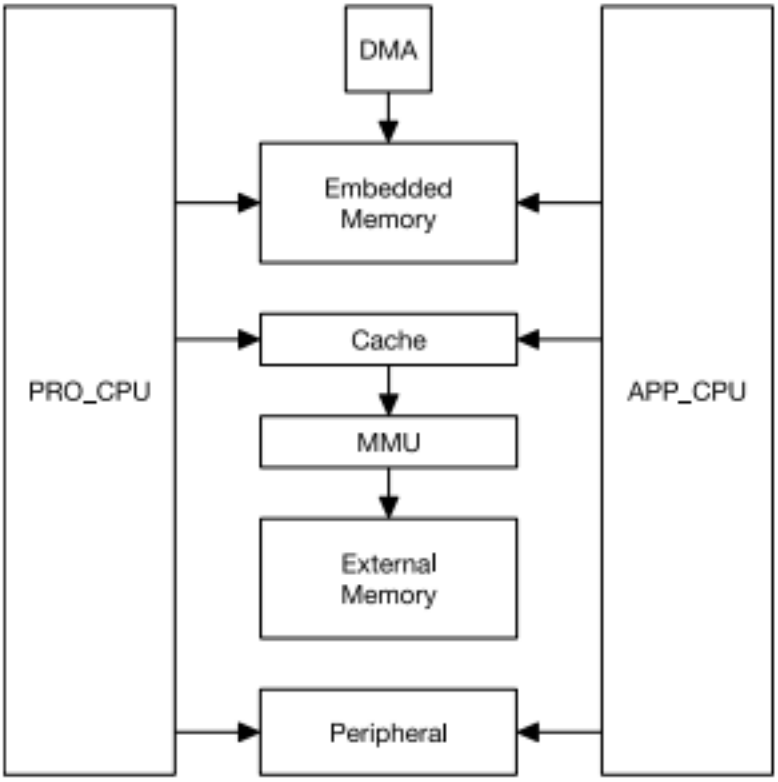


FIGURA 18 Estructura del ESP32 [11]

4.6 Diseño del sistema de adquisición y análisis de datos (medición)

La medición se obtiene utilizando dos clases de acelerómetros, un acelerómetro y un geófono, las capturas de aceleración son obtenidas utilizando uno de los núcleos del microcontrolador, es decir, el control es independiente de la medición. La medición de la aceleración tiene como propósito conocer la frecuencia a la que están sometidas las estructuras del simulador.

Para la captura de datos del geófono se utiliza el conversor análogo digital del microcontrolador, para el acelerómetro se utiliza el protocolo i2c que trae el mismo, luego de tener estos datos son obtenidos por el microcontrolador y encapsulados para su fácil trasmisión y obtener la transformada rápida de Fourier.

La razón principal de tener ambos acelerómetros es para tener una comparación y verificar los datos obtenidos, además el geófono sirve para calibrar el sistema ya que al medir su salida con un osciloscopio es posible obtener la frecuencia de las aceleraciones, al comparar esta medición con los valores calculados en Python se pueden corroborar los datos obtenidos.

4.6.1 Geófono electromagnético.

Consiste en un embobinado que se mueve dentro de un campo magnético suspendida sobre un sistema de masa resorte.

Principio de un geofono vertical suspendido de un resorte. Esta masa permanece en reposo hasta el momento que la base se empieza a mover con periodos inferiores al del propio sistema.

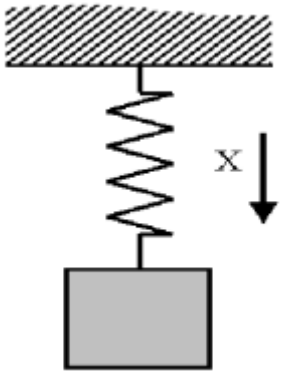


FIGURA 19 Sistema masa resorte. [1]

Para el simulador se utiliza el geófono Mark L28 que se muestra en la FIGURA 20, el geófono se utiliza junto con el microcontrolador y un amplificador de instrumentación, siendo la manera más sencilla de utilizarlo debido a que es analógico, en el microcontrolador únicamente se necesita leer un valor entre 1.5 V y – 1.5V. Esto se logra utilizando el amplificador de instrumentación ver FIGURA 23.

Tabla 2 Características del geófono Mark L28

Frecuencia propia	4.5 Hz
Resistencia de la bobina	410 Ohm
Amortiguamiento de circuito abierto	0.7
Constante de transducción G	0.304 V/(cm/s)
Masa en gramos	23g

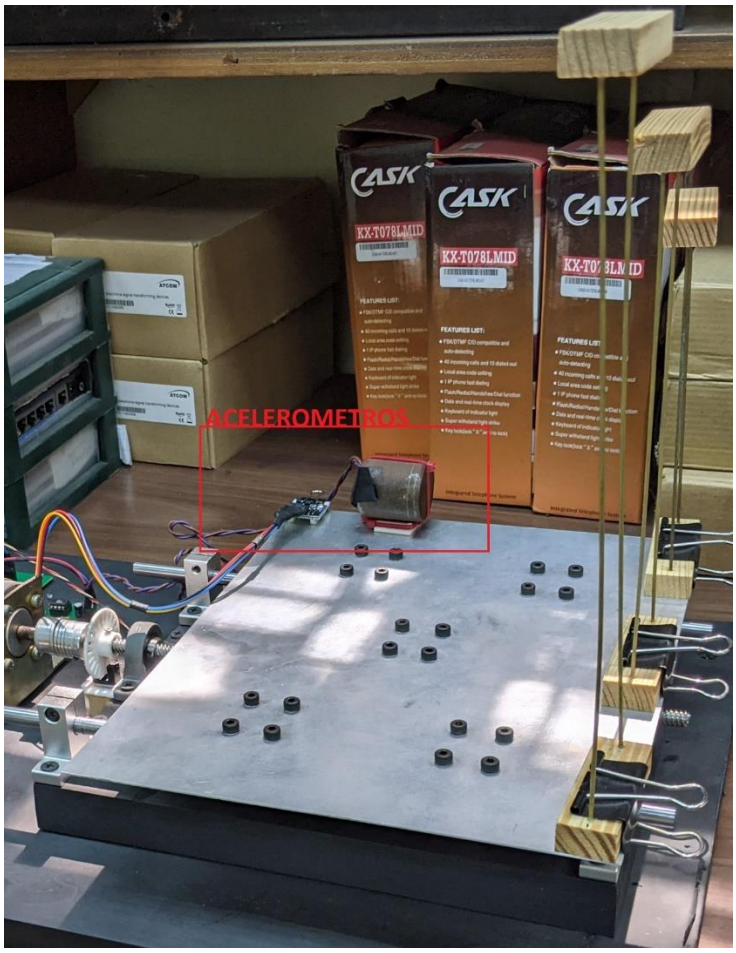


FIGURA 20 Acelerómetros en el simulador. [Elaboración propia]

4.6.2 Equivalente eléctrico del geófono electromagnético.

Un geófono electromagnético se puede representar mediante un circuito analógico utilizando componentes pasivos (R L C) y una fuente de tensión para representar el movimiento del suelo. Con ella se encuentra un circuito equivalente del geófono en el sentido de que la ecuación que lo describe es formalmente idéntica a la de éste, cambiando las variables. En el circuito propuesto C1 representa la masa móvil, L1 equivale a la constante de recuperación de la suspensión, R1 el amortiguamiento en circuito abierto, R2 la resistencia eléctrica de la bobina, L2 la inductancia eléctrica propia de la bobina y R3 la resistencia externa de amortiguamiento.

Este modelo es muy útil para diseñar el circuito del preamplificador, ya que se puede integrar sin dificultad en las técnicas habituales de diseño electrónico (ver FIGURA 21).

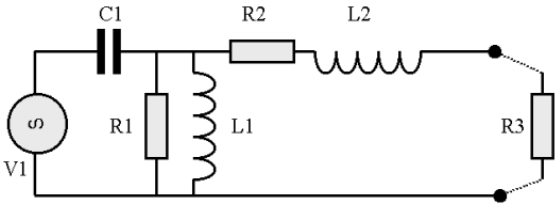


FIGURA 21 Circuito equivalente de un geófono electromagnético.

En general, el fabricante nos suministras la información básica del sensor, tal como su frecuencia propia, amortiguamiento en circuito abierto, constante de transducción y masa móvil. En general también nos proporcionan el valor de la resistencia y de la autoinducción de la bobina sensor R2 y L2. En caso contrario se puede medir directamente bloqueando la masa móvil y utilizando la instrumentación habitual de un laboratorio electrónico. Ver tabla 2.

4.6.3 Acoplamiento del Geófono Mark L28 de 4.5 Hz

El Geófono como sensor de aceleraciones no necesita circuito de polarización, pues el mismo genera una diferencia de potencial entre sus dos terminales al detectar vibraciones de la superficie.

El acoplamiento del Geófono con el ADC interno del microcontrolador se realiza mediante un filtro pasivo RC de primer orden paso bajo, que tiene una frecuencia de corte del orden de los 10 Hz y una etapa posterior de amplificación con ganancia 10 construida con un AD623 como el de la FIGURA 22.

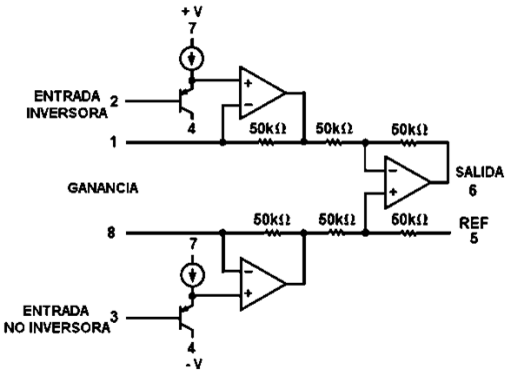


Figura 22 Diagrama Interno del OP-AMP AD623. [13]

El circuito de acoplamiento del geófono con el canal analógico de entrada del microcontrolador se muestra en la FIGURA 23. El cálculo de los valores de la resistencia (R) y capacitor (C) se realiza utilizando la fórmula para un filtro RC pasa bajo.

$$f_c = \frac{1}{2\pi RC}$$

Si consideramos la frecuencia de corte igual a 10 Hz y un capacitor de 1µF, el valor de la resistencia corresponde a:

$$R = \frac{1}{2\pi f_c C}$$

$$R = \frac{1}{2\pi * 10 * 1x10^{-6}}$$

$$R = 11\ k\Omega$$

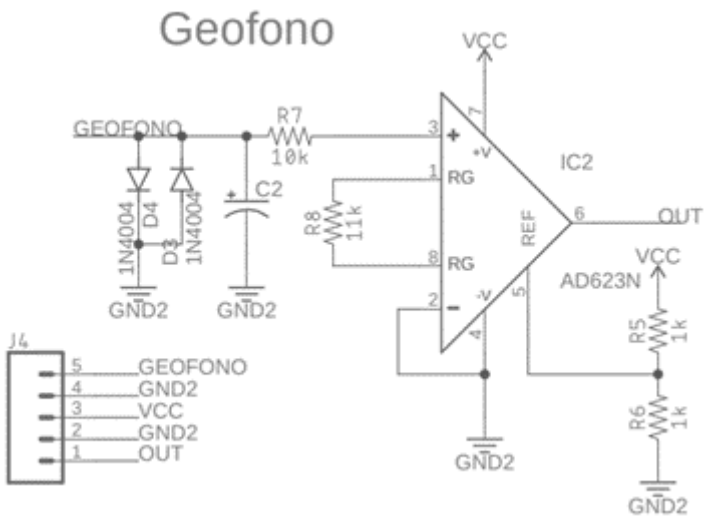


FIGURA 23 Amplificador de instrumentación para geófono. [Elaboración propia]

4.6.4 Acelerómetro MPU5060

El MPU6050 es un sistema microelectromecánico (MEMS) que consta de un acelerómetro de 3 ejes y un giroscopio de 3 ejes en su interior. Esto nos ayuda a medir la aceleración, la velocidad, la orientación, el desplazamiento y muchos otros parámetros relacionados con el movimiento de un sistema u objeto. Este módulo también tiene un procesador de movimiento digital (DMP) en su interior que es lo suficientemente potente como para realizar cálculos complejos y así liberar el trabajo del microcontrolador.

El hardware del módulo es muy simple, en realidad consta del MPU6050 como los componentes principales, como se muestra arriba. Dado que el módulo funciona con 3,3 V, también se utiliza un regulador de voltaje. Las líneas I2C se elevan usando una resistencia de 4.7k y el pin de interrupción se baja usando otra resistencia de 4.7k.

El módulo MPU6050 nos permite leer datos del mismo a través del bus I2C. Cualquier cambio en el movimiento se reflejará en el sistema mecánico que a su vez variará el voltaje. Luego, el IC tiene un ADC de 16 bits que utiliza para leer con precisión estos cambios de voltaje y lo almacena en el búfer FIFO y hace que el pin INT (interrupción) suba.

Esto significa que los datos están listos para ser leídos, por lo que usamos una MCU para leer los datos de este búfer FIFO a través de la comunicación I2C. Por fácil que parezca, es posible que se enfrente a algún problema al tratar de dar sentido a los datos. Sin embargo, hay muchas plataformas como Arduino con las que puede comenzar a usar este módulo en poco tiempo utilizando las bibliotecas fácilmente disponibles que se explican a continuación.[15]

La comunicación con el microcontrolador se lleva a cabo utilizando el bus I2C FIGURA 24, únicamente se utilizan dos líneas para comunicarse, para programar el microcontrolador se utiliza una librería construida para comunicarse con el acelerómetro y la implementación del programa es fácil utilizando esta librería.

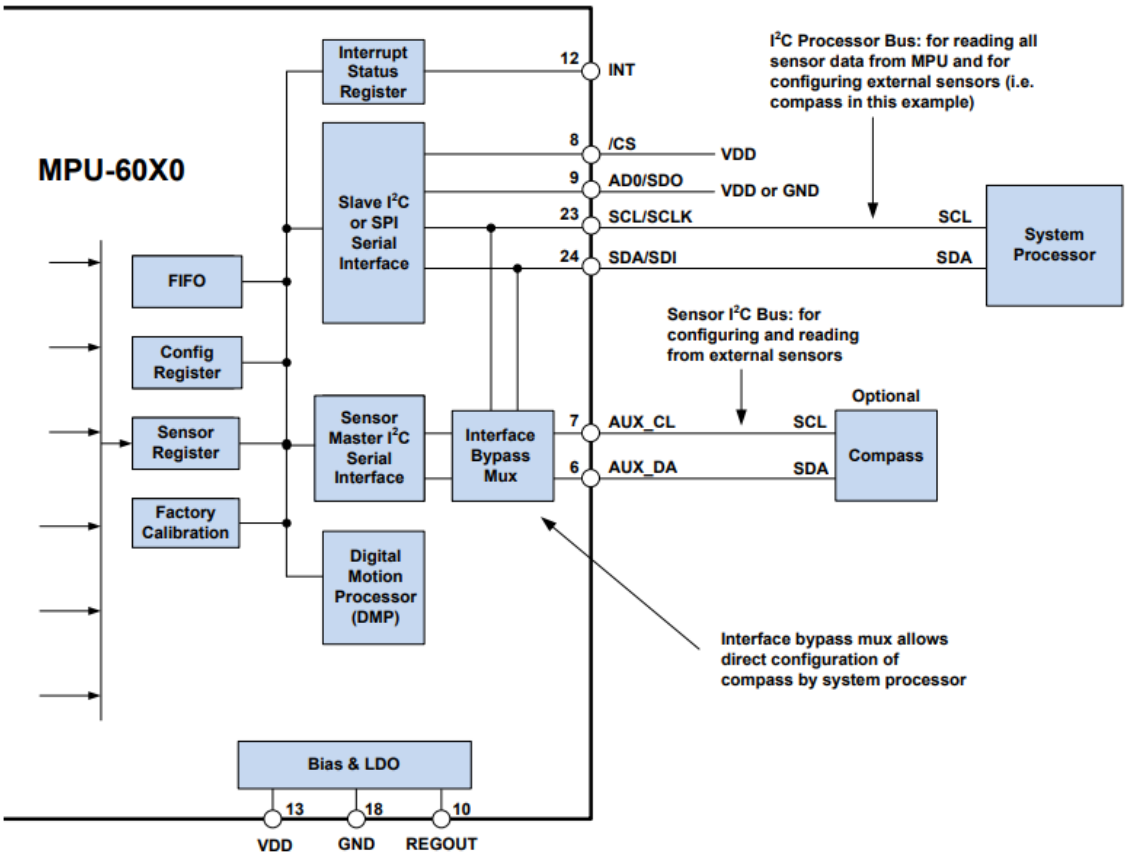


FIGURA 24 Comunicación del acelerómetro con un microcontrolador. [14]

4.6.5 Descripción general del flujo de datos del programa de medición

Los datos capturados por los sensores (acelerómetro) son recolectados por el microcontrolador utilizando el protocolo serial para el acelerómetro y el conversor análogo digital ya que el geófono proporciona un voltaje equivalente a la aceleración a la que el sensor está sometido, una vez lo datos están disponibles son enviados por USB para su análisis utilizando un script implementado en Python encargado de presentar los datos y calcular la FFT para obtener la frecuencia a la que el sistema se encuentra funcionando.

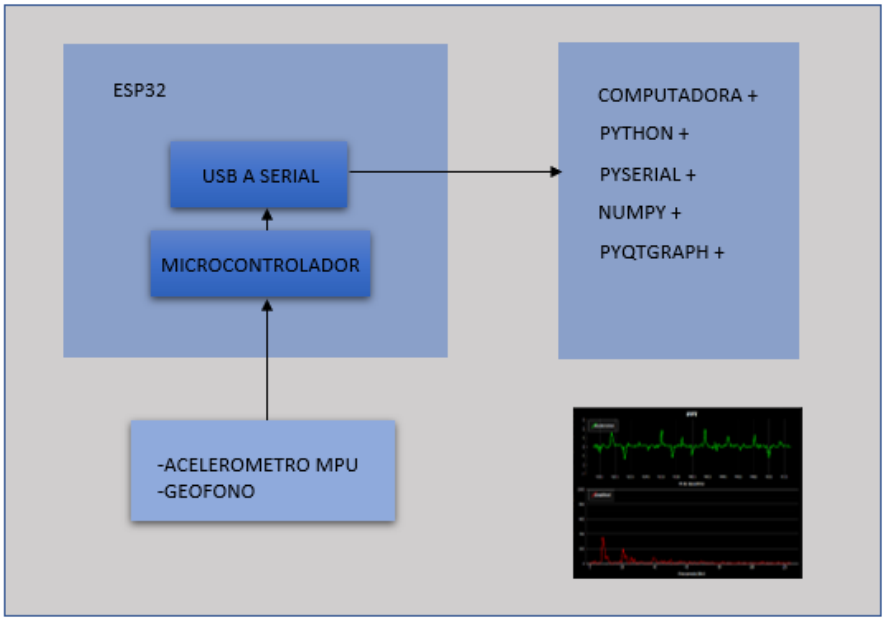


FIGURA 25 Flujo de datos del dispositivo [Elaboración propia]

En la FIGURA 25 se muestra el flujo de datos y los resultados obtenidos por el script, el programa principalmente se encarga de obtener las lecturas de aceleraciones y transmitirlas para esto se implementa una función encargada de transmitir estos datos de manera que puedan ser fácilmente recuperados en la computadora utilizando el script escrito en python.

Para la recuperación de los datos se implementa una clase en Python utilizando la librería Pyserial, esta es la encarga de establecer la comunicación con el microcontrolador, recibir los datos, desempaquetar los datos y hacerlo en segundo plano durante la ejecución de todo el script, esto permite tener las aceleraciones disponibles para graficar.

4.7 Diagrama de conexión y circuitos impresos

Para implementar el control y medición del dispositivo se diseña un circuito modular para permitir cambios a futuro y continuar el desarrollo del dispositivo. Para esto se tiene un circuito principal el cual tiene el microcontrolador y los pines correspondientes para cada módulo, es decir, sensores, drivers, botones.

Estos circuitos se diseñan en Eagle de Autodesk, luego se graban en una placa de circuito impreso (PCB), este proceso se sigue para los pulsadores y el circuito del geófono.

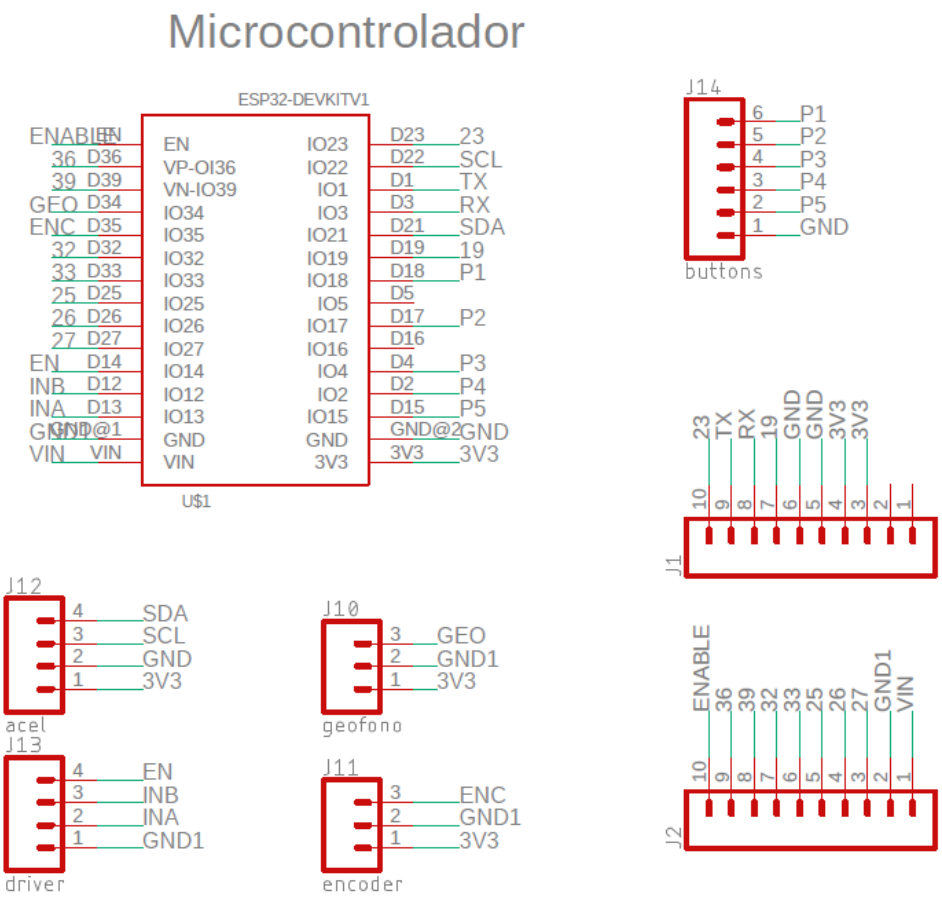


FIGURA 26 Diagrama de conexiones del dispositivo [Elaboración propia]

En la FIGURA 26 se muestra el microcontrolador con todos los puntos de conexión necesarios para conectar los módulos necesarios para el funcionamiento del dispositivo. Se tiene una bornera para el acelerómetro, una para el circuito del geófono, encoder, pulsadores y las dos restantes J1 y J2 son pines que pueden ser usados a futuro teniendo disponibles 12 pines de entras y salidas, que pueden ser configurados a conveniencia para mejoras futuras del dispositivo.

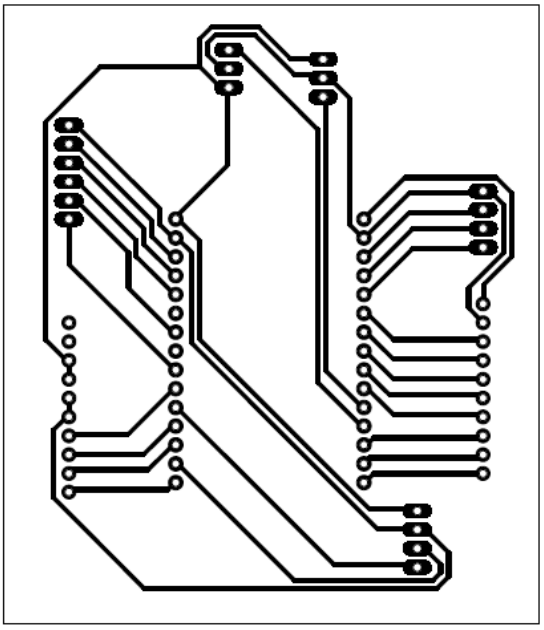


FIGURA 27 Pistas del circuito [Elaboración propia]

Este circuito impreso tiene la finalidad de evitar que los componentes se desconecten debido a las vibraciones generadas por el dispositivo, además integra de forma modular los componentes del simulador.

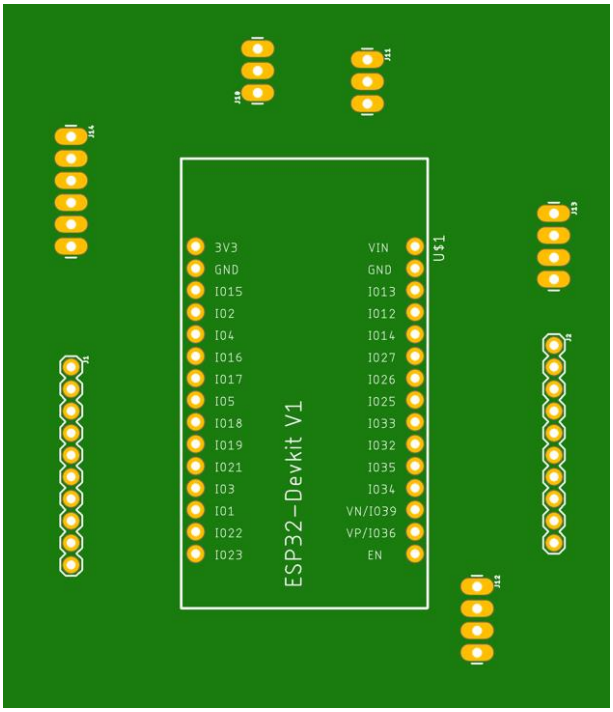


FIGURA 28 Disposición de pines [Elaboración propia]

Tabla 3 Configuraciones de pines [Elaboración propia]

Configuración de pines			
ESP32	Nombre	PCB	Función
13	InA1	INA	Controla puente A del driver
12	InB1	INB	Controla puente B del driver
14	enablePin	EN	Activa el motor
35	encodPinB1	ENC	Encoder
15	buttonPin	P1	Frecuencia 1
5	buttonPin1	P2	Frecuencia 2
4	buttonPin2	P3	Frecuencia 3
17	buttonPin3	P4	Aumenta frecuencia
18	buttonPin4	P5	Disminuye frecuencia
34	geoPin	GEO	Geófono
22	SCL	SCL	Acelerómetro
21	SDA	SDA	Acelerómetro

4.8 Diseño del experimento a realizar con el simulador

El análisis de vibraciones y la presencia de resonancia en una estructura son factores importantes en la enseñanza de la ingeniería, ya que pueden provocar la falla de la estructura o la producción de ruidos molestos. En aplicaciones de ingeniería, las vibraciones de una barra, con diferentes condiciones de contorno, pueden utilizarse para simular la respuesta de diversas estructuras. Por ejemplo, se pueden modelar las vibraciones de una antena, los brazos de un robot, distintas componentes utilizadas en la construcción, las estructuras de puentes y partes de instrumentos musicales. En un curso de física básica de nivel universitario, las vibraciones de una barra constituyen una opción, o un complemento, del sistema masa-resorte que se utiliza como prototipo para el estudio de la resonancia. La barra, además, como todo sistema distribuido, tiene muchas frecuencias naturales y modos de vibración. Esto permite introducir el concepto de modos normales en los primeros años de la formación de un ingeniero [12].

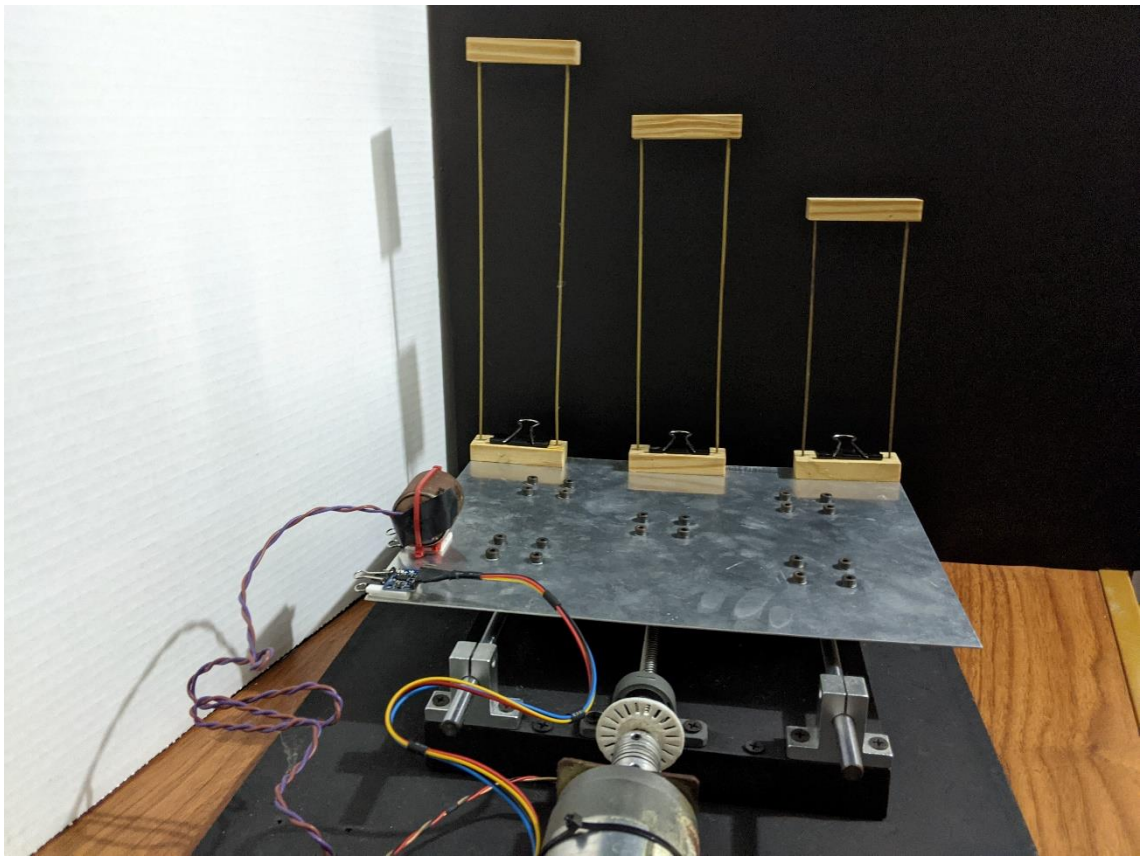


FIGURA 29 Estructuras sobre el simulador [Elaboración propia]

El experimento plateado para realizar la demostración del funcionamiento de la mesa o simulador, se basa en una estructura básica que puede modelarse como un péndulo, en este caso se tienen tres estructuras para poder observar que para cada una existe una frecuencia con la que entrara en resonancia. Se colocan tres estructuras de diferentes tamaños para apreciar el fenómeno de manera fácil y clara.

Se tienen tres estructuras en el dispositivo cada una con diferente altura y peso, pero de iguales características es decir diseño y materiales, las características de estas estructuras son las siguientes:

Tabla 4 Datos de las estructuras de prueba [Elaboración propia]

Estructura	**1**	**2**	**3**
Peso [g]	28	26	22
Altura [cm]	25	21	16

Capitulo IV: Implementación y resultados.

En este capítulo se presentará la implementación del simulador y sus resultados, teniendo en cuenta que visualizar los resultados es necesario observar el fenómeno, en este capítulo se exponen fotografías del resultado, teniendo en cuenta que la mejor forma de valorar el dispositivo es observando su funcionamiento, se presentan una serie de videos para observar el dispositivo.

5.1 Implementación del simulador

Luego de presentar la teoría y el diseño del dispositivo se presentan como se integran el hardware y el software para lograr el funcionamiento del dispositivo, logrando así el objetivo principal de hacer entrar en resonancia estructuras a escala.

En esta sección se presentan la implementación tanto del sistema de control como el de adquisición de datos, también se provee información necesaria para interactuar con el dispositivo, diagrama, puntos de conexión y manipulación de los programas, en el apartado de anexos se puede encontrar más información acerca la interacción con el programa de procesamiento de datos.

5.1.1 Lazo de control

La implementación del control se basa en un lazo cerrado con los componentes antes mencionados, el microcontrolador se encarga de comparar los datos obtenidos de encoder elemento encargado de la retroalimentación.

El programa es un ciclo infinito principalmente encargado de comparar el número de giros que el motor ejecuta, este valor se obtiene del encoder por medio de una interrupción que el microcontrolador se encarga de atender. Como ejemplo se muestra el diagrama de flujo para el numero de cuentas igual a 10 este valor equivale a una frecuencia de aproximadamente 7.5 Hz.

El valor de 10 equivale a que el motor de media vuelta, luego de contar a 10 se invierte el giro se espera a que el valor de cuentas sea el doble para llegar al valor de 10 cuentas en el sentido inverso generando así una onda senoidal aproximada que se puede observar en la gráfica de las aceleraciones obtenidas del geófono.

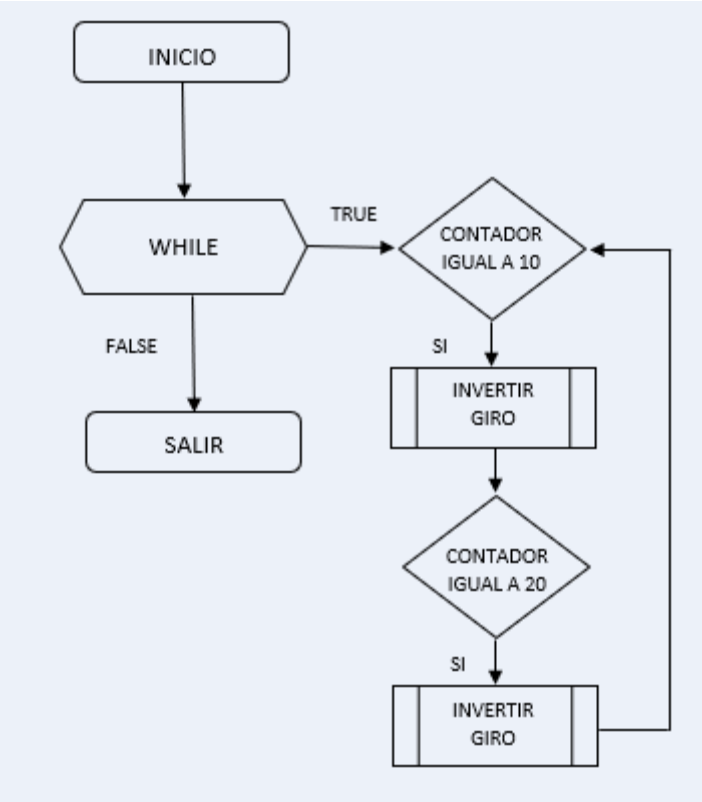


FIGURA 30 Diagrama de flujo, control del motor [Elaboración propia]

En el diagrama se muestra que el control es un programa sencillo, en el microcontrolador el valor de cuentas es una variable que se puede manipular para obtener diferentes frecuencias, este valor se puede manipular por medio de pulsadores que activan una nueva interrupción, se implementa utilizando interrupciones ya que el programa principal no puede dejar de ejecutarse porque se tendría una variación en las aceleraciones del dispositivo, es decir, se pierde el valor a comparar teniendo una variación del movimiento, por esta razón el programa de control debe estar separado de la medición.

```
if (count == count1) {
digitalWrite(InA1, LOW);
digitalWrite(InB1, HIGH);
}
if (count == count1*2) {
digitalWrite(InA1, HIGH);
digitalWrite(InB1, LOW);
count = 0;
```

FIGURA 31 Código del diagrama de flujo de la FIGURA 29. [Elaboración propia]

5.1.2 Programa implementado para el análisis de datos.

El análisis de datos se implementa utilizando las ventajas que Python ofrece para el análisis de datos, para esto se utiliza la librería numpy principalmente para el análisis de los datos. Esta librería es usada para calcular la FFT de las mediciones, obteniendo así la frecuencia del dispositivo.

La implementación busca el análisis y presentación de los datos en tiempo real de forma gráfica, obteniendo una gráfica para las mediciones de aceleración y otra para presentar la FFT en tiempo real, esto se realiza tanto para el acelerómetro como para el geófono teniendo de esta forma cuatro graficas en la ventana creada para el programa.

Para obtener los datos de las mediciones del simulador es necesario comunicar el microcontrolador con la PC encargada de ejecutar el script Python, para esto es necesario implementar una forma de comunicación, la librería Pyserial realiza esta tarea, la comunicación se realiza por protocolo serial, implementando una función en el microcontrolador encargada de encapsular y enviar los datos utilizando el protocolo serial y comunicación por USB. En el script se tiene una tarea en segundo plano encargada de recibir y poner a disposición de la tarea principal todos los datos del puerto serial.

El propósito de obtener la FFT es conocer la frecuencia la que el dispositivo se encuentra funcionando, de esta forma por observación se puede determinar la frecuencia a la que una estructura puesta a prueba entra en resonancia.

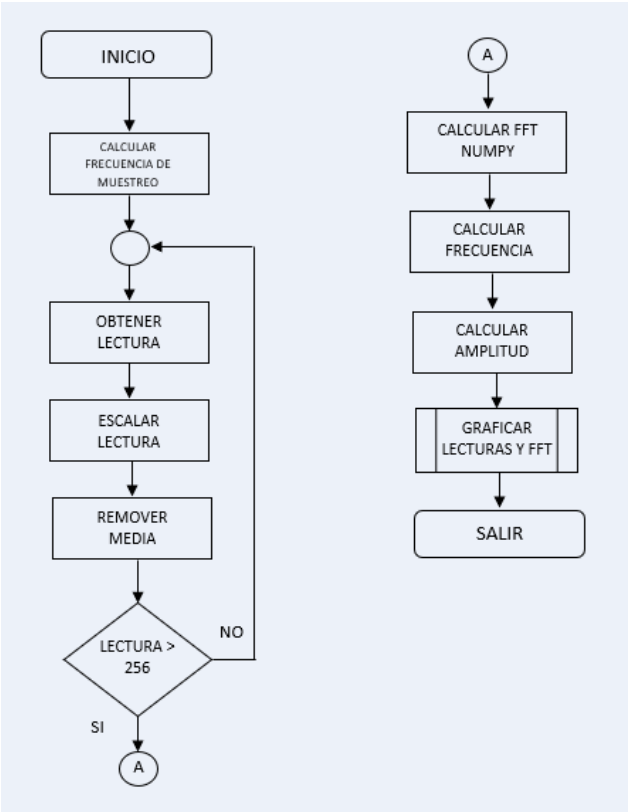


FIGURA 32 Diagrama de flujo para el análisis de datos. [Elaboración propia]

5.2 Implementación de Hardware y Software

Habiendo establecido el diseño del simulador, se procede a integrar los componentes necesarios. Este capítulo presenta como los componentes de hardware han sido integrados para lograr el funcionamiento del dispositivo, explicando su función, características y el resultado obtenido.

5.2.1 Implementación de Hardware

La implementación del hardware se realiza primero para que el sistema de control pueda ser desarrollado en una plataforma estable con la menor cantidad de vibraciones introducidas por los componentes mecánicos. El diagrama de bloques se muestra en la FIGURA 33.

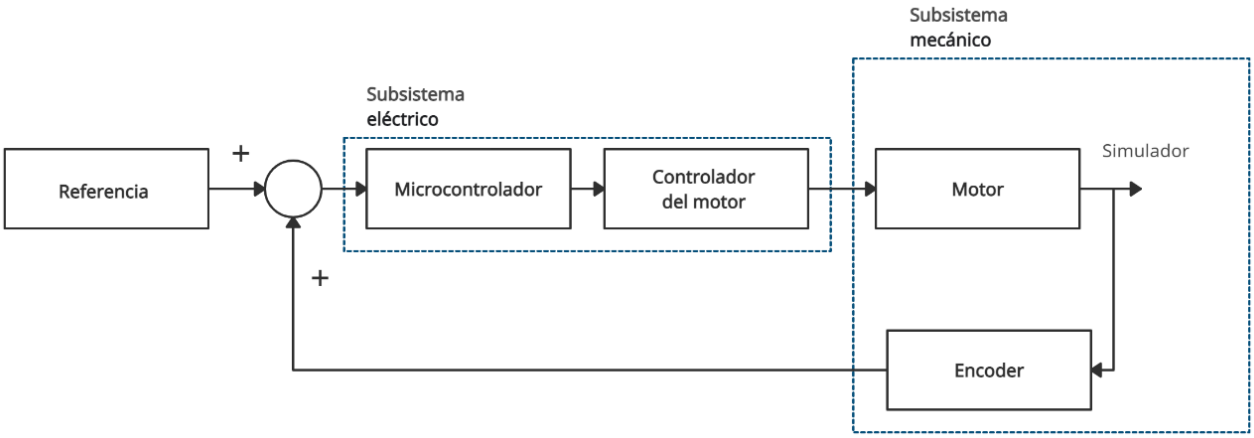


FIGURA 33 Diagrama de bloques del simulador [Elaboración propia]

La referencia representa el valor de frecuencia seleccionado para trabajar, es la entrada que el usuario introduce al sistema mecánico. El eje del motor es accionado por el controlador de acuerdo a este valor establecido, una vez el eje se encuentra en rotación el encoder comienza a generar valores que son recibidos por el microcontrolador y cambian el giro del motor, al mismo tiempo el simulador se encuentra generando vibraciones de acuerdo al giro del motor.

5.3 Control y medición.

El control y medición se programan en un mismo microcontrolador con la ventaja de tener doble núcleo teniendo así resultados sin retrasos en la medición que causaría un resultado erróneo del análisis de las muestras.

Cada núcleo tiene una función, encargándose de tareas específicas para la correcta operación del dispositivo, el núcleo cero se encarga de la medición, es decir, la toma de datos de los acelerómetros en la base del dispositivo y transmitir estos datos al programa encargado de su análisis El núcleo uno se encarga del control del dispositivo, el control se realiza por medio de modulación de ancho de pulso, controlada por el conteo realizado por el encoder situado en el eje del motor.

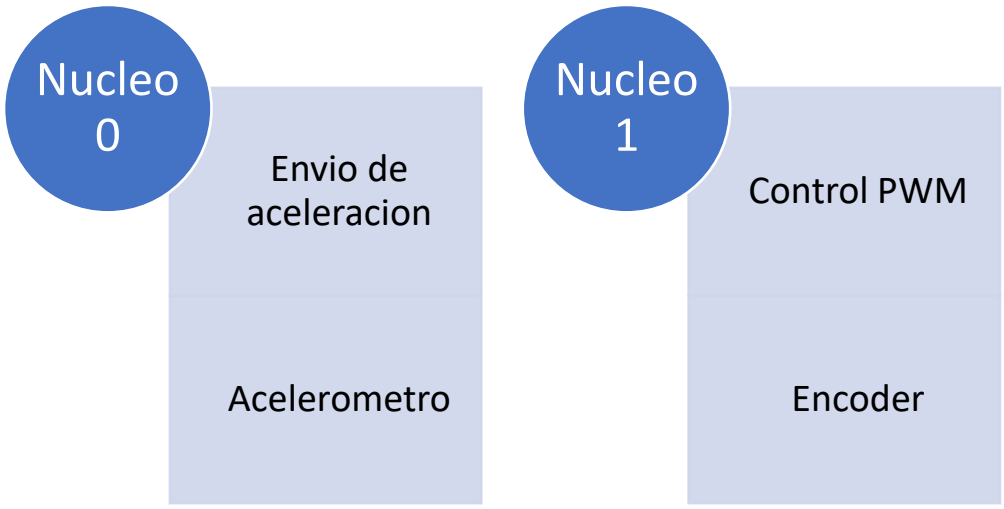


FIGURA 34 Tareas asignadas a cada núcleo [Elaboración propia]

El diagrama de flujo general de las tareas ejecutadas por el microcontrolador se observa en la FIGURA 34, al iniciar el programa se inician las variables necesarias tanto para el control como la medición, esto se hace por defecto en el núcleo 1 luego se le asigna una tarea específica a cada núcleo, el núcleo 0 para la medición y el núcleo 1 para el control del dispositivo. La asignación de tareas se hace por medio de una función preestablecida tomada del sistema RTOS en la cual se deben asignar parámetros como nombre, prioridad y a que núcleo se le asigna.

En el diagrama de la FIGURA 35 se observa la inicialización del microcontrolador es decir inicializar puertos de entra y salida, puertos de comunicación en este caso serial e I2C para las lecturas de los acelerómetros. Luego se corren en paralelo los programas

para controlar el motor y tomar las capturas de los datos cada uno en su respectivo núcleo, estos programas son bastantes simples, pero pueden tener sus inconvenientes cuando se tiene que atender una interrupción, en general es desempeño de este microcontrolador es el adecuado para el dispositivo.

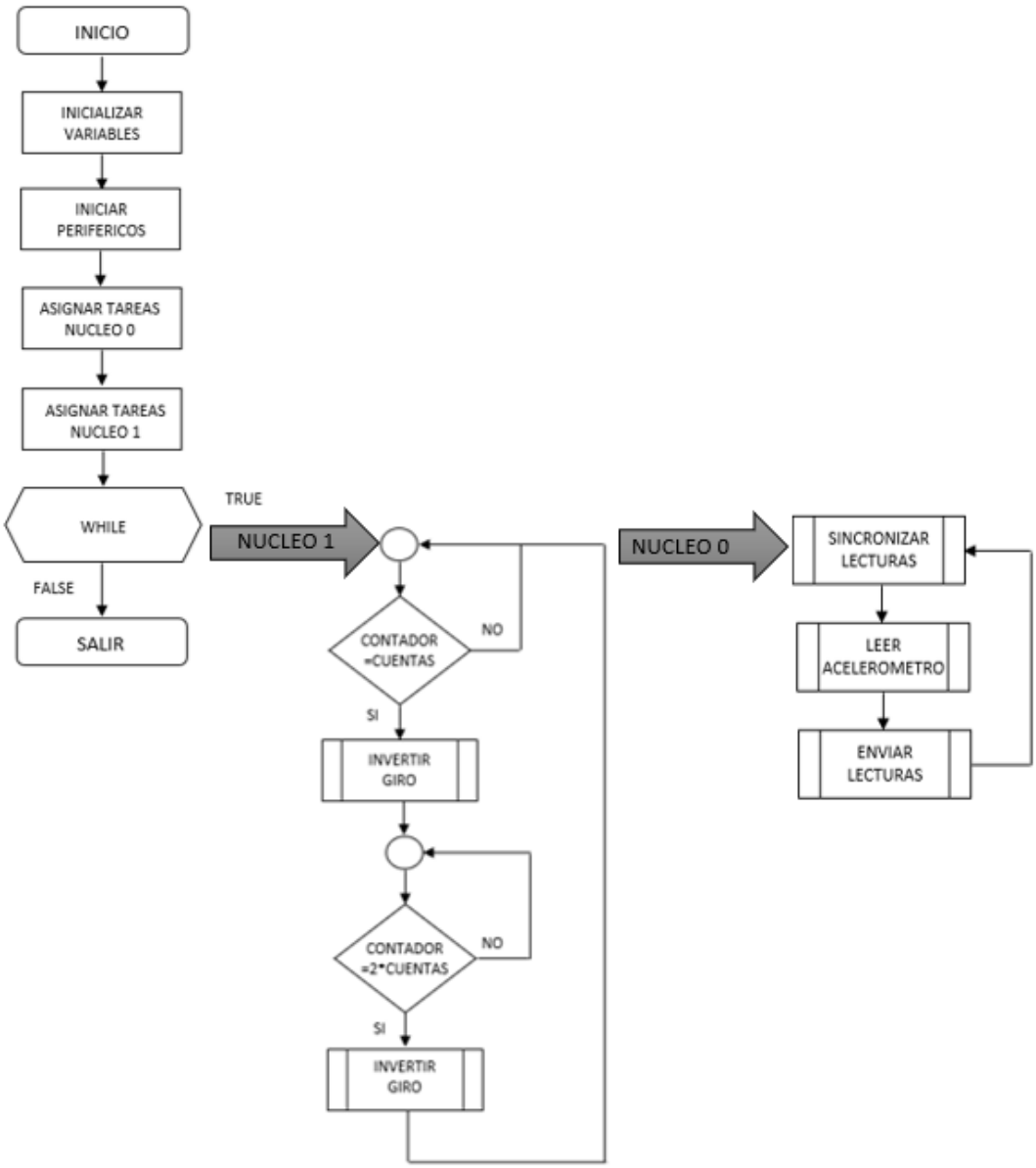


FIGURA 35 Diagrama de flujo, microcontrolador. [Elaboración propia]

5.4 Resultados del funcionamiento

Al ser un dispositivo para demostrar el fenómeno de la resonancia lo mejor sería observar su comportamiento, es decir, se vuelve difícil presentar el funcionamiento del dispositivo en un documento, por lo tanto, se tiene una serie de videos en los que se aprecia su funcionamiento.

https://drive.google.com/drive/folders/1rKZk-tiBL3voXlMxuKb5Q4yLUsB-Ztc-?usp=sharing

Para realizar la demostración se utilizan tres estructuras a escala compuestas por una masa en la parte superior y dos pequeños soportes, el propósito de tener estas estructuras es de facilitar la demostración del fenómeno ya que permiten oscilaciones bruscas sin dañarse siendo necesario para logra la característica de repetitividad sin necesidad de cambiar por estructuras nuevas, debido a que el fenómeno puede ser destructivo.

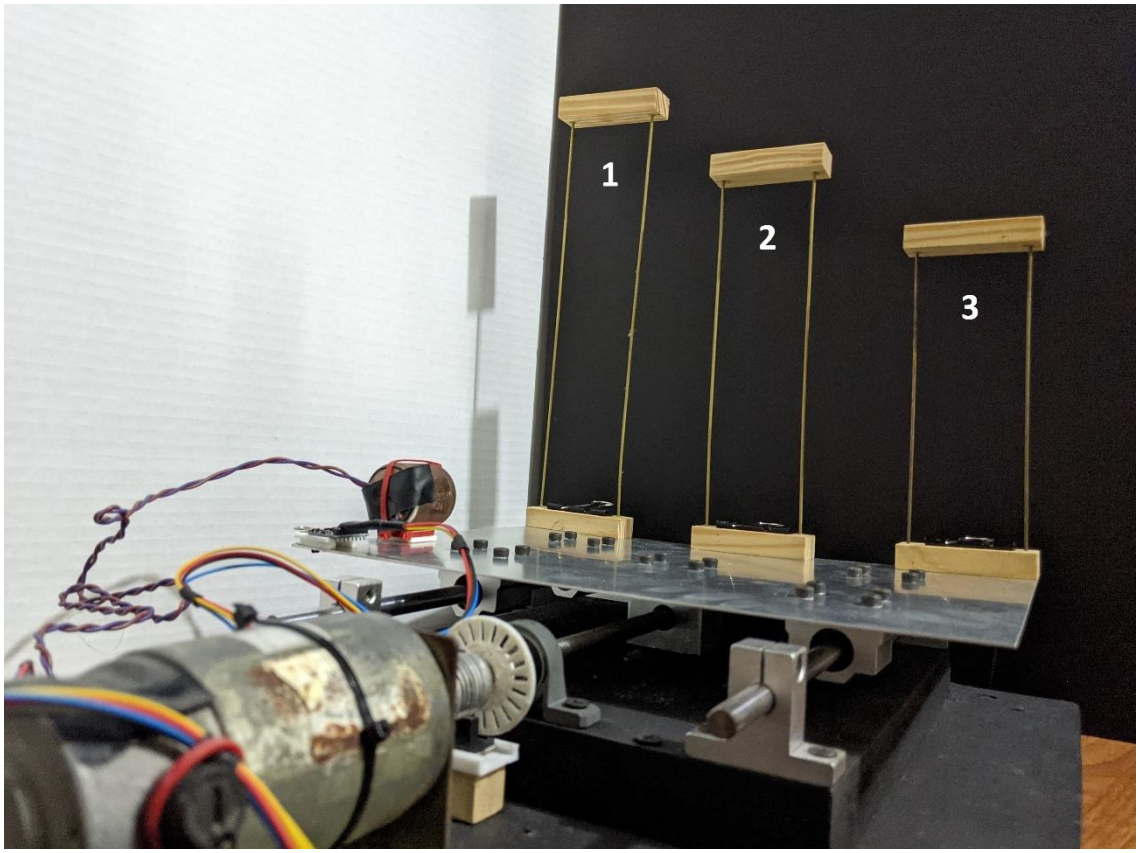


FIGURA 36 Simulador de resonancia. [Elaboración propia]

Se tienen tres estructuras en el dispositivo cada una con diferente altura y peso, pero de iguales características es decir diseño y materiales, las características de estas estructuras son las siguientes:

Estructura	1	2	3
Peso [g]	28	26	22
Altura [cm]	25	21	16

A continuación, se presenta una serie de fotografías y capturas de pantalla para demostrar el funcionamiento del dispositivo:

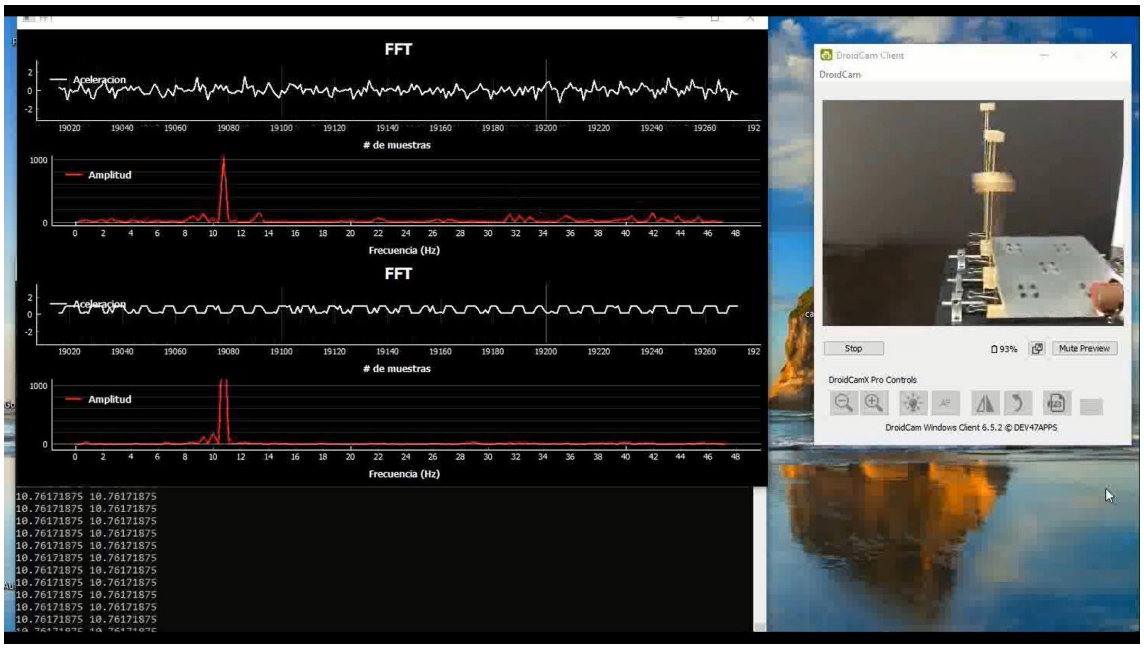


FIGURA 37 Estructura 3 en resonancia. [Elaboración propia]

En la FIGURA 37 se observa como la estructura 3 se encuentra en estado de resonancia, esto se logra haciendo funcionar el dispositivo a una frecuencia de 10 Hz aproximadamente, para obtener esta frecuencia primero se comienza a aumentar la frecuencia del dispositivo, se logra aumentando el giro del motor, es decir, la frecuencia con la que se detentan pasos por el disco ranurado, una vez identificada, se programa una variable para poder obtener esta frecuencia al apretar un pulsador. De esta forma el fenómeno se puede observar fácilmente y puede repetirse la cantidad de veces que se requiera solamente apretando un botón.

En la FIGURA podemos verificar en la parte superior los valores de aceleración obtenidos por el acelerómetro MPU6050, luego en la siguiente grafica se presenta la FFT de los datos de aceleración, en la esquina inferior izquierda se presenta el valor máximo de la FFT obteniendo así la frecuencia del dispositivo en este caso la frecuencia de resonancia de la estructura 3 ver FIGURA 38. Los datos a la par de los señalados so obtenidos usando el geófono, estos datos se muestran en las graficas 3 y 4 de la FIGURA 37.

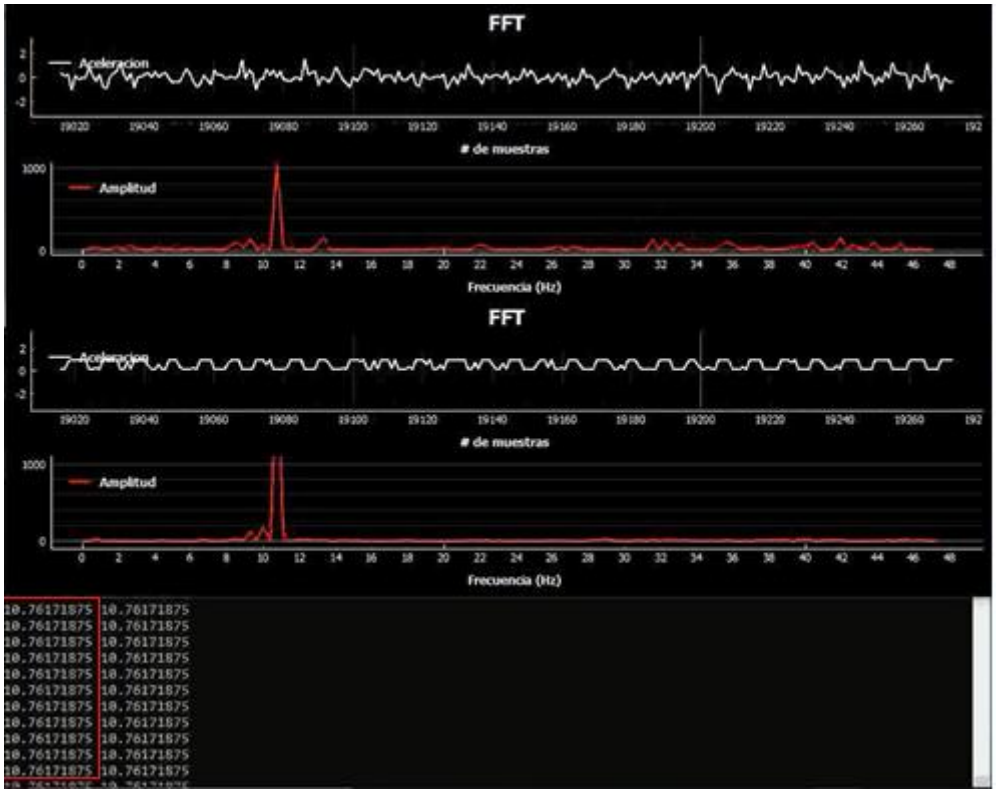


FIGURA 38 Frecuencia de resonancia, estructura 3 [Elaboración propia]

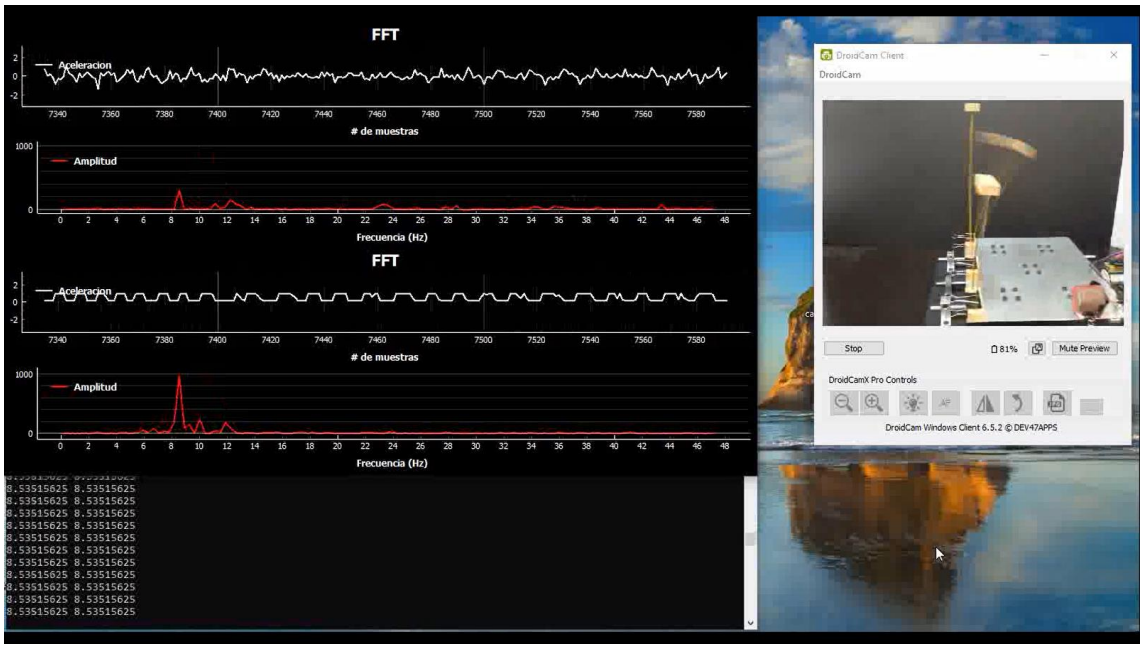


FIGURA 39 Estructura 2 en resonancia. [Elaboración propia]

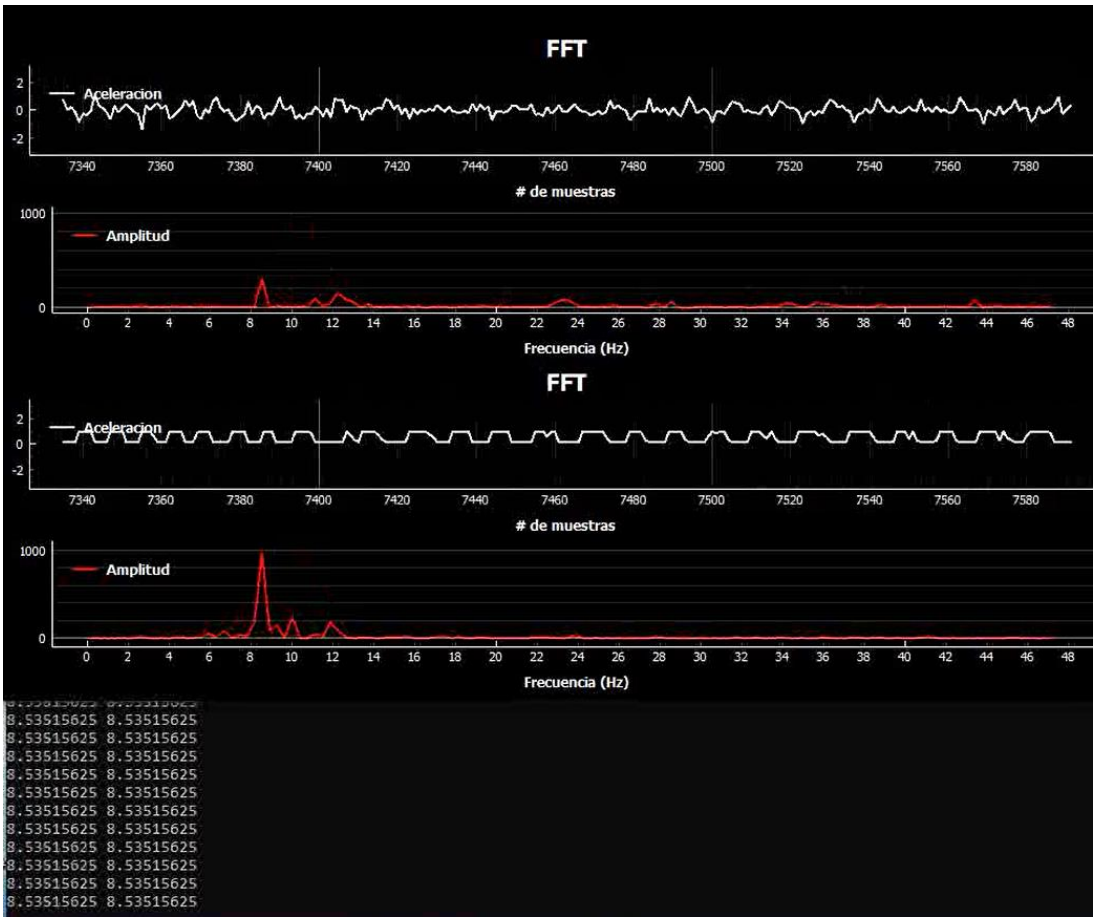


FIGURA 40 Frecuencia de resonancia, estructura 2 [Elaboración propia]

Las FIGURAS 39 y 40 muestran a la estructura 2 en resonancia se puede observar como las estructuras de los costados permanecen sin ningún cambio a la frecuencia de 8.5 Hz mientras que la estructura del centro presenta un desplazamiento fácilmente apreciable incluso en la imagen. En las gráficas se observan los datos de aceleración capturados por los acelerómetros, se aprecia que las gráficas son diferentes debido a la sensibilidad de cada acelerómetro el geófono presenta resistencia a los cambios bruscos, por eso se aprecia una gráfica más cercana a una onda senoidal, teniendo en cuenta la diferencia de los datos ambas FFT tienen el mismo máximo por lo tanto la misma frecuencia.

En la FIGURAS 41 y 42 se observa como la estructura mayor entra en resonancia a una frecuencia de 7 Hz aproximadamente, esto se observa en los datos de las gráficas, en este caso se observa como las otras estructuras se mantienen sin ningún cambio mientras que la estructura 1 está en resonancia.

En esta sección se presentaron capturas del dispositivo tanto de su funcionamiento como del procesamiento de datos que ocurre mientras está en funcionamiento, como se explica en secciones anteriores el procesamiento lo realiza una PC ejecutando un script Python, a pesar de la dificultad de presentar los resultados en el documento, porque es necesario observarlo para tener una mejor idea de lo que ocurre en el dispositivo, se observa en las imágenes lo que sucede a determinadas frecuencias.

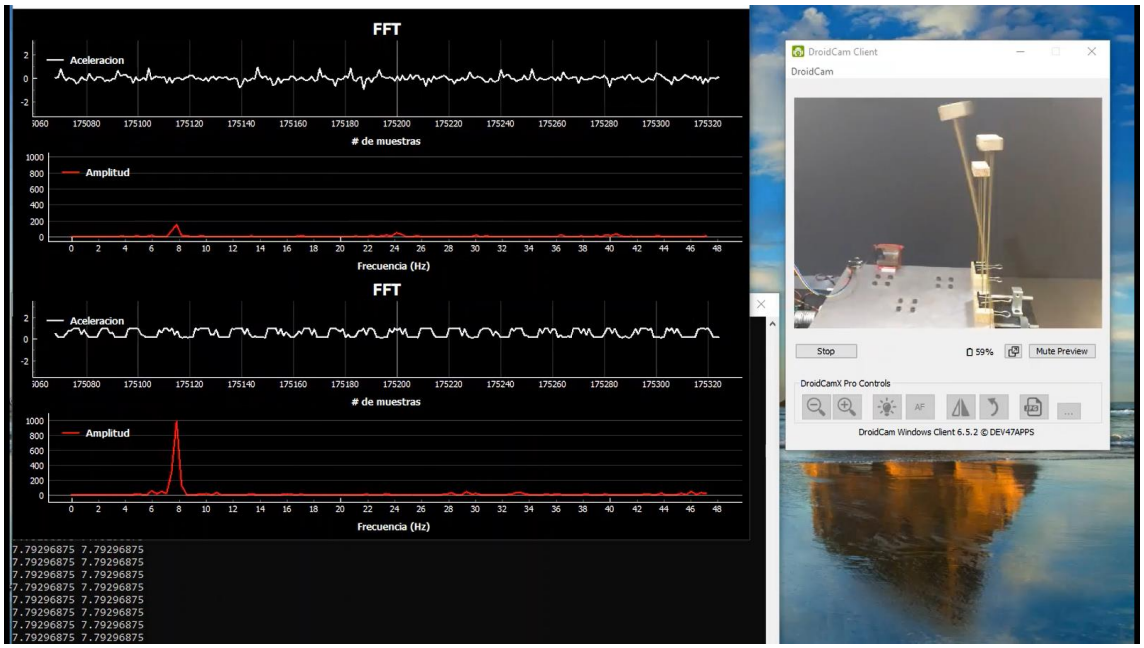


FIGURA 41 Estructura 3 en resonancia. [Elaboración propia]

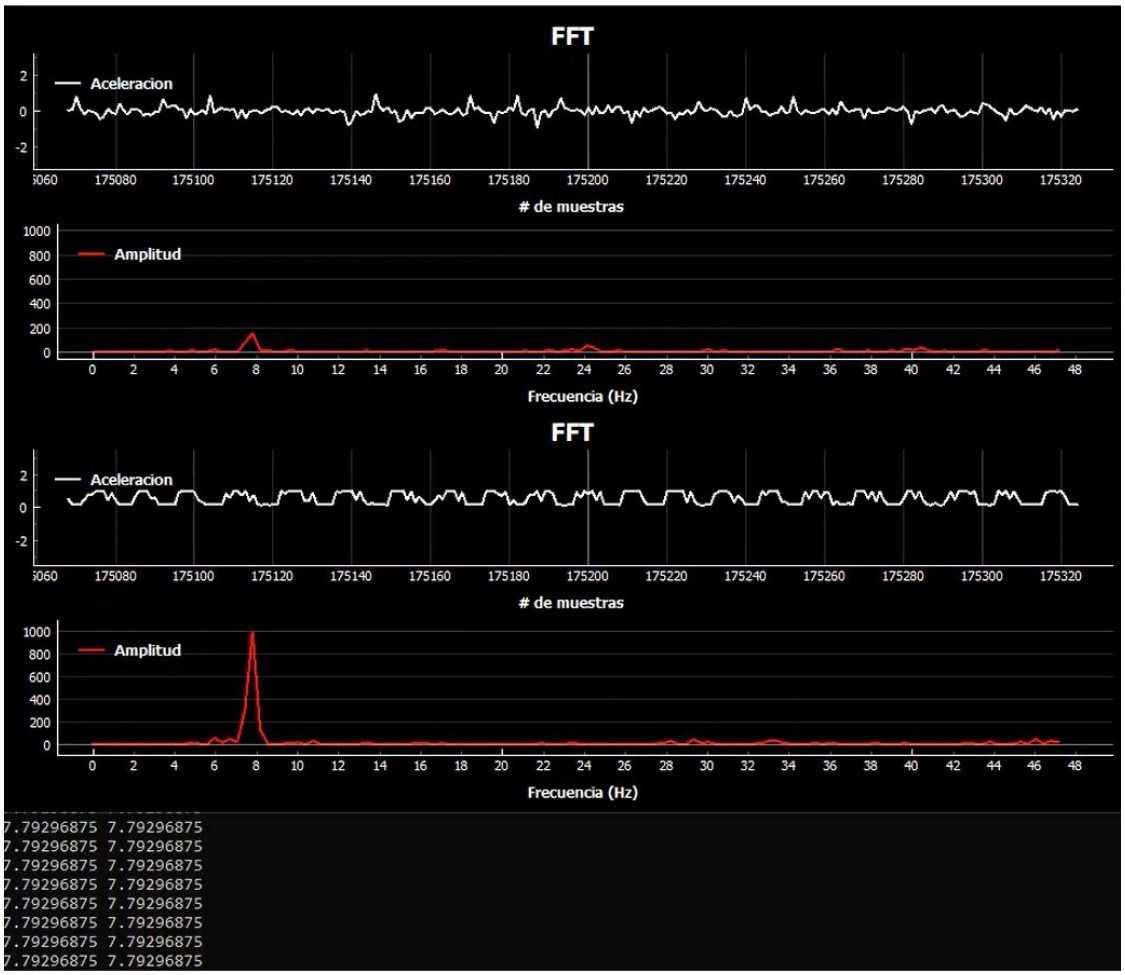


FIGURA 42 Frecuencia de resonancia, estructura 1 [Elaboración propia]

Capítulo V: Conclusiones y líneas futuras.

6.1 Conclusiones.

- A pesar de las limitantes en la medición de la posición del eje del motor, es decir, el encoder utilizado, se logra un desempeño favorable del simulador siendo capaz de lograr los resultados esperados provocando que las estructuras a escala sometidas al movimiento de la plataforma del simulador entren en resonancia cada una a su debida frecuencia.

- El microcontrolador ESP32 tiene la capacidad de ejecutar la operación del simulador de manera óptima, teniendo las ventajas de programación que ofrece Arduino, únicamente se debe tenerse claro que cada núcleo del microcontrolador ejecuta una serie de instrucciones específicas y debe seguirse una metodología de programación establecida, explicada en esta investigación.

- La utilización del dispositivo tiene su complejidad, pero se ofrecen una serie de detalles que permite su fácil entendimiento en identificación de cada cable necesario para el dispositivo, una estructura modular que permite la fácil comprensión de como conectar el dispositivo, el diseño de cómo se encuentra implementado el dispositivo, además de la documentación adecuada para cada programa necesario para modificar el comportamiento del dispositivo.

- Se realizo la demostración del fenómeno de la resonancia, utilizando estructuras a escala de diferente altura, observando que para cada estructura existe una frecuencia en la que esta estructura entra en resonancia.

- Se diseñaron los circuitos necesarios para el control y la medición del dispositivo, teniendo de esta forma un mejor resultado en el dispositivo, porque al tener un movimiento constante cada componente debe tener un contacto seguro para mantener la comunicación con el dispositivo.

- Se presenta el análisis en tiempo real de los datos de aceleraciones del dispositivo teniendo de esta forma la frecuencia a la que cada estructura entra en resonancia, este programa se implementa utilizando un script en Python.

6.2 Líneas futuras.

- Desarrollar el análisis matemático del simulador y estructuras implementados en la investigación, de esta forma podrán ser comparados los resultados obtenidos con el análisis teórico de las estructuras.

- Implementar un control mejorado utilizando un encoder de mayor precisión, con una mayor precisión se puede desarrollar un nuevo programa de control utilizando un PID para el control del simulador. Disminuyendo así los errores causados por factores externos, teniendo un mejor resultado en el comportamiento del simulador.

- Agregar características de control remoto para poder realizar demostraciones desde una ubicación x de manera que pueda ser presentado y manipulado en línea.

- Elaborar el análisis y desarrollo de estructuras más complejas que puedan ser sometidas a resonancia para tener diferentes tipos de demostraciones o experimentos para presentar.

- Implementar un programa de control capaz de simular el comportamiento de sismo para volver el simulador de resonancia una mesa sísmica y tener de esta forma un amplio rango de capacidades que el dispositivo pueda desempeñar.

- Mejorar la interfaz gráfica en Python para ser capaz de manipular el dispositivo desde la misma computadora o desde una red de internet, permitir la visualización de los datos de frecuencia en la misma pantalla de presentación de datos.

Bibliografía

[1] Resnick and Halliday (1977). *Física* (3rd edición). John Wiley & Sons. p. 324.

[2] The characterization and optimization of earquake shaking table performance. Crewe Adam J.

[3] Dinámica estructural aplicada al diseño sísmico, Luis Enrique Garcia, 1998.

[4] Evaluación del diseño de una pequeña mesa vibratoria para ensayos de ingeniería, Julian Carrillo

[5] Diseño de una mesa sísmica a escala, Pontificia universidad católica de Valparaíso. 2018

[6] Anil K. Chopra. Earthquake dynamics of structures (2dn edición), Eartquake engineering research institute.

[7] Horacio A. Coral. Diseño, construcción y control de un simulador sísmico uniaxial tele-operable para modelos estructurales a pequeña escala. Universidad del valle.

[8] Husillo y tuerca husillo, rodamientosbrasil.com/husillo

[9] T. Baran, A.K. Tanrikulu, C. Dundar, and A.H. Tanrikulu. Construction and performance test of a low-cost shake table.

[10] K. Ogata. Ingeniería de Control Moderna. Madrid: Pearson Education, 2010.

[11] Espressif Systems. ESP32 Datasheet, V3.8, 2021.

[12] C.E. RepettoA. RoattaR.J. Welti. Medición de frecuencias de resonancia, factor de pérdida y módulo de Young dinámico de varillas empotradas

[13] AD623 Datasheet, https://www.analog.com/media/en/technical-documentation/data-sheets/ad623.pdf

[14] MPU6050 Datasheet, https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[15] Módulo de acelerómetro y giroscopio MPU6050, https://components101.com/sensors/mpu6050-module

Anexos

Anexo 1 Componentes del control.

Los componentes del sistema de control son: un microcontrolador, un driver para el motor DC, un sensor óptico con un disco ranurado (encoder) y un motor DC.

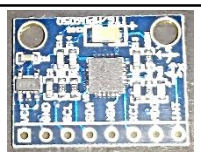
Componentes electronicos.		
Ilustración	Componente	Descripción
	Microcontrolador ESP32	Encargado del control y medicon del simulador.
	Acelerometro 6050	Mide las aceleraciones a la que es sometida la base (mesa).
	Sensor optico, usado como encoder	Permite conocer la posicion del eje del motor.
	Pulsadores	Cada uno se encarga de un modo de funcionamiento.
	L298N controlador del motor	Se encarga de comunicar el microcontrolador con el motor.
	AD623an amplificador de instrumentacion	Permite leer las aceleraciones del geofono.
	Geofono Mark L28	Mide aceleraciones, se compara con el acelerometro.

FIGURA 43 Componentes del control. [Elaboración propia]

Anexo 2 Componentes del sistema mecánico

Al seleccionar el sistema husillo tuerca se procede a listar los componentes necesarios para implementarlo, cada uno de estos componentes tiene una finalidad en el sistema, principalmente se cuenta con el tornillo trx8 que sería el husillo y la tuerca trx8, los otros componentes sirven para sostener la base (mesa), guiar el movimiento y fijar el tornillo a la base.

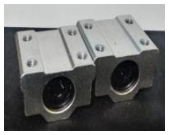
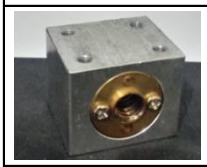
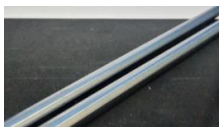
Componentes mecánicos.		
Ilustración	Componente	Descripción
	Soporte para eje de 8mm.	Permite sujetar el eje de 8mm
	Rodamiento lineal de 8mm	Permite el movimiento sobre el eje de 8mm
	Chumacera con rodamiento KP08	Sujeta el tornillo, eje de rotación de la mesa
	Base y tuerca trx8 x 400mm	Transfiere el movimiento del tornillo a la mesa
	Tornillo trx8	Acopla el motor al sistema
	Guías lineales de 8mm	Sostienen la base y la guían.
	Acople de 8mm para el motor	Permite acoplar el motor con el eje de 8mm.

FIGURA 44 Componentes del sistema. [Elaboración propia]

Estos componentes permiten un movimiento con la menor cantidad de fricción y sin introducir vibraciones o golpes no deseados, siempre y cuando se tenga una alineación y lubricación correcta de los componentes, al estar desalineados o sin lubricación se tiene fricción y desgaste de los componentes e incluso atascamientos en el simulador.

Anexo 3 Costos del dispositivo

Tabla 5 Costos del dispositivo [Elaboración propia]

#	Descripción	Precio
1	Piezas para movimiento. (ejes, tornillos, rodamientos, soportes)	$ 39.99
2	Acelerómetro mpu6050	$ 4.00
3	Camisa tuerca T8	$ 4.75
4	Driver L298N	$ 4.75
5	ESP32	$ 9.95
6	Encoder	$ 5.95
7	Placa PCB	$ 1.00
8	Lamina aluminio	$ 5.00
9	Termo contraíble (varios calibres)	$ 5.00
10	Cables varios	$ 3.00
11	Conectores	$ 4.95
12	Lata aerosol negro	$ 4.95
13	Tornillos varios	$ 5.00
	Total	$ 98.29

Anexo 4 ESP 32 PINOUT

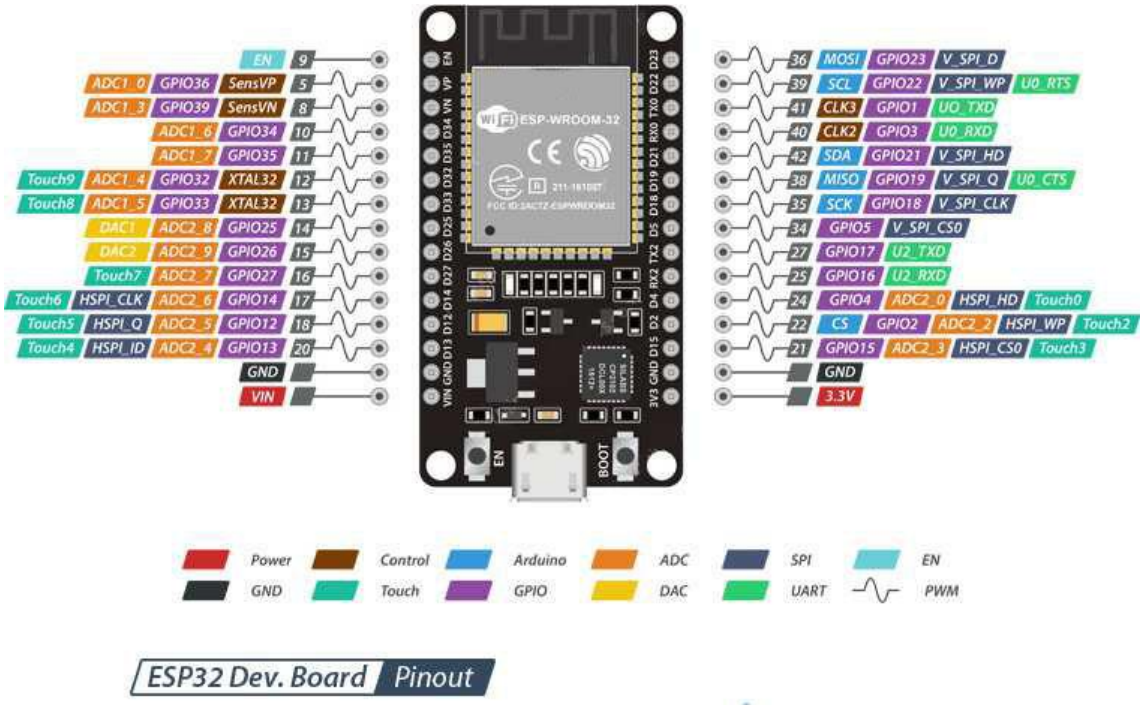


FIGURA 45 Pines del esp32

Anexo 5 Pasos para modificar script Python.

Para poder modificar el script encargado de analizar los datos obtenidos del simulador es necesario instalar las librerías de los recursos necesarios para poder correr el script los pasos son los siguientes:

Primero instalar librería serial que se encarga de todo lo relacionado con la comunicación serial con el microcontrolador, en el script se tiene una clase que se encarga de ejecutar todo lo relacionado con esta librería, la recepción de datos, etc.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pip install pyserial
```

Luego se procede a instalar la librería pyqtgraph que se encarga de la presentación del análisis de los datos del simulador, en el script se cuenta con una clase dedicada al análisis y presentación de datos en tiempo real utilizando esta librería.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pip install pyqtgraph
```

La librería pyqt5 se encarga de todo lo relacionado con la interfaz gráfica, es decir, la ventana en la que se presentan los datos.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pip install pyqt5
```

Por último, es necesario instalar la librería numpy que se encarga del análisis de los datos y que principalmente se utiliza la FFT que incluye esta librería.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pip install numpy
```

Para lograr un archivo ejecutable para tener la facilidad de solamente ejecutar el programa y comenzar a mostrar los datos del simulador, es necesario crear un archivo ejecutable para esto es necesario tener la librería pyintaller.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pip install pyinstaller
```

Una vez instalada la librería se puede crear un ejecutable del programa al poner el comando pyinstaller y el nombre del script.

```
PS C:\Users\ADRIAN\Documents\PythonScripts> pyinstaller tbe115.py
103 INFO: PyInstaller: 4.2
103 INFO: Python: 3.5.3
104 INFO: Platform: Windows-10-10.0.18362-SP0
106 INFO: wrote C:\Users\ADRIAN\Documents\PythonScripts\tbe115.spec
109 INFO: UPX is not available.
121 INFO: Extending PYTHONPATH with paths
['C:\\Users\\ADRIAN\\Documents\\PythonScripts',
 'C:\\Users\\ADRIAN\\Documents\\PythonScripts']
195 INFO: checking Analysis
283 INFO: Building because C:\Users\ADRIAN\Documents\PythonScripts\tbe115.py changed
284 INFO: Initializing module dependency graph...
288 INFO: Caching module graph hooks...
307 INFO: Analyzing base_library.zip ...
4405 INFO: Caching module dependency graph...
4507 INFO: running Analysis Analysis-00.toc
4535 INFO: Adding Microsoft.Windows.Common-Controls to dependent assemblies of final executable
  required by c:\users\adrian\appdata\local\programs\python\python35\python.exe
4605 INFO: Analyzing C:\Users\ADRIAN\Documents\PythonScripts\tbe115.py
5147 INFO: Processing pre-find module path hook PyQt5.uic.port_v2 from 'c:\\users\\adrian\\appdata\\local\\programs\\py
ller\\hooks\\pre_find_module_path\\hook-PyQt5.uic.port_v2.py'.
```

Anexo 6 Programa del microcontrolador

```
#include <Wire.h>
#include <Arduino.h>

TaskHandle_t Control;
TaskHandle_t Medicion;

#define InA1            13
#define InB1            12
#define enable1Pin      14
#define encodPinB1      35
#define buttonPin4      18
#define buttonPin3      17
#define buttonPin2      4
#define buttonPin1      2
#define buttonPin      15



int count1 = 2;
int var1 =0;
int var2 =0;
const int geoPin = 34;
int geoValue = 0;
#define fix      5 //pull up para pin
/***************VARIABLES DE CONTROL*********************/
int contador=100;     // Variable contador igual a cero
int count = 0;        //Contador encoder
static volatile unsigned long debounce = 0; //almacena tiempo encoder
static volatile unsigned long debounce2 = 0;
//variables para PWM
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 250;



const int MPU_addr=0x68;  // I2C address of the MPU-6050
/***************VARIABLES DE MEDICION*******************/
unsigned long timer = 0;
long loopTime = 5000;   // microsegundos
String g_recvString = "";
int AcX,AcY,AcZ;
double g_scaleFactor =  16384.0;

void setup() {
//////////////////////MEDICION/////////////////////////////////
  Serial.begin(115200);
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
//////////////////////CONTROL/////////////////////////////////
  //pines para control de motor
  pinMode(InA1, OUTPUT);
  pinMode(InB1, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
```

```
  attachInterrupt(buttonPin, isr2, FALLING);
  pinMode(buttonPin1, INPUT_PULLUP);
  attachInterrupt(buttonPin1, isr3, FALLING);
  pinMode(buttonPin2, INPUT_PULLUP);
  attachInterrupt(buttonPin2, isr4, FALLING);
  pinMode(buttonPin3, INPUT_PULLUP);
  attachInterrupt(buttonPin3, isr5, FALLING);
  pinMode(buttonPin4, INPUT_PULLUP);
  attachInterrupt(buttonPin4, isr6, FALLING);
  pinMode(fix, INPUT_PULLUP);

  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(enable1Pin, pwmChannel);
  //pin cono entreda de encoder
  pinMode(encodPinB1, INPUT);
  //interrupciones
  attachInterrupt(encodPinB1, isr, FALLING);  //Interrupcion encoder
  //direccion inicial de motor
  digitalWrite(InA1, HIGH);
  digitalWrite(InB1, LOW);
  ledcWrite(pwmChannel, dutyCycle);
  //tarea a ejecutar en core 1 (CONTROL)
  xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Control",   /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Control,    /* Task handle to keep track of
created task */
                    1);          /* pin task to core 1
*/
  delay(500);

  //tarea a ejecutar en core 0 (MEDICION)
  xTaskCreatePinnedToCore(
                    Task2code,   /* Task function. */
                    "Medicion",  /* name of task. */
                    10000,       /* Stack size of task */
                    NULL,        /* parameter of the task */
                    1,           /* priority of the task */
                    &Medicion,    /* Task handle to keep track of
created task */
                    0);          /* pin task to core 0 */
    delay(500);

  }
/**********************CONTROL*********************************/
void Task1code( void * pvParameters ){
  for(;;){
    if (count == count1) {
    digitalWrite(InA1, LOW);
    digitalWrite(InB1, HIGH);
    }

    if (count == count1*2) {
    digitalWrite(InA1, HIGH);
    digitalWrite(InB1, LOW);
    count = 0;

}
```

```
//Serial.println(count1); //obtener valor de estructura
}
}

/*********************MEDICION*******************************/
void Task2code( void * pvParameters ){

  for(;;){
timeSync(loopTime);

    mpu_read();
    geoValue = analogRead(geoPin);
    //Serial.println(geoValue);
    getSerialData();
    sendToPC(&geoValue, &AcY, &AcZ);

  }
}

void loop() {

}
/***************FUNCIONES  DE CONTROL*********************/
////////////INTERRUPCION DE ENCODER//////////////////////////////
void  isr()  {                  //leer encoder
  if(  digitalRead (encodPinB1) && (micros()-debounce > 1000) &&
digitalRead (encodPinB1) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        count++;}  // Suma el pulso bueno que entra.
        //else ;
}

void isr2() {
    if(  digitalRead (buttonPin) && (micros()-debounce2 > 900) &&
digitalRead (buttonPin) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce2 = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        count=0;
        count1=12;}  // Cambia el valor de cuentas.
}
void isr3() {
    if(  digitalRead (buttonPin1) && (micros()-debounce2 > 900) &&
digitalRead (buttonPin1) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce2 = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        count = 0;
        count1=9;}  // Cambia el valor de cuentas.
}
void isr4() {
```

```
    if(  digitalRead (buttonPin2) && (micros()-debounce2 > 900) &&
digitalRead (buttonPin2) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce2 = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        count = 0;
        count1=7;}  // Cambia el valor de cuentas.
}
void isr5() {
    if(  digitalRead (buttonPin3) && (micros()-debounce2 > 600) &&
digitalRead (buttonPin3) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce2 = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        //count=0;
        count1++;}  // Aumenta el valor de cuentas.
}
void isr6() {
    if(  digitalRead (buttonPin4) && (micros()-debounce2 > 600) &&
digitalRead (buttonPin4) ) {
// Vuelve a comprobar que el encoder envia una señal buena y luego
comprueba que el tiempo es superior a 1000 microsegundos y vuelve a
comprobar que la señal es correcta.
        debounce2 = micros(); // Almacena el tiempo para comprobar que
no contamos el rebote que hay en la señal.
        //count = 0;
        count1--;}  // Disminuye el valor de cuentas.
}



/***************FUNCIONES DE MEDICION*******************/
//////////////LECCTURA
ACELEROMETRO//////////////////////////////////////
void mpu_read(){
 Wire.beginTransmission(MPU_addr);
 Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
 Wire.endTransmission(false);
 Wire.requestFrom(MPU_addr,14,true);  // request a total of 14
registers
 AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
 AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E
(ACCEL_YOUT_L)
 AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40
(ACCEL_ZOUT_L)

 }
//////////////////////SINCRONIZACION DE
LECTURAS//////////////////////////////////
   void timeSync(unsigned long deltaT)
{
  unsigned long currTime = micros();
  long timeToDelay = deltaT - (currTime - timer);
  if (timeToDelay > 5000)
  {
```

```
    delay(timeToDelay / 1000);
    delayMicroseconds(timeToDelay % 1000);
  }
  else if (timeToDelay > 0)
  {
    delayMicroseconds(timeToDelay);
  }
  else
  {
      // timeToDelay is negative so we start immediately
  }
  timer = currTime + timeToDelay;
}
//////////////////LECTURA DE PUERTO SERIAL//////////////////////
void getSerialData()
{
  while (Serial.available())
  {
    char input = Serial.read();
    g_recvString += input;
    if (input == '%')   // this is the end of message marker
    {
      int index = g_recvString.indexOf('%');
      String tmp = g_recvString.substring(0, index);  // remove the
'%' sign
      g_scaleFactor = tmp.toFloat();
      g_recvString = "";
      break;
    }
  }
}

////////////////////ENVIO DE
DATOS////////////////////////////////////
/*funcion encargada del envio de aceleraciones para su analisis*/
void sendToPC(int* data1, int* data2, int* data3)
{
  byte* byteData1 = (byte*)(data1);
  byte* byteData2 = (byte*)(data2);
  byte* byteData3 = (byte*)(data3);
  byte buf[6] = {byteData1[0], byteData1[1],
                 byteData2[0], byteData2[1],
                 byteData3[0], byteData3[1]};
  Serial.write(buf, 6);
}
```

Anexo 7 Script Python

```
from threading import Thread
import serial
import time
import collections
import struct
import copy
import sys
import glob
import pyqtgraph as pg
from pyqtgraph.Qt import QtCore, QtGui
from PyQt5.Qt import QApplication, QClipboard
import numpy as np



class serialPlot:
    def __init__(self, serialPort='/dev/ttyUSB0', serialBaud=38400, plotL
ength=100, dataNumBytes=2, numPlots=1):
        self.port = serialPort
        self.baud = serialBaud
        self.plotMaxLength = plotLength
        self.dataNumBytes = dataNumBytes
        self.numPlots = numPlots
        self.rawData = bytearray(numPlots * dataNumBytes)
        self.dataType = None
        if dataNumBytes == 2:
            self.dataType = 'h'     # 2 byte integer
        elif dataNumBytes == 4:
            self.dataType = 'f'     # 4 byte float
        self.data = []
        for i in range(numPlots):   # hacer un arreglo para cada tipo de
dato
            self.data.append(collections.deque([0] * plotLength, maxlen=p
lotLength))
        self.isRun = True
        self.isReceiving = False
        self.thread = None
        self.plotTimer = 0
        self.previousTimer = 0


        print('Trying to connect to: ' + str(serialPort) + ' at ' + str(s
erialBaud) + ' BAUD.')
        try:
            self.serialConnection = serial.Serial(serialPort, serialBaud,
 timeout=4)
```

```
            print('Connected to ' + str(serialPort) + ' at ' + str(serial
Baud) + ' BAUD.')
        except:
            print("Failed to connect with " + str(serialPort) + ' at ' +
str(serialBaud) + ' BAUD.')

    def readSerialStart(self):
        if self.thread == None:
            self.thread = Thread(target=self.backgroundThread)
            self.thread.start()
            # se bloquea hasta que se reciban lecturas
            while self.isReceiving != True:
                time.sleep(0.1)

    def getSerialData(self):
        currentTimer = time.perf_counter()
        self.plotTimer = int((currentTimer - self.previousTimer) * 1000)
    # la primera lectura sera erronea
        self.previousTimer = currentTimer

        privateData = copy.deepcopy(self.rawData[:])    # sincronizar los
 valores a graficar
        for i in range(self.numPlots):
            data = privateData[(i*self.dataNumBytes):(self.dataNumBytes +
 i*self.dataNumBytes)]
            value,  = struct.unpack(self.dataType, data)
            self.data[i].append(value)    # anadimos el ultimo punto a nu
estro arreglo
            acelx = self.data[0][-1]
            acely = self.data[1][-1]
            acelz = self.data[2][-1]
            #print(acelx, acely, acelz)
        return [acelx, acely, acelz]
            #print(acelx, acely, acelz)
            #print([self.data[0], self.data[1][-1], self.data[2][-1]])



    def backgroundThread(self):    # recibir datos
        time.sleep(1.0)  # tiempo para recibir datos
        self.serialConnection.reset_input_buffer()
        while (self.isRun):
            self.serialConnection.readinto(self.rawData)
            self.isReceiving = True
            #print(self.rawData)

    def close(self):
        self.isRun = False
        self.thread.join()
```

```
        self.serialConnection.close()
        print('Disconnected...')



class Plots(object):
    def __init__(self):
        #Arduino
        self.portName = 'COM7'
        self.baudRate = 115200
        self.maxPlotLength = 100     # numero de puntos en el eje x de la
 grafica
        self.dataNumBytes = 2        # numbero de bytes en 1 dato
        self.numPlots = 3            # numbero de trazos en 1 gafica
        self.s = serialPlot(self.portName, self.baudRate, self.maxPlotLen
gth, self.dataNumBytes, self.numPlots)   # inicializar todas las variable
s
        self.s.readSerialStart()


#Variables
        self.i = 0
        self.acely = []
        self.acelx = []
        self.freq = 95              # 1/T   -> T  arduino delay 95
        self.guarda = 256             # Buffer para la FFT


        # pyqtgraph (inicializacion de graficas)
        #pg.setConfigOptions(antialias=True)
        self.traces = dict()
        self.win = pg.GraphicsWindow()
        self.win.setWindowTitle('FFT')
        pg.setConfigOption('foreground', 'w')


        self.p1 = self.win.addPlot(
            title="<span style='color: #ffffff; font-weight: bold; font-
size:20px'>FFT</span>")
        linea1 = pg.mkPen((0, 255, 0), width=2)  # style=QtCore.Qt.DashLi
ne)
        #linea2 = pg.mkPen((0, 0, 255), width=2)  # style=QtCore.Qt.DashL
ine)
        #linea3 = pg.mkPen((255, 0, 0), width=2)  # style=QtCore.Qt.DashL
ine)
        self.p1.addLegend(offset=(10, 5))

        self.curve1 = self.p1.plot(self.acely,
                        pen=linea1,
```

```
                        name="<span style='color: #ffffff; font-
weight: bold; font-size: 12px'>Aceleracion</span>")

        self.p1.setRange(yRange=[-2, 2])
        self.p1.setLabel('bottom',
                    text="<span style='color: #ffffff; font-
weight: bold; font-size: 12px'># de muestras</span>")
        self.p1.showGrid(x=True, y=False)

        self.win.nextRow()
        self.p2 = self.win.addPlot()
        linea4 = pg.mkPen((255, 0, 0), width=2)
        self.p2.addLegend(offset=(10, 5))

        self.curve2 = self.p2.plot(self.acely,
                        pen=linea4,
                        name="<span style='color: #ffffff; font-
weight: bold; font-size: 12px'>Amplitud</span>")

        self.p2.setRange(yRange=[0, 1000], xRange=[0, int(self.freq/2)+1]
)
        #p2.setRange(yRange=[0, 1000], xRange=[0, 100])
        self.p2.setLabel('bottom',
                    text="<span style='color: #ffffff; font-
weight: bold; font-size: 12px'>Frecuencia (Hz)</span>")
        self.p2.showGrid(x=False, y=True)




    def start(self):
        if (sys.flags.interactive != 1) or not hasattr(QtCore, 'PYQT_VERS
ION'):
            QtGui.QApplication.instance().exec_()
            self.s.close()

    def update(self):
        frecuencia = np.fft.fftfreq(self.guarda, d=1/self.freq)
        try:


            val = self.s.getSerialData()
            acelerometroy = val[0]/16384

            self.acely.append(acelerometroy)                        #anadi
r dato nuevo recibido
```

```
            mean_removed = np.ones_like(self.acely)*np.mean(self.acely)
#calcular media
            fft = self.acely - mean_removed
#remover media

            if self.i > self.guarda:
#esperar a tener 256 muestras
                try:
                    data = np.fft.fft(fft[-
self.guarda:])               #calcular fft
                    idx = np.argmax(np.abs(data))
                    f = frecuencia[idx]
                    freq_in_hertz = abs(f)
                    print(freq_in_hertz)

                except IOError:
                    pass
                amplitud = abs(np.real(data[:int(self.guarda/2)]))**2
            #calcular amplitud
                self.curve2.setData(frecuencia[:int(self.guarda/2)], ampl
itud)      #graficar amplitud y frecuencia

            self.curve1.setData(self.acely)     #graficar lecturas de ace
leracion
            self.curve1.setPos(self.i, 0)
            self.i += 1
            if self.i > self.guarda:
                del self.acely[0:1]            #eliminar el primer punto
cuando se superen 256 puntos

        except ValueError:
            pass

    def animation(self):                            #animar graficas
        timer = QtCore.QTimer()
        timer.timeout.connect(self.update)        #actualizar graficas
        timer.start(10)
        self.start()


# Start event loop.
if __name__ == '__main__':
    v = Plots()
    v.animation()
```