BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE INGENIERÍA
COLEGIO DE INGENIERÍA MECÁNICA Y ELÉCTRICA

“Interfaz para monitoreo de vibraciones mecánicas con 4 sensores MPU6050 y MATLAB en estructuras de 1gdl”

TESIS

PRESENTADA PARA OBTENER EL GRADO DE:

LICENCIATURA EN INGENIERÍA MECÁNICA Y ELÉCTRICA

PRESENTA:

Gerardo Alan Benítez Cruz

ASESOR:

Dr. FILIBERTO CANDIA GARCÍA

COASESOR

Dr. José Isrrael Rodríguez Mora

Puebla, Pue. Octubre 2022

Agradecimientos

A mis padres y mi hermana por todo el apoyo que me han brindado desde el momento en que inició todo, por su esfuerzo y sacrificio que han puesto para que esto sea posible.

A mi pareja, por su apoyo y motivación durante mi formación profesional.

A mi asesor y maestro, por su invaluable apoyo y orientación en la realización de este proyecto

A mis docentes, por su dedicación y paciencia.

Contenido

Resumen 5
Introducción 6
Capítulo I.- Protocolo de Investigación 8
Justificación 8
Objetivos 9
Hipótesis 10
Variables 10
Capítulo II.- Marco de Referencia 11
Marco Histórico 11
Estado del arte 12
Capítulo III.- Marco Metodológico 15
Diseño de la investigación 15
Tipo de investigación 15
Técnicas e Instrumentos 16
MATLAB 16
Simulink 16
Arduino 17
MPU6050 17
Comunicación Serial y comunicación I^2C 18
Impresión 3D y software CAD/CAM/CAE 18
Capítulo IV.- Desarrollo tecnológico 20
Conexión y comunicación serial Arduino Uno – Matlab 20
Script para obtener FFT de valores de aceleración detectados 23
Descripción general del script para obtener el espectro de frecuencias 23
Diseño de interfaz visual 26
Interfaces en Simulink 26
Interfaz de adquisición de datos de vibración 27
Interfaz de obtención de respuesta de vibraciones 27
Diseño de interfaz de usuario (UI) en GUIDE 27
Capítulo V.- Análisis y Resultados 28
Carga y uso de la interfaz de usuario 33

Procesamiento de datos de simulaciones previas. 33
Procesamiento de datos con simulación nueva. 37
Otras opciones 38
Discusión 39
Conclusiones 40
Bibliografía 42
Anexos 45
Código de interfaz gráfica de usuario 45

Resumen

El **antecedente** de este trabajo se enmarca en el estudio de la salud estructural, que es un término que se utiliza en evaluación y protección civil de las estructuras o de la industria de la transformación (automotriz, aeronáutica, aeroespacial, etc). Con la finalidad de determinar el estado de la integridad de la rigidez de estructuras con 1 grado de libertad (gdl), la cual se ve afectada por vibraciones eventuales y vibraciones por fenómenos operativos permanentes, generalizadas como vibraciones mecánicas aleatorias y no aleatorias.

El **objetivo** es proponer una interfaz gráfica de usuario funcional e intuitiva para el monitoreo de estructuras de un grado de libertad utilizando dispositivos de procesamiento de movimiento digital relacionados con el monitoreo de salud estructural.

La **metodología** es teórica-experimental, basada en el método científico-deductivo, los resultados se obtienen mediante manipulación de una estructura de 1 gdl y el análisis de las muestras de aceleración a través de una interfaz desarrollada en Arduino-Simulink-Matlab.

El **desarrollo** comprende la adquisición de señales por medio de acelerómetros MPU6050 y Arduino UNO, integrando una interfaz, que permite presentar, almacenar y graficar los datos adquiridos y procesados por medio de una aplicación elaborada en Simulink para obtener la respuesta del sistema. Con el apoyo del software MATLAB y Simulink se analizan los registros de aceleración adquiridos por medio de la transformada rápida de Fourier (FFT). Una vez determinado el espectro de las señales capturadas por el sensor se identifica la amplitud (pico) más alta y se determina la respuesta de las vibraciones detectadas en función de su masa y rigidez.

Como **resultado** se pretende que el sistema desarrollado sea capaz de mostrar e identificar perdidas de rigidez en estructuras de 1 gdl, asociadas al debilitamiento de los elementos de conexión y de su deformación geométrica, por medio de los cambios en la frecuencia de operación del sistema y facilitando el análisis del proceso mediante tareas automatizadas presentadas al usuario en paneles de botones dentro de la interfaz gráfica, con un diseño intuitivo y sencillo de utilizar.

Se busca **concluir** que es posible construir un sistema de bajo costo para el monitoreo de la salud estructural de estructuras de 1 gdl, evaluada en pérdida de rigidez y deformación geométrica, así como de establecer patrones de comportamiento mediante las gráficas de vibración generadas y con ello, que los estudiantes de ingeniería mecánica eléctrica y civiles, refuercen los conceptos aprendidos durante su formación referente al estudio de vibraciones mecánicas y con ello involucrarse en las problemáticas que métodos como el monitoreo de salud estructural pueden aportar a las soluciones que se requieren en el sector metal-mecánico y de la construcción.

Introducción

El monitoreo de salud estructural involucra la implementación de un sistema de detección de daños en ingeniería civil para la infraestructura, así como en la industria aeroespacial y mecánica. Las estructuras se deterioran bajo algunos eventos durante su tiempo de vida útil, por lo tanto, la detección de daños se torna esencial para tomar medidas de prevención lo antes posible y asegurar la integridad de los usuarios.

No se cuenta en la Facultad de Ingeniería de la BUAP, como en otras IES, con el equipamiento/dispositivos, para llevar a cabo el monitoreo de la salud estructural en función de la frecuencia natural, para estructuras de 1 gdl. Por lo que no se incluye/profundiza el tema del efecto negativo que ocasionan las vibraciones mecánicas, dentro de la formación profesional de los ingenieros Mecánicos y Eléctricos/Civiles.

Siendo esta debilidad una latente deficiencia curricular que conlleva a la comunidad de profesionistas a realizar suposiciones incorrectas, sobre el cómo optimizar las estructuras de 1 gdl, sin poner en riesgo su salud estructural. Poniendo de manifiesto la falta de un criterio valido cuando se requiere realizar una modificación a la función original de la infraestructura habitacional/comercial

La automatización y digitalización del monitoreo de salud estructural puede ser aplicado en estructuras cotidianas e industriales, por ejemplo, puentes, carreteras, edificios, plataformas petrolíferas, minas, etc. Para ello, se requiere de estudios especializados que involucran el monitoreo para predecir el comportamiento de dichas estructuras, mejorando el nivel de confiabilidad en las mismas. No obstante, la implementación de dichos sistemas que ofrece el mercado es extremadamente costosa, de ahí la importancia de desarrollar un sistema de bajo coste y fácil construcción que permita a los estudiantes relacionarse con el área y, con ello, fomentar el interés en el desarrollo e investigación de tecnología referente a esta área que permita poner al alcance de la población en general los beneficios que el monitoreo de salud estructural puede otorgar.

Por medio del método científico, se busca correlacionar el comportamiento de una estructura de 1GDL, que mediante excitación en sus elementos y por medio de acelerómetros sean registradas dichas excitaciones para posteriormente obtener el espectro de frecuencias, que permita calcular la respuesta del sistema para determinar el desplazamiento de los elementos estructurales y automatizar dicha tarea utilizando una interfaz gráfica diseñada específicamente para este procedimiento, optimizando el proceso para ser realizado de manera rápida y eficiente.

Que los alumnos comprendan la importancia del monitoreo de los parámetros mencionados, aprendan la importancia del procesamiento digital de señales, el uso de microcontroladores en la industria y de cómo se pueden diseñar soluciones de bajo coste para problemáticas que permitan modelar prototipos que en años próximos serán tendencia debido a la rápida

urbanización del entorno y de los problemas de seguridad en la infraestructura que esto implica, al comprender el uso de parámetros como la frecuencia natural del sistema, la frecuencia normal de operación, interpretar correctamente las gráficas de dichas frecuencias y entablar juicios y predicciones del comportamiento de las estructuras basados en los valores que sean detectados mediante este tipo de procedimientos.

Se concluye que las aportaciones realizadas por este proyecto en cuanto al monitoreo fomentado desde las Instituciones de Educación Superior (IES), permite proponer sistemas de bajo costo para vigilar la salud estructural de modelos de 1 gdl, evaluado en términos de perdida de la rigidez a través de las mediciones de la frecuencia del sistema, así como de la obtención de la respuesta de dichas vibraciones. Logrando con este trabajo transferir y aplicar la tecnología relacionada a las vibraciones mecánicas para la prevención de riesgos públicos.

Capítulo I.- Protocolo de Investigación

Contribuir al desarrollo de mecanismos y dispositivos que permitan incrementar la calidad de vida de la población de una comunidad. Es deber de las IES velar por la disminución de las afectaciones ocasionadas por grandes catástrofes como los sismos.

La pérdida del patrimonio es sin duda alguna una de las realidades, que disminuyen o detienen el progreso de una nación sobre todo en vías de desarrollo. Por lo tanto, estudiar las alternativas posibles, para disminuir esta afectación, será bienvenida por la comunidad y mejorará las funciones de dependencias gubernamentales como la protección civil.

Es entonces una meta palpable de este trabajo el desarrollo de una interfaz dirigida a la enseñanza de uso de parámetros relacionados con el monitoreo de salud estructural, y con ello incentivar la preservación del patrimonio residencial y comercial al conocer los elementos que posibilitan emitir juicios sobre las condiciones de seguridad de operación de una estructura. Como consecuencia, la creación de la interfaz se convierte en un dispositivo de enseñanza pragmática sobre los efectos de las vibraciones sobre estructuras afectadas en sus condiciones de operación normal, por cambios en su geometría o por efecto de vibraciones aleatorias.

Justificación

Desarrollar una propuesta de interfaz de gráfica con capacidad de mostrar parámetros pertinentes al monitoreo del estado de las estructuras de un grado de libertad (1gdl), aportando a la formación profesional de universitarios pertenecientes al área de ingeniería mecánica eléctrica e ingeniería civil, mediante la implementación de software especializado, posibilitando identificar, evaluar y analizar el comportamiento de un sistema al analizar la respuesta dinámica de una estructura ante fallas estructurales o funcionamiento inadecuado-inseguro, con base en el registro de la frecuencia natural de la estructura.

Una interfaz de usuario que reconozca la observación de variaciones en los registros de aceleración-frecuencia, ante vibraciones aleatorias o controladas es un recurso didáctico/formativo, que contribuye a establecer un sólido criterio, que priorice la consulta de un experto en diseño estructural antes de llevar modificaciones que pongan en riesgo los patrimonios personales y nacionales.

Gracias al avance tecnológico de las últimas décadas, es posible desarrollar un sistema de monitoreo de salud estructural a nivel didáctico que no represente una inversión incosteable debido a la necesidad de equipos especializados que serían inasequibles, sustituyendo algunos componentes por otros de fácil adquisición y manipulación como son los acelerómetros. Con ello, los estudiantes de Ingeniería Mecánica y Eléctrica/Civiles podrán ampliar sus conocimientos y capacidades en la toma de decisiones derivados del

comportamiento de las estructuras, así como evitar importantes pérdidas económicas derivadas de los tiempos muertos de operación en industrias con estructuras de 1gdl que requieren estar en funcionamiento constantemente, así como pérdidas humanas en caso de que una falla en la estructura no fuese detectada a tiempo, derivada del desgaste de los componentes en la edificación o de eventos no predecibles como algún evento sísmico, reforzando las estructuras en medida de lo posible, y diseñando estándares de operación y/o rutinas de mantenimiento preventivo según sea necesario tomando en cuenta los resultados de los análisis proporcionados por el sistema de monitoreo de salud estructural desarrollado.

Objetivos

General

Proponer un sistema capaz de automatizar el proceso de análisis digital de registros de aceleración (vibraciones mecánicas), mediante una interfaz que evalúe y muestre los resultados pertinentes a la frecuencia, amplitud y respuesta en estructuras de un grado de libertad de manera sencilla, fácil de replicar y de bajo coste, enriqueciendo el perfil de los egresados en ingeniería mecánica e ingeniería civil.

Específicos

- Desarrollar la interfaz de adquisición de vibraciones mediante Arduino, MATLAB Simulink y acelerómetros MPU6050
- Crear scripts dedicados al tratamiento de las señales transformando de la función tiempo a la función frecuencia en MATLAB.
- Desarrollar la interfaz que integre el registro de la adquisición de vibraciones, los resultados del análisis digital mediante la transformada rápida de Fourier y muestre los resultados de manera gráfica.
- Localizar valores de amplitud y frecuencia máximos mediante el espectro obtenido de las vibraciones graficadas en la interfaz.
- Realizar la simulación de escenarios en función de la variación de la frecuencia, que modifique el comportamiento de estructuras de 1 gdl, ante la presencia de cargas dinámicas.

Hipótesis

Contar con una interface capaz de visualizar el estado de una estructura de un grado de libertad que pueda representar los parámetros relacionados con su frecuencia de vibración, así como la amplitud de las mismas de forma semi automatizada, posibilita monitorear las estructuras de un grado de libertad y e inferir su estado basándose en la frecuencia de operación y en la amplitud de dichas frecuencias, además de la respuesta de las vibraciones del sistema determinada con los valores mencionados.

Variables

Los elementos que por la naturaleza de su condición pueden tratarse como variable independiente serán los elementos que conforman la estructura de un grado de libertad, los sensores utilizados para el monitoreo de las vibraciones y el equipo para el tratamiento de las señales detectadas. El comportamiento de la estructura de un grado de libertad, la frecuencia obtenida por el análisis digital de las señales, y la respuesta del sistema serán las variables dependientes

Capítulo II.- Marco de Referencia

Para los niveles educativos de pregrado y grado con referencia al área de conocimientos de la ingeniería mecánica y civil, es común que en su currículum definido en el sistema educativo mexicano, cuenten con asignaturas como: vibraciones mecánicas o dinámica estructural, las cuales se imparten mayormente de manera conceptual, pues se carece de los medios didácticos para exponer de manera experimental los efectos de las vibraciones en las estructuras de 1 gdl, esto por motivos relacionados con el costo que representa el implementar equipos especializados relacionados con el análisis de vibraciones.

Asociar los conceptos del comportamiento de las estructuras ante fenómenos como la vibración desde la perspectiva teórica y práctica, define los criterios de responsabilidad hacia los deberes de un ingeniero hacia la preservación de la sociedad, su bienestar y su autonomía económica.

Marco Histórico

El monitoreo de la condición tiene como objetivo inferir la salud de una máquina completa a partir de mediciones en un pequeño número de ubicaciones, y en la década de 1970, los investigadores comenzaron a buscar la posibilidad de evaluar la integridad de las estructuras a partir de un número igualmente pequeño de mediciones. El trabajo se concentró en las mediciones de vibración, ya que los modos de vibración son propiedades de toda la estructura, y las frecuencias de resonancia son independientes de la posición de medición, siempre que haya movimiento en el modo en el punto elegido. La principal motivación para el trabajo inicial del monitoreo de la salud estructural (SHM), fue evaluar la integridad de las plataformas petroleras en altamar en el Mar del Norte o el Golfo de México a partir de las mediciones de la respuesta de vibración a la excitación de las olas tomadas por encima de la línea de flotación. Sin embargo, se descubrió que las frecuencias de resonancia medidas eran sensibles a las condiciones ambientales, como el crecimiento marino que agrega una masa significativa a la estructura, y también a los cambios en la distribución de masa en la cubierta, y estos cambios efectivamente corrompieron los datos. Las pruebas también identificaron problemas de singularidad asociados con la predicción de daños si solo se usan frecuencias de resonancia, esta falta de éxito llevó a la industria petrolera a abandonar los estudios relacionados a mediados de la década de 1980 (Cawley, 2018).

Posteriormente han surgido gran cantidad de avances en el campo de investigación de SHM y estudio de vibraciones mecánicas en general, de los cuales se pueden mencionar los siguientes:

- Reemplazo de la evaluación subjetiva por medición de vibración electrónica y niveles cuantificados de severidad.

- Análisis de frecuencia, inicialmente a través de filtros analógicos, que permite una mejor identificación de la naturaleza de la falla.
- En la década de 1960, se desarrollaron sondas de proximidad sin contacto para permitir la medición del movimiento del eje de grandes turbomáquinas en relación con la carcasa del rodamiento. Aunque esto implica la modificación de la máquina para introducir la sonda, el método ahora se aplica universalmente.
- Acelerómetros que reemplazan los sensores originales de desplazamiento y velocidad.
- Análisis digital rápido de la transformación de Fourier (FFT) que ofrece un mejor análisis de frecuencia que los filtros analógicos originales.
- Análisis automatizado en computadoras locales que ayudan a la interpretación.
- Sensores instalados permanentemente e instrumentación para permitir el monitoreo continuo de maquinaria crítica. (Mitchell, 2007).

Las referencias históricas reconocen que es necesario el uso de acelerómetros y el análisis digital rápido de la transformada de Fourier (FFT). Por lo tanto, son métodos válidos para monitorear vibraciones mecánicas y el uso de estas dos técnicas integradas es viable para desarrollar un sistema que permita cumplir el propósito de monitorear el estado de la rigidez en una estructura de 1 gdl, y con ello garantizar la operación segura en la zona donde se encuentre instalado dicho sistema.

Por lo que se puede considerar que el SHM se encuentra en una etapa de desarrollo mucho más temprana que el monitoreo de máquinas rotativas, con una gran cantidad de investigación, pero con un número relativamente pequeño de aplicaciones industriales de rutina. La historia del monitoreo del estado de la máquina, la pequeña cantidad de sensores necesarios para una cobertura total y la gran base instalada en máquinas similares que facilitan las estadísticas de la flota son probablemente las principales razones de la gran cantidad de aplicaciones prácticas de monitoreo del estado de la máquina en comparación con el resto de SHM (Cawley, 2018).

Estado del arte

Si bien la referencia histórica establece que el SHM es un conocimiento que se obtiene con la experiencia profesional, es importante que se inicie desde la formación educativa superior en los campos de la ingeniería mecánica y civil, sobre todo donde la filosofía de la seguridad ha prevalecido sobre los enfoques de salvaguardar las vidas en el diseño de estructuras metálicas y compuestas. Asimismo, varias técnicas han surgido en los últimos años con la atención enfocada en estructuras inteligentes con capacidades de detección inherentes (Loutas & Panopoulou, 2012), por el amplio avance tecnológico que han tenido los sensores, las interfaces y el software de alto nivel, que permiten integrar soluciones de monitoreo.

Diversas interpretaciones sobre el SHM pueden ser referidas por cada profesionista con base a su experiencia, por lo tanto, para no ser ambiguos en el concepto se hace referencia a las definiciones de los siguientes autores:

- El monitoreo de la salud estructural (SHM) se ocupa de la detección, identificación y evaluación en tiempo real de la evolución de la seguridad y el rendimiento de las estructuras (Yuequan & Zhicheng, 2019).
- El SHM "es el seguimiento de las características estáticas o dinámicas de una estructura para identificar y localizar reducciones de rigidez para la detección de daños. Se utilizan diferentes indicadores de daños y cualquier indicador presenta ventajas y desventajas (Schommer & Nguyen, 2017).
- El SHM, "permite detectar daños en una etapa temprana, seguir su evolución y ayudar a reducir los costos y el tiempo de inactividad asociados con la reparación de condiciones peligrosas antes de una falla" (Sabato & Niezrecki, 2017).
- Una técnica de detección de daños es una parte esencial de cualquier sistema SHM. Su alcance consiste en la identificación de algunos parámetros estructurales y ambientales que deben ser monitoreados regularmente durante la operación de la máquina. Esta técnica debería poder distinguir si el daño está presente o no en la estructura, pero también debe ubicar y cuantificar el mismo daño. Un daño puede verse como un cambio en el material y/o las propiedades geométricas de una estructura, incluida la variación de las condiciones de contorno y las conexiones estructurales (Lorenzo, Manzato, & Peeters, 2017).

El resultado de la revisión del estado del arte del SHM, permite deducir que el desarrollo y la aplicación de tecnologías de detección innovadoras, así como las tecnologías relacionadas de procesamiento de señales y análisis de datos promoverán a gran y pequeña escala el avance de la tecnología SHM para una mejor evaluación del estado estructural, de seguridad y predicción de rendimiento (Chan, Li, & Caprani, 2018). Que pueden ser transferibles desde la formación en los niveles de pregrado y grado en ingeniería.

México es uno de los países con mayor actividad sísmica debido a su ubicación dentro del Cinturón de Fuego del Pacífico. La actividad sísmica es un peligro latente para esta región, así como la actividad volcánica que se deriva de ello. El crecimiento de la población incrementa el riesgo de que algún temblor o erupción volcánica derive en mayores pérdidas vitales y económicas, por este motivo, aunque no puede predecirse con exactitud cuándo ocurrirá algún desastre, es importante conocer el estado de las estructuras para poder reforzarlas en caso de ser necesario o de minimizar el daño si algún desastre llegara a presentarse. El monitoreo de salud estructural puede ayudar a facilitar dicha tarea, al conocer los parámetros sobre los cuales una estructura puede fallar, se pueden tomar medidas antes de la construcción de una nueva edificación o en su defecto, reparar y mejorar las condiciones de operación de dichas estructuras de ser posible, con el fin de evitar pérdidas mayores en caso de algún siniestro.

Concientizar a la población en general sobre los riesgos que conlleva la falta de métodos prácticos y económicos para realizar estudios que permitan conocer el estado de una estructura, es relevante para mantener la seguridad de la sociedad en general en caso de desastres naturales. Por ello, la elaboración de este proyecto puede ayudar a que los futuros profesionistas relacionados con la ingeniería estén conscientes de las aplicaciones y beneficios que el monitoreo de salud estructural puede brindar, aún en su modelo didáctico. Durante la formación profesional, muchas veces no se le da la debida importancia a este problema, la desinformación en estas áreas podría tener graves consecuencias. Integrar el monitoreo de salud estructural, conocer los parámetros bajo los cuales se pueden desarrollar condiciones de riesgo durante y después de alguna actividad sísmica, así como establecer planes de acción derivados de estos, son aspectos que deberían tomarse en cuenta desde la formación profesional.

Integrar un modelo de bajo coste utilizando acelerómetros, de fácil transporte y proporcionando un programa que permite aplicar la transformada de Fourier para mostrar de manera gráfica y comprensible la composición de las ondas que se generan a partir de las vibraciones al mostrar las frecuencias que las constituyen, cada una de ellas con su respectiva amplitud, y con ello determinar los parámetros necesarios para conocer el estado de una estructura, se vuelve un proyecto viable que permitiría a los estudiantes ampliar el alcance de sus conocimientos referentes al área, y con ello, pueden recibir una mejor preparación así como los docentes pueden también lograr un mayor interés en las ramas de la ingeniería que se relacionan con temas involucrados en la actividad sísmica, rigidez de estructuras, condiciones de suelo, etc.

Capítulo III.- Marco Metodológico

La metodología es teórica-experimental, basada en el método científico-deductivo. Los resultados se obtendrán mediante la adquisición de señales de vibración y el análisis digital de las muestras en una aplicación desarrollada en Arduino-Simulink-Matlab, los datos se comparan durante el comportamiento con respecto a la frecuencia, periodo natural y la respuesta de la estructura determinada de manera teórica.

Diseño de la investigación

1. Configurar la estructura a estudiar.
2. Asignar materiales y conforme a sus propiedades físicas y mecánicas.
3. Establecer las condiciones y precarga de las conexiones de sujeción.
4. Determinar la frecuencia natural de la estructura.
5. Adquisición y registro de los datos mediante gráficas y archivos de texto.
6. Aplicar la FFT a las muestras de aceleración adquiridas.
7. Análisis de las funciones de respuesta en frecuencia.
8. Cuantificación de los valores máximos
    a) Frecuencia
    b) Amplitud
9. Diseño de la interfaz gráfica para almacenar, cargar y automatizar el proceso de adquisición-análisis de las vibraciones muestreadas.
10. Correlación del estado operativo de la estructura de 1gdl, con respecto a los valores de riesgo de los esfuerzos y desplazamientos.

Tipo de investigación

Siendo el método científico deductivo la base de la investigación, su alcance es descriptivo-correlacional. El interés es describir la experimentación y correlacionar los efectos identificados, con la teoría que implica el análisis de las vibraciones y su efecto en estructuras metálicas de 1 gdl.

Técnicas e Instrumentos

Para el desarrollo del sistema propuesto se ha hecho mención del software Matlab, que será el eje principal en la creación de la interfaz gráfica y el análisis digital de las muestras de aceleración además de cargar y almacenar los valores que sean detectados mediante la placa Arduino UNO y los acelerómetros MPU6050, y posteriormente calcular la respuesta de las vibraciones por medio de Simulink.

Para el modelado de la estructura analizar se utilizó Siemens NX, un software CAM/CAD/CAE, donde se diseñó la estructura donde serían colocados los acelerómetros, además de que fue necesario para posteriormente, realizar la impresión 3D de los elementos, constituyéndose también como una parte importante del desarrollo del proyecto la aplicación de tecnologías de manufactura que muestran ser redituables para el desarrollo rápido de prototipos didácticos útiles en instituciones de educación superior.

Se describirán brevemente estos elementos para la comprensión del motivo de uso de cada uno de ellos durante el desarrollo y ejecución del proyecto.

MATLAB

MATLAB es un software de programación y cálculo numérico especializado para análisis de datos, desarrollo de algoritmos y creación de modelos que incluye un lenguaje de programación propio (MathWorks, 2022).Entre sus múltiples implementaciones, destaca la facilidad de creación de interfaces de usuario mediante GUIDE y App Designer, además de la capacidad de procesamiento numérico que hace posible el análisis digital de señales, lo cual hizo de MATLAB una herramienta imprescindible para el desarrollo del presente trabajo

Para la creación de la interfaz se utilizará GUIDE, un entorno con un conjunto de herramientas de diseño que simplifican el proceso de diseño y programación de interfaces de usuario de interfaces de usuario (UI) (MathWorks, 2022).

Simulink

Simulink es un entorno para diseño de sistemas basado en diagramas de bloque dentro de MATLAB, permite crear sistemas con modelos multidominio, realizar simulaciones previo a implementaciones en hardware y programar comportamientos de los sistemas (MathWorks, 2022) sin necesidad de escribir código, ya que este es generado automáticamente, dejando al usuario solo las opciones necesarias en cada elemento para establecer los parámetros necesarios sin ahondar en aspectos técnicos de programación. Integra librerías reutilizables con herramientas de modelado especializadas y que además puede ser ampliada mediante descargas directas en el navegador de complementos de MATLAB. Mediante Simulink se obtendrá la respuesta de las vibraciones recolectadas por el sistema.

Arduino

Arduino es una plataforma de código abierto, basado en el uso sencillo del hardware y software (Arduno, 2022). Las placas Arduino son de bajo coste, por lo que representa una opción viable en el desarrollo de proyectos, aunado a un entorno de programación sumamente documentado. Al ser un software open-source, las capacidades nativas del entorno de programación pueden ampliarse mediante la descarga de librerías de terceros, ampliando el alcance de las aplicaciones posibles en el uso de sus placas.

La versión utilizada para este proyecto será la placa Arduino UNO Y el software Arduino IDE 1.8.19. Para que la placa reconozca los módulos MPU6050, es necesario instalar la librería para dichos elementos mediante el repositorio desarrollado por terceros presente en GitHub.

Figura 1. Arduino UNO.

MPU6050

La adquisición de los registros de aceleración requiere de sensores que permitan la transmisión en tiempo real con un tiempo de muestreo corto, con el fin de evitar pérdidas de información relevante durante la toma de muestras.

El acelerómetro MPU6050 representa una opción viable debido a la facilidad de adquisición de los módulos. Es una unidad de medición inercial de 6 grados de libertad, combinando un acelerómetro y un giroscopio de 3 ejes en un solo chip, además de integrar un procesador digital de movimiento (DMP) capaz de realizar algoritmos complejos de captura de movimiento de 9 ejes (Naylamp Mechatronics, 2021).

El rango de trabajo del sensor comprende una escala programable de 250-2000 grados/segundo para el giroscopio y de 2g/4g/8g/16g para el acelerómetro, utilizando una alimentación de 3.3 V o 5 V

Figura 2. Acelerómetro y giroscopio MPU6050.

Comunicación Serial y comunicación I^2C

La comunicación serial permite la transmisión y recepción bit a bit de un byte completo. Regularmente este tipo de comunicación es usado para la transmisión de datos en formato ASCII. Para ello se vale de 3 líneas de transmisión: referencia (tierra), recepción y transmisión. Las velocidades de transmisión comunes (baud rate) para la comunicación serial son 4800, 9600 y 115200 baudios, pueden utilizarse velocidades más altas, pero la distancia máxima entre dispositivos a comunicar se vería afectada negativamente (Tecnológico Nacional de México, 2022).

La comunicación de dispositivos por este método es asíncrona, lo cual permite que puedan enviar datos por un lado y recibir datos por otra vía al mismo tiempo, a comparación del protocolo I^2C, que se realiza de manera síncrona.

I^2C (inter-intergrated circuit) se refiere a un protocolo de comunicación síncrono, que utiliza dos líneas de transmisión, uno para el reloj (SCL) y otro para datos (SDA). Es un protocolo de direccionamiento, por lo que los dispositivos conectados por este método tienen una dirección única de 7 bits para cada uno, estos se diferencian entre dispositivos maestros y dispositivos esclavos (Rivero, 2018).

Debido al direccionamiento que utiliza este protocolo y las rutas utilizadas por los acelerómetros MPU6050 (0x68 y 0x69), la conexión de múltiples dispositivos se complica, y deja de ser una opción viable para el desarrollo del proyecto, por lo que se decidió utilizar el protocolo de comunicación serial.

Impresión 3D y software CAD/CAM/CAE

También conocida como manufactura por adición, se refiere a un conjunto de procesos que producen objetos a través de la adición de material en capas que corresponden a las sucesivas secciones transversales de un modelo 3D. Los plásticos y las aleaciones de metal son los materiales más usados para impresión 3D (Autodesk Inc., 2022). La impresión 3D ha sido utilizada desde hace algunos años para producir prototipos y modelos visuales de manera rápida, así como ensambles que facilitan el entendimiento de la función de algunos mecanismos.

Para desarrollar el modelo de la estructura de un grado de libertad, se utilizó el software SIEMENS NX, que entre sus múltiples aplicaciones, está el desarrollo de modelos mediante dibujo asistido por computadora y manufactura asistida por computadora, siendo la herramienta principal para el desarrollo de las placas que conforman la estructura donde serán posicionados los módulos MPU6050, las piezas donde serán colocados los elementos de sujeción realizando las adecuaciones necesarias mediante el software y finalmente preparar los prototipos para poder ser impresos en 3D.

Capítulo IV.- Desarrollo tecnológico

Para el desarrollo del sistema propuesto se tomaron en cuenta los siguientes aspectos:

- El tipo y la gravedad de los defectos u otros problemas que el sistema debe ser capaz de detectar.
- La recolección, el tiempo y la automatización del tratamiento de los datos.
- Análisis de los datos resultantes.
- Interacción entre el usuario y la interfaz gráfica
- Validación de los resultados.
- Qué decisiones deberán tomarse dependiendo del análisis de los resultados obtenidos.

Figura 3. Esquema simplificado de conexión entre tecnologías.

El sistema de monitoreo desarrollado puede resolver estos aspectos, la adquisición y análisis de los datos se facilitará al integrar todos los parámetros dentro de una interfaz gráfica de fácil manipulación para el usuario final, sin requerir de una inversión significativa que representaría el adquirir equipamiento especializado que ya existe en el mercado, pero que, debido a su costo, no sería de implementación viable para las IES.

Conexión y comunicación serial Arduino Uno – Matlab

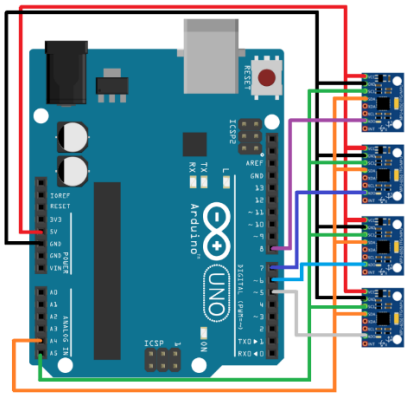

Figura 4. Diagrama de conexión Arduino uno – 4 MPU6050

Después de realizar pruebas de conexión y comunicación hacia equipo de cómputo, la conexión adecuada para adquirir las señales de los 4 acelerómetros es la mostrada en la figura.

Para validar la adquisición de datos utilizando esta conexión, se utilizó el software de Arduino. Mediante las herramientas “Serial plotter” y “Monitor serie” integradas en el software es posible identificar si los datos de aceleración están siendo recolectados adecuadamente.

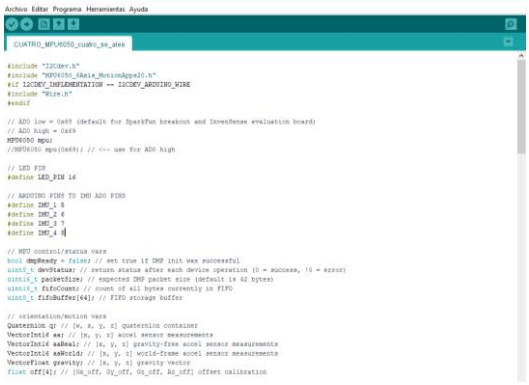

Figura 5 Código de muestreo en Arduino para 4 MPU6050.

Durante las simulaciones realizadas, se ha utilizado el puerto serie “COM3”, tanto para Arduino, como para Matlab, este puerto puede variar dependiendo del equipo que se utilice y deberá tomarse en cuenta este detalle si llegasen a presentarse problemas en caso de que la placa Arduino no sea detectada, las simulaciones no se ejecutarán si el puerto no está configurado correctamente. Es importante resaltar que son necesarias librerías extra que no están incluidas en una instalación común de Matlab, pero es posible descargarlas para su utilización mediante el navegador de complementos integrado en Matlab.

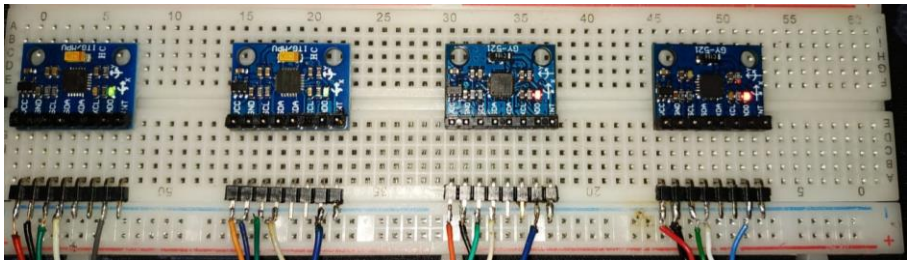

Figura 6. Conexión de 4 MPU6050.

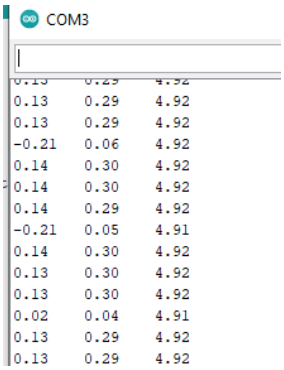

Figura 7. Registro de datos de aceleración en Monitor serie de Arduino.

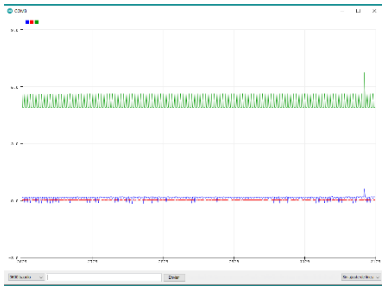

Figura 8. Registro de datos de aceleración en Serial plotter de Arduino.

Es importante que el cableado entre los acelerómetros y la placa Arduino Uno esté conectado de manera adecuada, si existe falso contacto en alguna de las conexiones esto da lugar a lecturas erróneas en los valores de aceleración debido a una lectura de 0V y después a otro valor aleatorio cuando se restablezca la conexión, dando como resultado picos que podrían indicar falsos positivos en ciertas condiciones de trabajo, por ello todas las conexiones deben realizarse de manera que, aunque exista movimiento en los sensores, no haya momentos en los que se desconecten por ningún motivo, de esta manera, podemos confiar en los datos que se hayan registrado por cada uno de los acelerómetros sin importar si alguno de los sensores experimentó algún movimiento brusco durante la simulación de la aplicación desarrollada en Simulink.

Script para obtener FFT de valores de aceleración detectados

Para obtener datos de los acelerómetros, se necesita hacer una simulación previamente en Simulink antes de ejecutar el script en Matlab, ya que las variables que se utilizarán en el código sólo se generan después de la ejecución en la interfaz creada en Simulink, o en su defecto pueden guardarse datos de simulaciones anteriores y cargarlos posteriormente en el entorno de trabajo de Matlab para realizar los análisis correspondientes a cada señal; este último proceso también se incluirá en la interfaz desarrollada.

En ocasiones, la potencia de los diferentes procesadores en los equipos hace que las simulaciones realizadas se vean afectadas, tanto en rendimiento como en la carga de estos. Matlab crea archivos temporales en cada ejecución de script y de simulación en los diagramas de bloques en Simulink, por ello, las primeras ejecuciones del código pueden llegar a ser más lentas de lo normal, sin embargo, esto se soluciona después de las primeras ejecuciones en ambos entornos, para el tratamiento de las señales captadas por los acelerómetros MPU6050.

Descripción general del script para obtener el espectro de frecuencias

La adquisición de señales por los acelerómetros requiere de un tratamiento de estas para poder identificar eventos importantes durante la ejecución. Esto es necesario para aplicar filtros y trabajar con los registros de aceleración y pasarlos al dominio de la frecuencia por medio de la aplicación de la transformada rápida de Fourier.

En primera instancia se obtienen los valores de aceleración y tiempo de la última ejecución del programa. Estas variables son almacenadas en el entorno de trabajo de Matlab, y quedan disponibles para su tratamiento mediante scripts previamente realizados. De esta manera, pueden ser recuperados y analizados de la forma que se requiera.

Se generan gráficas de los valores de aceleración en los 3 ejes de los sensores (X, Y, Z) y el tiempo en que cada señal fue tomada, posterior a esto, se deben ajustar las dimensiones de las matrices de datos generadas para continuar con el proceso de tratamiento de señales.

Para obtener el espectro de frecuencias de los valores de aceleración registrados, es necesario establecer la frecuencia de muestreo (Fs). Para configurar este parámetro se debe tomar en cuenta las limitaciones del propio sensor (1kHz para el acelerómetro y 8 kHz para el giroscopio). Establecer una frecuencia de muestreo incorrecta dará paso a pérdidas de información importantes y, por lo tanto, resultados erróneos o inconsistentes.

Con la frecuencia de muestreo establecida, ahora se deben declarar las variables requeridas para la aplicación de la transformada rápida de Fourier (FFT) y con ello determinar el espectro de frecuencias, así como los valores RMS de las vibraciones registradas.

```
%%Gráfica de aceleración EjeX vs tiempo
%Asignación de valores para eje X tomados desde la última simulación
ejex=squeeze(EjeX.signals.values);
%Asignación de tiempo correspondiente a cada valor de aceleracíon en X
t=EjeX.time;
%Ventana donde se mostrará la gráfica con los datos Aceleración X vs tiempo
figure('Name','Aceleración Eje X vs tiempo')
%Datos de las variables a graficar
plot(t,ejex)
%Título de la gráfica
title('Registro de aceleración EjeX vs tiempo')
%Etiqueta en el eje X de la gráfica
xlabel('tiempo (s)')
%Etiqueta en el eje Y de la gráfica
ylabel('Aceleración EjeX ()')
%Habilitar opción de "mallado" en la gráfica
grid on
```

Figura 9. Código para graficar muestras de aceleración.

Se especifica la frecuencia de muestreo para realizar el tratamiento de las señales obtenidas por el acelerómetro, para configurar este parámetro debe tomarse en cuenta también la frecuencia con la que el sensor puede recolectar información, de lo contrario puede perderse información relevante al intentar analizar la señal mediante la aplicación de la transformada rápida de Fourier (FFT)

Los acelerómetros censan las vibraciones de forma combinada, dando como resultado formas de onda compuestas o la sumatoria de cada una de las fuentes de vibración, para poder separar la energía mecánica entregada por cada fuente de una forma de onda compuesta, se utiliza un proceso matemático llamado transformada de Fourier, que separa cada onda o fuente de vibración con base a su frecuencia, obteniendo un gráfico que conocemos como espectro de vibración. Con esta función de Matlab, podremos conocer el espectro de frecuencia para cada eje y representarlo debidamente en términos de amplitud y frecuencia.

El espectro es el gráfico más utilizado para el diagnóstico de fallas en máquinas cuyos ejes están soportados por rodamientos, gracias a que a través de la frecuencia podemos identificar la fuente de vibración, así como su amplitud o severidad. Esto también es aplicable en el monitoreo de salud estructural, ya que se utilizarán los parámetros de amplitud, frecuencia y respuesta de las vibraciones detectadas, y con base en ellos, interpretar los resultados de las diferentes simulaciones adecuadamente.

```
%Frecuencia de muestreo
Fs = (1/(t(2)-t(1)))*100; %1000;
%Aplicación de Transformada rápida de Fourier para valores de aceleración
%detectados en eje X
Nx = length(t);
    freqx = 0:Fs/length(t):Fs/2;
    xdft = fft(ejex);
    xdft = xdft(1:floor(Nx/2)+1);
    xdft = 1/length(t).*xdft;
    xdft(2:end-1) = 2*xdft(2:end-1);
    xxdft = 1/(length(ejex)*Fs)*abs(xdft).^2;
    xdft = xdft';
    phase = unwrap(angle(xdft));
    xdft = abs(xdft);
```

Figura 10. Código para aplicación de transformada rápida de Fourier.

- Obtención de valores RMS de las vibraciones

Los valores más utilizados para la determinación de valores de alarma en vibraciones mecánicas son los RMS, debido a que están normalizados y que en la práctica descartan picos atípicos o transitorios en un instante del tiempo.

El valor global de RMS en términos simples es aproximadamente el 70% de la amplitud cero-pico, que al ser calculada por los colectores o software de vibración descarta picos asociados a impactos temporales convirtiéndose en un buen referente de la generalidad de las fuerzas destructivas que operan sobre una máquina y el más utilizado en programas de monitoreo de condición por vibraciones.

Con la obtención de estos valores, podemos analizar las tendencias de las vibraciones cuando los equipos se encuentran en condiciones óptimas de operación, si necesitan mantenimiento, o en casos más graves, si es necesario realizar una parada de emergencia para evitar daños mayores.

```
%% RMS De FFT
    %Determinar variables y mostrar tamaño de variables
    R_1=length(xdft');
    %Donde:
    %R= Row(Fila)
    %C= Column(Columna)
    t_FFT = freqx'; %tiempo en segundos
    x_FFT = xdft'; %Matriz de datos para RMS y FFT en eje X
    y_FFT = ydft'; %Matriz de datos para RMS y FFT en eje Y
    z_FFT = zdft'; %Matriz de datos para RMS y FFT en eje Z
    Fs_1 = 1/(t_FFT(2)-t_FFT(1));

    %Determina RMS y graficar para valores en eje X
    w_1 = floor(Fs_1); %Ancho de la ventana para calular RMS
    steps_1 = floor(R_1/w_1); %Numero de pasos para RMS
    tx_RMS_FFT = zeros(steps_1,1); %Crear matriz para los valores de tiempo de RMS
    x_RMS_FFT = zeros(steps_1,1); %Crear matriz para valores RMS
    for i=1:steps_1
        range_1 = ((i-1)*w_1+1):(i*w_1);
        tx_RMS_FFT(i) = mean(t_FFT(range_1));
        x_RMS_FFT(i) = sqrt(mean(x_FFT(range_1).^2));
    end
```

Figura 11. Código para determinar valores RMS.

Diseño de interfaz visual

El diseño de la interfaz constará de tres apartados para otorgar al usuario un manejo de la información de manera sencilla. El primero se encargará de la lectura y almacenamiento de las muestras, el segundo se ocupará de calcular la respuesta de las vibraciones detectadas y así mandarlas a la interfaz principal elaborada en GUIDE de Matlab, abordada en el tercer apartado del diseño de la interfaz gráfica. Los modelos elaborados deberán estar almacenados en la misma ubicación que la interfaz principal y los scripts, esto es necesario ya que de almacenarse en ubicaciones distintas, la ejecución del programa no será posible.

Interfaces en Simulink

Para poder adquirir las señales de los acelerómetros, y guardar variables para tratarlas posteriormente con scripts de MATLAB es necesario crear un modelo en Simulink que interactúe con el entorno de trabajo principal.

Los modelos por utilizar deben contener bloques que permitan comunicarse entre la placa Arduino y Matlab. Simulink también proporciona bloques que funcionan como osciloscopios, los cuales permiten visualizar las aceleraciones en cada eje de cada sensor durante el tiempo que esté activa alguna simulación, por ello se implementarán en los diagramas de bloques para monitorear que los sensores estén trabajando correctamente, que las señales sean consistentes y estén escaladas de manera adecuada para que las variables

enviadas a Matlab proporcionen resultados válidos y verificables con el script que muestra los espectros de frecuencia.

Interfaz de adquisición de datos de vibración

Para el desarrollo de la rutina de adquisición de datos se diseñó un sistema en MATLAB que utiliza las librerías referentes a dispositivos conectados mediante el puerto serie (COM3 para el ejemplo). El objetivo de este script es que mediante el establecimiento de comunicación entre el equipo de cómputo y la placa Arduino UNO, sean recolectadas las muestras provenientes de los acelerómetros MPU6050, y se almacenen por medio de variables que serán enviadas al entorno de trabajo principal de MATLAB para su análisis por medio de la transformada rápida de Fourier, reacomodando los datos recopilados en arreglos numéricos que contengan un valor de aceleración con el tiempo respectivo en que se tomó esa muestra y que sean capaces de ser procesados por los scripts implicados en la generación de las gráficas de aceleración, frecuencia y respuesta de las señales adquiridas.

Interfaz de obtención de respuesta de vibraciones

Para complementar el procesamiento de los datos de vibración por la transformada rápida de Fourier que se mostrará en la interfaz principal, se diseñó otro sistema en Simulink para obtener la respuesta de las vibraciones detectadas, que trabajará con datos recientes obtenidos de la última simulación ejecutada, o con datos que hayan sido guardados con anterioridad por la interfaz de usuario creada en GUIDE.

Este modelo no interactúa directamente con los sensores, en cambio, utiliza las variables almacenadas y las procesa para posteriormente ser graficadas en la interfaz principal.

Los bloques responsables de indicar la masa y la rigidez para este modelo pueden ser configurados directamente desde la interfaz de usuario principal, sin necesidad de modificar bloque por bloque la configuración para cada eje de todos los acelerómetros, los resultados se mostrarán en gráficas en su apartado correspondiente, con posibilidad de abrirse en una vista rápida o en ventanas separadas para mayor detalle.

Diseño de interfaz de usuario (UI) en GUIDE

Matlab dispone de herramientas capaces de crear aplicaciones nuevas a partir de programación orientada a objetos, esto con el objetivo de que el usuario pueda diseñar sus propias aplicaciones a partir de sus necesidades. La creación de programas desde Matlab permite utilizar scripts y modelos creados en Simulink e integrarlos dentro de una sola aplicación para facilitar su comprensión y manejo al usuario que no está tan familiarizado con el entorno de trabajo nativo de Matlab.

Gracias a la función “GUIDE” es posible crear una interfaz gráfica que sea de fácil manejo para el usuario final, así como simplificar ciertos procesos que son repetitivos al ponerlos bajo control de una aplicación desarrollada con el objetivo de mostrar los datos obtenidos de

una simulación en Simulink y los resultados del análisis digital de las señales realizado desde los scripts que se han desarrollado para el sistema propuesto.

Capítulo V.- Análisis y Resultados

Para facilitar el análisis de los datos obtenidos a partir de las simulaciones, se creó una interfaz para que se muestren los resultados de la ejecución del modelo creado en Simulink. Consiste en varios botones programados para mostrar las gráficas generadas gracias a los scripts desarrollados anteriormente.

La evaluación del rendimiento de la interfaz, así como de las simulaciones y de la fiabilidad de los resultados se hará montando los sensores en marcos diseñados para esta tarea.

Para ello se ha hecho uso del software Siemens NX, realizando el modelado en 3D de todos los componentes que se utilizarán en el modelo físico y generando el ensamble, de esta manera, mediante simulaciones en el software CAE, podrán validarse los resultados que se generen en los espectros obtenidos de las señales recopiladas por los acelerómetros MPU6050.

Se realizarán pruebas en 2 modelos distintos, los cuales se muestran a continuación:

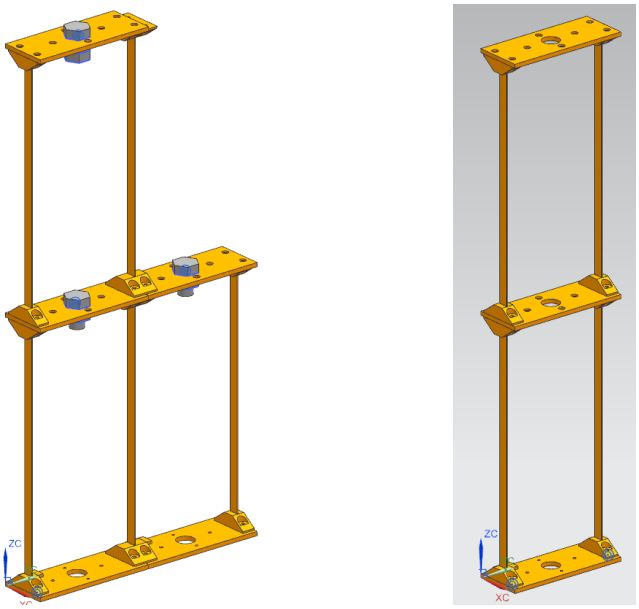

Figura 12. Modelos en 3D de las estructuras para pruebas y validación de resultados.

Los sensores serán colocados en posiciones determinadas previamente en el diseño de la estructura, como se muestra en la siguiente imagen:

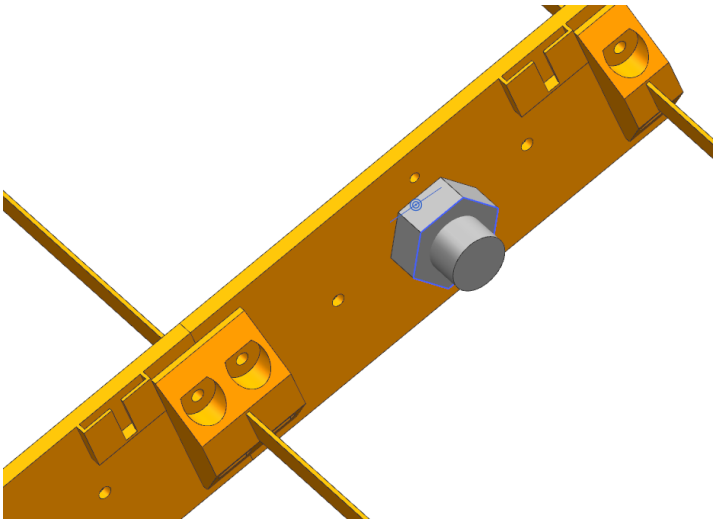

Figura 13. Ubicación de sensores MPU6050 en estructuras de 1GDL.

Como puede apreciarse en las figuras, el sistema creado es sencillo de replicar y los elementos están disponibles para impresión 3D, por lo que implementar este modelo junto con los sensores MPU6050, no requiere de una inversión alta. Para propósitos educativos, el sistema es eficiente para ayudar a los alumnos a valorar y comprender los beneficios que el monitoreo de salud estructural puede otorgar en el diseño y análisis de estructuras, así como generar interés en como las vibraciones mecánicas pueden ayudar a optimizar el diseño de estas, con el fin de incrementar el tiempo de vida útil de los proyectos que se realicen a futuro, y salvaguardar la integridad de los usuarios.

Las piezas para el ensamble de los modelos se imprimen en 3D y quedan de la siguiente forma:

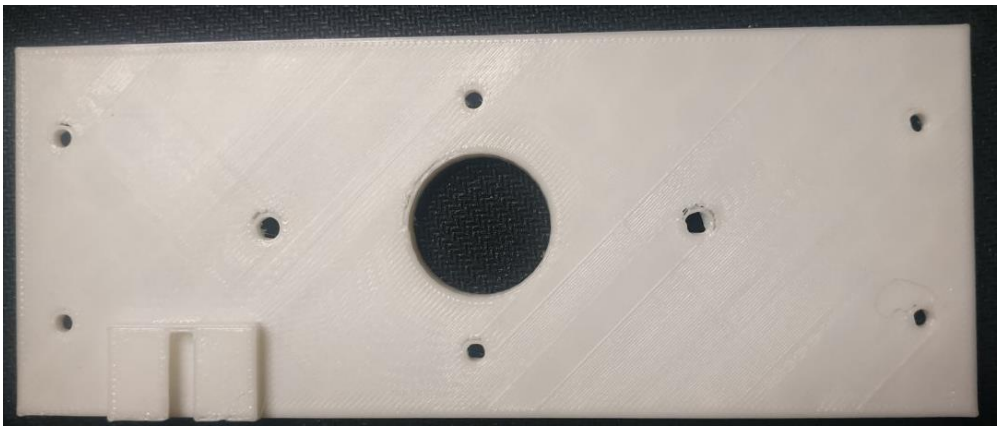

Figura 14. Bases en 3d con espacio para insertar sensores MPU6050.

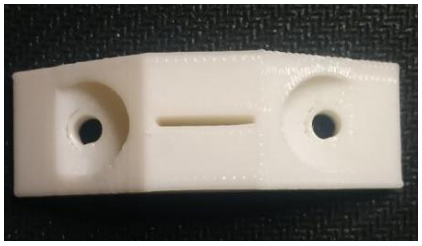

Figura 15. Unión simple para placas de la estructura.

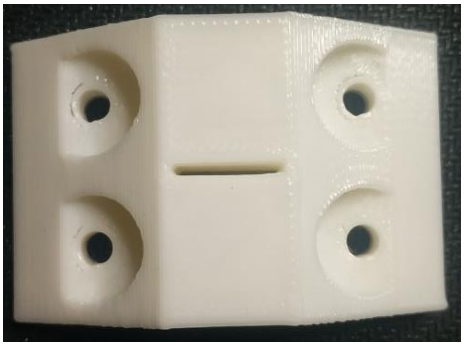

Figura 16. Unión doble para placas de la estructura.

Figura 17. Elementos de sujeción utilizados.

El ensamble de las estructuras es muy sencillo, consta de utilizar las uniones simples o dobles según sea el caso de la estructura a analizar.

Figura 18. Estructuras ensambladas con piezas impresas en 3D.

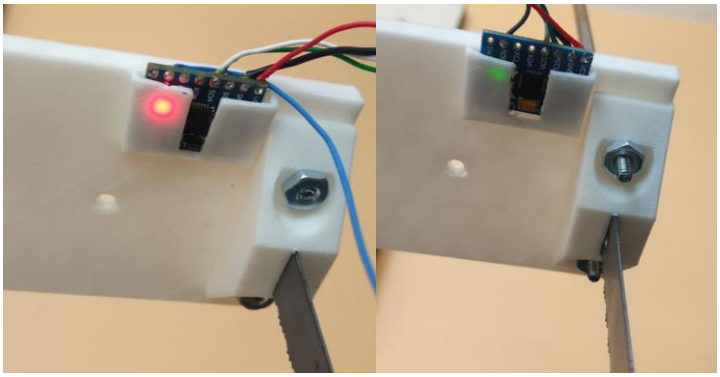

Figura 19. Sensores conectados y anclados a estructura (el LED indicador puede variar su color dependiendo del fabricante).

Figura 20. Masa colocada en estructura.

Para validar los resultados de las simulaciones también se agregarán masas en la estructura de 1gdl como en la figura anterior. Esto con el fin de comprobar el comportamiento de las señales capturadas por los acelerómetros en diversas condiciones, cambiando de posición las masas y con ello la manera en que se registra la respuesta de las vibraciones mediante los scripts desarrollados para la interfaz gráfica diseñada.

Siendo un modelo didáctico, fácil de replicar y de bajo coste, su implementación en los programas referentes a vibraciones mecánicas y seguridad estructural es factible, ayudando a los alumnos a reforzar los conceptos estudiados en asignaturas relacionadas con el análisis estructural y vibraciones mecánicas, así como despertar el interés en esta rama de la seguridad estructural, ya que, en años próximos, será un referente en investigación y una herramienta muy valiosa para salvaguardar la infraestructura.

Carga y uso de la interfaz de usuario

La interfaz puede trabajar con resultados de una simulación de Simulink que se ejecute desde la misma interfaz o trabajar con resultados guardados. El procedimiento para el tratamiento de los valores de aceleración es similar.

Procesamiento de datos de simulaciones previas.

1. Dar clic en el botón “Cargar” ubicado en la parte inferior izquierda de la interface

2. Se desplegará una ventana para cargar el archivo .m con los datos requeridos, se selecciona y se da clic en abrir.

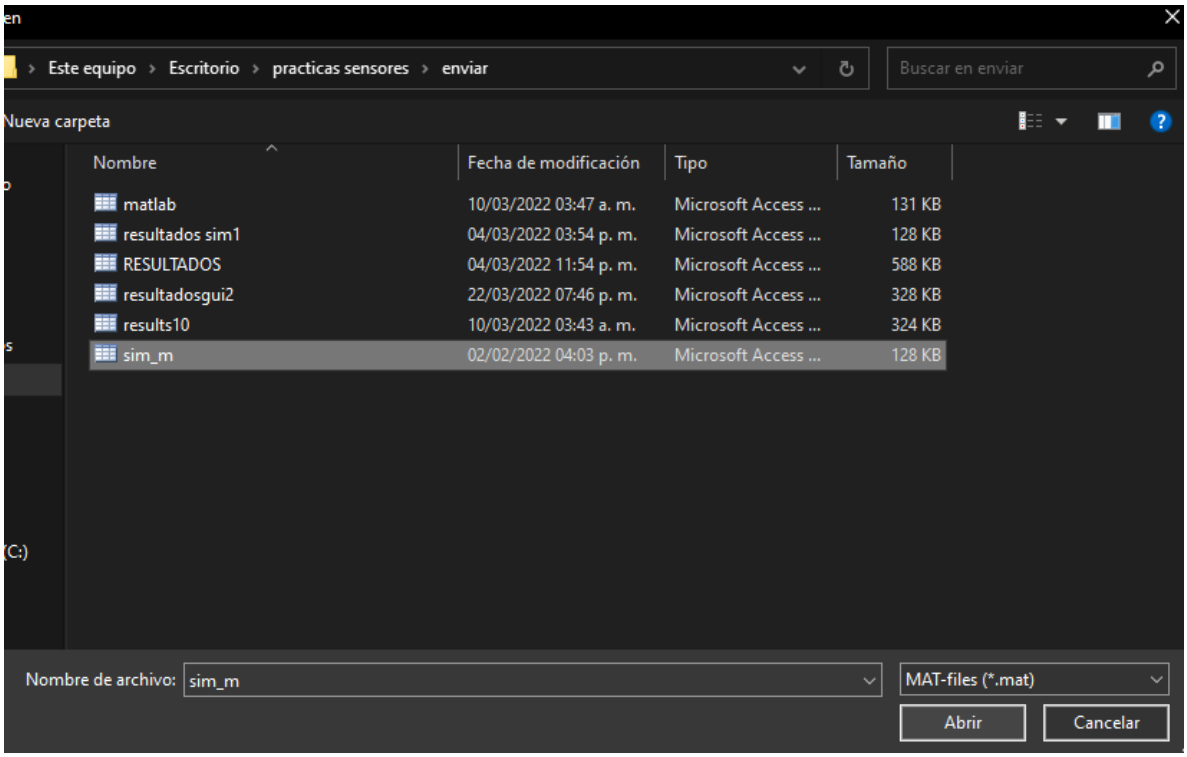

Figura 21. Carga de archivos .m previous de simulación.

3. Con los datos cargados, es importante presionar primero el botón “Confirmar adquisición de datos” antes de utilizar algún botón referente a la generación de gráficas, esto con el objetivo de evitar fallas al ejecutar la simulación del modelo de respuesta así como de la obtención correcta de las gráficas.

4. Podemos cerrar las gráficas que aparezcan en pantalla o dejarlas si se desea observarlas con mayor detalle.

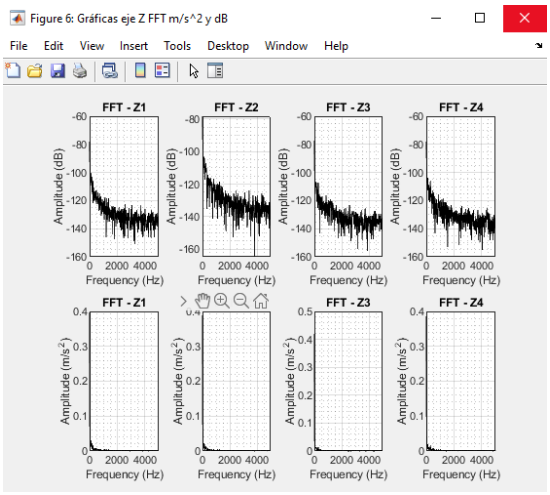

Figura 22. Gráficas resultants por aplicación de FFT a registros de aceleración.

5. Si se desea que los resultados se muestren directamente en la interface, se utilizan los botones del panel llamado “Graficar en GUI”. Los subpaneles contienen la información referente a las gráficas que serán mostradas en la GUI.

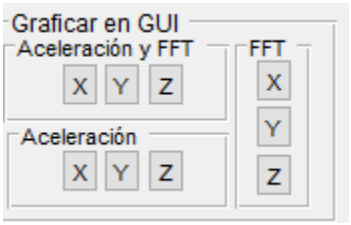

Figura 23. Panel de control de generación de gráficas en IU.

6. Los resultados se mostrarán ahora en la gui, viéndose de la siguiente manera:

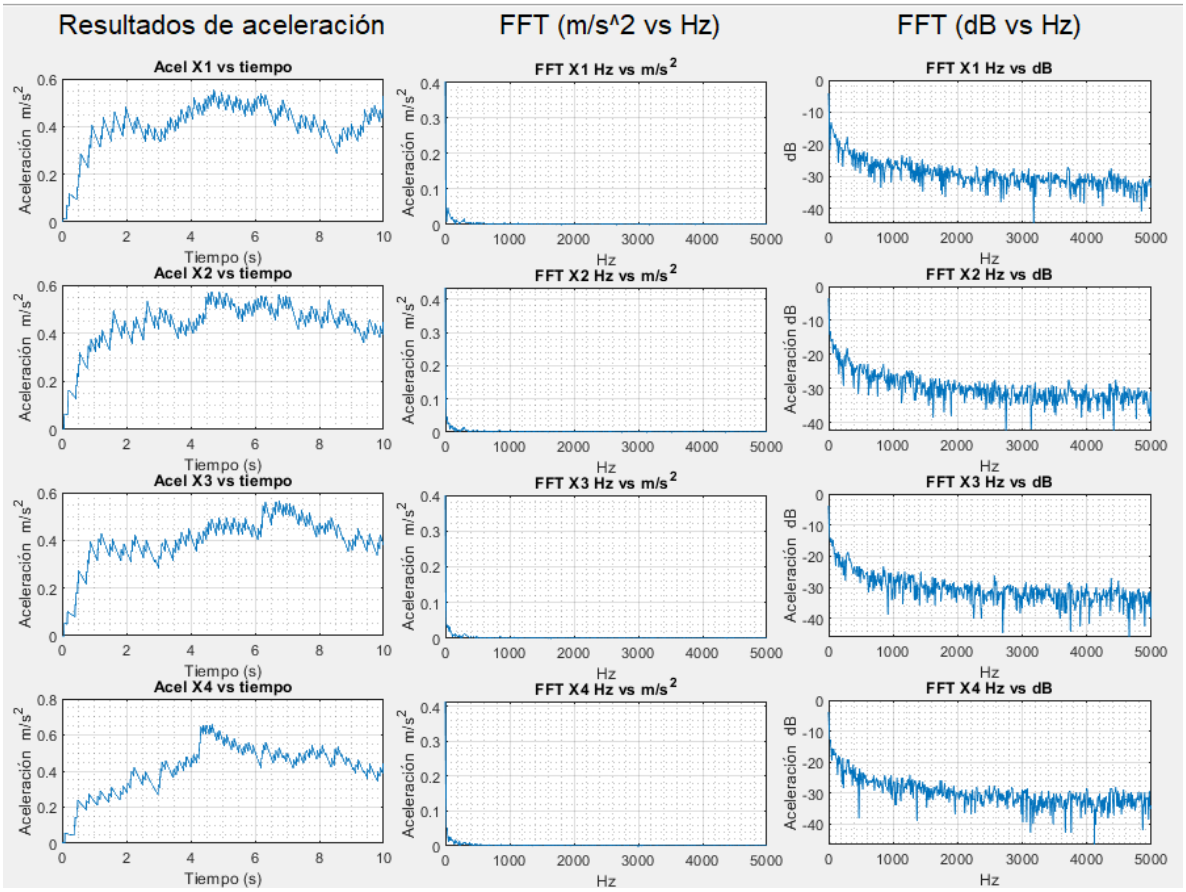

Figura 24. Gráficas de resultados en IU.

7. El panel “Gráficas detalladas” puede mostrar cada gráfica por separado en una ventana más grande, mejorando el modo de visualización de los resultados y añadiendo opciones extra con las herramientas propias de Matlab para las ventanas generadas con gráficas.

   Al igual que el panel anterior, los botones muestran la información necesaria para seleccionar la información que se requiera.

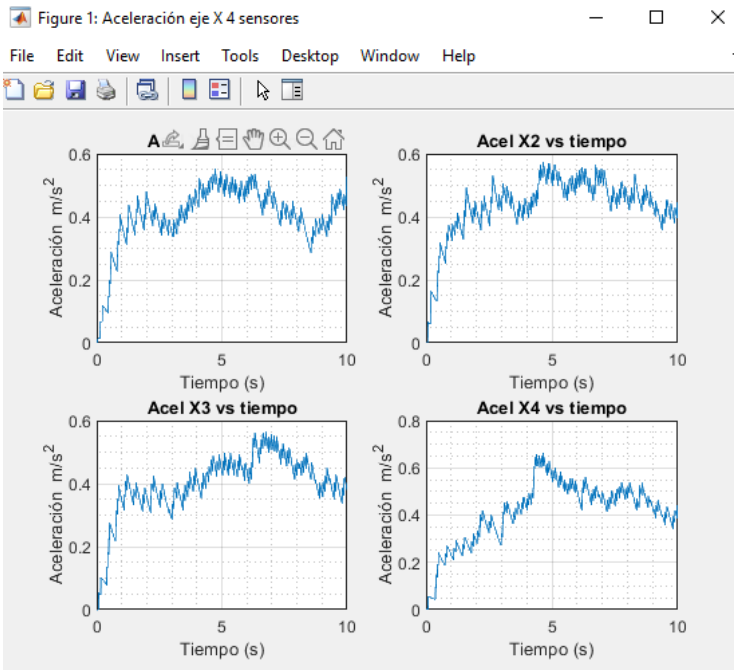

Figura 25. Gráfica independiente de IU para detalle de registros de aceleración.

8. El botón “Graficar todos los resultados” muestra los resultados del tratamiento de las señales adquiridas por los 4 sensores, referentes a los datos de aceleración y de la FFT calculada en ventanas separadas por cada eje de cada acelerómetro.

9. Para obtener la respuesta de las señales de aceleración, se da clic primero en el botón “Cargar modelo”, y se desplegará una ventana de Simulink con el modelo correspondiente, Podemos configurar la simulación desde ahí o desde la GUI, donde estarán únicamente los controles necesarios para manipular los parámetros relevantes de la simulación.

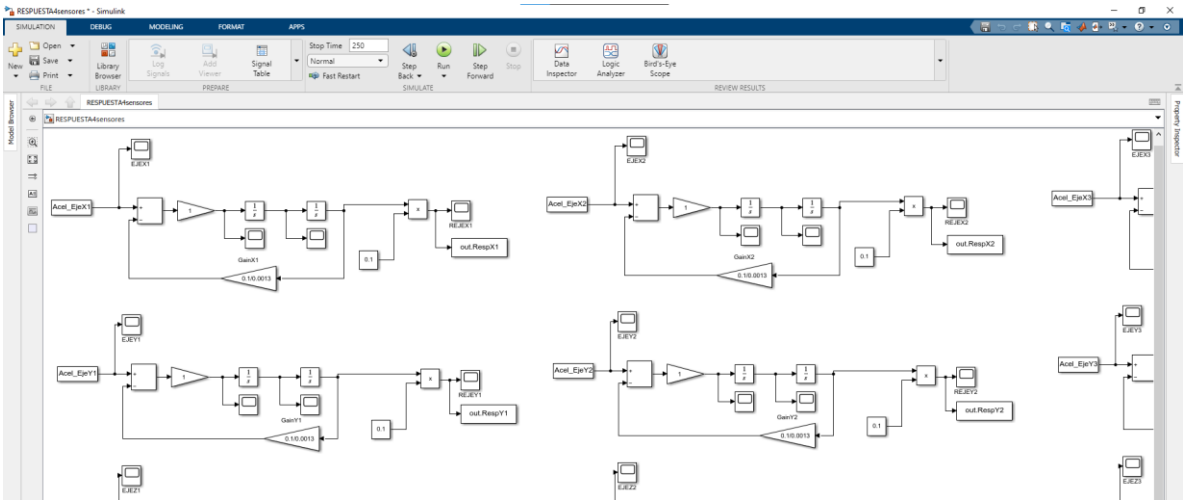

Figura 26. Modelo de obtención de respuesta en Simulink para los 4 sensores.

10. Se configuran los parámetros necesarios tomando en cuenta los resultados de las gráficas de FFT, y se da clic en el botón “calcular” del panel de control de la simulación del modelo de respuesta. Una vez mostrados los resultados del cálculo, se da clic en “Simular respuesta”.

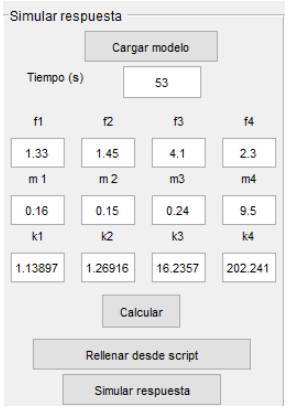

Figura 27. Panel de control de obtención de respuesta de vibraciones en IU.

11. Una vez finalizada la simulación, ya se puede dar clic en los botones referentes a los ejes de los acelerómetros, Los resultados se mostrarán en las gráficas ubicadas en la parte derecha de la interfaz:

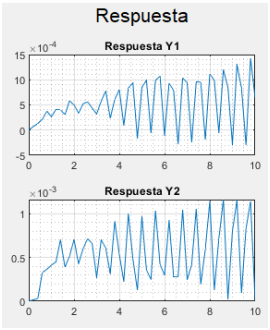

Figura 28. Gráficas de respuesta de muestras de aceleración.

Procesamiento de datos con simulación nueva.

El tratamiento de los datos se realiza de la misma forma que al trabajar con datos previos, se debe seguir las instrucciones a partir de la número 3 del apartado anterior.

1. Para recabar nuevos datos, se da clic en el botón “Muestreo”, que desplegará una ventana con la adquisición visible de los datos y para finalizar la recoleccion de señales se dará clic en el botón “Parar”.
   Debe esperar a que la simulación termine, o en su defecto, finalizarla con el botón “Terminar”. Los resultados ya estarán listos para procesarse con la GUI cuando termine la simulación.

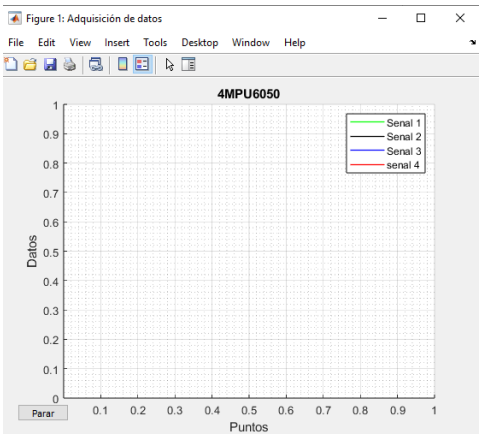

Figura 29. Panel de control de simulación de adquisición de datos de aceleración.

Otras opciones

1. El botón “Limpiar GUI” elimina los gráficos generados de la interfaz, pero los datos seguirán guardados en Segundo plano por si se desea graficar nuevamente los resultados.

2. El botón “Cerrar gráficas” cierra las gráficas generadas en ventanas independientes, también se mostrará una ventana emergente preguntando si desea salir de la interfaz al presionar este botón.

3. En la parte superior izquierda de la interfaz se encontrará un menú de ayuda con las siguientes opciones:

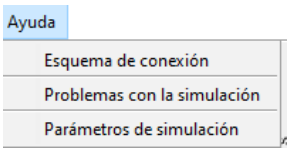

Figura 30. Panel de ayuda de interfaz de usuario.

Cada opción desplegará ayuda relacionada con la interfaz y su uso, así como problemas comunes que podrían presentarse durante su uso y las posibles causas de las fallas en la ejecución de la GUI.

4. Para guardar los resultados, se presiona el botón “Guardar” ubicado debajo del panel de control del modelo de respuesta.

Discusión

Se intentó utilizar el protocolo I2C para obtener los datos de los acelerómetros, debido a que hay más documentación sobre este método de adquisición con los sensores MPU6050, sin embargo, este se ve limitado a obtener únicamente datos confiables al conectar únicamente 2 sensores. Al intentar la adquisición de 4 sensores a la vez mediante este método, las simulaciones no se ejecutan, o muestran lecturas nulas. Por lo que se decidió continuar con la adquisición de los datos mediante el puerto serial, aunque utiliza más recursos del equipo, la comunicación entre los 4 sensores, la placa Arduino Uno y el software, se realiza de manera adecuada. Hace falta documentación sobre la adquisición de los datos de acelerómetros MPU6050 con conexiones de este tipo, la mayoría de las referencias en internet utilizan únicamente 1 sensor, lo cual no es suficiente como para recolectar valores que sean significativos a la hora de realizar las simulaciones que se hicieron con el modelo propuesto.

Aunque existen multiplexores para utilizar el protocolo I2C en la comunicación entre Arduino y los sensores mpu6050 para utilizar más de 2 de ellos, la documentación es muy escasa con respecto a este tipo de dispositivos, y aunque en teoría deberían funcionar normalmente, las lecturas que se toman con sensores mpu6050 conectados con multiplexores I2C suelen tener mucho ruido y los resultados se vuelven poco relevantes. Quizá en investigaciones posteriores la adquisición de señales mediante protocolo I2C con acelerómetros mejore, sin embargo, por el momento el protocolo serial es la mejor opción ante la inconsistencia en los resultados que puede otorgar utilizar multiplexores I2C con Arduino.

El monitoreo de salud estructural aún está en vías de desarrollo a comparación del monitoreo del estado de máquinas rotativas por medio de análisis de vibraciones, debido a que se había abandonado por mucho tiempo el estudio de esta área, sin embargo, los avances en la fabricación de hardware y equipos que pueden utilizarse para esta tarea han despertado el interés de más investigadores en esta área y recientemente se han realizado avances significativos. La tendencia actual apunta a que combinar el SHM con software que permita validar los resultados u optimizar diseños (software CAE, CAM y CAD) será cada vez más común en un futuro, no obstante, el equipo existente en el mercado para el monitoreo de salud estructural (en el caso de acelerómetros) no es asequible para todos y aún existen pocas empresas dedicadas a este segmento; entonces es necesario que empiece a contemplarse dentro de la formación de los ingenieros mecánicos y civiles una revisión más profunda del monitoreo de salud estructural, a falta de personal capacitado en esta área, conocer sus aplicaciones pondría en ventaja a los egresados de dichas licenciaturas.

Conclusiones

Como conclusiones, la implementación de sistemas de monitoreo de salud estructural en instituciones de educación superior es viable, debido al rápido crecimiento demográfico que se experimenta en la actualidad, y en consecuencia, el proceso de urbanización se ha acelerado exponencialmente, incrementando el requerimiento de personal capacitado que pueda asegurar la integridad de las edificaciones cada vez de mayor altura que requieren de inspecciones y mantenimiento constante de manera eficiente y segura, debido a que están expuestas constantemente a los daños provocados por eventos de vibración, deslizamientos de tierra, impactos directos, envejecimiento, factores ambientales como la humedad, inundaciones, entre otros. Esto hace que, formar ingenieros que sean conscientes de la existencia de técnicas no destructivas para la evaluación del estado de una estructura sea de vital importancia para salvaguardar las edificaciones y las vidas humanas, al utilizar el SHM como una herramienta más para asegurar la elaboración de proyectos de construcción y diseño de estructuras que sean eficientes.

La finalidad del sistema creado es que los estudiantes de ingeniería mecánica y eléctrica/civil puedan identificar de mejor manera la aplicación de conceptos relacionados con los parámetros involucrados en la integridad de las estructuras de un grado de libertad (masa y rigidez) y cómo estos influyen en el comportamiento de las señales captadas por los acelerómetros, así como estos pueden ayudar en el análisis del estado en que se encuentran las estructuras mencionadas, respaldadas por el tratamiento digital de señales y de las normas establecidas, enriqueciendo el perfil de los egresados de estas áreas en las instituciones de educación superior.

Si bien el sistema presentado es a nivel didáctico, el saber interpretar correctamente los resultados que se muestran en la interfaz, así como la elaboración de diagnósticos basados en los espectros de vibración obtenidos y la comunicación adecuada de los resultados, fomentará en los alumnos la búsqueda e interés por la investigación en pro del desarrollo de dicha tecnología como un complemento que podría marcar una diferencia significativa en el entorno al que se enfrentarán en un futuro donde la urbanización cambiará de forma drástica los estándares de seguridad y mantenimiento requeridos para la ejecución de proyectos de construcción.

Con el presente trabajo, se busca ayudar de alguna manera al progreso en la investigación del potencial del SHM, poniéndolo al alcance de los estudiantes y docentes que pueden replicar fácilmente el sistema y hacer uso de la interfaz para comenzar a comprender el comportamiento de las estructuras, predecir y elaborar diagnósticos con los resultados, incentivando en los alumnos el interés en el monitoreo y ayudándolos a comprender mejor cómo la variación en las frecuencias de sistemas de 1GDL pueden mostrar si hay condiciones de operación inseguras, si algún diseño requiere de una revisión más profunda, así como de

proporcionar una herramienta más a las ya existentes para la evaluación y mantenimiento de sistemas similares.

Como trabajos de investigación futuros relacionados con el desarrollo de la interfaz creada, se buscará el filtrado optimizado de los datos obtenidos y la mejora en lo que a comunicación serial entre dispositivos se refiere, así como actualizaciones posteriores para mejorar el rendimiento del programa que, aunque es funcional, se sigue investigando para alcanzar en algún momento un rendimiento óptimo, así como una depuración exhaustiva en la programación de la interfaz, ya que debido a las funciones que se le fueron integrando, el código del programa resultó muy extenso.

Bibliografía

Arduino. (2022). *Arduino*. Obtenido de https://www.arduino.cc/en/Guide/Introduction

Autodesk Inc. (2022). *3d-printing*. Obtenido de https://latinoamerica.autodesk.com/solutions/3d-printing

Beer, F. P., & Jhonston, E. R. (2009). *Mecánica de Materiales* (Quinta Edición ed.). México DF: Mc Graw Hill.

Cawley, P. (2018). Structural health monitoring: Closign the gap between research and industrial deployment. *Structural Health Monitoring, 17*(5), 1225-1244. doi:https://doi.org/10.1177/1475921717750047

Chan, T., Li, J., & Caprani, C. (2018). Structural Identification and evaluation for SHM applications. *Journal of Civil Structural Health Monitoring*, 719-720. doi:https://doi.org/10.1007/s13349-018-0308-1

Farrar, C. R., & Worden, K. (2013). Appendix A: Signal Processing For SHM. En C. Farrar, & K. Worden, *Structural Health monitoring: A Machine Learning Perspective* (págs. 479-592). Chichester, United Kingdom: John Wiley & Sons. doi:doi:10.1002/9781118443118.app1

Farrar, C. R., & Worden, K. (2013). Appendix B: Essential Linear Structural Dynamics. En C. Farrar, & K. Worden, *Structural Health Monitoring: A machine Learning Perspective* (págs. 593-621). Chichester, United Kingdom: Wiley & Sons, Ltd. doi:doi:10.1002/9781118443118.app2

Farrar, C., & Worden, K. (2007). An introduction to structural health monitoring. *An introduction to structural health monitoring*, 303-315. doi:DOI: 10.1098 / rsta.2006.1928

Gere, J. M., & Goodno, B. J. (2009). *Mecánica de Materiales* (Séptimo ed.). México D.F.: CENGAGE.

Lorenzo, E., Manzato, S., & Peeters, B. (2017). Structural Health Monitoring strategies based on the estimation of modal parameters. *Procedia Engineering, 199*(1), 3182-3187. doi:https://doi.org/10.1016/j.proeng.2017.09.521.

Loutas, T., & Panopoulou, A. (2012). Intelligent health monitoring of aerospace composite structures based on dynamic strain measurements. *Expert Systems with Applications, 39*(9), 8412-8422. doi:https://doi.org/10.1016/j.eswa.2012.01.179.

MathWorks. (2022). *MathWorks Help Center*. Obtenido de https://la.mathworks.com/help/matlab/ref/guide.html?searchHighlight=guide&s_tid=srchtitle_guide_1

MathWorks. (marzo de 2022). *MATLAB*. Obtenido de https://la.mathworks.com/products/matlab.html

MathWorks. (2022). *Simulink*. Obtenido de https://la.mathworks.com/products/simulink.html

MCI Electronics. (1 de Febrero de 2022). *Arduino UNO*. Obtenido de https://arduino.cl/arduino-uno/

Mejía, Á., & Salazar, V. (2019). *Diseño e implementación de un prototipo para el análisis de salud estructural de un puente a través de una red de sensores inalámbricos utilizando el software MATLAB.* Universidad Politécnica de Salesiana, Ecuador.

Mitchell, J. (2007). From vibration measurements to condition based maintenance - Seventy years of continuous progress. *Sound & Vibration, 41*(1), 62-78. Obtenido de https://www.researchgate.net/publication/294821369_From_vibration_measurements_to_condition_based_maintenance_-_Seventy_years_of_continuous_progress

MPU 6050 Datasheet. (s.f.). Obtenido de https://pdf1.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html

Naylamp Mechatronics. (2021). *Naylamp Mechatronics*. Obtenido de https://naylampmechatronics.com/sensores-posicion-inerciales-gps/33-modulo-mpu6050-acelerometro-giroscopio-i2c.html

Olivares, D. (24 de Junio de 2018). Estudio e implementación de algoritmos para la estimación de la posición mediante sistemas inerciales con arduino. Valencia, España. Obtenido de https://riunet.upv.es/bitstream/handle/10251/109790/Olivares%20-%20Estudio%20e%20implementaci%C3%B3n%20de%20algoritmos%20para%20la%20estimaci%C3%B3n%20de%20la%20posici%C3%B3n%20mediante%20sist....pdf?sequence=1&isAllowed=y

Rivero, E. (2018). Communications of embedded systems using the IEEE 802.15.4. La Laguna, Tenerife, España.

Sabato, A., & Niezrecki, C. (2017). Wireless MEMS-Based Accelerometer Sensor Boards for Structural Vibration Monitoring: A Review. *IEEE Sensors Journal, 17*(2), 226-235. doi:https://doi.org/10.1109/JSEN.2016.2630008

Salazar, D., & Serrano, K. (2018). *Implementación de un prototipo para el monitoreo sísmico utilizando acelerómetros de bajo costo.* Escuela Politécnica nacional, Ecuador.

Schommer, S., & Nguyen, V. (2017). Model updating for structural health monitoring using static and dynamic measurements. *Procedia Engineering, 199*(1), 2146-2153. doi:https://doi.org/10.1016/j.proeng.2017.09.156.

Tecnológico Nacional de México. (24 de mayo de 2022). *Apuntes de materias.* Obtenido de Microcontroladores: http://www.itq.edu.mx/carreras/IngElectronica/archivos_contenido/Apuntes%20de%20materias/CDC1203_DSP/1_Senales_&_Sistemas.pdf

Vanciu Rau , A., Mihai, D., & Tataru, D. (2021). INTEGRATING ENGINEERING AND TECHNOLOGY IN EARTH SCIENCE EDUCATION PRACTICES. Romanian Academy Publishing House.

Vanciu-Rău, A., Ţigănescu, A., & Dragoş , T. (2020). The Seismic Simulator as a Didactic Tool for Seismic Education. *AIP Conference Proceedings 2218*(2218), 060004. doi:https://doi.org/10.1063/5.0001037

Vara, D. (Julio de 2014). Sistemas para determinar la posición y orientación de herramientas quirúrgicas en operaciones de cirugía laparoscópica. Valladolid, España. Obtenido de https://uvadoc.uva.es/bitstream/handle/10324/12884/TFG-P-161.pdf;jsessionid=E28E85C10833F4BFC1EDA99005EB965E?sequence=1

Woodman, O. J. (2007). An introduction to inertial navigation. Cambridge. Obtenido de https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

Yuequan, B., & Zhicheng, C. (2019). The State Of the Art Data Science and Engineering in Structrual Health Monitoring. *Engineering, 5*(2), 234-242. doi:https://doi.org/10.1016/j.eng.2018.11.027

Anexos

Código de interfaz gráfica de usuario

```
function varargout = GUIDE2(varargin)
% GUIDE2 MATLAB code for GUIDE2.fig
%      GUIDE2, by itself, creates a new GUIDE2
or raises the existing
%      singleton*.
%
%      H = GUIDE2 returns the handle to a new
GUIDE2 or the handle to
%      the existing singleton*.
%
%
GUIDE2('CALLBACK',hObject,eventData,handles,..
.) calls the local
%      function named CALLBACK in GUIDE2.M
with the given input arguments.
%
%      GUIDE2('Property','Value',...) creates
a new GUIDE2 or raises the
%      existing singleton*.  Starting from the
left, property value pairs are
%      applied to the GUI before
GUIDE2_OpeningFcn gets called.  An
%      unrecognized property name or invalid
value makes property application
%      stop.  All inputs are passed to
GUIDE2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE2's Tools
menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE2, GUIDATA, GUIHANDLES

% Edit the above text to modify the response
to help GUIDE2

% Last Modified by GUIDE2 v2.5 05-Mar-2022
00:10:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
mfilename, ...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@GUIDE2_OpeningFcn, ...
                   'gui_OutputFcn',
@GUIDE2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
function GUIDE2_OpeningFcn(hObject, ~,
handles, varargin)

load_system('SERIAL_TIME_BIEN');
find_system;
set_param('SERIAL_TIME_BIEN','StopTime','5');

%obtener posición de axes para reescalar
handles.sfig=get(hObject,'Position');
handles.s1 =
get(handles.plotacel1,'Position');
handles.s2 =
get(handles.plotacel2,'Position');
handles.s3 =
get(handles.plotacel3,'Position');
handles.s4 =
get(handles.plotacel4,'Position');
handles.s5 = get(handles.fftaxes1,'Position');
handles.s6 = get(handles.fftaxes2,'Position');
handles.s7 = get(handles.fftaxes3,'Position');
handles.s8 = get(handles.fftaxes4,'Position');
handles.s9 = get(handles.fftaxm1,'Position');
handles.s10 = get(handles.fftaxm2,'Position');
handles.s11 = get(handles.fftaxm3,'Position');
handles.s12 = get(handles.fftaxm4,'Position');
handles.s13 = get(handles.resp1,'Position');
handles.s14 = get(handles.resp2,'Position');
handles.s15 = get(handles.resp3,'Position');
handles.s16 = get(handles.resp4,'Position');
handles.s17 = get(handles.text9,'Position');
handles.s18 = get(handles.text11,'Position');
handles.s19 = get(handles.text10,'Position');
handles.s20 = get(handles.text12,'Position');
handles.s21 =
get(handles.RENOMBRAYGRAF,'Position');
handles.s22 = get(handles.grt,'Position');
handles.s23 = get(handles.cargar2,'Position');
handles.s24 =
get(handles.guardar2,'Position');
handles.s25 = get(handles.cerrarg,'Position');
handles.s26 = get(handles.limpiar,'Position');
%panel de parámetros de simulación
handles.spanel1=get(handles.uipanel1,'Position
');
handles.p1=get(handles.simt,'Position');
handles.p2=get(handles.tiemposim,'Position');
handles.p3=get(handles.text8,'Position');
handles.p4=get(handles.Run,'Position');
handles.p5=get(handles.Terminar,'Position');
%panel de graficar en gui
handles.spanel2=get(handles.uibuttongroup5,'Po
sition');
handles.p6=get(handles.uibuttongroup8,'Positio
n');
handles.p7=get(handles.uibuttongroup7,'Positio
n');
handles.p8=get(handles.uibuttongroup6,'Positio
n');
%botones panel acel y fft gui
handles.p9=get(handles.gfftacelx,'Position');
handles.p10=get(handles.gfftacely,'Position');
```

```
handles.p11=get(handles.gfftacelz,'Position');
%botones panel acel gui
handles.p12=get(handles.graficarx,'Position');
handles.p13=get(handles.graficarY,'Position');
handles.p14=get(handles.graficarZ,'Position');
%botones panel fft gui
handles.p15=get(handles.grafffftx,'Position');
handles.p16=get(handles.grafffty,'Position');
handles.p17=get(handles.grafffftz,'Position');
%panel gráficas detalladas
handles.spanel3=get(handles.uipanel2,'Position
');
handles.p18=get(handles.uibuttongroup1,'Positi
on');
handles.p19=get(handles.uibuttongroup2,'Positi
on');
handles.p20=get(handles.uibuttongroup3,'Positi
on');
%botones acel y fft graf detalladas
handles.p21=get(handles.acfftx,'Position');
handles.p22=get(handles.acffty,'Position');
handles.p23=get(handles.acfftz,'Position');
%botones acel graf detalladas
handles.p24=get(handles.acx,'Position');
handles.p25=get(handles.acy,'Position');
handles.p26=get(handles.acz,'Position');
%botones fft graf detalladas
handles.p27=get(handles.gffx,'Position');
handles.p28=get(handles.gffy,'Position');
handles.p29=get(handles.gffz,'Position');
%panel respuesta
handles.spanel4=get(handles.respuesta,'Positio
n');
handles.p30=get(handles.respload,'Position');
handles.p31=get(handles.trxt3,'Position');
handles.p32=get(handles.RIG,'Position');
handles.p33=get(handles.text13,'Position');
handles.p34=get(handles.TiempoResp,'Position')
;
handles.p35=get(handles.RunResp,'Position');
handles.p36=get(handles.text14,'Position');
handles.p37=get(handles.respX,'Position');
handles.p38=get(handles.respY,'Position');
handles.p39=get(handles.respZ,'Position');
handles.p40=get(handles.respdet,'Position');
handles.output = hObject;
guidata(hObject, handles);

% UIWAIT makes GUIDE2 wait for user response
(see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned
to the command line.
function varargout = GUIDE2_OutputFcn(~, ~,
handles)
% varargout  cell array for returning output
args (see VARARGOUT);
varargout{1} = handles.output;

function Run_Callback(hObject, ~, handles)
stoptime =
str2double(get(handles.tiemposim,'String'));
iniciarsim = questdlg({'¿Iniciar
simulación?','(Cerrar esta ventana','la
iniciará
automáticamente)'},'Simulación','Simular','No'
,'modal');
if strcmp(iniciarsim,'No')
    return
else
    warndlg('Simulación iniciada','Aviso de
simlación')
end
if stoptime ~= Inf
    set(hObject,'Style','pushbutton');

set_param('SERIAL_TIME_BIEN','SimulationComman
d','Start');
        guidata(hObject, handles);
else
    set(hObject,'Style','togglebutton');
    guidata(hObject, handles);

    v = get(hObject,'Value');

set_param('SERIAL_TIME_BIEN','SimulationComman
d','Start');
    if v == 0
       set(hObject,'String','Reanudar');

set_param('SERIAL_TIME_BIEN','SimulationComman
d','Pause');
    else
       set(hObject,'String','Pausar');

set_param('SERIAL_TIME_BIEN','SimulationComman
d','Continue');
    end
    guidata(hObject, handles);
end
function tiemposim_Callback(hObject, ~, ~)
v = get(hObject,'String');
set_param('SERIAL_TIME_BIEN','StopTime',v)
function tiemposim_CreateFcn(hObject, ~, ~)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Terminar_Callback(~, ~, ~)

set_param(gcs,'SimulationCommand','Stop');
warndlg('Simulación
Terminada','Aviso','modal')

function grt_Callback(~, ~, ~)
evalin('base','grafs4xyz2')

function cerrarg_Callback(~, ~, ~)
close all

function gffx_Callback(~, ~, ~)
evalin('base','FFTX42')
function gffy_Callback(~, ~, ~)
evalin('base','FFTY42')
function gffz_Callback(~, ~, ~)
evalin('base','FFTZ42')
function acx_Callback(~, ~, ~)
evalin('base','ACCELX42')
function acy_Callback(~, ~, ~)
evalin('base','ACCELY42')
function acz_Callback(~, ~, ~)
evalin('base','ACCELZ42')
function acfftz_Callback(~, ~, ~)
evalin('base','grafs4Z2')
function acffty_Callback(~, ~, ~)
evalin('base','grafs4Y2')
```

```
function acfftx_Callback(~, ~, ~)
evalin('base','grafs4X2')
function guardar2_Callback(hObject, ~,
handles)

Selsave = questdlg('¿Guardar
resultados?','Guardar','Guardar','Cancelar','G
uardar');
if strcmp(Selsave,'Cancelar')
  return
else

uisave

end
guidata(hObject, handles);
function cargar2_Callback(hObject, ~, handles)
evalin('base','cargar2')
guidata(hObject, handles);
function listbox1_Callback(hObject, ~,
handles)
lista = cellstr(get(hObject,'String'));
seleccion = lista{get(hObject,'Value')};
archivo = [handles.direccion, seleccion];
set(handles.activex1, 'URL', archivo);
function listbox1_CreateFcn(hObject, ~, ~)

if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function cargar_Callback(hObject, ~, handles)

[file, path] = uigetfile({'*.mat', 'Datos de
simulación'});
handles.direccion = path;
handles.archivo = file;
set(handles.listbox1, 'String', file);
guidata(hObject, handles);
function figure1_CloseRequestFcn(hObject, ~,
~)
Seleccion = questdlg('¿Cerrar la
interfaz?','Saliendo...','Salir','No','No');
if strcmp(Seleccion,'No')
    return
else
    clc; delete(hObject);
end
function ayuda_Callback(~, ~, ~)
function conex_Callback(~, ~, ~)
winopen('diagrama4mpu.png')
function prob_Callback(~, ~, ~)
winopen('problemas.txt')

function guardasimres_Callback(~, ~, ~)
uisave
Selsave = questdlg('¿Guardar
resultados?','Guardar','Guardar','Cancelar','G
uardar');
if strcmp(Selsave,'Cancelar')
  return
else
     evalin('base','grsimr')
     warndlg('Resultados guardados','Guardado
con éxito','modal')
end

function configsimhp_Callback(~, ~, ~)
helpdlg('En el cuadro de texto editable, solo
puede ingresar valores mayores a "0". Por
default, el tiempo de simulación queda
establecido en 5 segundos, si desea realizar
un monitoreo en tiempo real por más tiempo, se
escribe "inf" (sin comillas") en el cuadro de
texto.Los botones "Iniciar" y "Terminar" están
configurados para que controlar la simulación,
puede pausar o reanudarla si lo requiere.
Estas últimas opciones solo estarán
disponibles mientras se esté ejecutando la
simulación, y aparecerán en el mismo botón que
inicia la ejecución del modelo.','Configurar
simulación')

function graficarx_Callback(~, ~, handles)
vacx1= evalin('base','asimresX1');
 evalin('base','asimresX1')
    vatx1= evalin('base','asimrest');
    valcx1=squeeze(vacx1);
    axes(handles.plotacel1);
    plot(vatx1,valcx1);
    title('Acel X1 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
          grid minor

    vacx2= evalin('base','asimresX2');
    vatx2= evalin('base','asimrest');
    valcx2=squeeze(vacx2);
    axes(handles.plotacel2);
    plot(vatx2,valcx2);
    title('Acel X2 vs tiempo')
    xlabel('Tiempo (s)')
    ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacx3= evalin('base','asimresX3');
    vatx3= evalin('base','asimrest');
    valcx3=squeeze(vacx3);
    axes(handles.plotacel3);
    plot(vatx3,valcx3);
        title('Acel X3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacx4= evalin('base','asimresX4');
    vatx4= evalin('base','asimrest');
    valcx4=squeeze(vacx4);
    axes(handles.plotacel4);
    plot(vatx4,valcx4);
    title('Acel X4 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

function graficarY_Callback(~, ~, handles)
vacy1= evalin('base','asimresY1');
    vaty1= evalin('base','asimrest');
    valcy1=squeeze(vacy1);
    axes(handles.plotacel1);
    plot(vaty1,valcy1);
    title('Acel Y1 vs tiempo')
        xlabel('Tiempo (s)')
```

```
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacy2= evalin('base','asimresY2');
    vaty2= evalin('base','asimrest');
    valcy2=squeeze(vacy2);
    axes(handles.plotacel2);
    plot(vaty2,valcy2);
    title('Acel Y2 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacy3= evalin('base','asimresY3');
    vaty3= evalin('base','asimrest');
    valcy3=squeeze(vacy3);
    axes(handles.plotacel3);
    plot(vaty3,valcy3);
    title('Acel Y3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacy4= evalin('base','asimresY4');
    vaty4= evalin('base','asimrest');
    valcy4=squeeze(vacy4);
    axes(handles.plotacel4);
    plot(vaty4,valcy4);
    title('Acel Y4 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
             grid minor

function graficarZ_Callback(~, ~, handles)
vacz1= evalin('base','asimresZ1');
    vatz1= evalin('base','asimrest');
    valcz1=squeeze(vacz1);
    axes(handles.plotacel1);
    plot(vatz1,valcz1);
    title('Acel Z1 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    vacz2= evalin('base','asimresZ2');
    vatz2= evalin('base','asimrest');
    valcz2=squeeze(vacz2);
    axes(handles.plotacel2);
    plot(vatz2,valcz2);
    title('Acel Z2 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    vacz3= evalin('base','asimresZ3');
    vatz3= evalin('base','asimrest');
    valcz3=squeeze(vacz3);
    axes(handles.plotacel3);
    plot(vatz3,valcz3);
    title('Acel Z3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    vacz4= evalin('base','asimresZ4');
    vatz4= evalin('base','asimrest');
    valcz4=squeeze(vacz4);
    axes(handles.plotacel4);
    plot(vatz4,valcz4);
    title('Acel Z4 vs tiempo')
xlabel('Tiempo (s)')
ylabel('Aceleración  m/s^2')
grid on
    grid minor

function grafffftx_Callback(~, ~, handles)
%FFTXGUIDE
 frqx1= evalin('base','freqx1');
    fmsx1= evalin('base','xdft1');
    axes(handles.fftaxes1);
    plot(frqx1,fmsx1);
    title('FFT X1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqx2= evalin('base','freqx2');
    fmsx2= evalin('base','xdft2');
    axes(handles.fftaxes2);
    plot(frqx2,fmsx2);
    title('FFT X2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                 grid minor

 frqx3= evalin('base','freqx3');
    fmsx3= evalin('base','xdft3');
    axes(handles.fftaxes3);
    plot(frqx3,fmsx3);
    title('FFT X3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqx4= evalin('base','freqx4');
    fmsx4= evalin('base','xdft4');
    axes(handles.fftaxes4);
    plot(frqx4,fmsx4);
    title('FFT X4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    axes(handles.fftaxm1);
    plot(frqx1,10*log10(fmsx1));
    title('FFT X1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
            grid on
                grid minor

    axes(handles.fftaxm2);
    plot(frqx2,10*log10(fmsx2));
    title('FFT X2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
```

```
                grid minor

    axes(handles.fftaxm3);
    plot(frqx3,10*log10(fmsx3));
    title('FFT X3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

    axes(handles.fftaxm4);
    plot(frqx4,10*log10(fmsx4));
    title('FFT X4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor
function grafffty_Callback(~, ~, handles)
%FFTYGUIDE
frqy1= evalin('base','freqy1');
    fmsy1= evalin('base','ydft1');
    axes(handles.fftaxes1);
    plot(frqy1,fmsy1);
    title('FFT Y1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
            grid minor

 frqy2= evalin('base','freqy2');
    fmsy2= evalin('base','ydft2');
    axes(handles.fftaxes2);
    plot(frqy2,fmsy2);
    title('FFT Y2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqy3= evalin('base','freqy3');
    fmsy3= evalin('base','ydft3');
    axes(handles.fftaxes3);
    plot(frqy3,fmsy3);
    title('FFT Y3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqy4= evalin('base','freqy4');
    fmsy4= evalin('base','ydft4');
    axes(handles.fftaxes4);
    plot(frqy4,fmsy4);
    title('FFT Y4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

    axes(handles.fftaxm1);
    plot(frqy1,10*log10(fmsy1));
    title('FFT Y1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
            grid on
                grid minor

    axes(handles.fftaxm2);
    plot(frqy2,10*log10(fmsy2));
    title('FFT Y2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
                grid minor

    axes(handles.fftaxm3);
    plot(frqy3,10*log10(fmsy3));
    title('FFT Y3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

    axes(handles.fftaxm4);
    plot(frqy4,10*log10(fmsy4));
    title('FFT Y4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor
function grafffftz_Callback(~, ~, handles)
%FFTZGUIDE
frqz1= evalin('base','freqz1');
    fmsz1= evalin('base','zdft1');
    axes(handles.fftaxes1);
    plot(frqz1,fmsz1);
    title('FFT Z1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqz2= evalin('base','freqz2');
    fmsz2= evalin('base','zdft2');
    axes(handles.fftaxes2);
    plot(frqz2,fmsz2);
    title('FFT Z2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor


 frqz3= evalin('base','freqz3');
    fmsz3= evalin('base','zdft3');
    axes(handles.fftaxes3);
    plot(frqz3,fmsz3);
    title('FFT Z3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqz4= evalin('base','freqz4');
    fmsz4= evalin('base','zdft4');
    axes(handles.fftaxes4);
    plot(frqz4,fmsz4);
    title('FFT Z4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    axes(handles.fftaxm1);
    plot(frqz1,10*log10(fmsz1));
    title('FFT Z1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
```

```
            grid on
                grid minor

    axes(handles.fftaxm2);
    plot(frqz2,10*log10(fmsz2));
    title('FFT Z2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
                grid minor

    axes(handles.fftaxm3);
    plot(frqz3,10*log10(fmsz3));
    title('FFT Z3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

    axes(handles.fftaxm4);
    plot(frqz4,10*log10(fmsz4));
    title('FFT Z4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

function gfftacelx_Callback(~, ~, handles)
 vacx1= evalin('base','asimresX1');
    vatx1= evalin('base','asimrest');
    valcx1=squeeze(vacx1);
    axes(handles.plotacel1);
    plot(vatx1,valcx1);
    title('Acel X1 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

vacx2= evalin('base','asimresX2');
    vatx2= evalin('base','asimrest');
    valcx2=squeeze(vacx2);
    axes(handles.plotacel2);
    plot(vatx2,valcx2);
    title('Acel X2 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

  vacx3= evalin('base','asimresX3');
    vatx3= evalin('base','asimrest');
    valcx3=squeeze(vacx3);
    axes(handles.plotacel3);
    plot(vatx3,valcx3);
    title('Acel X3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    vacx4= evalin('base','asimresX4');
    vatx4= evalin('base','asimrest');
    valcx4=squeeze(vacx4);
    axes(handles.plotacel4);
    plot(vatx4,valcx4);
    title('Acel X4 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor
%%
frqx1= evalin('base','freqx1');
    fmsx1= evalin('base','xdft1');
    axes(handles.fftaxes1);
    plot(frqx1,fmsx1);
    title('FFT X1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

 frqx2= evalin('base','freqx2');
    fmsx2= evalin('base','xdft2');
    axes(handles.fftaxes2);
    plot(frqx2,fmsx2);
    title('FFT X2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqx3= evalin('base','freqx3');
    fmsx3= evalin('base','xdft3');
    axes(handles.fftaxes3);
    plot(frqx3,fmsx3);
    title('FFT X3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
        grid on
            grid minor

 frqx4= evalin('base','freqx4');
    fmsx4= evalin('base','xdft4');
    axes(handles.fftaxes4);
    plot(frqx4,fmsx4);
    title('FFT X4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
            grid minor

    axes(handles.fftaxm1);
    plot(frqx1,10*log10(fmsx1));
    title('FFT X1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
            grid on
            grid minor

    axes(handles.fftaxm2);
    plot(frqx2,10*log10(fmsx2));
    title('FFT X2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
                grid minor

    axes(handles.fftaxm3);
    plot(frqx3,10*log10(fmsx3));
    title('FFT X3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

    axes(handles.fftaxm4);
    plot(frqx4,10*log10(fmsx4));
```

```
    title('FFT X4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor
function gfftacely_Callback(~, ~, handles)
vacy1= evalin('base','asimresY1');
    vaty1= evalin('base','asimrest');
    valcy1=squeeze(vacy1);
    axes(handles.plotacel1);
    plot(vaty1,valcy1);
    title('Acel Y1 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

vacy2= evalin('base','asimresY2');
    vaty2= evalin('base','asimrest');
    valcy2=squeeze(vacy2);
    axes(handles.plotacel2);
    plot(vaty2,valcy2);
    title('Acel Y2 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor


  vacy3= evalin('base','asimresY3');
    vaty3= evalin('base','asimrest');
    valcy3=squeeze(vacy3);
    axes(handles.plotacel3);
    plot(vaty3,valcy3);
    title('Acel Y3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    vacy4= evalin('base','asimresY4');
    vaty4= evalin('base','asimrest');
    valcy4=squeeze(vacy4);
    axes(handles.plotacel4);
    plot(vaty4,valcy4);
    title('Acel Y4 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor
frqy1= evalin('base','freqy1');
    fmsy1= evalin('base','ydft1');
    axes(handles.fftaxes1);
    plot(frqy1,fmsy1);
    title('FFT Y1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqy2= evalin('base','freqy2');
    fmsy2= evalin('base','ydft2');
    axes(handles.fftaxes2);
    plot(frqy2,fmsy2);
    title('FFT Y2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqy3= evalin('base','freqy3');
    fmsy3= evalin('base','ydft3');
    axes(handles.fftaxes3);
    plot(frqy3,fmsy3);
    title('FFT Y3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqy4= evalin('base','freqy4');
    fmsy4= evalin('base','ydft4');
    axes(handles.fftaxes4);
    plot(frqy4,fmsy4);
    title('FFT Y4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    axes(handles.fftaxm1);
    plot(frqy1,10*log10(fmsy1));
    title('FFT Y1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
            grid on
                grid minor

    axes(handles.fftaxm2);
    plot(frqy2,10*log10(fmsy2));
    title('FFT Y2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
               grid minor

    axes(handles.fftaxm3);
    plot(frqy3,10*log10(fmsy3));
    title('FFT Y3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
              grid minor

    axes(handles.fftaxm4);
    plot(frqy4,10*log10(fmsy4));
    title('FFT Y4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor
function gfftacelz_Callback(~, ~, handles)
vacz1= evalin('base','asimresZ1');
    vatz1= evalin('base','asimrest');
    valcz1=squeeze(vacz1);
     axes(handles.plotacel1);
    plot(vatz1,valcz1);
    title('Acel Z1 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

vacz2= evalin('base','asimresZ2');
    vatz2= evalin('base','asimrest');
    valcz2=squeeze(vacz2);
    axes(handles.plotacel2);
    plot(vatz2,valcz2);
```

```
    title('Acel Z2 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
              grid minor

  vacz3= evalin('base','asimresZ3');
    vatz3= evalin('base','asimrest');
    valcz3=squeeze(vacz3);
    axes(handles.plotacel3);
    plot(vatz3,valcz3);
    title('Acel Z3 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
              grid minor

    vacz4= evalin('base','asimresZ4');
    vatz4= evalin('base','asimrest');
    valcz4=squeeze(vacz4);
    axes(handles.plotacel4);
    plot(vatz4,valcz4);
    title('Acel Z4 vs tiempo')
        xlabel('Tiempo (s)')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor
%%
frqz1= evalin('base','freqz1');
    fmsz1= evalin('base','zdft1');
    axes(handles.fftaxes1);
    plot(frqz1,fmsz1);
    title('FFT Z1 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqz2= evalin('base','freqz2');
    fmsz2= evalin('base','zdft2');
    axes(handles.fftaxes2);
    plot(frqz2,fmsz2);
    title('FFT Z2 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqz3= evalin('base','freqz3');
    fmsz3= evalin('base','zdft3');
    axes(handles.fftaxes3);
    plot(frqz3,fmsz3);
    title('FFT Z3 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

 frqz4= evalin('base','freqz4');
    fmsz4= evalin('base','zdft4');
    axes(handles.fftaxes4);
    plot(frqz4,fmsz4);
    title('FFT Z4 Hz vs m/s^2')
        xlabel('Hz')
        ylabel('Aceleración  m/s^2')
            grid on
                grid minor

    axes(handles.fftaxm1);
    plot(frqz1,10*log10(fmsz1));
    title('FFT Z1 Hz vs dB')
        xlabel('Hz')
        ylabel('dB')
            grid on
                grid minor

    axes(handles.fftaxm2);
    plot(frqz2,10*log10(fmsz2));
    title('FFT Z2 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración dB')
            grid on
               grid minor

    axes(handles.fftaxm3);
    plot(frqz3,10*log10(fmsz3));
    title('FFT Z3 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

    axes(handles.fftaxm4);
    plot(frqz4,10*log10(fmsz4));
    title('FFT Z4 Hz vs dB')
        xlabel('Hz')
        ylabel('Aceleración  dB')
            grid on
                grid minor

function respX_Callback(~, ~, handles)
rx1 = evalin('base','out.RespX1');
trx1 = evalin('base','out.tout');
axes(handles.resp1)
plot(trx1,rx1)
title('Respuesta X1')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor

rx2 = evalin('base','out.RespX2');
trx2 = evalin('base','out.tout');
axes(handles.resp2)
plot(trx2,rx2)
title('Respuesta X2')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor

rx3 = evalin('base','out.RespX3');
trx3 = evalin('base','out.tout');
axes(handles.resp3)
plot(trx3,rx3)
title('Respuesta X3')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor

rx4 = evalin('base','out.RespX4');
trx4 = evalin('base','out.tout');
axes(handles.resp4)
plot(trx4,rx4)
title('Respuesta X4')
xlabel('t')
ylabel('Amplitud')
```

```
grid on
    grid minor
function respY_Callback(~, ~, handles)
ry1 = evalin('base','out.RespY1');
try1 = evalin('base','out.tout');
axes(handles.resp1)
plot(try1,ry1)
title('Respuesta Y1')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor

ry2 = evalin('base','out.RespY2');
try2 = evalin('base','out.tout');
axes(handles.resp2)
plot(try2,ry2)
title('Respuesta Y2')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor


ry3 = evalin('base','out.RespY3');
try3 = evalin('base','out.tout');
axes(handles.resp3)
plot(try3,ry3)
title('Respuesta Y3')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor


ry4 = evalin('base','out.RespY4');
try4 = evalin('base','out.tout');
axes(handles.resp4)
plot(try4,ry4)
title('Respuesta Y4')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor
function respZ_Callback(~, ~, handles)
rz1 = evalin('base','out.RespZ1');
trz1 = evalin('base','out.tout');
axes(handles.resp1)
plot(trz1,rz1)
title('Respuesta Z1')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor

rz2 = evalin('base','out.RespZ2');
trz2 = evalin('base','out.tout');
axes(handles.resp2)
plot(trz2,rz2)
title('Respuesta Z2')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor


rz3 = evalin('base','out.RespZ3');
trz3 = evalin('base','out.tout');
axes(handles.resp3)
plot(trz3,rz3)
title('Respuesta Z3')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor


rz4 = evalin('base','out.RespY4');
trz4 = evalin('base','out.tout');
axes(handles.resp4)
plot(trz4,rz4)
title('Respuesta Z4')
xlabel('t')
ylabel('Amplitud')
grid on
    grid minor
function limpiar_Callback(~, ~, handles)
axes(handles.plotacel1);
cla reset
axes(handles.plotacel2);
cla reset
axes(handles.plotacel3);
cla reset
axes(handles.plotacel4);
cla reset
axes(handles.fftaxes1);
cla reset
axes(handles.fftaxes2);
cla reset
axes(handles.fftaxes3);
cla reset
axes(handles.fftaxes4);
cla reset
axes(handles.fftaxm1)
cla reset
axes(handles.fftaxm2);
cla reset
axes(handles.fftaxm3);
cla reset
axes(handles.fftaxm4);
cla reset
axes(handles.resp1)
cla reset
axes(handles.resp2);
cla reset
axes(handles.resp3);
cla reset
axes(handles.resp4);
cla reset

set_param(gcs,'SimulationCommand','Start');


function RunResp_Callback(~, ~, ~)
set_param('RESPUESTA4sensores','SimulationComm
and','Start');
inicioresp = questdlg({'¿Iniciar simulación?
Las variables para graficar aceleraciones y
FFT detalladamente se sobreescribirán con los
datos de la simulación de respuesta, si desea
conservarlos, guarde los resultados antes de
iniciar el modelo de respuesta','(Cerrar esta
ventana','la iniciará
automáticamente)'},'Simulación','Simular','No'
,'modal');
if strcmp(inicioresp,'No')
    return
else
```

```
    warndlg('Simulación iniciada','Aviso de
simlación')
end
function TiempoResp_Callback(hObject, ~, ~)
u=get(hObject,'String');
set_param('Respuesta4sensores','StopTime',u);
function TiempoResp_CreateFcn(hObject, ~, ~)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function respload_Callback(~, ~, ~)
load_system('RESPUESTA4sensores')
find_system('RESPUESTA4sensores')
open_system('RESPUESTA4sensores')
set_param('RESPUESTA4sensores','StopTime','250
')

function RENOMBRAYGRAF_Callback(~, ~, ~)
evalin('base','renombrar')

function RIG_Callback(hObject, ~, ~)
valorrig = get(hObject,'String');
set_param('RESPUESTA4sensores/GainX1','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainX2','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainX3','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainX4','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainY1','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainY2','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainY3','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainY4','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainZ1','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainZ2','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainZ3','Gain',n
um2str(valorrig));
set_param('RESPUESTA4sensores/GainZ4','Gain',n
um2str(valorrig));

function RIG_CreateFcn(hObject, ~, ~)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function figure1_SizeChangedFcn(hObject, ~,
handles)
handles.sfig2 = get(hObject,'Position');
escala=handles.sfig2(3:4)./handles.sfig(3:4);
%cambio de variables para elementos fuera de
paneles
M1=handles.s1;
M2=handles.s2;
M3=handles.s3;
M4=handles.s4;
M5=handles.s5;
M6=handles.s6;
M7=handles.s7;
M8=handles.s8;
M9=handles.s9;
M10=handles.s10;
M11=handles.s11;
M12=handles.s12;
M13=handles.s13;
M14=handles.s14;
M15=handles.s15;
M16=handles.s16;
M17=handles.s17;
M18=handles.s18;
M19=handles.s19;
M20=handles.s20;
M21=handles.s21;
M22=handles.s22;
M23=handles.s23;
M24=handles.s24;
M25=handles.s25;
M26=handles.s26;

%cambio de variables para reescalar panel de
simulación
sp0=handles.spanel1;
sp1=handles.p1;
sp2=handles.p2;
sp3=handles.p3;
sp4=handles.p4;
sp5=handles.p5;

%cambio de variables para reescalar panel de
gráficas en gui
sp00=handles.spanel2;
sp6=handles.p6;
sp7=handles.p7;
sp8=handles.p8;
sp9=handles.p9;
sp10=handles.p10;
sp11=handles.p11;
sp12=handles.p12;
sp13=handles.p13;
sp14=handles.p14;
sp15=handles.p15;
sp16=handles.p16;
sp17=handles.p17;

%cambio de variables para reescalar panel de
gráficas detalladas
sp000=handles.spanel3;
sp18=handles.p18;
sp19=handles.p19;
sp20=handles.p20;
sp21=handles.p21;
sp22=handles.p22;
sp23=handles.p23;
sp24=handles.p24;
sp25=handles.p25;
sp26=handles.p26;
sp27=handles.p27;
sp28=handles.p28;
sp29=handles.p29;

%cambio de variables para panel de simulación
de respuesta
sp0000=handles.spanel4;
sp30=handles.p30;
sp31=handles.p31;
sp32=handles.p32;
sp33=handles.p33;
sp34=handles.p34;
```

```
sp35=handles.p35;
sp36=handles.p36;
sp37=handles.p37;
sp38=handles.p38;
sp39=handles.p39;
sp40=handles.p40;

%Elementos fuera de paneles, axes y etiquetas
set(handles.plotacel1,'Position',[M1(1)*escala
(1),M1(2)*escala(2),M1(3)*escala(1),M1(4)*esca
la(2)]);
set(handles.plotacel2,'Position',[M2(1)*escala
(1),M2(2)*escala(2),M2(3)*escala(1),M2(4)*esca
la(2)]);
set(handles.plotacel3,'Position',[M3(1)*escala
(1),M3(2)*escala(2),M3(3)*escala(1),M3(4)*esca
la(2)]);
set(handles.plotacel4,'Position',[M4(1)*escala
(1),M4(2)*escala(2),M4(3)*escala(1),M4(4)*esca
la(2)]);
set(handles.fftaxes1,'Position',[M5(1)*escala(
1),M5(2)*escala(2),M5(3)*escala(1),M5(4)*escal
a(2)]);
set(handles.fftaxes2,'Position',[M6(1)*escala(
1),M6(2)*escala(2),M6(3)*escala(1),M6(4)*escal
a(2)]);
set(handles.fftaxes3,'Position',[M7(1)*escala(
1),M7(2)*escala(2),M7(3)*escala(1),M7(4)*escal
a(2)]);
set(handles.fftaxes4,'Position',[M8(1)*escala(
1),M8(2)*escala(2),M8(3)*escala(1),M8(4)*escal
a(2)]);
set(handles.fftaxm1,'Position',[M9(1)*escala(1
),M9(2)*escala(2),M9(3)*escala(1),M9(4)*escala
(2)]);
set(handles.fftaxm2,'Position',[M10(1)*escala(
1),M10(2)*escala(2),M10(3)*escala(1),M10(4)*es
cala(2)]);
set(handles.fftaxm3,'Position',[M11(1)*escala(
1),M11(2)*escala(2),M11(3)*escala(1),M11(4)*es
cala(2)]);
set(handles.fftaxm4,'Position',[M12(1)*escala(
1),M12(2)*escala(2),M12(3)*escala(1),M12(4)*es
cala(2)]);
set(handles.resp1,'Position',[M13(1)*escala(1)
,M13(2)*escala(2),M13(3)*escala(1),M13(4)*esca
la(2)]);
set(handles.resp2,'Position',[M14(1)*escala(1)
,M14(2)*escala(2),M14(3)*escala(1),M14(4)*esca
la(2)]);
set(handles.resp3,'Position',[M15(1)*escala(1)
,M15(2)*escala(2),M15(3)*escala(1),M15(4)*esca
la(2)]);
set(handles.resp4,'Position',[M16(1)*escala(1)
,M16(2)*escala(2),M16(3)*escala(1),M16(4)*esca
la(2)]);
set(handles.text9,'Position',[M17(1)*escala(1)
,M17(2)*escala(2),M17(3)*escala(1),M17(4)*esca
la(2)]);
set(handles.text11,'Position',[M18(1)*escala(1
),M18(2)*escala(2),M18(3)*escala(1),M18(4)*esc
ala(2)]);
set(handles.text10,'Position',[M19(1)*escala(1
),M19(2)*escala(2),M19(3)*escala(1),M19(4)*esc
ala(2)]);
set(handles.text12,'Position',[M20(1)*escala(1
),M20(2)*escala(2),M20(3)*escala(1),M20(4)*esc
ala(2)]);
set(handles.RENOMBRAYGRAF,'Position',[M21(1)*e
scala(1),M21(2)*escala(2),M21(3)*escala(1),M21
(4)*escala(2)]);
set(handles.grt,'Position',[M22(1)*escala(1),M
22(2)*escala(2),M22(3)*escala(1),M22(4)*escala
(2)]);
set(handles.cargar2,'Position',[M23(1)*escala(
1),M23(2)*escala(2),M23(3)*escala(1),M23(4)*es
cala(2)]);
set(handles.guardar2,'Position',[M24(1)*escala
(1),M24(2)*escala(2),M24(3)*escala(1),M24(4)*e
scala(2)]);
set(handles.cerrarg,'Position',[M25(1)*escala(
1),M25(2)*escala(2),M25(3)*escala(1),M25(4)*es
cala(2)]);
set(handles.limpiar,'Position',[M26(1)*escala(
1),M26(2)*escala(2),M26(3)*escala(1),M26(4)*es
cala(2)]);
%panel simulación serial
set(handles.uipanel1,'Position',[sp0(1)*escala
(1),sp0(2)*escala(2),sp0(3)*escala(1),sp0(4)*e
scala(2)]);
set(handles.simt,'Position',[sp1(1)*escala(1),
sp1(2)*escala(2),sp1(3)*escala(1),sp1(4)*escal
a(2)]);
set(handles.tiemposim,'Position',[sp2(1)*escal
a(1),sp2(2)*escala(2),sp2(3)*escala(1),sp2(4)*
escala(2)]);
set(handles.text8,'Position',[sp3(1)*escala(1)
,sp3(2)*escala(2),sp3(3)*escala(1),sp3(4)*esca
la(2)]);
set(handles.Run,'Position',[sp4(1)*escala(1),s
p4(2)*escala(2),sp4(3)*escala(1),sp4(4)*escala
(2)]);
set(handles.Terminar,'Position',[sp5(1)*escala
(1),sp5(2)*escala(2),sp5(3)*escala(1),sp5(4)*e
scala(2)]);
%panel graficar en gui
set(handles.uibuttongroup5,'Position',[sp00(1)
*escala(1),sp00(2)*escala(2),sp00(3)*escala(1)
,sp00(4)*escala(2)]);
set(handles.uibuttongroup8,'Position',[sp6(1)*
escala(1),sp6(2)*escala(2),sp6(3)*escala(1),sp
6(4)*escala(2)]);
set(handles.uibuttongroup7,'Position',[sp7(1)*
escala(1),sp7(2)*escala(2),sp7(3)*escala(1),sp
7(4)*escala(2)]);
set(handles.uibuttongroup6,'Position',[sp8(1)*
escala(1),sp8(2)*escala(2),sp8(3)*escala(1),sp
8(4)*escala(2)]);
    %subpanel acel y fft en gui
set(handles.gfftacelx,'Position',[sp9(1)*escal
a(1),sp9(2)*escala(2),sp9(3)*escala(1),sp9(4)*
escala(2)]);
set(handles.gfftacely,'Position',[sp10(1)*esca
la(1),sp10(2)*escala(2),sp10(3)*escala(1),sp10
(4)*escala(2)]);
set(handles.gfftacelz,'Position',[sp11(1)*esca
la(1),sp11(2)*escala(2),sp11(3)*escala(1),sp11
(4)*escala(2)]);
    %subpanel acel en gui
set(handles.graficarx,'Position',[sp12(1)*esca
la(1),sp12(2)*escala(2),sp12(3)*escala(1),sp12
(4)*escala(2)]);
set(handles.graficarY,'Position',[sp13(1)*esca
la(1),sp13(2)*escala(2),sp13(3)*escala(1),sp13
(4)*escala(2)]);
```

```
set(handles.graficarZ,'Position',[sp14(1)*escala(1),sp14(2)*escala(2),sp14(3)*escala(1),sp14(4)*escala(2)]);
    %subpanel fft en gui
set(handles.grafffftx,'Position',[sp15(1)*escala(1),sp15(2)*escala(2),sp15(3)*escala(1),sp15(4)*escala(2)]);
set(handles.grafffty,'Position',[sp16(1)*escala(1),sp16(2)*escala(2),sp16(3)*escala(1),sp16(4)*escala(2)]);
set(handles.grafffftz,'Position',[sp17(1)*escala(1),sp17(2)*escala(2),sp17(3)*escala(1),sp17(4)*escala(2)]);
%panel graficas detalladas
set(handles.uipanel2,'Position',[sp000(1)*escala(1),sp000(2)*escala(2),sp000(3)*escala(1),sp000(4)*escala(2)]);
set(handles.uibuttongroup1,'Position',[sp18(1)*escala(1),sp18(2)*escala(2),sp18(3)*escala(1),sp18(4)*escala(2)]);
set(handles.uibuttongroup2,'Position',[sp19(1)*escala(1),sp19(2)*escala(2),sp19(3)*escala(1),sp19(4)*escala(2)]);
set(handles.uibuttongroup3,'Position',[sp20(1)*escala(1),sp20(2)*escala(2),sp20(3)*escala(1),sp20(4)*escala(2)]);
    %subpanel acel y fft en graf detalladas
set(handles.acfftx,'Position',[sp21(1)*escala(1),sp21(2)*escala(2),sp21(3)*escala(1),sp21(4)*escala(2)]);
set(handles.acffty,'Position',[sp22(1)*escala(1),sp22(2)*escala(2),sp22(3)*escala(1),sp22(4)*escala(2)]);
set(handles.acfftz,'Position',[sp23(1)*escala(1),sp23(2)*escala(2),sp23(3)*escala(1),sp23(4)*escala(2)]);
   %subpanel acel graf detalladas
set(handles.acx,'Position',[sp24(1)*escala(1),sp24(2)*escala(2),sp24(3)*escala(1),sp24(4)*escala(2)]);
set(handles.acy,'Position',[sp25(1)*escala(1),sp25(2)*escala(2),sp25(3)*escala(1),sp25(4)*escala(2)]);
set(handles.acz,'Position',[sp26(1)*escala(1),sp26(2)*escala(2),sp26(3)*escala(1),sp26(4)*escala(2)]);
   %subpanel FFT graf detalladas
set(handles.gffx,'Position',[sp27(1)*escala(1),sp27(2)*escala(2),sp27(3)*escala(1),sp27(4)*escala(2)]);
set(handles.gffy,'Position',[sp28(1)*escala(1),sp28(2)*escala(2),sp28(3)*escala(1),sp28(4)*escala(2)]);
set(handles.gffz,'Position',[sp29(1)*escala(1),sp29(2)*escala(2),sp29(3)*escala(1),sp29(4)*escala(2)]);
%panel Repuesta
set(handles.respuesta,'Position',[sp0000(1)*escala(1),sp0000(2)*escala(2),sp0000(3)*escala(1),sp0000(4)*escala(2)]);
set(handles.respload,'Position',[sp30(1)*escala(1),sp30(2)*escala(2),sp30(3)*escala(1),sp30(4)*escala(2)]);
set(handles.trxt3,'Position',[sp31(1)*escala(1),sp31(2)*escala(2),sp31(3)*escala(1),sp31(4)*escala(2)]);
set(handles.RIG,'Position',[sp32(1)*escala(1),sp32(2)*escala(2),sp32(3)*escala(1),sp32(4)*escala(2)]);
set(handles.text13,'Position',[sp33(1)*escala(1),sp33(2)*escala(2),sp33(3)*escala(1),sp33(4)*escala(2)]);
set(handles.TiempoResp,'Position',[sp34(1)*escala(1),sp34(2)*escala(2),sp34(3)*escala(1),sp34(4)*escala(2)]);
set(handles.RunResp,'Position',[sp35(1)*escala(1),sp35(2)*escala(2),sp35(3)*escala(1),sp35(4)*escala(2)]);
set(handles.text14,'Position',[sp36(1)*escala(1),sp36(2)*escala(2),sp36(3)*escala(1),sp36(4)*escala(2)]);
set(handles.respX,'Position',[sp37(1)*escala(1),sp37(2)*escala(2),sp37(3)*escala(1),sp37(4)*escala(2)]);
set(handles.respY,'Position',[sp38(1)*escala(1),sp38(2)*escala(2),sp38(3)*escala(1),sp38(4)*escala(2)]);
set(handles.respZ,'Position',[sp39(1)*escala(1),sp39(2)*escala(2),sp39(3)*escala(1),sp39(4)*escala(2)]);
set(handles.respdet,'Position',[sp40(1)*escala(1),sp40(2)*escala(2),sp40(3)*escala(1),sp40(4)*escala(2)]);
function uipanel1_SizeChangedFcn(~, ~, ~)
function uipanel2_SizeChangedFcn(~, ~, ~)
function respuesta_SizeChangedFcn(~, ~, ~)
function respdet_Callback(~, ~, ~)

%Graficas de respuesta (detalladas)
rx1 = evalin('base','out.RespX1');
trx1 = evalin('base','out.tout');
figure('Name','Respuesta Ejes X')
subplot(2,2,1)
plot(trx1,rx1)
title('Respuesta X1')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

rx2 = evalin('base','out.RespX2');
trx2 = evalin('base','out.tout');
subplot(2,2,2)
plot(trx2,rx2)
title('Respuesta X2')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

rx3 = evalin('base','out.RespX3');
trx3 = evalin('base','out.tout');
subplot(2,2,3)
plot(trx3,rx3)
title('Respuesta X3')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

rx4 = evalin('base','out.RespX4');
trx4 = evalin('base','out.tout');
subplot(2,2,4)
plot(trx4,rx4)
title('Respuesta X4')
xlabel ('t')
ylabel ('Amplitud')
grid on
```

```
    grid minor

    %%
ry1 = evalin('base','out.RespY1');
try1 = evalin('base','out.tout');
figure('Name','Respuesta Ejes Y')
subplot(2,2,1)
plot(try1,ry1)
title('Respuesta Y1')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

ry2 = evalin('base','out.RespY2');
try2 = evalin('base','out.tout');
subplot(2,2,2)
plot(try2,ry2)
title('Respuesta Y2')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor


ry3 = evalin('base','out.RespY3');
try3 = evalin('base','out.tout');
subplot(2,2,3)
plot(try3,ry3)
title('Respuesta Y3')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

ry4 = evalin('base','out.RespY4');
try4 = evalin('base','out.tout');
subplot(2,2,4)
plot(try4,ry4)
title('Respuesta Y4')
xlabel ('t')
ylabel ('Amplitud')
grid on
```

```
    grid minor
%%
rz1 = evalin('base','out.RespZ1');
trz1 = evalin('base','out.tout');
figure('Name','Respuesta ejes Z')
subplot(2,2,1)
plot(trz1,rz1)
title('Respuesta Z1')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

rz2 = evalin('base','out.RespZ2');
trz2 = evalin('base','out.tout');
subplot(2,2,2)
plot(trz2,rz2)
title('Respuesta Z2')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor


rz3 = evalin('base','out.RespZ3');
trz3 = evalin('base','out.tout');
subplot(2,2,3)
plot(trz3,rz3)
title('Respuesta Z3')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor

rz4 = evalin('base','out.RespY4');
trz4 = evalin('base','out.tout');
subplot(2,2,4)
plot(trz4,rz4)
title('Respuesta Z4')
xlabel ('t')
ylabel ('Amplitud')
grid on
    grid minor
```

Código de adquisición de datos

```
clear all
%close all
%clc

format short
delete(instrfind({'port'},{'COM3'}));
canal_serie = serial('COM3','BaudRate',9600,'Terminator', 'CR/LF');
fopen(canal_serie);
figure('Name','Adquisición de datos')
title ('4MPU6050')
xlabel('Puntos'); ylabel('Datos');
grid on; hold on; grid minor;
prop = line(nan,nan,'Color','g','LineWidth',1);
prop1 = line(nan,nan,'Color','k','LineWidth',1);
prop2 = line(nan,nan,'Color','b','LineWidth',1);
```

```
prop3 = line(nan,nan,'Color','r','LineWidth',1);
legend('Senal 1','Senal 2','Senal 3', 'senal 4');  %% Podeis poner el mombre que
querais. Las señales llegan en el mismo orden que las pongamos en Arduino
dato = fscanf(canal_serie, '%f %f %f %f',[4,1]);

i = 1;

figure(1);
uicontrol('String','Parar','Callback','parada=0;');

parada=1;

while (parada)

    dato = fscanf(canal_serie, '%f %f %f %f',[4,1]);
    t(i) = dato(1,:);  %% POTENCIA
    x(i) = dato(2,:);  %% PITCH
    y(i) = dato(3,:);  %% ROLL
    z(i) = dato(4,:);  %% ROLL

    if i<1000
    set(prop,'YData', t(1:i),'XData',(1:i));
    set(prop1,'YData', x(1:i),'XData',(1:i));
    set(prop2,'YData', y(1:i),'XData',(1:i));
    set(prop3,'YData', z(1:i),'XData',(1:i));
    end

    if i>1000
    set(prop,'YData', t(i-1000:i),'XData',(i-1000:i));
    set(prop1,'YData', x(i-1000:i),'XData',(i-1000:i));
    set(prop2,'YData', y(i-1000:i),'XData',(i-1000:i));
    set(prop3,'YData', z(i-1000:i),'XData',(i-1000:i));
    end

    xlim([i-200 i])
    drawnow;
    i = i+1;
end
fclose(canal_serie);


i=i-1;
tiempo=1:1:i;
tiem=tiempo;

%t=t';
%x=x';
%y=y';
%z=z';

Acel=[t;tiem;x;y;z];
Acel=Acel';
```