REPÚBLICA DEL ECUADOR

Escuela Politécnica Nacional

"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN ANDROID PARA EL ANÁLISIS DE VIBRACIONES USANDO EL ACELERÓMETRO DE UN SMARTPHONE

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN "ELECTRÓNICA Y TELECOMUNICACIONES"

GIOVANNA LUCÍA LÓPEZ ARMAS

DIRECTOR: Dr. DIEGO JAVIER REINOSO CHISAGUANO

CODIRECTOR: Ing. RICARDO XAVIER LLUGSI CAÑAR, MSc.

Quito, marzo 2018

AVAL

Certificamos que el presente trabajo fue desarrollado por Giovanna Lucía López Armas, bajo nuestra supervisión.

Dr. DIEGO JAVIER REINOSO CHISAGUANO

DIRECTOR DEL TRABAJO DE TITULACIÓN

Ing. RICARDO XAVIER LLUGSI CAÑAR, MSc.

CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo Giovanna Lucía López Armas, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

GIOVANNA LUCÍA LÓPEZ ARMAS

DEDICATORIA

Este trabajo se lo dedico principalmente a quiénes representan mi mayor tesoro en la vida, mi motor de cada día, mi razón por la cual me he levantado una y otra vez, y lo seguiré haciendo miéntras pueda poner una sonrisa en sus rostros y brindarles esa sensación de orgullo y satisfacción.

A quiénes desde pequeña me han apoyado en cada reto o aventura que he emprendido y han sabido guiarme en cada uno de ellos para llegar a conseguirlo. Quiénes desde mis primeros pasos me enseñaron como hacerlo mejor y como llegar a correr. Quiénes curaban mis rodillas tras cada caída y remendaban mis mallas de lana para volverlas a usar como si nada hubiese pasado, enseñándome así, de forma tan sencilla que las heridas sanan y que no hay nada tan roto que no se pueda remendar.

Todo lo que soy es gracias a ustedes, mi papito hermoso, mi eterno protector Sr. Jaime López Rosas, mi gordita preciosa, mi guerrera ejemplar Sra. Guadalupe Armas Cadena. Es a ustedes, mis papitos queridos, a quiénes les debo todo y a quiénes dedico cada logro alcanzado, porque esto lo conseguimos juntos.

Mis logros son suyos y me quiero convertir en su mejor reflejo, para que el resto del mundo vea en mí la gran educación y valores que me han dado.

Un hijo representa la imagen de sus padres y yo quiero que sepan todos, las magníficas personas que ustedes son. Los amo con mi vida y a ustedes les entrego esta primer gran meta alcanzada.

AGRADECIMIENTO

Este proyecto no habría sido posible sin cada una de las personas que me acompañaron durante este trayecto. He podido quedarme con lo mejor de cada uno, grandes valores, enseñanzas, experiencias, consejos y todo lo que me han entregado, convirtiéndose para mí en un apoyo incondicional en mi vida tanto personal como profesional.

Quiero agradecer primeramente a quiénes contribuyeron en mi formación como profesional, especialmente a quién me ha dirigido en este proyecto, Dr. Diego Reinoso, quién tuvo toda la predisposición para culminar todos los objetivos planteados. Gracias por los consejos, guías y todos los materiales que puso a mí disposición para poder llegar a culminar cada etapa de este trabajo.

A mis padres, mis hermanos Paúl y Jaime, mi Drinky y mi Matías, quiénes en cada quiebre que he tenido han sabido apoyarme y ayudar a levantarme para poder llegal al final de esta meta. Gracias, porque ustedes se convertían en mi empuje y pañuelo para poder super cada obstáculo. A mis tíos, primos y demás familiares, les debo mis agradecimientos a cada uno por demostrarme su apoyo de una u otra forma, especialmente, a mis tíos Rommel y Teresita, quiénes han sido un apoyo fundamental y se han llegado a convertir en otros padres para mí, gracias por cuidarme y preocuparse desde el día que vine sola a esta extraña ciudad. A mis ñaños Jaime, Julio, Jhenny, Yadira, mis sobrinos, Wilo, Sra. Loly y a todos quiénes me acompañaron en el trayecto y lo vivieron conmigo, ya sea en pequeños o grandes momentos, pero para mí han sido muy significativos y me han ayudado a llegar a donde estoy. Mi cariño y agradecimiento será eterno hacia ustedes.

Quiero agradecer también a una persona muy especial quién me acompañó en esta última etapa, siendo un gran aporte y apoyo para hacerme saber que todo se puede lograr cuando lo queremos. Gracias Carlitos, le debo mucho y gracias por todo lo que me brindó en este tiempo. Lo quiero mucho.

Y que es la vida sin los grandes amigos que uno hace en el camino. A mis hermanas Liss y Chinita gracias por cambiar las tristezas por locuras. A mis morenitas Alejita y Fersita, a mis ñaños Alexis, Isma e Isra. Xavy, Jona, Leo, Vickyto, Danny, Karlita, Pame, Carlitos, Santi C., Jairo, mi negro Edu, Juank, Santi N., Juani y a todos quiénes los he llamado amigos siéntasen identificados aquí. Un Dios le pague por cada aventura compartida.

Finally, I want to thank a great friend who, without knowing me, accepted to help me and clarified many doubts. I never hesitate to share his knowledge in order to finish this project. Dear Julian Bunn you are great, thank you very much for helping me from so far away. I appreciate you so much from Ecuador.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

RESUMEN

El presente proyecto trata sobre el uso de la medición y análisis de vibraciones como un mecanismo para poder realizar un estudio in situ de un evento que ha producido una señal de vibración como resultado. Esto se puede suscitar en diversos casos como son la sismografía, fallos en maquínarias, riesgos en edificios frente a un evento natural o deterioro de una estructura debido a vibraciones aleatorias, entre otras eventualidades que se generan debido a vibraciones.

Este proyecto presenta como solución una aplicación que permite acceder a la información del acelerómetro integrado dentro de los Smartphones con un sistema operativo Android. El acelerómetro tiene la capacidad de detectar aceleración y vibraciones en los tres ejes.

La aplicación permite detectar y guardar la señal del acelerómetro en un archivo de texto plano y realizar un análisis para adquirir parámetros como el PGA (Peak Ground Acceleration) que se lo define como el valor máximo absoluto de la señal detectada, más utilizado en sismografía y la frecuencia más predominante mediante el análisis espectral con ayuda de la Transformada de Fourier. Estos datos obtenidos por la aplicación permiten un análisis de las vibraciones de forma rápida, sencilla y utilizando un dispositivo que normalmente siempre está con nosotros.

Se finaliza este proyecto con la implementación de la aplicación en dos diferentes Smartphones para comprobar su correcto funcionamiento. Además, se obtienen conclusiones de las pruebas de comparación con dos equipos dedicados a este tipo de mediciones, el sensor 1043_0 – PhidgetsSpatial Precision 0/0/3 y el Guralp CMG-5TD.

PALABRAS CLAVE: acelerómetro, Android, FFT, PGA, Smartphone, vibraciones.

ABSTRACT

This project is about the use of vibration measurement and analysis as a mechanism to perform an in situ study of an event that has produced an vibration signal as a result. This can arise in various cases such as seismography, machinery failures, building risks in the face of a natural event or deterioration of a structure due to random vibrations, among other eventualities that are generated due to vibrations.

This project presents as solution an application that allows access to the information of the accelerometer integrated into the Smartphones with an Android operating system. The accelerometer has the ability to detect acceleration and vibrations in all three axes.

The application allows detecting and saving the accelerometer signal into a plain text file and allows an analysis to acquire parameters such as the PGA (Peak Ground Acceleration) that is defined as the absolute maximum value of the detected signal, most used in seismography and the most predominant frequency through the spectral analysis with the help of the Fourier Transform. This data obtained by the application allows an analysis of the vibrations quickly, easily and using a device that usually is always with us.

This project is completed with the implementation of the application in two different Smartphones to verify its correct functioning. In addition, the conclusions are obtained through comparisons with two types of instruments dedicated to this type of measurements, the sensor 1043_0 - PhidgetsSpatial Precision 0/0/3 and the accelerometer Guralp CMG-5TD.

KEYWORDS: accelerometer, Android, FFT, PGA, Smartphone, vibrations.

1. INTRODUCCIÓN

Este proyecto tiene como enfoque el desarrollo de una aplicación dedicada para dispositivos inteligentes con un sistema operativo Android, la cual sirve para el análisis de vibraciones usando el acelerómetro integrado en el Smartphone.

La importancia del desarrollo de este proyecto radica en la gran utilidad que representa dentro del análisis de vibraciones, ya que, gracias a sus ventajas de portabilidad, bajo costo y fácil manejo, se puede realizar una evaluación inmediata de las vibraciones cuando no se tiene a mano un equipo profesional de medición, de esta forma se facilita el análisis del problema y una pronta respuesta a la solución.

El análisis de vibraciones tiene un amplio campo de usos y aplicaciones, este proyecto se centra para aplicaciones en el campo de la Ingeniería Civil y para el análisis de vibraciones mecánicas. En el campo de la Ingeniería Civil es de gran utilidad para el análisis in situ de las estructuras, edificios o suelos; detectando y guardando la señal generada para posteriormente a través del análisis obtener parámetros como el PGA (Peak Ground Acceleration), parámetro muy usado en sismografía y la frecuencia fundamental de una estructura, que es el resultado del procesamiento de la señal con la Transformada de Fourier. De igual forma dentro del uso del análisis de vibraciones mecánicas se requiere obtener la frecuencia fundamental de la vibración de la máquina, y de acuerdo a esto verificar si se encuentra dentro de los rangos aceptables de frecuencias de vibración definidas por los fabricantes. Dado el caso de darse una frecuencia de vibración diferente, se puede detectar si existe un componente fallando dentro de la maquinaria como un tornillo, engrane u otros elementos que la conforman.

1.1 Objetivos

El objetivo general de este Proyecto Técnico es: “Desarrollar una aplicación compatible con el sistema operativo Android que permita el análisis de las vibraciones con los datos obtenidos del acelerómetro de un Smartphone”

Los objetivos específicos de este Proyecto Técnico son:

- Desarrollar una aplicación compatible con el sistema operativo Android que permita graficar la aceleración de las vibraciones en los tres ejes XYZ en tiempo real.
- Permitir el almacenamiento de los datos de la aceleración generados por el acelerómetro en un intervalo de tiempo para poder obtener el espectro y realizar su procesamiento con la FFT (Fast Fourier Transform).

- Encontrar el parámetro PGA (Peak Ground Acceleration) para identificar el pico con mayor amplitud positivo o negativo.
- Comparar los resultados obtenidos de la aplicación, instalada en dos Smartphone diferentes.
- Realizar pruebas y comparaciones con equipos especializados para este tipo de mediciones como son: el sensor 1043_0 - PhidgetsSpatial Precision 0/0/3 (media gama) y el acelerógrafo Guralp CMG-5TD (alta gama).

1.2 Alcance

El proyecto está orientado a la realización de una aplicación compatible con el sistema operativo Android, la cual tendrá características de transportabilidad, bajo costo y de fácil manejo. La aplicación permitirá realizar un análisis de las vibraciones de un objeto u ambiente mediante el registro de la aceleración proporcionada por el acelerómetro del Smartphone. Este análisis se realizará con la Transformada de Fourier la cual permitirá obtener las gráficas del espectro y a su vez extraer los valores de la frecuencia más notable que se produce en una vibración. La aplicación también permitirá obtener el parámetro PGA (Peak Groun Acceleration) y el tiempo en que se produce.

Se realizará la implementación en dos diferentes Smartphones para comprobar su correcto funcionamiento y posteriormente se efectuará pruebas de comparación con los resultados obtenidos con el sensor 1043_0 - PhidgetsSpatial Precision 0/0/3 (media gama) [1] y el acelerógrafo Guralp CMG-5TD (alta gama) [2].

1.3 Marco Teórico

1.3.1 Vibraciones

La medición de vibraciones se ha reconocido como un importante método de análisis desde que se vio la posibilidad de extraer datos de una señal generada. Estos datos pueden ser analizados y tratados con diferentes métodos, de acuerdo con la necesidad del investigador y extraer cierta información que sirven para llegar a conclusiones de un problema o situación.

La vibración se define como la variación de un sistema con respecto a su punto de equilibrio estable, en relación con el tiempo [3]. El estudio de vibraciones puede incluir vibraciones periódicas como es el caso de sistemas mecánicos y vibraciones aleatorias que pueden surgir en casos relacionados con la sismografía, donde se analizan fenómenos naturales

emergentes y pueden generar este tipo de señales aleatorias. Las excitaciones periódicas presentan como ventaja que basta con extraer un periodo de la vibración para realizar su análisis y extraer la información necesaria, no obstante las vibraciones aleatorias son las que se encontraran en un mayor porcentaje dentro del campo de trabajo donde sea indispensable realizar un análisis de vibraciones [4].

Para realizar estas mediciones se han requerido equipos que realizan la tarea específica de detectar una señal de vibración y grabarla, para posteriormente con la ayuda de un software o mediante procesos matemáticos poder tratarla y analizarla, a estos equipos se los denomina sensores. Las vibraciones se caracterizan por la frecuencia, desplazamiento, aceleración o velocidad, dirección y duración, y se cuantifican en función a estas.

Frecuencia

Se define como el número de ciclos que se cumplen dentro de un periodo de tiempo, su unidad de medida es Hertzios (Hz). La frecuencia afecta a la prolongación con la que se transmiten las vibraciones a través del sistema o estructura. Existe una relación entre la aceleración y el desplazamiento a través de la frecuencia como se indica en la Ecuación 1.1.

$$a = (2\pi f)^2 \,.\, d$$

Ecuación 1.1 Aceleración en función del desplazamiento [3].

Para conocer esta frecuencia a la que oscilan los cuerpos se utiliza el análisis espectral, de donde se puede visualizar y obtener de forma más clara los picos en aquellas frecuencias a las cuales la vibración es mayor.

Se debe tomar en cuenta en lo referente a una estructura, la intensidad de la vibración se incrementa cuando la frecuencia a la que vibra coincide con la frecuencia de resonancia[1] de la estructura. Por lo que se debe prestar atención con estas frecuencias ya que se puede generar daños, ya sea en el caso de maquinarias, estructuras o cualquier sistema, inclusive si un humano se somete a estas frecuencias se puede producir efectos nocivos.

[1] Frecuencia de resonancia de una estructura: se define así a la frecuencia natural de cada estructura inherente a sí misma. Cuando una frecuencia de vibración producida por una excitación externa (sismo, la fuerza del viento, motor sobre una estructura, etc.) se aproxima a este valor de frecuencia natural, se producirá el fenómeno de resonancia generando desplazamientos exagerados en la estructura.

Desplazamiento y velocidad

El desplazamiento representa la distancia que recorre el sistema con respecto a su posición de reposo o equilibrio, dentro del análisis de vibraciones la unidad de medida se expresa en Micrones (μm). La velocidad representa la proporción de cambio del desplazamiento con respecto al tiempo, en un espectro de vibración la unidad de medida se encuentra en pulgadas por segundo (in/s) [5].

El desplazamiento de un sistema que está sometido a un movimiento armónico simple genera una onda sinusoidal, al igual que la velocidad y aceleración del movimiento. En el caso donde el desplazamiento se encuentra en su valor máximo la velocidad es igual a cero, ya que es en ese momento cuando la dirección del movimiento se invierte; por lo contrario, cuando el desplazamiento tiene un valor cero, la velocidad tendrá un valor máximo, indicando que la fase de la sinusoidal de la velocidad tiene un desplazamiento de 90° hacia la izquierda en comparación con la sinusoidal del desplazamiento; es decir, la velocidad se adelanta 90° al desplazamiento [6].

Aceleración

Esta magnitud brinda una medida correspondiente al cambio de velocidad con respecto al tiempo, se la relaciona con la fuerza que produce la vibración.

Una fuerza se la define como un esfuerzo, acción o influencia que altera el estado de reposo o equilibrio de un objeto; lo que quiere decir, que la fuerza da una aceleración de forma que modifica su velocidad, su dirección y el sentido del movimiento. La unidad de medida en un espectro de vibración se expresa en gravedades (g's) y se encuentra relacionada con la aceleración de la gravedad [5].

La velocidad y la aceleración se relacionan de tal forma que cuando la velocidad se encuentra en su valor máximo; es decir, ya no cambia, por lo tanto la aceleración es cero. Caso contrario, cuando la aceleración toma su máximo valor, la velocidad es cero ya que cambia rápidamente.

La sinusoide de la aceleración, generada cuando un cuerpo se encuentra sometido a un movimiento armónico simple, en función del tiempo presenta un desfase de 90° hacia la izquierda con relación a la sinusoide de la velocidad y por lo tanto de 180° en relación al desplazamiento [6].

Los desfases entre estas magnitudes claves en el análisis de vibraciones se presenta en la Figura 1.1.

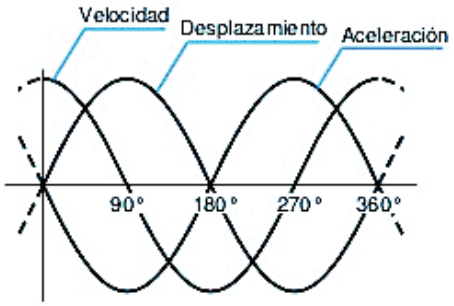


Figura 1.1. Desfase entre velocidad, desplazamiento y aceleración [6].

El análisis de vibraciones se lo puede realizar con cada una de las señales de vibración ya sea de velocidad, desplazamiento y aceleración. La señal de vibraciones que se debe analizar depende de la frecuencia en la que se trabaje. En la Figura 1.2. se puede observar los rangos de frecuencia en donde se manifiestan cada una de las magnitudes.

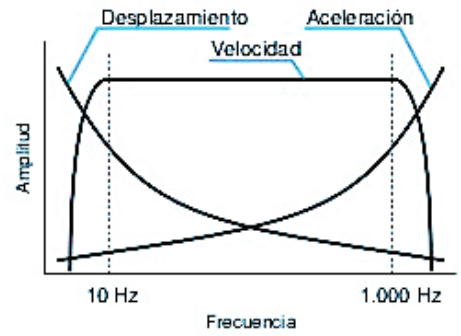


Figura 1.2. Velocidad, desplazamiento y aceleración en frecuencia [6].

El desplazamiento tiene una mayor amplitud en bajas frecuencias, típicamente debajo de los 10 Hz, la velocidad se presenta dentro de las frecuencias 10 Hz y 1000 Hz, y finalmente la aceleración se destaca con una mayor amplitud en frecuencias altas, por sobre 1000 Hz generalmente.

Dirección

La mayoría de los acelerómetros utilizados para la medición de las vibraciones son triaxiales, debido a que las vibraciones se producen en estas tres direcciones; sin embargo, también existen direcciones rotacionales. Las vibraciones detectadas en los ejes: X, Y y Z,

se las conoce como direcciones líneas y se las denomina: Longitudinal, Lateral y Vertical respectivamente. Las direcciones rotacionales son aquellas que ejercen una rotación alrededor de cada uno de los ejes y se los denomina: balanceo (alrededor del eje X), cabeceo (alrededor del eje Y) y deriva (alrededor del eje Z).

1.3.2 Instrumento de medición de vibraciones

Dentro de la medición de vibraciones se ocupan equipos específicos conocidos como acelerómetros o acelerógrafos, los cuales detectan las oscilaciones y vibraciones de una máquina, una instalación, una estructura o cualquier sistema en el que se produzca una excitación y se altere su posición de equilibrio. Los medidores de vibración representan una ayuda insustituible in situ para el profesional, por lo que, se caracterizan por ser portátiles y pueden almacenar los resultados.

El acelerómetro es un instrumento electromécanico para medir las aceleraciones estáticas o dinámicas, como la constante gravitatoría o aceleraciones generadas por vibraciones respectivamente. Este al ser un sensor tiene como objetivo captar una señal eléctrica que es una diferencia de potencial, proporcional a la magnitud a medir, como en este caso la aceleración. Dependiendo de la tecnología que utilicen para medir esta magnitud, se diferencian los tipos de acelerómetros; existen: mecánicos, capacitivos, piezoeléctricos, láser, de inducción magnética, ópticos [3].

Funcionamiento de un acelerómetro

Ya que la aceleración no se puede medir por una observacipon directa, basa su funcionamiento en la medición de otras variables y conjuntamente con las leyes que rigen sus efectos se obtiene la magnitud de medida, como en este caso la aceleración.

El funcionamiento de los acelerómetros se evidencia en el acelerómetro mecánico, que consta de dos partes fundamentales: Una base unida al objeto del cual se desea medir la aceleración y una masa, que a pesar de estar unida a la base mediante un mecanismo elástico, ésta puede moverse y romper su posición de equilibrio; esto se indica en la Figura 1.3. Al cambiar la posición del sistema se puede medir la distancia de elongación del resorte y calcular la fuerza de gravedad; esto se realiza con la ayuda de la Ley de la Elasticidad de Hooke y la Segunda Ley de Newton.

Figura 1.3. Mecanismo de un acelerómetro mecánico [7].

Según la Ley de Hooke: El alargamiento que experimenta un material elástico, como el resorte que sostiene la masa dentro del tubo, es directamente proporcional al módulo de la fuerza aplicada. Entonces se tiene que:

$$F = K \, . \, x$$

Ecuación 1.2 Ecuación de la Ley de Hooke [7].

Donde, F representa a la fuerza aplicada, K es la constante de elongación del resorte y *x* la distancia desplada por la masa pendiendo del resorte. Posteriormente, al aplicar el concepto de la segunda Ley de Newton: La aceleración de un elemento es proporcional a la fuerza que actúa sobre él y ésta a su vez es directamente proporcional a la masa del objeto, teniendo así la ecuación:

$$F = m \, . \, a$$

Ecuación 1.3 Segunda Ley de Newton [7].

Por lo tanto, cumpliendo estas dos leyes y dado que la fuerza F en ambos casos es la misma aplicada, se puede establecer una igualdad de donde se obtiene la aceleración. Los datos de la constante de elongación y la masa son conocidos, y la distancia desplazada x se puede medir. Entonces de esta forma se puede obtener la aceleración sobre la línea que se ejerce la fuerza.

Al juntar tres de estas bases (tubos) se puede simular los tres ejes de coordenadas tridimensionales, cumpliendo su ortogonalidad. De este modo, se obtienen las tres componentes de la aceleración en cualquier dirección como se indica en la Figura 1.4.

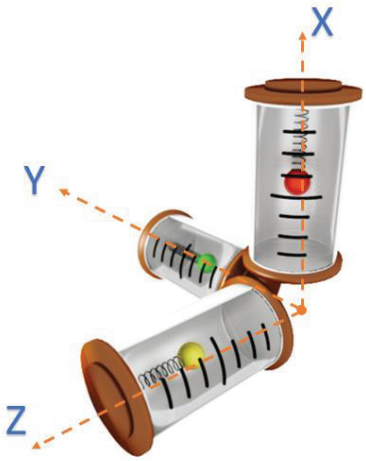


Figura 1.4. Ubicación de tres tubos para simular la orientación tridimensional del móvil [7].

Replicar este mecanismo en un tamaño reducido que se pueda implementar dentro de los teléfonos inteligentes puede resultar un tanto difícil, sin embargo se aplican distintos fenómenos físicos con base en la misma idea.

Acelerómetro piezoeléctrico

El fenómeno en el que basa su funcionamiento este tipo de acelerómetro, es el que ocurre en materiales con este mismo nombre, los elementos piezoeléctricos comúnmente hechos con circonato de plomo. Cuando el retículo cristalino es comprimido bajo una fuerza aplicada en una dirección, se genera una carga eléctrica que es proporcional a dicha fuerza. Los efectos que se producen en un acelerómetro piezoeléctrico cuando se detecta una vibración se puede evidenciar en la Figura 1.5 [3].

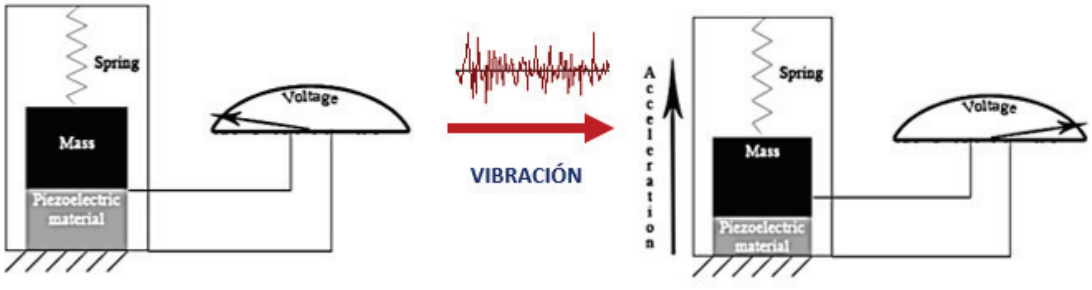


Figura 1.5. Funcionamiento de un acelerómetro piezoeléctrico sometido a una vibración.

Cuando se encuentra sometido a una vibración, la masa producirá un voltaje debido a la fuerza que se ejercerá sobre el cristal piezoeléctrico. Al medir este voltaje que se genera,

dependiendo del tipo de acelerómetro, se tendrá un potencial variable proporcional a la aceleración y de esta forma se puede cuantificar la aceleración.

Acelerómetro piezoresistivo

Tiene un funcionamiento similar que el anterior, con la diferencia del material con el que trabajan, como su nombre lo indica se utiliza un material piezoresistivo que es el sustrato, Figura. 1.6. El sustrato sufre una deformación cuando es sometido a una vibración y de esta forma cambia la resistencia del material en función de la deformación que sufre.

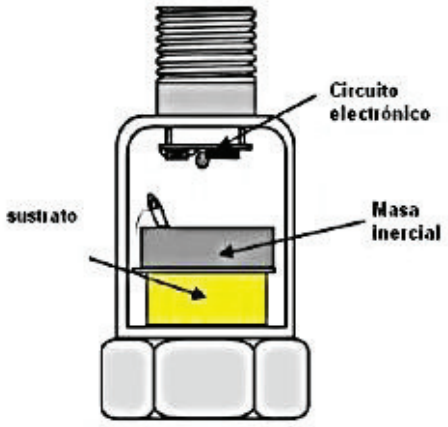


Figura 1.6. Esquema de un acelerómetro piezoresistivo [8].

El valor de la resistencia forma parte de un circuito electrónico que con el fundamento de un puente de Whetstone permite medir la intensidad de corriente y de esta forma cuantificar los valores de aceleración obtenidos. Una ventaja notoria frente a los acelerómetros piezoeléctricos es que da la posibilidad de medir aceleraciones con una frecuencia hasta de 0 Hz.

Acelerómetro capacitivo

Este tipo de acelerómetros se componen de condensadores que permiten almacenar energía eléctrica entre dos placas conductoras separadas una cierta distancia por un material dieléctrico. La distancia que separa estas dos placas viene con una capacitancia definida, por lo tanto, una alteración en la distancia a la que se encuentran separadas las placas, provocará una variante en el valor de su capacitancia como respuesta a la variación de la aceleración.

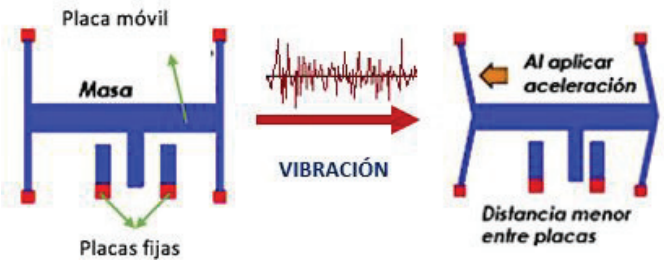


Figura 1.7. Funcionamiento de un acelerómetro capacitivo sometido a vibración.

El material del que están formados los circuitos de estos sensores capacitivos, es el silicio, este material permite reducir los problemas derivados de la humedad, temperatura, capacitancias parásitas, altas impedancias de entrada, entre otros.

Al existir una aceleración o desaceleración en la placa móvil, se efectúa una fuerza en la masa central y desplaza las placas del condensador, alterando su separación y cambiando la capacitancia del condensador, este valor es procesado y se obtiene el voltaje de salida proporcional a la aceleración.

Estos acelerómetros capacitivos son adecuados para producirse en pequeñas dimensiones como circuitos integrados que pueden ser soldados en placas de silicio conjuntamente con un software adecuado para interpretar las señales generadas desde el acelerómetro; es por este motivo que este tipo de acelerómetros se convierten en los más calificados para ser usados en los dispositivos inteligentes.

Aplicaciones comunes del acelerómetro

La aplicación más común de un acelerómetro se encuentra en los aribags del coche. Este sensor detecta una aceleración muy fuerte y hace que el sistema se accione y despliega el airbag de forma inmediata.

Otra aplicación en la cual se utiliza este sensor es para protección de los móviles, se detecta un cambio brusco de velocidad cuando este sufra una caída para evitar que el sistema operativo sufra daños.

Igualmente se abarcan al ocio y ejercicio. Se usan con aplicaciones de actividad física, encargándose de medir la distancia que se recorre o las calorías que se queman, se hacen cálculos y se elaboran estadísticas a partir de estos [9].

1.3.3 Tecnología MEMS

Sus siglas refieren a Sistemas Micro-Electro-Mecánicos (Microelectromechanical Systems), que son dispositivos formados por elementos activos y pasivos de pequeñas dimensiones que perciben, procesan y comunican datos. Estos dispositivos pueden tener una estructura simple sin ninguna parte móvil, así como una estructura electromecánica compleja donde distintos elementos son controlados por una electrónica integrada.

La fabricación de los dispositivos MEMS pueden utilizar distintas técnicas de fabricación y una gran variedad de materiales dependiendo de los fines para los cuales vayan a ser destinados los dispositivos y los usos comerciales que se les vaya a dar. Las dimensiones típicas de los MEMS van desde un micrómetro a un milímetro. Su tamaño reducido se presenta como la principal ventaja dentro de otras como un menor peso, bajo costo, menor consumo de energía y proporciona un gran desempeño.

Dentro de los primeros sensores desarrollados con esta tecnología están los acelerómetros, surgen dentro del final de la década de 1980. Fueron desarrollados principalmente para aplicaciones en los mercados de la automoción y aeronaútica. Posteriormente, la evolución de la tecnología, la producción en gran volumen y su bajo costo, los han proyectado con éxito para implementarse dentro de otras áreas, tales como médica, de transporte e industrial.

Se distingue tres principales categorías de acelerómetros de esta tecnología: acelerómetros capacitivos de silicio, piezoresistivos y acelerómetros térmicos. Sin embargo, los que predominan dentro del mercado son los acelerómetros capacitivos de silicio.

Los acelerómetros MEMS son cada vez más usados en mercados diferentes al área automotriz o de la aviación, donde su principal funcionalidad es medir la inercia. Así, también pueden medir la inclinación que es una característica usada por los transportes, navegación de personas no videntes, telemetría, aplicaciones médicas, monitoreo del estado de las máquinas, mediciones sísmicas [8].

Acelerómetro de un Smartphone

La evolución de la tecnología y de los dispositivos de comunicación, ha permitido contar con un teléfono inteligente al alcance de nuestras manos que permite explotar al máximo sus características de hardware mediante la implementación de un sin número de aplicaciones que manejan sus componentes internos.

Los *Smartphones* están llenos de componentes electrónicos tanto en su interior como en su exterior. Cada uno de ellos tienen una funcionalidad específica, que trabajando en conjunto hacen que el dispositivo móvil sea una potencial herramienta para el desarrollo de aplicaciones. Así, se facilitará la resolución de problemas y análisis in situs de ciertos fenómenos o circunstancias para las cuales se necesita un equipo de medición que se encuentre al alcance en cualquier momento.

La gran mayoría de los componentes internos de un Smartphone son sensores. “Un sensor es un dispositivo capaz de dar una respuesta eléctrica a estímulos físicos externos”. Estos componentes convierten magnitudes físicas o químicas, como la temperatura, aceleración, longitud, nivel de azúcar, etc., en unidades eléctricas que un dispositivo electrónico sea capaz de interpretar, generalmente es el voltaje.

Las unidades posteriormente son analizadas mediante un sistema inteligente que da respuesta a la acción que se haya llevado a cabo; es decir, son analizadas y tratadas para obtener conclusiones y resultados útiles para cada una de las circunstancias en las que se las requiera. La principal función de los sensores integrados en el móvil es el recopilar información importante que estos generan para el funcionamiento, basado en datos que ya se han registrado previamente [9].

El acelerómetro es un chip de tamaño muy reducido, definido como un elemento sensor que permite la medida de la aceleración y es responsable de detectar los cambios de orientación en los terminales, aunque es posible en base a un acelerómetro determinar la orientación, normalmente se usa un giroscopio. Es muy común que en el mismo circuito integrado pueda existir un acelerómetro y giroscopio. En la Figura 1.8 se puede observar la ubicación física dentro del terminal. Los teléfonos móviles utilizan los acelerómetros del tipo capacitivos [10].

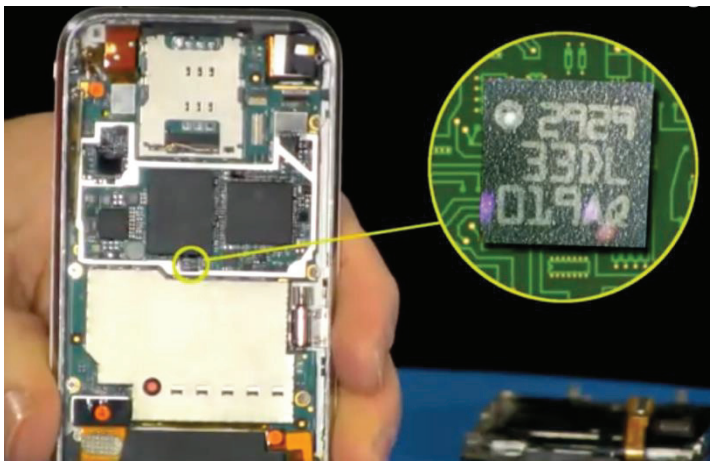


Figura 1.8. Ubicación del acelerómetro en un Iphone 3g [10].

El acelerómetro es uno de los principales sensores en un dispositivo Android y brinda una lectura de la aceleración dada en unidades de m/s^2 que se encuentra repartida en cada uno de los tres ejes de coordenadas. En la Figura 1.9 se muestra como se relaciona los ejes del móvil con las coordenadas principales. Un factor principal que afecta la aceleración del sensor es la gravedad, por lo que siempre existirá una aceleración aproximada de 9,8 m/s^2 en alguno de los ejes de coordenadas dependiendo de la orientación del smartphone.

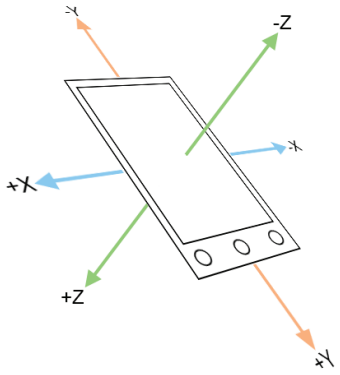


Figura 1.9. Representación de las coordenadas en un smartphone [9].

Dentro del teléfono no se utiliza la masa y el resorte, pero los principios son los mismos. El acelerómetro dentro de un Smartphone esta hecho de silicio, Figura 1.10. Consta de dos placas metálicas una fija y otra móvil que se encuentran una frente a la otra, éstas se componen de un condensador y un material dieléctrico como piezas fundamentales del funcionamiento. Los acelerómetros se encuentran dentro de la categoría de dispositivos electromecánicos denominados MEMS (Micro Electro-Mechanical Systems), ya que se refieren a elementos construidos generalmente a base de silicio policristalino modelado y se miden en micrómetros (μm). La principal ventaja de este tipo de transductores es que pueden ser fabricados un tamaño tan pequeño que su influencia sea casi despreciable en el dispositivo vibrador.

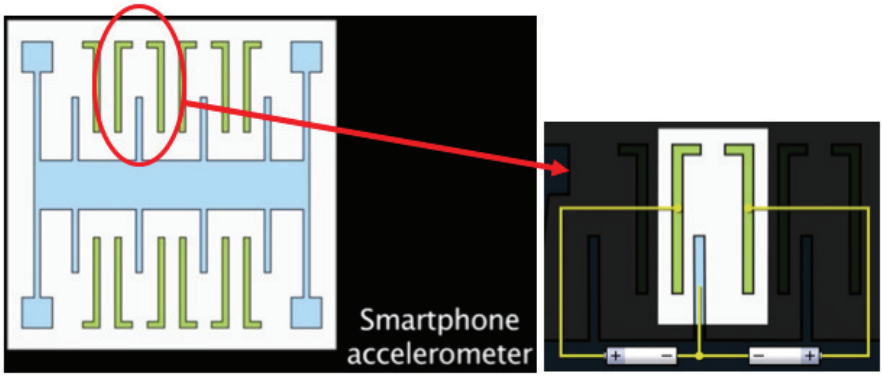


Figura 1.10. Acelerómetro capacitivo hecho de silicio utilizado en los móviles.

El acelerómetro de un *Smartphone* funciona cuando detecta un cambio de su posición de equilibrio, variando sus valores en cada uno de los ejes triaxiales y cambiado de esta forma el valor de la capacidad del condensador. Cuando se produce esta aceleración del dispositivo, la placa del móvil se desplaza hacia donde aumenta o reduce la distancia. El sensor cuantifica y envía la carga que se produce al chip para que lo procese y se ejecute una acción en el sistema, traduciendo el valor de aceleración en una unidad de lectura fácil para el usuario.

El movimiento de las placas se debe a que llega energía procedente de la batería del móvil que será en función de la magnitud del movimiento del dispositivo y se reflejará en cada uno de los ejes de coordenadas como se indica en la Figura 1.11. La carga puede ser almacenada en el material dialéctrico de las placas.

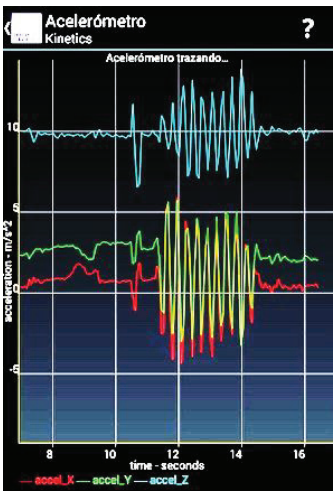


Figura 1.11. El móvil detecta la aceleración a la que se le somete en los tres ejes [11].

Para que no se confunda los movimientos producidos y las acciones que se debe tomar después de una aceleración producida, ya se tiene registrada dentro de una base de datos la carga eléctrica (voltaje) que genera cada uno de los movimientos y así se puede determinar la fuerza y aceleración a la que ha sido sometido el dispositivo móvil. Todo este trabajo se lo realiza en menos de una milésima de segundo [9].

1.3.4 Aplicación del análisis de vibraciones

El análisis de vibraciones es mayormente utilizado para lo que se conoce como mantenimiento predictivo, en donde se implementa una etapa de monitoreo con el fin de establecer el estado mecánico o físico de una máquina, estructura, instalación o un área de investigación en donde se tenga vibraciones y se requiera realizar un estudio de la situación.

El objetivo del análisis de vibraciones es determinar la frecuencia de vibración a la que se encuentran sometidos los sistemas o cuerpos, ya que todos estos presentan una frecuencia de resonancia a la cual sufren cambios o daños cuando coinciden estas frecuencias.

Mecánica

Esta área de aplicación se dirige especialmente al análisis predictivo del estado de maquinarias, particularmente de los elementos que la componen como son: los engranajes, rodamientos, descansos, elementos rotadores, entre otros. El principal objetivo es verificar el correcto trabajo e instalación de la máquina en su lugar de trabajo y así prevenir fallas cuando se encuentre en funcionamiento. En la figura 1.12 se puede ver algunas de las causas por las que se puede producir una frecuencia de vibración diferente a la establecida por el fabricante en el manual de la máquina.

De igual forma se utiliza para verificaciones periódicas de funcionalidad de la máquina, ya que con el uso, los componentes se van desgastando produciendo una variación en la frecuencia de vibración.

Un cambio en el nivel de vibraciones indica un cambio de las condiciones de estabilidad y gracias al acelerómetro se puede detectar y reemplazar los elementos dañados o desgastados, prolongando la vida del equipo y evitando así un desgaste mayor, e incluso un colapso total de la máquinaria, y reducción en los costos que se generan por las paradas de los equipos ya que estos representan cerca del 50% de los costos [12].

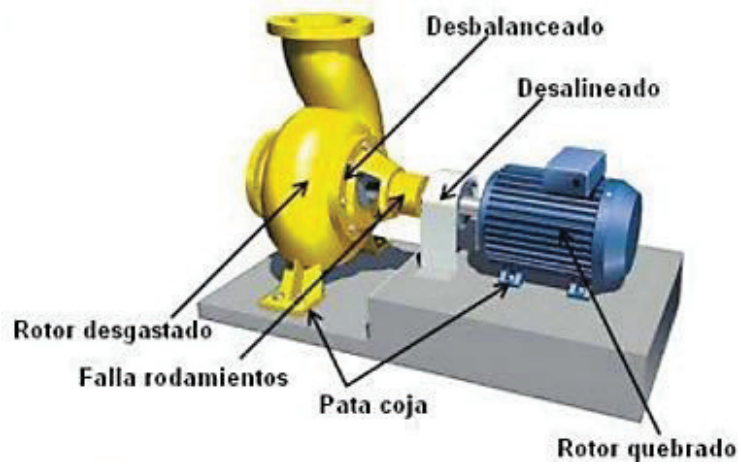


Figura 1.12. Causas que generan variación en las vibraciones de un elemento [13].

Estructuras, edificios e Ingeniería Civil

Dentro de la industria de la construcción se busca obtener los periodos fundamentales de los edificios, estructuras y las diferentes construcciones de esta área, debido a que para el diseño deben cumplir ciertos parámetros o requerimientos mínimos para la condición de sismoresistente, según el código vigente de la “Norma Ecuatoriana de la Construcción” (NEC). La determinación del periodo fundamental se realiza a través del análisis de las vibraciones a las que se encuentran sometidas las estructuras.

Existen vibraciones ambientales que son propias del entorno en el que se desarrollan las estructuras, se define a este tipo de vibraciones a las que son generados por elementos constantes dentro de los sistemas como son: el tráfico de los vehículos alrededor del edificio, maquinarias situadas dentro de la estructura, fenómenos atmosféricos, actividad humana y en sí, cualquier tipo de alteraciones sufridas durante su vida útil, particularmente fenómenos naturales que puedan darse en cualquier momento como sismos o terremotos.

El acelerómetro es el instrumento utilizado para receptar las vibraciones que se tienen en una construcción y posteriormente realizar su análisis mediante la Transformada Rápida de Fourier y obtener la frecuenica de vibración fundamental y por ende, el periodo fundamental de la estructura. La ingeniería de la contrucción utiliza este parámetro del periodo fundamental para desarrollar los cálculos de las fuerzas sísmicas.

Los cálculos que se realizan en la Ingeniería de la contrucción tienen como base el cálculo del periodo fundamental de un edificio, ya que de esta forma se puede tener un historial de los efectos que se pueden producir dependiendo de las eventualidades que se presenten. El análisis es de vital importancia, tanto desde el procedimiento inicial de una obra, donde se requiere un estudio del suelo o de la superficie donde se va a edificar y así como el estudio periódico de las afectaciones que sufren edificaciones o estructuras ya elaboradas; de esta forma se puede tomar medidas preventivas y de mantenimiento para las respectivas estructuras.

En la Figura 1.13. se puede ver los efectos que pueden producir las vibraciones en un edificio y las medidas que tome el profesional para reducir su impacto y prolongar su vida útil.

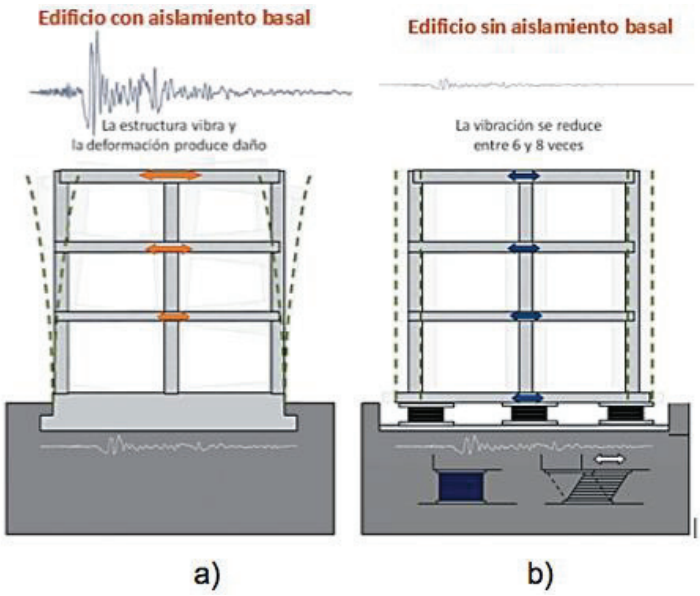


Figura 1.13. a) Efectos de las vibraciones en un edificio. **b)** Medidas preventivas para disminución de los efectos de las vibraciones [14].

De acuerdo a los datos obtenidos del análisis de vibraciones de las estructuras, se puede definir las características de construcción, materiales y aspectos de diseño que se debe implementar o mejorar según sea el caso para antiguas o nuevas contrucciones.

Sismografía

En esta área existen parámetros que sirven para cuantificar los efectos que se pueden producir frente a un terremoto o sismo, uno de ellos es la *aceleración sísmica*, la cual es una medida directa de las aceleraciónes que sufre la superficie del suelo. Se diferencia de otras medidas comúnmente utilizadas para cuantificar terremotos, como la escala de Richter y de magnitud de momento, que esta representa la intensidad más no la magnitud, ya que no es una medida de la energía total liberada.

La medición de la aceleración sísmica se mide por la determinación del PGA *(Peak Ground Acceleration)*, se define como el mayor valor absoluto de la aceleración del suelo producida por las vibraciones durante un sismo o terremoto como se indica en la figura 1.14. Las vibraciones durante un sismo se producen en las tres direcciones del eje de coordenadas, por lo que el PGA se divide a menudo en componentes horizontales y verticales.

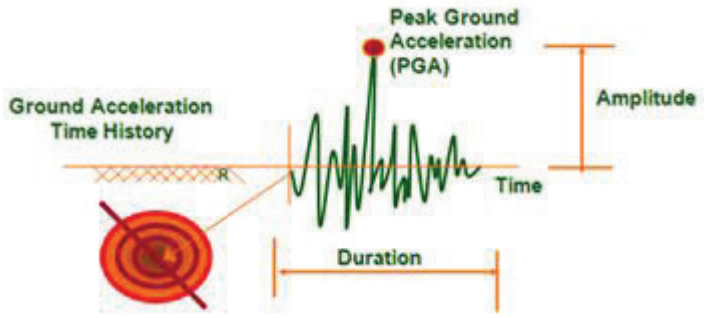


Figura 1.14. Representación del PGA en la señal de vibraciones [15].

Determinar el PGA es un objetivo importante dentro de la ingeniería de terremotos ya que el diseño de movimiento del suelo en un terremoto se define a menúdo en términos de PGA. Comunmente se utiliza también la escala de intensidad de Mercalli, que cuantifica según los daños que produce un terremoto, utilizando informes personales y observaciones; pero el PGA se mide mediante el acelerómetro y ambos se pueden correlacionar.

En el momento de un terremoto, los daños en los edificios y la infraestructura se relacionan estrechamente con las vibraciones del suelo, determinadas por el PGA, en lugar de la magnitud misma del sismo. Para el caso de los terremotos moderados, el PGA es un determinante razonable del daño y en los terremotos severos, es cuantificado más a menudo con la velocidad máxima del suelo.

1.3.5 Aplicaciones disponibles en la plataforma de Android para la medición de vibraciones

El desarrollo constante de la tecnología ha proporcionado un gran avance en el campo de la telefonía y de las tecnologías que integran los *Smartphones*. Android es una plataforma que permite tener un libre acceso al desarrollo de aplicaciones que permiten aprovechar los componentes que se encuentran internamente en un *Smartphone.* La mayoría de los componentes internos de estos dispositivos inteligentes son sensores que brindan información del entorno del dispositivo.

El desarrollo de aplicaciones que brindan diferentes usos para estos sensores, han permitido tener diversas aplicaciones en el mercado de aplicaciones Android. Los dispositivos con el sistema operativo Android cuenta con una plataforma digital de distribución de aplicaciones móviles, conocida como Play Store o Google Play, dónde se puede encontrar una gran variedad de aplicaciones disponibles para descargar y está operada por Google.

Aplicación “Vibrations”

Esta es una aplicación realizada para el análisis de vibraciones mecánicas y el ruido acústico de los motores, las máquinas, el medio ambiente y otros equipos. Vibrations es la aplicación que mayor similaridad tiene con la aplicación que se desarrolla en este proyecto ya que permite tener un análisis de las vibraciones en el dominio de la frecuencia y poder visualizar su espectro. Utiliza los datos recibidos desde el acelerómetro y el micrófono del *Smartphone* debido a que realiza el análisis espectral de las medidas mecánicas y acústicas de forma simultánea. Permite también grabar durante un periodo de tiempo los datos receptados en los tres ejes para poder extraerlos en un archivo de texto plano. Los ficheros son guardados con una extensión *.dat*, el archivo de texto contiene los la información de los datos receptados, dispuesta en columnas, de los tres ejes, el tiempo en el que se tomaron cada uno de ellos.

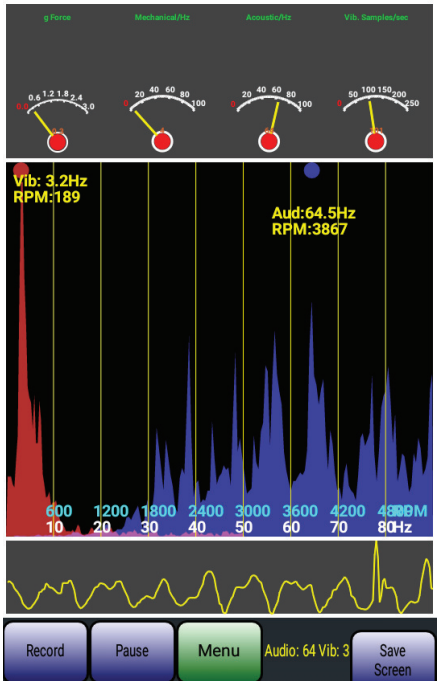


Figura 1.15 Interfaz de la aplicación Vibrations .

Se puede observar las gráficas en el dominio del tiempo y de la frecuencia como se observa en la Figura 1.15. Permite ver la aceleración que se encuentra sobre el dispositivo medida en gravedades (g). En el área del análisis espectral de las vibraciones presenta dos gráficas diferenciando las vibraciones que son de origen mecánico (espectro de color rojo) y las de origen acústico (espectro de color azul). Los valores de frecuencia predominante de cada uno de los espectros se muestra en medida de hertzios (Hz) y revoluciones por minuto (RPM).

Algo que se puede destacar, es que presenta como diferencia, un botón para pausar la ejecución de la aplicación y para realizar una captura de la pantalla; sin embargo, la aplicación solo permite grabar los datos durante un periodo de 20 segundos y en el espectro que se muestra la gráfica solo indica el valor de frecuencia predominante de entre los tres ejes. Como semejanza, esta aplicación también crea una carpeta en la memoria interna del dispositivo, en la cual almacena los archivos de texto plano y las imágenes generadas por la aplicación.

Aplicación "Vibrómetro Detector Terremoto"

Esta aplicación está diseñada para permitir la detección y registro de las ondas sísmicas generadas por terremotos, golpes y cualquier tipo de eventos que sean fuentes de vibraciones. Al igual que todo este tipo de aplicaciones, utiliza los datos receptados desde el acelerómetro para realizar la medición. Presenta una gráfica que contiene las aceleraciones de cada uno de los tres ejes sobrepuestas pero diferenciadas con un color distinto cada eje. Se puede activar o desactivar la gráfica de un eje o de otro indistintamente para solo visualizar el que se desee. Los valores de aceleración son en tiempo real en unidades de m/s^2.

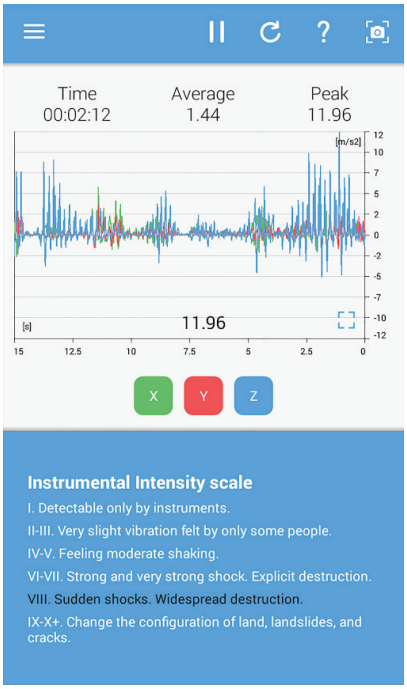


Figura 1.16 Interfaz de la aplicación Vibrómetro.

Además, como se puede observar en la Figura 1.16, en la parte superior indica el tiempo transcurrido durante la medición, el valor medio y el valor máximo de la aceleración. Al ser

una aplicación para detección y registro de las vibraciones de un terremoto, utiliza como escala sísmica la escala de intensidad Mercalli para cuantificar la magnitud de energía liberada por un terremoto y los efectos que éste pudo producir.

Aplicación “Sismómetro: Vibration meter”

En su descripción, dentro de la plataforma de distribución Google Play Store, indica como objetivo principal medir una vibración o un terremoto a manera de un detector sísmico. Los valores son detectados a través del sensor de aceleración del dispositivo y los relaciona con la escala de Intensidad de Mercalli Modificada (MMI). Esta escala se basa en 12 niveles descritos por números romanos para evaluar la intensidad de terremotos con relación al daño o efectos en estructuras.

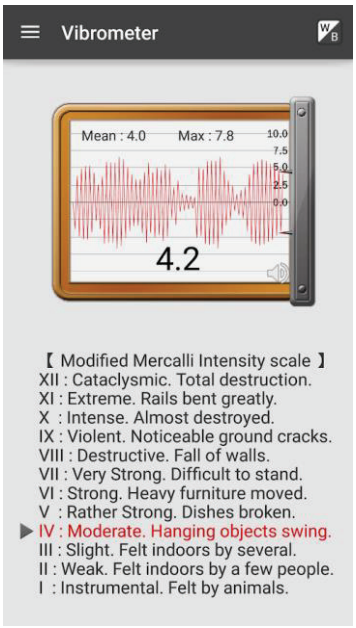


Figura 1.17 Interfaz de la aplicación Sismómetro: Vibration meter.

Esta aplicación no realiza un análisis en el dominio de la frecuencia para poder establecer la frecuencia de vibración, únicamente realiza un relación con el grado de intensidad de las vibraciones con la escala de Mercalli.

Aplicación “Análisis de Vibraciones”

Esta aplicación, al igual que las anteriores mecionadas, presenta características como el análisis en el dominio del tiempo mostrando la aceleración en tiempo real, análisis en el dominio de la frecuencia mostrando un gráfico de amplitud vs. frecuencia. Al arrancar la

aplicación se puede seleccionar los parámetros que se desea ver, si se prefiere de forma gráfica o mediante parámetros característicos de las medidas tomadas como son: valores máximo, mínimo, promedio, RMS, la varianza entre los valores anteriores de aceleración y otros.

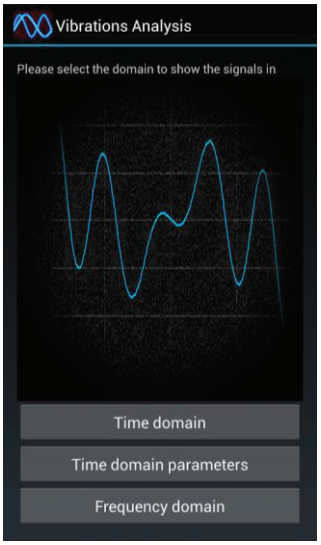


Figura 1.18 Interfaz principal de la aplicación Análisis de vibraciones [16].

Se puede hacer el análisis en los tres ejes por separado. Además, muestra también dentro de los parámetros la asimetría estadística, que se refiere a indicar cómo es la simetría de la distribución de una variable respecto a su media aritmética y por último indica la curtosis, que es un parámetro para medir que tan achatada es la curva representada.

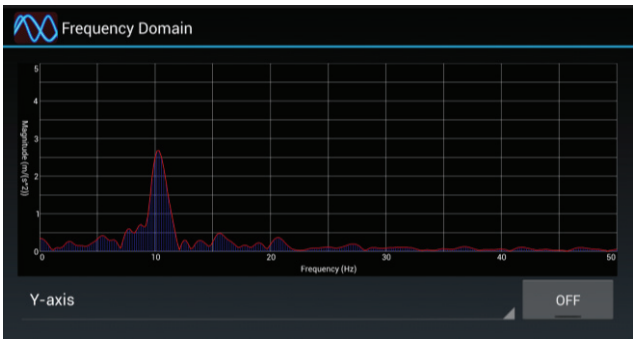


Figura 1.19 Captura del análisis en el dominio de la frecuencia de la aplicación Análisis de vibraciones [16].

Cabe mencionar que esta aplicación a pesar de tener el análisis en el dominio de la frecuencia, esta opción no se encuentra disponible para la versión gratuita. La versión pagada permite obtener los parámetros del análisis del espectro de frecuencias y realizar una captura de la ejecución de la aplicación.

Aplicación “Vibration Analyzer”

Esta aplicación, como sus semejantes, permite realizar el análisis de vibraciones a través del espectro de frecuencias que se obtiene al implementar la FFT en las medidas receptadas desde el acelerómetro del Smartphone. La aplicación muestra el análisis independientemente para el eje seleccionado, al seleccionar el eje se muestra un gráfico en la parte superior de la aceleración con el tiempo y debajo de este el espectro de frecuencias correspondiente.

Posee íconos que permiten realizar un análisis durante cinco segundos y guardar los datos durante este periodo. De igual forma se puede guardar una captura de las gráficas que se están obteniendo, así como pausar la ejecución de la aplicación.

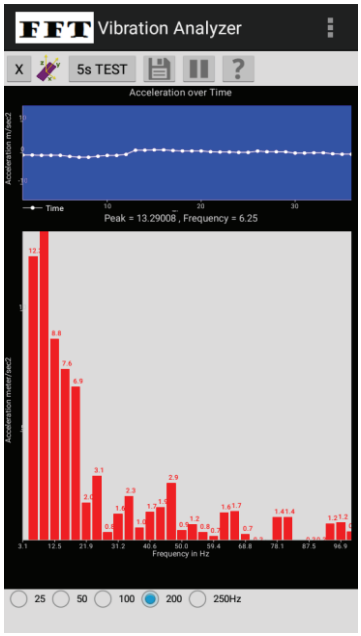


Figura 1.20 Ejecución de la aplicación Vibration Analyzer.

Aplicación “VibSensor”

La aplicación VibSensor transforma el *Smartphone* en un vibrómetro o sismómetro que permite la recolección, procesamiento y almacenamiento de los datos detectados desde el acelerómetro del dispositivo. Dentro de la interfaz principal que se despliega al abrir la aplicación, se visualiza información acerca de la vibración e inclinación del teléfono. Para el procesamiento de los datos se debe acceder a la opción de *Acquire*, donde se ingresa la información para grabar el archivo, permite ingresar el nombre del archivo, tiempo de retardo para iniciar la grabación y el tiempo de duración.

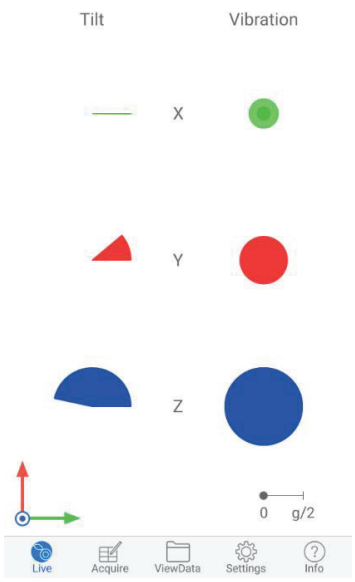


Figura 1.21 Interfaz principal de la aplicación VibSensor.

Una vez que se ha ingresado la información para el almacenamiento de los datos en un archivo, se inicia el proceso y posteriormente se puede visualizar los resultados en la opción *ViewData* donde se despliega una pantalla como se puede ver en la Figura 1.22. En la pantalla de resultados se indica la fecha, hora, duración, frecuencia de muestreo, los valores pico en cada uno de los ejes y su respectiva frecuencia. Ésta, se presenta como una aplicación didáctica ya que tiene una gran facilidad de uso y muy práctica. Presenta gráficas para elegir en unidades lineales o logaritmicas, así como se puede elegir que ejes se desea visualizar.

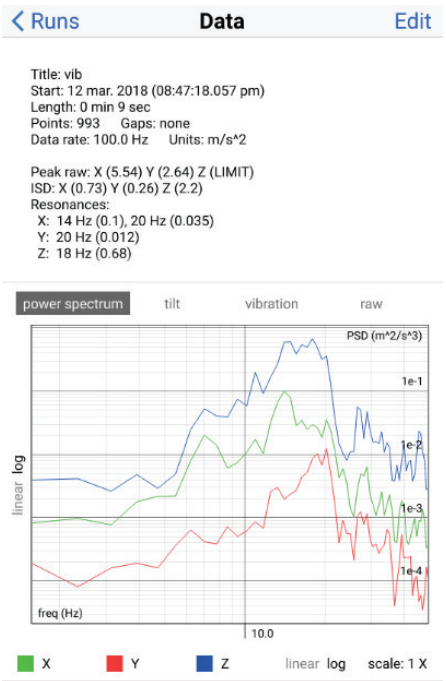


Figura 1.22 Resultados de la ejecución de la aplicación VibSensor.

2. METODOLOGÍA

Este capítulo tratará el desarrollo de la aplicación Android a nivel de software usando una investigación del tipo exploratorio y descriptivo, ya que se utilizarán diversas fuentes para conocer e implementar las herramientas y conceptos necesarios en el software de desarrollo para aplicaciones compatibles con el sistema operativo Android. Para conseguir el desarrollo de la aplicación se utilizará como herramienta principal el software Android Studio 3.0 que brinda un entorno de programación con grandes características para cumplir con el alcance del proyecto. Se consultarán los medios necesarios que permitan acceder a los datos del acelerómetro del *Smartphone,* leerlos y procesarlos.

En el desarrollo de la aplicación Android, primero se recibirán los datos del acelerómetro en sus tres ejes (X, Y y Z), los cuales se imprimirán y graficarán dentro de una misma actividad. En una segunda actividad de Android Studio se recibirán estos mismos valores de cada eje y se calculará el parámetro del PGA (*Peak Ground Acceleration*) que se define como el mayor valor absoluto de aceleración respectivamente con el instante de tiempo que se genera en cada uno de los ejes. Estos datos se guardarán dentro de un archivo de texto plano en la memoria del dispositivo inteligente para poder ser extraídos posteriormente y ser tratados de acuerdo a la necesidad del usuario. Como etapa final se obtendrá el procesamiento espectral de los datos grabados mediante dos clases implementadas FFT.java y Complex.java que permiten realizar la Transformada Rápida de Fourier de datos complejos que se manejarán con la correspondiente clase y finalmente obtener la frecuencia fundamental de la señal de vibraciones obtenida en un principio. Se imprimirá el valor de amplitud máximo que se obtiene del espectro de frecuencias, la frecuencia a la que se produce y las gráficas del espectro en cada uno de los tres ejes.

2.1 Herramientas para el desarrollo de la aplicación

2.1.1 Transformada Rápida de Fourier

La FFT (*Fast Fourier Transform*) se considera una herramienta muy versátil para la implementación de algoritmos y aplicaciones de procesamiento de señales digitales.

Para poder realizar el procesamiento de los datos, estos primeramente pasan al dominio de la frecuencia, en este caso el proyecto trata datos discretos que son el tipo de datos que se dispone de la aceleración. Este proceso se lo realiza con ayuda de la Transformada de Fourier, toma la señal en el dominio del tiempo y la transfiere a una serie de señales sinusoidales y finalmente al dominio de la frecuencia, Figura 2.1; así, se puede conocer la

amplitud de la señal en cada uno de los valores de las frecuencias que se encuentran contenidas en la señal.

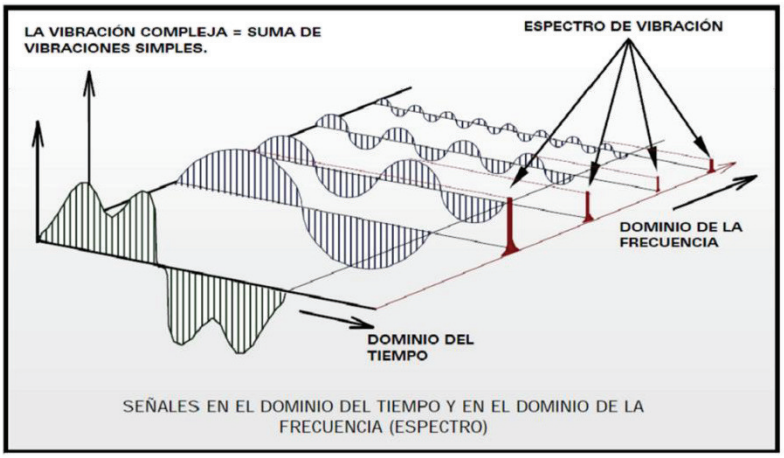


Figura 2.1 Representación de una señal en el dominio del tiempo y su paso al dominio de la frecuencia [16].

La señal de las vibraciones contiene toda la información acerca del comportamiento de la causa que provoca esta aceleración, la Transformada de Fourier permite descomponer la señal del dominio del tiempo en una serie de sinusoides que permitan ver de una mejor manera esta información. El espectro de vibración muestra las señales sinusoidales individualmente, es la gráfica de la amplitud versus la frecuencia y es la principal herramienta para el análisis de vibraciones [17].

2.1.2 Herramienta de desarrollo de la aplicación: Android Studio

Esta herramienta basada en IntelliJ IDEA, brinda un entorno de desarrollo integrado para la plataforma Android, que es uno de los sistemas operativos que predominan dentro del mercado de la telefonía celular. Apareció como el reemplazo del anterior software Eclipse, utilizado anteriormente para desarrollo de aplicaciones compatibles con este sistema operativo. Tiene una licencia Apache 2.0 y se encuentra disponible para Microsoft Windows, macOS y GNU/Linux [18].

Android Studio brinda un fácil alcance de potenciales herramientas para el desarrollo de aplicaciones que permiten personalizar o mejorar los equipos Android. Brinda una rápida detección de errores dentro del código para porceder a su pronta resolución. Al ser apoyado en IntelliJ IDEA es capaz de soportar varios lenguajes basados en JVM (*Java Virtual Machine*), como: Java, Groovy, Clojure, Scala y Kotin [19].

Dentro de la estructura de una aplicación Android se definen cuatro tipos de componentes desde los cuales el sistemas puede ejecutar la aplicación con diferentes propósitos:

- **Actividades (*activities*).-** Estas son las diferentes ventanas o pantallas que se abrirán en la aplicación y presentan una interfaz del usuario, esto dependerá de todas las funciones que vaya a cumplir la aplicación.
- **Servicios (*services*).-** Los servicios son procesos que se ejecutan dentro de un segundo plano y no poseen una interfaz de usuario; es decir, son acciones que se ejecutan detrás de una interfaz gráfica correspondiente a otra aplicación o ventana.
- **Receptores de difusión (*broadcast receivers*).-** Estos actúan cuando existen anuncios que son enviados a todo el sistema, como por ejemplo cuando la pantalla del móvil se apaga y se debe pausar los procesos de la aplicación que se encuentra ejecutándose.
- **Proveedores de contenido (*content providers*):** Estos cumplen la función de gestión de la información que se comparte con otras aplicaciones.

La gestión de los procesos que se inician con cada uno de los componentes de una aplicación se realiza mediante la ejecución de hilos que se van añadiendo al hilo principal. Es decir, si ninguno de los componentes se encuentran en ejecución, el sistema crea un nuevo proceso con un único hilo de ejecución y por defecto el resto de componentes que se empiezan a ejecutar, están contenidos dentro de este y así conforme se van iniciando se van añadiendo al proceso.

Los procesos se mantienen en memoria el mayor tiempo posible, pero cuando el sistema requiere una mayor capacidad de la memoria para la ejecución de procesos con un grado mayor de prioridad, comienza a detener los procesos. La prioridad es asignada a cada proceso por el sistema. Dado el caso de que existan procesos con un igual valor de prioridad, el sistema opta por detener el proceso con menor periodo de uso, de esta forma se puede definir las diferentes prioridades como:

- **Primer plano.-** Son los procesos que se encuentran dentro de la actividad con la que interactua el usuario o bien los servicios que se requieran ejecutar para que la actividad de interacción con el usuario cumpla su proceso.
- **Visible.**- Esto se refiere a los procesos generados por una actividad que es visible para el usuario pero que no se encuentran ejecutando dentro de un primer plano. Esto

puede darse cuando se empieza a ejecutar la nueva actividad pero aun se puede ver la actividad anterior u ofrezca un servicio que sea dependiente de otro proceso visible.

- **Servicio.**- Son procesos que tienen una relación directa con el usuario, pero no se encuentra en un plano visible con el que interactua directamente.
- **Segundo plano.**- Se define así a los procesos secundarios que no tiene ninguna relación directa con el usuario y por esto son mayormente detenidos o pausados.
- **Vacío.**- No realizan ninguna tarea de ejecución de procesos ya que no contienen ningún componente activo. Están a la espera de futuros componentes que necesiten ser agilizados y ejecutados dentro de este proceso [19].

2.2 Estructura general de la aplicación Android

El proyecto tiene como objetivo desarrollar una aplicación que sea compatible con el sistema operativo Android que permita realizar un análisis de una señal de las vibraciones que afectan a una estructura o sistema. Los datos de las vibraciones serán obtenidos a través del sensor de aceleraciones que viene integrado en el interior de un *Smartphone*, para posteriormente ser procesados y llegar a la obtención de las frecuencias predominantes de los tres ejes de orientación. En la Figura 2.2 se muestra un diagrama con las etapas que debe cumplir la aplicación para llegar a la obtención de los resultados finales, con estos resultados se podrá conocer la frecuencia a la cual se produce la mayor intensidad de vibración de la medición que se realice.

Figura 2.2 Diagrama de la estructura general de la aplicación Android.

De acuerdo a los valores obtenidos los profesionales para los que será de utilidad esta herramienta de fácil acceso y manejo, podrán realizar un análisis in situ de la situación, extraer conclusiones de acuerdo al fenómeno que se suscite y brindar soluciones de así requerirse.

2.2.1 Estructura de las actividades realizadas

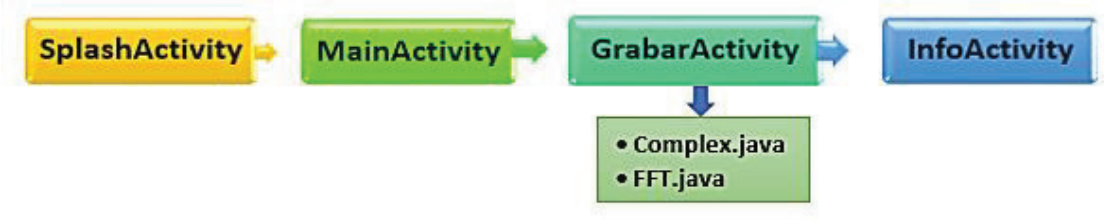


Figura 2.3. Estructura organizacional de las Actividades de la aplicación Android.

- **SplashActivity.-** Esta actividad tiene la característica launcher definido en el archivo AndroidManifest.xml de la aplicación, que significa que es la primera actividad en ser lanzada cuando se ejecuta la aplicación, contiene el mensaje de bienvenida y el logo de la aplicación.

- **MainActivity.-** Esta es la actividad principal, ya que en ella se efectúa el proceso inicial que permite llegar al objetivo final de la aplicación. En ella se realiza la lectura de los datos del acelerómetro de cada uno de sus ejes.

- **GrabarActivity.-** Dentro de esta actividad se realiza el procesamiento y almacenamiento de los datos. Se crea un archivo de texto plano que guarda los datos procedentes de los tres ejes durante el periodo de tiempo ingresado por el usuario. De los datos grabados se obtiene la FFT (*Fast Fourier Transform*) y se extrae como resultados los valores de amplitud y frecuencia más notables. Esta actividad hace uso de dos clases también incluidas: la clase FFT.java y Complex.java, ambas permiten el procesamiendo de los datos para pasar del dominio del tiempo al dominio de la frecuencia.

- **InfoActivity.-** Esta aplicación contiene la información correspondiente a datos básicos de la aplicación como la orientación de los ejes de coordenadas del celular, los colores que representan a cada uno en las gráficas correspondientes y datos del desarrollador de la aplicación.

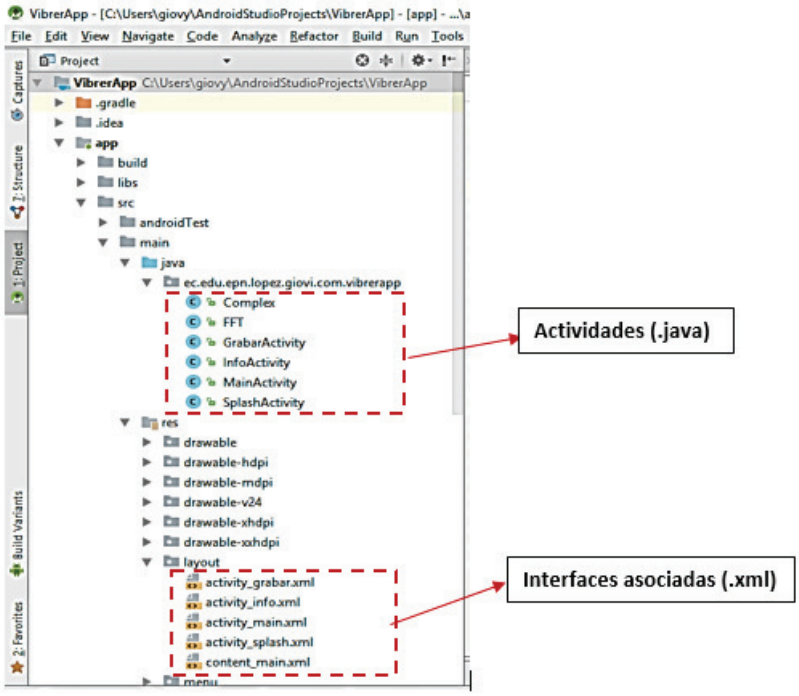


Figura 2.4. Estructura de la aplicación dentro de Android Studio. Actividades (.java) y sus layout (.xml) asociados.

En la Figura 2.4 se presenta como está creada la estructura de la aplicación dentro del software de desarrollo *Android Studio*. Una aplicación Android está conformada principalmente por sus actividades, que contienen la parte lógica o programática de la aplicación y la parte gráfica que define el diseño del interfaz de usuario.

La parte lógica es una clase *Java* que contiene las funcionalidades o acciones que cumple cada objeto añadido en el interfaz de usuario. La parte gráfica está definida dentro de un archivo xml, el cual contiene todos los componentes gráficos que forman la interfaz gráfica del usuario, esta sección se puede realizar tanto a nivel de código ingresando los objetos y sus acciones textualmente, o se tiene de igual forma la posibilidad de desarrollar de forma gráfica colocando los objetos de una forma más amigable y fácil [20].

2.3 Actividad de presentación de la aplicación (*SplashActivity.java*)

Esta actividad está dedicada para la parte inicial de la aplicación, se define como una actividad que es lanzada durante un cierto periodo de tiempo y es la carta de presentación hacia la aplicación.

Contiene el logo de la aplicación y el mensaje de bienvenida que indica el objetivo principal de la aplicación que está dedicada para el análisis de vibraciones, en la Figura 2.5 se puede ver la presentación de esta actividad al momento de ejecutarse la aplicación.

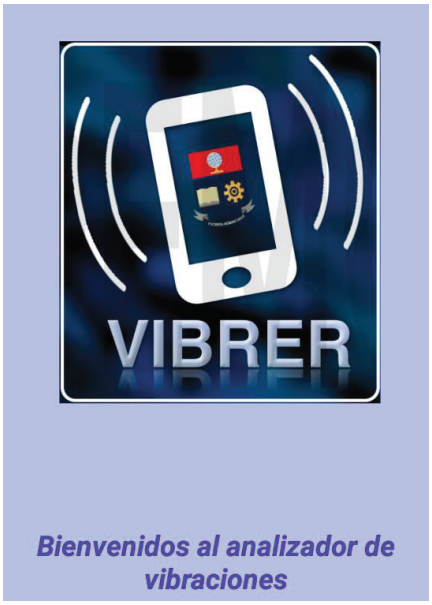


Figura 2.5. Actividad lanzada al iniciarse la ejecución de la aplicación Android.

Como cualquier otra actividad contiene su parte gráfica y lógica, dentro del archivo *SplashActivity.java* se definen parámetros como la orientación y forma de cómo se quiere visualizar la pantalla, la duración que tendrá esta actividad durante su ejecución, ya que inmediatamente una vez transcurrido este periodo de tiempo se lanzará la actividad principal. No tiene una interacción con el usuario ya que solo se espera a que trasncurra el lapso de ejecución.

Todos los recursos gráficos que se requieren utilizar en la aplicación, como imágenes de presentación, fondos, íconos, entre otros, se ubican dentro del directorio app/src/main/res/drawable. Este directorio contiene imágenes con distintos índices de resolución de acuerdo al uso que se le dé a cada recurso gráfico.

2.3.1 Ícono de la aplicación.

El ícono de la aplicación es la imagen que representará a la aplicación dentro del menú de aplicaciones y por el cual se identificará a la misma. *Android Studio* tiene ya definida la imagen tradicional del Androide, Figura 2.6, como ícono por defecto de todas las aplicaciones que se creen; sin embargo, este puede ser cambiado según se requiera por la imagen que se desee.

Para crear el ícono y cambiarlo, se debe tomar en cuenta la gran diversidad de pantallas y de la resolución que existen en la actualidad. Es por esto, que se debe realizar una imagen con distinta dimensión de píxeles, para que de esta forma el ícono se pueda acoplar de acuerdo a la resolución del dispositivo en el cual se instale la aplicación.

Figura 2.6. a) El Androide, ícono por defecto de Android Studio. **b)** Ícono de la aplicación para sustituir al anterior.

Una vez diseñado y creado el ícono, es necesario adaptarlo para cada una de las densidades de pantalla en la que puede funcionar la aplicación, los distintos tamaños se detallan en la Tabla 2.1. Se debe sustituir el nuevo ícono con el nombre de **ic_launcher.png** dentro de cada una de las carpetas con los diferentes tamaños, teniendo en cuenta que corresponda a la correcta densidad de cada una de las carpetas y que sea una imagen con la extensión .png. Estas carpetas se encuentran en el directorio de recursos de la aplicación: app/src/main/res.

Tabla 2.1 Tamaños de resolución del ícono de una aplicación.

Tipos de tamaños	Dimensiones (píxeles)	Aplicaciones
Mipmap-mdpi	48x48	Pantallas de media resolución
Mipmap-hdpi	72x72	Pantallas de alta densidad con alta resolución
Mipmap-xhdpi	96x96	Pantallas de muy alta densidad de resolución
Mipmap-xxhdpi	144x144	Pantallas de extra alta densidad (ej: tablets)
Mipmap-xxxhdpi	192x192	Pantallas de ultra alta densidad (ej: televisores)

2.4 Actividad principal (*MainActivity.java*)

Dentro de esta actividad se implementan las funciones principales para iniciar con el desarrollo de la aplicación y las cuales llevarán al cumplimiento del objetivo funcional de la

misma. Las etapas que se deberán cumplir para obtener los resultados de esta aplicación se observan en el organigrama de la Figura 2.7.

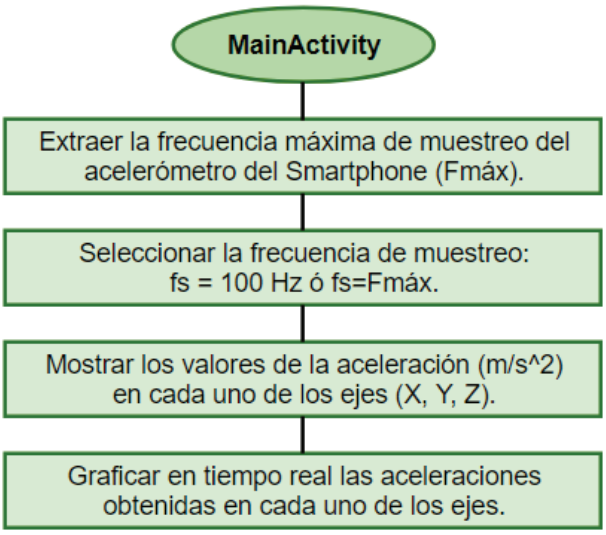


Figura 2.7. Diagrama de las etapas a cumplir en la Actividad Principal de la aplicación.

2.4.1 Extraer la frecuencia máxima de muestreo del acelerómetro del *Smarphone* (Fmáx)

Uno de los principales parámetros que se debe tomar en cuenta para el correcto funcionamiento de la aplicación es conocer la frecuencia máxima a la que el sensor acelerómetro puede trabajar. Se pueden conocer distintos detalles acerca de las capacidades físicas que poseen los sensores instalados en el dispositivo en el que se encuentre la aplicación, datos como: tipo de sensor, resolución, periodo de retardo mínimo y la potencia. Esto se hace invocando el método SensorManager getSensorList() del *framework* de Android para sensores, ya que proporciona diversas clases e interfaces que permiten acceder a los sensores del dispositivo, esto se detalla en la Tabla 2.2.

Los datos que se toman del acelerómetro están dispuestos en los tres ejes del dispositivo de acuerdo a la orientación indicada en el marco teórico del proyecto. Al trabajar con sensores se debe manejar los eventos que ocurren en el sensor. Se define como el evento de un sensor, al cambio que se detecta en alguno de los parámetros que se van a medir.

Tabla 2.2 Framework para sensores compatibles con la plataforma Android.

Nombre	Tipo	Descripción
SensorManager	Clase	Crea una instancia del servicio de sensores. Ofrece distintos métodos para el acceso a los sensores, el registro y eliminación de registros de la escuchas de los eventos de sensores, entre otros.
Sensor	Clase	Crea una instancia de un sensor específico.
SensorEvent	Clase	Permite publicar los datos del sensor, tipo de sensor, precisión de los datos y la marca de tiempo.
SensorEventListener	Interfaz	Ofrece métodos de llamada de regreso para recibir avisos del SensorManager cuando los datos o la precisión del sensor han sufrido un cambio.

Para las aplicaciones a las que están destinadas el uso de la aplicación Android a desarrollar, se necesita una frecuencia de muestreo adecuada de 100 Hz como mínimo, ya que dentro de esta frecuencia se pueden observar los efectos que sufren las máquinas, estructuras o ambientes en donde se requiere el análisis de vibraciones. Es por esto que se requiere conocer como parámetro la frecuencia máxima que trabaja el acelerómetro del dispositivo.

Para realizar este proceso se implementa la interfaz SensorEventListener, la cual llamará a la escucha de los cambios que se den en los valores del sensor, se define las variables de objeto de clase para manejar el sensor:

```
SensorManager sensorManager;  //Gestor de sensores Sensor Manager
private Sensor acelerometro;  // Declaración del sensor a manejar
```

Se debe conectar el gestor de sensores con el servicio para obtener la instancia de la clase, se llama a la función de gestión del servicio del sistema con el argumento SENSOR_SERVICE, de esta forma se obtiene un servicio a nivel de sistema y así ya se podrá acceder a los sensores del sistema del dispositivo, una vez que ya se tiene el acceso se indica el tipo de sensor que se va a ocupar, en este caso el acelerómetro Sensor.TYPE_ACCELEROMETER:

```
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
acelerometro = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

El parámetro que ayuda a conocer la frecuencia de muestreo se lo conoce como getMinDelay, se define como el valor de mínimo retardo entre cada evento del sensor, devuelve el valor en microsegundos (µs).

```
MinDelay = acelerometro.getMinDelay();
```

Al valor que se devuelve de este se lo debe tratar como el periodo que transcurre entre cada evento y se puede obtener la frecuencia equivalente de acuerdo a la siguiente ecuación:

$$F = \frac{1}{T}$$

Ecuación 2.1 Ecuación para obtener la frecuencia en función del periodo de tiempo [22].

Donde, T es el periodo entre cada evento y F la frecuencia a calcular. Se debe tomar en cuenta la transformación de las unidades de medida con las que se trabaja ya que la frecuencia se debe obtener en Hertz [Hz].

Como resultado se mostrará en la pantalla el valor obtenido de la frecuencia de muestreo máxima (Fmáx) que tiene la capacidad el acelerómetro del dispositivo para que el usuario tenga conocimiento de la frecuencia a la que se puede trabajar y si cumple la frecuencia mínima de 100 Hz a la que se debe ejecutar la aplicación.

Seleccionar la frecuencia de muestreo

A pesar que de acuerdo a la investigación realizada de las aplicaciones en las cuales se realiza el análisis de vibraciones se observó que se tiene como parámetro común el uso de la frecuencia de 100 Hz, se tiene también como objetivo usar el potencial máximo del hardware del *Smarphone* y en el caso de que este permita una frecuencia mayor a la mínima requerida, se dará la posibilidad de que el usuario elija la frecuencia a la que desea trabajar.

Anteriormente ya se informa al usuario la capacidad de frecuencia máxima que posee el sensor del dispositivo, entonces la selección se realiza mediante la utilización de un RadioGroup integrado con dos RadioGroup que indican la opción de 100 Hz y la frecuencia máxima disponible.

Para activar la escucha del RadioGroup se implementa la interfaz RadioGroup.OnCheckedChangeListener, que es invocada cuando la opción marcada cambia en los RadioButton. Como todo objeto que forma parte de la interfaz gráfica de la aplicación se declara y se enlaza con las variables que van a representar al objeto dentro de la aplicación.

```
RadioGroup Radiogrupo1;
RadioButton optfrecuencia1, optfrecuencia2;
```

Dentro del método de devolución de llamada de los eventos del sensor onResume(), se relaciona el objeto con la variable y se llama al método onCheckedChange (RadioGroup radioGroup, int CheckedId), que registra la opción seleccionada dentro del Radiogroup.

```
RadioGroup Radiogrupo1 = (RadioGroup) findViewById(R.id.Radiogrupo1);
Radiogrupo1.setOnCheckedChangeListener(this);// Se asigna el evento
onclick o touch de los botones
```

La opción seleccionada se la maneja con el establecimiento de un condicional (switch) y en cada caso se define la acción que tomará de acuerdo a la opción que se elije. En esta función se incluye la velocidad a la que trabajará el sensor.

Opción 1:

```
sensorManager.registerListener(this, acelerometro, 10000 );
```

Opción 2:

```
sensorManager.registerListener(this, acelerometro,
SensorManager.SENSOR_DELAY_FASTEST);
```

Mediante esta instrucción se invoca el método registerListener() y se define el retardo que se desea tener entre los eventos que se envían a la aplicación Android a través del onSensorChanged() que es el método de devolución de llamada. Existen cuatro principales retardos que se puede utilizar para ser definidos en la aplicación, esto se detalla en la Tabla 2.3 conjuntamente con los retardos de cada opción.

Tabla 2.3 Retraso de datos o tasa de muestreo predeterminados

Nombre	Retardo (µs)	Descripción
SENSOR_DELAY_NORMAL	200000	Este viene predeterminado generalmente para monitorear cambios típicos de orientación de pantalla.
SENSOR_DELAY_GAME	20000	Tasa de muestreo adecuada para juegos.
SENSOR_DELAY_UI	60000	Tasa de muestreo para la interfaz de usuario.
SENSOR_DELAY_FASTEST	0	Esta tasa permite obtener los datos del sensor lo más rápido posible.

La aplicación tiene como ser diseñada para poder ejecutarse en distintos dispositivos con un sistema operativo Android, pero al existir una gran gama de variedades y marcas, cada uno tendrá características diferentes. Esto, quiere decir que no tendrán una frecuencia de muestreo definida para todos ellos y por lo tanto no se puede especificar un valor exacto de frecuencia de muestreo.

Para la aplicación desarrollada se toma en cuenta el mínimo retardo que se puede tener para poder obtener una frecuencia de muestreo máxima. Tomando en cuenta que desde Android 3.0, una API nivel 11, se tiene la posibilidad de ingresar un retraso específico como un valor absoluto pero en unidades de microsegundos, por esto para este caso se ingresa un retardo de 10000 μs que representa a una frecuencia de muestreo de 100 Hz [23].

Se debe tomar en cuenta que al pasar la señal del dominio del tiempo al de la frecuencia se puede producir el efecto de aliasing o solapamiento del espectro, esto puede ocurrir cuando no se cumple el Teorema de Nyquist-Shannon. Este teorema indica que la frecuencia de muestreo debe ser igual o mayor al doble de la frecuencia máxima de la señal.

En el caso de que no se cumpliera lo mencionado, las réplicas que se producirían al aplicar la FFT se sobrelaparían, haciendo que al momento de tratar la señal no presente la exactitud deseada. La aplicación está desarrollada para que el análisis sea realizado en base a la frecuencia de 100 Hz o a la frecuencia de muestreo máxima del celular y dependiendo de cual se elija se determina la frecuencia máxima que se mostrará en el espectro.

La solución para evitar el aliasing es mediante la implementación de filtros para mejorar la señal antes de procesarla. Sin embargo, en base a la investigación realizada se conoce que los acelerómetros que vienen integrados en los Smartphones, ya poseen un filtro interno que realiza este trabajo, estableciendo su frecuencia de corte en base a la frecuencia máxima de muestreo que tengan en su diseño. Por lo que, realizar otro filtro antes de pasar a implementar la FFT, no presentaría una mayor ventaja.

2.4.2 Mostrar los valores de la aceleración en cada uno de los ejes triaxiales

Una vez que se tiene definida la frecuencia con la que se va a recibir los datos desde el acelerómetro se debe capturar estos eventos que suceden en cada uno de los tres ejes de coordenadas. La captura de los eventos como ya se mencionó se realiza mediante la escucha del SensorEventListener y a través del método onSensorChanged(). Este método es llamado cada vez que el sensor sufre un cambio en su valor u orientación capturando el motivo o valor que provoca el cambio en el sensor.

Inicialmente se definen las variables sobre las cuales se captarán los eventos, se define como un valor del tipo flotante ya que contiene valores decimales que pueden cambiar instantáneamente al mínimo movimiento del sensor.

```
float ejeX, ejeY, ejeZ;
```

Dentro del onSensorChange se captura el evento de cada uno de los ejes mediante los valores asignados a cada uno, eje X (values [0]), eje Y (values [1]) y finalmente eje Z (values [2]), de acuerdo a estos valores se puede referir a cada uno de los ejes de orientación.

```
ejeX = sensorEvent.values[0];
ejeY = sensorEvent.values[1];
ejeZ = sensorEvent.values[2];
```

La impresión de los datos se realiza en un campo de edición de texto:

```
texto_ejes.setText("");
texto_ejes.append("El valor de X:  " + ejeX + "\n" + "El valor de Y:  " +
ejeY + "\n" + "El valor de Z:  " + ejeZ + "\n " );
```

Se debe tomar en cuenta resetear el campo de edición de texto ya que los valores del acelerómetro cambian constantemente y si no se realiza este paso, la pantalla del interfaz gráfica se llenará de valores infinitos hasta que la aplicación se detenga.

Cabe recalcar que los valores que se obtiene en cada uno de los ejes, representa a la aceleración que se produce en cada uno de ellos. Y se encuentran en m/s^2 como unidad de medida. Esto depende del cambio de desplazamiento que se provoque al dispositivo y en el sentido en el que se lo realice.

Un dato muy importante a tener en cuenta del correcto funcionamiento de la captura de datos, es observar que la constante de la gravedad (9,8 m/s^2) sea receptada en uno de los tres ejes de acuerdo a la posición que encuentre el dispositivo. Como por ejemplo como se muestra en la Figura 2.8, se puede ver que la gravedad afecta netamente al eje Z ya que le dispositivo se encuentra dispuesto paralelamente a la superficie y con el eje Z perpendicularmente. Por esto la gravedad afecta directamente en esta orientación.

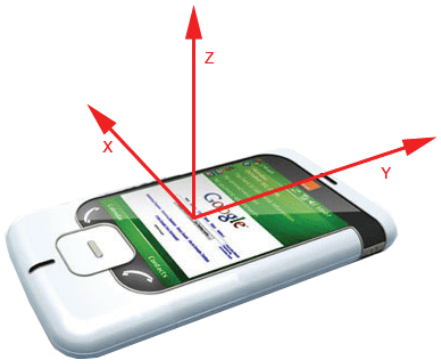


Figura 2.8. Ubicación de los ejes de orientación de un celular dispuesto cara arriba sobre la superficie [24].

En ocasiones, cuando el dispositivo no se encuentre en una alineación perpendicular con respecto a algún eje, esta componente de la gravedad se distribuirá en cada una de las componentes de los ejes que se vean influenciados por la gravedad.

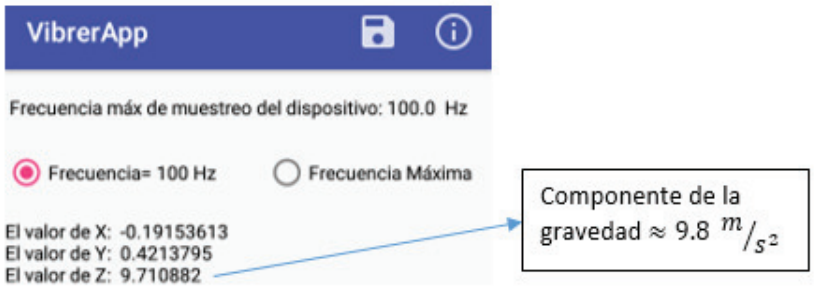


Figura 2.9. Captura de los resultados obtenidos en la aplicación con la orientación indicada en la Figura 2.8.

2.4.3 Graficar en tiempo real las aceleraciones obtenidas en cada uno de los ejes

Añadir la biblioteca GraphView al directorio de la aplicación

Para realizar la gráfica de los valores de aceleración que se captan en cada uno de los tres ejes de coordenadas se implementará una librería disponible para Android y que brinda grandes características para generar gráficas. La librería GraphView es una biblioteca disponible para Android que se puede implementar libremente dentro de cualquier tipo de aplicación.

Se escogió la librería GraphView debido a que presenta grandes características favorables para la realización de gráficas. Entre las características principales de esta librería se tienen:

- Permite crear diversos tipos de gráficas, de líneas, de barras, de puntos e incluso combinar todos los tipos de gráficas.
- Se puede incluir varias series de datos dentro de un mismo gráfico y diferenciarlos a cada grupo de datos con un color y descripción individual.
- Realiza gráficos en tiempo real, añadiendo los datos a las series en vivo y actualizándolos inmediatamente.
- Permite activar las funciones de zoom, arrastre y selección.

- Se puede añadir etiquetas a los ejes, leyenda para identificarlos, los colores, el fondo, entre otras funcionalidades.
- Permite agregar valores a las series de la gráfica desde un archivo XML.
- Tiene una aplicación que se encuentra disponible en Google Play donde se presenta una guía de todo lo que se puede hacer con esta librería.

La documentación e información necesaria para incluir esta librería se puede encontrar en su propia página GraphView (http://www.android-graphview.org//). Dentro de esta pagina se encuentran las características, la sección de descarga, donde indica como añadir la biblioteca y la documentación necesaria para implementar todas las funcionalidades de la librería. Se presenta el código en sí, que debe añadirse en cada una de las secciones del proyecto según lo que se quiera lograr con la librería.

Una vez descargada la librería se tiene un archivo .jar el cual se debe añadir a la carpeta *libs* dentro de la carpeta de la aplicación Android. Desde la plataforma de desarrollo Android Studio se hace clic derecho sobre el directorio /app/libs y se pega la librería GraphView-4.2.1.jar. Android Studio desplegará la ventana donde se indica que se añadirá la librería al directorio de la aplicación.

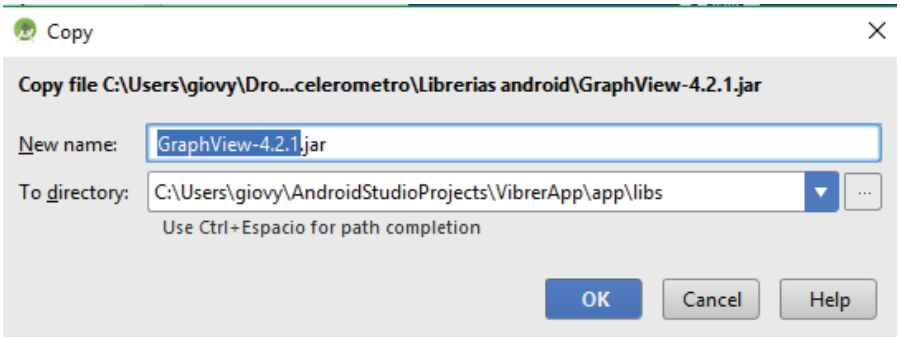


Figura 2.10 Añadir la librería GraphView a la carpeta de librerías de la aplicación.

Pero una vez copiada la librería se debe añadir las dependicias de la misma para poder comenzar a utilizarla, esto se realiza a través de la línea de código: `compile 'com.jjoe64:graphview:4.2.1'` dentro del archivo build.gradle del directorio de la aplicación. Otra forma es haciendo clic derecho sobre el archivo .jar una vez que se ha copiado y elegir la opción de 'añadir librería', esto hará todo el proceso automáticamente.

Unas capturas del proceso a seguir para incluir la librería se encuentran en el Anexo II de este proyecto.

Implementación en la aplicación

Para comenzar con su implementación se debe manejar como cualquier objeto en donde se añade la parte gráfica y se define sus funciones en la parte lógica. Dentro del archivo content_main.xml se añade el campo que introduce el área donde se realizará la gráfica de la señal de vibraciones.

```
<com.jjoe64.graphview.GraphView
    android:id="@+id/grafico"
    android:layout_width="match_parent"
    android:layout_height="298dp"
    android:layout_marginBottom="8dp"
    android:layout_marginTop="32dp" />
```

En la parte gráfica se define el identificador con el que se va a manejar dentro de la parte lógica, las dimensiones del área del gráfico y su ubicación en el interfaz gráfico de la actividad en la que se está trabajando.

Dentro del archivo MainActivity.java se definen las funciones que cumplirá la librería de la gráfica y se añade los datos obtenidos directamente desde el acelerómetro del *Smartphone.* Se declaran primeramente las series en donde se receptarán los datos que se enviarán al gráfico, se crea uno para cada eje respectivamente y se inicializa el eje horizontal del gráfico.

```
private LineGraphSeries<DataPoint> dataseriesX, dataseriesY, dataseriesZ;
private float ejeXgrafico = 0;
```

Se crea la instancia del GraphView, enlazando con la variable y el identificador del objeto. De igual forma se crea las series que contendrá los datos, para dar una idea de la implementación se muestra como ejemplo lo que se realiza para el eje x; sin embargo, esto se debe realizar para cada uno de los ejes.

```
GraphView grafico = (GraphView) findViewById(R.id.grafico);
dataseriesX = new LineGraphSeries<DataPoint>();
grafico.addSeries(dataseriesX);
```

Para realizar el manejo de los datos en cada una de las series, se crea un método llamado addEntry(), en el cual se añadirá todas las entradas de los valores de aceleración a cada uno de los tres ejes. En esta instrucción se relaciona los valores del eje horizontal (ejeXgrafico) que irá incrementándose en medida que surjan los eventos y el eje vertical (ejeX) que contiene los valores de aceleración en m/s^2 del eje x del acelerómetro.

```
dataseriesX.appendData(new DataPoint(ejeXgrafico++ , ejeX ), true ,
1000);
```

El método addEntry() es llamado dentro de onSensorChanged() ya que se necesita que los datos ingresen al gráfico cada vez que se produzca un evento en el sensor. De esta forma se tiene una sincronización en tiempo real del ingreso de los datos y se muestran inmediatamente en la gráfica.

Gracias a las propiedades del Viewport de la librería GraphView se puede realizar diferentes personalizaciones del gráfico. Se puede definir el tamaño y el valor de inicio de los ejes del gráfico, permite activar la función de zoom, desplazamiento, escalabilidad y otros.

```
Viewport viewport = grafico.getViewport();
viewport.setYAxisBoundsManual(true);
viewport.setMinY(-20);
viewport.setMaxY(20);

viewport.setXAxisBoundsManual(true);
viewport.setMinX(0);
viewport.setMaxX(800);

viewport.setScalable(true);
viewport.setScrollable(true);
viewport.setScalableY(false);
```

De igual forma se puede modificar la apariencia o visualización de las series de datos de cada uno de los ejes, GraphView permite asignar un color, título, cuadro de leyenda y etiquetas a los ejes.

```
grafico.getGridLabelRenderer().setVerticalAxisTitle("Aceleración
(m/s^2)");
dataseriesX.setTitle("EJE X");
dataseriesX.setColor(Color.GREEN);
grafico.getLegendRenderer().setVisible(true);
grafico.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.TOP);
```

De esta forma se tiene como resultado la gráfica de la aceleración en unidades de m/s^2 de los tres ejes como se puede ver en la Figura 2.11, en la cual se muestra una captura del interfaz gráfico completo de la MainActivity y se observa cada uno de los temas tratados como: la frecuencia máxima de muestreo, la selección de la frecuencia con la que se desea trabajar, muestra los valores de aceleración de los tres ejes y su respectiva gráfica en tiempo real.

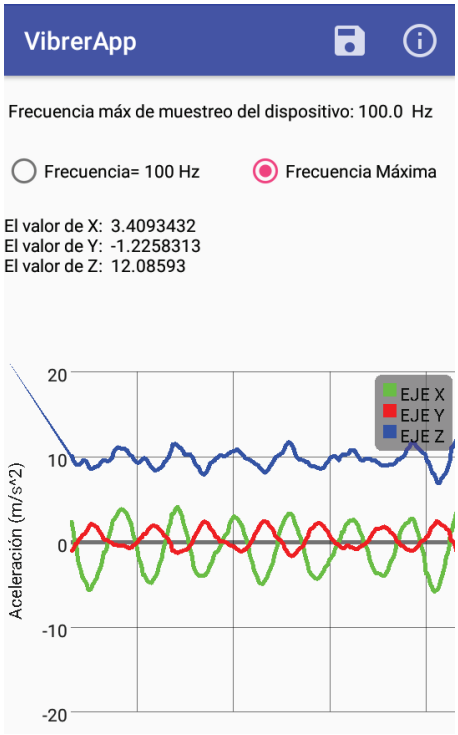


Figura 2.11. Actividad principal de la aplicación para el análisis de vibraciones.

2.5 Actividad de almacenamiento y procesamiento de datos (*GrabarActivity.java*)

Esta actividad abarca como objetivo el almacenamiento de los datos dentro de un archivo de texto plano, el cual se guardará en la memoria del dispositivo para posteriormente ser extraído y tener la posibilidad de tratar los datos externamente. El periodo que durará la grabación será ingresado por el usuario. Aquí también se realizará el procesamiento de los datos para obtener los valores en el dominio de la frecuencia. Se implementan dos clases que permitirán trasladar a un nuevo dominio a través de la FFT. Las clases implementadas se encuentran en el Anexo III de este proyecto.

Finalmente, se obtendrá como resultados los valores máximos tanto en el dominio del tiempo (PGA), como en el dominio de la frecuencia con su respectiva amplitud máxima. En la Figura 2.12 se muestra un diagrama de todas las etapas que se deben realizar para obtener los resultados finales de esta actividad.

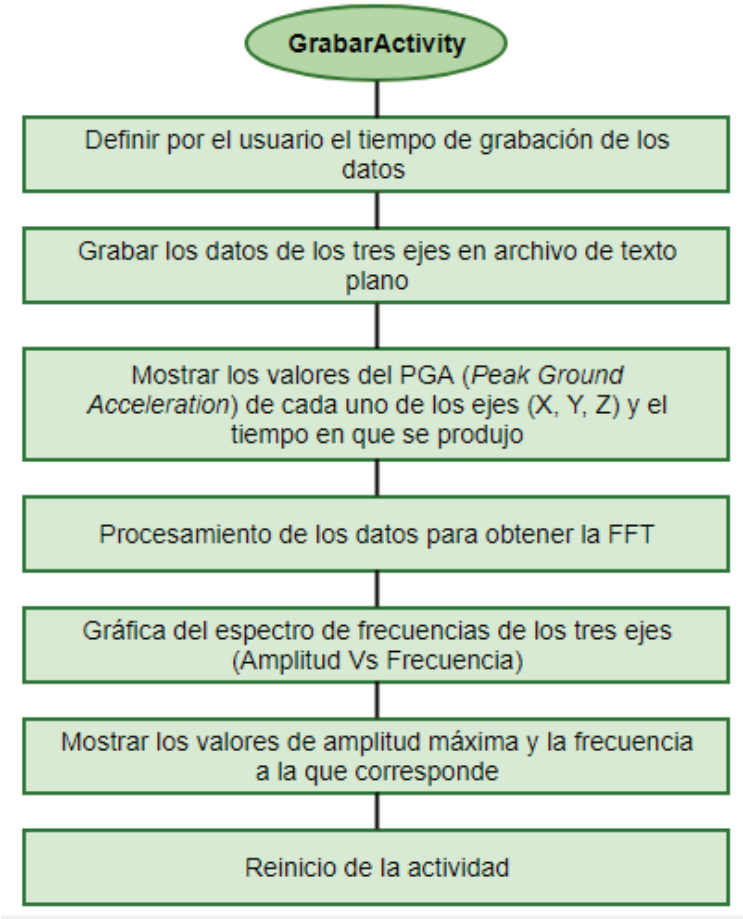


Figura 2.12 Diagrama de las etapas a cumplir en la Actividad de procesamiento de la aplicación.

2.5.1 ETAPA 1: Definir por el usuario el tiempo de grabación de los datos

Es necesario controlar el inicio y fin de la grabación en relación con el inicio y fin de un temporizador que mostrará el tiempo transcurrido de grabación, por lo que se utiliza variables booleanas. En el diagrama de la Figura 2.13 se muestra el comportamiento de las variables a lo largo de la aplicación, detallando más su funcionamiento en el transcurso de la sección.

Una variable booleana ayuda a controlar el estado de ésta, existen dos estados que permiten a los procesos que se encuentren ejecutando conocer que acción tomar frente a cada uno de los estados de la variable, se pueden asignar el estado de *true* o *false*. Un buen manejo de las acciones que se cumplen mientras la variable se encuentre en uno de los estados, se lo realiza mediante bucles y dentro de ellos se define el proceso a seguir hasta que el estado de la variable cambie.

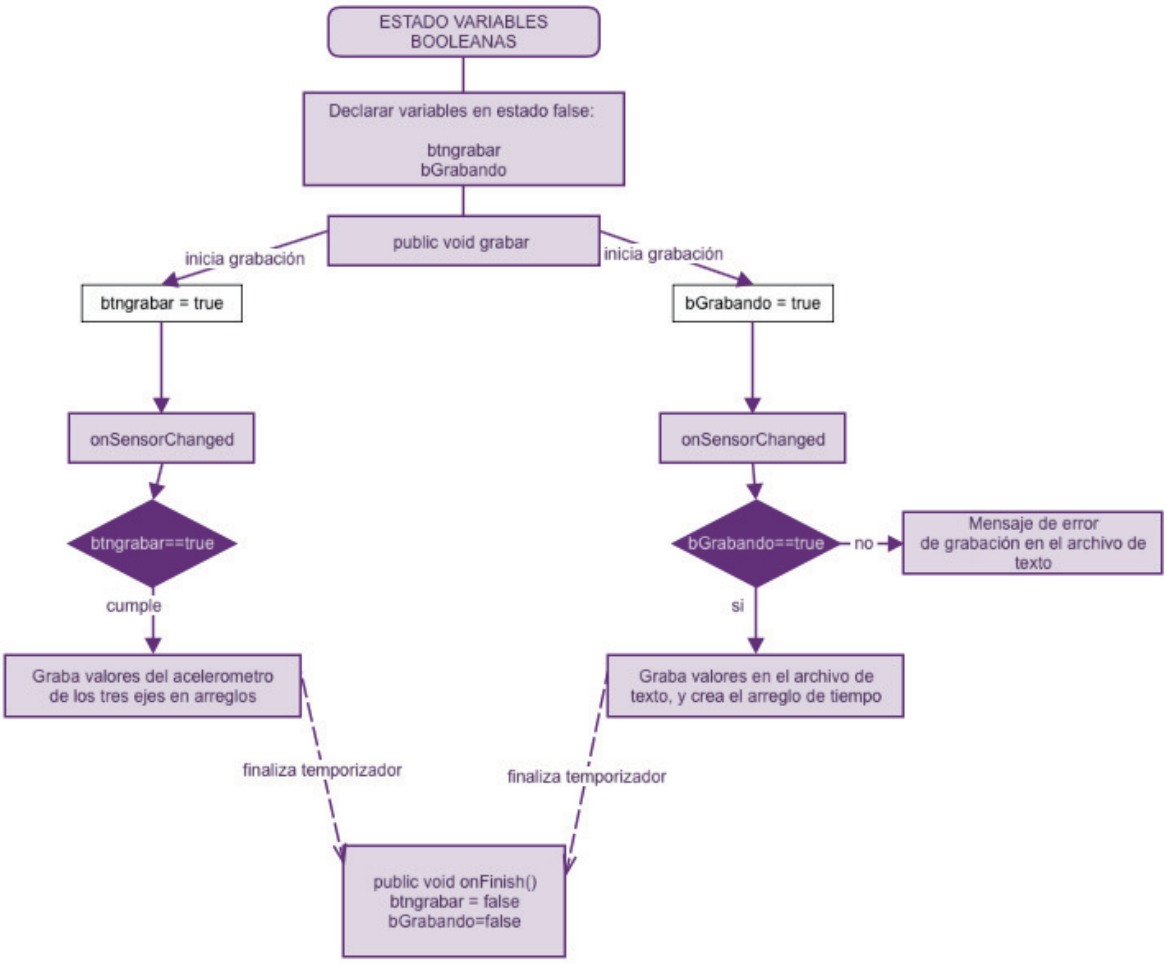


Figura 2.13 Comportamiento de las variables booleanas en la aplicación

La aplicación se basa en capturar muestras en un intervalo de tiempo para poder analizarlas respectivamente en los dos dominios. Con el fin de que esta sea más amigable se optó porque este intervalo sea ingresado de acuerdo a las necesidades que tenga el usuario que va hacer uso de la aplicación. Esto conlleva varios aspectos adicionales como la validación de los datos ingresados y los límites que estos tienen.

Al tener que realizar el análisis en el dominio de la frecuencia es necesario utilizar la Transformada de Fourier, el algoritmo que se usa requiere que la longitud de muestras que se utiliza sean en potencia de 2. Todos estos aspectos llevan a establecer un parámetro adicional, el número de muestras se establecen en 8192 para poder tener un rango de almacenamiento de 60 segundos (para los 60 segundos se requiere alrededor de 6000 muestras con una frecuencia de muestreo de 100 Hz.

Varios de los parámetros que necesitan establecerse en la caja de texto correspondiente al tiempo, se los realiza dentro del diseño de la actividad, es decir, en el layout y configuración correspondiente. Dentro del establecimeinto de los parámetros del diseño se

puede especificar las características del formato que se debe ingresar dentro de los campos de edición de texto.

Esta parte gráfica del diseño del código, muestra las propiedades del EditText:

```
<EditText
    android:id="@+id/tiempo"
    android:digits="1234567890"
    android:hint="Ingrese tiempo para grabar (5-60) seg"
    android:inputType="number"
    android:maxLength="2"
    android:singleLine="true" />
```

- *digits:* Permite controlar qué caracteres son permitidos ingresar.
- *hint:* Indica el texto de ayuda para el ingreso de parámetros.
- *inputType:* Controla qué tipo de datos admite.
- *maxLength:* Establece la longitud máxima de caracteres autorizados.
- *singleLine:* Bloquea la opción de ingresar un enter, solo admite una línea.

Con los parámetros configurados anteriormente se tiene la mayoría de validaciones para que el usuario no ingrese datos incorrectos. Permite aun ingresar valores de tiempo en un rango de 0 a 99 segundos, por lo que hay que realizar una validación final, esta parte se controla desde el código de la actividad, en el método grabar.

Para poder realizar una comparación primero se debe obtener el valor que se ingresa, el procedimiento para realizarlo es con las siguientes líneas de código:

```
datotiempo=tiempo.getText().toString();
temp_ingresado=Integer.parseInt(datotiempo);
```

Se toma lo ingresado en la caja de texto y se lo convierte a una variable de tipo string. A continuación se realiza la conversión a tipo entero para utilizar el condicional if en la validación.
Una vez con el valor de tiempo almacenado en una variable de tipo entero se crea una estructura condicional (if) que compara si el rango de tiempo está entre 5-60 segundos para seguir o mostrar un mensaje de alerta.

```
if (temp_ingresado >= 5 && temp_ingresado <= 60) {
//Instrucciones: cumple condición

…
…
…
```

```
} else {
    //Mensaje de error
    final String text = "Campo de tiempo incorrecto !!";
    Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
    tiempo.setText("");
}
```

Una vez que se cumplan las condiciones de validación se procede a crear el temporizador con el valor de segundos ingresados. Primero se desactiva al botón grabar ya que cumple todos los parámetros previamente establecidos dentro de los campos: nombre de archivo y tiempo.

```
boton_grabar.setEnabled(false);
temp_ingresado = temp_ingresado * 1000;
```

La variable *temp_ingresado* es la que tiene amacenado el valor de tiempo en segundos, pero el programa utiliza el tiempo en valores de milisegundos por lo que es necesario realizar la conversión.

```
new CountDownTimer(temp_ingresado, 1000) {

    @Override
    public void onTick(long l) {
        texto_contador.setText("Tiempo Restante: " + l / 1000);
    }

    @Override
    public void onFinish() {
      boton_fft.setEnabled(true);
      texto_contador.setText("Fin temporizador - Grabacion Exitosa");
      bGrabando=false;
      btngrabar=false;

}.start();
```

El método que se utiliza para el temporizador es el *CountDownTimer* que requiere de dos variables iniciales que son el tiempo total del temporizador y los pasos en los cuales se va a decrementar (los dos valores deben estar en milisegundos), a más de eso se necesitan de dos sub-métodos adicionales que son el *onTick* y el *onFinish.*

El método *onTick* funciona similar a un lazo for en decremento, cada vez que se produce un salto se pueden realizar acciones determinadas por el usuario. En este caso se indica mediante un texto el valor de la cuenta regresiva en pasos de uno.

Finalmente en el método *onFinish* se realiza el proceso determinado por el usuario al finalizar el contador. Inicialmente se establecen algunos parámetros que dan por terminado el conteo regresivo; entre ellos se activa el botón FFT que da paso al proceso consecuente. Se muestra también un mensaje que indica que el proceso terminó, así como el cambiar el

estado de las variables que funcionan como controladoras al momento de almacenar los datos. Es en esta etapa donde se muestra la información del PGA y datos de la grabación.

En un inicio es importante mostrar datos generales sobre el archivo de grabación. Entre los que se tiene inicialmente la fecha y hora de grabación asi como la frecuencia a la que se está trabajando.

Antes de generar las variables para imprimir el tiempo es necesario darles un formato de cómo se desa mostrar los resultados tanto a la fecha como a la hora.

```
SimpleDateFormat formato_hora = new SimpleDateFormat("HH:mm:ss");
Date horaActual = Calendar.getInstance().getTime();
str_hora = formato_hora.format(horaActual);

final Calendar fecha = new GregorianCalendar();
Date fechaActual = fecha.getTime();
SimpleDateFormat formato_fecha = new SimpleDateFormat("yyyy-MM-dd");
str_fecha = formato_fecha.format(fechaActual);
```

Para las dos instancias se utiliza la estructura que se importa, adicionalmente relacionada con el calendario gregoriano. Finalmente, se les asigna a variables de tipo string el valor de la fecha y la hora con su respectivo formato.

```
texto_datos.setText(String.format("Inicio de grabacion: %s
%s",str_fecha,str_hora));
```

De esta manera queda impresa la fecha y hora a la que se inicio la grabación ya que los valores se toman cuando se presiona el botón grabar.
Trabajar con el valor de la frecuencia de muestreo es un tanto diferente, ya que ésta es una variable heredada de la anterior actividad por lo que primero es necesario recuperarla con el mismo formato que se envió para evitar errores (en este caso se usa la instruccion `getFloat`).
Para ello se utiliza la instancia bundle que extrae en la actividad actual (propiedad: *this*) la variable que se envió con el nombre de "*Frecuencia".* La frecuencia recibida como variable es en realidad el get-min-delay por lo que es necesario realizar una operación previa para que esta sea mostrada como frecuencia de muestreo.

```
Bundle bundle=this.getIntent().getExtras();
frecibida=bundle.getFloat("Frecuencia");
Fs=(1/frecibida)*1000000;
str_fimp=String.valueOf(Fs);
```

Finalmente es necesario cambiar esta variable de tipo float a tipo string para su posterior impresión como información adicional del archivo.

```
texto_datos.append("\nLa frecuencia de muestreo es: "+str_fimp);
```

2.5.2 ETAPA 2: Grabar los datos de los tres ejes en un archivo de texto plano

Para iniciar el proceso de almacenamiento de los datos se debe tener en cuenta el diagrama de la Figura 2.13 donde se indica los estado de las variables booleanas que manejarán el proceso de grabado. Se inicializa las variables en un estado de *false*.

```
boolean btngrabar=false,bGrabando=false;
```

Otro factor que se debe tomar en cuenta para poder crear un archivo en la memoria del dispositivo, es tener habilitados los permisos de escritura y de acceso a la memoria de almacenamiento. Esto se debe realizar asignando los permisos adecuados dentro del archivo de AndroidManifest.xml. En versiones anteriores de Android bastaba con la instrucción:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

Sin embargo, para versiones actuales se debe habilitar la solicitud de permiso mediante pantalla, donde el usuario debe elegir si permite el acceso de almacenamiento a la aplicación. Este proceso completo se indica en el Anexo I de este proyecto.

Las acciones de grabar los datos se realizan en el método grabar(), que es llamado cuando el botón Grabar es accionado y después de que las comprobaciones de los campos de edición de texto hayan cumplido con los parámetros requeridos; es decir, que se ingrese un nombre del archivo y un tiempo adecuado dentro del rango definido.

Dentro del método grabar() se crean variables para manejar los datos ingresados en los *EditText* donde se ingresa tanto el nombre del archivo con el que se guardará en la memoria del dispositivo y el tiempo de duración de la grabación. Se empieza seteando las variables booleanas en *true*, indicando que el proceso de grabación ha iniciado.

```
//Obtener el valor del edit text del campo nombre
datonombre=nombre_archivo.getText().toString();

//Obtener el valor del edit text del campo tiempo
datotiempo=tiempo.getText().toString();
```

El archivo de texto plano que se creará se guardará como un tipo de documento de formato sencillo para representar los datos en forma de tablas, donde cada columna está separada por un carácter sencillo, sea una coma, punto y coma o un espacio. Se busca que el archivo sea fácil de manejar y de abrir en Excel donde se podrá separar los datos en columnas

para posteriormente ser analizados en otro medio como Matlab, es por esto que se añade la extensión .csv a la cadena del nombre del archivo ingresado por el usuario.

```
nom_archivo=datonombre+".csv";
```

Se crea primeramente una carpeta en donde se almacenarán todos los archivos creados por la aplicación, se creará bajo el nombre de VibrerApp, dentro del directorio de la memoria de almacenamiento del dispositivo. Se obtiene el directorio raíz, a través del método getExternalStorageDirectory() y retorna un objeto de la clase File. Se realiza una comprobación de que se encuentra creado el archivo anteriormente y después se procede a crear el directorio a través de *mkdirs();*

```
File tarjeta = new
File(Environment.getExternalStorageDirectory(),"VibrerApp");
if(!tarjeta.exists()){
        tarjeta.mkdirs();
}
```

Cuando se ha creado la carpeta que contendrá todos los archivos generados por la aplicación, se muestra un mensaje Toast en la pantalla indicando la ubicación en el dispositivo.

Una vez creado el directorio se crea dentro de éste el archivo de texto; es decir se crea un nuevo objeto de la clase File y con el nombre ingresado desde la pantalla. Para escribir dentro del archivo se utiliza las clases OutputStreamWriter que convierte una cadena de caracteres en una secuencia de bytes y escribe a un FileOutputStream. Al envolverlo dentro de un OutputStreamWriter permite que un OutputStream acepte caracteres o cadenas de caracteres.

```
File file = new File (tarjeta.getAbsolutePath(),nom_archivo);
osw = new OutputStreamWriter(new FileOutputStream(file));
```

Para grabar los datos en tiempo real conforme van siendo receptados los valores de aceleración en los eventos para cada eje, se acciona la variable booleana bGrabando, esta indica que los datos serán escritos en el archivo mientras tenga un estado activo, la función de esta variable se define dentro de onSensorChanged() ya que se requiere que cada evento ocurrido en cada uno de los ejes sea escrito dentro del archivo. Finalmente se usa el método write(contenido) para escribir el contenido de los datos dentro del archivo [23].

```
bGrabando=true;
//Escribir los campos en el archivo de texto
osw.write("Fecha: "+str_fecha+"\n");
```

La acción de la variable booleana que permite la grabación se define dentro de un condicional (if), que permanecerá funcionando mientras la variable booleana permanezca en un estado *true,* se escribirá dentro del archivo los datos de los tres ejes dispuestos en columnas conjuntamente con la columna de la estampa de tiempo.

Se debe tomar en cuenta que este proceso se debe realizar dentro de un gestionador de excepciones, en este caso se ha usado *Try* y *catch*. Se lo conoce también como un sistema de tratamiento de errores que captura las excepciones que se pueden generar dentro de un bloque de código, *catch* capta los errores y contiene un código que gestionará el tipo de error que se produzca [24].

Existen dos tipos principales de excepciones: RuntimeException que se refiere a errores generados por el programador como un error de división e IOException haciendo referencia a errores que no pueden ser evitados por el programador [25].

Una vez que se ha cumplido el tiempo de grabación y el temporizador se ha detenido, se setea la variable booleana bGrabando en un estado *false*, indicando que se debe dejar de escribir el archivo de texto, se implementa el método flush () que indica que debe vaciar en el archivo de texto todos los datos que hayan quedado en el buffer y así no perder ningún dato. Seguidamente se implementa close() para cerrar el archivo y terminar con la edición.

```
osw.flush();
osw.close();
```

De esta forma se cumple con el objetivo de guardar los datos de aceleración de los tres ejes y su estampa de tiempo dispuestos cada uno en columnas dentro de un archivo de texto plano que se podrá extraer del dispositivo y realizar un procesamiento de los datos con la ayuda de otro medio como Matlab.

2.5.3 ETAPA 3: Mostrar los valores del PGA (*Peak Ground Acceleration*) de cada uno de los ejes (X, Y, Z) y el tiempo en que se produjo

Los valores de PGA se los obtiene de cada uno de los ejes, pero en este caso se presenta como ejemplo el proceso a seguir sobre el eje x siendo similares en los otros dos restantes (el código completo se encuentra en los anexos).

Es preciso almacenar los valores de cada eje en arreglos para poder trabajar con cada uno de ellos y encontrar el de mayor valor. Estos datos se obtienen desde el método *onSensorChanged* que con la ayuda de la variable booleana btngrabar y el contador numdemuestras se almacenan los datos mientras dura el temporizador antes aplicado.

```
if(btngrabar){
    vector_ejex[numdemuestras]=ejex;
    vector_ejey[numdemuestras]=ejey;
    vector_ejez[numdemuestras]=ejez;
    numdemuestras++;
}
```

Ya que los valores que pueden tomar cada uno de los vectores pueden ser positivos o negativos para encontrar el mayor sin distinción de signo se utiliza la operación valor absoluto almacenando los vectores en nuevas variables.

```
double [] pga_ejex=new double[n];
…

for(int i=0;i<numdemuestras;i++){
    pga_ejex[i]=Math.abs(vector_ejex[i]);
}
```

Para empezar, se debe asumir que el número mayor es el primero y que de igual manera su posición es la de cero.

Se realiza una comparación de todo el array utilizando la estructura repetitiva for. Por cada elemento del array que se recorre se hace una comparación si dicho elemento es mayor que el que ya se tiene almacenado. De cumplir con la condición se pone a este como nuevo número mayor y la variable de iteración del lazo como la posición.

```
maximox=pga_ejex[0];
postx=0;
for(int i=0;i<numdemuestras;i++){
    if(pga_ejex[i]>maximox){
        maximox=pga_ejex[i];
        postx=i;
    }
}
```

La variable mas importante que hay que tener en cuenta es la *postx* que almacena la posición del momento en que se produce el mayor valor.

Es importante presentar los datos del PGA acompañados del momento en el que se produjo cada uno de ellos (para los tres ejes). Por lo que se necesita crear una variable adicional con valores de tiempo que relaciona la posición en que éstos se encuentran.

Los valores de PGA son almacenados en arreglos con la ayuda de un contador auxiliar (muestrasaux) que incrementa cada vez que se agrega una muestra del acelerómetro al vector. El paso a seguir con el vector tiempo sucede de manera similar ya que hay que crear una estampa de tiempo cada vez que se guarda una muestra, para así obtener una

relación de muestras grabadas vs muestras de tiempo, es decir, ambos arreglos se generan con la misma longitud por lo que están relacionados entre si.

Y así, se tiene una relación entre los valores máximos de la señal de vibraciones y el momento en el que este valor se da, cumpliendo uno de los objetivos de la aplicación, el cual es indicar el tiempo en que se produce un mayor desplazamiento y por lo tanto una mayor amplitud de la señal de vibraciones.

Ya que cada uno de los valores se obtienen desde el método *onSensorChanged* es ahí de igual manera en las que se tiene que almacenar los valores de tiempo en los cuales se van generando las muestras.

```
if(bGrabando){
    try{
        cadenatemp[muestrasaux]=System.currentTimeMillis();
        muestrasaux++;

    }catch (IOException e){
    e.printStackTrace();
    Toast.makeText(this, "No se pudo grabar",Toast.LENGTH_SHORT).show();
    }
}
```

Los valores del tiempo se empiezan a grabar cuando el estado de la variable booleana esta en *true*. Las muestras que se guardan son generadas mediante una propiedad del sistema (currentTimeMillis) que devuelve en un valor de milisegundos la hora actual del dispositivo estos posteriormente se almacenan en el vector cadenatemp. Al utilizar el try es necesario agregar un catch de error indicando un mensaje de que la grabación se interrumpio, en caso de que suceda.

Los datos almacenados tienen el tiempo en milisegundos, por lo que para que sea más fácil de identificar para el usuario, se utiliza un formato más general (hora:minutos:segundos) para mostrar estos valores.

```
DateFormat df=new SimpleDateFormat("HH:mm:ss");
long lx =(new Double(cadenatemp[postx]).longValue());
Date horax=new Date(lx);
String salidax =df.format(horax);
```

El formato de hora acepta variables de tipo *long*[2] por lo que se realiza la transformación. Luego se utiliza el formato que se definió previamente para mostrar el tiempo en horas, minutos y segundos. Finalmente se transforma esta variable a string para su impresión.

---

[2] Variable de tipo long: El tipo long (o long int) es un tipo entero que es superior o igual al tamaño del tipo int.

Una vez con los dos valores encontrados, se imprime la información en un TextView.

```
texto_datos.append("\nPGA EJE X: "+decimales.format(vector_ejex[postx])+"
en: "+salidax);
```

En la Figura 2.14 se muestra un ejemplo de cómo se observan los datos impresos del PGA.

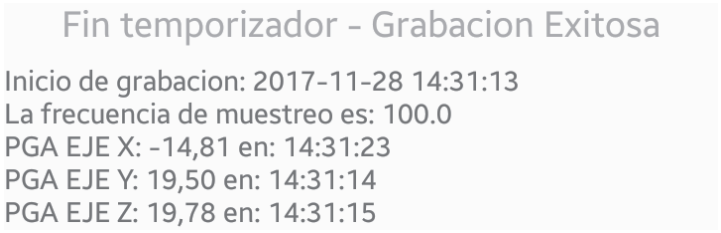


Figura 2.14. TextView con información del PGA de cada eje.

2.5.4 ETAPA 4: Procesamiento de los datos para obtener la FFT

En muchas ocasiones es necesario observar los datos en otro dominio para su análisis, ya que se puede encontrar información adicional que en el dominio original no se puede tener a la vista. En este caso los datos del acelerómetro que están en el dominio del tiempo se los pasa al dominio de frecuencia para observar la amplitud en cada componente de frecuencia de la señal temporal.

Se utiliza una función para poder cambiar de domino, en la Figura 2.15 se muestra un esquema de una idea general de lo que se realiza.

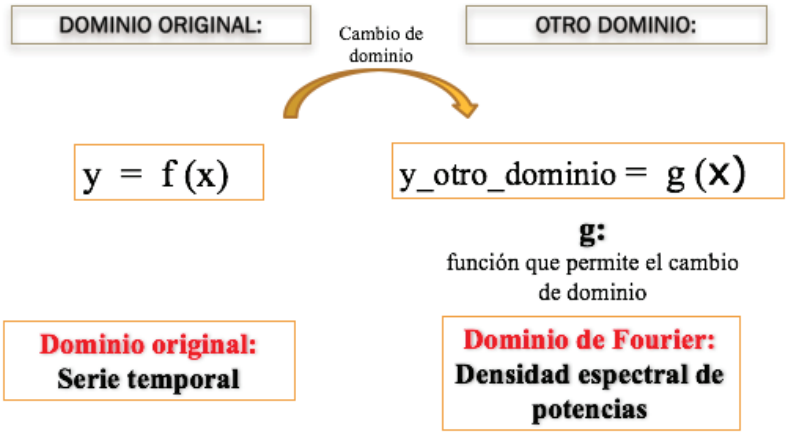


Figura 2.15. Esquema que indica el cambio de dominio de tiempo al dominio de la frecuencia.

El proceso para poder realizar el procesamiento de los datos obtenidos en el dominio del tiempo y realizar el traspaso al dominio de la frecuencia se inicia al momento de que el

usuario presiona el botón FFT, este botón se enlaza con el método Fourier, en el diagrama de la Figura 2.16 se indica un resumen de manera general de las etapas que se tienen en el método.

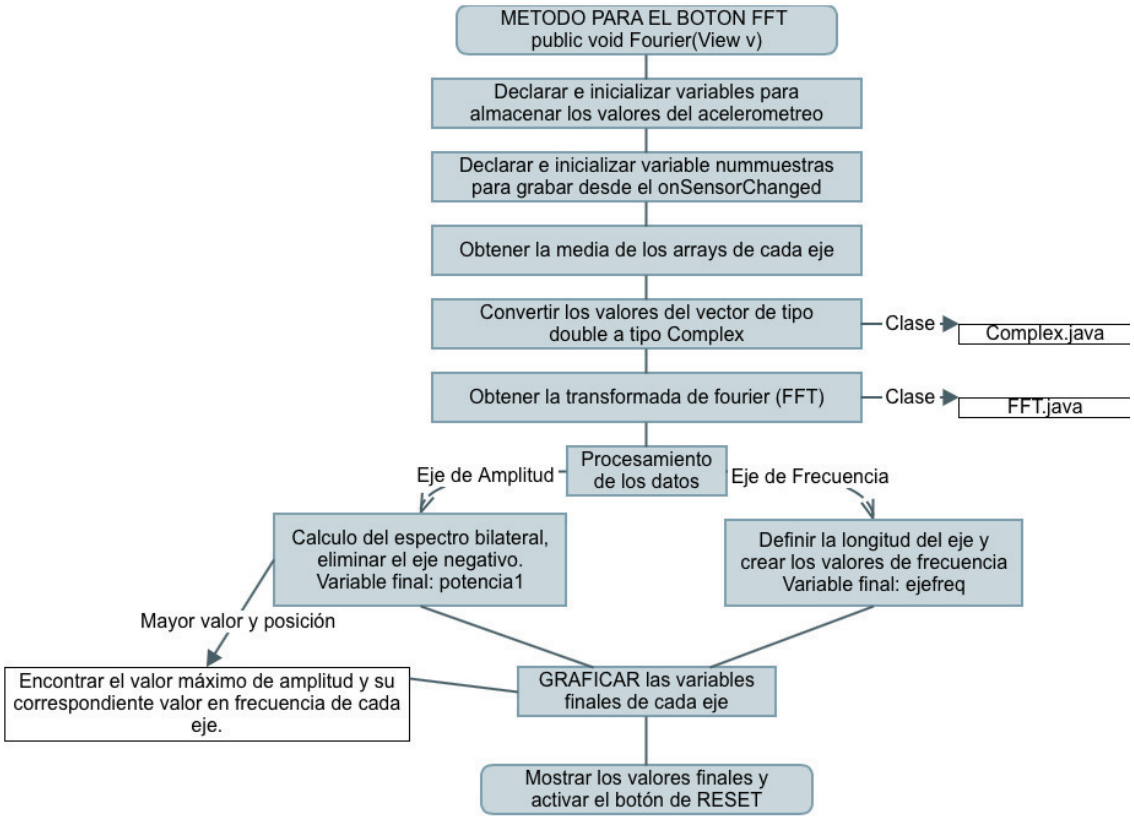


Figura 2.16. Diagrama de las etapas a cumplir al presionar el botón FFT.

Declarar e inicializar variables para almacenamiento de los datos

Existen varias formas de declarar una variable entre las que se tiene sencilla, arreglos (vectores) y matrices, a cada una se le puede agregar diferentes tipos de datos. Por facilidad al momento de realizar el procesamiento y porque se trabaja con datos que son previamente almacenados. En este caso es necesario utilizar vectores de tipo double para almacenar los datos obtenidos de cada eje.

Igualmente se necesita de variables constantes como la longitud de los vectores n (es necesario para que ésta sea de potencia 2), variables auxiliares como el número de muestras que se utilizará actuando como un contador al momento de almacenar los datos y variables booleanas que controlan el proceso de inicio y finalización de la grabación.

La variable n se inicializa en 8192 muestras debido a que el tiempo de grabación es de máximo 60 segundos. En otras palabras, cada diez segundos equivale a grabar alrededor de 1000 muestras como el máximo valor es de 60 segundos significaría que son necesarias 6000 muestras; pero esta variable debe ser estrictamente potencia de 2 por lo que el siguiente valor que abarca las 6000 muestras es 8192.

La declaración de estas variables se lo realiza en la parte inicial del código ya que quedan establecidas de manera general consiguiendo ser utilizadas en diferentes métodos, que es lo que se pretende más adelante.

```
int n = 8192, numdemuestras;
double [] vector_ejex=new double[n];
double [] vector_ejey=new double[n];
double [] vector_ejez=new double[n];
```

Variable booleana que se utiliza para controlar el almacenamiento de los datos

Ya que el método *onSensorChanged* es llamado cada vez que se produce un cambio en el sensor es aquí donde se requiere que las variables sean almacenadas en arreglos. Como se vio en el método del botón grabar, se utiliza una variable booleana que controla el estado en el que se ha presionado o no el botón de grabar y se empieza a almacenar igualmente los datos a la par en el archivo de texto como en los vectores.

La variable que controla la grabación para el proceso de fourier en este caso es la variable *btngrabar*, es una variable booleana que al estar en *true* empieza a almacenar los datos de cada eje en los respectivos vectores, en el código se puede observar como la estructura if comprueba el estado de la variable, y en caso de ser necesario graba una a una las muestras aumentando la posición del arreglo para no sobreescribir los datos. Cuando termina el conteo del temporizador esta variable cambia al estado *false* deteniendo así el proceso de almacenamiento.

```
ejex=sensorEvent.values[0];
ejey=sensorEvent.values[1];
ejez=sensorEvent.values[2];
if(btngrabar){
    vector_ejex[numdemuestras]=ejex;
    vector_ejey[numdemuestras]=ejey;
    vector_ejez[numdemuestras]=ejez;
    numdemuestras++;

}
```

Como se observa en cada parte del código el procedimiento es tratar y trabajar a cada eje por separado, por lo que de aquí en adelante se mencionará como referencia solo al tratamiento sobre el eje x (en caso de necesitar el código completo ver los anexos).

Una vez almacenados los datos en las variables concernientes y al ser estas globales, el resto del proceso se lo realiza en el método *Fourier,* que es el que está asociado al botón desde la propiedad onClick.

```
public void Fourier(View v){
…
…
…
}
```

Dentro de este método es en donde se escribirán todas las líneas de código para realizar el cambio de dominio, del tiempo a frecuencia.

Obtener la media de los valores

Inicialmente los datos que se obtienen se ven influenciados por la gravedad que afecta a uno o varios ejes en particular dependiendo de la ubicación del dispositivo. Por esta razón es indispensable obtener la media en cada eje para eliminar estos valores que afectan la apreciación de los resultados.

Este procedimiento consiste en encontrar la media del conjunto de valores almacenados y restarlos de los mismos. Se utilizan las variables media y suma de tipo double e inicializadas en cero. Es necesario aclarar que al ser arreglos, se debe trabajar dentro de lazos for (longitud máxima: número de muestras almacenadas) para poder procesar cada uno de los valores dentro de los respectivos vectores.

```
double mediax = 0;
double sumax = 0;
for(int i=0;i<=numdemuestras;i++){
    sumax=sumax+vector_ejex[i];

}

mediax=sumax/numdemuestras;
for(int i=0;i<=numdemuestras;i++){
    vector_ejex[i]=vector_ejex[i]-mediax;
```

Convertir los valores de tipo double a tipo Complex

Siguiendo con la estructura del diagrama de la Figura 2.16 antes de pasar al dominio de la frecuencia, es estrictamente necesario que los datos pasen de ser tipo double a tipo complejo (Complex) para poder utilizar la funcion *fft* de la respectiva clase, esto se lo realiza previamente añadiendo las clases necesarias.

Se hace uso de la clase "FFT.java" (http://bit.ly/2hYlCMv) que utiliza el método *Cooley-Tukey FFT*[3] que calcula la FFT de una secuencia de longitud n compleja. A su vez se tiene dependencia de la clase "Complex.java" (http://bit.ly/2hYbB1Y) que se utiliza para convertir los datos que se obtienen del acelerómetro en complejos para su respectivo procesamiento.

La manera de agregar una clase al software se muestra a continuación:

En primer lugar, se tiene que ubicar la carpeta java del proyecto, al hacer clic derecho sobre ésta se despliega un menú en el cual se puede elegir entre otras opciones agregar una nueva clase, como se muestra en la Figura 2.17.

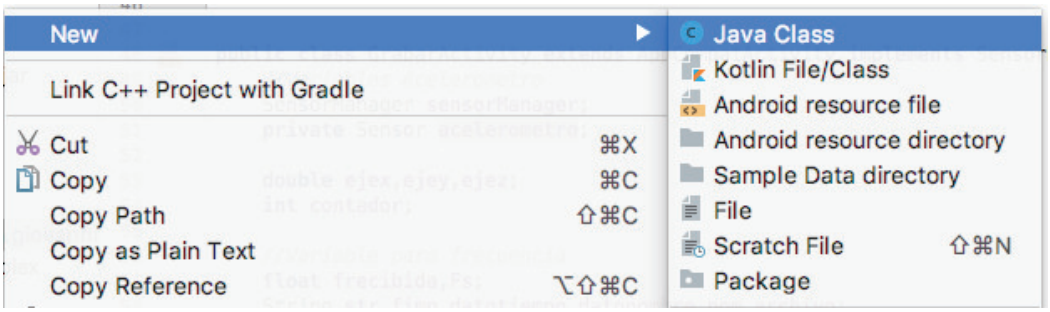


Figura 2.17. Menú para agregar una nueva clase java al proyecto.

Al elegir *Java Class* se despliega una ventana emergente (Figura 2.18) en el cual se pueden editar entre otras opciones el nombre de la clase que en este caso será FFT para la primera y Complex para la otra.

[3] The radix 2 Cooley-Tukey FFT: Usa la fórmula recursiva para calcular la Transformada discreta de Fourier.

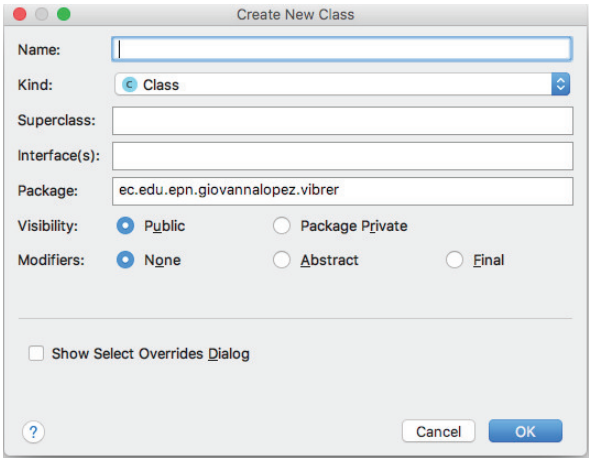


Figura 2.18. Ventana para editar las opciones que se tienen al agregar una nueva clase.

Al realizar ese procedimiento para las dos clases por separado, automáticamente se crean los archivos .java en los cuales se agrega el código mencionado anteriormente. En la Figura 2.19 se muestra una parte del código insertado en la clase Complex.java

Complex

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

/**
 * Created by giovy on 20/11/2017.
 */

public class Complex {

    private final double re;   // the real part
    private final double im;   // the imaginary part

    // create a new object with the given real and imaginary parts
    public Complex(double real, double imag) {
        re = real;
        im = imag;
    }

    // return a string representation of the invoking Complex object
    public String toString() {
        if (im == 0) return re + "";
        if (re == 0) return im + "i";
        if (im <  0) return re + " - " + (-im) + "i";
        return re + " + " + im + "i";
    }
```

Figura 2.19. Parte del código de la clase Complex.

Retomando el cambio a tipo de dato complejo este paso no afecta a los valores ya almacenados ya que únicamente se añade la parte imaginaria pero con un valor de 0.

```
Complex[] vecxcomplejo=new Complex[n];

for(int i=0;i<n;i++){
    vecxcomplejo[i]= new Complex(vector_ejex[i],0);
}
```

Obtener la Transformada de Fourier (FFT)

Como anteriormente ya se añadieron ambas clases, para poder utilizar la función *fft* es necesario pasar un valor de tipo double (no un vector), pero el procesamiento se lo realiza a todo el vector indistintamente, almacenando los datos en un arreglo diferente.

```
Complex[] vectorxfourier=fft(vecxcomplejo);
```

Con este paso se obtienen ya los valores en el dominio de la frecuencia para cada uno de los datos que se recibe del acelerómetro sobre los tres ejes.

2.5.5 ETAPA 5: Gráfica del espectro de frecuencias de los tres ejes (Amplitud Vs Frecuencia)

Para poder realizar el gráfico del espectro se debe realizar un procesamiento de los datos. Formar los ejes tanto de la amplitud (ordenadas) como de la frecuencia (abscisas). Para ello primeramente se detalla el procedimiento a seguir para el eje de amplitud.

Procesamiento de datos en el eje de amplitud

Es en la variable *potencia2* donde se calcula el espectro bilateral, es decir, se toma únicamente la mitad ya que el eje negativo es solo teórico. Para ello se utiliza la variable *aux* que sólo toma como valor la mitad de la longitud total declarada inicialmente (n). Para calcular el espectro de un solo lado se utiliza la variable *potencia1* basado en los valores de *potencia2* y en la longitud n. Se multiplica por dos para que exista el efecto de sobrelapamiento y no simplemente eliminar el eje negativo ya que se perdería información.

```
int aux=(n/2)+1;
double [] potencia2x=new double[n];
for(int i=0;i<n;i++){
    potencia2x[i]=(vectorxfourier[i].abs())/n;
}
double [] potencia1x= new double[aux];

for(int i=1;i<aux;i++){
    potencia1x[i]=2*potencia2x[i];

}
```

La variable potencia1x, potencia1y y potencia1z son las que se obtienen finalmente, en estas variables constan ya procesados los valores de amplitud de cada una de las componentes de la señal de vibración, que posteriormente se utilizará para el gráfico y para

la obtención del valor de la frecuencia predominante; es decir, la frecuencia con una mayor valor de amplitud.

Procesamiento de datos en el eje de frecuencias

El eje de frecuencias se crea a partir de la relación que existe con el número de muestras de los valores de amplitud. Los valores inicialmente se crean en pasos de 1 hasta la variable *aux* mencionada anteriormente. Para luego relacionarlas con la longitud inicial n y multiplicarlas por la frecuencia de muestreo para obtener los valores en Hertzs.

```
double [] f=new double[aux];
f[0]=0;
double[] ejefreq=new double [aux];

for(int i=1;i<aux;i++){
    f[i]=f[i-1]+1;
    ejefreq[i]=(f[i]/n)*Fs;
}
```

El arreglo de datos ejefreq, es la que se obtiene finalmente después de este proceso. Contiene los valores procesados de frecuencia, que posteriormente se utilizarán para el grafico.

Realizar la gráfica con los datos obtenidos en el eje vertical y horizontal

Las variables que se obtienen para graficar son potencia1x para el eje de amplitud y ejefreq para el eje de frecuencias. Estos datos se agregan al gráfico utilizando la misma librería que se ocupó en la actividad principal.

```
for(int i=1;i<aux;i++){
  seriesx.appendData(newDataPoint(ejefreq[i],potencia1x[i]),true,aux);
   }
```

Se asignan los valores de los tres ejes a las series que recibe la librería GraphView, que es la utilizada para realizar los gráficos de la aplicación, se ingresa los valores para el eje vertical (potencia1x[i]), los valores para el eje horizontal (ejefreq[i]) y el número de muestras. De esta forma se relaciona en el gráfico cada valor de la amplitude generada por la señal de vibraciones con el valor de la frecuencia a la que se produce dicha vibración.

2.5.6 ETAPA 6: Mostrar los valores de amplitud máxima y la frecuencia a la que corresponde

La idea inicial es buscar el número mayor dentro del array *potencia1* de cada eje y la posición que ocupa. Para ello se utilizan dos variables, en las cuales se almacenará el número mayor y su respectiva posición.

```
double potenciamayorx;
int posicionx;
```

Para empezar, se debe asumir que el número mayor es el primero y que de igual manera su posición es la cero (posición inicial del arreglo).

Se realiza una comparación de todo el array utilizando la estructura repetitiva for. Por cada elemento del array que se recorre se hace una comparación si dicho elemento es mayor que el que ya se tiene almacenado. De cumplir con la condición se pone a este como nuevo número mayor y la variable de iteración del lazo como la posición.

```
potenciamayorx=potencia1x[0];
posicionx=0;
for (int i=1;i<aux;i++){
    if(potencia1x[i]>potenciamayorx){
        potenciamayorx=potencia1x[i];
        posicionx=i;
    }
}
```

Al modificar las propiedades de títulos, leyendas y colores del gráfico se tiene el resultado final que se puede observar en la Figura 2.20 donde se muestra el gráfico y los respectivos datos de la amplitud mayor correspondiente a cada eje.

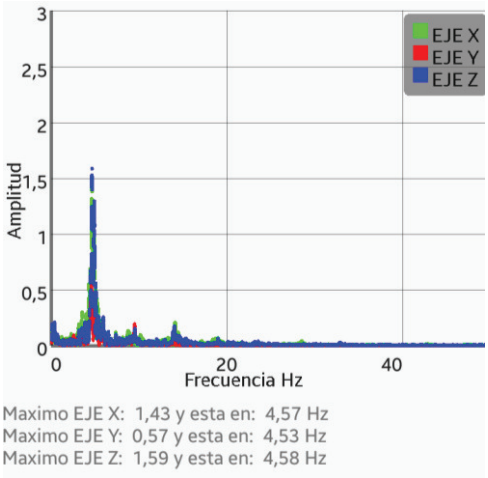


Figura 2.20. Gráfico del espectro de frecuencias y resultado de la mayor amplitud de los datos del acelerómetro en una prueba de 10 segundos.

2.5.7 ETAPA 7: Reinicio de la actividad

Finamente para terminar con la funcionalidad del botón FFT, se debe activar el botón *RESET* donde primeramente se hace visible el botón y se desactiva la FFT hasta realizar una próxima grabación de datos.

```
boton_reset.setVisibility(View.VISIBLE);
boton_fft.setEnabled(false);
```

Al relacionar las diferentes estructuras del layout con una variable sea de tipo editText, TextView o Button se puede tener acceso a las propiedades de cada una de ellas, por lo que es práctico al momento de restablecer la actividad.

Cuando se presionó el botón grabar se desactivó las dos cajas de texto hasta que termine todo el procesamiento. Pero cuando se desea realizar una nueva grabación se los debe activar y borrar. Además de entregar el foco al primer editText que es el nombre del archivo.

```
nombre_archivo.setEnabled(true);
tiempo.setEnabled(true);

nombre_archivo.setText("");
tiempo.setText("");

nombre_archivo.requestFocus();
```

Los estados de los botones regresan a su fase inicial donde está activado el de grabar pero desactivado la fft hasta que concluya la grabación. También se borran todos los campos en donde se indicó la respectiva información.

```
boton_grabar.setEnabled(true);
boton_fft.setEnabled(false);
texto_contador.setText("");
texto_datos.setText("");
texto_fourier.setText("");
```

Para que no existan posibles errores en el procesamiento se debe borrar todos los valores que estén almacenados en los arreglos que se utilizan para obtener los datos del acelerómetro.

```
for(int i=0;i<n;i++){
    vector_ejex[i]=0;
    vector_ejey[i]=0;
    vector_ejez[i]=0;
    cadenatemp[i]=0;
}
numdemuestras=0;
muestrasaux=0;
```

Finalmente y de igual manera se debe borrar los datos que se hayan agregado al gráfico, para después ocultarlo al igual que el botón de reset hasta que finalice otro proceso de grabación y fourier.

```
GraphView graficofourier=(GraphView)findViewById(R.id.grafico_fft);
graficofourier.removeAllSeries();
graficofourier.setVisibility(View.INVISIBLE);
boton_reset.setVisibility(View.INVISIBLE);
```

Una vez concluída esta actividad se cumple con el objetivo de procesamiento de los datos y obtener la Transformada de Fourier que permitirá conocer la frecuencia más notable de la señal de vibraciones y el valor de su amplitud. En la figura 2.21 se puede ver la interfaz gráfica de la actividad ejecutada en el *Smartphone.*

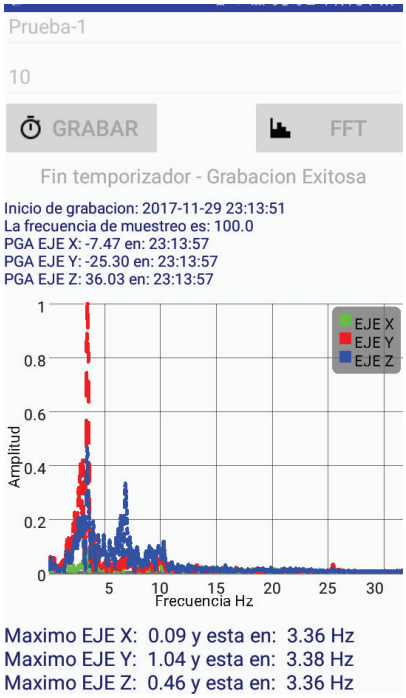


Figura 2.21. Ejecución de la segunda actividad GrabarActivity.

2.6 Actividad de información de la aplicación (*InfoActivity.java*)

Esta actividad es utilizada únicamente con el fin de mostrar información extra de la aplicación. Se tiene acceso a ésta desde el botón ⓘ ubicado en la barra de acción de la actividad principal (MainActivity).

Contiene una imagen que muestra cómo están ubicados los ejes de orientación de acuerdo al dispositivo. Además del título del proyecto así como información de la persona

responsable del desarrollo, en la Figura 2.22 se puede ver la representación de esta actividad en tiempo de ejecución.

Figura 2.22. Actividad lanzada cuando se accede a la información de la aplicación.

Al agregar esta nueva actividad al proyecto se crea en simultáneo el archivo .java que contiene el código; así como también el archivo .xml que es el que contiene el diseño de la estructura a presentar como información. No tiene interacción con el usuario aparte de mostrar datos generales sobre el proyecto.

Como se mencionó anteriormente las imágenes o recursos adicionales que se deseen agregar se deben ubicar en el directorio correspondiente (app/src/main/res/drawable).

3. RESULTADOS Y DISCUSIÓN

Como una de las partes finales de este proyecto, se tratará en este capítulo las pruebas de funcionalidad de la aplicación Android, para lo cual como se estableció en el plan del proyecto, se realizará la implementación en dos celulares *Smartphones* para comprobar su correcto funcionamiento y realizar comparaciones entre ellos para evidenciar la similitud de los resultados o diferencias. Posteriormente, se efectuarán pruebas de comparación con los resultados obtenidos en sensores de mayor gama y dedicados especialmente al análisis de vibraciones. De esta forma se podrá confirmar el cumplimiento del objetivo para lo cual fue planteado el desarrollo de esta aplicación y validar su funcionalidad.

3.1. Equipos utilizados

Para comprobar el funcionamiento del proyecto desarrollado, se procederá a instalar el APK de la aplicación, que es el fichero compacto que contiene el código desarrollado, en dos *Smartphones* diferentes. La instalación de la aplicación se realizará desde la ejecución de Android Studio en cada uno de los dispositivos. Se utilizará el Samsung J2 prime y el Huawei P9 lite.

3.1.1. Smartphone Samsung Galaxy J2 prime

Este dispositivo cuenta con un sistema operativo Android sobre el cual fue desarrollada la aplicación y por lo que permite su implementación sobre el. Cuenta con una pantalla táctil de 5 pulgadas (960 x 540 píxeles) y dimensiones de 144.8 mm x 72.1 mm x 8.9 mm. Posee un procesador de cuatro núcleos (Quad Core) de 1.4 GHz, una memoria interna de 8 GB, expandible hasta 256 GB y una memoria RAM de 1.5 GB. Tiene compatibilidad con la red 4G LTE y un Sistema Operativo Android 6.0.1 – Marshmallow. El número de modelo es SM-G532M.

Figura 3.1. Smarthphone Samsung Galaxy J2 prime [26].

3.1.2. Smartphone Huawei Nova P9 lite (2017)

Este dispositivo igualmente cuenta con un un Sistema Operativo Android 7.0 – Nougat. Cuenta con una pantalla táctil de 5.2 pulgadas (1920 x 1080 píxeles) y dimensiones de 147.2 mm de alto x 72.9 mm de ancho x 7.6 mm de espesor. Posee un procesador de ocho núcleos, 4 de 2.1 GHz y 4 de 1.7 Ghz, una memoria interna de 16 GB, expandible hasta 128 GB y una memoria RAM de 3 GB. Tiene agregado un sensor de reconocimiento dactilar para poder desbloquear fácilmente y rápido. Tiene compatibilidad con la red 4G LTE. El número de modelo es DIG-L23.

Figura 3.2 Smarthphone Huawei Nova P9 lite (2017) [27].

3.1.3. Acelerógrafo Guralp CMG-5TD (alta gama)

Este es un acelerómetro triaxial y digital, de la familia de los sensores Guralp CMG-5 con la diferencia de que lleva adicionado el digitalizador de 24 bits de una alta resolución DM-24. Este es un equipo diseñado para realizar mediciones de análisis de efecto de sitio, de estructuras y de sismos fuertes, cuenta con batería y una memoria interna de 8 Gb de almacenamiento lo que permite trasladar el equipo al lugar de análisis y garantizar un funcionamiento durante un tiempo suficiente sin necesidad de energía extra. Realiza una transmisión de los datos en tiempo real a través de la conexión a un computador mediante USB.

Se debe destacar que este acelerómetro tiene incluido internamente un filtro para tratar la señal, es un filtro de 50 Hz lo que es una gran ventaja para evitar el fenómeno de aliasing al momento de extraer la señal para tratarla con la Transformada Rápida de Fourier.

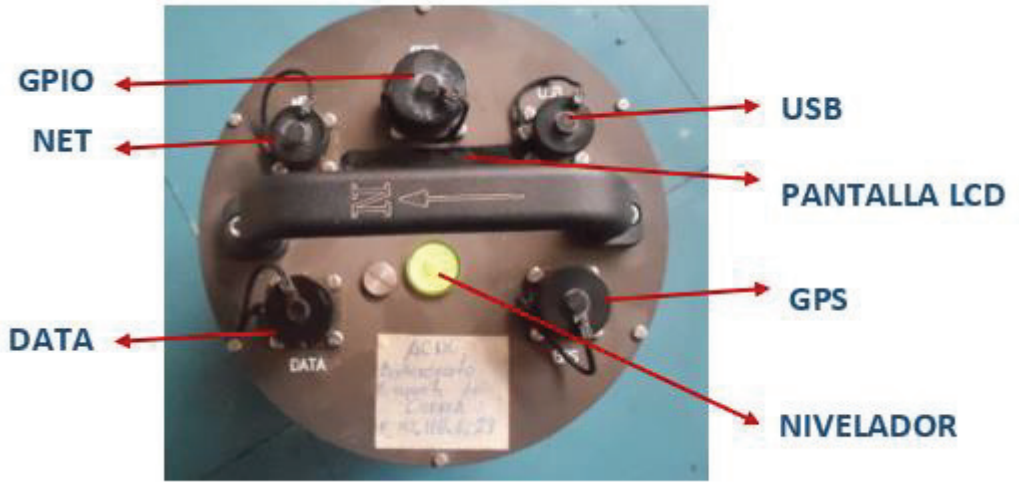


Figura 3.3 Acelerómetro Guralp Systems CMG-5TDE.

En la Figura 3.3 se puede ver el equipo utilizado originalmente para las pruebas de comprobación de este proyecto. Utiliza además una tecnología de retroalimentación por fuerza balanceada, cuenta con una interfaz para la configuración y control del sensor a través del protocolo web *http*. Trabaja con una banda de frecuencia de 100 Hz, posee 5 rangos de operación de 0.1g, 0.5g, 1g, 2g y 4g, puede trabajar en temperaturas de -20 °C a 70 °C y posee comunicaciones RS232, Ethernet y modem serial [28].

- **GPIO.-** Puerto utilizado para la supervisión, control del equipo y configuración.
- **NET.**- Puerto de entrada de la fuente de poder y puerto serial de propósito general.
- **DATA.**- Puerto para la conexión Ethernet a 100Base-TX. Soporta conexión a hub, switch o router.
- **USB.**- Puerto para la conexión de un dispositivo de almacenamiento externo mediante USB para la extracción de los datos.
- **PANTALLA LCD.**- Visualiza la configuración del equipo en 16x2 bloques.
- **GPS.**- Sirve para conectar un receptor GPS y usarlo como corrección de tiempo.
- **NIVELADOR.**- Es una burbuja de nivel para ubicación correcta del dispositivo sobre la superficie.

3.1.4. Sensor 1043_0 - PhidgetSpatial Precision 0/0/3 (media gama)

Este es un acelerómetro muy práctico y accesible ya que gracias a su tamaño brinda una fácil transportabilidad y no tiene un costo elevado. Sin embargo, no puede realizar un procesamiento independiente de datos, este se presenta como una placa electrónica, Figura 3.4, y requiere de un computador al cual se debe conectar mediante un cable USB

para poder visualizar el interfaz controlador del acelerómetro. Al adquirir este material de experimentación, se incluye el Panel de Control de Phidget disponible para máquinas Windows, iOS, Linux y MacOS.

Figura 3.4 Acelerómetro 1043 PhidgetSpatial Precision 0/0/3 [29].

El sensor *PhidgetSpatial* es un acelerómetro triaxial con la capacidad de medir hasta ± 8 g (± 78 m / s^2) de aceleración en cada uno de los ejes. Recepta tanto la vibración dinámica que se refiere a los cambios de velocidad, como la aceleración estática representada por el vector gravedad; es por esto, que se encuentra calibrado internamente. El interfaz gráfico del controlador del *PhidgetSpatial* se muestra en la Figura 3.5, los valores de cada uno de los ejes se pueden visualizar a través de las etiquetas o en los diales gráficos.

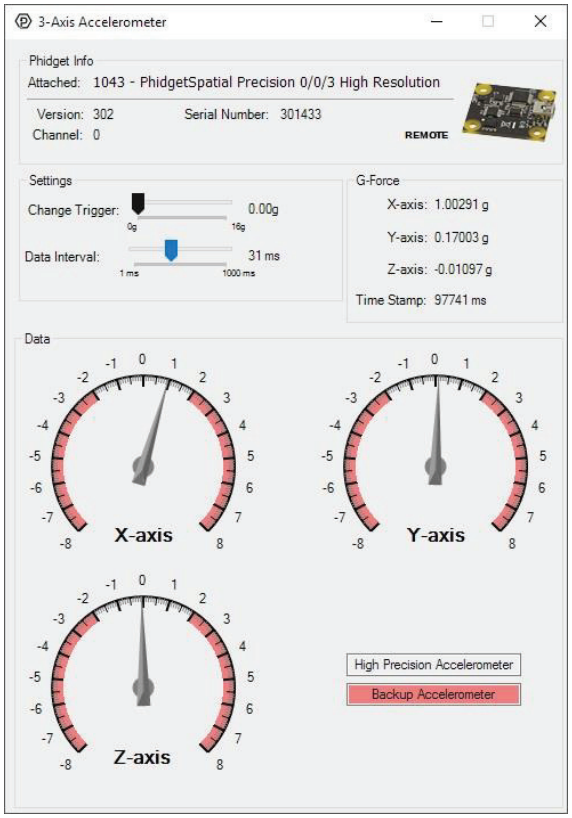


Figura 3.5 Panel de control del acelerómetro PhidgetSpatial [29].

Este acelerómetro posee un modo de alta resolución, en el cual al medir un valor menor a 2g (gravedades), toma los valores de un chip de mayor precisión. Para este tipo de mediciones se reduce aproximadamente en un factor de 10, el valor promedio del ruido blanco en cada eje, aumenta su resolución de 976 µg a 76 µg [29].

3.1.5. Acelerómetro Reftek

Este es uno de los acelerómetros más dinámicos de los sistemas sísmicos, posee un sistema autónomo de estudio de réplicas, con una batería que le permite tener una autonomía de siete días. Es un acelerómetro de 6 canales con sensores de 2 Hz de tres componentes para cada uno de los ejes de coordenas y posee también un acelerómetro MEMS triaxial. El sistema Aftershock 160-03 permite cargar desde una fuente externa que es la energía solar.

La lectura de los datos se puede realizar de manera muy sencilla, retirando la memoria USB y conectándola a un computador, o de igual forma los fabricantes ofrecen una aplicación, que es el controlador iFSC, el cual permite conectarse con el acelerómetro mediante una comunicación inalámbrica WiFi y recibir los datos en tiempo real [30].

Figura 3.6 Acelerómetro REF TEK 160-03 High Resolution Aftershock System.

3.2. Escenario de experimentación

Para realizar las pruebas de funcionamiento y comparación se utilizará la mesa de vibraciones como un generador de movimientos oscilatorios y se receptará la señal de la vibración en el analizador de vibraciones *ADQ-1600 Multichannel Analyzer*. Ambos equipos

serán proporcionados por el laboratorio de Análisis de vibraciones de la facultad de Ingeniería Mecánica de la Escuela Politécnica Nacional. La ubicación de los equipos utilizados se puede ver en la Figura 3.7 en donde se encuentran: la mesa de vibraciones como generador de la señal, el analizador multicanal que recibe la señal y el computador que contiene el software del analizador y que permite visualizar y grabar los datos receptados.

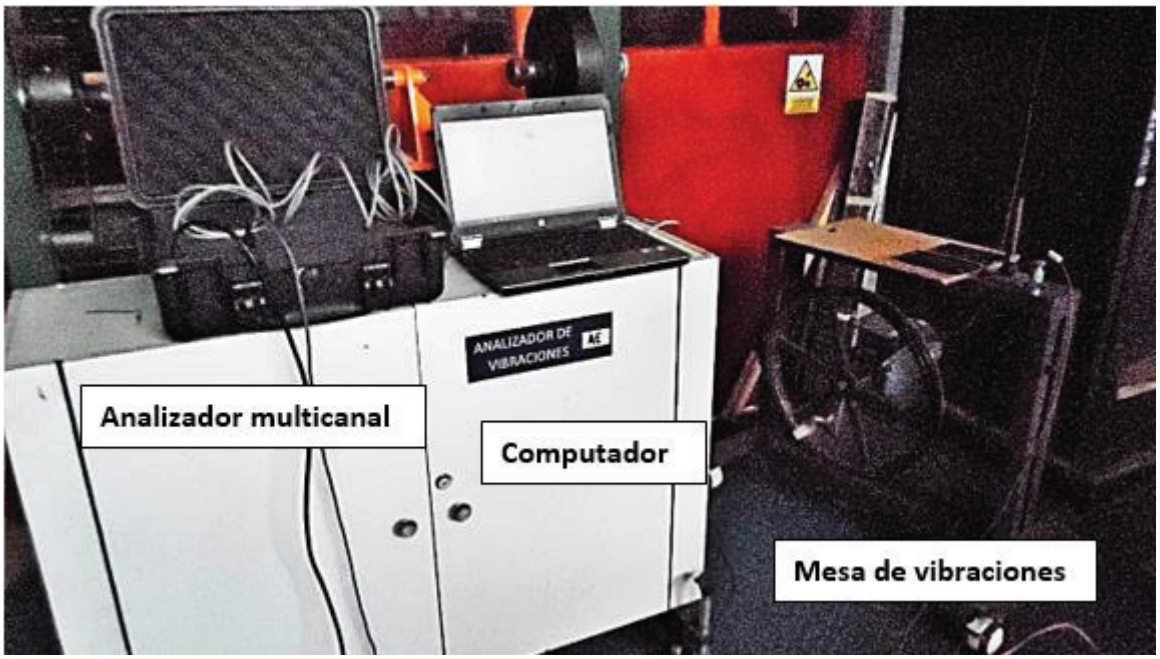


Figura 3.7. Esquema de los equipos utilizados para las pruebas de análisis de señales de vibraciones.

Las señales de las vibraciones generadas en la mesa serán receptadas tanto por los dos *Smartphones* para comprobar el correcto funcionamiento de la aplicación en cada uno de ellos y con cada uno de los sensores externos mencionados anteriormente.

Tanto la aplicación desarrollada como los respectivos sensores generarán un archivo de texto con los datos receptados en cada uno para posteriormente ser procesados y analizados con el software de Matlab.

Los sensores y *Smartphones* son colocados sobre la mesa de vibraciones como se puede ver en la Figura 3.8 para que estos puedan recibir y guardar los datos generados por las vibraciones de la mesa.

Figura 3.8. Ubicación de los sensores sobre el generador de vibraciones.

Matlab nos permite ingresar los datos de los tres ejes de los archivos de texto dentro de una variable y mediante código procesarlos para obtener la FFT; de esta forma, se puede obtener los resultados que la aplicación nos brinda al terminar la ejecución de la pruebas después del periodo de grabación y así proceder a la comparación de los resultados obtenidos entre los diferentes equipos.

3.3. Cálculo del error relativo porcentual de la frecuencia y amplitud en cada uno de los ejes

En una experimentación no se puede conocer el valor exacto de una magnitud debido a las imprecisiones del ambiente de experimentación, como imperfecciones en los equipos utilizados o a causa de las limitaciones de los mismos. El cálculo del error relativo porcentual sirve como una medida de comparación para conocer que tan exactos son los resultados obtenidos con respecto a un valor real. La teoría de errores tiene como objetivo fundamental acotar el valor de las imprecisiones conocidas como errores experimentales [33].

Para comprobar la validez de los valores obtenidos en la aplicación Android desarrollada se calculará el error porcentual entre los valores de la aplicación en cada uno de los

Smartphones en donde se encuentra instalada la aplicación con respecto al valor obtenido del sensor utilizado. Para obtener el error en porcentaje se utiliza la Ecuación 3.1.

$$\%Error\ porcentual = \frac{|Valor\ exacto - Valor\ aproximado|}{Valor\ exacto} \times 100$$

Ecuación 3.1 Ecuación del error porcentual [34].

Para este ambiente de experimentación se tiene como datos los valores captados tanto por la aplicación instalada en los *Smartphones* Huawei Nova P9 lite y el Smansung J2 Prime, como los datos receptados por los acelerómetros especializados Guralp Systems CMG-5TDE, 1043 PhidgetSpatial Precision 0/0/3 y Acelerómetro REF TEK 160-03 High Resolution. Por lo tanto, se tomará como valores aproximados los obtenidos desde la aplicación Android en cada uno de los celulares y como valores exactos a los valores de los acelerómetros especializados debido a que estos presentan una mayor precisión para receptar las vibraciones de la mesa de vibraciones.

3.4. Pruebas y resultados

Las pruebas fueron realizadas durante el mismo periodo de tiempo con cada uno de los equipos mecionados y se procesó los documentos de texto con los datos almacenados a través de Matlab. Mediante un script de Matlab se tomó los datos de cada uno de los ejes y se calculó el valor medio de todas las medidas para posteriormente ser restado este valor a cada una de las mediciones, este proceso se encuentra definido en Matlab como *detrend*, esto lo que hace es quitar la componente DC de la gravedad. Finalmente, se obtiene la FFT con la propia función de Matlab y como resultados se muestra el espectro de la señal y los valores del PGA en cada uno de los ejes.

Las pruebas se realizarán colocando todos los dispositivos sensores sobre la mesa de vibraciones y se grabarán los datos durante un periodo de 60 segundos, tiempo suficiente para poder obtener los resultados deseados y poder procesarlos a través de la FFT para conocer la frecuencia predominante que genera la señal de vibraciones de la mesa.

3.4.1. Prueba con el acelerómetro Guralp Systems

Para las pruebas con este tipo de acelerómetro se necesitó de la conexión del sensor a la fuente energética, conectar el computador y el sensor a través de los puertos ethernet y USB, para poder receptar los datos y guardar en el computador los datos generados durante el ensayo. El esquema de ubicación de los equipos se puede ver en la Figura 3.9.

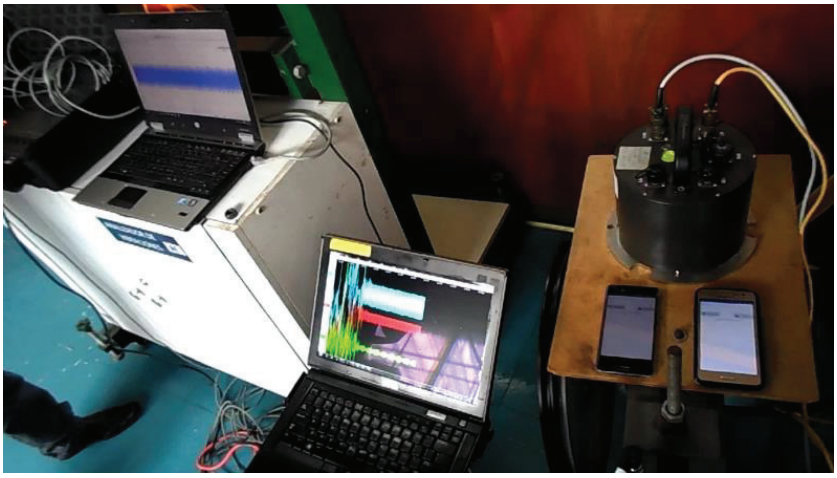
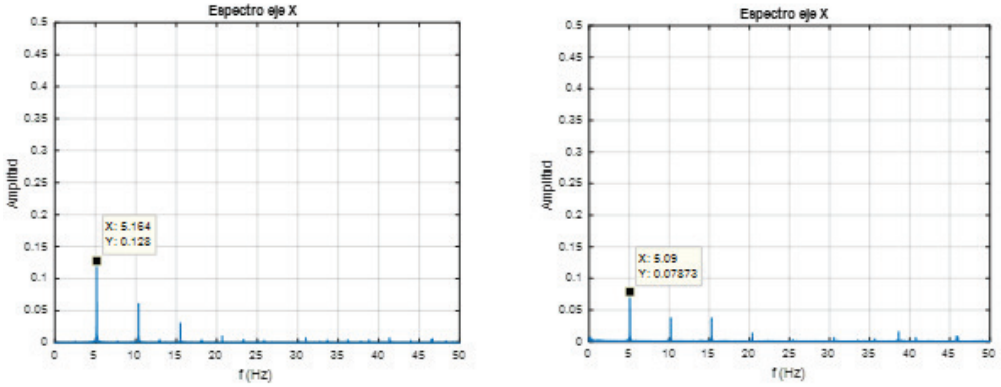


Figura 3.9. Pruebas con los dos celulares de prueba y el sensor Guralp CMG-5TDE

Tabla 3.1. Valores del PGA de cada eje obtenidos en Matlab de los dos *Smartphone* y acelerómetro Guralp Systems.

EQUIPO	PGA X (m/s^2)	PGA Y (m/s^2)	PGA Z (m/s^2)
SAMSUNG J2 PRIME	-0.3333	0.1777	0.1635
HUAWEI P9 LITE	-0.3689	-0.1348	0.3026
ACELERÓMETRO	-0.2612	-0.2677	0.0037



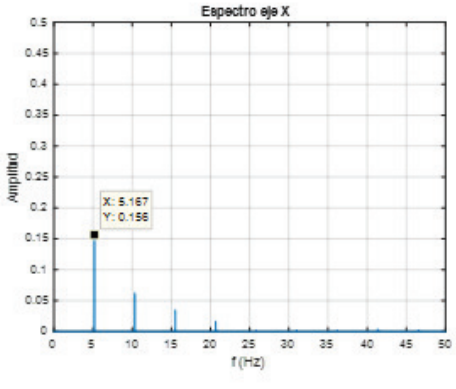


Figura 3.10. Eje X **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Guralp CMG-5TDE.

En la Figura 3.10 se presenta el espectro de frecuencia obtenido en Matlab con los datos grabados durante las pruebas para el eje X. Esta Figura muestra el espectro de frecuencia para a) smartphone Samsung J2 prime, b) smartphone Huawei P9 lite y c) Acelerómetro GURALP CMG-5TDE. De igual forma en las Figuras 3.11 y 3.12 se presentan los resultados obtenidos para los ejes Y y Z, respectivamente.

Ejemplo de cálculo del error porcentual:

$$\%Error\ porcentual = \frac{|Valor\ exacto - Valor\ aproximado|}{Valor\ exacto} \times 100$$

$$\%Error\ porcentual = \frac{|5.167 - 5.164|}{5.167} \times 100$$

$$\%Error\ porcentual = 0.058$$

Ejemplo de cálculo del eje de amplitud:

$$\%Error\ porcentual = \frac{|0.156 - 0.128|}{0.156} \times 100$$

$$\%Error\ porcentual = 17.95$$

Los resultados del cálculo del error porcentual para el eje X se los presenta en la Tabla 3.2. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.2 Error relativo porcentual obtenido entre los resultados del eje X de los *Smartphone* y el acelerómetro Guralp Systems.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.167	5.164	0.058	0.156	0.128	17.95
HUAWEI P9 LITE	5.167	5.09	1.49	0.156	0.07873	49.53

Como se puede observar en los resultados mostrados en la Tabla 3.2 no existe un gran porcentaje de error con respecto a los valores de frecuencia obtenidos en la aplicación instalada en los celulares. Sin embargo, al referirse al valor de amplitudes, en este caso si se puede notar que el celular Huawei tiene un porcentaje de error alto, lo que nos indica que se posee una menor sensibilidad para receptar la intensidad de las vibraciones.

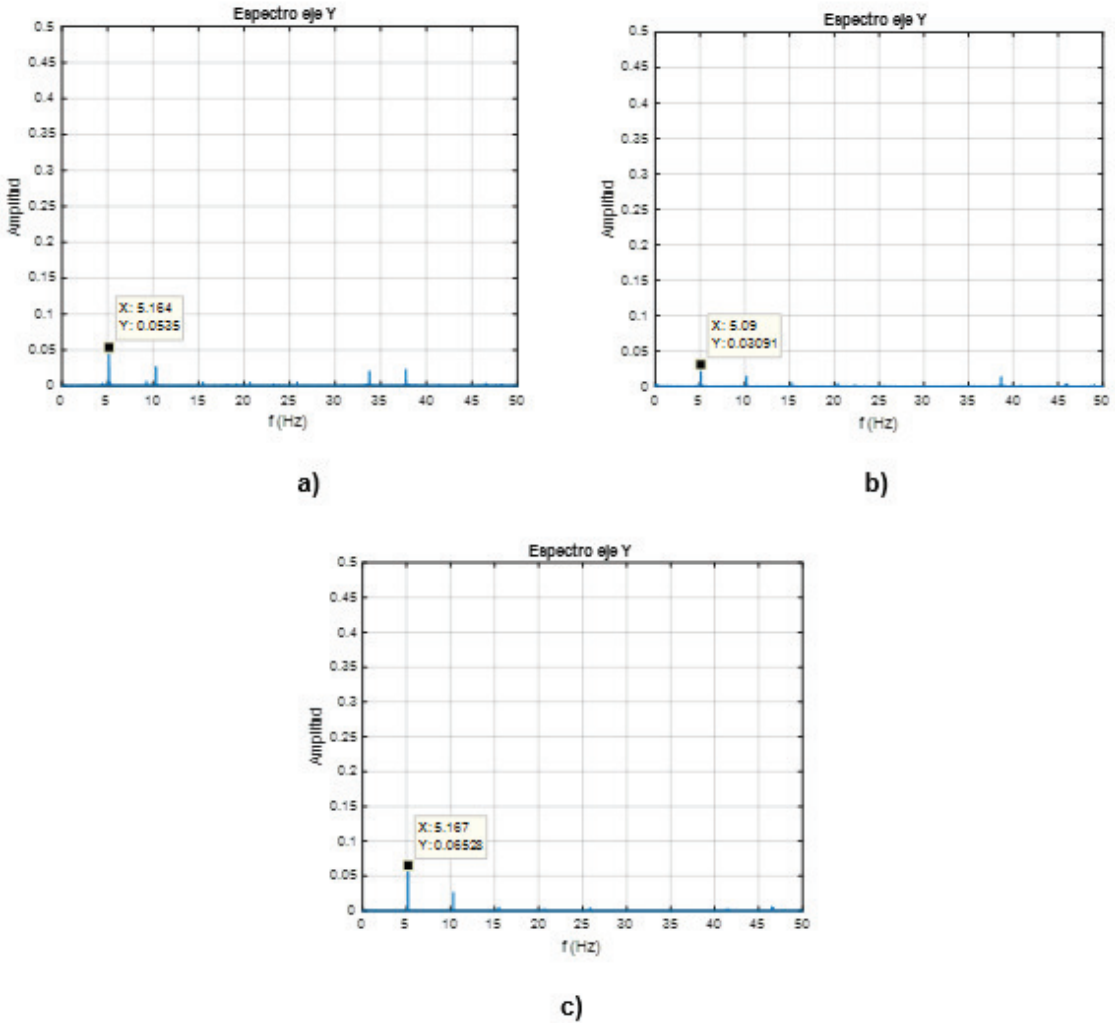


Figura 3.11. Eje Y **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Guralp CMG-5TDE.

Los resultados del cálculo del error porcentual para el eje Y se los presenta en la Tabla 3.3. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.3 Error relativo porcentual obtenido entre los resultados del eje Y de los *Smartphone* y el acelerómetro Guralp Systems.

EQUIPO	FRECUENCIA			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.167	5.164	0.058	0.06528	0.0535	18.95
HUAWEI P9 LITE	5.167	5.09	1.49	0.06528	0.03091	52.65

Como se puede ver en la Tabla 3.3 los resultados son similares a los anteriores del eje X ya que no presentan una notable diferencia con respecto a los valores de frecuencia y de igual forma con respecto al valor de amplitud obtenido por el celular Huawei es notoria la diferencia con respecto al valor obtenido por el sensor Guralp.

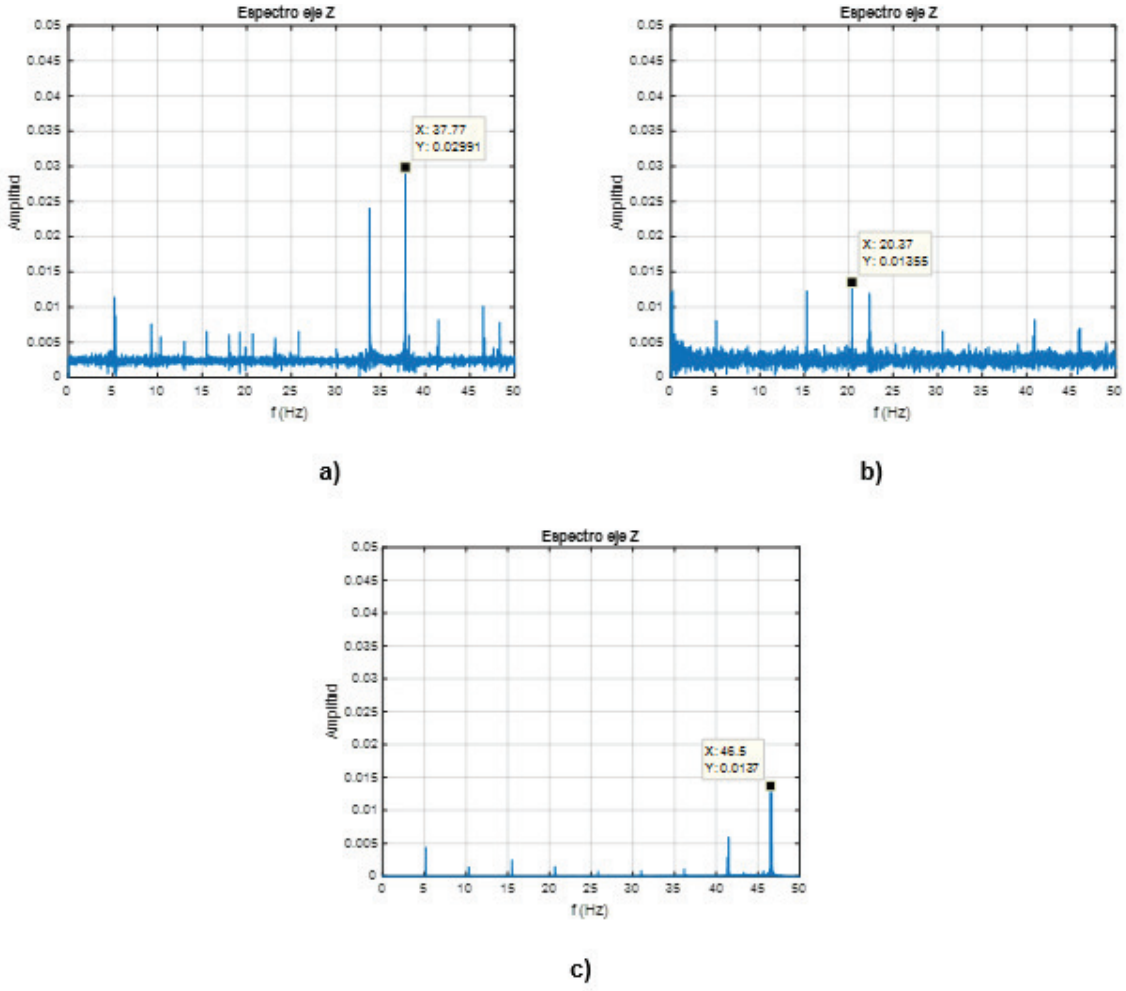


Figura 3.12. Eje Z **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Guralp CMG-5TDE.

Los resultados del error porcentual para para el eje Z se los presenta en la Tabla 3.4. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente. Como se observa en los resultados de esta tabla con respecto al eje Z, existe una notable variación en la amplitud detectada por el celular Samsung lo que pudo ocasionarse debido a la sensibilidad del teléfono en el momento de la experimentación o por la posición en la que se encontraba dispuesto.

Tabla 3.4 Error relativo porcentual obtenido entre los resultados del eje Z de los *Smartphone* y el acelerómetro Guralp Systems.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	46.5	37.77	18.77	0.0137	0.02991	118.3
HUAWEI P9 LITE	46.5	20.37	56.19	0.0137	0.01355	1.46

Como se puede evidenciar en los espectros obtenidos de cada equipo utilizado, en cada uno de los ejes surgen frecuencias similares, pero con diferente amplitud. Existe una gran similitud entre las frecuencias obtenidas en los ejes X y Y debido a que las vibraciones se realizan en dirección a estos ejes. Es notorio que el valor de la frecuencia predominante se encuentra alrededor de los 5,1 Hz y el resto de las frecuencias se generan con una mejor amplitud, pero con una separación similar de 5Hz aproximadamente.

En el eje Z se ve una diferencia en el espectro con respecto a los otros ejes, ya que las vibraciones no se iban a evidenciar sobre esta componente debido a la ubicación de los equipos. Aun así, se puede ver las frecuencias generadas aproximadamente cada 5.1 Hz.

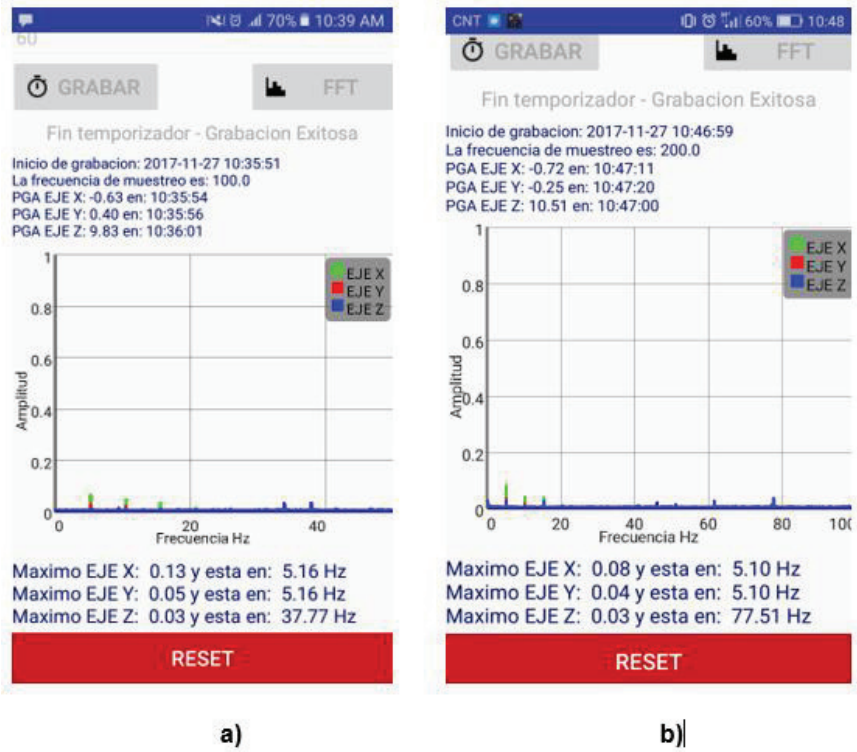


Figura 3.13. Resultados de la aplicación Android **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite.

Con respecto a los resultados obtenidos de la aplicación Android que se pueden ver en la Figura 3.13, los espectros visualizados en ambos celulares tienen gran similtud entre ellos y a su vez con los obtenidos con el acelerómetro Guralp. Según los datos que brinda la aplicación se ve que la frecuencia predominante se encuentra de igual forma en 5.1 Hz aproximadamente.

Los espectros obtenidos desde el acelerómetro Guralp Systems se presentan en el Anexo V de este proyecto, en ellos se puede comprobar de igual forma que la frecuencia predominante se da alrededor de los 5 Hz.

3.4.2. Prueba con acelerómetro Phidgets

En la Figura 3.14 se presenta la ubicación de los equipos sobre la mesa de vibraciones, este acelerómetro es de dimensiones menores con respecto al utilizado anteriormente. Pero trabaja de igual manera, a través de una conexión USB con el computador, recepta la información que el sensor registra de la señal de vibraciones de la mesa de experimentación. Al igual, permite guardar los archivos de las pruebas con los datos de los tres ejes.

Figura 3.14. Pruebas realizadas con el celular Huawei P9 lite y el sensor Phidgets Spatial.

Tabla 3.5 Valores del PGA de cada eje obtenidos en Matlab de los dos *Smartphone* y acelerómetro Phidgets Spatial

EQUIPO	PGA X (m/s^2)	PGA Y (m/s^2)	PGA Z (m/s^2)
SAMSUNG J2 PRIME	-0.2831	0.2038	-2.2007
HUAWEI P9 LITE	-0.4307	-0.1461	1.2149
ACELERÓMETRO	-0.0371	0.0419	0.0189

En la Figura 3.15 se presenta el espectro de frecuencia obtenido en Matlab con los datos grabados durante las pruebas para el eje X. Esta figura muestra el espectro de frecuencia para a) smartphone Samsung J2 prime, b) smartphone Huawei P9 lite y c) Acelerómetro Phidgets Spatial. De igual forma en las Figuras 3.16 y 3.17 se presentan los resultados obtenidos para los ejes Y y Z, respectivamente.

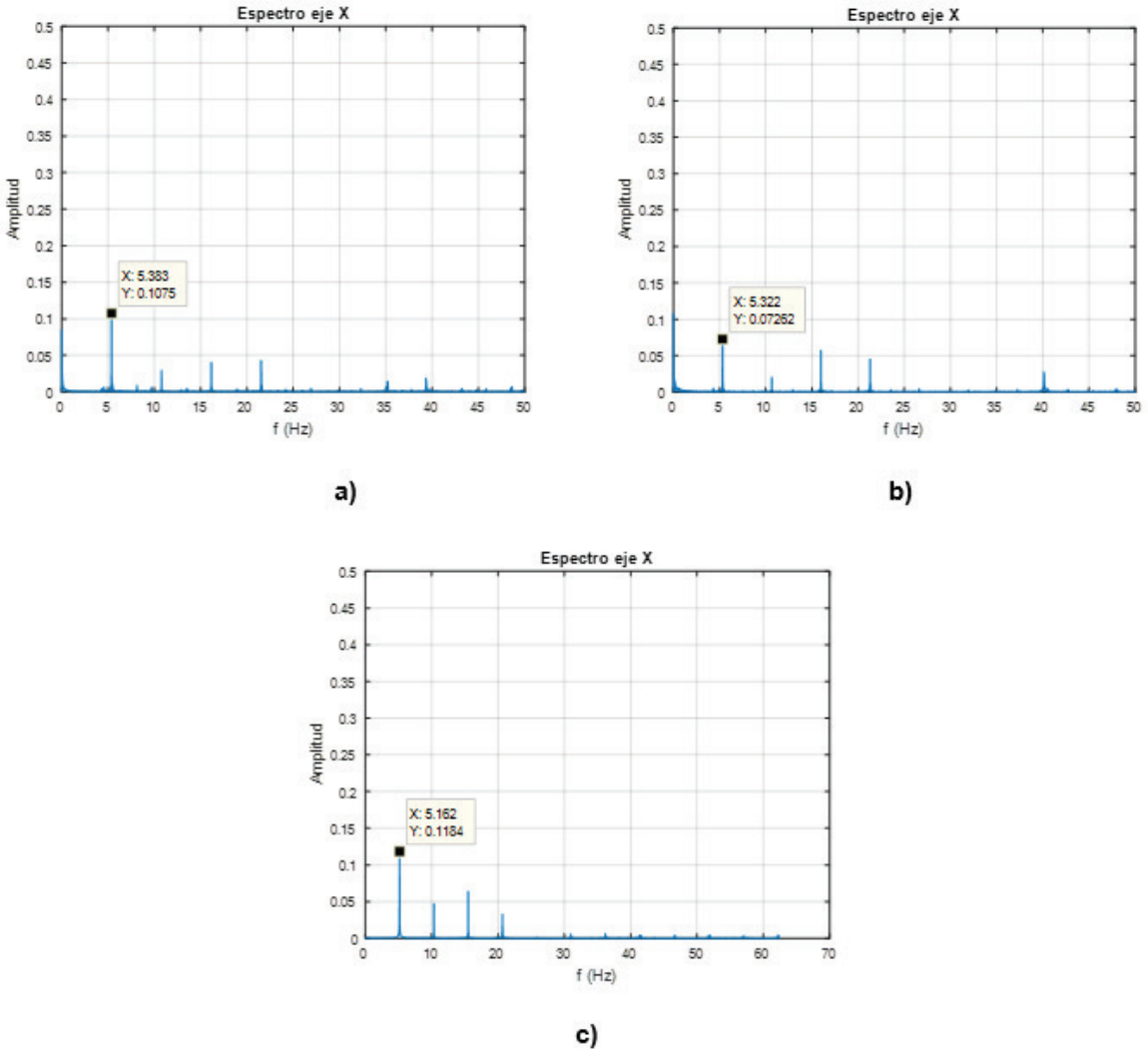


Figura 3.15. Eje X **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Phidgets Spatial

Los resultados del cálculo del error porcentual para el eje X se los presenta en la Tabla 3.6. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.6 Error relativo porcentual obtenido entre los resultados del eje X de los *Smartphone* y el acelerómetro Phidgets Spatial.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.162	5.383	4.28	0.1184	0.1075	9.2
HUAWEI P9 LITE	5.162	5.322	3.1	0.1184	0.07262	38.6

De acuerdo a la Tabla 3.6 se observa que los resultados de la frecuencia predominante obtenida no tiene mayor diferencia entre cada uno de los dispositivos ya que el porcentaje de error no supera el 5%; sin embargo, en el valor de amplitud si se puede evidenciar que existe una gran variación lo que pudo producirse por la calibración del sensor Phydgets ya que puede presentar la amplitud en unidades diferentes a las de la aplicación.

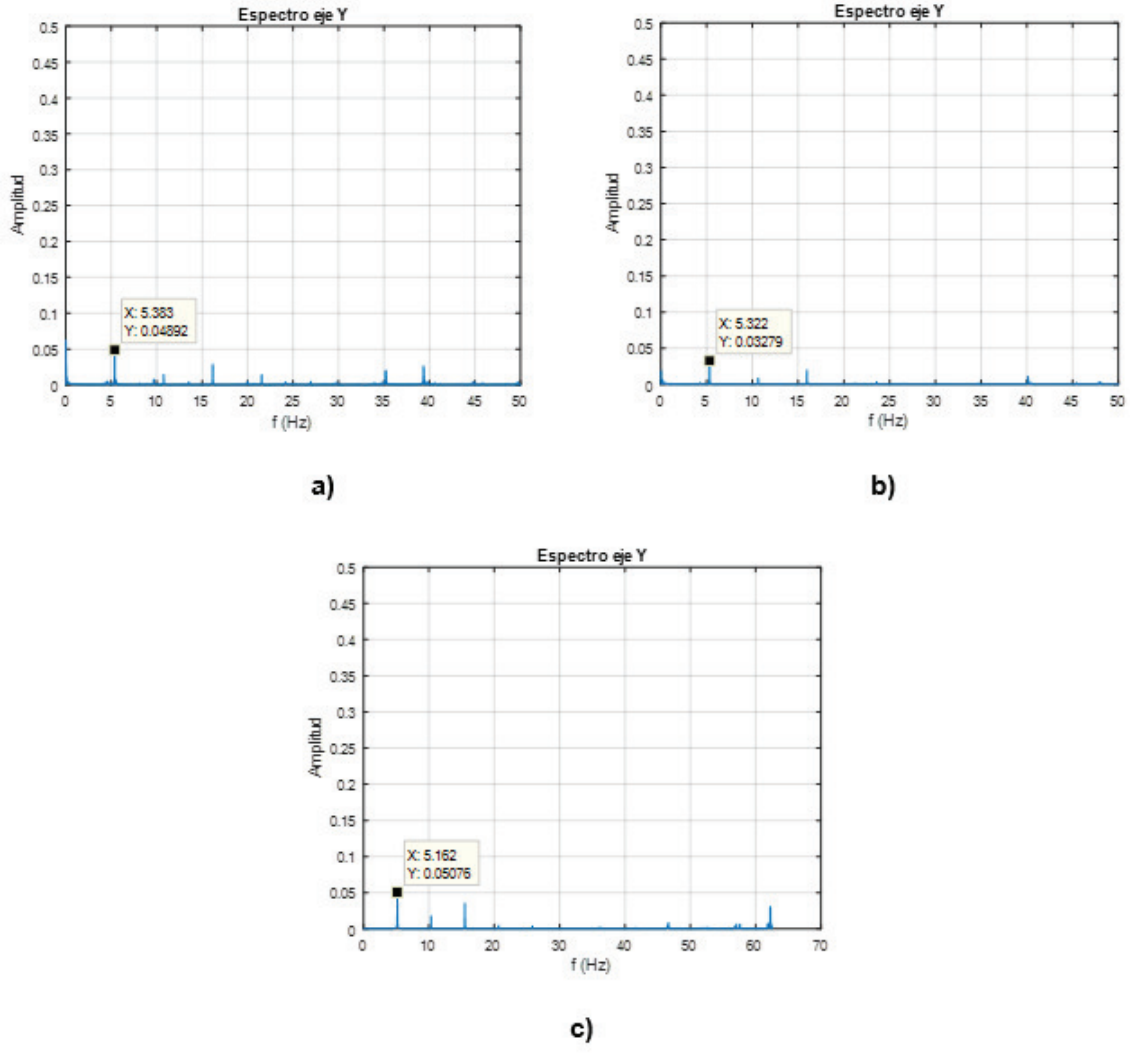


Figura 3.16. Eje Y **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Phidgets Spatial

Los resultados del error porcentual para el eje Y se los presenta en la Tabla 3.7. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.7 Error relativo porcentual obtenido entre los resultados del eje Y de los *Smartphone* y el acelerómetro Phidgets Spatial.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.162	5.383	4.28	0.05076	0.04892	3.62
HUAWEI P9 LITE	5.162	5.322	3.1	0.05076	0.03279	35.4

De igual forma que en el eje X, los resultados de la Tabla 3.7 para la frecuencia predominante son muy similares y sin mayor variación; pero para los valores de amplitud existe nuevamente una gran diferencia de los resultados obtenidos.

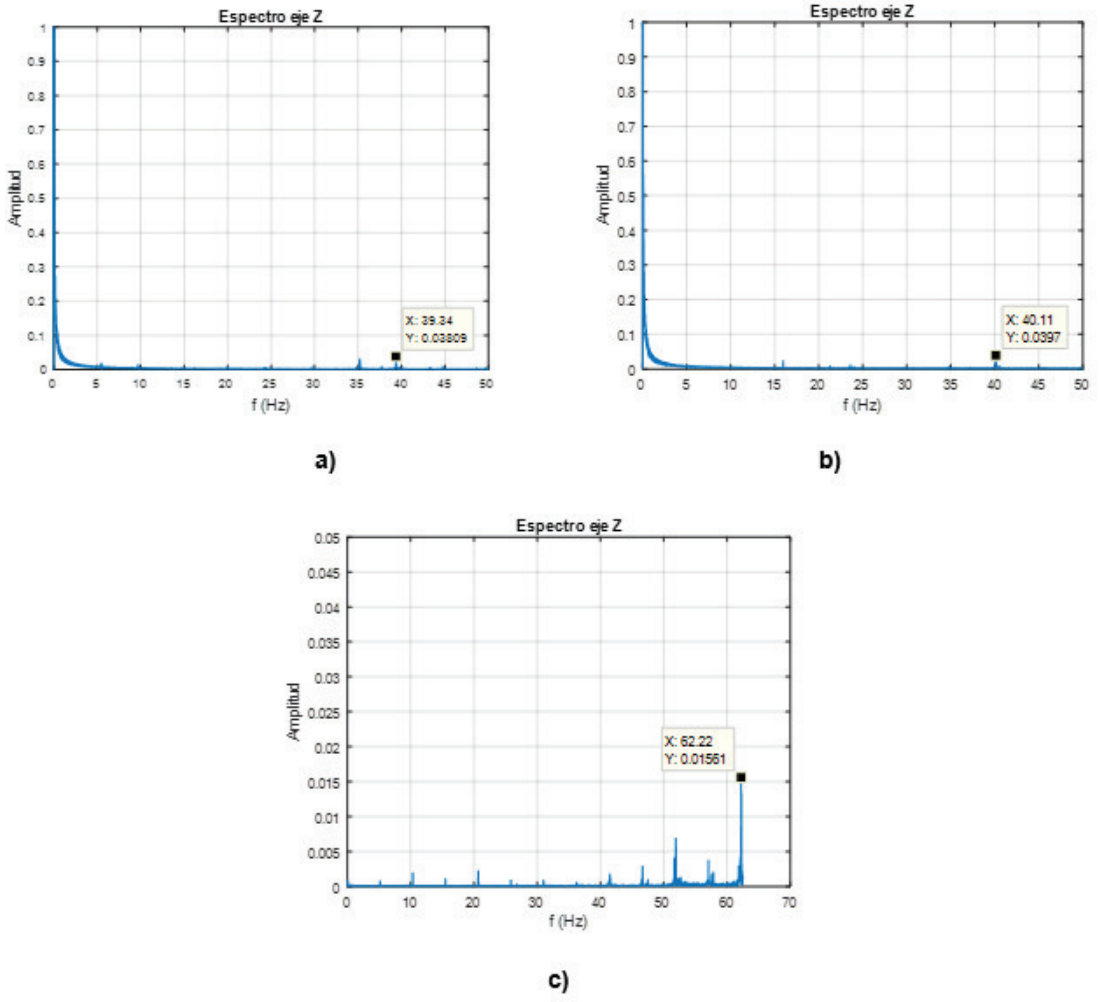


Figura 3.17. Eje Z **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro Phidgets Spatial

Los resultados del error porcentual para el eje Z se los presenta en la Tabla 3.8. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.8 Error relativo porcentual obtenido entre los resultados del eje Z de los *Smartphone* y el acelerómetro Phidgets Spatial.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	62.22	39.34	36.7	0.01561	0.03809	144
HUAWEI P9 LITE	62.22	40.11	35.5	0.01561	0.0397	154

Tanto en las imágenes presentadas de los espectros como en las tablas de error porcentual obtenidos de los dos celulares y el acelerómetro Phidgets Spatial (Figura 3.15, 3.16, 3.17, Tabla 3.8), se puede ver que se asemejan a los obtenidos con el acelerómetro Phidgets. La frecuencia predominante en los ejes X y Y está alrededor de 5.3 Hz y el resto de frecuencias se generan con la misma separación de aproximadamente 5 Hz.

En los espectros obtenidos en el eje Z a pesar de ser frecuencias con una menor amplitud, por lo que se debió reducir el tamaño del eje vertical de la gráfica para poder visualizar mejor, se puede ver que se generan frecuencias en los mismos valores pero con la diferencia que la predominante está alrededor de los 40 Hz en el caso de ambos *Smartphones*.

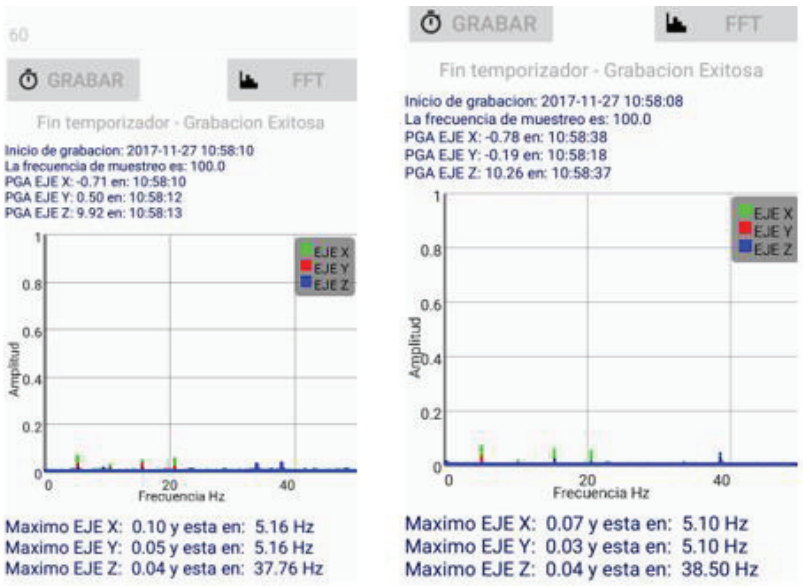


Figura 3.18. Resultados de la aplicación Android **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite.

Con respecto a los porcentajes de error obtenidos en el eje Z se destaca el grab porcentaje de error que se produce, incluso para poder visualizar los picos de las frecuencias predominantes del espectro se tuvo que aumentar la escala de la gráfica. Esto se puede deber a la configuración interna que maneja este sensor, ya que las unidades de amplitud son muy diferentes a la escala que maneja la aplicación Android.

De igual forma, en esta ocasión se obtuvo espectros similares en ambos dispositivos como se puede ver en la Figura 3.18, en los resultados de cada eje coinciden con los obtenidos en el acelerómetro Phidgets Spatial ya que en la componente X y Y, se ve la frecuencia predominante en 5.1 Hz. En la componente del eje Z, la frecuencia predominante está alrededor de los 38 Hz, muy cercana a los 40 Hz que se obtuvo en el sensor Phidgets Spatial.

Con respecto a los valores del PGA, los valores mostrados en la Tabla 3.5 son valores obtenidos despues de ser restada su media, sin embargo, en la aplicación móvil no se realiza este proceso y se presenta los valores reales en cada eje. Estos valores varían mucho de acuerdo a la precisión de cada uno de los equipos, la mayor similtud que se puede ver está entre los dos *Smartphones*.

A pesar de estas diferencias lo que se debe notar y dar mayor importancia es al espectro obtenido en cada uno de los equipos ya que estos nos brindan una información de mayor relevancia.

3.4.3. Prueba con acelerómetro REF TEK

Este acelerómetro a diferencia de los anteriores no posee una conexión por cable para poder receptar y visualizar los datos que el sensor está recibiendo, cuenta con una conexión WiFi a un dispositivo donde se encuentra instalada la aplicación que recibe los datos de cada uno de los ejes. Este acelerómetro se encuentra siempre grabando por lo que se puede extraer los datos de cualquier intervalo de tiempo.

En el dispositivo donde se visualiza las vibraciones del sensor, se muestra independientemente las señales de cada uno de los ejes, uno debajo de otro. Estas señales son mostradas en tiempo real gracias a la conexión WiFi con el sensor.

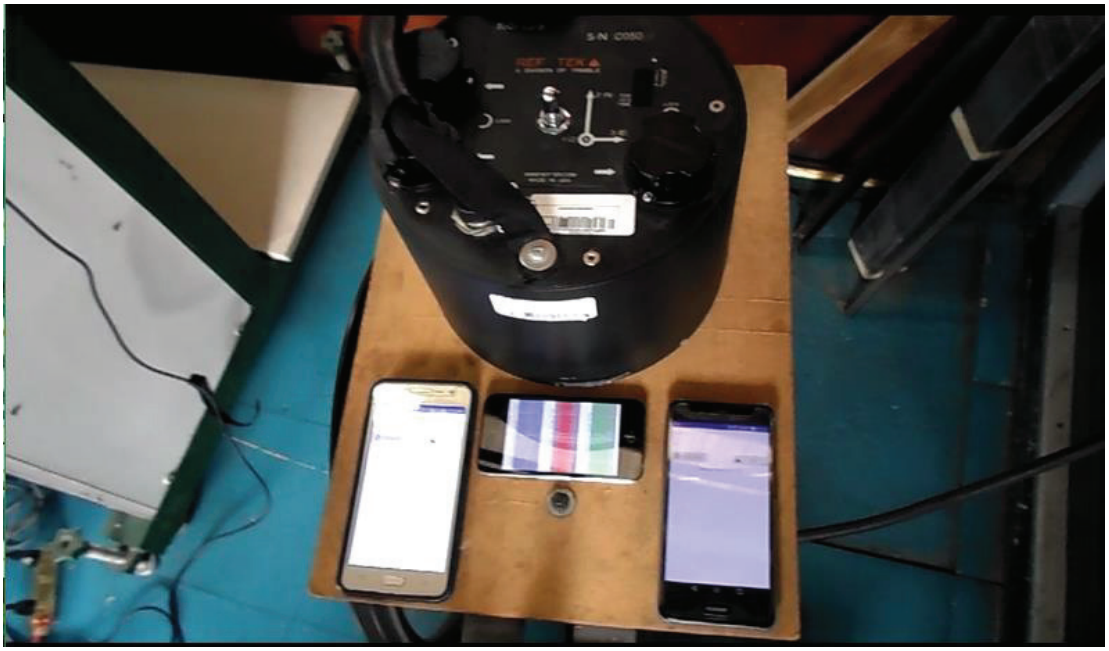


Figura 3.19. Pruebas realizadas con los dos celulares de prueba y el sensor REF TEK.

Tabla 3.9 Valores del PGA de cada eje obtenidos en Matlab de los dos *Smartphone* y acelerómetro Reftek

EQUIPO	PGA X (m/s^2)	PGA Y (m/s^2)	PGA Z (m/s^2)
SAMSUNG J2 PRIME	-0.3580	0.2299	0.5585
HUAWEI P9 LITE	-0.2949	-0.2711	-0.3821
ACELERÓMETRO	0.3991	-0.1893	0.0728

En la Figura 3.20 se presenta el espectro de frecuencia obtenido en Matlab con los datos grabados durante las pruebas para el eje X. Esta figura muestra el espectro de frecuencia para a) smartphone Samsung J2 prime, b) smartphone Huawei P9 lite y c) Acelerómetro REF TEK. De igual forma en las Figuras 3.21 y 3.22 se presentan los resultados obtenidos para los ejes Y y Z, respectivamente.

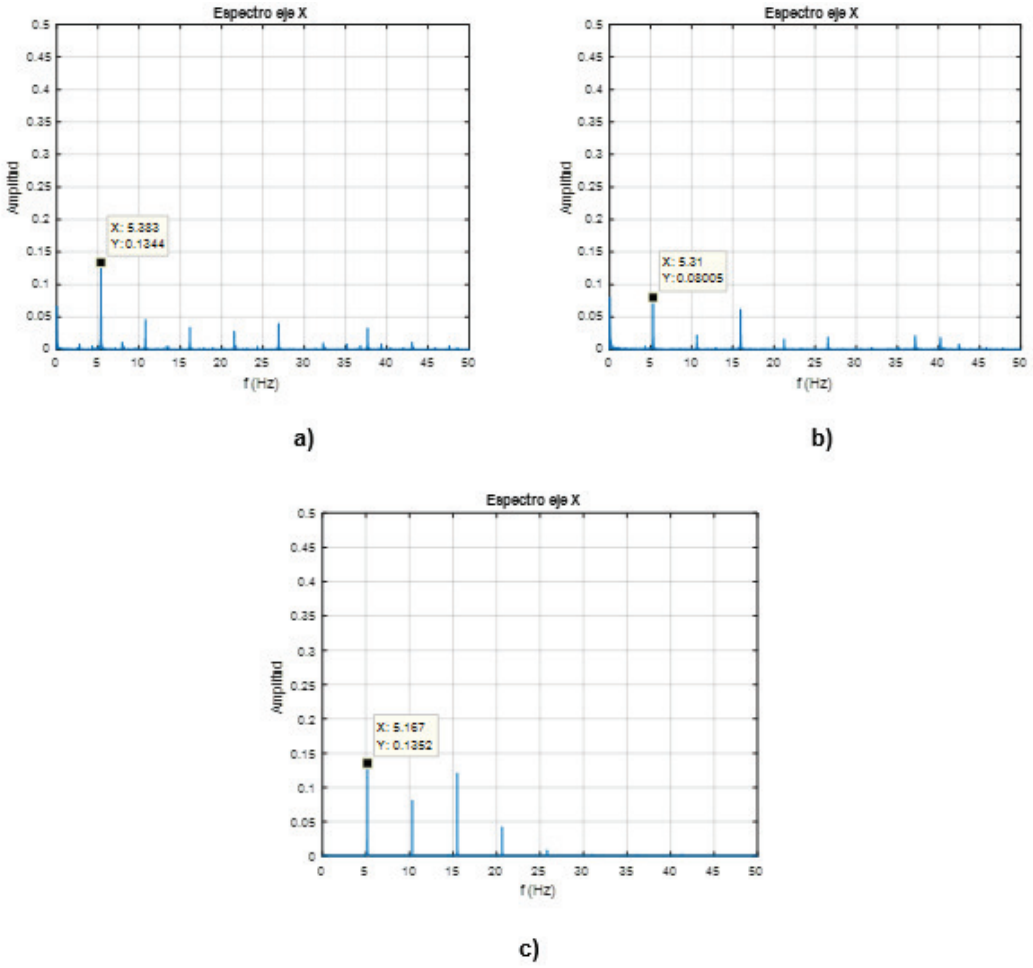


Figura 3.20. Eje X **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro REF TEK.

Los resultados del error porcentual para el eje X se los presenta en la Tabla 3.10. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.10 Error relativo porcentual obtenido entre los resultados del eje X de los *Smartphone* y el acelerómetro REF TEK.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.167	5.383	4.18	0.1352	0.1344	0.6
HUAWEI P9 LITE	5.167	5.31	2.8	0.1352	0.08005	40.8

De acuerdo a los resultados mostrados en la Tabla 3.10 se puede observar que no existe gran diferencia a excepción del valor de amplitud obtenido en el celular Huawei, una vez más se puede ver que este dispositivo no tiene una buena sensibilidad frente a la recepción de intensidad de las vibraciones.

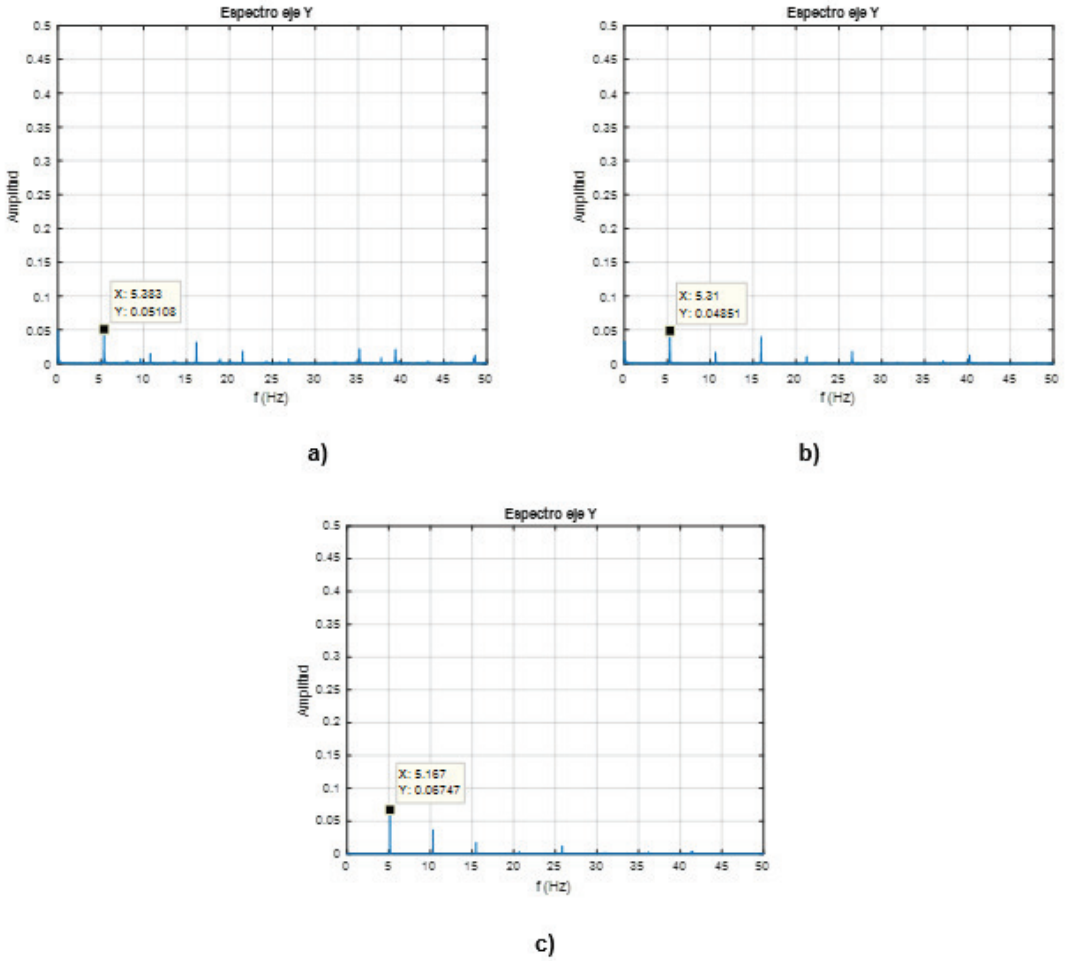


Figura 3.21. Eje Y **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro REF TEK.

Los resultados del error porcentual para el eje Y se los presenta en la Tabla 3.11. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.11 Error relativo porcentual obtenido entre los resultados del eje Y de los *Smartphone* y el acelerómetro REF TEK.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	5.167	5.383	4.18	0.06747	0.05108	24.3
HUAWEI P9 LITE	5.167	5.31	2.8	0.06747	0.04851	28.1

En este caso los resultados obtenidos son muy aceptables ya que no existe un porcentaje de error significante para llegar a conclusiones negativas de los datos obtenidos en cada una de las experimentaciones. El porcentaje de error entre ambos dispositivos es muy similar y no muy lejana de los valores resultantes del sensor especializado.

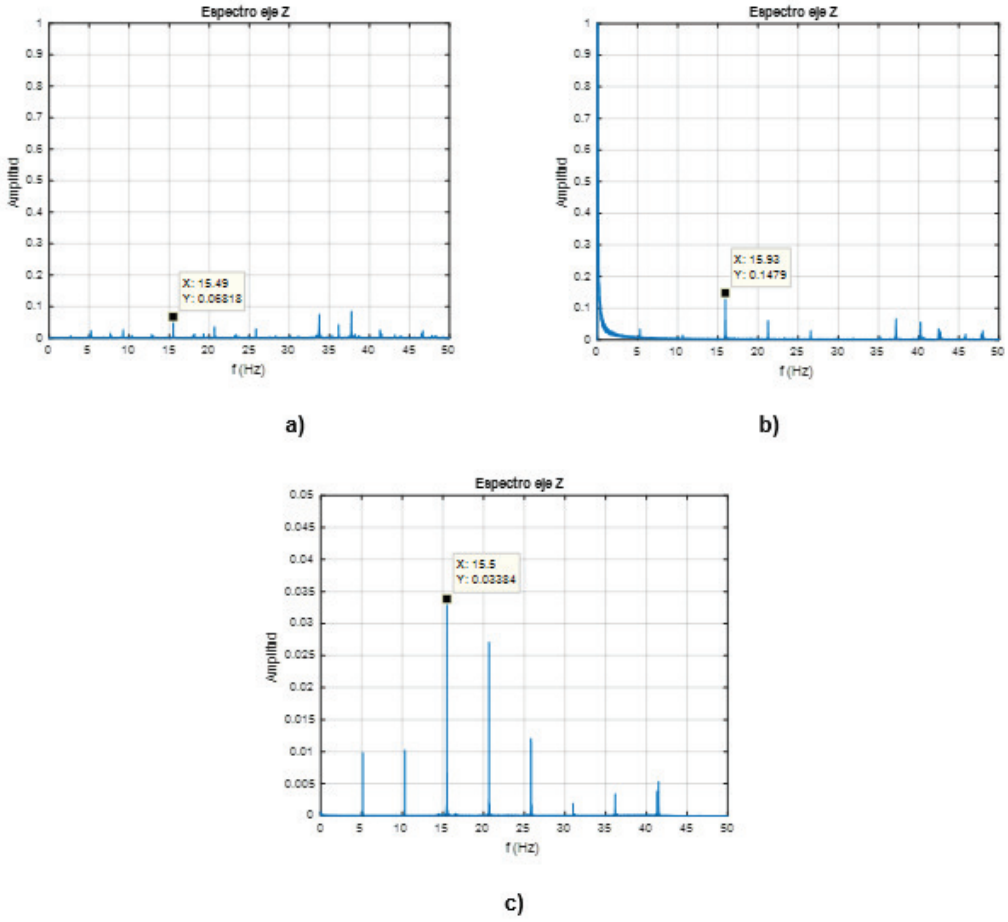


Figura 3.22. Eje Z **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite. **c)**Acelerómetro REF TEK.

Los resultados del error porcentual para el eje Z se los presenta en la Tabla 3.12. Se presenta tanto los valores para el eje de frecuencias como para el eje de amplitud y el error porcentual obtenido para cada uno respectivamente.

Tabla 3.12 Error relativo porcentual obtenido entre los resultados del eje Z de los *Smartphone* y el acelerómetro REF TEK.

EQUIPO	FRECUENCIA (Hz)			AMPLITUD		
	Valor exacto	**Valor aproximado**	**% Error**	**Valor exacto**	**Valor aproximado**	**% Error**
SAMSUNG J2 PRIME	15.5	15.49	0.06	0.03384	0.06818	101
HUAWEI P9 LITE	15.5	15.93	2.8	0.03384	0.1479	337

A pesar de los resultados obtenidos en casos anteriores, donde en el eje Z tiene una gran variación entre los resultados obtenidos en la frecuencia, en este caso no sucedió así ya que el porcentaje de error es mínimo y los valores son muy próximos entre los tres dispositivos. La diferencia se puede observar en los valores de amplitud; aún así, en comparación con los anteriores casos el porcentaje de error es menor.

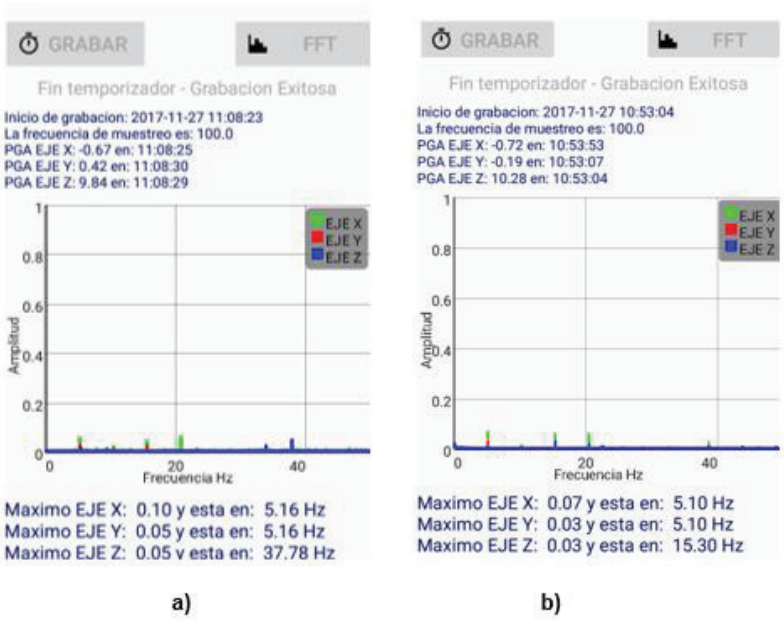


Figura 3.23. Resultados de la aplicación Android **a)** Celular Smartphone Samsung J2 prime. **b)** Huawei P9 lite.

Los resultados obtenidos en este último equipo presentan las mismas características de los anteriores espectros, la frecuencia predominante en el eje X y Y alrededor de los 5 Hz y la componente del eje Z, en este caso, prevaleció en la frecuencia de alrededor de los 15 Hz.

Los resultados obtenidos en todos los espectros que se generaron en Matlab del procesamiento de los archivos de texto de cada uno de los equipos, muestran resultados similares al espectro que se presenta en la aplicación Android, lo que valida el correcto funcionamiento de la aplicación.

Las comparaciones de los valores obtenidos en la aplicación Android, con los de los acelerómetros de mayor precisión, muestran que la aplicación brinda una gran proximidad de los valores de frecuencia de una señal de vibraciones. Validando de esta forma la

funcionalidad y utilidad de la aplicación para poder realizar un análisis in situ de una vibración.

3.4.4. Comparación entre aplicaciones similares disponibles en la Play Store de Android.

Como se mencionó en el primer capítulo de este proyecto, existen aplicaciones disponibles dentro de la plataforma virtual de distribución de aplicaciones Android, con funcionalidades similares que proponen un análisis de las vibraciones receptadas por el acelerómetro.

Para estas pruebas de comparación se tomó en cuenta ciertas aplicaciones que presentan mayores semejanzas con respecto a los parámetros e información que brindan como resultado del análisis de vibraciones. Las aplicaciones se instalaron en el mismo dispositivo *Smartphone* que posee una frecuenicia de muestreo de 100 Hz.

El ambiente experimental para estas pruebas se desarrolló sobre las vibraciones generadas por una lavadora LG de 16 Kg en su ciclo de centrifugado, con una carga vacía en su interior. El Smartphone, con cada una de las aplicaciones semejantes, se colocó sobre la tapa de la máquina y se tomó como referencia el punto de centrifugación máxima dado justo antes de que el motor del cilindro se detenga.

Las aplicaciones con mayor similaridad con las que se realizó las experimentaciones son tres: Vibratiions, VibSensor y Vibration Analyzer, todas ellas disponibles en la *Google Play Store* y gratuitas.

Resultados obtenidos de la aplicación VibrerApp desarrollada en este proyecto

Se realizaron distintas medidas capturando los datos durante 20 segundos. En este periodo se alcanzó a realizar 3 capturas de los resultados obtenidos. En la Figura 3.24 se muestra dos capturas en intervalos diferentes, se puede observar que la frecuencia de vibración predominantes coinciden en el eje X y Y con una frecuencia de 11.72 Hz y en el eje Z se tiene una de 35 Hz.

La amplitud como se puede distinguir en un caso es mayor, esto puede darse debido a que la intensidad de vibración va aumentando gradualmente durante el ciclo de centrifugado pero a una misma frecuencia.

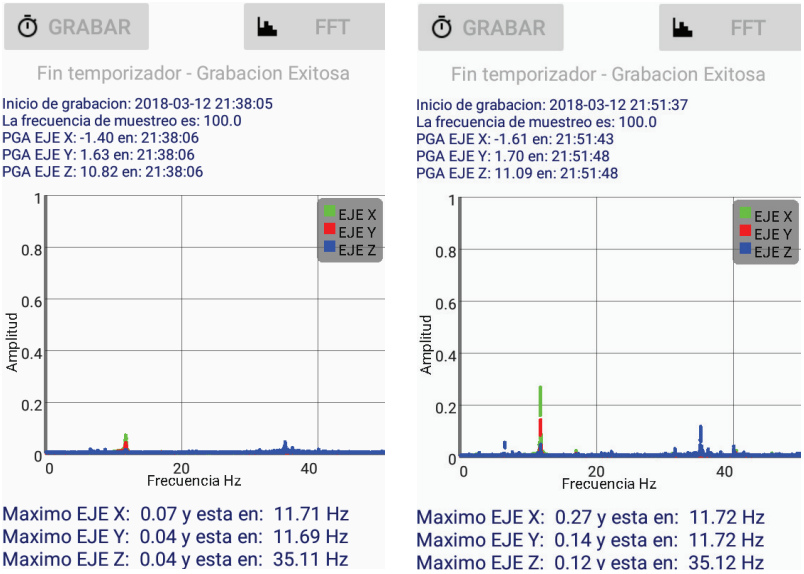


Figura 3.24 Resultados de la aplicación VibrerApp de las vibraciones de una lavadora.

Resultados obtenidos de la aplicación Vibrations

Esta aplicación nos presenta como resultado el valor de frecuencia que se recibe en el eje que se encuentra perpendicular al dispositivo más no el valor del resto de sus componentes en el eje restante. En este caso el eje que se encontraba perpendicular era el eje Z con una frecuencia de 35 Hz.

Esta aplicación recepta información tanto del acelerómetro como del micrófono ya que realiza dos análisis simultaneos y muestra el espectro de las medidas mecánicas en color rojo y el espectro de las medidas acústicas en color azúl.

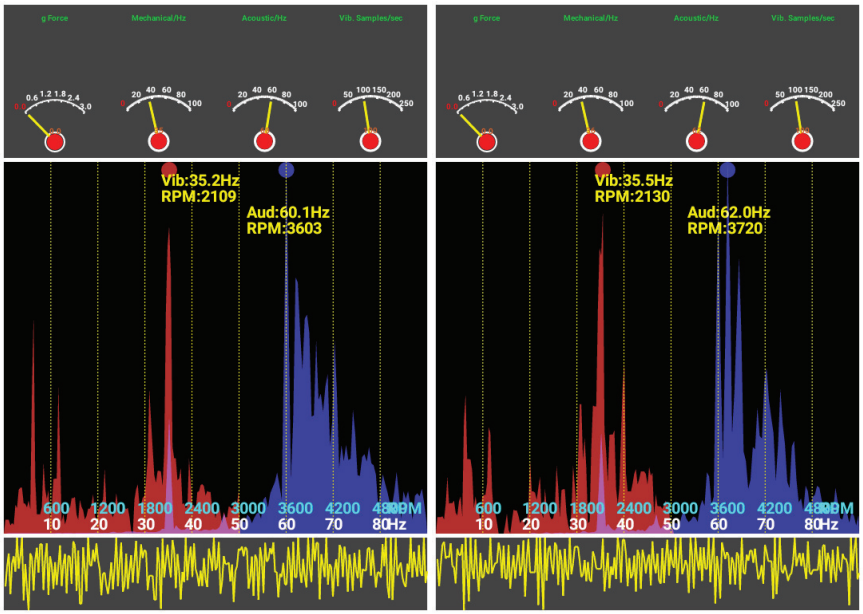


Figura 3.25 Resultados de la aplicación Vibrations de la vibración de una lavadora.

Tabla 3.13 Comparación VibrerApp Vs Vibrations.

EJE	FRECUENCIA (Hz)		
	VibrerApp (aproximado)	**Vibrations (exacto)**	**% Error**
EJE PERPENDICULAR	35.12	35.2	0.22

Los resultados en ambas aplicaciones son muy similares y dado que solo se toma en cuenta el valor del eje perpendicular al dispositivo se realizó el cálculo de error con respecto al valor del eje Z obtenido en la aplicación del proyecto. Sin embargo, se debe destacar que en el archivo que permite guardar de la medición de datos, si se muestra los valores receptados en cada eje y mediante un análisis externo se puede determinar la frecuencia en el resto de las componentes.

Resultados obtenidos de la aplicación VibSensor

Esta aplicación nos muestra resultados similares a los de la aplicación realizada en este proyecto, el valor pico en el dominio del tiempo, denominado PGA, es el valor que se muestra en la variable *Peak Raw*, de igual forma presenta el valor de la frecuencia de muestreo máxima posible por el celular y los valores de frecuencia en cada uno de los ejes con su respectiva amplitud.

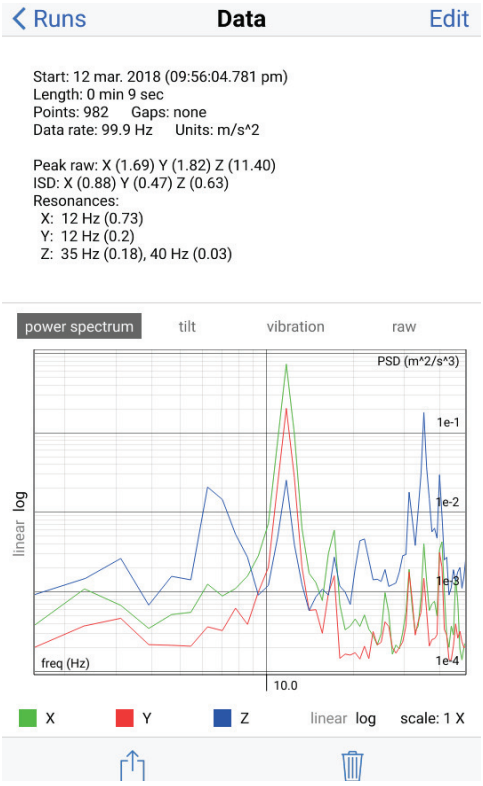


Figura 3.26 Resultados de la aplicación VibSensor de la vibración de una lavadora.

Esta es una de las aplicaciones más completas para el análisis de vibraciones ya que presenta diversas opciones para el procesamiento de la información e incluye los datos necesarios para poder cuantificar las vibraciones que se recepten.

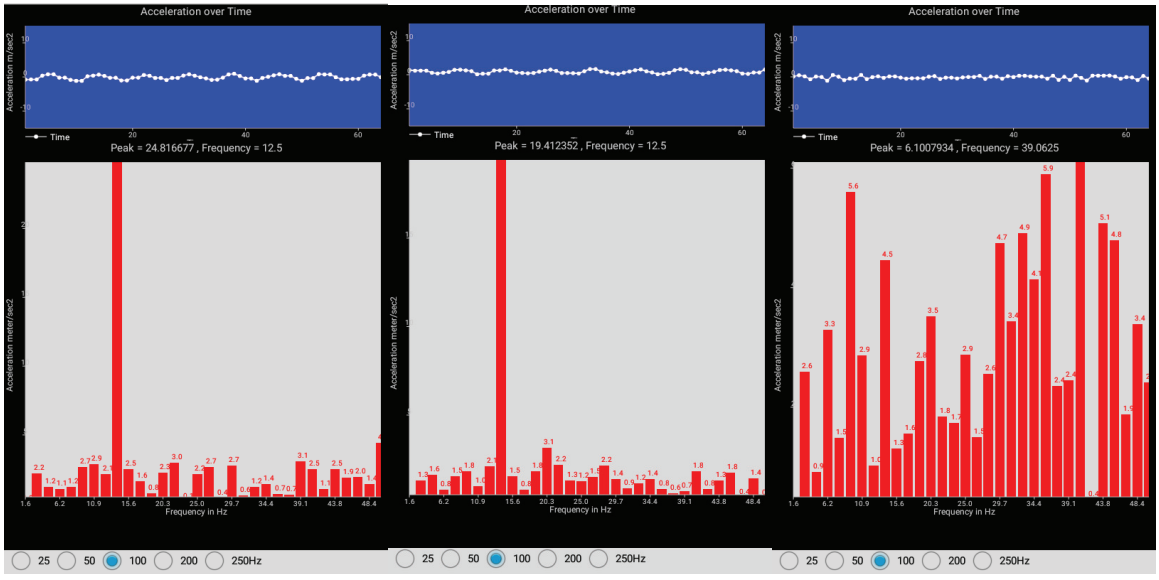
Es una aplicación muy didáctica ya que permite alternar los gráficos de acuerdo a la necesidad que se tenga, muestra gráficos del espectro, del tilt, de las vibraciones en el dominio del tiempo y raw. Tiene la propiedad de almacenamiento de las experimentaciones que se realicen para poder ingresar en cualquier momento y observar los resultados, esta información se guarda en una carpeta creada por la aplicación con su nombre

Tabla 3.14 Comparación de VibrerApp Vs VibSensor.

EJE	PGA (m/s^2)			FRECUENCIA (Hz)		
	VibrerApp (aproximado)	**VibSensor (exacto)**	**% Error**	**VibrerApp (aproximado)**	**VibSensor (exacto)**	**% Error**
X	-1.61	1.69	4.7	11.72	12	2.3
Y	1.70	1.82	6.6	11.72	12	2.3
Z	11.09	11.40	2.7	35.12	35	0.3

Ya que esta aplicación nos brinda los mismos valores que la aplicación desarrollada se puede realizar la comparación de ambos parámetros principales como son: el PGA y la frecuencia predominante del espectro de frecuencias en cada uno de los ejes. Y como se puede observar la diferencia entre ambos resultados no es muy significante y son muy parecidos, por lo que el porcentaje de error es mínimo.

Resultados obtenidos de la aplicación FFT Vibration Analyzer



a) **b)** **c)**

Figura 3.27 Resultados de la aplicación Vibration Analyzer de las vibraciones de una lavadora. a) Eje X. b) Eje Y. c) Eje Z.

Tabla 3.15 Comparación de la aplicación VibrerApp Vs Vibration Analyzer.

EJE	FRECUENCIA (Hz)		
	VibrerApp (aproximado)	**Vibration Analyzer (real)**	**% Error**
X	11.72	12.5	6.2
Y	11.72	12.5	6.2
Z	35.12	39.06	10

Como se observa en la Tabla 3.15 el porcentaje de error es muy aceptable mostrando una gran similaridad en los resultados obtenidos.

3.4.5. Limitaciones de la aplicación VibrerApp

Ciertas limitaciones se presentan debido a las características del sensor que viene instalado en cada uno de los Smartphones, ya que no poseen un sensor universal para todos los telefónos. Estos varían dependiendo de que tan avanzada es su tecnología ya que la frecuencia de muestreo del sensor puede variar entre 100 y 200 Hz. Esto también implica una relación con el costo del dispositivo, debido a que un teléfono de mayor precio poseerá mejores características de hardware instalado en su interior en comparación a uno de bajo costo. Se ha podido incluso visualizar que ciertos *Smartphones* de mayor costo tienen en su interior no solo un sensor acelerómetro, sino varios, ya que ha sido diseñados para brindar mayor realidad virtual en juegos u otras aplicaciones relacionadas. Esto influye en gran manera en la sensibilidad del dispositivo frente a las vibraciones externas.

El problema de aliasing se puede generar si no hay un proceso de filtrado de la señal muestreada para respetar el Teorema de Nyquist. Por esta razón se analizaron los *datasheet* de los acelerómetros de dos *smartphones*. Estos sensores entregan a su salida señales ya digitalizadas y poseen filtros pasabajos con frecuencia de corte configurable. Así mismo, estos sensores también poseen frecuencias de muestreo configurables. Cabe recalcar que las frecuencias de muestreo y frecuencias de corte de los filtros son diferentes en los dos sensores analizados.

Los valores de aceleración que entrega la herramienta de programación Android Studio son digitales y solo se puede seleccionar el período de muestreo (inverso de la frecuencia de muestreo). Se asume que Android Studio configura una frecuencia de corte adecuada dependiendo del período de muestreo que se seleccione y de esta forma no existiría el problema de aliasing. El no poder configurar manualmente la frecuencia de corte del filtro

interno del acelerómetro se considera una limitación de la aplicación desarrollada pero está fuera del control del desarrollador de la aplicación.

De las pruebas de comparación realizadas con acelerómetros de alta gama se pudo observar una limitada sensibilidad de los acelerómetros de los *Smartphone.* Esto era de esperarse ya que los unos son equipos dedicados con alta sensibilidad pero también alto costo, mientras que los *Smartphones* están diseñados para otros fines y el acelerómetro que usualmente posee, no tiene muy buena sensibilidad. Sin embargo, en las pruebas se puede observar que a pesar de una limitada sensibilidad, la aplicación desarrollada para los *Smartphone* es capaz de determinar los valores de las frecuencias predominantes de forma bastante aproximada comparado con los acelerómetros de alta gama.

4. CONCLUSIONES

- El análisis de vibraciones puede brindar grandes ventajas al profesional que requiere realizar un mantenimiento, estudio o registro de eventos suscitados en una maquinaria, estructura o área geográfica. Permitiendo conocer detalles que se encuentran inmersos como información valiosa dentro de una señal de vibraciones.
- El análisis de vibraciones requiere una conversión de la señal obtenida en el dominio del tiempo, al dominio de la frecuencia. Este procedimiento se lo realiza gracias a la Transformada de Fourier. De esta forma se puede obtener la información que guarda una señal de vibraciones, la cual es conocer las frecuencias predominantes de la señal.
- Cuando la frecuencia de excitación a la que vibra una estructura se acerca a su frecuencia de resonancia, se producen grandes amplitudes de vibración de la estructura teniendo el desplazamiento máximo de la estructura. Esto puede provocar grandes afectaciones en la estructura y daños significativos.
- La importancia de conocer los niveles de frecuencias de vibración de un objeto radica en estos conceptos de frecuencias de resonancia y frecuencia natural, ya que cada uno marca los niveles dentro de los cuales son aceptables las amplitudes de vibración. Para prevenir un daño o colapso de la estructura no debe llegar a vibrar a valores aproximados a la frecuencia de resonancia.
- Dentro de las aplicaciones para Ingeniería Mecánica se conoce el término de *mantenimiento predictivo*, el cual tiene como objetivo conocer la condición en la que se encuentra la maquinaria, esto se hace monitoreando las frecuencias de vibración de la misma. Las frecuencias de vibración admisibles vienen detalladas dentro del manual de usuario y garantizan una operación segura y eficiente.
- Una correcta implementación de esta técnica para análisis espectral de las vibraciones aporta una información muy valiosa para un análisis in situ de una cierta eventualidad que se puede suscitar en las áreas de trabajo mencionadas. Permitiendo un diagnóstico temprano de las fallas o efectos nocivos que puedan darse.
- Android Studio es una potente herramienta para el desarrollo de aplicaciones compatibles con este sistema operativo, permitió un manejo adecuado del sensor acelerómetro que viene instalado dentro de los dispositivos, pudiendo explotar las

capacidades del sensor, como manejar la frecuencia de muestreo para la adquisición de datos.

- No existe un método público definido en Android para conocer la tasa de muestreo a la que el marco del sensor está enviando los eventos del sensor a la aplicación y por lo tanto comprobar la verdadera frecuencia de muestreo puede ser complicado; si embargo, se usa las estampas de tiempo que se encuentran relacionadas a los eventos del sensor y así realizar un contador que permite calcular la frecuencia de muestreo.

- De acuerdo con las pruebas realizadas en distintos dispositivos móviles se pudo evidenciar que en la actualidad los *Smartphones* de última generación incluyen ya todos, un acelerómetro triaxial dentro de su circuitería, los cuales tienen una frecuencia de muestreo de entre 100 Hz y 200 Hz. Dispositivos más antiguos pueden portar también un acelerómetro, pero con una frecuencia menor que no es práctico para el análisis de vibraciones.

- De la experimentación, con la aplicación instalada en *Smartphones* con diferente frecuencia de muestreo máxima, se puede ver que dependiendo de esta será el rango de frecuencias predominantes que se pueda visualizar en el espectro. Ya que esto se basa en el teorema de muestreo de Nyquist-Shannon, donde indica que la frecuencia de muestreo debe ser mayor a dos veces la frecuencia máxima y por lo tanto el valor de frecuencia máximo del espectro será la mitad de la frecuencia de muestreo que se seleccione en la aplicación.

- De las pruebas de comparación entre los *Smartphone,* se puede observar que, para la detección de amplitud de la frecuencia predominante, el celular Samsung tiene una mejor sensibilidad comparado al celular Huawei. Por otra parte, con relación al celular Huawei, este presenta la ventaja de que el dispositivo alcanza una frecuencia de muestreo mayor.

- Se debe tomar en cuenta que al receptar los valores del acelerómetro en el dominio del tiempo, la aplicación no elimina la componente de la gravedad, la cual se puede presentar como una componente DC del valor de la gravedad (9.8 m/s^2) en uno de los ejes del acelerómetro dependiendo de la ubicación del dispositivo; esto al pasar al dominio de la frecuencia se observaría como una amplitud bien alta en la frecuencia de 0 Hz (componente DC). Por esta razón para obtener el espectro de frecuencia primero se le resta la media a las señales en el dominio del tiempo y eliminar la componente DC.

- Los valores del PGA obtenidos en la aplicación instalada en diferentes *Smartphones* pueden variar mínimamente uno respecto del otro debido a la sensibilidad de cada uno de ellos; sin embargo, al comparar con los valores obtenidos de los datos receptados en los acelerómetros de mayor gama y dedicados precisamente para el análisis de vibraciones, presentan diferencias notorias que se deben a la precisión de los equipos.

- El acelerómetro de un *Smartphone* a pesar de que no tiene grandes capacidades ya que no fue desarrollado para este tipo de trabajos específicos, posee un desarrollador de aplicaciones Android que permite explotar las características de sus componentes internos y obtener así, una herramienta que brinda resultados muy aproximados a los equipos especializados.

- A pesar de que el acelerómetro de un Smartphone no tiene gran precisión, con el desarrollo de una aplicación Android es posible realizar el procesamiento de los datos del acelerómetro para generar el espectro de frecuencia y analizar las frecuencias predominantes en una vibración mecánica, obteniendo resultados válidos para establecer conclusiones instantáneas de un análisis in situ.

- La practicidad de tener un acelerómetro que recepte las señales de vibraciones incluido en uno de los dispositivos de mayor uso en la actualidad como son los celulares, brinda grandes ventajas con respecto a los acelerómetros especializados, ya que el uso de estos implica un transporte un tanto dificultoso de todo el equipo necesario. La mayoría de los acelerógrafos son de tamaño y peso considerable y sumado a esto, requieren un computador u otro equipo para la recepción de los datos.

- Los acelerómetros utilizados para la etapa de experimentación requieren de un tiempo considerable para la ubicación e instalación del equipo en el sitio que se va a analizar, también un tiempo necesario para que los niveles en los sensores de cada eje se estabilicen después del movimiento ocasionado por su traslado. Esto no sucede con la aplicación Android, que sólo requiere el tiempo para iniciar la aplicación y colocar el celular en la superficie de análisis.

- Para una correcta obtención de los datos es recomendable prestar atención a la superficie sobre la cual se coloque el dispositivo, ya que de esto dependerá los datos que se recepten en cada una de las componentes de los tres ejes. Una superficie nivelada permitirá que los valores se destaquen en una de las tres orientaciones.

- En el caso de necesitar una análisis de un área o estructura específica se recomienda apartar objetos u fuentes de vibraciones que no sean relevantes y que puedan afectar en el resultado de la captura de los datos de la señal de vibraciones de una estructura.
- La aplicación permite la elección de la frecuencia con la que se desea trabajar y aunque para la mayoría de las aplicaciones para las que fue diseñada es suficiente una frecuencia de muestreo de 100 Hz, si se busca aprovechar al máximo las capacidades del *Smartphone,* se recomienda usar la frecuencia máxima de muestreo que permita el dispositivo.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] P. Inc., «Products for USB Sensing and Control,» 20 Abril 2016. [En línea]. Available: http://www.phidgets.com/docs/1043_User_Guide. [Último acceso: 3 Enero 2017].

[2] G. SYSTEMS, «CMG-5TD Digital Accelerograph System,» [En línea]. Available: http://manualzilla.com/doc/5717059/manual-for-the-gural-cmg-5td-strong. [Último acceso: 3 Enero 2017].

[3] B. L. Valle, «UVa Biblioteca Universitaria,» Universidad de Valladolid. Escuela Técnica Superior de Ingenieros de Telecomunicación, Junio 2017. [En línea]. Available: http://uvadoc.uva.es/handle/10324/23113. [Último acceso: 2017].

[4] Grupo de Investigación IMAC , «Ingeniería Mecánica Aplicada y Computacional,» 2006. [En línea]. Available: http://www.imem.unavarra.es/EMyV/pdfdoc/vib/vib_fourier.pdf. [Último acceso: septiembre 2017].

[5] A. Jimenez, «MANTENIMIENTO LA,» 08 Octubre 2013. [En línea]. Available: https://maintenancela.blogspot.com/2013/10/vibraciones-mecanicas-analisis-expectral.html. [Último acceso: 14 Septiembre 2017].

[6] Sinais - Ingeniería de mantenimiento, «Curso básico de análisis de vibraciones,» [En línea]. Available: http://www.sinais.es/Recursos/Curso-vibraciones/fundamentos/magnitudes.html#Figura18. [Último acceso: 14 Septiembre 2017].

[7] Sheldon, «Átomos y bits,» 13 05 2014. [En línea]. Available: http://atomosybits.com/la-fisica-tras-el-acelerometro/. [Último acceso: Septiembre 2017].

[8] M. Arenas Mas, «e-REdING Biblioteca Universidad de Sevilla,» Junio 2008. [En línea]. Available: http://bibing.us.es/proyectos/abreproy/11638. [Último acceso: Julio 2017].

[9] L. E. Asri, «Así funcionan las tripas de tu móvil: el acelerómetro, un sensor que te puede salvar la vida,» *Hoja de Router,* 26 Junio 2014.

[10] E. A. E. Stories, «Engineer Guy,» 2012. [En línea]. Available: http://www.engineerguy.com/elements/. [Último acceso: septiembre 2017].

[11] M. J. Gutierrez., «El androide libre,» El Español, 10 Julio 2014. [En línea]. Available: https://elandroidelibre.elespanol.com/2014/07/cuales-son-y-para-que-sirven-los-sensores-de-nuestros-android.html. [Último acceso: Septiembre 2017].

[12] C. González, J. Hidalgo, J. Manila y P. Aponte, «Slide Share,» Noviembre 2012. [En línea]. Available: https://www.slideshare.net/JuanHidalgo15/presentacin-analisis-de-vibraciones. [Último acceso: Septiembre 2017].

[13] C. S. Flores Rivera, «TodoInfo: Ingeniería Mecánica y más.,» 7 Agosto 2016. [En línea]. Available: http://todoinfodeingenieriamecanicaymas.blogspot.com/2016/08/tipos-de-vibraciones-mecanicas.html. [Último acceso: 14 Junio 2017].

[14] P. Acuña Vigil, «POLIS - CIVITAS BITÁCORA DE URBANISMO Y PLANEAMIENTO,» 29 Septiembre 2015. [En línea]. Available: https://pavsargonauta.wordpress.com/2015/09/29/disipadores-de-energia-en-la-estructura-de-edificios/. [Último acceso: 22 Septiembre 2017].

[15] R. Tangri, S. Jena y S. Roy, «Geospatial World,» Geospatial Media and Communications, 15 Septiembre 2009. [En línea]. Available: https://www.geospatialworld.net/article/earthquake-disaster-management-using-gis-and-probabilistic-risk-assessment/. [Último acceso: Junio 2017].

[16] RENOVETEC, «Energiza,» 2016. [En línea]. Available: http://www.energiza.org/mantenimiento-de-plantas/19-mantenimiento-de-plantas/516-analisis-de-vibraciones-una-tecnologia-clave-del-mantenimiento-predictivo. [Último acceso: Julio 2017].

[17] SINAIS, «Soluciones de monitorización industrial,» 2013. [En línea]. Available: http://www.sinais.es/Recursos/Curso-vibraciones/fundamentos/transformada_fourier.html. [Último acceso: Octubre 2017].

[18] J. Girones, El gran libro de Android, Tercera ed., MARCOMBO S.A., 2013.

[19] DesdeLinux, «Características y cualidades de Android Studio,» 24 Mayo 2016. [En línea]. Available: https://blog.desdelinux.net/caracteristicas-y-cualidades-de-android-studio/. [Último acceso: 2 Noviembre 2017].

[20] Sgoliver, «SGOLIVER.NET,» 12 2014. [En línea]. Available: http://www.sgoliver.net/blog/estructura-de-un-proyecto-android-android-studio/. [Último acceso: Noviembre 2017].

[21] J. A. Rodríguez Gázquez, «Academia Android,» 26 Agosto 2014. [En línea]. Available: https://academiaandroid.com/activity-y-fragments/. [Último acceso: 14 Julio 2017].

[22] T. Cornez y R. Cornez, Android Programming Concepts, Jones & Bartlett Learning, 2015, p. 834.

[23] Android Developers, «Sensors Overview,» [En línea]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html. [Último acceso: 16 Junio 2017].

[24] Dpto. de Ciencia de la Computación e Inteligencia Artificial, «Curso de desarrollo de aplicaciones para Android,» Universidad de Alicante, 26 Enero 2014. [En línea]. Available: http://www.jtech.ua.es/cursos/apuntes/moviles/daa2013/sesion04-apuntes.html. [Último acceso: 14 Octubre 2017].

[25] D. Moisset, «Tutoriales Programación Ya,» 2014. [En línea]. Available: http://www.tutorialesprogramacionya.com/javaya/androidya/detalleconcepto.php?codigo=144&inicio=. [Último acceso: Junio 2017].

[26] Sun Microsystems, «ORACLE - Java Documentation,» 2005. [En línea]. Available: https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html. [Último acceso: Septiembre 2017].

[27] F. Berzal, «Programación en Java,» 2005. [En línea]. Available: http://elvex.ugr.es/decsai/java/pdf/B2-excepciones.pdf. [Último acceso: Septiembre 2017].

[28] L. Ortega, «ANDROIDPIT,» 25 Octubre 2016. [En línea]. Available: https://www.androidpit.es/samsung-galaxy-grand-prime-plus-especificaciones-lanzamiento-precio. [Último acceso: 1 Diciembre 2017].

[29] W. Meza, «ANDROID PE,» 22 Abril 2017. [En línea]. Available: http://www.droidperu.com/2017/04/precio-comprar-huawei-nova-lite-p9-lite-2017-peru-specs-android-celular.html. [Último acceso: Noviembre 2017].

[30] GURALP SYSTEMS, «GURALP.COM,» [En línea]. Available: https://www.guralp.com/documents/DAS-050-0006.pdf. [Último acceso: Noviembre 2017].

[31] Phidgets Inc., «Phidget Support,» 8 Noviembre 2017. [En línea]. Available: https://www.phidgets.com/docs/1043_User_Guide#Getting_Started. [Último acceso: 18 Noviembre 2017].

[32] Trimble Infrastructure, «Trimble Transforming the way the world works,» [En línea]. Available: http://www.trimble.com/Infrastructure/Trimble-REF-TEK-160-03.aspx. [Último acceso: 1 Diciembre 2017].

[33] Universidad de Málaga Departamento de Física Aplicada II, «Teoría de Errores,» 2003. [En línea]. Available: http://webpersonal.uma.es/~JMPEULA/teoria_de_errores.html. [Último acceso: Enero 2018].

[34] M. A. Aguilar Suarez, «Instituto Tecnológico de Tuxtla Gutiérrez,» Departamento de Metal-Mecánica , 31 enero 2012. [En línea]. Available: https://sites.google.com/site/metalnumericos/home/unidad-1/1-2-tipos-de-errores-error-absoluto-error-relativo-error-porcentual-errores-de-redondeo-y-truncamiento. [Último acceso: Enero 2018].

[35] Sun Microsystems, «ORACLE - Java Documentation,» 2005. [En línea]. Available: https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html. [Último acceso: Septiembre 2017].

6. ANEXOS

ANEXO I. Código de las actividades desarrolladas en Android Studio.

ANEXO II. Instrucciones para adición de la librería GraphView.

ANEXO III. Clases Complex.java y FFT.java utilizadas.

ANEXO IV. Scripts de Matlab utilizados para el procesamiento de los resultados.

ANEXO V. Espectrograma del equipo Guralp Systems.

ANEXO I

SPLASHACTIVITY

Código de la parte gráfica de la primera actividad de la aplicación, el diseño de esta actividad se encuentra en el archivo **activity_splash.xml.**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#B5C2FC">

    <ImageView
        android:id="@+id/ivSplash"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:src="@drawable/splashimagen" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="78dp"
        android:text="Bienvenidos al analizador de vibraciones"
        android:textAlignment="center"
        android:textColor="#4720E7"
        android:textColorHint="?android:attr/colorControlActivated"
        android:textSize="24dp"
        android:textStyle="bold|italic" />

</RelativeLayout>
```

Código de la parte lógica de la actividad, esto se define dentro del archivo **SplashActivity.java**, en el se define las funciones que hará esta actividad.

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;
```

```
public class SplashActivity extends Activity {
    private final int DURACION_SPLASH = 2000;//Duracion de la pantalla
splash 2 segundos

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(new Runnable(){
            public void run(){

                //Enlace con la Actividad principal para que sea lanzada
después de trascurrido el retardo del SplashActivity
                Intent intent = new Intent(SplashActivity.this,
MainActivity.class);
                startActivity(intent);
                finish();
            };
        }, DURACION_SPLASH);
    }
}
```

MAINACTIVITY

Código de la parte gráfica de la segunda actividad de la aplicación, el diseño de esta actividad se encuentra en el archivo **content_main.xml,** este archivo es llamado a través del **activity_main.xml.**

- **Activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="ec.edu.epn.lopez.giovi.com.vibrerapp.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />


</android.support.design.widget.CoordinatorLayout>
```

- **Content_main.xml**

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:orientation="vertical"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:showIn="@layout/activity_main"
        xmlns:android="http://schemas.android.com/apk/res/android">

        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
            android:orientation="vertical">


            <TextView
                android:id="@+id/texto_datos"
                android:layout_width="match_parent"
                android:layout_height="34dp"
                android:layout_marginTop="18dp"
                android:text=" "
                android:textColor="@android:color/black" />

            <RadioGroup
                android:id="@+id/Radiogrupo1"
                android:layout_width="match_parent"
                android:layout_height="41dp"
                android:layout_marginTop="8dp"
                android:orientation="horizontal">

                <RadioButton
                    android:id="@+id/optfrecuencia1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_below="@+id/texto_ejes"
                    android:text="Frecuencia= 100 Hz            " />

                <RadioButton
                    android:id="@+id/optfrecuencia2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_below="@+id/optfrecuencia1"
                    android:text="Frecuencia Máxima" />
            </RadioGroup>

            <TextView
                android:id="@+id/texto_ejes"
                android:layout_width="match_parent"
                android:layout_height="81dp"
                android:layout_marginTop="8dp"
                android:text=" "
                android:textColor="@android:color/black" />

            <com.jjoe64.graphview.GraphView
                android:id="@+id/grafico"
                android:layout_width="match_parent"
                android:layout_height="298dp"
                android:layout_marginBottom="8dp"
                android:layout_marginTop="32dp" />

        </LinearLayout>

    </ScrollView>

</LinearLayout>
```

Código de la parte lógica de la actividad, esto se define dentro del archivo **MainActivity.java**, en el se define las funciones que hará esta actividad.

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

import android.content.Intent;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.LegendRenderer;
import com.jjoe64.graphview.Viewport;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

public class MainActivity extends AppCompatActivity implements
SensorEventListener, RadioGroup.OnCheckedChangeListener{

    //DECLARACIÓN DE VARIABLES Y OBJETOS

    float ejeX, ejeY, ejeZ;
    float MinDelay;          // Mínimo retardo del sensor acelerometro
    float Max_Range;         //Máximo rango de potencia del sensor
    float Fmax;               //Frecuencia máxima de muestreo del
dispositivo
    float Fs;                 //Frecuencia de muestreo

    TextView texto_datos, texto_ejes;

    RadioGroup Radiogrupo1;
    RadioButton optfrecuencia1, optfrecuencia2;

    //----Variables para manejo del sensor.....//

    SensorManager sensorManager;  //Variable de objeto de clase Gestor de
sensores Sensor Manager
    private Sensor acelerometro;  // Declaración del sensor a manejar

    //----Variables para contador de frecuencia----//

    private long last_update = 0;
    int cont = 0;

    //----Variables para Graficar datos de los ejes ----//

    private LineGraphSeries<DataPoint> dataseriesX, dataseriesY,
dataseriesZ;
    private float ejeXgrafico = 0;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        // ENLAZAR VARIABLES CON OBJETOS

        texto_datos = (TextView) findViewById(R.id.texto_datos);
        texto_ejes = (TextView) findViewById(R.id.texto_ejes);

        GraphView grafico = (GraphView) findViewById(R.id.grafico); //
obtener una instancia graph view



        // Conectar la clase Sensor Manager con los servicios

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        //Enlazar con el tipo de sensor (seleccionar acelerometro)

        acelerometro =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        //Obtener las carácteristicas del sensor

        MinDelay = acelerometro.getMinDelay();
        Max_Range = acelerometro.getMaximumRange();

        Fmax = (1 / MinDelay) * (1000000);   //Obtención de la frecuencia
máxima de muestreo

        //Impresión de las características del sensor

       // texto_datos.append("Min Delay: " + MinDelay + "\n");
        texto_datos.append("Frecuencia máx de muestreo del dispositivo: "
+ Fmax + "  Hz" + "\n");
        texto_datos.append("\n Seleccione la frecuencia que desea
utilizar: \n" );
        //text.append( "Max Range: " + Max_Range + "\n");

        // ----GRÁFICO ----//

        dataseriesX = new LineGraphSeries<DataPoint>();
        dataseriesY = new LineGraphSeries<DataPoint>();
        dataseriesZ = new LineGraphSeries<DataPoint>();
        grafico.addSeries(dataseriesX);
        grafico.addSeries(dataseriesY);
        grafico.addSeries(dataseriesZ);

        //Propiedades del gráfico

        grafico.getGridLabelRenderer().setNumHorizontalLabels(4);
        grafico.getGridLabelRenderer().setHorizontalLabelsVisible(false);
        grafico.getGridLabelRenderer().setVerticalAxisTitle("Aceleración
(m/s^2)");
```

```
        Viewport viewport = grafico.getViewport();
        viewport.setYAxisBoundsManual(true);
        viewport.setMinY(-20);
        viewport.setMaxY(20);

        viewport.setXAxisBoundsManual(true);
        viewport.setMinX(0);
        viewport.setMaxX(800);

        viewport.setScalable(true);
        viewport.setScrollable(true);
        viewport.setScalableY(false);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        //noinspection SimplifiableIfStatement

        switch (item.getItemId()) {
            case R.id.grabar_datos:

                Intent intent = new Intent(MainActivity.this,
GrabarActivity.class);
                intent.putExtra("Frecuencia", Fs);

                startActivity(intent);
                return true;

            case R.id.info_app:
                Intent intentinfo = new Intent(MainActivity.this,
InfoActivity.class);
                startActivity(intentinfo);

                return true;

            default:
                return super.onOptionsItemSelected(item);
        }
    }

    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {

        ejeX = sensorEvent.values[0];
        ejeY = sensorEvent.values[1];
        ejeZ = sensorEvent.values[2];
```

```
        // Para monitorear la frecuencia de  muestreo del acelerometro

               long current_time = sensorEvent.timestamp;
        long difference_time = current_time - last_update;
        if (difference_time > 10E8 && cont != 0) {
            last_update = current_time;
            cont = 0;            // Cuenta los eventos que pasan por
sensor, mindiendo la frecuencia de muestreo
        }
        cont++;

        texto_ejes.setText("");
        texto_ejes.append("El valor de X:  " + ejeX + "\n" + "El valor de
Y:  " + ejeY + "\n" + "El valor de Z:  " + ejeZ + "\n " );
       // texto_ejes.append("\n" + "Frecuencia: " + cont + " Hz" );

        // ----LLAMAR A LA FUNCIÓN QUE AÑADE LOS DATOS AL GRAFICO----//

        addEntry();          // Añade los datos de cada uno de los ejes
para poder graficar


    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }

    @Override
    protected void onResume() {
        super.onResume();

        RadioGroup Radiogrupo1 = (RadioGroup)
findViewById(R.id.Radiogrupo1);
        Radiogrupo1.setOnCheckedChangeListener(this);// Se asigna el
evento onclick o touch de los botones

    }

    private void addEntry() {

        dataseriesX.appendData(new DataPoint(ejeXgrafico++ , ejeX ), true
, 1000);
        dataseriesY.appendData(new DataPoint(ejeXgrafico++ , ejeY ), true
, 1000);
        dataseriesZ.appendData(new DataPoint(ejeXgrafico++ , ejeZ ), true
, 1000);
//
        dataseriesX.setTitle("EJE X");
        dataseriesY.setTitle("EJE Y");
        dataseriesZ.setTitle("EJE Z");

        dataseriesX.setColor(Color.GREEN);
        dataseriesY.setColor(Color.RED);
        dataseriesZ.setColor(Color.BLUE);

        GraphView grafico = (GraphView) findViewById(R.id.grafico);
        grafico.getLegendRenderer().setVisible(true);
```

```
grafico.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.TOP);

    }

    @Override
    protected void onPause() {
        super.onPause();
    }


    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int checkedId) {

        switch (checkedId){
            case R.id.optfrecuencia1:

                sensorManager.registerListener(this, acelerometro, 10000
);
                Fs = 10000;
                break;

            case R.id.optfrecuencia2:

                sensorManager.registerListener(this, acelerometro,
SensorManager.SENSOR_DELAY_FASTEST);
                // Fmax = SensorManager.SENSOR_DELAY_FASTEST;
                Fs = MinDelay;

                break;}



    }
}
```

GRABAR ACTIVITY

Código de la parte gráfica de la tercera actividad de la aplicación, el diseño de esta actividad se encuentra en el archivo **activity_grabar.xml.**

```
<LinearLayout xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <EditText
                android:id="@+id/nombre_archivo"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:ems="10"
                android:hint="Ingrese nombre del archivo"
                android:inputType="textPersonName"
                android:maxLength="15"
                android:singleLine="true" />

            <EditText
                android:id="@+id/tiempo"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:digits="1234567890"
                android:ems="10"
                android:hint="Ingrese tiempo para grabar (5-60) seg"
                android:inputType="number"
                android:maxLength="2"
                android:singleLine="true" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">

                <Button
                    android:id="@+id/boton_grabar"
                    android:layout_width="140dp"
                    android:layout_height="wrap_content"
                    android:layout_gravity="left"
                    android:drawableLeft="@drawable/btn_timer"
                    android:onClick="grabar"
                    android:text="Grabar"
                    android:textSize="20sp" />

                <Button
                    android:id="@+id/boton_fft"
```

```
                android:layout_width="139dp"
                android:layout_height="wrap_content"
                android:layout_gravity="right"
                android:layout_marginLeft="85dp"
                android:drawableLeft="@drawable/btn_fourier"
                android:drawablePadding="8dp"
                android:onClick="Fourier"
                android:text="FFT"
                android:textSize="20sp"/>
        </LinearLayout>

        <TextView
            android:id="@+id/texto_contador"
            android:layout_width="match_parent"
            android:layout_height="38dp"
            android:gravity="center_vertical"
            android:textAlignment="center"
            android:textColor="@android:color/darker_gray"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/texto_datos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="#041166" />

        <com.jjoe64.graphview.GraphView
            android:id="@+id/grafico_fft"
            android:layout_width="match_parent"
            android:layout_height="300dp"
            android:visibility="invisible" />

        <TextView
            android:id="@+id/texto_fourier"
            android:layout_width="match_parent"
            android:layout_height="71dp"
            android:textColor="#041166"
            android:textSize="18sp" />

        <Button
            android:id="@+id/boton_reset"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/holo_red_dark"
            android:onClick="reset"
            android:text="RESET"
            android:textColor="@android:color/background_light"
            android:textSize="20sp"
            android:visibility="invisible" />


    </LinearLayout>

</ScrollView>

</LinearLayout>
```

Código de la parte lógica de la actividad, esto se define dentro del archivo **GrabarActivity.java**, en el se define las funciones que hará esta actividad.

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Environment;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.LegendRenderer;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

import static ec.edu.epn.lopez.giovi.com.vibrerapp.FFT.fft;


public class GrabarActivity extends AppCompatActivity implements
SensorEventListener{
    //Variables Acelerometro
    SensorManager sensorManager;
    private Sensor acelerometro;

    double ejex,ejey,ejez;
    int contador;

    //Variable para frecuencia
    float frecibida,Fs;
    String str_fimp,datotiempo,datonombre,nom_archivo;
    String str_fecha,str_hora;
```

```
    //Variables para trabajar
    TextView texto_contador,texto_datos;
    EditText nombre_archivo,tiempo;
    Button boton_grabar, boton_fft, boton_reset;

    //Variable para obtener el tiempo del temporizador
    int temp_ingresado,cont2=0;

    //Variables para grabar los datos del acelerometro en arreglos
    int n = 8192, numdemuestras;
    double [] vector_ejex=new double[n];
    double [] vector_ejey=new double[n];
    double [] vector_ejez=new double[n];

    //Variable para obtner el vector tiempo
    int muestrasaux;
    double []cadenatemp=new double[n];

    //Variable booleana para controlar el estado del boton grabar
    boolean btngrabar=false,bGrabando=false;

    //Variables para el archivo de texto
    OutputStreamWriter osw;
    private long last_update=0,start_time;

    //Variables para la FFT
    TextView texto_fourier;

    //Inicializar el gráfico
    LineGraphSeries<DataPoint>seriesx,seriesy,seriesz;


    //Variable para permisos
    private static final int
MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE = 1 ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grabar);

        //Recibir el dato de frecuencia
        Bundle bundle=this.getIntent().getExtras();
        frecibida=bundle.getFloat("Frecuencia");
        Fs=(1/frecibida)*1000000;
        str_fimp=String.valueOf(Fs);

        //Enlazar objetos
        texto_contador=(TextView)(findViewById(R.id.texto_contador));
        texto_datos=(TextView)(findViewById(R.id.texto_datos));

        nombre_archivo=(EditText)(findViewById(R.id.nombre_archivo));
        tiempo=(EditText)(findViewById(R.id.tiempo));

        boton_grabar=(Button)(findViewById(R.id.boton_grabar));
        boton_fft=(Button)(findViewById(R.id.boton_fft));
        boton_reset=(Button)(findViewById(R.id.boton_reset));

        sensorManager=(SensorManager) getSystemService(SENSOR_SERVICE);
```

```
acelerometro=sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager.registerListener(this,acelerometro,(int)frecibida);

        texto_fourier=(TextView)(findViewById(R.id.texto_fourier));

        //Estados de botones
        boton_grabar.setEnabled(true);
        boton_fft.setEnabled(false);

        //Inicializar los vectores con ceros
        for(int i=0;i<n;i++){
            vector_ejex[i]=0;
            vector_ejey[i]=0;
            vector_ejez[i]=0;
        }

        //PERMISOS
        int permissionCheck = ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE);
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {

            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)) {

            } else {

                ActivityCompat.requestPermissions(this,
                        new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE);
            }
        }

    }


    //PERMISOS
    @Override
    public void onRequestPermissionsResult(int requestCode,String
permissions[], int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE: {
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

                } else {

                }
                return;
            }
        }
    }

    //Metodo onClick para el boton grabar
    public void grabar (View v){
```

```
        numdemuestras=0;
        contador=0;
        muestrasaux=0;
        nombre_archivo.setEnabled(false);
        tiempo.setEnabled(false);
        //Cambiar variable booleana para activar la grabacion
        btngrabar=true;

        //Obtener el valor del edit text del campo nombre
        datonombre=nombre_archivo.getText().toString();

        //Obtener el valor del edit text del campo tiempo
        datotiempo=tiempo.getText().toString(); //Cadena de caracteres
        temp_ingresado=Integer.parseInt(datotiempo); //Convertir a entero
        //Ocultar el teclado
        InputMethodManager
imm=(InputMethodManager)getSystemService(INPUT_METHOD_SERVICE);
        imm.hideSoftInputFromWindow(v.getWindowToken(),0);

        //DATOS PARA EMPEZAR A CREAR EL ARCHIVO DE TEXTO
        nom_archivo=datonombre+".csv";

        SimpleDateFormat formato_hora = new SimpleDateFormat("HH:mm:ss");
        Date horaActual = Calendar.getInstance().getTime();
        str_hora = formato_hora.format(horaActual);

        final Calendar fecha = new GregorianCalendar();
        Date fechaActual = fecha.getTime();
        SimpleDateFormat formato_fecha = new SimpleDateFormat("yyyy-MM-
dd");
        str_fecha = formato_fecha.format(fechaActual);

        //Metodo para tomar los errores producidos
        try {
            File tarjeta = new
File(Environment.getExternalStorageDirectory(),"VibrerApp");
            if(!tarjeta.exists()){
                //Crea el directorio en caso de no existir antes
                tarjeta.mkdirs();
            }

Toast.makeText(this,tarjeta.getAbsolutePath(),Toast.LENGTH_LONG).show();

            //Creo el archivo en el directorio creado anteriormente
            File file = new File (tarjeta.getAbsolutePath(),nom_archivo);
            osw = new OutputStreamWriter(new FileOutputStream(file));
            bGrabando=true;
            //Escribir los campos en el archivo de texto
            osw.write("Fecha: "+str_fecha+"\n");
            osw.write("Hora de Inicio: "+str_hora+"\n");
            osw.write("Frecuencia: "+str_fimp+"\n");
            osw.write("\t t \t\t X  \t\t  Y \t\t  Z \n");

            if(contador==1){
                osw.flush();
                osw.close();
                contador=0;
            }

        }catch (IOException ioe){
```

```
            Toast.makeText(this, "No se pudo grabar",
Toast.LENGTH_SHORT).show();

        }


        if(datonombre.equals("")){
            Toast.makeText(this,"Campos
VACIOS!!!",Toast.LENGTH_LONG).show();
        }else {

            if (temp_ingresado >= 5 && temp_ingresado <= 60) {
                boton_grabar.setEnabled(false);
                temp_ingresado = temp_ingresado * 1000;

                //Crear el temporizador
                new CountDownTimer(temp_ingresado, 1000) {

                    @Override
                    public void onTick(long l) {
                        texto_contador.setText("Tiempo Restante: " + l /
1000);
                    }

                    @Override
                    public void onFinish() {
                        boton_fft.setEnabled(true);
                        texto_contador.setText("Fin temporizador -
Grabacion Exitosa");
                        bGrabando=false;
                        btngrabar=false;
                        contador=1;

                        //Imprimir datos
                        texto_datos.setText(String.format("Inicio de
grabacion: %s %s",str_fecha,str_hora));
                        texto_datos.append("\nLa frecuencia de muestreo
es: "+str_fimp);

                        //ENCONTRAR PGA
                        //Variables
                        double maximox,maximoy,maximoz;
                        int postx,posty,postz;

                        //VALOR ABSOLUTO Y CONSERVACION DEL SIGNO

                        //Vectores para obtener el PGA verdadero
                        double [] pga_ejex=new double[n];
                        double [] pga_ejey=new double[n];
                        double [] pga_ejez=new double[n];

                        for(int i=0;i<numdemuestras;i++){
                            pga_ejex[i]=Math.abs(vector_ejex[i]);
                            pga_ejey[i]=Math.abs(vector_ejey[i]);
                            pga_ejez[i]=Math.abs(vector_ejez[i]);
                        }

                        DateFormat df=new SimpleDateFormat("HH:mm:ss");
                        //EJE X
                        maximox=pga_ejex[0];
```

```
                        postx=0;
                        for(int i=0;i<numdemuestras;i++){
                            if(pga_ejex[i]>maximox){
                               // maximox=vector_ejex[i];
                                maximox=pga_ejex[i];
                                postx=i;
                            }
                        }
                        long lx =(new
Double(cadenatemp[postx]).longValue());
                        Date horax=new Date(lx);
                        String salidax =df.format(horax);

                        //EJE Y
                        maximoy=pga_ejey[0];
                        posty=0;
                        for(int i=0;i<numdemuestras;i++){
                            if(pga_ejey[i]>maximoy){
                                //maximoy=vector_ejey[i];
                                maximoy=pga_ejey[i];
                                posty=i;
                            }
                        }
                        long ly =(new
Double(cadenatemp[posty]).longValue());
                        Date horay=new Date(ly);
                        String saliday =df.format(horay);
                        //EJE Z
                        maximoz=pga_ejez[0];
                        postz=0;
                        for(int i=0;i<numdemuestras;i++){
                            if(pga_ejez[i]>maximoz){
                                //maximoz=vector_ejez[i];
                                maximoz=pga_ejez[i];
                                postz=i;
                            }
                        }
                        long lz =(new
Double(cadenatemp[postz]).longValue());
                        Date horaz=new Date(lz);
                        String salidaz =df.format(horaz);

                        //Imprimir resultados
                        DecimalFormat decimales=new
DecimalFormat("0.00");
                        texto_datos.append("\nPGA EJE X:
"+decimales.format(vector_ejex[postx])+" en: "+salidax+"\nPGA EJE Y:
"+decimales.format(vector_ejey[posty])+" en: "+saliday+"\nPGA EJE Z:
"+decimales.format(vector_ejez[postz])+" en: "+salidaz);


                    }
                }.start();
            } else {
                final String text = "Campo de tiempo incorrecto !!";
                Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
                tiempo.setText("");
            }
        }
```

```
}

//Metodo onclick para el boton fourier
public void Fourier(View v){

    boton_fft.setEnabled(false);

    //Variables para obtener la media
    double mediax = 0;
    double mediay = 0;
    double mediaz = 0;
    double sumax = 0;
    double sumay = 0;
    double sumaz = 0;

    for(int i=0;i<=numdemuestras;i++){
        sumax=sumax+vector_ejex[i];
        sumay=sumay+vector_ejey[i];
        sumaz=sumaz+vector_ejez[i];
    }
    mediax=sumax/numdemuestras;
    mediay=sumay/numdemuestras;
    mediaz=sumaz/numdemuestras;

    for(int i=0;i<=numdemuestras;i++){
        vector_ejex[i]=vector_ejex[i]-mediax;
        vector_ejey[i]=vector_ejey[i]-mediay;
        vector_ejez[i]=vector_ejez[i]-mediaz;
    }

    //Convertir los datos a complejos para aplicar fourier
    Complex[] vecxcomplejo=new Complex[n];
    Complex[] vecycomplejo=new Complex[n];
    Complex[] veczcomplejo=new Complex[n];
    for(int i=0;i<n;i++){
        vecxcomplejo[i]= new Complex(vector_ejex[i],0);
        vecycomplejo[i]= new Complex(vector_ejey[i],0);
        veczcomplejo[i]= new Complex(vector_ejez[i],0);
    }

    //Aplicar la clase FFT
    Complex[] vectorxfourier=fft(vecxcomplejo);
    Complex[] vectoryfourier=fft(vecycomplejo);
    Complex[] vectorzfourier=fft(veczcomplejo);

    //Empezar a crear el vector de potencia-EJE Y
    double [] potencia2x=new double[n];
    double [] potencia2y=new double[n];
    double [] potencia2z=new double[n];

    for(int i=0;i<n;i++){
        potencia2x[i]=(vectorxfourier[i].abs())/n;
        potencia2y[i]=(vectoryfourier[i].abs())/n;
        potencia2z[i]=(vectorzfourier[i].abs())/n;
    }

    //GRAFICO
    //Modificar la potencia para adaptar a la mitad del espectro
    int aux=(n/2)+1;
    double [] potencia1x= new double[aux];
```

```
        double [] potencia1y= new double[aux];
        double [] potencia1z= new double[aux];

        double [] f=new double[aux];
        f[0]=0;
        double[] ejefreq=new double [aux];

        //Inicializar el grafico
        GraphView
graficofourier=(GraphView)findViewById(R.id.grafico_fft);
        seriesx=new LineGraphSeries<DataPoint>();
        seriesy=new LineGraphSeries<DataPoint>();
        seriesz=new LineGraphSeries<DataPoint>();

        graficofourier.removeAllSeries();
        graficofourier.setVisibility(View.VISIBLE);

        for(int i=1;i<aux;i++){
            potencia1x[i]=2*potencia2x[i];
            potencia1y[i]=2*potencia2y[i];
            potencia1z[i]=2*potencia2z[i];

            //Eje frecuencia - eje x
            f[i]=f[i-1]+1;
            ejefreq[i]=(f[i]/n)*Fs;
            seriesx.appendData(new
DataPoint(ejefreq[i],potencia1x[i]),true,aux);
            seriesy.appendData(new
DataPoint(ejefreq[i],potencia1y[i]),true,aux);
            seriesz.appendData(new
DataPoint(ejefreq[i],potencia1z[i]),true,aux);
        }

        //ENCONTRAR EL MAYOR VALOR DE POTENCIA
        //Variables para almacenar los datos auxiliares
        double potenciamayorx,potmaxy,potmaxz;
        int posicion,posy,posz;

        //Procedimiento EJEX
        potenciamayorx=potencia1x[0];
        posicion=0;
        for (int i=1;i<aux;i++){
            if(potencia1x[i]>potenciamayorx){
                potenciamayorx=potencia1x[i];
                posicion=i;
            }
        }
        //Procedimiento EJEY
        potmaxy=potencia1y[0];
        posy=0;
        for (int i=1;i<aux;i++){
            if(potencia1y[i]>potmaxy){
                potmaxy=potencia1y[i];
                posy=i;
            }
        }
        //Procedimiento EJEZ
        potmaxz=potencia1z[0];
        posz=0;
        for (int i=1;i<aux;i++){
```

```
            if(potencia1z[i]>potmaxz){
                potmaxz=potencia1z[i];
                posz=i;
            }
        }


        DecimalFormat decimales=new DecimalFormat("0.00");

        texto_fourier.append("Maximo EJE X:
"+decimales.format(potenciamayorx)+" y esta en:
"+decimales.format(ejefreq[posicion])+" Hz \n");
        texto_fourier.append("Maximo EJE Y:
"+decimales.format(potmaxy)+" y esta en:
"+decimales.format(ejefreq[posy])+" Hz \n");
        texto_fourier.append("Maximo EJE Z:
"+decimales.format(potmaxz)+" y esta en:
"+decimales.format(ejefreq[posz])+" Hz ");
        //varios.append("numero de muestras: " + numdemuestras);


        //Estilo del grafico
        //Scrooll and Zoom
        //EJE X MANUAL
        graficofourier.getViewport().setYAxisBoundsManual(true);
        graficofourier.getViewport().setMinY(0);
        graficofourier.getViewport().setMaxY(1);

        graficofourier.getViewport().setXAxisBoundsManual(true);
        graficofourier.getViewport().setMinX(0);
        graficofourier.getViewport().setMaxX(Fs/2);

        //Habilitar la escala
        graficofourier.getViewport().setScalable(true);
        //graficofourier.getViewport().setScalableY(true);

        seriesx.setTitle("EJE X");
        seriesx.setColor(Color.GREEN);
        seriesy.setTitle("EJE Y");
        seriesy.setColor(Color.RED);
        seriesz.setTitle("EJE Z");
        seriesz.setColor(Color.BLUE);
        graficofourier.addSeries(seriesx);
        graficofourier.addSeries(seriesy);
        graficofourier.addSeries(seriesz);

        graficofourier.getLegendRenderer().setVisible(true);

graficofourier.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.TO
P);

        //Titulos

graficofourier.getGridLabelRenderer().setVerticalAxisTitle("Amplitud ");

graficofourier.getGridLabelRenderer().setHorizontalAxisTitle("Frecuencia
Hz");
```

```
        //RESET
        boton_reset.setVisibility(View.VISIBLE);

    }


    //RESET
    public void reset(View v){
        nombre_archivo.setEnabled(true);
        tiempo.setEnabled(true);

        //Setear a cero los campos
        nombre_archivo.setText("");
        tiempo.setText("");


        boton_grabar.setEnabled(true);
        boton_fft.setEnabled(false);
        nombre_archivo.requestFocus();    //Ubicación del cursor,
solicitar enfoque de cursor
        texto_contador.setText("");
        texto_datos.setText("");
        texto_fourier.setText("");


        for(int i=0;i<n;i++){
            vector_ejex[i]=0;
            vector_ejey[i]=0;
            vector_ejez[i]=0;
            cadenatemp[i]=0;
        }
        numdemuestras=0;          //fourier
        muestrasaux=0;            //Muestras para el vector cadena de
tiempo

        GraphView
graficofourier=(GraphView)findViewById(R.id.grafico_fft);
        graficofourier.removeAllSeries();
        graficofourier.setVisibility(View.INVISIBLE);
        boton_reset.setVisibility(View.INVISIBLE);


    }

    //METODOS PARA EL ACELEROMETRO
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {

        ejex=sensorEvent.values[0];
        ejey=sensorEvent.values[1];
        ejez=sensorEvent.values[2];

        //Variable booleana para grabar en un array
        if(btngrabar){
            vector_ejex[numdemuestras]=ejex;
            vector_ejey[numdemuestras]=ejey;
            vector_ejez[numdemuestras]=ejez;
            numdemuestras++;
        }
```

```
        if(bGrabando){
            try{
                osw.write (String.format ( "%20.6f %10.5f %10.5f %10.5f
\n",  (sensorEvent.timestamp * 1.0E-9) , ejex, ejey, ejez) );
                cadenatemp[muestrasaux]=System.currentTimeMillis();
                muestrasaux++;

            }catch (IOException e){
                e.printStackTrace();
                Toast.makeText(this, "No se pudo grabar sensor",
Toast.LENGTH_SHORT).show();
            }
        }

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }
}
```

INFOACTIVITY

Código de la parte gráfica de la cuarta actividad de la aplicación, el diseño de esta actividad se encuentra en el archivo **activity_info.xml.**

```
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Orientación de los ejes sobre el
dispositivo: "
                android:textSize="15dp" />

            <ImageView
                android:id="@+id/imageView"
                android:layout_width="283dp"
                android:layout_height="254dp"

android:layout_gravity="center_vertical|center_horizontal"
                app:srcCompat="@drawable/ejes_info" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Informacion Adicional: " />

            <TextView
                android:id="@+id/textView2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center_vertical|center_horizontal"
                android:text="DESARROLLO DE UNA APLICACIÓN ANDROID PARA
EL ANÁLISIS DE VIBRACIONES USANDO EL ACELERÓMETRO DE UN SMARTPHONE"
                android:textAlignment="center"
                android:textColor="@android:color/black" />

            <TextView
                android:id="@+id/textView3"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Desarrollado por: " />

            <LinearLayout
```

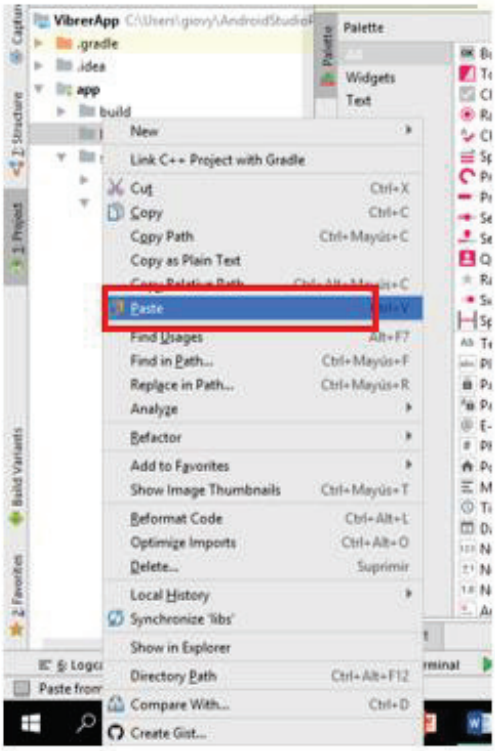
```
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">

                <ImageView
                    android:id="@+id/imageView2"
                    android:layout_width="117dp"
                    android:layout_height="140dp"
                    android:layout_gravity="left"
                    app:srcCompat="@drawable/img_gio" />

                <TextView
                    android:id="@+id/textView4"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:gravity="center_vertical|center_horizontal"
                    android:text="ESCUELA POLITÉCNICA NACIONAL Giovanna
Lucía López Armas giovylu19@hotmail.com Director de proyecto: Dr. Diego
Javier Reinoso "
                    android:textAlignment="center"
                    android:textColor="@android:color/black" />
            </LinearLayout>



        </LinearLayout>

    </ScrollView>

</LinearLayout>
```

Código de la parte lógica de la actividad, esto se define dentro del archivo **InfoActivity.java**, en el se define las funciones que hará esta actividad.

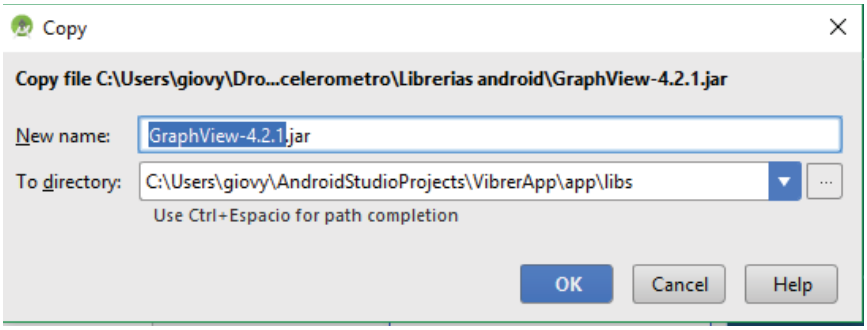
```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class InfoActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_info);
    }
}
```

ANEXO II

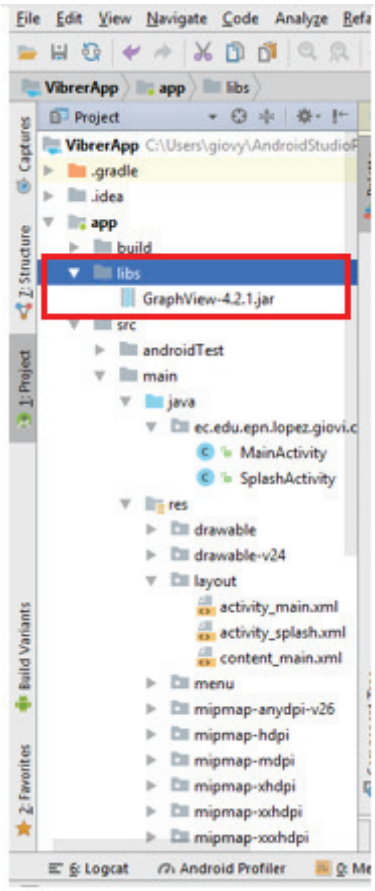
La librería GraphView permite añadir gráficos en tiempo real dentro de una aplicación Android. Continuando con el procedimiento indicado en capítulo 2, una vez que se tiene ya descargada la librería con la extensión .jar, se procede a copiar dentro la carpeta *libs* como se indica en la figura inferior.



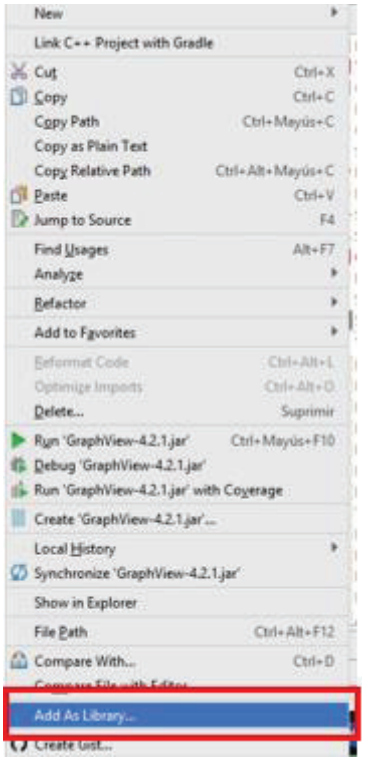
Al copiar dentro de la carpeta, se despliega una ventana emergente donde se indica el nombre y el directorio en el que se va a copiar.



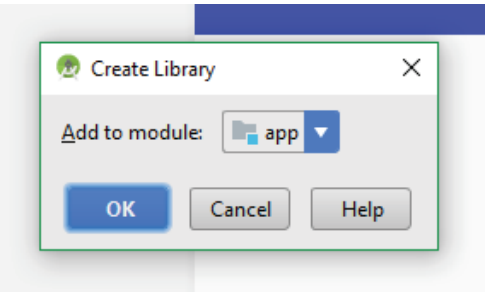
Una vez ya copiada, ya se podrá visualizar en el panel de navegación; sin embargo, esto no significa que ya se puede utilizar, para poder acceder a la librería e implementarla dentro de la aplicación, se debe añadir las dependencias.



Para agregar las dependencias se hace clic derecho sobre la librería copiada y dentro del menú que se despliega se escoge *Add to Lybrary…*



Se despliega una ventada donde indica en donde se va a añadir y se lo confirma con OK. Y de esta forma se habrá añadido correctamente y ya estará disponible para utilizar todas sus características.



ANEXO III

Clases implementadas en la aplicación de Android Studio para obtener la FFT de los datos generados por el acelerómetro.

COMPLEX.JAVA

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

/**
 * Created by giovy on 20/11/2017.
 */

public class Complex {

    private final double re;   // the real part
    private final double im;   // the imaginary part

    // create a new object with the given real and imaginary parts
    public Complex(double real, double imag) {
        re = real;
        im = imag;
    }

    // return a string representation of the invoking Complex object
    public String toString() {
        if (im == 0) return re + "";
        if (re == 0) return im + "i";
        if (im <  0) return re + " - " + (-im) + "i";
        return re + " + " + im + "i";
    }

    // return abs/modulus/magnitude
    public double abs() {
        return Math.hypot(re, im);
    }

    // return angle/phase/argument, normalized to be between -pi and pi
    public double phase() {
        return Math.atan2(im, re);
    }

    // return a new Complex object whose value is (this + b)
    public Complex plus(Complex b) {
        Complex a = this;             // invoking object
        double real = a.re + b.re;
        double imag = a.im + b.im;
        return new Complex(real, imag);
    }

    // return a new Complex object whose value is (this - b)
    public Complex minus(Complex b) {
        Complex a = this;
        double real = a.re - b.re;
        double imag = a.im - b.im;
        return new Complex(real, imag);
    }
```

```
    // return a new Complex object whose value is (this * b)
    public Complex times(Complex b) {
        Complex a = this;
        double real = a.re * b.re - a.im * b.im;
        double imag = a.re * b.im + a.im * b.re;
        return new Complex(real, imag);
    }

    // return a new object whose value is (this * alpha)
    public Complex scale(double alpha) {
        return new Complex(alpha * re, alpha * im);
    }

    // return a new Complex object whose value is the conjugate of this
    public Complex conjugate() {
        return new Complex(re, -im);
    }

    // return a new Complex object whose value is the reciprocal of this
    public Complex reciprocal() {
        double scale = re*re + im*im;
        return new Complex(re / scale, -im / scale);
    }

    // return the real or imaginary part
    public double re() { return re; }
    public double im() { return im; }

    // return a / b
    public Complex divides(Complex b) {
        Complex a = this;
        return a.times(b.reciprocal());
    }

    // return a new Complex object whose value is the complex exponential
of this
    public Complex exp() {
        return new Complex(Math.exp(re) * Math.cos(im), Math.exp(re) *
Math.sin(im));
    }

    // return a new Complex object whose value is the complex sine of
this
    public Complex sin() {
        return new Complex(Math.sin(re) * Math.cosh(im), Math.cos(re) *
Math.sinh(im));
    }

    // return a new Complex object whose value is the complex cosine of
this
    public Complex cos() {
        return new Complex(Math.cos(re) * Math.cosh(im), -Math.sin(re) *
Math.sinh(im));
    }

    // return a new Complex object whose value is the complex tangent of
this
    public Complex tan() {
        return sin().divides(cos());
```

```
    }


    // a static version of plus
    public static Complex plus(Complex a, Complex b) {
        double real = a.re + b.re;
        double imag = a.im + b.im;
        Complex sum = new Complex(real, imag);
        return sum;
    }

    // See Section 3.3.
    public boolean equals(Object x) {
        if (x == null) return false;
        if (this.getClass() != x.getClass()) return false;
        Complex that = (Complex) x;
        return (this.re == that.re) && (this.im == that.im);
    }
}
```

FFT.JAVA

```
package ec.edu.epn.lopez.giovi.com.vibrerapp;

/**
 * Created by giovy on 20/11/2017.
 */

public class FFT {

    // compute the FFT of x[], assuming its length is a power of 2
    public static Complex[] fft(Complex[] x) {
        int n = x.length;

        // base case
        if (n == 1) return new Complex[] { x[0] };

        // radix 2 Cooley-Tukey FFT
        if (n % 2 != 0) { throw new RuntimeException("n is not a power of
2"); }

        // fft of even terms
        Complex[] even = new Complex[n/2];
        for (int k = 0; k < n/2; k++) {
            even[k] = x[2*k];
        }
        Complex[] q = fft(even);

        // fft of odd terms
        Complex[] odd  = even;  // reuse the array
        for (int k = 0; k < n/2; k++) {
            odd[k] = x[2*k + 1];
        }
        Complex[] r = fft(odd);

        // combine
        Complex[] y = new Complex[n];
        for (int k = 0; k < n/2; k++) {
            double kth = -2 * k * Math.PI / n;
            Complex wk = new Complex(Math.cos(kth), Math.sin(kth));
            y[k]       = q[k].plus(wk.times(r[k]));
            y[k + n/2] = q[k].minus(wk.times(r[k]));
        }
        return y;
    }


    // compute the inverse FFT of x[], assuming its length is a power of
2
    public static Complex[] ifft(Complex[] x) {
        int n = x.length;
        Complex[] y = new Complex[n];

        // take conjugate
        for (int i = 0; i < n; i++) {
            y[i] = x[i].conjugate();
        }

        // compute forward FFT
```

```
        y = fft(y);

        // take conjugate again
        for (int i = 0; i < n; i++) {
            y[i] = y[i].conjugate();
        }

        // divide by n
        for (int i = 0; i < n; i++) {
            y[i] = y[i].scale(1.0 / n);
        }

        return y;
    }

    // compute the circular convolution of x and y
    public static Complex[] cconvolve(Complex[] x, Complex[] y) {

        // should probably pad x and y with 0s so that they have same
length
        // and are powers of 2
        if (x.length != y.length) { throw new
RuntimeException("Dimensions don't agree"); }

        int n = x.length;

        // compute FFT of each sequence
        Complex[] a = fft(x);
        Complex[] b = fft(y);

        // point-wise multiply
        Complex[] c = new Complex[n];
        for (int i = 0; i < n; i++) {
            c[i] = a[i].times(b[i]);
        }

        // compute inverse FFT
        return ifft(c);
    }

    // compute the linear convolution of x and y
    public static Complex[] convolve(Complex[] x, Complex[] y) {
        Complex ZERO = new Complex(0, 0);

        Complex[] a = new Complex[2*x.length];
        for (int i = 0;        i <   x.length; i++) a[i] = x[i];
        for (int i = x.length; i < 2*x.length; i++) a[i] = ZERO;

        Complex[] b = new Complex[2*y.length];
        for (int i = 0;        i <   y.length; i++) b[i] = y[i];
        for (int i = y.length; i < 2*y.length; i++) b[i] = ZERO;

        return cconvolve(a, b);
    }
}
```

ANEXO IV

Scripts generados en Matlab para el procesamiento de los archivos de texto de los equipos.

- **SCRIPT PARA EL PROCESAMIENTO DE LOS DATOS DEL ACELERÓMETRO GURALP CMG-5TDE**

proceso_guralp.m

```
close all
a_vert=(a(:,1));
a_norte=(a(:,2));
a_este=(a(:,3));
z=detrend(a_vert);
y=detrend(a_norte);
x=detrend(a_este);

Fs = 100;            % Sampling frequency
T = 1/Fs;             % Sampling period
L = 84000;             % Length of signal
t = (0:L-1)*T;        % Time vector

Y = fft(y);
X = fft(x);
Z = fft(z);

P2_Y = abs(Y/L);
P1_Y = P2_Y(1:L/2+1);
P1_Y(2:end-1) = 2*P1_Y(2:end-1);

P2_Z = abs(Z/L);
P1_Z = P2_Z(1:L/2+1);
P1_Z(2:end-1) = 2*P1_Z(2:end-1);

P2_X = abs(X/L);
P1_X = P2_X(1:L/2+1);
P1_X(2:end-1) = 2*P1_X(2:end-1);

f = Fs*(0:(L/2))/L;
figure;
plot(f,P1_Y)
title('Espectro eje Y')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

figure;
plot(f,P1_Z)
title('Espectro eje Z')
xlabel('f (Hz)')
ylabel('Amplitud')
```

```
ylim([0 0.05])
grid on

figure;
plot(f,P1_X)
title('Espectro eje X')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

[M,I]= max(abs(a_vert))
PGA_X=x(I);

[M,I]= max(abs(a_norte))
PGA_Y=y(I);

[M,I]= max(abs(a_este))
PGA_Z=z(I);

fprintf('PGA X: %.4f  PGA Y: %.4f   PGA Z: %.4f \n',PGA_X,PGA_Y,PGA_Z);
```

- **SCRIPT PARA EL PROCESAMIENTO DE LOS DATOS DEL ACELERÓMETRO PHIDGETS SPATIAL**

phidgets.m

```
close all
a_x=(a(:,1));
a_y=(a(:,2));
a_z=(a(:,3));
x=detrend(a_x);
y=detrend(a_y);
z=detrend(a_z);

Fs = 125;            % Sampling frequency
T = 1/Fs;             % Sampling period
L = 8500;             % Length of signal
t = (0:L-1)*T;        % Time vector

Y = fft(y(1:8500));
X = fft(x(1:8500));
Z = fft(z(1:8500));

P2_Y = abs(Y/L);
P1_Y = P2_Y(1:L/2+1);
P1_Y(2:end-1) = 2*P1_Y(2:end-1);

P2_Z = abs(Z/L);
P1_Z = P2_Z(1:L/2+1);
P1_Z(2:end-1) = 2*P1_Z(2:end-1);

P2_X = abs(X/L);
P1_X = P2_X(1:L/2+1);
P1_X(2:end-1) = 2*P1_X(2:end-1);
```

```
f = Fs*(0:(L/2))/L;
figure;
plot(f,P1_Y)
title('Espectro eje Y')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.05])
grid on

figure;
plot(f,P1_Z)
title('Espectro eje Z')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.005])
grid on

figure;
plot(f,P1_X)
title('Espectro eje X')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.05])
grid on

[M,I]= max(abs(a_x))
PGA_X=x(I);

[M,I]= max(abs(a_y))
PGA_Y=y(I);

[M,I]= max(abs(a_z))
PGA_Z=z(I);

fprintf('PGA X: %.4f  PGA Y: %.4f   PGA Z: %.4f \n',PGA_X,PGA_Y,PGA_Z);
```

- **SCRIPT PARA EL PROCESAMIENTO DE LOS DATOS DEL ACELERÓMETRO REFTEK**

reftek.m

```
close all
a_vert=(a(:,1));
a_norte=(a(:,2));
a_este=(a(:,3));
z=detrend(a_vert);
y=detrend(a_norte);
x=detrend(a_este);

Fs = 100;            % Sampling frequency
T = 1/Fs;             % Sampling period
L = 48000;             % Length of signal
t = (0:L-1)*T;        % Time vector

Y = fft(y);
X = fft(x);
Z = fft(z);
```

```
P2_Y = abs(Y/L);
P1_Y = P2_Y(1:L/2+1);
P1_Y(2:end-1) = 2*P1_Y(2:end-1);

P2_Z = abs(Z/L);
P1_Z = P2_Z(1:L/2+1);
P1_Z(2:end-1) = 2*P1_Z(2:end-1);

P2_X = abs(X/L);
P1_X = P2_X(1:L/2+1);
P1_X(2:end-1) = 2*P1_X(2:end-1);

f = Fs*(0:(L/2))/L;
figure;
plot(f,P1_Y)
title('Espectro eje Y')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

figure;
plot(f,P1_Z)
title('Espectro eje Z')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.05])
grid on

figure;
plot(f,P1_X)
title('Espectro eje X')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

[M,I]= max(abs(a_vert))
PGA_X=x(I);

[M,I]= max(abs(a_norte))
PGA_Y=y(I);

[M,I]= max(abs(a_este))
PGA_Z=z(I);

fprintf('PGA X: %.4f  PGA Y: %.4f   PGA Z: %.4f \n',PGA_X,PGA_Y,PGA_Z);
```

- **SCRIPT PARA EL PROCESAMIENTO DE LOS DATOS DE LOS SMARTPHONES**

teléfono.m

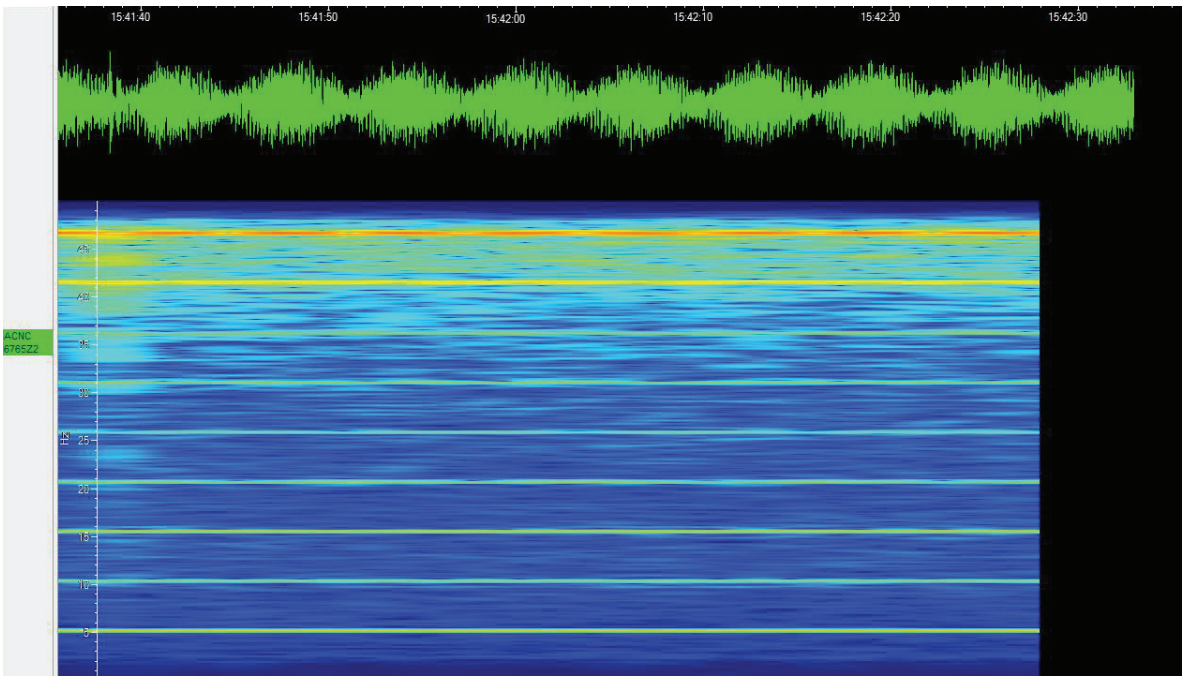
```
close all
a_x=(a(:,1));
a_y=(a(:,2));
a_z=(a(:,3));
x= detrend(a_x);
y=detrend(a_y);
z=detrend(a_z);

Fs = 100;            % Sampling frequency
T = 1/Fs;             % Sampling period
L = 8192;             % Length of signal
t = (0:L-1)*T;        % Time vector

x=[x ; zeros(8192-numel(x),1)];
y=[y ; zeros(8192-numel(y),1)];
z=[z ; zeros(8192-numel(z),1)];

Y = fft(y);
X = fft(x);
Z = fft(z);

P2_Y = abs(Y/L);
P1_Y = P2_Y(1:L/2+1);
P1_Y(2:end-1) = 2*P1_Y(2:end-1);

P2_Z = abs(Z/L);
P1_Z = P2_Z(1:L/2+1);
P1_Z(2:end-1) = 2*P1_Z(2:end-1);

P2_X = abs(X/L);
P1_X = P2_X(1:L/2+1);
P1_X(2:end-1) = 2*P1_X(2:end-1);

f = Fs*(0:(L/2))/L;
figure;
plot(f,P1_Y)
title('Espectro eje Y')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

figure;
plot(f,P1_Z)
title('Espectro eje Z')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 1])
grid on

figure;
plot(f,P1_X)
```

```
title('Espectro eje X')
xlabel('f (Hz)')
ylabel('Amplitud')
ylim([0 0.5])
grid on

[M,I]= max(abs(a_x))
PGA_X=x(I);

[M,I]= max(abs(a_y))
PGA_Y=y(I);

[M,I]= max(abs(a_z))
PGA_Z=z(I);

fprintf('PGA X: %.4f  PGA Y: %.4f   PGA Z: %.4f \n',PGA_X,PGA_Y,PGA_Z);
```

ANEXO V

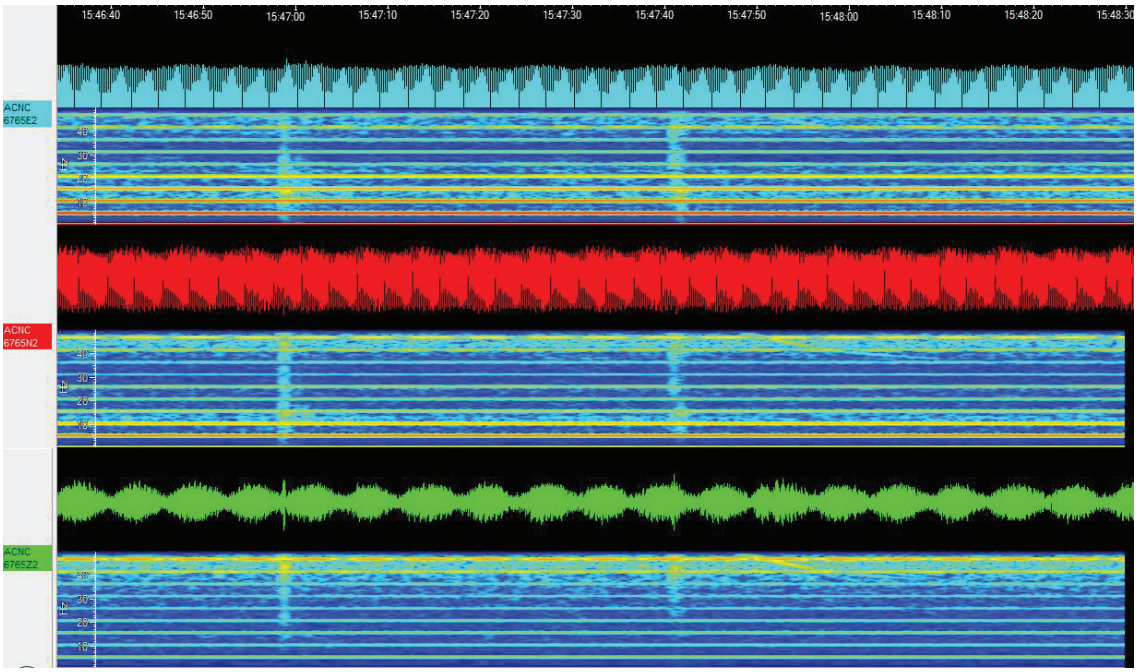
En este anexo se presenta los espectrogramas obtenidos en el computador durante las pruebas de experimentación con el equipo Guralp Systems.

Spectrogram de la componente Vertical (Z)

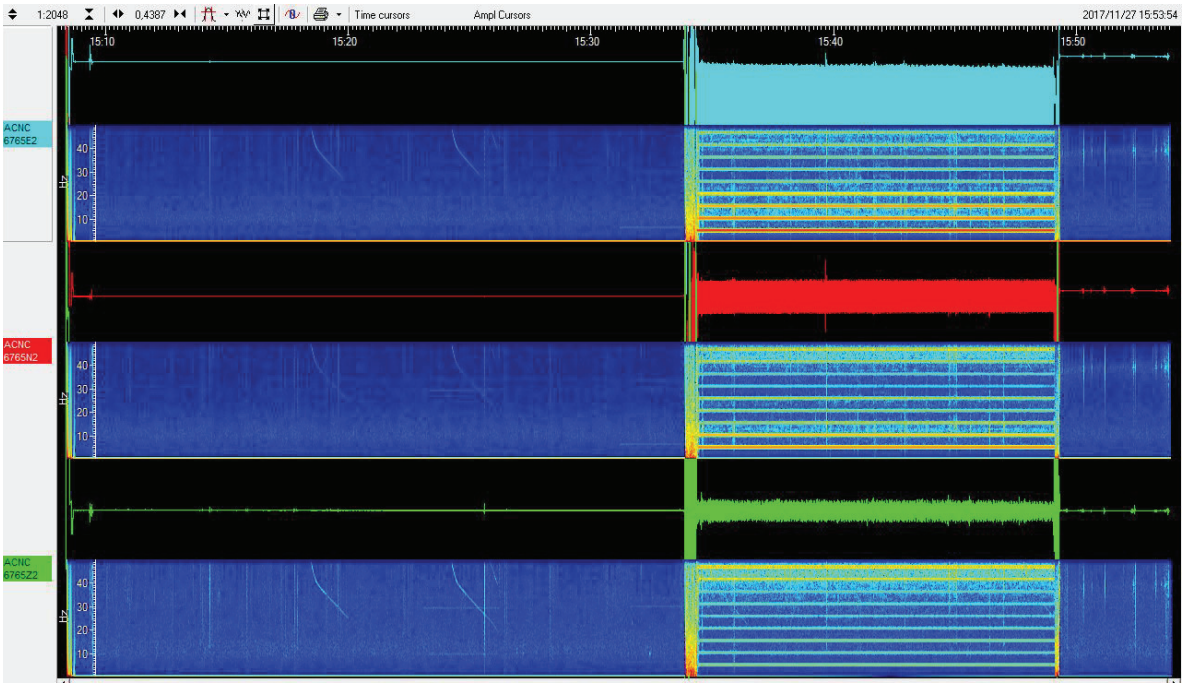


En el espectrograma se observa las frecuencias predominantes como las franjas de color amarillo y celestes en forma horizontal. Se puede ver que coinciden con el espectro de Matlab obtenido en el desarrollo del capítulo 3, las cuales se presentan en 5 Hz y con el mismo espaciamiento a lo largo del ensayo.

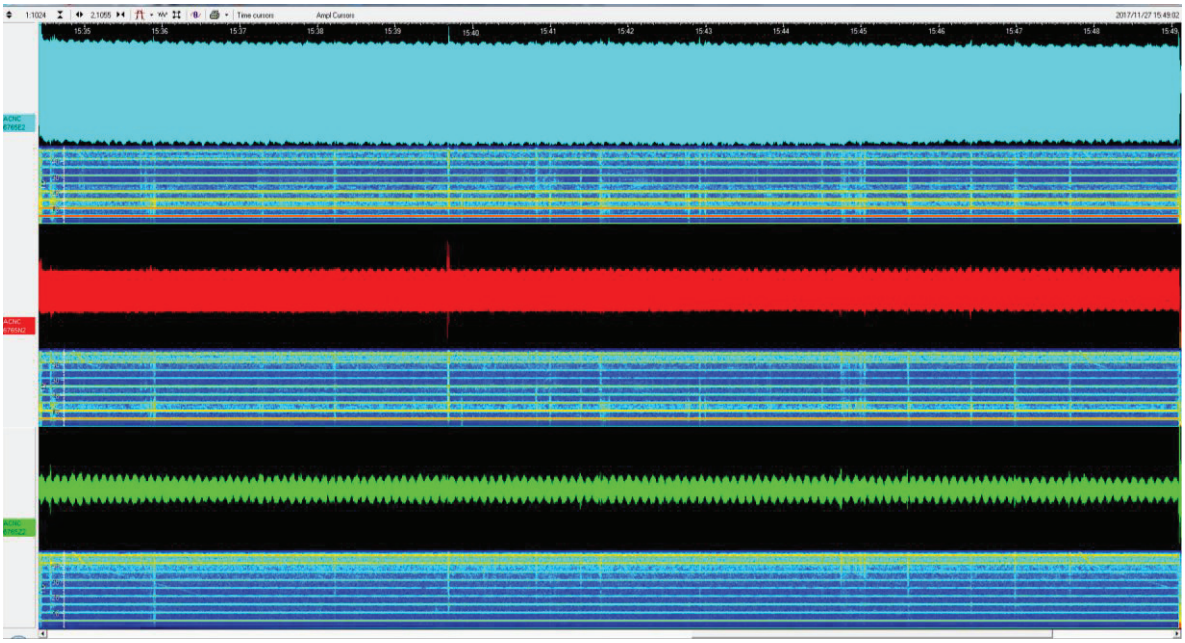
Spectrogram de las 3 componentes Este-Oeste (E), Norte-Sur (N) y Vertical (Z)



Ventana completa antes durante y después del ensayo



Ventana completa del todo el ensayo



ORDEN DE EMPASTADO