Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA PRE-EVALUACIÓN DEL COMPORTAMIENTO DINÁMICO DE ESTRUCTURAS ESBELTAS

Departamento: Construcciones Arquitectónicas, Ingeniería del Terreno y Mecánica de Medios Continuos y Teoría de Estructuras

Autor: D. Juan Miguel Alario Bercianos

Tutor: D. Antonio Foces Mediavilla

Valladolid, Junio de 2018

Resumen

La necesidad de registrar y medir fenómenos físicos ha sido y es una constante en el ámbito de la ingeniería. A esta problemática se le ha dado respuesta tradicionalmente con equipo e instrumentación profesional para la adquisición de datos, equipo específico generalmente caracterizado por un elevado coste y complejidad de uso, además de ajustarse a un uso concreto. Sin embargo, en la actualidad y con la aparición y generalización de la electrónica de consumo, existe una gran variedad de dispositivos con capacidad de registrar magnitudes físicas a través de los sensores que en ellos se incluyen. En concreto, los Smartphone reúnen todas las características necesarias para actuar como registradores: sensores de todo tipo, capacidad de procesamiento y conectividad. En base a esta problemática, el presente Trabajo de Fin de Máster pretende plantear una herramienta de adquisición de datos basada en el uso de Smartphones y microcontroladores de bajo coste para el caso concreto de registrar el comportamiento dinámico de estructuras esbeltas.

Palabras clave: aplicación, apk, Android, Matlab, Arduino, vibraciones, MTVV, RMS, FFT, pasarelas.

Abstract

The need to record and measure physical phenomena has been constantly present in the engineering field. This problem has traditionally been answered with professional instrumentation for the acquisition of data, specific equipment generally characterized by high cost and usage complexity, apart from generally having just one specific use. However, currently and with the soaring rise and large generalization of consumer electronics, there is a wide variety of devices with the ability to record physical magnitudes through their included sensors. Specifically, the Smartphones have all the necessary characteristics to act as recorders: sensors of all types, processing capacity and connectivity. Given this problematic, the present Master's Thesis aims to develop a data acquisition tool based on the use of Smartphones and low cost microcontrollers in the specific case of dynamic behavior recording of slender structures.

Keywords: application, apk, Android , Matlab, Arduino, vibrations, MTVV, RMS, FFT, pedestrian bridges.

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN

1.1-ANTECEDENTES

En el campo de la ingeniería estructural, bien sea de ámbito industrial como civil, es necesario asegurar que la estructura cumple ciertos criterios exigibles denominados "estados límite". Estos estados se clasifican en "últimos" y "de servicio", que de acuerdo a la IA-11 se definen como:

- **Estados límite últimos (ELU):** son los que, de sobrepasarse, conllevan el agotamiento o colapso de la estructura o de una parte de ella.
  Se consideran:
  - *ELU de equilibrio (EQU):* causada por la pérdida de estabilidad estática de una parte o del conjunto de la estructura, considerada como un cuerpo rígido. Se caracteriza por que pequeñas variaciones en el valor o en la distribución espacial de acciones con un mismo origen resultan significativas y porque la resistencia de los materiales estructurales o del terreno no son en general determinantes.
  - *ELU de rotura (STR):* se produce por agotamiento resistente o deformación plástica excesiva, donde la resistencia de los materiales estructurales es determinante.
  - *ELU de fatiga (FAT):* relacionado con los daños que puede sufrir una estructura o cualquiera de sus elementos como consecuencia de solicitaciones variables repetidas.
- **Estados límite de servicio:** son los que, de sobrepasarse, la estructura dejará de cumplir el cometido para el que fue proyectada por razones funcionales, de durabilidad o de aspecto, sin que ello suponga el colapso de la misma.
  Se consideran:
  - *ELS de fisuración,* que afecten a la durabilidad o estética del puente.
  - *ELS de deformación,* que afecte a la apariencia o funcionalidad de la obra, o que cause daño a elementos no estructurales.
  - ***ELS de vibraciones,*** que no sean aceptables para los usuarios del puente o que puedan afectar a la funcionalidad o provocar daños en los elementos no estructurales.
  - *ELS de plastificación,* en zonas localizadas de la estructura que puedan provocar daños o deformaciones irreversibles.
  - *ELS de deslizamiento* en uniones mediante tornillos de alta resistencia.

Como se ve, entre los estados límite de servicio se encuentra el de "comodidad ante vibraciones". Según el artículo 38 de la Instrucción de Acero Estructural (EAE) referente a este **Estado Límite de vibraciones**, la adecuada respuesta en servicio de una estructura o elemento estructural aislado debe garantizar:

- El confort de los usuarios.
- La ausencia de deteriores en la propia estructura o en los elementos no resistentes soportados por ella, originados por efectos dinámicos.
- El correcto funcionamiento y durabilidad de posibles maquinarias, servicios, instalaciones, etc., sensibles a estos fenómenos.

Bajo la perspectiva de las vibraciones, los efectos a considerar pueden ser inducidos por maquinarias, movimientos sincronizados de gente (andando, bailando o saltando), viento y oleaje, etc., y pueden resultar amplificados por condiciones de resonancia. Para limitar esto, se deben respetar unas condiciones de diseño de tal manera que las frecuencias fundamentales estén lo suficientemente alejadas (generalmente superiores) de las frecuencias producidas por las fuentes de excitación.

Tipo de estructura	Frecuencia fundamental (Hz)
Gimnasios y polideportivos	> 8,0
Salas de fiestas y locales sin asientos fijos	> 7,0
Estadios, locales de concierto o espectáculo con asientos fijos	> 3,4
Oficinas, centros comerciales	> 3,0

Tabla 1.1.1-Valores límite indicativos y recomendados para las frecuencias fundamentales de estructuras abiertas al público. Fuente: EAE 2011

Ahondado en lo expuesto en esta norma, concretamente el art. 38.3, se habla de la problemática de las **pasarelas**, siendo esta una de las líneas de investigación de los tutores de este Trabajo de Fin de Máster (de ahora en adelante TFM) e inspirando el desarrollo del mismo, al tratar de dar respuesta a la problemática del análisis dinámico de este tipo de estructuras.

Volviendo a la problemática expuesta en el art. 38.3.2, resultan susceptibles de fenómenos vibratorios que pueden afectar al confort de los peatones las pasarelas cuyas frecuencias fundamentales estén comprendidas entre los siguientes rangos críticos:

- Para oscilaciones en el plano vertical: entre 1.25 y 4.60 Hz.
- Para oscilaciones en el plano horizontal o de torsión: entre 0.50 y 1.20 Hz.

En caso de que las frecuencias fundamentales de las pasarelas se encuentren fuera de los mencionados rangos críticos, no suele resultar necesario ningún análisis dinámico. Tampoco es necesario dicho análisis en el caso de pasarelas convencionales cuyas frecuencias fundamentales en el plano vertical estén comprendidas en el citado rango crítico si se satisface el valor límite de deformaciones citado en el art. 37.3.1: *L/700*.

Por el contrario, sí resulta necesario realizar un estudio dinámico específico para casos como los siguientes:

- Estructuras singulares no convencionales.
- Pasarelas de luces superiores a 50 m.
- Pasarelas de anchura superior a 3.0 m.
- Pasarelas ubicadas en zonas donde puede esperarse un tráfico intenso de peatones o exista riesgo de concentraciones de gente sobre la propia pasarela.
- Pasarelas cuyas frecuencias fundamentales para oscilaciones, en plano horizontal o de torsión, se sitúen en el rango crítico antes indicado.

En cuanto a las frecuencias correspondientes a las acciones dinámicas habituales sobre este tipo de estructuras, nos encontramos con las siguientes:

- Entre 1.25 Hz y 2.4 Hz, excitación debida a peatones andando o corriendo sobre la plataforma.
- Entre 2.5 Hz y 4.6 Hz, corresponde a la influencia del segundo armónico de la excitación generada por la acción de peatones que pueda afectar a pasarelas de acero de bajo amortiguamiento. Las fuerzas dinámicas en este rango son siempre menores que en el caso anterior, con lo que el riesgo de excitación no tolerable será menor.
- Existe también un rango de entre 2 Hz y 3.5 Hz donde la excitación se corresponde a la acción de peatones corriendo a cierta velocidad, pero resulta poco probable que se sincronicen varios corredores, con lo que las fuerzas dinámicas no deberían ser relevantes.

Con lo dicho y en lo relativo a los estudios dinámicos, para este tipo de estructuras esbeltas se dan dos circunstancias por las cuales no es necesario muestrear a altas velocidades. Por una parte, suelen tener modos de vibración a bajas frecuencias, por debajo de 10 Hz. Y por otra, las principales excitaciones provienen, o bien del viento, o bien del tráfico. Las vibraciones inducidas por el viento también suelen ser a bajas frecuencias y las debidas al uso de las personas se centran, como se ha visto a partir de las normas, también en una banda entorno a los 2 Hz. Por tanto, las oscilaciones más significativas que sufren estas estructuras, tanto propias como forzadas, están por debajo de los 10 Hz.

Si se dispusiera de acelerómetros para su registro, con muestreos de unos 20 registros por segundo evitaríamos ya problemas de *aliasing*. Este efecto de *aliasing* o solapamiento produce que diferentes señales se conviertan en indistinguibles (o *aliases*, alias en inglés) cuando son muestreadas. Este efecto podemos verlo reflejado en la figura 1.1.1.

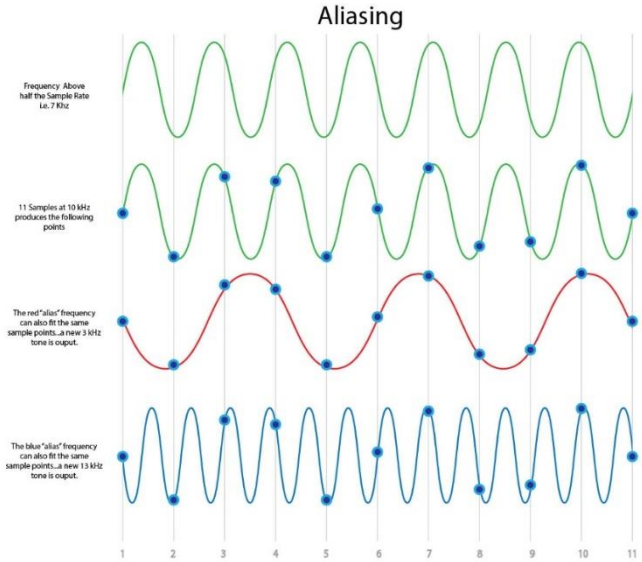


Figura 1.1.1-Señal muestreada a determinada frecuencia y diferentes señales reconstruidas a partir de este muestreo, diferentes a la original debido al efecto aliasing. Fuente: http://www.realhd-audio.com/?p=2983

Cuando se muestrea este tipo de señales continuas, el número de ciclos por muestra se corresponde con $^{f}/_{f_s}$, conocido como frecuencia normalizada, donde f es la frecuencia de la señal medida y f_s la frecuencia de muestreo. Esta señal muestreada es indistinguible de cualquier otra señal sinusoidal cuya frecuencia normalizada difiera de la citada $^{f}/_{f_s}$ por un entero (negativo o positivo). Estas frecuencias de las señales *alias* se pueden expresar por tanto como $f_{alias}(N) \stackrel{\text{def}}{=} |f - Nf_s|$, siendo N un número entero y expresado en ciclos por muestra y $f_{alias}(0) = f$ el verdadero valor de la señal.

Este efecto aparece cuando se trata de reconstruir la señal muestreada, puesto que las técnicas más comunes de reconstrucción lo hacen en base la menor de las $f_{alias}(N)$ frecuencias, de modo que es importante que $f_{alias}(0)$ sea el único mínimo. Esto se garantiza cumpliendo la condición necesaria y suficiente de que $^{f_s}/_2 > |f|$, donde $^{f_s}/_2$ es la frecuencia de *Nysquit*, y de aquí que se haya afirmado que, dado que las frecuencias a analizar estarán por debajo de los 10 Hz, con un muestreo de unos 20 registros por segundo se evitaría este problema.

Aun así, se suelen recomendar frecuencias de muestreo de unas 10 veces la máxima frecuencia de interés. Si esta es de 10 Hz conviene muestrear a 100 Hz, es decir, 100 veces en un segundo o tomar un dato cada 10 ms. Esta capacidad de postproceso está, en principio, al alcance de cualquier smartphone, y de ahí nace la motivación de este TFM.

1.2-OBJETIVOS

Se establece como objetivo básico el siguiente:

- Desarrollo de una aplicación móvil que permita estimar de forma rápida ciertas características dinámicas de las estructuras como son sus frecuencias de vibración y su amortiguamiento. Asimismo, la aplicación estimará ciertos indicadores sobre el estado límite de servicio respecto a vibraciones como son: aceleraciones pico, aceleraciones eficaces (RMS) y MTVV. Para este objetivo se usará, únicamente, un smartphone que servirá tanto como equipo de adquisición de datos (a través de su acelerómetro interno) como procesador de las señales registradas.

Adicionalmente y a medida que se ha ido trabajando en la consecución de este objetivo básico, se ha visto la conveniencia de completarlo con los siguientes objetivos secundarios:

- Desarrollo de un entorno para PC bajo Matlab, con mayor capacidad de procesado, en el que la adquisición se lleva a cabo a través del acelerómetro embebido en el smartphone. Para ello, será necesario garantizar la correcta comunicación entre el PC y el smartphone.

- Sustituir el smartphone del punto anterior por un microcontrolador *low cost* basado en Arduino y dotado de un módulo con acelerómetro y un módulo de comunicaciones.

- Explorar otras posibilidades de comunicación entre los equipos comentados anteriormente. Concretamente se han hecho pruebas conectando varios módulos con acelerómetros al Arduino, conectando el Arduino directamente al smartphone y otras alternativas consistentes en múltiples equipos formando una red.

Como se verá a lo largo del desarrollo de este trabajo y se resumirá en las conclusiones, las posibilidades que ofrecen las nuevas tecnologías son muy atractivas, habiéndose conseguido buenos resultados en todos los objetivos indicados. No obstante, las prestaciones son limitadas precisamente por las limitaciones de los equipos utilizados, básicamente debidas al número de bits, velocidades de procesado y las propias de los protocolos comunicación. Estas limitaciones hacen que las aplicaciones no sean adecuadas para la medida de estructuras con altas frecuencias de vibración. Sin embargo, para estructuras esbeltas con modos por debajo de 10 Hz los resultados obtenidos son perfectamente válidos, al menos para estudios

preliminares. Con estos estudios preliminares se podría valorar la necesidad de recurrir a despliegues de equipos profesionales, siendo este otro objetivo del trabajo realizado.

1.3-PLANTEAMIENTO

Para cumplir los propósitos descritos en el punto anterior, se van a plantear una serie de soluciones utilizando softwares y hardwares de diferente tipo y que permita, en todo caso, cumplir con los objetivos planteados: registrar y procesar con sencillez y coste comedido la aceleración que se produce en una estructura.

En la actualidad, se encuentran disponibles infinidad de dispositivos de uso cotidiano como los smartphones u ordenadores personales que ofrecen al usuario medio una experiencia que tiende a ser cerrada, pero que por la naturaleza de estos dispositivos plantean un enorme abanico de posibilidades. Un caso a destacar es el de los smartphones, que reúnen una cantidad de aplicaciones y funcionalidades destacable, así como una potencia de procesamiento elevada, lo que da a este dispositivo un potencial nada desdeñable a la hora de llevar a cabo actividades más allá de su propósito original de elemento de ocio. La presencia de elementos como acelerómetros o giróscopos, en ocasiones de notable calidad, junto con la capacidad de procesar estas entradas de datos, implementadas a veces por los propios fabricantes, hace que el smartphone sea el elemento de hardware/software principal para el desarrollo de las aplicaciones del presente trabajo.

El hecho de que el propósito con el que el fabricante desarrolla un smartphone sea el de un elemento principalmente de ocio hace que la existencia de software de ingeniería específico que utilice las capacidades implementadas en un smartphone sea escasa, por lo que, en muchos casos, como el presente, el desarrollo de una aplicación desde cero es necesario.

Además de la existencia de los smartphones, en cuanto a hardware existen en la actualidad otras opciones que permiten el desarrollo de dispositivos modulares para cualquier usuario medio y por un bajo coste, como es el caso de los Arduino, o más recientemente de las RaspberryPi. Para el caso del presente proyecto, se considera oportuno el uso de un Arduino, ya que nos va a permitir registrar y enviar a otro dispositivo con más capacidad de procesamiento los datos gracias que se trata de un microcontrolador. Esta funcionalidad también sería capaz de llevarla a cabo una RaspberryPi, pero, al funcionar este último como un microprocesador y no como un microntrolador, obedecerá con prioridad a su sistema operativo y no solo al programa que se le introduzca, lo que resulta menos indicado para la tarea que realizará.

Por otra parte, mientras que en el campo de los smartphones las aplicaciones de ingeniería apenas están presentes, podemos encontrar infinidad de ellas disponibles para nuestros PC, con una potencia de cálculo y procesamiento muy elevadas, y que además ofrecen al usuario herramientas optimizadas y de uso relativamente sencillo. Por esto, el uso de este tipo de herramientas es de interés para el desarrollo de las aplicaciones que conforman este Trabajo Fin de Máster. En concreto, el uso de Matlab, probablemente el software de ingeniería por antonomasia, es de elevado interés, ya no sólo porque ofrezca una elevada capacidad de cálculo, sino porque también ofrece una gran variedad de posibilidades de uso y de conectividad gracias a los “Add-on” y “toolbox”, aplicaciones y librerías desarrolladas por el propio Mathworks o terceros que añaden funcionalidades extra al software. En este caso son de espacial interés las que permiten conectar el programa con smartphones y/o arduinos para registrar datos y después procesarlos con el propio software, algo que se corresponde

exactamente a lo que se plantea en el presente trabajo y en el desarrollo de las aplicaciones comprendidas en él.

En resumen, el punto de partida para el desarrollo de las aplicaciones gira en torno, principalmente, al uso del smartphone y en concreto a su capacidad de procesamiento y toma de registros mediante sus sensores, pero también al de otro tipo de hardware como es el del Arduino, que al ser "*open source*" permite desarrollar fácilmente aplicaciones para una gran variedad de usos y con herramientas abiertas. Por último, el uso de un software como Matlab también resulta interesante para el caso de estas aplicaciones porque permite el uso conjugado de estos dos hardwares con, además, la potencia y capacidad de cálculo y procesamiento que ofrece el propio programa y el hardware de un ordenador, habitualmente, mucho más potente que el de un smartphone.

CAPÍTULO 2: APLICACIONES

Siguiendo el planteamiento descrito en el punto 1.3, en el presente capítulo se van a exponer las aplicaciones desarrolladas en este TFM, tanto sus funcionalidades y uso como consideraciones relevantes sobre su desarrollo.

2.1-APLICACIÓN PARA SMARTPHONE

Como se ha introducido previamente, la potencia actual de estos dispositivos móviles y el gran número y variedad de útiles electrónicos, tales como acelerómetros o giróscopos, incluidos en ellos hacen del Smartphone un elemento ideal para desarrollar una aplicación que cumpla con los objetivos propuestos en el presente trabajo. Además, en la actualidad, prácticamente todo el mundo en los países desarrollados dispone de un smartphone, con lo que la posibilidad de dar un uso específico de ingeniería a estos dispositivos está al alcance de todo el mundo. Y no solo es una posibilidad que surge del uso cotidiano de estos dispositivos, si no que abre la posibilidad a reutilizar smartphones en desuso, descartados por nuevas revisiones, pero con plenas capacidades para las tareas a acometer, cumpliendo así uno de los objetivos básicos del presente proyecto, que es el de utilizar como entrada de registro elementos que no supongan un gasto adicional o importante.

Ahondando en lo concerniente a la situación actual de este tipo de dispositivos, nos encontramos una gran variedad de dispositivos disponibles, pero un número mucho más reducido de sistemas operativos para ellos. En concreto, reduciendo este número a aquellos que no tienen una presencia marginal en el mercado, quedan solamente dos: iOS y Android. Estos sistemas, propiedad de las grandes tecnológicas Apple y Google respectivamente, ocupan entre los dos casi el 100% del mercado de los smartphones, como se muestra en la figura 3.1.1, con lo que la elección de estos para desarrollar nuestra aplicación es evidente.

En concreto, la aplicación smartphone se va a desarrollar sobre el sistema operativo Android, por tratarse este no sólo del más extendido de los dos con una amplia diferencia, sino que además es el sistema operativo de los dos que permite más libertad al usuario de cara a desarrollar aplicaciones, con lo que resulta la opción más conveniente.

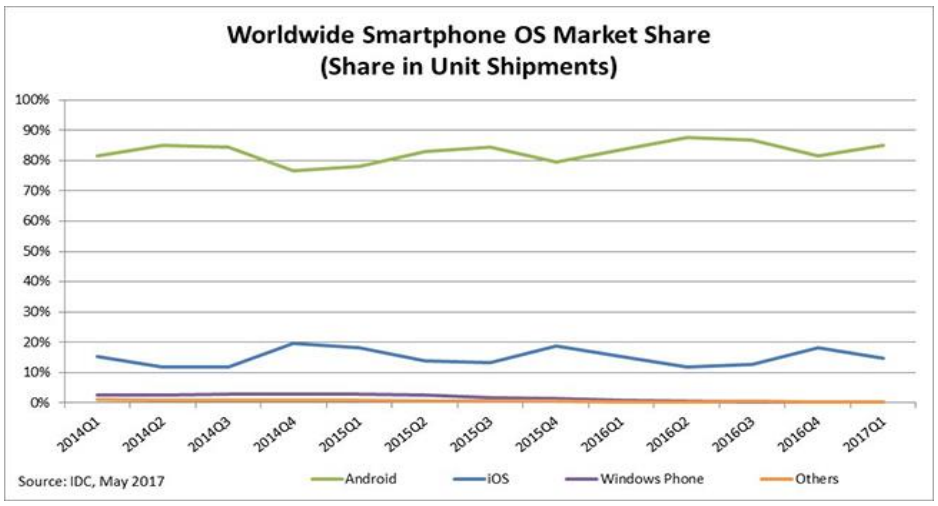


Figura 2.1.1-Posicionamiento en el mercado de los diferentes SO móviles. Fuente: https://www.idc.com/promo/smartphone-market-share/os

2.1.1-PLATAFORMA DE DESARROLLO

Tras esta breve introducción y antes de entrar a tratar en profundidad la aplicación desarrollada, conviene que se comenten brevemente algunos aspectos sobre la plataforma para la que se ha desarrollado esta aplicación.

SISTEMA OPERATIVO ANDROID

Lo primero a tratar es el sistema operativo utilizado: **Android**.

Con respecto a la nomenclatura para identificar la versión del sistema, podemos destacar que existen 3 maneras de nombrarlo:

- La comercial, con el nombre de un postre.
- La de los fabricantes, con la versión y subversión. Por ejemplo 4.4.
- La de desarrollador con el nivel de API.

En el caso de la presente aplicación, se ha desarrollado para la versión de Android 4.0.3 IceCreamSandwich API15, siendo esta la elegida por ser suficientemente avanzada para el desarrollo de la aplicación y por abarcar, de acuerdo a la propia IDE de Google como muestra la figura 2.1.1.1, la práctica totalidad de dispositivos Android.

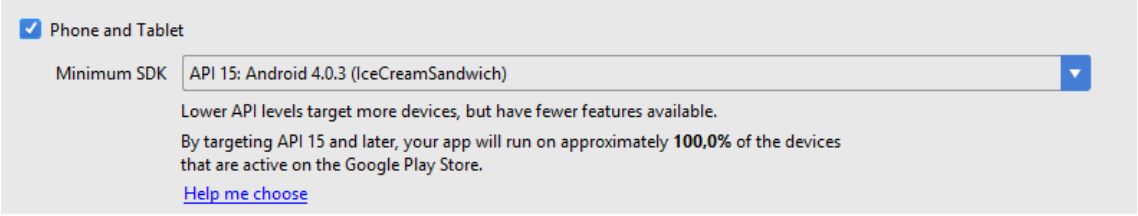


Figura 2.1.1.1- Recorte de la pantalla de selección de la API a utilizar en la aplicación de la IDE Android Studio.

ENTORNO DE DESARROLLO

El entorno que se ha usado para desarrollar la aplicación ha sido Android Studio, que es el entorno de desarrollo integrado (IDE) oficial de Google para la plataforma Android.

Las características de este IDE se pueden resumir en los siguientes aspectos:

- Renderización en tiempo real. Lee y ejecuta código XML en tiempo real para agilizar el desarrollo de elementos gráficos.
- Consola de desarrollador: consejos de optimización, ayudas, estadísticas…
- Soporte para construcción basada en Gradle.
- Refactorización en tiempo real.
- Plantillas precargadas con modelos de diseño comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.

En resumen, es una IDE como pueden ser NetBeans o Eclipse, pero enfocada enteramente al desarrollo de programas para Android y programar en los lenguajes que éste admite (esencialmente Java, C/C++ y en la versión 3.0 de este IDE el nuevo lenguaje Koltan), y que además nos permite ejecutar nuestras aplicaciones a través de un emulador virtual o de un dispositivo real a través de USB.

Algunos ejemplos de las funcionalidades que nos ofrece esta IDE se muestran en las siguientes figuras:

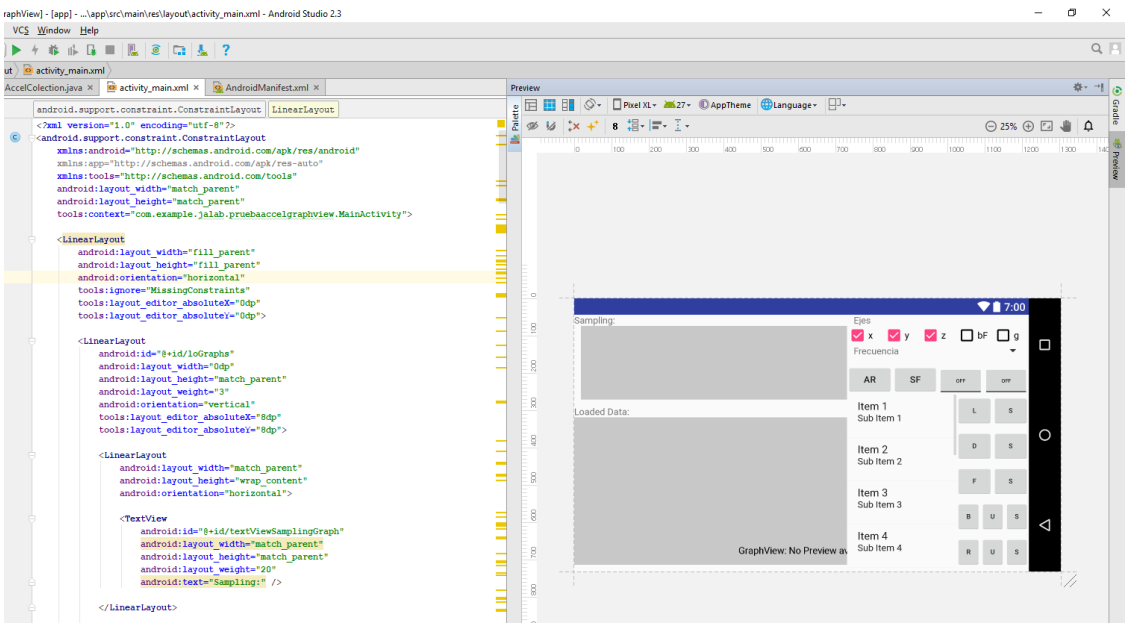


Figura 2.1.1.2-Renderizado en tiempo real de código XML para diseño de layouts

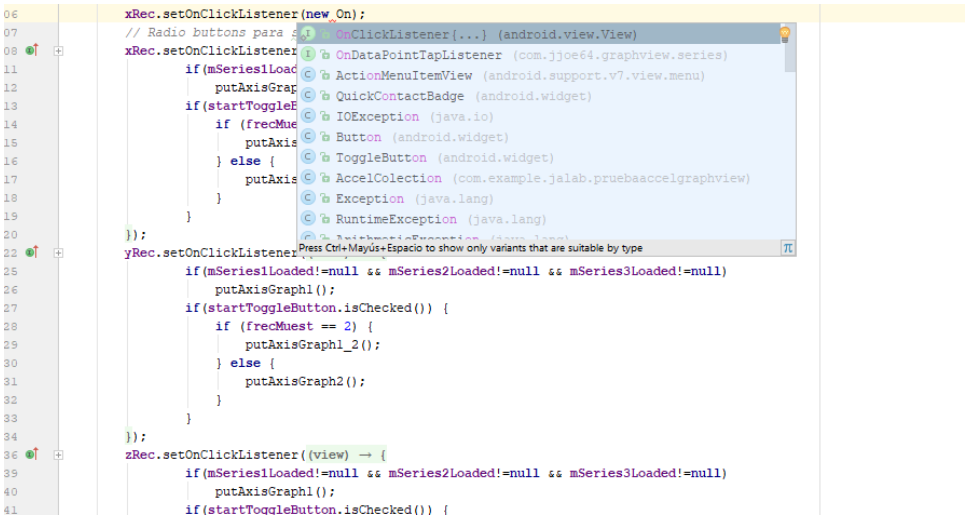


Figura 2.1.1.3-Sugerencias de código

FUNDAMENTOS DE PROGRAMACIÓN EN ANDROID

También es conveniente tocar algunos aspectos básicos sobre los fundamentos de programación en Android de cara a entender tanto el desarrollo posterior de la aplicación como algunas decisiones de diseño.

Primero destacar que, como ya se ha venido introduciendo, para desarrollar una aplicación en Android hace falta programar en más de un lenguaje. Principalmente podemos distinguirlos de la siguiente manera:

- ***Programación Dinámica:*** en nuestro caso (y lo más habitual) **Java**, aunque en la última versión del IDE. Con el programamos la parte de la lógica. En definitiva, como nos deja entrever su nombre, la parte dinámica de la aplicación.
- ***Programación Estática:*** para esto se utiliza **XML**, con el que se programa básicamente los elementos visuales y recursos, los de carácter estático, puesto que también se pueden programar mediante Java si no lo son (transiciones, animaciones, reajustes de parámetros gráficos, etc.).

Por otro lado, en cuanto a la estructura de programación propia de Android, se compone de los siguientes elementos básicos:

- **Activity**: de acuerdo a la documentación de Google, una activity es "*a single, focused thing that the user can do*", o sea, una tarea única con la que el usuario interactúa.
- **Service**: Se ejecuta en segundo plano. Sirve para realizar aplicaciones que necesiten ejecutarse de continuo o realizar trabajo que provenga de diferentes procesos. Lo que vienen a ser threads o hilos diferentes de la activity en curso.
- **Content provider**: Permite a una App acceder a datos de otra App.
- **Broadcast Receiver**: Responde a mensajes difundidos a todo el sistema.

Todos estos componentes influyen en el funcionamiento de nuestro dispositivo, por lo que deben de ser declarados sus permisos en el AndroidManifest, que es un archivo XML donde se declaran qué componentes puede ejecutar la aplicación, como en el caso de la aplicación desarrollada, que requiere de declarar los permisos para utilizar el Bluetooth. Estos permisos los podemos ver en la figura 2.1.1.4.

```
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.jalab.pruebaaccelgraphview">
4
5     <uses-permission android:name="android.permission.BLUETOOTH" />
6     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Figura 2.1.1.4-Extracto del Manifest.xml de la aplicación donde se especifican los permisos de acceso de la misma.

2.1.2-APLICACIÓN

Una vez introducidos algunos aspectos básicos y relevantes para entender el funcionamiento de la aplicación, es momento de tratar la aplicación en sí.

PLANTEAMIENTO

Los objetivos de la aplicación ya han quedado perfectamente definidos, por lo que antes de desarrollar las funcionalidades implementadas y el desarrollo de las mismas, conviene introducir el planteamiento adoptado de cara a cumplir estos requerimientos iniciales.

El punto de partida de la aplicación ha sido el de acceder a los sensores embebidos en el dispositivo de cara a utilizarlos para registrar aceleraciones. En concreto se busca el uso del acelerómetro. Esto permitiría usar el dispositivo de manera "*standalone*", como dispositivo funcional independiente, sin necesidad de conexión a ningún servidor o hardware de procesamiento externo.

Pero el hecho de acceder al sensor acelerómetro del dispositivo y almacenar los registros del mismo "en bruto", es algo que está resuelto en numerosas aplicaciones disponibles en Google Play, como podemos ver en la figura 2.1.2.1 tras una simple búsqueda.

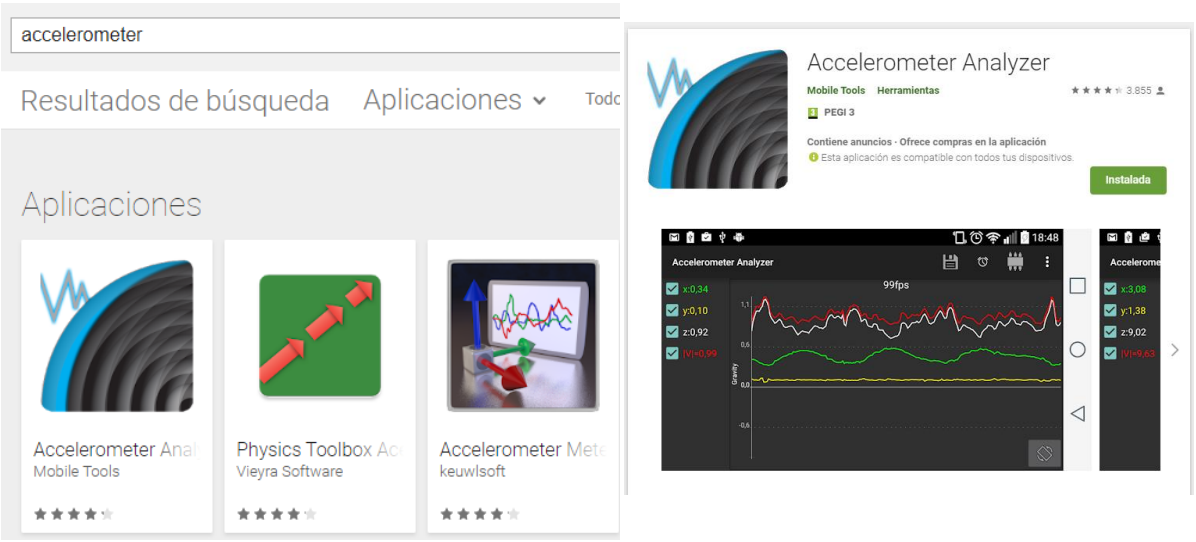


Figura 2.1.2.1-A la izquierda, búsqueda en Google Play de aplicaciones que registran o hacen uso del acelerómetro. A la derecha, Aplicación disponible en Google Play que permite registrar y almacenar la aceleración.

Considerando lo anterior, no tiene mucho sentido ni interés desarrollar una aplicación que simplemente registre, puesto que no sólo es un problema ya resuelto, sino que además se queda corto de cara a cumplir el objetivo del presente Trabajo de Fin de Máster, que es el de sustituir en aplicaciones de registro de estructuras a equipo de laboratorio, más profesional. Por estos motivos se plantea que la aplicación para smartphone realice algunas tareas de procesado y no sólo registro, lo que permite que el Smartphone se convierta en una herramienta útil para medir vibraciones en el terreno y sacar conclusiones sobre el mismo en base a los resultados que la aplicación nos muestre una vez procesado el registro.

Figura 2.1.2.2-Representacion del planteamiento inicial del dispositivo como registrador y procesador.

Una vez resuelto el planteamiento inicial, el de registrar con los sensores del móvil y realizar determinadas tareas de procesado como se representa en la figura 2.1.2.2, se abre la posibilidad de implementar también el registro mediante hardware externo, en este caso un Arduino, que envié mediante bluetooth los registros provenientes de un acelerómetro para que sea el smartphone el que procese, gestione y almacene esos datos, utilizando el mismo código de almacenado y procesado ya desarrollado e implementado en el planteamiento inicial. Este planteamiento complementario se corresponde a lo esquematizado en la figura 2.1.2.3.

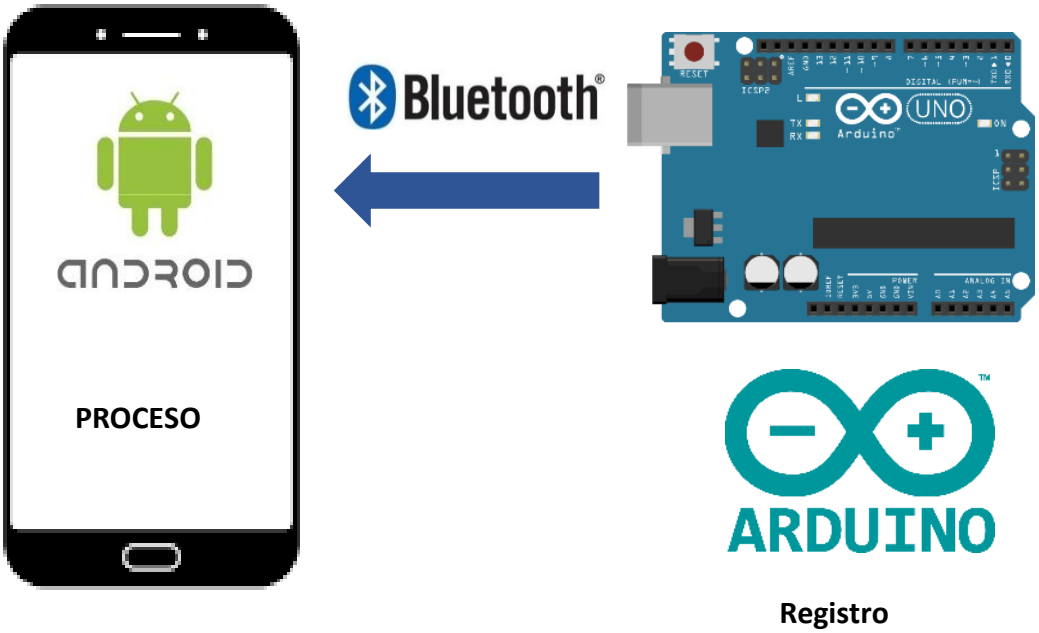


Figura 2.1.2.3-Representación del planteamiento alternativo de la aplicación

FUNCIONALIDADES Y USO

La primera y fundamental funcionalidad de la aplicación es, como ya se ha comentado, la de registrar la aceleración, pero esta no ha sido la única. Las funcionalidades implementadas en la versión final de la aplicación son las siguientes:

- Registro en tiempo real
- Selector de frecuencia de registro
- Selectores de ejes de representación
- Resampleado o remuestreo del registro
- Carga y visualización
- Detrend o eliminación de la tendencia lineal
- Selector de aceleraciones absolutas o relativas (efecto de la gravedad
- Cálculo y representación de FFT (Transformada Rápida de Fourier)
- Filtro Butterworth, tanto sobre una muestra almacenada como en tiempo real
- Cálculo y representación de la RMS Trend y sus máximos (MTVV)

Todas estas funcionalidades se muestran ubicadas en la aplicación en las siguientes figuras:

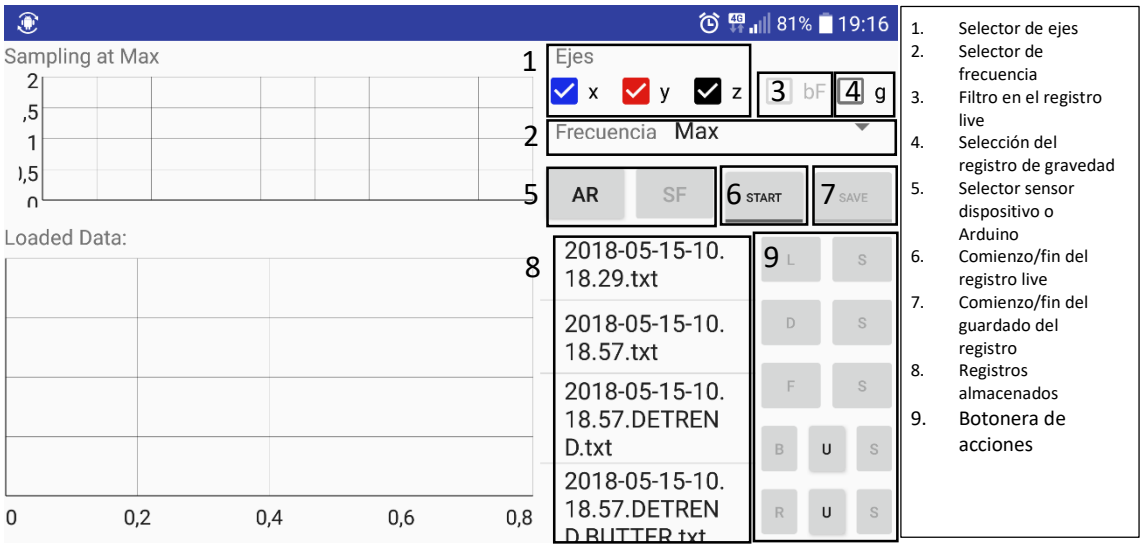


Figura 2.1.2.4-Leyenda de la parte interactiva de la interfaz gráfica de la aplicación

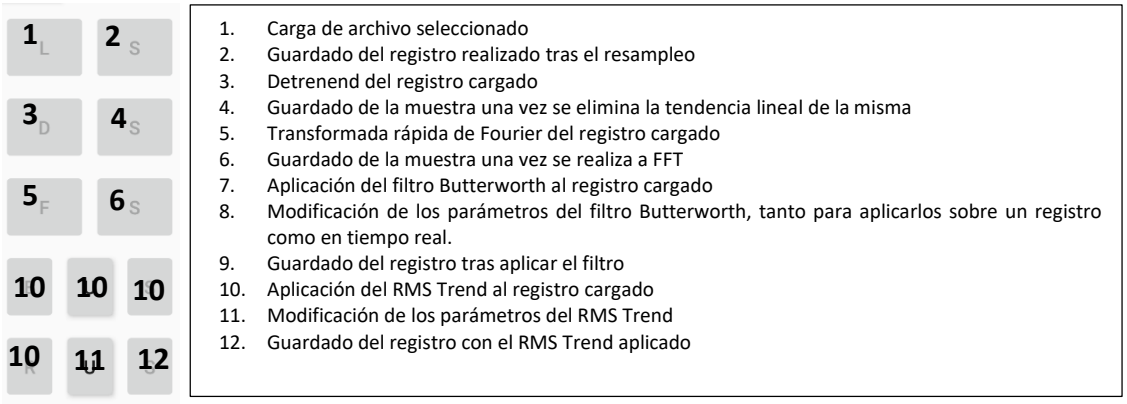


Figura 2.1.2.5-Leyenda de la botonera de acciones

Con las funcionalidades introducidas, pasamos a comentar el uso de las mismas:

REGISTRO EN TIEMPO REAL

La primera y más evidente de las funcionalidades introducidas en la aplicación es la del registro "live", el guardado de los registros del sensor en un archivo. El inicio del registro comienza al activar el botón de START, que se trata de un *"toggle button"* o botón de activación, que se quedará activado hasta que pulsemos sobre él otra vez. Durante el tiempo que está activo este botón se habilita la opción de guardado. Estas opciones se pueden ver en la figura 2.1.2.6.

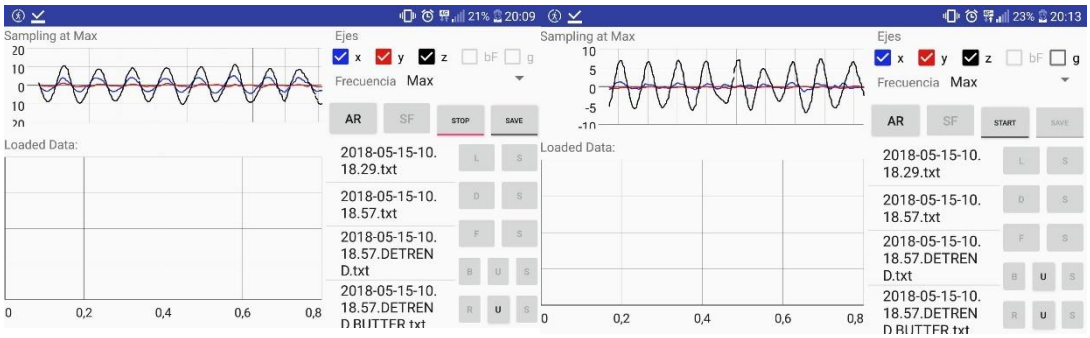


Figura 2.1.2.6-Registro en tiempo real: a la izquierda cuando comienza, habilitando la opción de guardar, y a la derecha con el registro pausado.

Una vez esté activado este botón, se habilitará el botón de SAVE, que funciona análogo al anterior, puesto que también es un *"toggle button"*. Al pulsar sobre él una primera vez se comienza a grabar. Al finalizar, en función del tipo de registro que se esté haciendo (básicamente si lo hace a la frecuencia máxima que es capaz de alcanzar el dispositivo o a una frecuencia seleccionada), ejecuta el guardado directamente o primero procesa el registro para ajustarlo a la frecuencia seleccionada, y ya entonces habilitará el botón para guardar definitivamente el registro. Esto lo podemos ver en la figura 2.1.2.7.

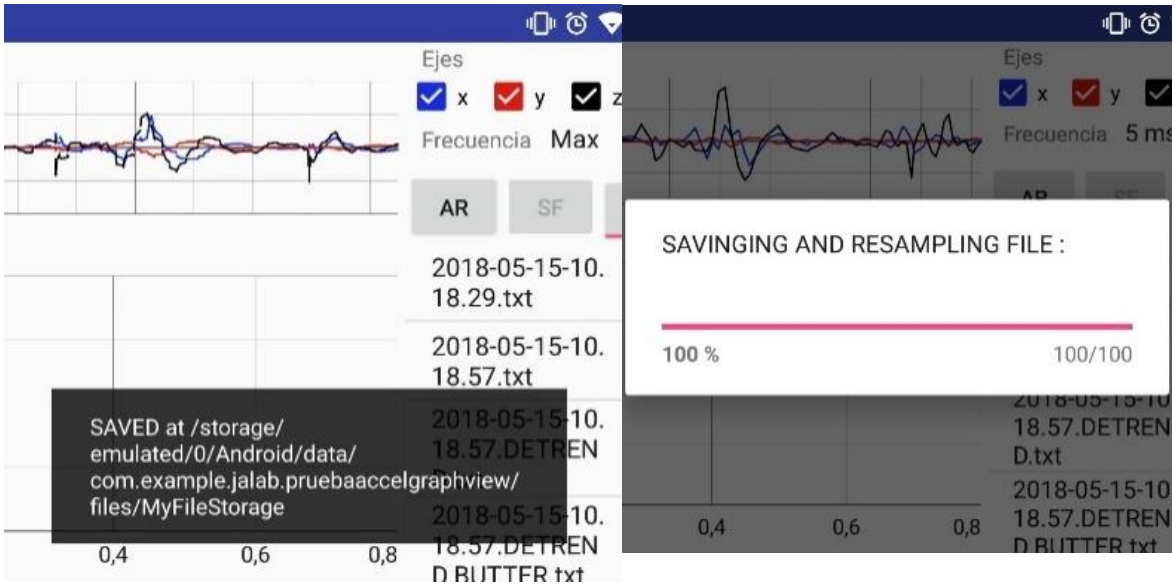


Figura 2.1.2.7-Registro en tiempo real: a la izquierda en el caso de muestreo a máxima frecuencia, a la derecha el guardado con frecuencia definida, que resamplea y habilitan la opción de guardado final.

También existe la opción, mientras se trabaje con el sensor del móvil, de elegir si queremos registrar la gravedad o no, mediante una "Check Box" con la opción correspondiente. De esta manera se deja a elección del usuario si desea registrar o no la gravedad, en función de sus requerimientos. En todo caso, de realizarse el registro con la gravedad y quererla retirar posteriormente, está disponible la opción de eliminar la tendencia lineal posteriormente como función de postprocesado de la señal, como se explicará en el apartado correspondiente.

Figura 2.1.2.8-Registro en tiempo real: a la izquierda con la gravedad eliminada, a la derecha con ella activa.

Además, la entrada de datos "live" puede llevarse a cabo tanto a través del propio sensor del móvil como de un señor externo. En el caos del presente trabajo este sensor externo se trata, como ya se ha comentado, de un Arduino que registra la aceleración mediante un acelerómetro y la envía por bluetooth. Para alternar entre ellos se deberá pulsar el botón AR, que dará paso a una pantalla de diálogo que nos permitirá introducir la dirección MAC de este dispositivo bluetooth, y de ahí conectarse. Nótese que primero deberá activarse el bluetooth del dispositivo de manera manual para que se pueda establecer la comunicación. Una vez se ha hecho esto, la entrada de datos de aceleración ya corresponderá al registro enviado desde el Arduino.

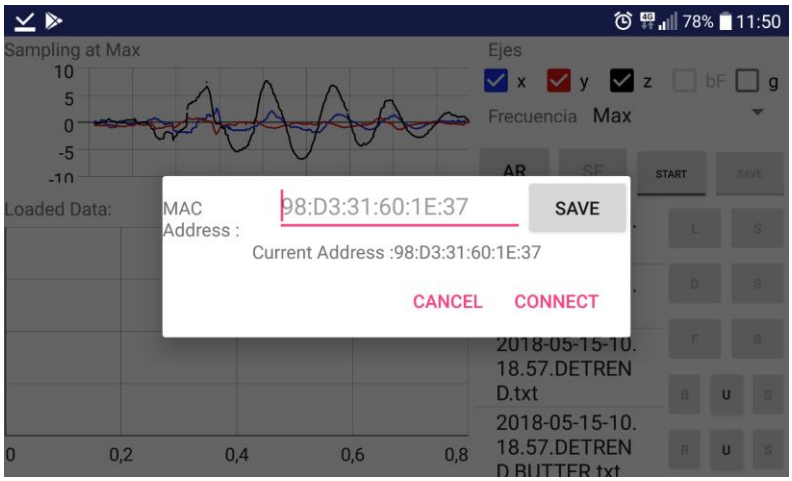


Figura 2.1.2.9-Pantalla de diálogo para la conexión con el Arduino, que permite modificar la dirección MAC a la que se va a conectar.

Como consideración previa a realizar esta conexión, el Arduino debe ser emparejado con el dispositivo móvil previamente, introduciendo la contraseña del mismo (que depende del modelo, pero por defecto son "0000" o "1234").

En cuanto a los aspectos relativos al Arduino, serán tratados en un apartado específico para él.

Por otra parte, el registro se puede llevar a cabo a la velocidad máxima que sea capaz de registrar el sensor o a una frecuencia definida. En el caso de hacerlo a una frecuencia definida, se habilita también la opción de filtrado de la señal en tiempo real. En este caso se trata de un filtro Butterworth de "lowpass", con un orden y frecuencia de corte especificado. Esto se verá en más detalle cuando se comente la utilidad del filtro.

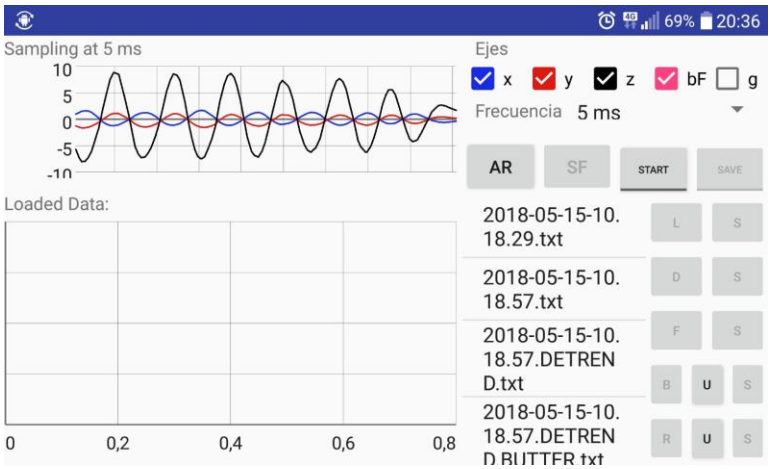


Figura 2.1.2.10-Registro en tiempo real con el filtro Butterworth activo. Vemos como se suaviza la medida.

En resumen, la utilidad de registro presenta las siguientes utilidades:

- Comienzo y pausado de la **entrada de registros** de aceleración: Pulsando botón START START: habilita la opción de guardar y deshabilita la selección de registro con o sin gravedad. Tras esto, se detiene pulsando el botón STOP STOP. Nótese que durante la ejecucción de la entrada de registros no se podrá activar o desactivar el filtrado en tiempo real.
- **Guardado de registros**: mediante el botón SAVE SAVE, que pasará a SAVING SAVING, indicando que esta almacenando los registros. Nótese que durante el guardado se deshabilita el selector de frecuencias y la opción de activar/desactivar el filtro en tiempo real. En el caso de tratarse de un registro a frecuencia máxima, el guardado tras este proceso es automático. Sin embargo, si se selecciona una frecuencia a la que registrar, se habilita un botón para guardar el registro, el botón S L S, contiguo al de carga.
- **Registrar o no la gravedad**: mediante la check box ☐ g, que en el caso de estar utilizando el sensor del móvil podemos seleccionar si detectar la aceleración con la aceleración o sin ella. Esta opción está habilitada cuando el registro esta pausado.
- **Filtro en tiempo real**: mediante la check box ☑ bF, que activa o desactiva el filtrado. Esta ocpión se encuentra activa en el caso de trabajar con una frecuencia máxima distinta de Max y se puede activar/desactivar siempre que no se esté guardando.
- **Selección de frecuencia**: accesible mediante el "spinner" Frecuencia 5 ms, que despliega una serie de opciónes de frecuencias disponibles. No se puede cambiar durante el guardado.
- **Selector de sensor**: nos permite alternar entre Arduino AR y Smartphone SF como sensor de medida.

CARGA Y VISUALIZACIÓN DE REGISTROS ALMACENADOS

Otra de las funcionalidades básicas de la aplicación es la de cargar un registro almacenado para su visualización. Esto se ejecuta seleccionando el nombre del archivo que se quiera cargar, apareciendo en pantalla mediante un "Toast" el nombre del archivo a cargar, como se muestra en figura 2.1.2.11. Tras hacer esto, se habilitará la opción de cargar el archivo mediante el botón L.

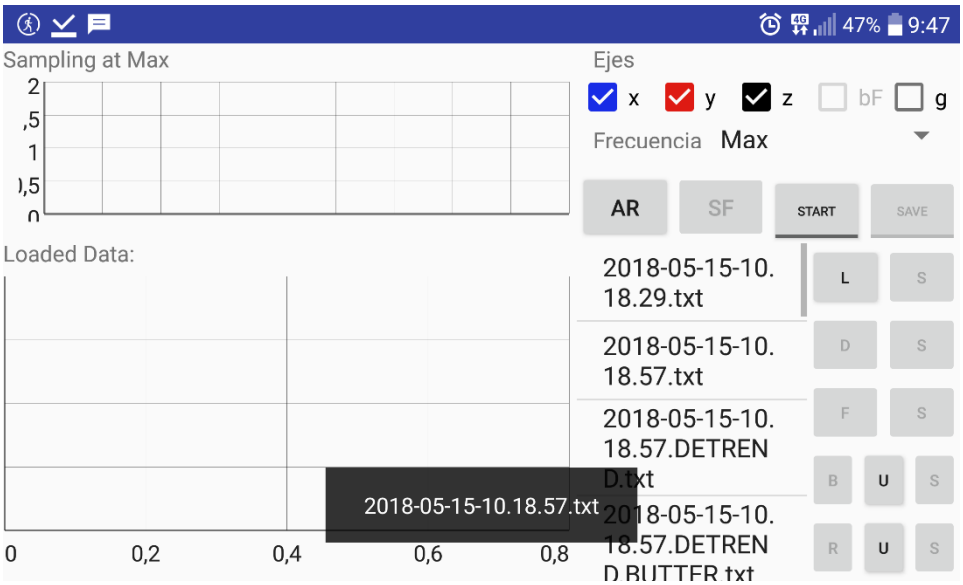


Figura 2.1.2.11-Archivo seleccionado para cargar. Vemos el quemado en pantalla con el nombre del archivo, así como el botón de carga L habilitado.

Tras seleccionar el archivo ya podemos cargarlo. El cargado se ejecuta mediante una barra de carga, que tras finalizar mostrará el registro en el gráfico habilitado a tal efecto en la parte inferior izquierda de la pantalla. Tras esto, se habilitarán los botones en la parte derecha de la pantalla que dan acceso a las operaciones que se pueden realizar sobre este archivo cargado. Ambas acciones se pueden ver en la figura 2.1.2.12.

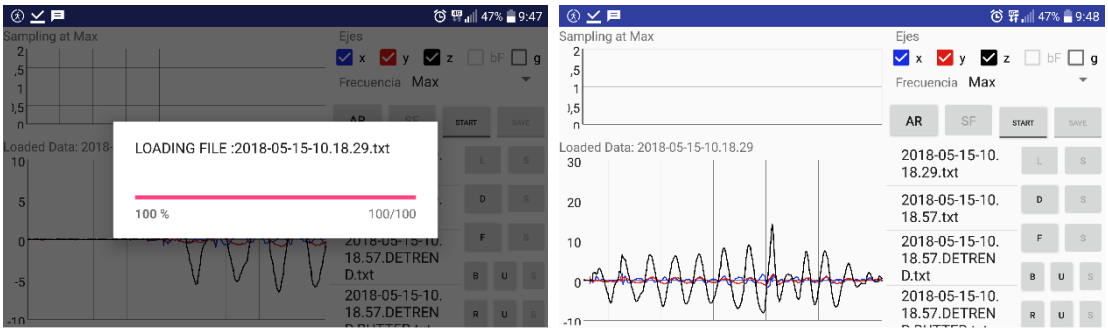


Figura 2.1.2.12-Cargado del archivo: A la izquierda, el proceso de cargado. A la derecha, el registro ya cargado y las operaciones habilitadas en la botonera a la derecha.

DETREND

La primera de las operaciones disponibles, es la de eliminar la tendencia lineal del registro. Esta operación es de utilidad en el caso de tratarse de un registro efectuado con la medición de gravedad activa, o a partir del sensor del Arduino. Se ejecuta mediante el botón D . En el caso de efectuar la operación de registro con el sensor del teléfono sin la gravedad (con el sensor de aceleración lineal, como se verá más adelante), esta operación no tendrá efecto, ya que la tendencia lineal ya se elimina en la propia entrada de datos. El efecto de esta operación se puede ver en la figura 2.1.2.13.

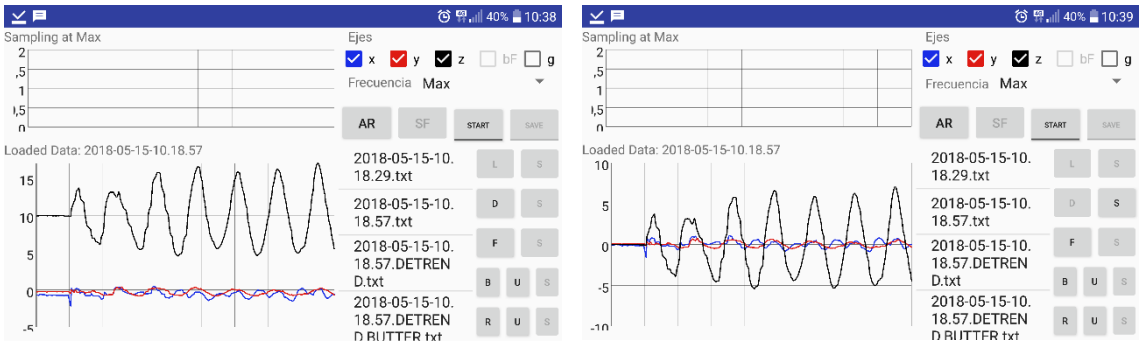


Figura 2.1.2.13-Capturas de la operación de Detrend: a la izquierda, el registro original. A la derecha, el registro tras eliminar la tendencia lineal de la señal. También podemos ver como se ha habilitado la opción de guardado de este nuevo registro, y de cómo se guarda con la extensión DETREND.

Tras ejecutar esta operación, se habilitará la opción de guardar este nuevo registro, que se almacenará con el nombre anterior añadiendo al final la extensión DETREND.

FFT: TRANSFORMADA RÁPIDA DE FOURIER

Otra operación disponible a realizar sobre el registro es la de trasladarlo del dominio del tiempo al de la frecuencia, de cara a analizar cual o cuales son las frecuencias dominantes en el registro realizado, algo fundamental entre los objetivos de la aplicación, como ya se ha introducido en los apartados correspondientes.

Tras cargar un registro, se nos habilita la opción de ejecutar esta operación, pero cabe destacar que **sólo se podrá ejecutar si la operación de registro se ha realizado a una frecuencia determinada y no a la máxima**. Esto es así porque se necesita que el registro este temporalmente equiespaciado para llevar a cabo la operación, y que la implementación del registro en tiempo real se ha implementado como registrador "en bruto", y no con la finalidad de que ese registro a frecuencia máxima sea operable desde el terminal.

Por último, destacar que esta operación funciona como la descrita anteriormente (y como las que se describirán a continuación): se ejecuta pulsando el botón F , habilita la opción de guardado (botón S contiguo al botón F de la operación), y el nombre del archivo guardado corresponde a al nombre del archivo sobre el que se hace la operación con la extensión FFT. Se pueden ver capturas de esta operación en la figura 2.1.2.14.

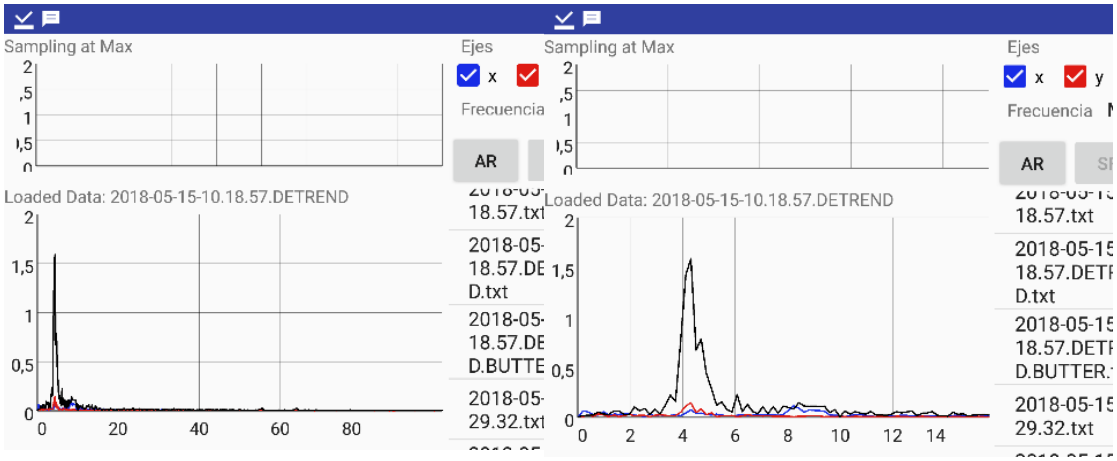


Figura 2.1.2.14-Capturas de la operación de FFT: A la izquierda, el rango completo de frecuencias que se abarcan, a la derecha ampliado, sonde vemos donde se encuentran las frecuencias dominantes del registro.

Un aspecto a destacar sobre esta operación es que, debido a cómo ha sido programada, no se ha podido implementar una “progressbar” que muestre el estado de ejecución, con lo que en registros largos el programa se “congela” durante la ejecución de la operación. Se aconseja al usuario que espere durante esos segundos, puesto que al terminar todo vuelve a la normalidad, con el cálculo realizado.

FILTRO BUTTERWORTH

Como ya se comentó en el apartado referente al registro “live”, existe la opción de aplicar un filtro al registro. El filtro que se ha seleccionado para implementar ha sido un filtro Butterworth de paso bajo, por sus propiedades en frecuencia.

La primera opción disponible en cuanto a esta operación es la de introducir las propiedades de este filtro, que se llevará a cabo mediante la pantalla de diálogo que se despliega tras pulsar el botón U contiguo al botón B correspondiente a esta operación. La ventana de dialogo desplegada es la que se muestra en la figura 2.1.2.15, y habilita al usuario la opción de seleccionar el orden del filtro e introducir la frecuencia de corte “cutoff frec”. Estos datos se corresponden tanto al filtro que se ejecuta sobre la señal “live”, como el filtro que se aplica sobre un registro.

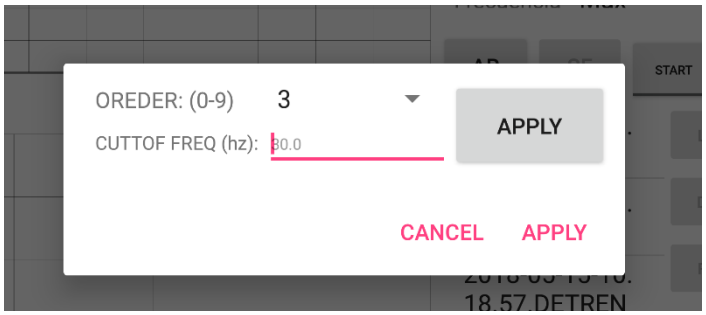


Figura 2.1.2.15- Ventana de diálogo que permite introducir la frecuencia de corte del filtro y seleccionar el orden del mismo.

Una vez se han determinado las características del filtro, se puede aplicar sobre el registro cargado, de nuevo sólo si se trata de un registro realizado a frecuencia definida. Cabe destacar que en este caso el filtro **sólo se podrá ejecutar si la frecuencia de corte seleccionada es menor que la mitad de la frecuencia de muestreo**, siendo esto así por las propiedades del filtro. Si se intenta ejecutar el filtro de la manera descrita, ocurre lo que se muestra en la figura 2.1.2.16.

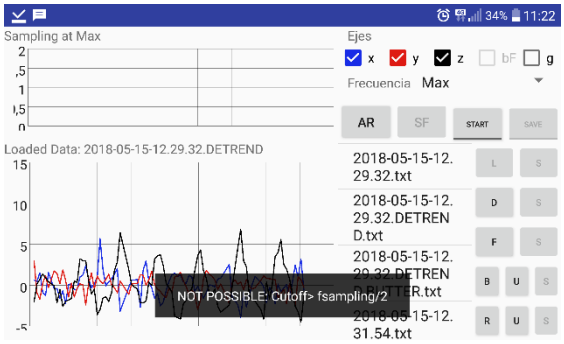


Figura 2.1.2.16-Captura del mensaje de error al intentar usar una frecuencia de corte demasiado

En cuanto al funcionamiento, es análogo a los descritos anteriormente: pulsar B para ejecutar, lo que habilita la opción de guardado. El archivo guardado tras esta operación lo hace con el nombre del archivo empleado añadiéndole la extensión BUTTER.

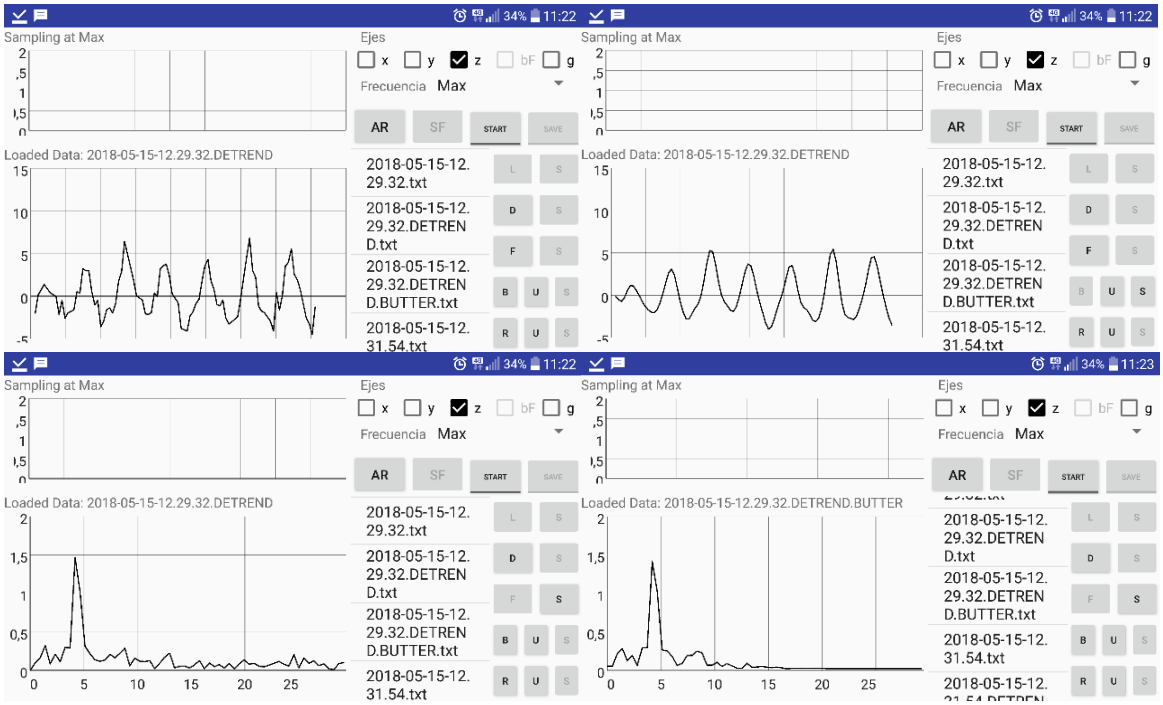


Figura 2.1.2.17-Capturas de la operación de aplicación del filtro: a la izquierda, la señal original con bastante ruido (se trata del sensor del Arduino), con su FFT que remarca la presencia de este ruido. A la derecha, la señal filtrada claramente suavizada, con la FFT que muestra la acción del filtro eliminando progresivamente las frecuencias a partir de 10 hz (frecuencia de corte introducida al filtro).

RMS TREND

Mediante esta operación calculamos la media cuadrática en forma de tendencia y el máximo de la misma, correspondiente a los MTVV de cada eje medido.

Esto se lleva a cabo de la manera descrita también en las operaciones anteriores. Primero introducimos mediante el botón U los parámetros referentes a la ventana de calculo por la que se desplaza esta media cuadrática como son el tamaño de la misma (win) y su desplazamiento (T2), y su introducción se realiza a través de la pantalla de diálogo que se muestra en la figura 2.1.2.18.

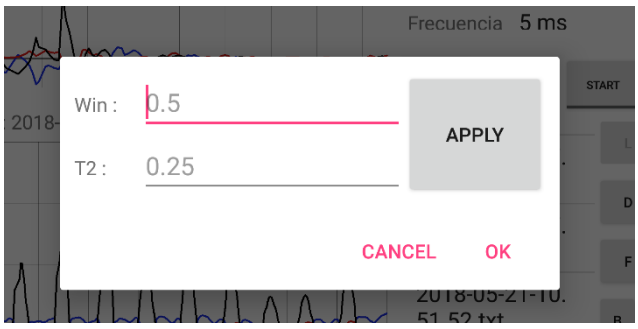


Figura 2.1.2.18-Ventana de diálogo para introducir los parámetros de la ventana de cálculo del RMS Trend.

Con los parámetros definidos, el procedimiento de cálculo es análogo a lo visto hasta ahora: botón R ejecuta el cálculo y lo muestra en la gráfica, después se habilita el guardado, que añade al nombre original la extensión de RMS_Trend.

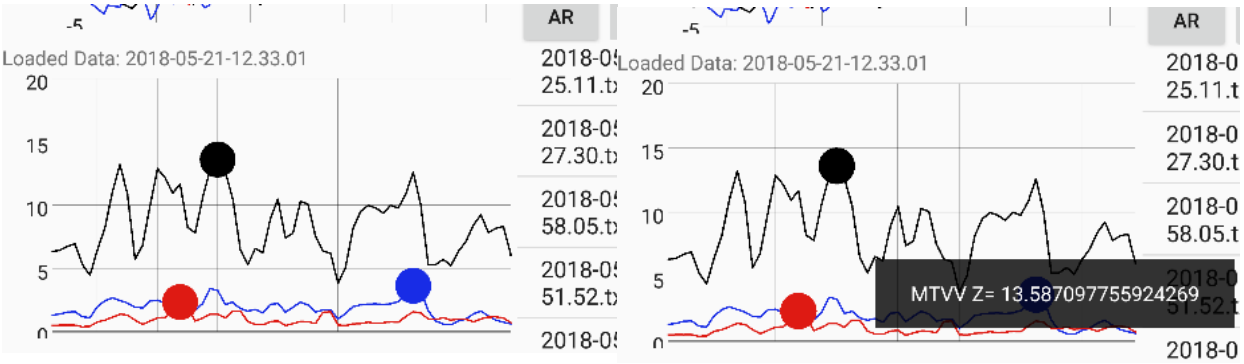


Figura 2.1.2.19-Resultado del RMS Trend: Se señalan sobre la gráfica los valores máximos. Al "clickar" sobre ellos, nos aparece en pantalla el valor del mismo

RECOMENDACIONES DE USO

Para terminar la descripción de las utilidades de la aplicación para smartphones Android, una breve recomendación en cuanto al uso de la misma: es recomendable que si se desea realizar operaciones sucesivas sobre el mismo registro (por ejemplo, aplicar un filtro Butterworth para después realizar una FFT), se guarden las operaciones intermedias y se cargue el nuevo registro operado para para ejecutar sobre él la nueva operación. De esta manera se asegura que todas las operaciones se ejecutan correctamente.

DESARROLLO DE LA APLICACIÓN

En la presente sección se van a comentar diferentes aspectos del desarrollo de la aplicación, centrados principalmente en el planteamiento adoptado para la implementación de las utilidades y no tanto en la implementación en sí, puesto que los códigos incluidos en el anexo se encuentran debidamente comentados. El objetivo de esta sección es, por tanto, dar unas nociones sobre cómo se ha desarrollado la aplicación de cara a que el presente TFM pueda ser ampliado en un futuro y pueda ampliarse y mejorarse fácilmente la propia aplicación.

Como ya se ha introducido en apartados anteriores, la aplicación se ha desarrollado sobre el sistema operativo Android y mediante la IDE Android Studio, y el lenguaje de programación utilizado es Java.

Un objetivo destacado en el desarrollo de la aplicación ha sido el que esta fuera funcionalmente sencilla, con lo que se ha evitado usar pantallas y transiciones y se ha condensado todo lo posible las funcionalidades en una pantalla. Esto ha sido así porque, aunque a nivel estético la presencia de pantallas y transiciones hubiera sido positivo, a nivel de programación habría supuesto un esfuerzo mayor al haber tenido que hacer uso de más partes del ciclo de vida de las actividades sobre el que se sustenta este sistema operativo, aparte de que hubiera supuesto la necesidad de parcializar los objetos para trasladarlos entre las distintas actividades del programa, aumentando una carga de trabajo del dispositivo que, como veremos más adelante, ya resulta suficientemente elevada si los registros son largos.

SENSORES Y ACCESO A LOS MISMOS

En cuanto a los sensores propios del dispositivo, el sistema operativo Android nos permite el acceso a un gran número de ellos, que estarán o no disponibles en función del dispositivo sobre el que se trabaje. Podemos ver la lista de sensores a los que se pueden acceder que la documentación nos facilita en la tabla 2.1.2.1.

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.
TYPE_ORIENTATION	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the getRotationMatrix() method.	Determining device position.
TYPE_PRESSURE	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with the TYPE_AMBIENT_TEMPERATURE sensor in API Level 14	Monitoring temperatures.

Tabla 2.1.2.1-Lista, descripción y uso de los sensores accesibles en un dispositivo Android. Fuente: https://developer.android.com/guide/topics/sensors/sensors_overview

De esta lista de diferentes sensores, resultan de interés para la aplicación dos: TYPE_ACCELEROMETER y TYPE_LINEAR_ACCELERATION. Estos, de acuerdo a la documentación do Google nos dan acceso a lo siguiente:

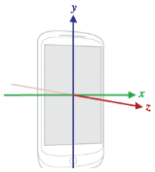


Figura 2.1.2.20- Ejes de un dispositivo estándar.

- **TYPE_ACCELEROMETER**: mide la aceleración en m/s^2 que se aplica sobre el dispositivo en los tres ejes (x, y, z), incluyendo la gravedad.
- **TYPE_LINEAR_ACCELERATION**: Análogo, pero sin gravedad. De acuerdo a la documentación, se supone que este sensor siempre tendrá un offset que es conveniente calibrar, pero en los dispositivos a los que se ha tenido acceso durante el desarrollo de esta aplicación no ha sido así, puesto que este offset se elimina solo. Es conveniente en cualquier caso tener esto en cuenta, especialmente si se trabaja con dispositivos más antiguos que pudieran no eliminar este offset de manera automática.

Por tanto, habiendo establecido el acceso a estos dos sensores, se ha decidido habilitar la opción de elegir usar uno u otro al usuario de la aplicación, puesto que el uso del TYPE_LINEAR_ACELERATION puede no responder a lo que se busca de la aplicación debido a que no está perfectamente definido como funciona. De acuerdo a la documentación, este sensor hace uso tanto del acelerómetro como del giróscopo, pero deja a criterio del fabricante su implementación, con lo que en cada dispositivo puede funcionar de distinta manera y sin estar perfectamente definidas las operaciones que se realizan, con lo que queda a criterio del usuario si seleccionar este sensor, que por otra parte puede ser muy útil, o no hacerlo.

Una vez se ha seleccionado e implementado el acceso a los sensores, se procede a registrar. Para ello sobrescribimos el método "onSensorChanged", cuyo acceso nos queda habilitado tras implementar la interfaz "SensorEventListener":

Implementación de los sensores y registro de los mismos

```
  // Se implementa el acceso a los sensores mediante la Interfáz
SensorEventListener
  public class MainActivity extends AppCompatActivity implements
SensorEventListener{
        … Código implementado en la MainActivity…
}
  // Se sobrescribe el método para registrar la aceeración
@Override
public void onSensorChanged(SensorEvent event) {
    // Código de registro....
  }
```

Mediante la sobreescritura de este método lo que se consigue es que cada vez que se detecte un cambio en la lectura del sensor de aceleración se lance el código escrito, lo que viene a ser el registro. Esto es lo que se conoce como programación por eventos, siendo el evento de detección de cambio en la aceleración el que lanza la ejecución del código.

Este método de acceso a los sensores permite además definir la velocidad a la que este evento puede ser lanzado, especificando el "delay" con el que se ejecuta esta función en la instanciación de los objetos sensor:

Instancia y definición del sensor

```
   sensorType[0]=Sensor.TYPE_LINEAR_ACCELERATION;
 sensorType[1]=Sensor.TYPE_ACCELEROMETER;
mAccelerometer = mSensorManager.getDefaultSensor(sensorType[0]);
mSensorManager.registerListener(this,mAccelerometer,SensorManager
       .SENSOR_DELAY_FASTEST);
```

Como vemos, se especifica la velocidad como ***SENSOR_DELAY_FASTEST***, siendo esta la velocidad máxima. De acuerdo a la documentación de Google, esta velocidad introduce un delay de 0 microsegundos a la función, con lo que se ejecuta a la máxima velocidad posible. Esta velocidad dependerá en última instancia del dispositivo que se utilice, puesto que será la calidad de su sensor y el software que lo acompañe el que delimitará el ritmo. Por ejemplo, en el dispositivo utilizado para desarrollar la aplicación y realizar numerosas pruebas, un LG g5 del 2016, las características del sensor, obtenidas accediendo por código al mismo, son las siguientes:

```
sensor Vendor: BOSCH
sensor Name: LGE Accelerometer
sensor Resolution: 0.0023956299m/s^2; Max Range: 78.4532; Min Delay: 5000 µs
```

Con estas características y tras realizar numerosas pruebas, se ha determinado que el sensor es capaz de registrar datos a una velocidad máxima de aproximadamente 4,9 ms, 201 hz, de media, con lo que a la luz de los resultados estas características obtenidas por software tampoco son rigurosas y dependerán de más factores, pero dan una idea aproximada de las capacidades del software, ya que se ajustan bastante a lo obtenido en las pruebas.

Esto en cuanto a la ejecución a frecuencia máxima, pero también cuando se ejecuta a otras frecuencias, puesto que la solución adoptada para poder muestrear a una frecuencia determinada ha sido la de ejecutar un “resampleo” o “remuestreo“ al registro una vez se finaliza el periodo de muestreo, con lo que la toma de registros no varía de un caso a otro más allá de su representación gráfica, que sí varía por consideración estética.

GUARDADO DE REGISTROS

Una vez se establece el sensor, el siguiente paso es almacenar ese registro. A nivel interno, estos registros se almacenan en una colección de objetos que apila cada uno de los datos de la aceleración, que a su vez forman también un objeto. Esto es así mientras se encuentra en el programa, pero el objetivo de las aplicaciones es que los archivos guardados sean intercambiables, con lo que se ha optado por guardar estos registros en un archivo de texto (.txt), tal y como se muestra en la figura 2.1.2.21, de fácil acceso tanto desde el dispositivo Android como desde Matlab, además de que se puede exportar fácilmente a otros programas como Excel por si quisiera visualizar desde ahí.

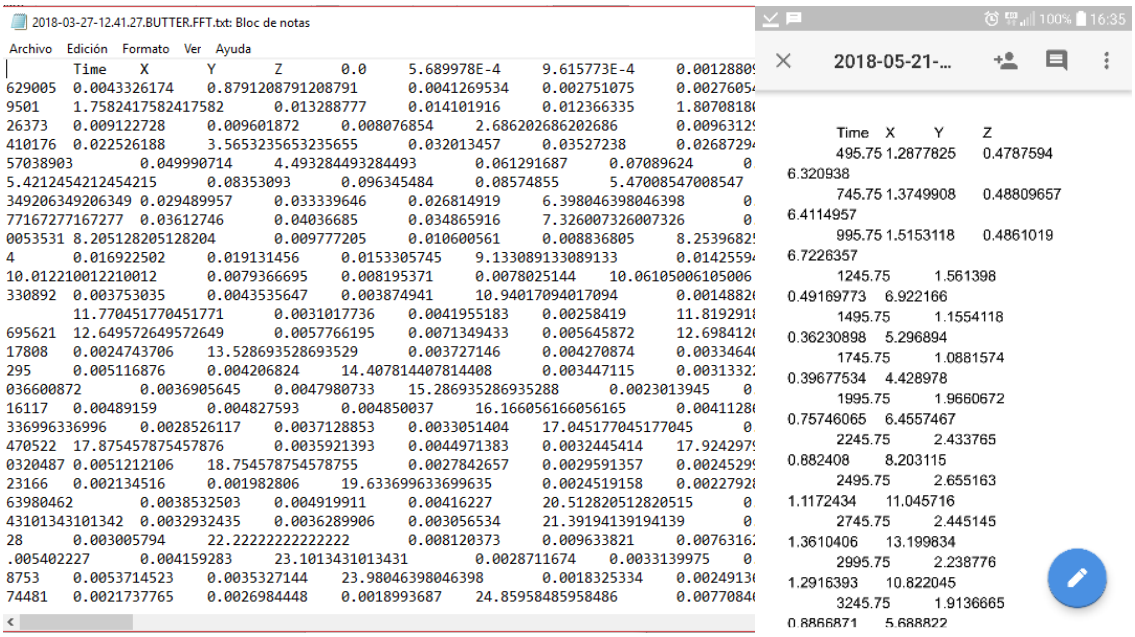


Figura 2.1.2.21-Archivo .txt con la información del registro almacenada: a la izquierda visto desde el ordenador. A la derecha desde el propio smartphone.

Estos archivos de texto son almacenados por la aplicación en la carpeta del dispositivo dispuesta a tal efecto, desde la que se tendrá acceso a todos los registros que se realicen, y que ser leído o enviado por diferentes métodos, como se muestra en la figura 2.1.2.22.

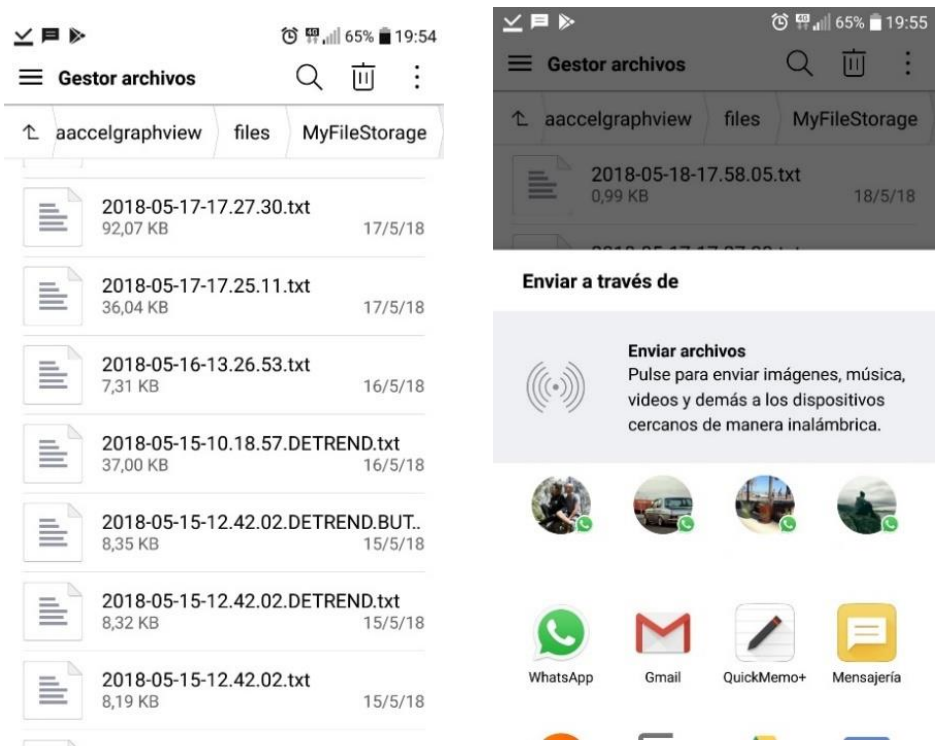


Figura 2.1.2.22-Acceso a los archivos txt almacenados. Gestión del envío de los mismos.

En cuanto a la implementación en código de la ejecución del guardado, el método utilizado ha sido el de crear un string en la colección en la que se apilan los objetos aceleración,

añadiendo cada dato a la vez que se apila el objeto. Así queda en forma de string y es más rápido pasarlo a un archivo de texto.

Código de guardado de datos de aceleración

```
stringData="\t"+"Time"+"\t"+"X"+"\t"+"Y"+"\t"+"Z"+"\n";

   public void add(AccelDataSimple acc){
  set(acc, length);
  stringData=stringData+"\t"+
          String.valueOf(acc.getTime())+"\t"+
          String.valueOf(acc.getX())+"\t"+
          String.valueOf(acc.getY())+"\t"+
          String.valueOf(acc.getZ())+"\n";
```

En el caso de que el registro a guardar se halla realizado a una frecuencia determinada y no la máxima, el proceso de guardado se ejecuta tras un resampleo de la muestra para ajustarlo a la frecuencia pertinente, como ya se comentó en el apartado anterior. Esto se ejecuta mediante una interpolación lineal determinando los nuevos puntos en función de los anteriores. Sobre esto decir que, además, se ajustará a la capacidad de muestreo del sensor empleado. Esto es que, si por ejemplo se muestre con un sensor cuyo tiempo de muestreo medio es de 12,3 milisegundos por muestra y se selecciona una frecuencia de muestreo de 5 ms, el resampleo se ejecuta a 13 ms (el siguiente entero) y no a la determinada, de cara a no falsear las propiedades de la muestra para operaciones posteriores.

REPRESENTACIÓN GRÁFICA

En cuanto a las representaciones gráficas, se han programado con una librería externa, GraphView, publicada bajo licencia Apache v2, lo que significa que se permite su uso incluso en aplicaciones de código cerrado.

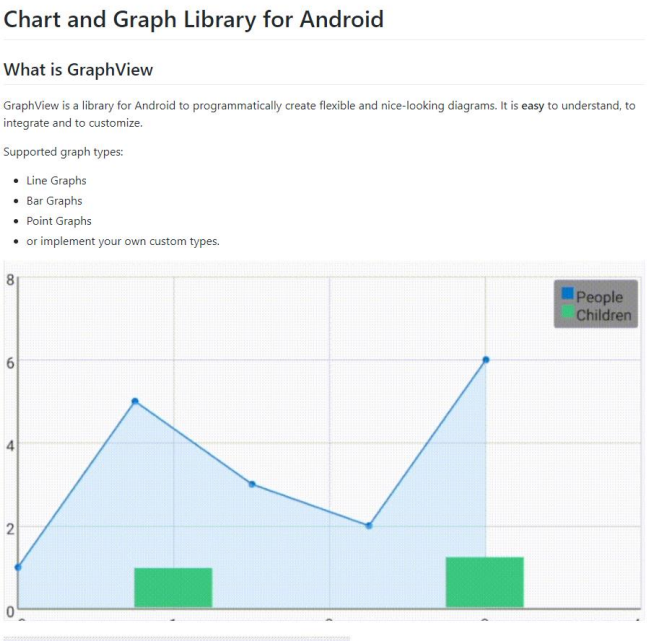


Figura 2.1.2.23-Librería GraphView para representaciones gráficas en Android: https://github.com/jjoe64/GraphView

CONEXIÓN, RECEPCIÓN Y PROCESADO DE DATOS DE ACELERÓMETRO EXTERNO MEDIANTE ARDUINO

Como ya se ha comentado, una de las opciones implementadas en la aplicación es la de conectar un sensor externo a través de un Arduino. Esta conexión se ha implementado mediante el uso de conexión bluetooth.

Sin entrar a comentar en mayor profundidad aspectos relativos a la propia programación de esta conexión, decir solamente que en el caso de esta aplicación solamente se ha **habilitado un socket de entrada**, puesto que solo se necesita recibir la información del Arduino y a este no se le envía nada, puesto que emite de continuo. La conexión con este Arduino se establece "*hardcodeando*" la dirección MAC del dispositivo bluetooth del dispositivo mediante una entrada de texto tal y como se especificó en el apartado de uso. A través de esta dirección y habiendo emparejado ambos dispositivos (Arduino y smartphone) previamente ya tenemos la conexión establecida.

Una vez se reciben los datos, a estos se les aplican una media móvil para suavizar la señal y ocupan el lugar de lo que antes era el sensor del dispositivo. Esta lectura de los datos procedentes del Arduino se ejecuta ahora en un "*thread*" o "hilo" de ejecución diferentes, que continúa ejecutándose en segundo plano permitiendo realizar otras tareas mientras se reciben los datos del sensor.

Cabe destacar que este método de entrada de datos es sustancialmente más lento que el de registro mediante el propio sensor, al menos en el caso de los dispositivos a los que se ha tenido acceso durante el desarrollo de este trabajo (un terminal Android de cierta calidad y un Arduino chino de calidad más dudosa). En el caso de habilitar los tres ejes del acelerómetro en el Arduino el tiempo de muestreo será de aproximadamente 16 milisegundos, reduciéndose hasta casi un tercio si sólo consideramos uno. Este elevado tiempo no se ha visto reducido al hacer pruebas sustituyendo el medio de transmisión de bluetooth por cable, con lo que el origen de este elevado tiempo seguramente esté en la lectura de las entradas analógicas y su digitalización, con lo que una posible línea futura de desarrollos será la de intentar mejorarlo.

ELIMINACIÓN DE LA TENDENCIA LINEAL

Para esto se ha optado por una solución sencilla, que es la de calcular la media del registro en un eje y restarla a cada uno de los valores. Esto ha sido así porque el uso que se le pretende dar es de registrador, con lo que se supone que, tras situarlo en su posición de medida, el dispositivo no se va a desplazar, con lo que se considera que esta forma de "detrend" es suficiente.

FFT

La implementación de este cálculo se ha llevado a cabo mediante código, programando. Para ello, se ha adaptado a las necesidades de la aplicación el código distribuido para este cálculo por la universidad de Princeton, disponible en el siguiente enlace:

https://introcs.cs.princeton.edu/java/97data/FFT.java.html (disponible a fecha 21/05/2018).

FILTRO BUTTERWORTH

Para este filtro se ha usado la librería open source IIRJ, distribuida bajo licencia Apache License 2.0.

Esta librería se encuentra disponible en:

https://github.com/berndporr/iirj/blob/Máster/LICENSE.txt (disponible a fecha 21/05/2018).

2.2-APLICACIONES CON MATLAB

A diferencia de la aplicación para smartphones ya descrita, la aplicación desarrollada para escrito mediante el uso del software Matlab surge de la posibilidad de aprovechar una serie de herramientas de software ya disponibles, ampliamente extendidas y con unas capacidades de cálculo y desarrollo ya conocidas.

Con esto en mente, el desarrollo mediante Matlab busca, por tanto, cumplir los mismos objetivos que la aplicación anterior, que son a grandes rasgos registrar aceleración mediante dispositivos de bajo coste o reutilizados y realizar determinadas tareas de procesado sobre los registros, pero ahora desde un prisma distinto: utilizar y adaptar para el uso específico del presente trabajo herramientas ya disponibles y desarrolladas por terceros.

Para llevar a cabo esto, se ha optado pues por el software Matlab principalmente por, como se comentó en la introducción del presente apartado, disponer no solo de herramientas de cálculo muy eficientes sino también la gran variedad de diferentes herramientas como "Add-ons" o "toolbox" que permiten desde el procesamiento de señales con funciones como detrend o fft, que en el caso anterior tuvieron que ser programadas e implementadas, hasta herramientas que permiten la conexión de dispositivos externos habilitando diferentes entradas, ya sea mediante bluetooth, wifi o incluso cable, y permitiendo la gestión de los mismos mediante programación orientada a objetos.

2.2.1-PLATAFORMA DE DESARROLLO

Tras la breve introducción anterior, conviene de nuevo tratar someramente algunas cuestiones referentes a este software como paso previo a tratar el desarrollo, funcionalidades y uso de la aplicación completa.

MATLAB

Este software se trata de un entorno de computación numérica multi-paradigma, capaz de comunicarse con diferentes programas y aplicaciones usando diferentes lenguajes. Además, mientras que otros lenguajes de programación generalmente trabajan con números como entidades únicas, Matlab lo hace directamente con conjuntos, en forma de "*arrays*" (podríamos decir vectores) o matrices (de ahí el nombre de Matlab, "laboratorio de matrices").

De acuerdo a la propia web de Matlab, "*MATLAB combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente*", con lo que nos encontramos ante un software que incluye no solo la programación optimizada para el tratamiento de array y matrices, sino que lo integra en un entorno de desarrollo (IDE) que

además permite el desarrollo de aplicaciones, conexión de aplicaciones externas e interacción con las mismas.

PROGRAMACIÓN EN MATLAB

En cuanto a la programación en Matlab, este software posee su lenguaje de programación propio, conocido como Lenguaje M, que es el comentado lenguaje especializado en el tratamiento de matrices y array.

Además de esto, Matlab permite interactuar con otros programas, aplicaciones y lenguajes. De acuerdo al fabricante, Matlab provee una integración flexible y bidireccional con otros lenguajes de programación, incluyendo:

- Instanciar Matlab desde otro lenguaje:

  Gracias al APIs de Matlab, este software se puede usar desde otros entornos de desarrollo. Este APIs habilita la ejecución de Matlaba desde otros lenguajes de programación sin la necesidad de arrancar una sesión de Matlab. Esto está disponible desde los siguientes lenguajes:

  - C/C++
  - Fortran
  - Java
  - Python

Figura 2.2.1.1-Interacción de Matlab con un entorno de programación externo a sí mismo. Fuente: https://es.mathworks.com/solutions/matlab-and-other-programming-languages.html

- Usar librerías escritas en otro lenguaje:

  En caso de requerir el uso de librerías, propias o ajenas, programadas en otro lenguaje (existen infinidad de ellas que facilitan el trabajo en infinidad de campos), pueden ser llamadas desde este mismo software. Algunas de las librerías a las que se pueden acceder:

  - Librerías Java
  - Librerías Python libraries
  - Funciones en C/C++ o Fortran MEX-file
  - Librerías compartidas C
  - Librerías .NET
  - Objetos COM
  - Servicios web RESTful and WSDL

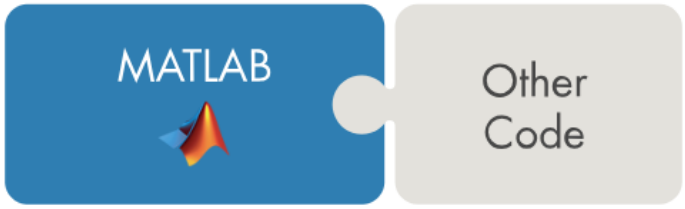


Figura 2.2.1.2-Interacción de Matlab con código externo. Fuente: https://es.mathworks.com/solutions/matlab-and-other-programming-languages.html

- Convertir código Matlab a C/C++:

  Otra interesante opción disponible a nivel programación es la de poder convertir algoritmos programados o disponibles en Matlab a C/C++ usando la aplicación distribuida por el fabricante en el propio software de MATLAB Coder™. Incluso se puede generar código desde Simulink, otra potente herramienta de Matalab para programación, en este caso mediante elementos gráficos.

  Con esta herramienta, es posible trasladar funcionalidades desarrolladas en Matlab a otros lenguajes que implementen C/C++, como puede ser el propio Android, con lo que es una funcionalidad importante y a tener en cuenta de cara a implementar funcionalidades más complejas en la aplicación desarrollada para smartphones.

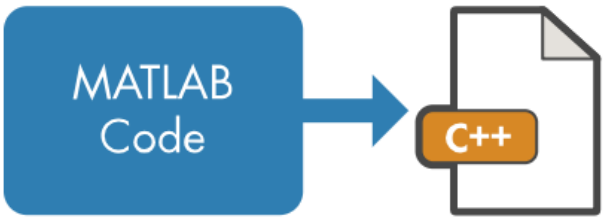


Figura 2.2.1.3-Matlab Coder para exportar código Matlab a C/C++. Fuente: https://es.mathworks.com/solutions/matlab-and-other-programming-languages.html

- Empaquetar programas en Matlab como componentes de software:

  Por último, Matlab también permite empaquetar programas desarrollados en el en componentes para otros softwares con lenguajes específicos, de manera que se pueda integrar en ellos. De nuevo otra característica relevante de cara a revisiones futuras de la aplicación móvil desarrollada ene l presente trabajo.

  Algunos componentes que se pueden desarrollar mediante este empaquetado son:

  - Librerías compartidas en C/C++
  - Microsoft® .NET assemblies
  - Paquetes Python
  - Clases Java®

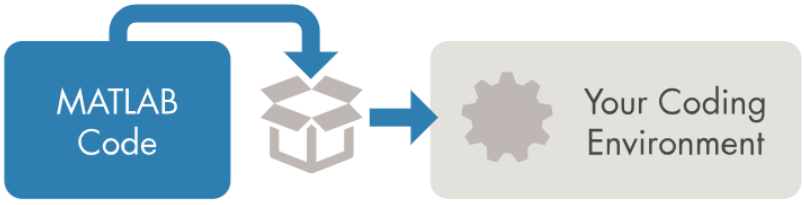


Figura 2.2.1.4-Matlab Coder para exportar código Matlab a C/C++. Fuente: https://es.mathworks.com/solutions/matlab-and-other-programming-languages.html

Esto resume las capacidades, características y ventajas que ofrecen la programación en Matlab, y que hacen de este software una opción excelente para desarrollar el presente apartado.

DESARROLLO DE APLICACIONES: GUIDE

A parte de la programación propia en Matlab, la parte relativa a los cálculos, también existe la posibilidad de desarrollar interfaces gráficas que permitan gestionar el conjunto de la

aplicación, permitiendo así funcionar como una aplicación cerrada y gestionar los cálculos programados de manera intuitiva para un usuario ajeno a la programación del código.

Para ello, existen fundamentalmente dos herramientas provistas por Mathwork para el desarrollo de estas interfaces gráficas: App Designer y GUIDE.

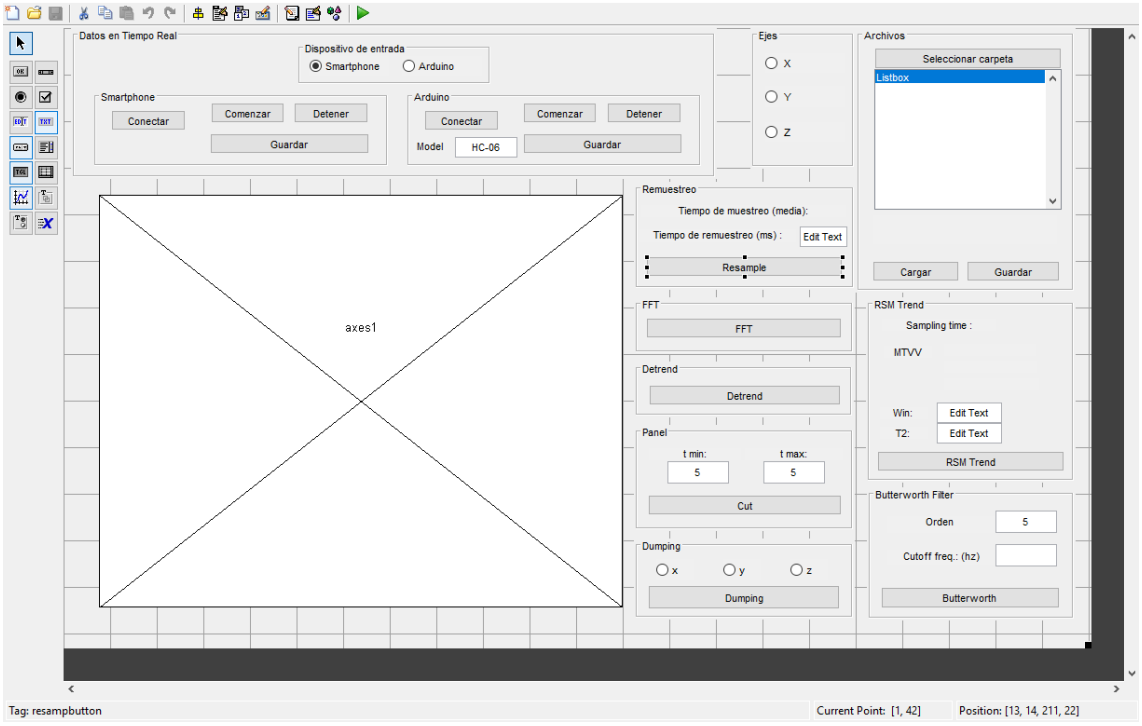


Figura 2.2.1.5-Recorte de pantalla del diseño mediante GUIDE de la interfaz de la aplicación

Para nuestro caso se ha optado por el uso de la segunda, GUIDE, a pesar de no ser la opción más moderna de las dos ni la que más funciones ofrece, principalmente por dos motivos. El primero, al ser GUIDE una opción más "trabajada", con más usuarios y documentación disponible, el desarrollo de aplicaciones mediante esta herramienta es más accesible para alguien sin conocimiento previo. El segundo, de acuerdo a la propia documentación ofrecida por Mathworks respecto a App Designer, "*this approach supports most apps that do not require polar plots, subplots, or graphics interactions such as mouse and key-press customizations*", lo que se resume en que no recomienda su uso en el caso de trabajar con determinados tipos de gráficos o interacción con ellos, funcionalidades que sí se buscan en esta aplicación.

Con esto dicho, GUIDE se trata de un entorno de desarrollo "*drag-and-drop*" (arrastra y suelta), análoga a lo visto con la IDE Android Studio y su parte de desarrollo gráfico. El software nos permite programar gráficamente a través de una ventana, arrastrando y ubicando los elementos de la interfaz y asociarlos a elementos del código, para después ser el propio software el que escribe el código de estos elementos gráficos. Posteriormente a este desarrollo gráfico, ya es el programador el que implementa los códigos de ejecución que "lanzaran" los objetos gráficos como botones, entradas de texto, etc., así como las salidas código-interfaz como las gráficas.

En la figura 2.2.1.5-podemos ver como se ha usado GUIDE y sus funcionalidades para desarrollar e implementar las diferentes funcionalidades de la aplicación. Por otra parte, en la figura 2.2.1.6-podemos ver la transcripción a código que lleva a cabo Matlab.

```
% --- Executes just before gui_app is made visible.
function gui_app_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_app (see VARARGIN)
```

Figura 2.2.1.6-Fragmento de la transcripción a código llevada a cabo por el software GUIDE de Matlab

MATLAB MOBILE

Para terminar de comentar los aspectos relativos al software empleado, conviene destacar también la aplicación Matlab Mobile, puesto que será la aplicación encargada de permitirnos tanto la comunicación entre la versión de escritorio de Matlab y el dispositivo smartphone.

Esta aplicación móvil, disponible tanto para Android como para iOS permite la conexión vía internet con una sesión de trabajo de escritorio, de tal manera que sea esta, a través de dicha conexión, la que ejecuta las tareas de calculo que se soliciten a la aplicación y envía de vuelta la información pertinente al dispositivo.

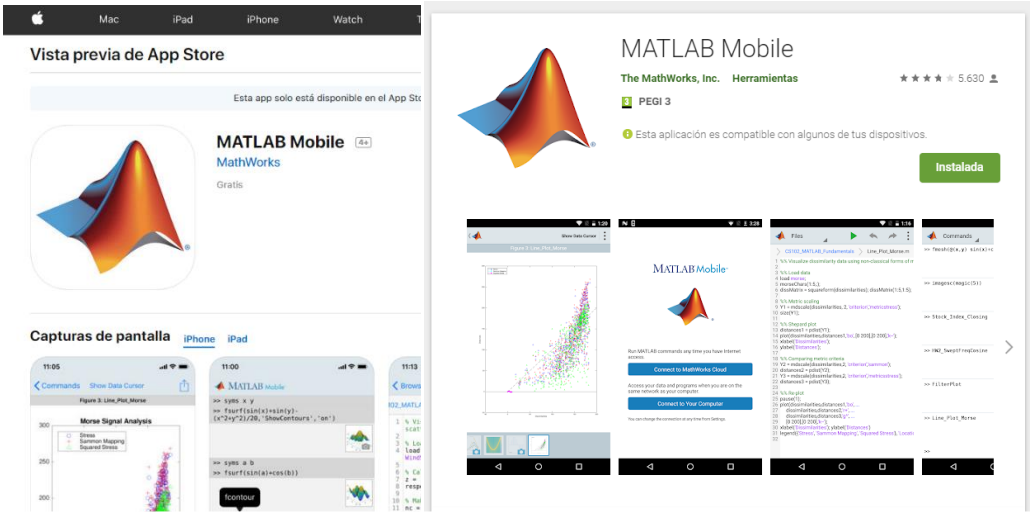


Figura 2.2.1.7-Aplicación MATLAB Mobile disponible en iTunes (izquerda) y en Google Play (derecha)

Además de la función de trabajar con una sesión de Matlab desde el dispositivo móvil, permite la opción de enviar datos de los sensores del dispositivo móvil a la aplicación de escritorio, resultando esta una característica ideal para el desarrollo de este trabajo, ahorrando el trabajo que supondría establecer de manera programática dicha conexión.

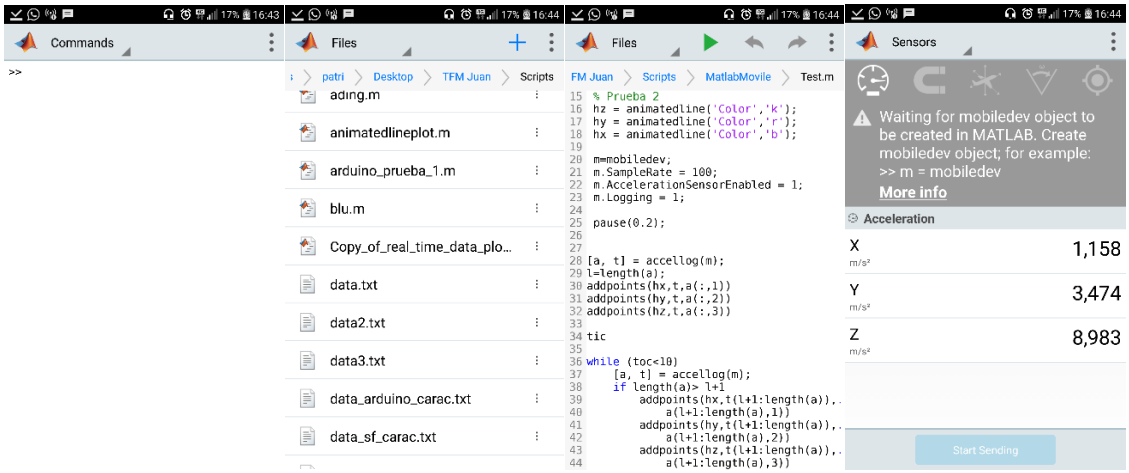


Figura 2.2.1.8-Capturas de la aplicación MATLAB Mobile, de izquierda a derecha: pantalla de entrada de comandos, explorador de archivos, apertura y escritura de scripts y acceso a sensores.

2.2.2-APLICACIÓN

Habiendo ya introducido diferentes aspectos básicos sobre Matlab, toca entrar a tratar la aplicación desarrollada mediante este software. A lo largo del presente apartado se van a exponer el desarrollo, funcionalidades, uso y configuración de la aplicación de Matlab.

PLANTEAMIENTO

Al igual que la aplicación desarrollada para smartphone, esta tiene como objetivo registrar y procesar la aceleración a partir de dispositivos y sensores disponibles y/o de bajo coste, que en este caso serán también un smartphone y un Arduino.

En este caso se ha elegido el software Matlab por, como ya se ha introducido, su elevada potencia y la cantidad de funciones y complementos disponibles que podrían ser de gran utilidad para el cumplimiento de los objetivos propuestos en este trabajo. Por lo tanto, si el objetivo en el caso de la aplicación para smartphones era el de desarrollar una aplicación desde cero para aquél dispositivo, en este caso el objetivo es el de utiliza un software ya disponible y preparado como es Matlab y la gran cantidad de opciones que este posee para desarrollar una aplicación que satisfaga nuestras necesidades.

Con esto en mente, el concepto inicial ahora es el que se muestra en la figura 2.2.1.9. Lo que se busca es transmitir, ya sea desde el smartphone y su sensor o desde el Arduino, la información de la aceleración hasta la aplicación de escritorio, la cual se encargará de procesar y gestionar estos registros.

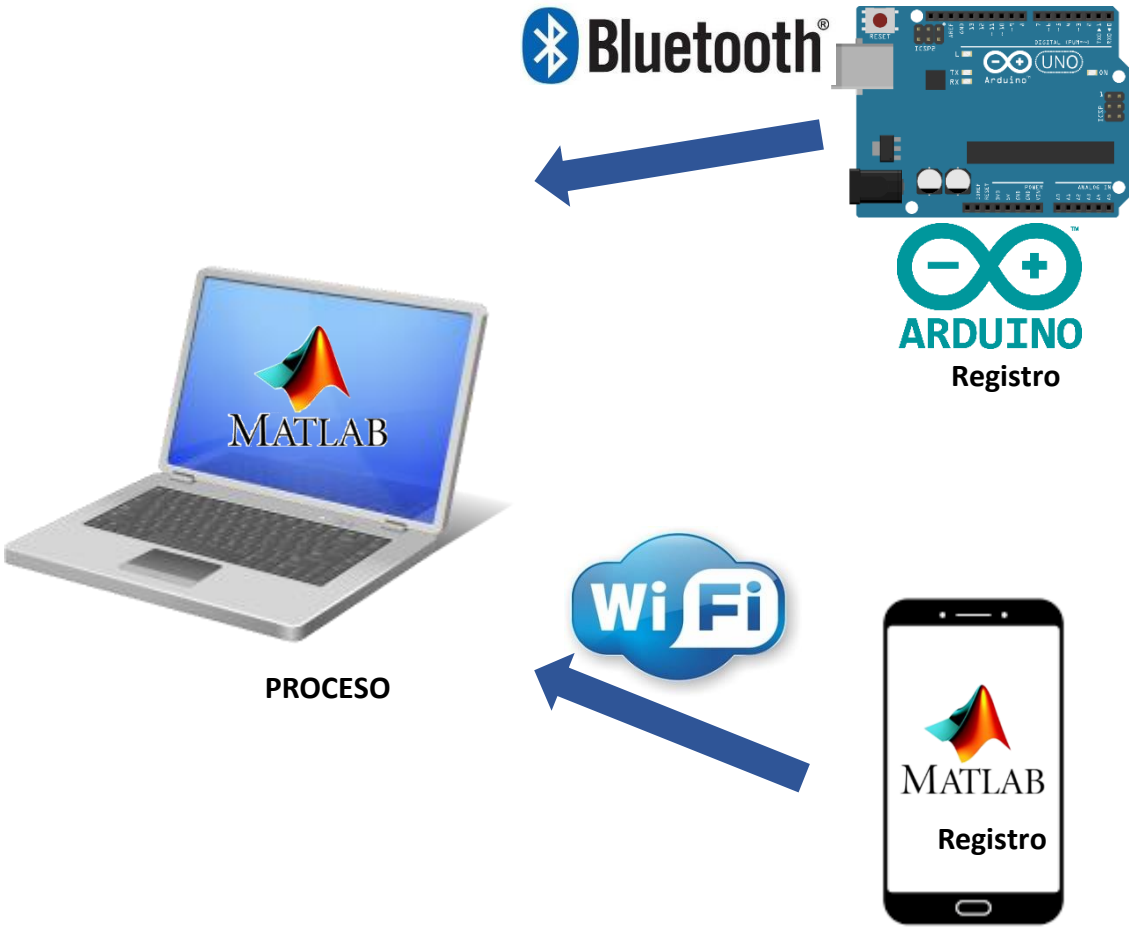


Figura 2.2.1.9-Esquema del funcionamiento de la aplicación de escritorio desarrollada en Matlab

Esto se llevará a cabo gracias a varias aplicaciones (o "Add-ons") de Matlab que permiten la conexión entre dispositivos, como son la aplicación ya expuesta Matlab Mobile, que habilita la conexión con la aplicación de escritorio mediante wifi, y la Instrument Control Toolbox™ Bluetooth®, que habilita una conexión SSP mediante bluetooth a través de la cual podremos conectar el Arduino.

Además, gracias a la citada aplicación Matlab Mobile para smartphone, podemos no solo enviar datos de los sensores del terminal a la aplicación de escritorio, sino que nos da la funcionalidad de ejecutar "scripts", secciones de código propias de Matlab, desde el propio dispositivo, recibiendo la información que ha sido procesada de vuelta. Esto nos da la posibilidad de, mediante scripts programados y almacenados en el ordenador, trabajar como en el caso de la aplicación móvil, utilizando el smartphone como sensor y como "lanzador" de las operaciones, pero con la ventaja de que es el ordenador y Matlab los que ejecutan los comandos, aprovechando así la mayor potencia del ordenador y todas las funcionalidades ya implementadas en Matlab. Este esquema alternativo de funcionamiento es el mostrado en la figura 2.2.1.9.

Figura 2.2.1.9-Esquema del funcionamiento de la aplicación Matlab Mobile

De esta manera, gracias a este doble planteamiento que permite el conjunto Matlab de escritorio con la aplicación Matlab Mobile, conseguimos resolver el problema propuesto utilizando como elemento de control tanto el dispositivo móvil como el ordenador, aunque sea en todo caso el ordenador el que ejecute los cálculos.

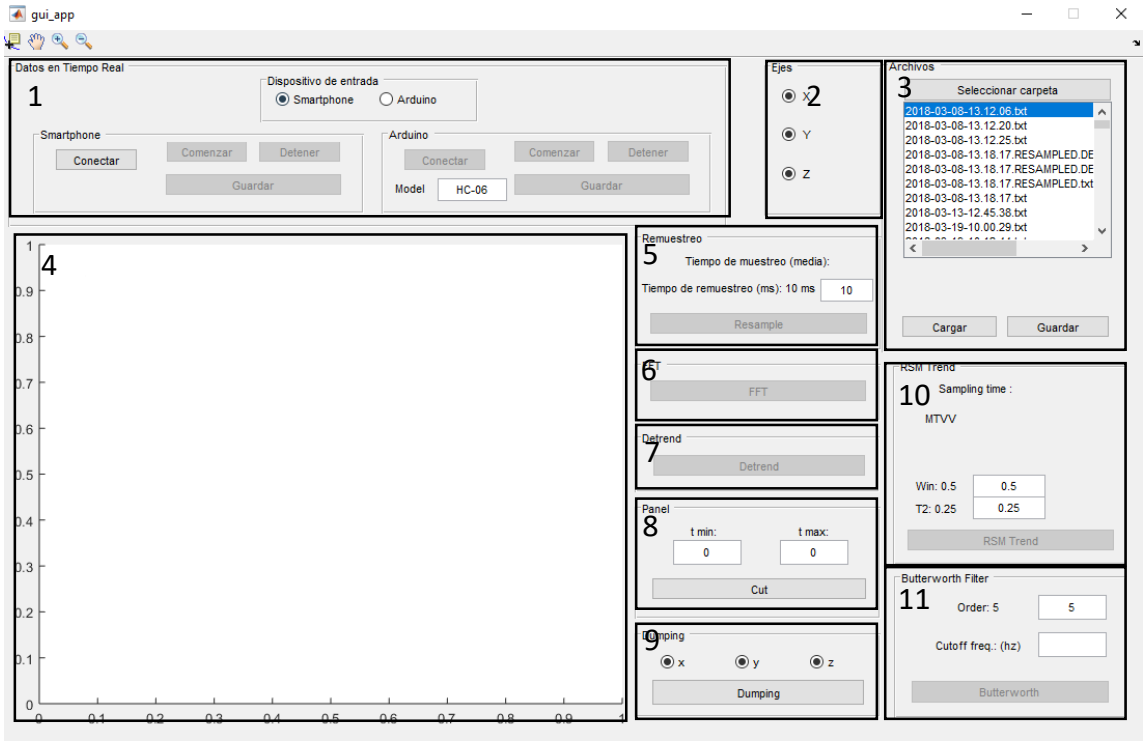
FUNCIONALIDADES Y USO:

Al igual que en el caso de la aplicación para smartphones previamente descrita, la funcionalidad fundamental vuele a ser la de registrar la aceleración, pero de nuevo, esta no ha sido la única funcionalidad implementada.

Básicamente las funcionalidades implementadas han sido las mismas que la otra aplicación, añadiendo en esta aplicación dos nuevas. Estas funcionalidades son:

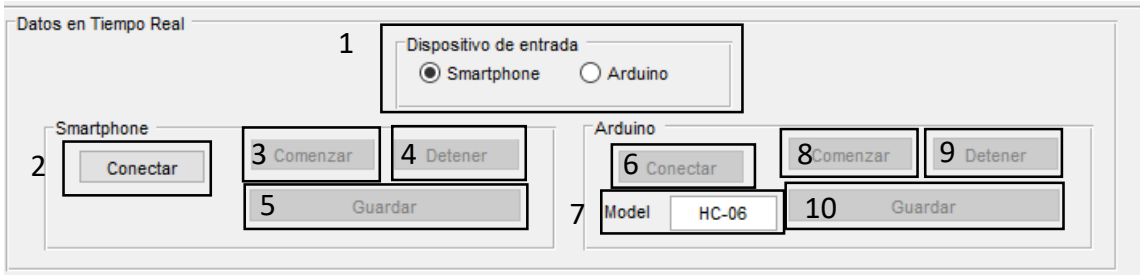
- Registro en tiempo real
- Selector de frecuencia de registro
- Selectores de ejes de representación
- Resampleado o remuestreo del registro
- Carga y visualización
- Detrend o eliminación de la tendencia lineal
- Selector de gravedad
- Cálculo y representación de FFT (Transformada Rápida de Fourier)
- Filtro Butterworth sobre un registro guardado
- Cálculo y representación de la RMS Trend y sus máximos (MTVV)
- Recorte del registro
- Ajuste exponencial del registro

Todas estas funcionalidades se muestran ubicadas en la aplicación en las siguientes figuras:



1. Selector de sensor de entrada
2. Selector de ejes para representación gráfica
3. Gestor de archivos almacenados
4. Gráficos
5. Remuestreo o resample
6. FFT
7. Detrend o eliminar tendencia lineal
8. Recorte de registro
9. Ajuste logarítmico
10. RMS Trend
11. Filtro butterworth

Figura 2.2.1.10-Leyenda general de la ubicación de las distintas utilidades de la aplicación de escritorio de en Matlab



1. Selección del sensor
2. Conectar con smartphone
3. Comenzar registro “live” a partir del smartphone
4. Detener registro “live” del smartphone
5. Guardar resgistro “live” del smartphone
6. Conectar con Arduino
7. Identificador del dispositivo Bluetooth del Arduino
8. Comenzar registro “live” a partir del Arduino
9. Detener registro “live” del Arduino
10. Guardar resgistro “live” del Arduino

Figura 2.2.1.11-Leyenda general de la ubicación de las distintas utilidades de la aplicación de escritorio de Matlab

Con las funcionalidades introducidas, se pasa a comentar el uso de las mismas:

La entrada de registros se ejecuta, de nuevo, en tiempo real, habilitándose la opción de almacenar dicho registro una vez este se detiene.

Se pueden encontrar los botones que gestionan esta conexión en la parte superior de la interfaz, y lo primero que se destaca de ellos son las opciones que ofrecen, mostradas en la figura 2.2.1.10.

A diferencia que en la aplicación anterior donde la única diferencia entre seleccionar uno u otro sensor es la modificación del sensor de registro, en el caso de esta aplicación el funcionamiento del registro cambia ligeramente:

- **Registro mediante Smartphone:**

  En este caso, como paso previo se deben conectar tanto el ordenador desde el que se ejecuta la aplicación de escritorio como el dispositivo móvil a la misma red. Hecho esto, se debe introducir el comando `connecto on` seguido por la contraseña que se elija en el espacio de Matlab. Esto nos dará como resultado lo mostrado en la figura 2.2.1.12, donde vemos que Matlab nos devuelve los datos de la conexión establecida, como la IP, el nombre del dispositivo o, en caso de no conectarse al puerto por defecto (31415), el número del puerto al que se conecta.

Figura 2.2.1.12-A la izquierda, datos de la conexión que el comando connector on devuelve. A la derecha, conexión establecida en la aplicación móvil.

Tras esto, debe procederse a conectar también la aplicación del dispositivo móvil. Para ello, lo hacemos introduciendo los datos pertinentes a la aplicación, con especial atención a la dirección IP, el puerto en caso de no utilizar el de por defecto, y la contraseña, que tendrá que ser la misma introducida en la aplicación de escritorio. Con esto ya puede utilizarse como parte de la aplicación.

- **Registro mediante Arduino:**

  En el caso del Arduino se deberá proceder de manera análoga, emparejando el ordenador con el dispositivo mediante Bluetooth. El proceso, en el caso de un

ordenador con Windows 10, es e l que se muestre en la figura 2.2.1.13. La contraseña por defecto para el HC-06 es 1234.

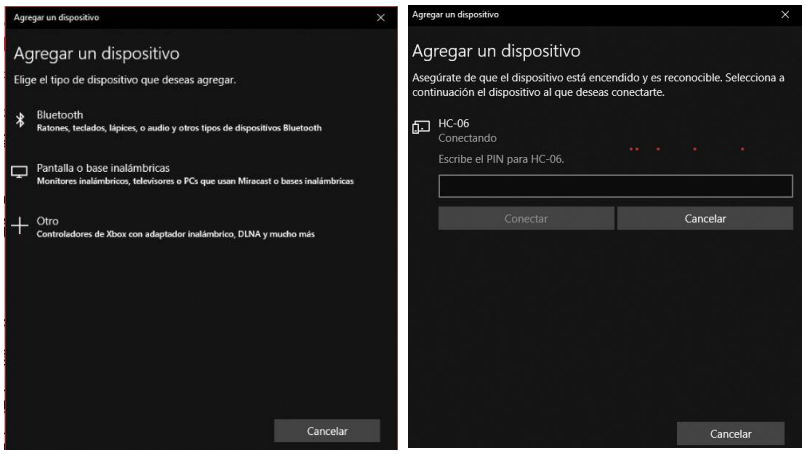


Figura 2.2.1.13-Búsqueda y emparejamiento con dispositivo HC-06 a través de PC con Windows 10

Una vez emparejados, ya se puede empezar a trabajar con el sensor a través del Arduino. Nótese que la conexión en este caso es gestionada desde Matlab, y lleva un tiempo en establecerse la conexión y en abrir el puerto establecido, así que en caso de usar esta opción se recomienda tener en cuenta que se tardará en torno a 20 segundos en establecer la conexión y otro tanto en comenzar la entrada de registro.

Una vez se ha establecido la conexión con el sensor que se desee, ya podemos pasar a registrar en tiempo real con el dispositivo conectado. La siguiente figura 2.2.1.14 muestra la entrada de registros en los dos casos disponibles.

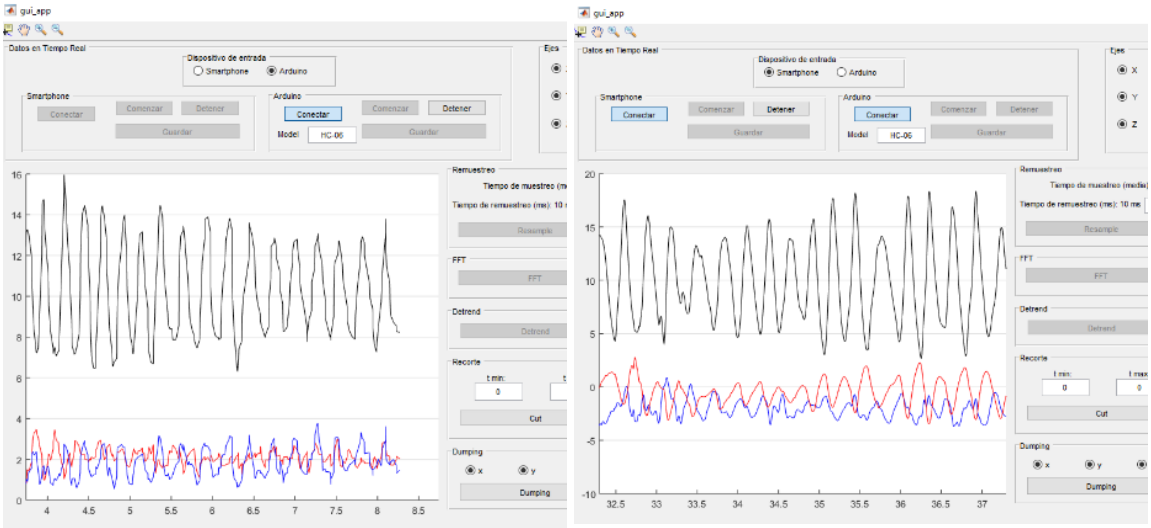


Figura 2.2.1.14-Entrada de registro Live en la aplicación de Matlaba: izquierda mediante Arduino, derecha mediante Smartphone.

Una vez detenemos el registro, se nos habilita la opción de **Guardar**, mediante la cual guardamos el registro completo de la entrada "live" efectuada. Este archivo se carga automáticamente en la lista de archivos guardados dándonos acceso a cargarla y procesarla.

Al igual que en la aplicación para smartphones, el registro se almacena en un archivo de texto con el mismo formato que la otra aplicación, de tal manera que sean intercambiables.

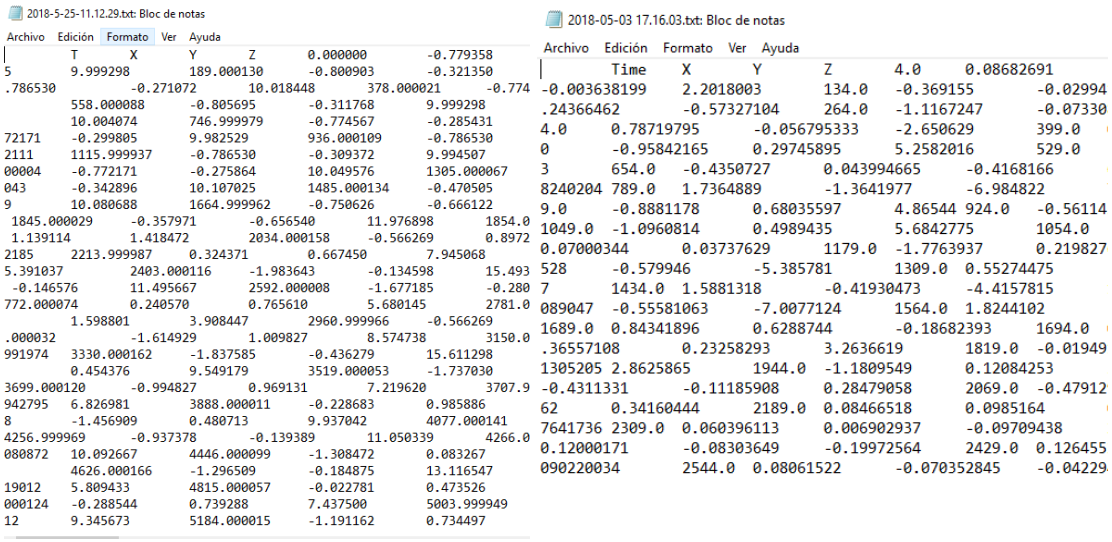


Figura 2.2.1.15-Archivos de texto con registros almacenados. A la izquierda, registrado con Matlab. A la derecha, registrado con un Smartphone. Vemos que tienen el mismo formato.

Con esto queda indicada la funcionalidad de registro.

CARGADO DE REGISTRO

Una vez se ha realizado y guardado el registro, este aparecerá disponible para ser cargado en la lista de archivos, tal y como se ha comentado anteriormente.

Para realizar el cargado del archivo se tiene disponible a la derecha una serie de opciones, mostradas en la figura 2.2.1.16. Lo primero será seleccionar la carpeta donde se tengan los archivos, que será la misma donde se van a guardar. Una vez se tenga la carpeta con los archivos, se selecciona el archivo que se quiera cargar en la lista disponible a la derecha de la aplicación, y acto seguido se pulsa el botón de cargar.

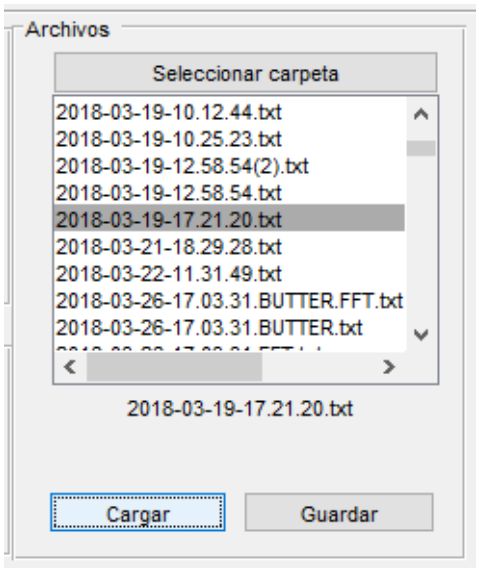


Figura 2.2.1.16-Lista de archivos disponibles y las diferentes opciones para cargar, seleccionar la carpeta de destino y guardar los archivos con las operaciones.

Una vez se carga el archivo, se habilitarán las operaciones que correspondan, lo que dependerá, al igual que en la aplicación para smartphone, de si el registro tiene una frecuencia de muestro fija o no.

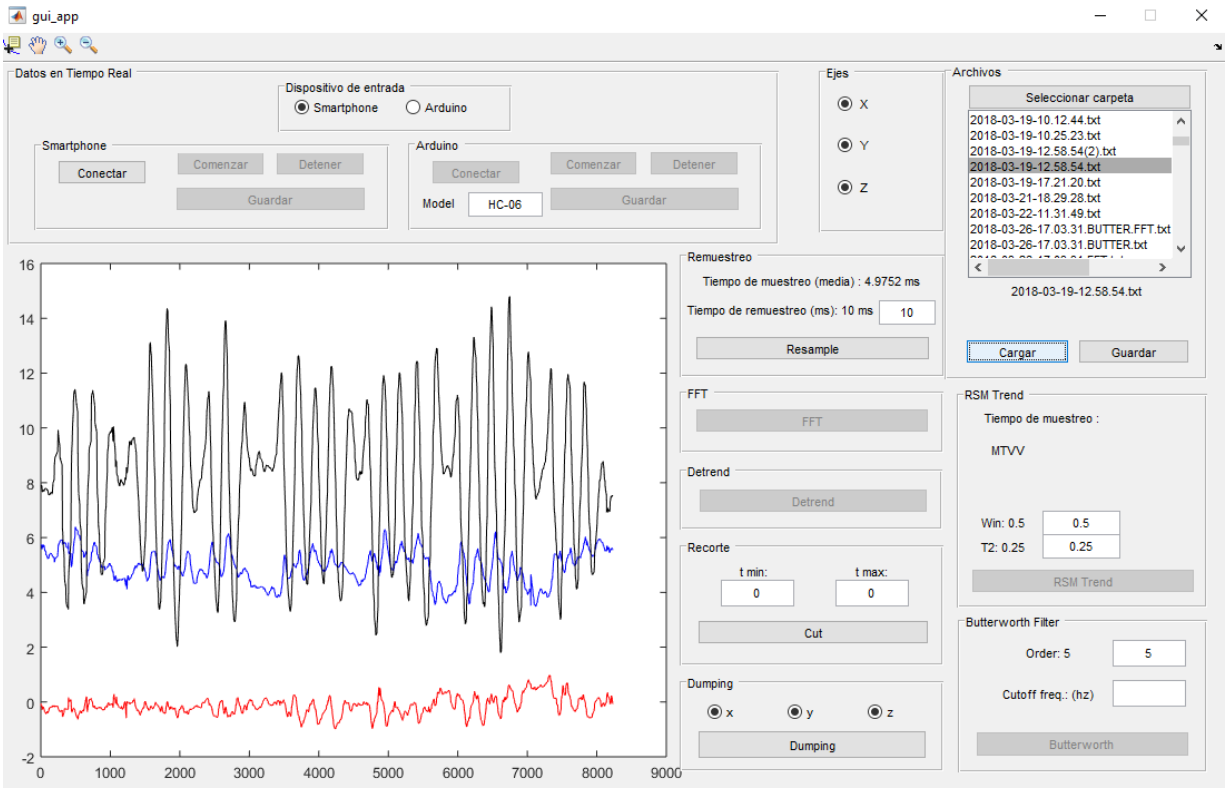


Figura 2.2.1.17-Registro cargado y opciones correspondientes de procesado habilitadas. En este caso solo el resampleo al no tratarse de un registro equiespaciado.

REMUESTREO

En este caso, no existe la opción de registrar a una frecuencia determinada, con lo que el tiempo de muestreo vendrá dado por el dispositivo que se use. Por esto, y de cara a equiespaciar temporalmente la muestra para que sea procesable, se ha habilitado una opción de remustreo o resampleo, de tal manera que la muestra pueda ser procesada.

Su funcionamiento es básico: se introduce el tiempo al que se quiere remuestrear y se ejecuta.

Cabe destacar que además se muestra el tiempo medio de muestreo del registro, con lo que se puede introducir un tiempo de remuestreo próximo a este de cara a alterar lo menos posible la muestra original.

Esta operación se habilitará al cargar el archivo en el caso de que no se detecte un tiempo de muestreo fijo.

Para guardar este nuevo registro, pulsando el botón guardar del cuadro de Archivos, en la esquina superior derecha. El nuevo archivo se guarda con el nombre del anterior más la extensión ".RESAMPLED".

FFT

Ejecuta la transformada rápida de Fourier y muestra el resultado en pantalla. Para guardar este nuevo registro, pulsando el botón guardar del cuadro de Archivos, en la esquina superior derecha. El nuevo archivo se guarda con el nombre del anterior más la extensión ".FFT".

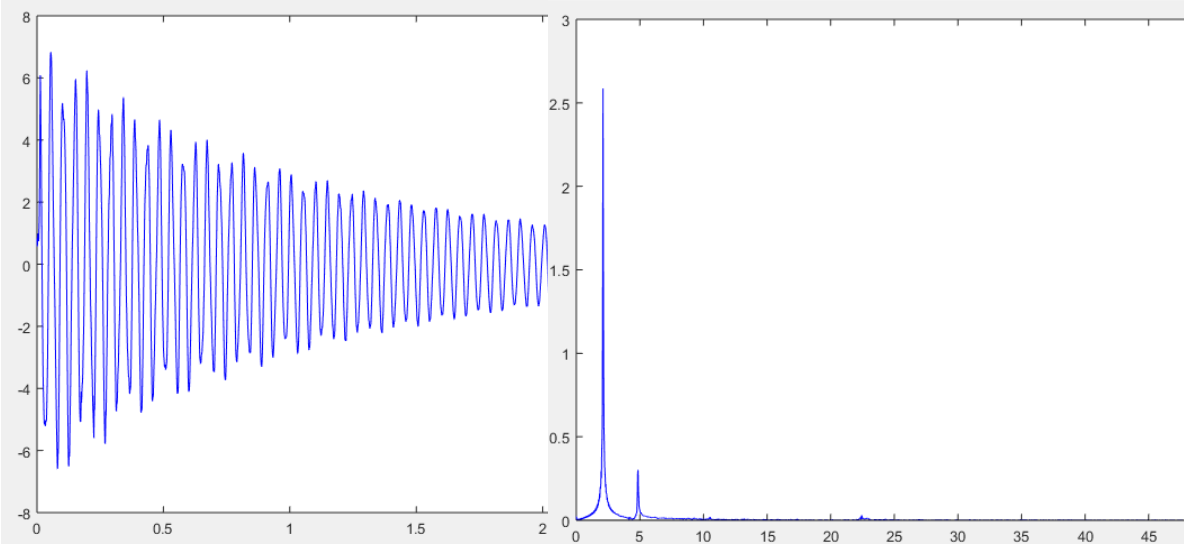


Figura 2.2.1.18-Recorte del resultado de ejecutar la FFT: a la izquierda el registro original, a la derecha la FFT.

DETREND

Se elimina la tendencia lineal y se muestra en pantalla. Para guardar este nuevo registro, pulsando el botón guardar del cuadro de Archivos, en la esquina superior derecha. El nuevo archivo se guarda con el nombre del anterior más la extensión ".DETREND".

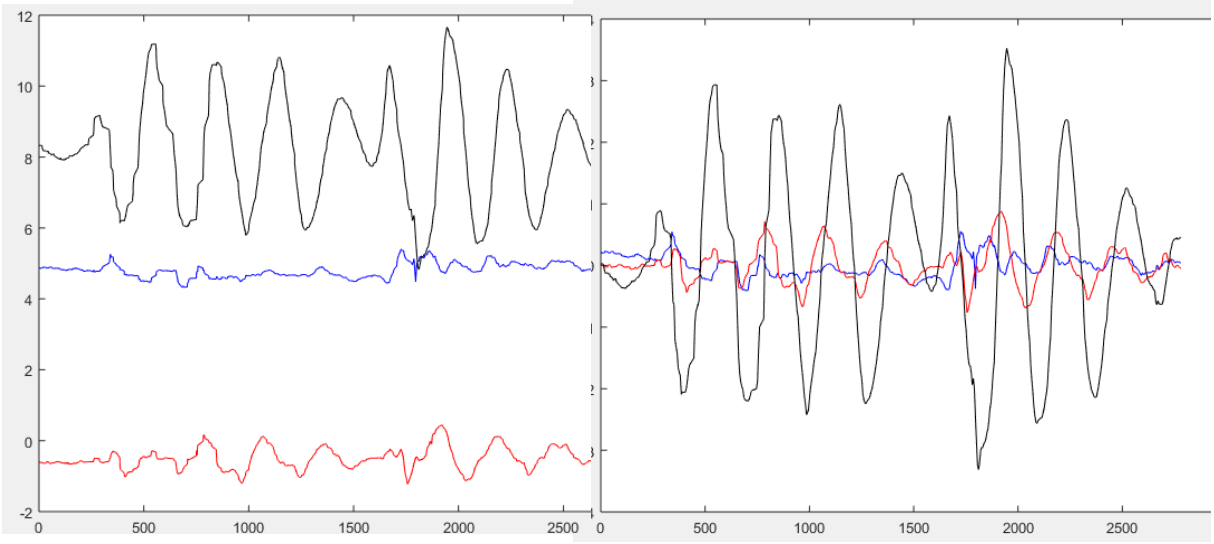


Figura 2.2.1.19-Recorte del resultado de ejecutar la operación Detrend: a la izquierda el registro original, a la derecha con la tendencia lineal eliminada.

RECORTE

Esta operación nos permite recortar la señal. Para ello, introducimos el tiempo de inicio, el tiempo de fin o ambos. Cabe destacar que este tiempo está en milisegundos, y que una opción para seleccionar este tiempo puede ser mediante el "Data Cursor" habilitado en la barra de herramientas superior de la aplicación. Un ejemplo de esto lo vemos en la figura 2.2.1.20.

Para guardar este nuevo registro, pulsando el botón guardar del cuadro de Archivos, en la esquina superior derecha. El nuevo archivo se guarda con el nombre del anterior más la extensión ".CUT".

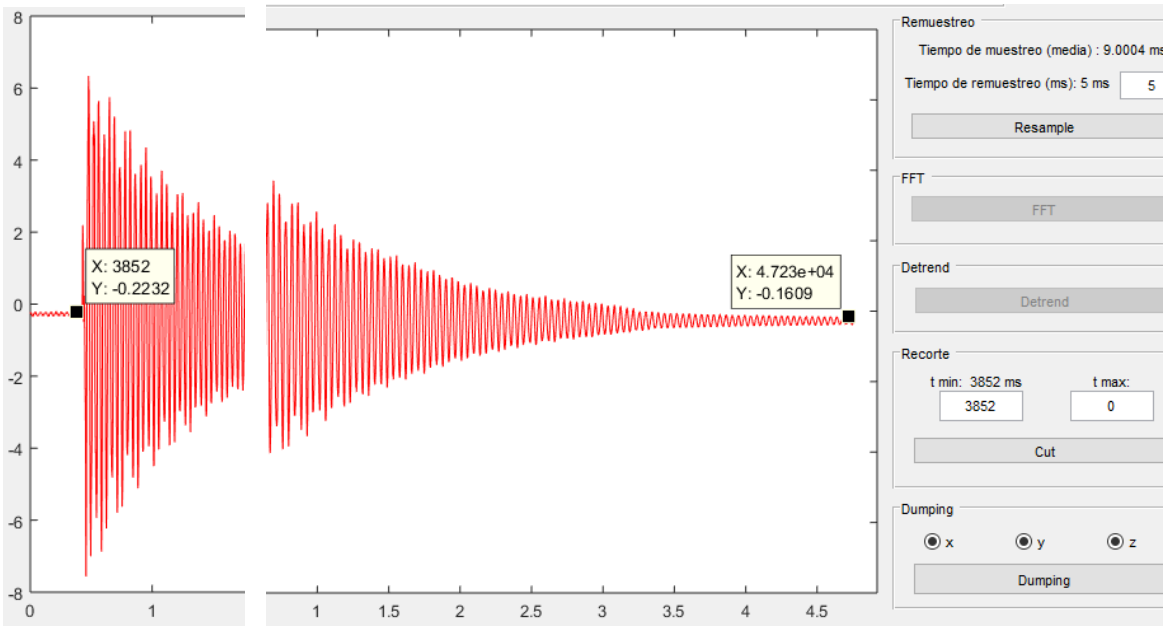


Figura 2.2.1.20-Recorte del resultado de ejecutar la operación Cortar: a la izquierda el registro original, señalando el inicio del corte mediante el Data Cursora, a la derecha con tras el corte realizado.

RMS Trend

Realiza el cálculo de la "root mean square", la media cuadrática, y su tendencia. Se representa en pantalla sobre la señal original y los valores máximos en cada eje, correspondientes a los MTVV, que además también aparecen recogidos en el cuadro de diálogo de la operación.

Al igual que en la aplicación de Smartphones, podemos introducir los parámetros referentes a la ventana de cálculo por la que se desplaza esta media cuadrática como son el tamaño de la misma (win) y su desplazamiento (T2).

Para guardar este nuevo registro, pulsando el botón guardar del cuadro de Archivos, en la esquina superior derecha. El nuevo archivo se guarda con el nombre del anterior más la extensión ".RMS_Trend".

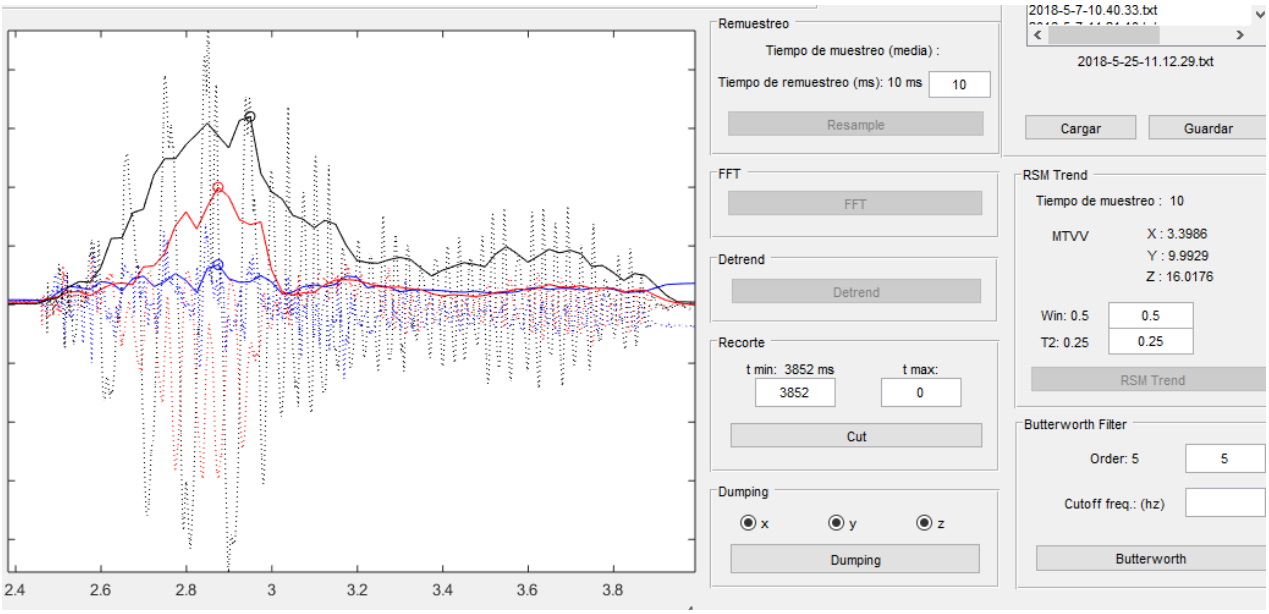


Figura 2.2.1.21-Recorte del resultado de ejecutar la operación RMS Trend: vemos la gráfica con las tendencias representadas sobre la señal original, que se muestra mediante línea discontinua. También se resaltan los máximos mediante círculos, además de mostrarse a la derecha en el panel de la operación.

BUTTERWORTH FILTER

Al igual que en el caso de la aplicación móvil, se ha optado por un filtro de tipo Butterworth. Mediante el panel de la operación podemos introducir el orden del filtro y su frecuencia de corte. Tras ejecutar la operación, esta mostrará el resultado por pantalla.

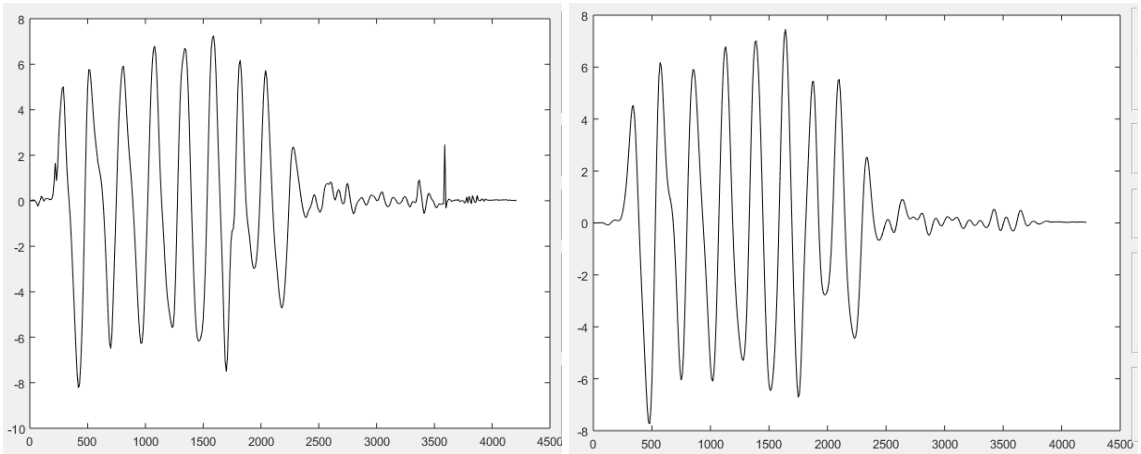


Figura 2.2.1.22-Recorte del resultado de ejecutar la operación Butterworth Filter: vemos la gráfica con el registro original a la izquierda, y a la derecha tras aplicar un filtro de orden 5 y frecuencia de corte de 10 hz.

DUMPING

La última de las operaciones disponibles es la envolvente exponencial de la señal, de utilidad en el caso de vibraciones que se vayan atenuando.

En el panel de la operación podemos seleccionar el registro de qué ejes queremos procesar.

Una vez ejecutamos la operación, irán apareciendo por pantalla una serie de cuadros de diálogo y gráficos, en el siguiente orden:

- Picos: primero se intentarán encontrar los picos del registro. Mediante una ventana de alerta, se preguntará si los picos identificados son correctos. En caso de que lo sean, se continua con la operación, si se considera que no, vuelven a buscarse las crestas. En este caso se encontrarán menos, las más exteriores. Y así sucesivamente

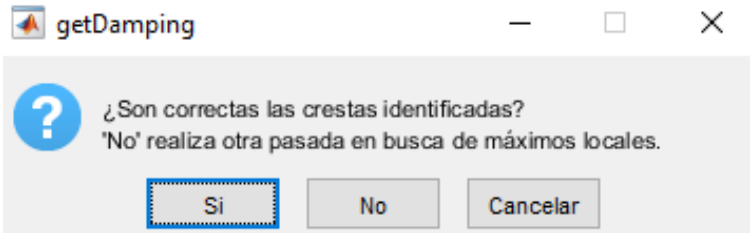


Figura 2.2.1.23-Cuadro de alerta para confirmar las crestas encontradas.

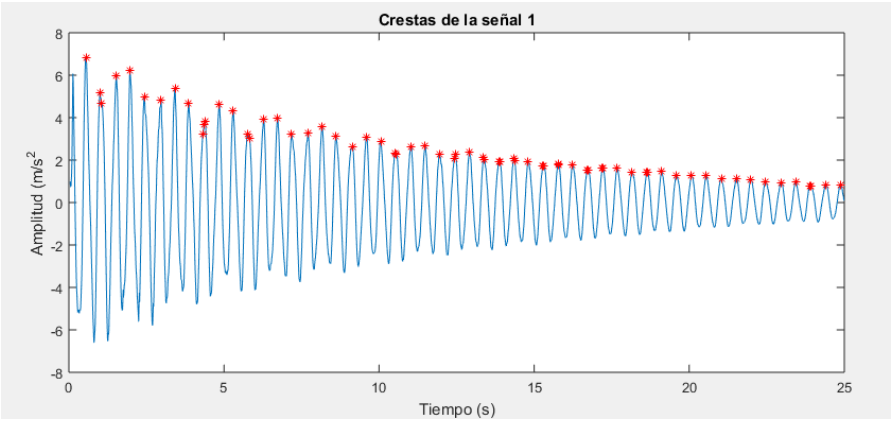


Figura 2.2.1.24-Crestas identificadas.

- Tramos de división: la función detecta ella sola el tramo decreciente, pero nos da la opción de determinar manualmente el tramo a procesar.

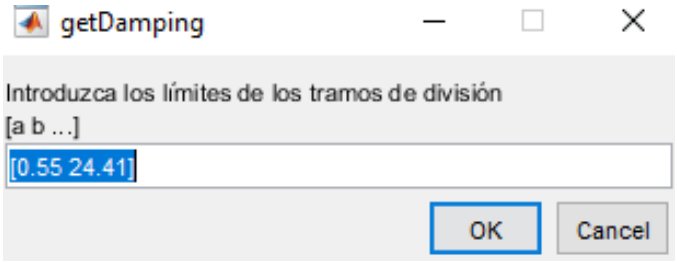


Figura 2.2.1.25-Cuadro de diálogo para introducir manualmente la ventana de cálculo.

- Regresión lineal: tras seleccionar la ventana de cálculo, se realiza una regresión lineal con las crestas comprendidas en esta ventana. De nuevo nos da la opción de aceptar la regresión o de no hacerlo, en cuyo caso eliminará el valor más alejado de la recta, hasta que la regresión satisfaga al usuario.

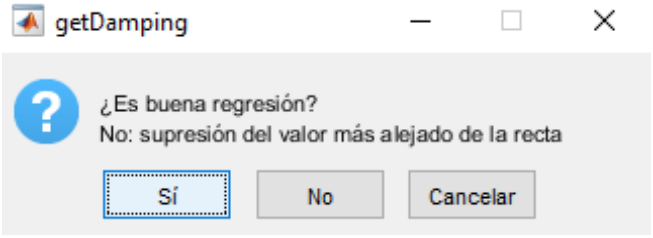


Figura 2.2.1.26-Cuadro de diálogo para confirmar o no el ajuste de la regresión lineal ejecutada.

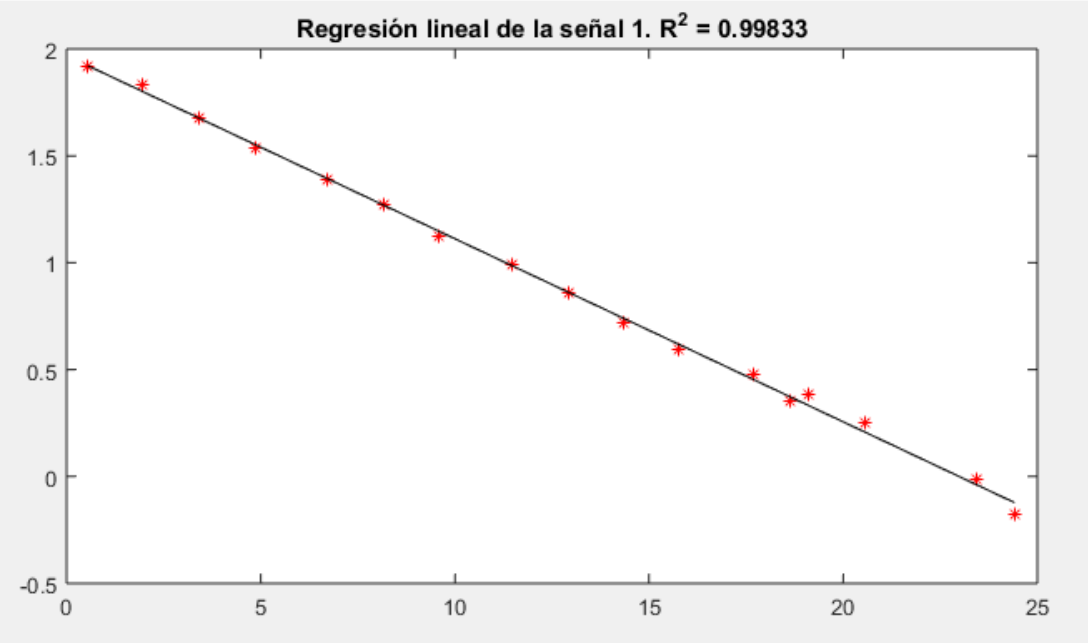


Figura 2.2.1.27-Regresión lineal a partir de las crestas de la señal.

- Frecuencia de oscilación de la señal: por último, nos permitirá introducir la frecuencia de oscilación de la señal. Esta es calculada por el programa, que nos la mostrará en el propio cuadro de diálogo, pero aun así existe la opción de introducirla manualmente.

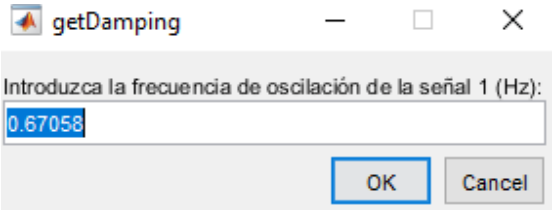


Figura 2.2.1.28-Cuadro de diálogo para introducir manualmente la frecuencia de oscilación de la señal.

Tras finalizar los pasos hasta aquí descritos, aparecerá en pantalla las gráficas tal y como se muestra en la figura 2.2.1.29 mostrada a continuación:

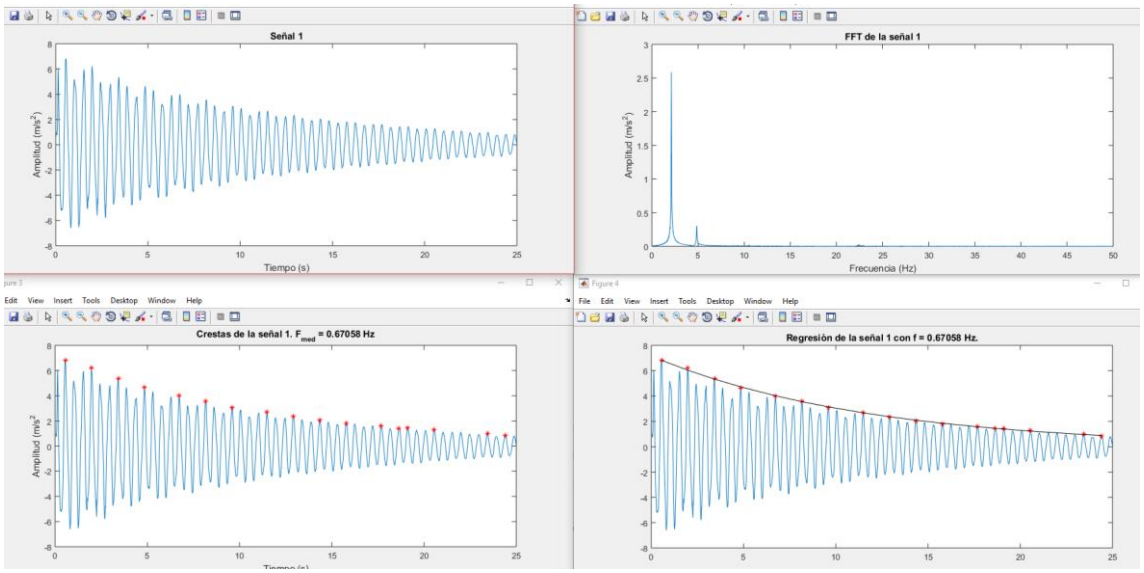


Figura 2.2.1.29-Resultado final de la operación de Dumping. De izquierda a derecha y de arriba abajo: señal original, FFT de la señal original, señal con las crestas utilizadas y envolvente exponencial

Por último, comentar el uso de la aplicación Matlab Mobile, que nos permitirá usar las capacidades de Matlab, pero desde el propio Smartphone. Para ello, lo primero establecer la conexión entre Matlab de escritorio y el teléfono, tal y como se describe en el apartado de registro en tiempo real. Cabe destacar que para esto también se puede utilizar la conexión mediante la nube que permite la propia aplicación y no sólo mediante conexión a través de la misma red wifi, pero esto es algo que no se ha utilizado en el presente proyecto y queda en manos de futuros usuarios.

Una vez establecida la conexión, ya tendremos acceso al contenido disponible en el escritorio desde el smartphone. Como se avanzó cuando se describió la aplicación móvil de Matlab, su funcionamiento es el siguiente:

- Se habilita por defecto la entrada de comandos. En la parte superior de la aplicación se encuentra un desplegable con las diferentes opciones disponibles, tal y como muestra la Figura 2.2.1.30. En este caso interesara el apartado de "Files", que da acceso a los scripts que tengamos accesibles en nuestra sesión de Matlab de escritorio.

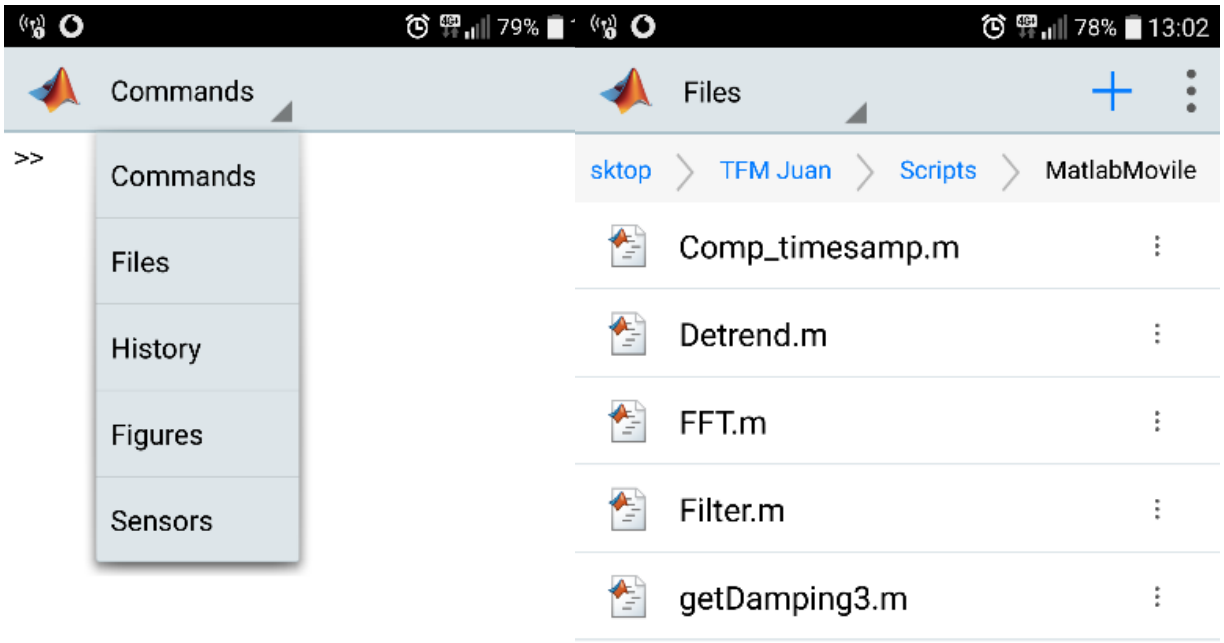


Figura 2.2.1.30-Opciones disponibles en la aplicación MATLAB MOBILE. A la derecha, el acceso a los scripts guardados.

- Una vez se accede a los scripts, estos pueden ser ejecutados. Para ver como se lleva esto a cabo, se va a mostrar un ejemplo mediante un script preparado para la toma de datos:

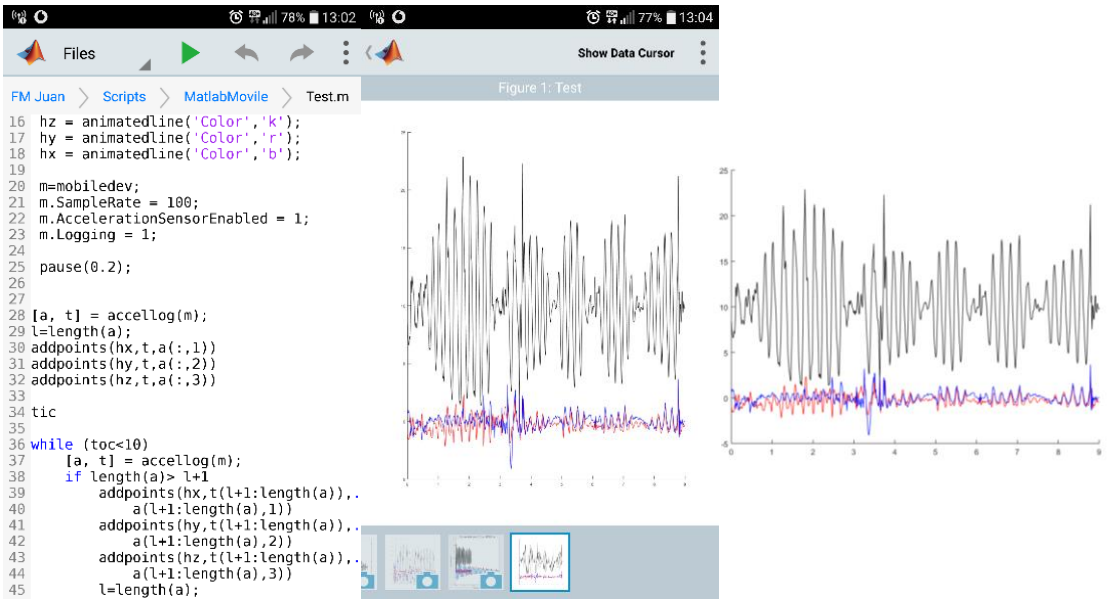


Figura 2.2.1.31-Capturas de pantalla de la ejecución del script de registro desde la aplicación móvil. A la izquierda, el script con el código, en el centro, la gráfica vista desde el móvil, a la derecha la gráfica mostrada en el escritorio.

Vemos que ejecuta el script igual que lo hace desde el escritorio, y que además de mostrar el resultado por el móvil lo hace, en tiempo real (por estar así programado) en el escritorio.

Tras esto, quedara disponible el registro para más operaciones o para su guardado. Algunos de operaciones:

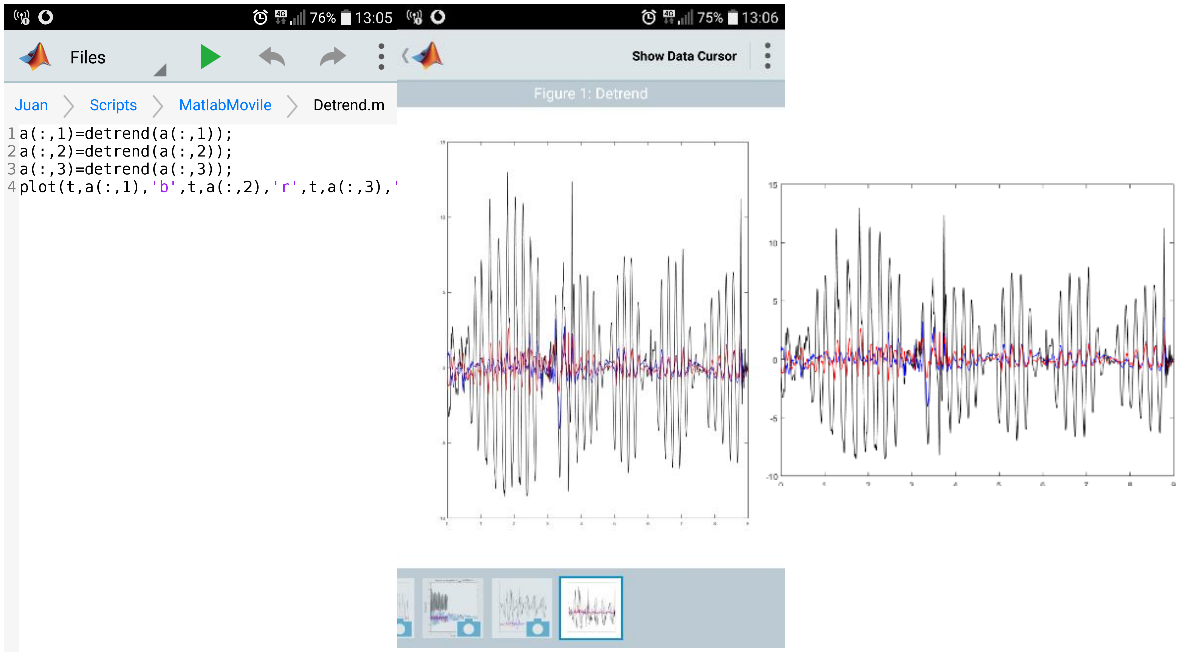


Figura 2.2.1.32-Capturas de pantalla de la ejecución del script de DETREND sobre el registro de la figura 2.2.1.31 desde la aplicación móvil. A la izquierda, el script con el código, en el centro, la gráfica vista desde el móvil, a la derecha la gráfica mostrada en el escritorio.

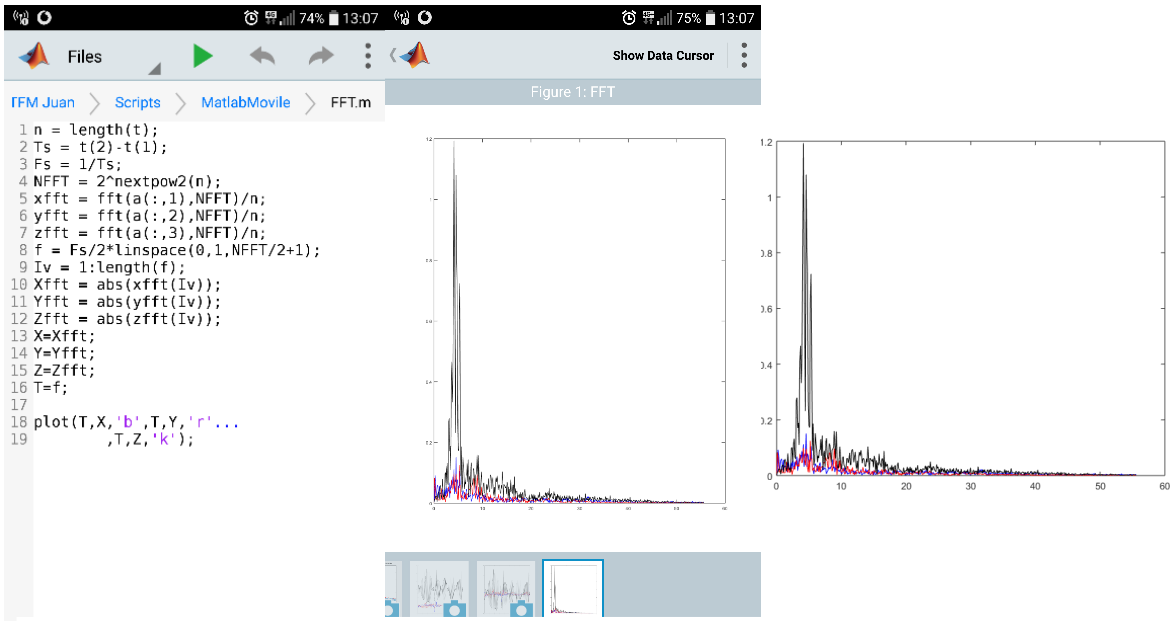


Figura 2.2.1.33-Capturas de pantalla de la ejecución del script de FFT sobre el registro de la figura 2.2.1.32 desde la aplicación móvil. A la izquierda, el script con el código, en el centro, la gráfica vista desde el móvil, a la derecha la gráfica mostrada en el escritorio.

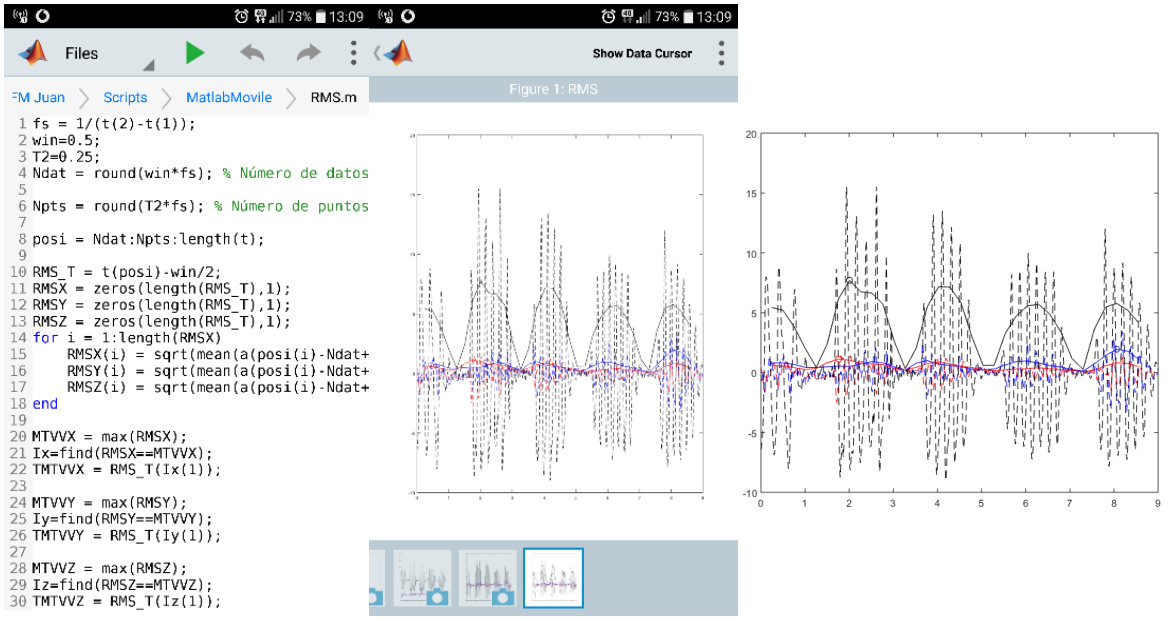


Figura 2.2.1.34-Capturas de pantalla de la ejecución del script de RMS Trend sobre el registro de la figura 2.2.1.32 desde la aplicación móvil. A la izquierda, el script con el código, en el centro, la gráfica vista desde el móvil, a la derecha la gráfica mostrada en el escritorio.

Y con esto quedarían descritas todas las funcionalidades implementadas mediante las aplicaciones de Matlab.

DESARROLLO DE LA APLICACIÓN

Al igual que en el capítulo sobre la aplicación Smartphone, en esta sección se van a comentar diferentes aspectos del desarrollo de la aplicación. En este caso se tratarán principalmente el acceso a las diferentes funciones incluidas y como habilitarlas en la sesión de Matlab a utilizar. Será pues una sección considerablemente más breve que la misma para Smartphones, debido a que en este caso se ha utilizado software y aplicaciones ya existentes, cuya implementación ha sido algo más sencillo y se encuentra suficientemente detallada y documentada en el anexo correspondiente al código de Matlab, con lo que, en caso de requerir continuar el presente proyecto, el propio anexo sirve como guía suficiente en la mayoría de los casos.

Aun así, se van a comentar algunos aspectos relativos al desarrollo necesarios para entender ciertas partes de la misma y ser capaz de reproducirla para implementaciones futuras.

CONEXIÓN CON DISPOSITIVOS SMARTPHONES

Como ya se ha comentado repetidamente a lo largo de la presente memoria, se establece una conexión wifi con el dispositivo smartphone para el registro a través de los sensores del mismo dispositivo. Esto se hace mediante la aplicación móvil distribuida por la propia Mathworks, y el establecimiento de dicha conexión está documentado en la sección funcionalidades y uso de la aplicación Matlab, en la subsección para el registro “live”.

Pero para establecer esta conexión, antes hay que **instalar** en la sesión de Matlab las **herramientas correspondientes** que habilitan estas opciones. En este caso se instalará el **paquete de soporte de hardware para Android**. Para ello, vamos a la barra de herramientas “Home” de Matlab, y en la sección de “Environment”, desplegamos las opciones que se muestran bajo la pestaña de “Add-Ons”, Tal y como se muestra en la figura 2.2.1.35.

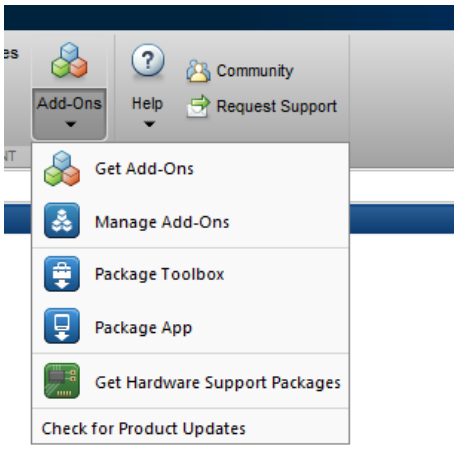


Figura 2.2.1.35-Opciones desplegables bajo icono de Add-Ons

Pulsamos la opción de “Get Hardware Support Packages”, que nos llevará a otra ventana. Esta nueva ventana pedirá al usuario que se “logee” si no lo, lo cual deberá hacerse para poder instalarse los “Add-Ons” correspondientes. Para ello es necesaria una cuenta en Mathworks. Una vez se haya conectado, ya redireccionará al usuario a la pantalla mostrada en la figura 2.2.1.36, donde se deberá seleccionar la opción de “Instal from Internet”, y pulsar “next”.

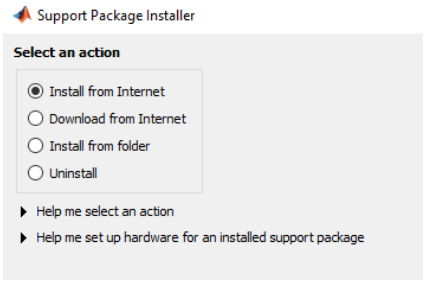


Figura 2.2.1.36-Opciones disponibles en el instalador de paquetes de soporte de Matlab

Esto nos llevará a la pantalla que se muestra en la Figura 2.2.1.37, donde se nos muestran los diferentes paquetes disponibles, de entre los cuales se seleccionará el soporte para Android. Hecho esto, quedará instalado el paquete y ya se tendrá acceso a las funcionalidades programadas en la aplicación.

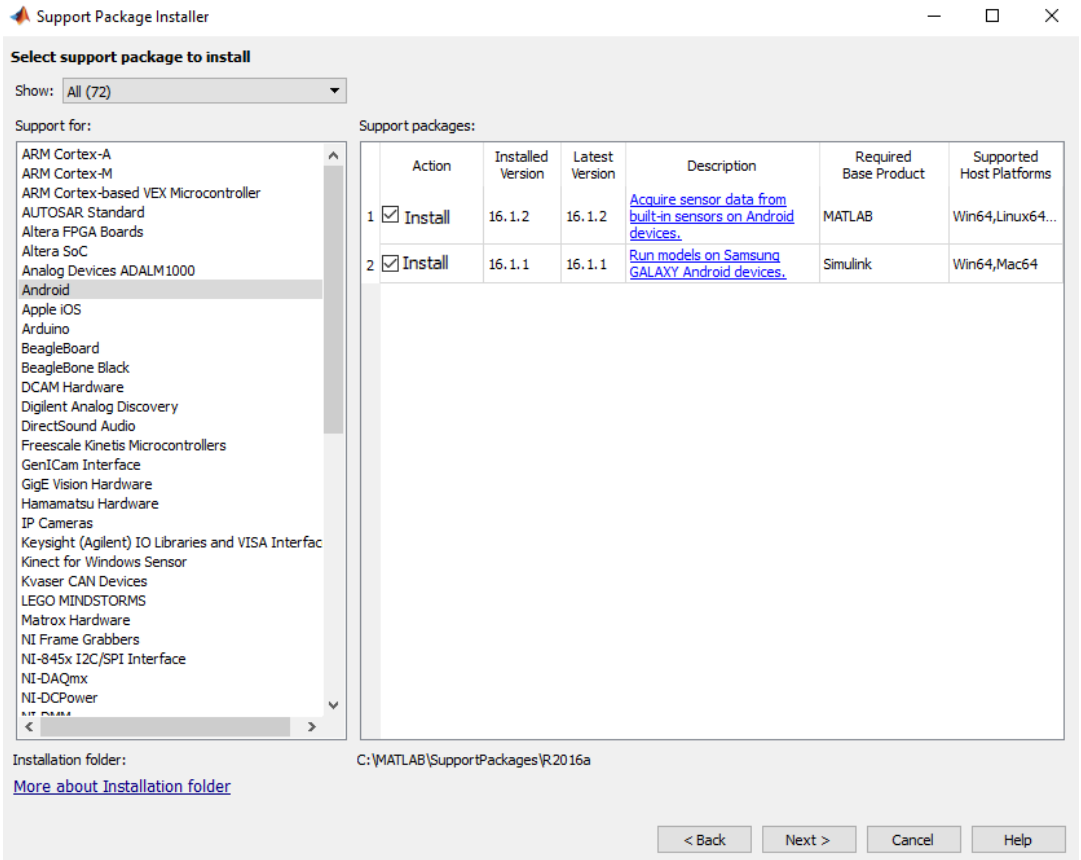


Figura 2.2.1.38-Paquete a instalar para la conexión con dispositivos Android.

Hecho esto, ya se tendrá disponible la conexión con la aplicación móvil descrita en la sección correspondiente.

En cuanto a la implementación programática del acceso a la aplicación móvil y a los sensores del dispositivo, se lleva a cabo, tras haber establecido la conexión, de la siguiente manera:

Conexión con dispositivo Smartphone y acceso a sensores

```
% Creo el objeto que representa el móvil conectado
 m=mobiledev;
% Determino la máxima frecuencia disponible
 m.SampleRate = 100;
% Activo el acelerómetro
 m.AccelerationSensorEnabled = 1;
% Establezco la conexión
 m.Logging = 1;
% Leo el registro cada vez que instancio esto
[a, t] = accellog(m);
```

La lectura se lleva a cabo con el último comando (`accellog(m)`), y lo que devuelve este comando es la lectura completa de las aceleraciones en cada eje (matriz a) y su tiempo correspondiente (vector t) desde el momento en que se establece la conexión (`m.Logging=1`). Debido a que cada vez que se invoca esta función nos devuelve el registro completo, la implementación del registro en tiempo real se ha implementado de la siguiente manera:

Conexión con dispositivo Smartphone y acceso a sensores

```
if length(a)> l+1
     addpoints(hx,t(l+1:length(a)),...
         a(l+1:length(a),1))
     addpoints(hy,t(l+1:length(a)),...
         a(l+1:length(a),2))
     addpoints(hz,t(l+1:length(a)),...
         a(l+1:length(a),3))
     l=length(a);
 end
```

De este modo, gracias al parámetro `l`, que almacena la longitud del ultimo registro, se asegura que no se almacenen nunca los mismos datos de tiempo-aceleración varias veces.

CONEXIÓN MEDIANTE BLUETOOTH CON ARDUINO

La otra conexión a establecer mediante esta aplicación es la que se realiza gracias a la conectividad bluetooth de los distintos dispositivos. Al igual que en el caso anterior, ya ha quedado documentado en la sección funcionalidades y uso de la aplicación Matlab, en la subsección para el registro “live”, así que aquí sólo se tratarán algunos aspectos programáticos de dicha conexión.

Lo primero a destacar es que esta conexión se establece mediante el siguiente comando:

```
Bluetooth('Name',1)
```

De esta forma construimos un objeto que contiene la conexión bluetooth con el dispositivo identificado por `'Name'`, su nombre, y es un comando disponible a través de la **Instrument Control Toolbox™ Bluetooth®** de Matlab. En el caso de la versión de Matlab utilizada para desarrollar la aplicación (R2106a) esta Toolbox venia implementada de serie, pero en versiones anteriores quizás haya que instalarla manualmente.

Creado el objeto que contiene la conexión, ya podemos acceder a la información contenida en la misma, y esto, programáticamente, ha sido implementado tal y como aparece a continuación:

Conexión con dispositivo Arduino mediante Bluetooth y aceso a acelerómetro

```
% Almacena en una variable global el objeto bluetooth
global a;
% Creo el objeto con la conexión-> Esto tarda unos segundos
a=Bluetooth(get(handles.ardport,'String'),1)
% Abro el objeto bluetooth, para poder leerlo después
% -> Esto también tarda unos segundos
fopen(a);
% Leo el objeto bluetooth: Se envían los datos tal que:
% "#XXXX YYYY ZZZZ", que corresponden con la salida digital entre 0 y
% 1023 de cada eje.
p=fgets(a);
% No leo el primer elemento de la cadena de caracteres, ya que es #,
% que sirve para detectar el inicio del registro.
A= sscanf(p(2:end),'%f %f %f');
```

Cabe destacar que, a diferencia del caso anterior, en este caso no se lee el registro completo cuando se ejecuta el comando de lectura (`sscanf(p(2:end),'%f %f %f')`), si no solo el último registro recibido, con lo que no se necesita la consideración tenida en el registro vía smartphone.

Por el contrario, algo que sí se ha tenido que implementar para este tipo de registro ha sido, tal y como se hizo en la aplicación smartphone, una media móvil de cara a suavizar el ruido del registro. Esto se ha implementado de la siguiente forma:

Media móvil para el registro a través del Arduino

```
%   Variables para la media móvil. Almacenarán los 3 últimos registros
      xprev=zeros(1,3);
      yprev=zeros(1,3);
      zprev=zeros(1,3);

%   Roto los valores almacenados en un vector de tres posiciones para
%   calcular la media móvil
      xprev=circshift(xprev,[0,1]);
      yprev=circshift(yprev,[0,1]);
      zprev=circshift(zprev,[0,1]);

%   Almaceno los nuevos:
      xprev(1)=A(1);
      yprev(1)=A(2);
      zprev(1)=A(3);

%   Comienzo el registro. Para ello paso el valor que viene del
%   Arduino a m/s^2
      addpoints(hx,toc-initialTime,(mean(xprev)*...
          (3.3/1023)-1.65)*(10/0.8))
      addpoints(hy,toc-initialTime,(mean(yprev)*...
          (3.3/1023)-1.65)*(10/0.8))
      addpoints(hz,toc-initialTime,(mean(zprev)*...
          (3.3/1023)-1.65)*(10/0.8))
```

Y con esto quedaría documentada la implementación de esta conexión y su uso para registro.

OTRAS CONSIDERACIONES DEL DESARROLLO

Para terminar de exponer los aspectos tenidos en cuenta en el desarrollo de la aplicación en Matlab, comentar brevemente dos consideraciones:

- El guardado se ha implementado de manera análoga a como se implementó en la aplicación Smartphone: archivo .txt con el mismo formato dado a la aplicación anterior, de tal manera que sean intercambiables, y con el mismo nombre por defecto, fecha y hora de guardado del registro.
- Se han utilizado herramientas y funciones de Matlab no disponibles en todas las versiones del software. En concreto, la función más moderna de las utilizadas es la `animatedline`, introducida en la versión de Matlab R2014b, con lo que para hacer funcionar correctamente la aplicación se necesitará usar esta versión o una posterior.

CAPÍTULO 3: SENSOR EXTERNO: ARDUINO

En el presente capítulo se va a documentar la parte del trabajo abordada con el uso de un Arduino. Como se ha ido exponiendo a lo largo de la memoria y como anuncia el título de este capítulo, el uso del Arduino está enfocado a la utilización de un sensor externo al dispositivo Smartphone y como sustitutivo a los sensores que este incluye.

En cuanto al por qué de incluir esta funcionalidad, la razón principal ha sido la de dotar a ambas alternativas propuestas en este trabajo, la aplicación Smartphone y la aplicación de escritorio de Matlab, de la funcionalidad de registro a través de un elemento específico para ello (la disposición diseñada del Arduino consiste únicamente en un registrador que envía la señal procedente de un acelerómetro) de bajo coste y alternativo a los sensores embebidos del smartphone. Con esta decisión, se provee de aún más alternativas para la consecución de los objetivos planteados, ofreciendo al posible usuario o futuro desarrollador de la aplicación de más posibilidades de sensorización.

Además, se ha optado por el uso del Arduino y no de otras alternativas como una Raspberry Pi gracias a que este funciona como un microcontrolador que procesa únicamente las tareas que se le asignan, aliviando así de una posible carga extra de trabajo que en el caso de las aplicaciones desarrolladas ya llevan a cabo los hardwares sobre los que se ejecutan dichas aplicaciones.

3.1-INTRODUCCIÓN A ARDUINO

Antes de desarrollar el trabajo realizado con este hardware, conviene comentar brevemente algunos aspectos del mismo, como qué es exactamente y qué podemos hacer con él.

Si se acude a la web del fabricante y distribuidor del Arduino original, que por cierto no es el que se ha usado para el desarrollo presente, nos dice que Arduino es una plataforma electrónica *open-source* basada en hardware y software de fácil uso. Esto significa que se trata de un componente electrónico basado en código abierto y orientado a ser una plataforma de entrada a la electrónica para cualquier nivel de usuario.

Por lo tanto, Arduino provee a la comunidad de desarrolladores de diferentes ámbitos con microcontroladores de placa única (*single-board*) y kits para construir dispositivos digitales que interaccionen con objetos físicos, a través de actuadores o sensores de todo tipo, o digitales, a través de, por ejemplo, internet. Esto es posible gracias a que tanto hardware como software se distribuye, como ya se ha comentado, como *open-source* y bajo licencia GNU *Lesser General Public License* (LGPL) o GNU *General Public License* (GPL).

En cuanto a su uso, Arduino ofrece, de acuerdo a su distribuidor, de una serie de ventajas sobre sus competidores en el campo de los microcontroladores como *Parallax Basic Stamp*, *Netmedia's BX-24*, *Phidgets* o *MIT's Handyboard* entre otros. Como ventaja principal expone su **bajo coste**, puesto que pueden incluso ser construidas por el propio usuario, ya que los esquemas de los circuitos están disponibles. Aún en el caso de comprar una ya ensamblada, una oficial oscila entre los 20 € y los 50 €, y una no oficial mucho menos, como la usada en el presente proyecto, que costó tan solo 4 €.

3.1.1-HARDWARE

Existen disponibles una gran variedad de microcontroladores y dispositivos para interactuar con estos. En la figura 3.1.1.1 podemos ver los oficiales que ofrece Arduino.

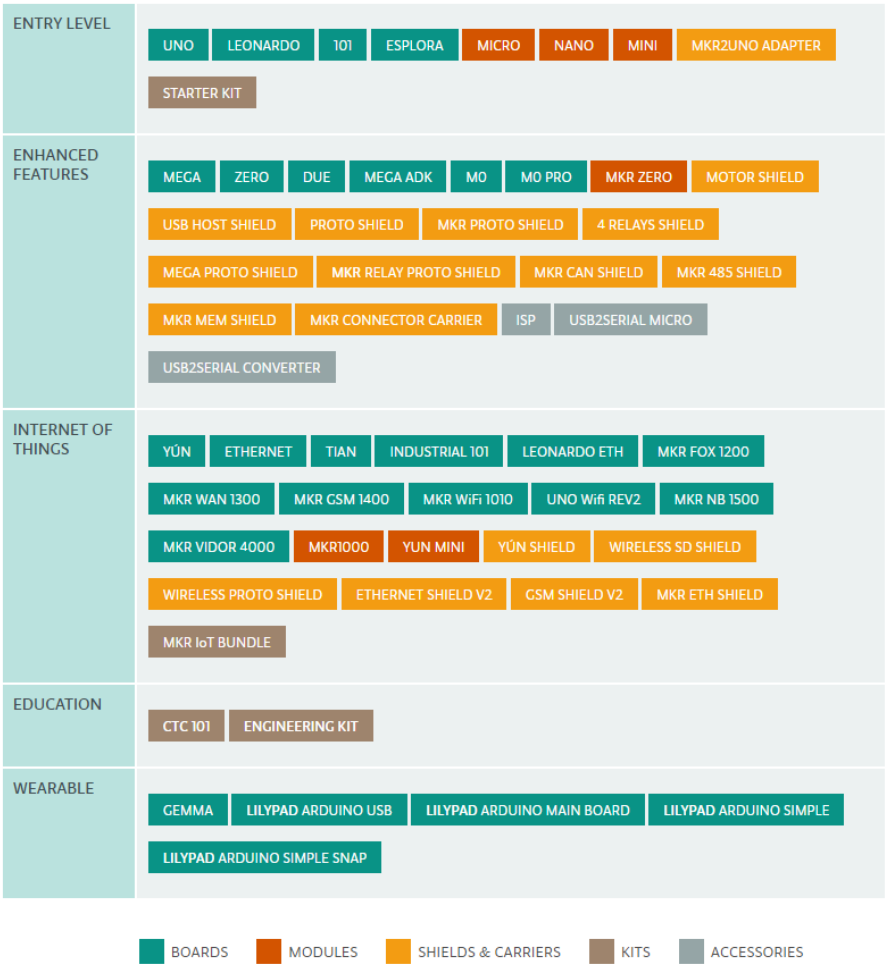


Figura 3.1.1.1-Placas, módulos, shields & carriers, kits y accesorios ofrecidos por Arduino. Fuente: https://www.arduino.cc/en/Main/Products

La mayoría de las placas Arduino consisten en un microcontrolador Amtel de 8-bits (aunque las hay de hasta 32) con una cantidad variable de memoria flash, pins de entrada/salida digitales y analógicos, además de otras características como la comunicación serie gracias a una conexión USB (Universal Serial Port) que permite su comunicación con cualquier dispositivo que acepte comunicación serial, además de la programación de la placa, y que se implementa mediante chip adaptador USB-serie tales como el FTDI FT232. La mayoría de estas placas incluyen también un regulador lineal de 5 V y un oscilador de cristal o resonador cerámico de 16 MHz, aunque algunos funcionan a 8 MHz.

Dado el gran abanico y variedad de modelos de Arduino y sus diferentes complementos, existen infinidad de aspectos relativos a sus hardwares, pero no es objeto del presente trabajo su desarrollo, con lo que es suficiente con estas pinceladas para hacerse una idéa general del hardware de este microcontrolador. Más adelante se profundizará en el hardware específico usado para el presente trabajo.

3.1.2-SOFTWARE-PROGRAMACIÓN

Al tratarse el Arduino de un microcontrolador, admite cualquier lenguaje de programación de alto nivel que posea un compilador a código máquina.

En cuanto a su IDE (Entorno Integrado de Desarrollo), se trata de un entorno multiplataforma programado en Java, que además de permitir la programación del dispositivo, provee de la capacidad de compilar y cargar dicho programa, además de leer las salidas del Arduino a través del puerto serie. Se puede ver una captura de la IDE en la figura 3.1.2.1.

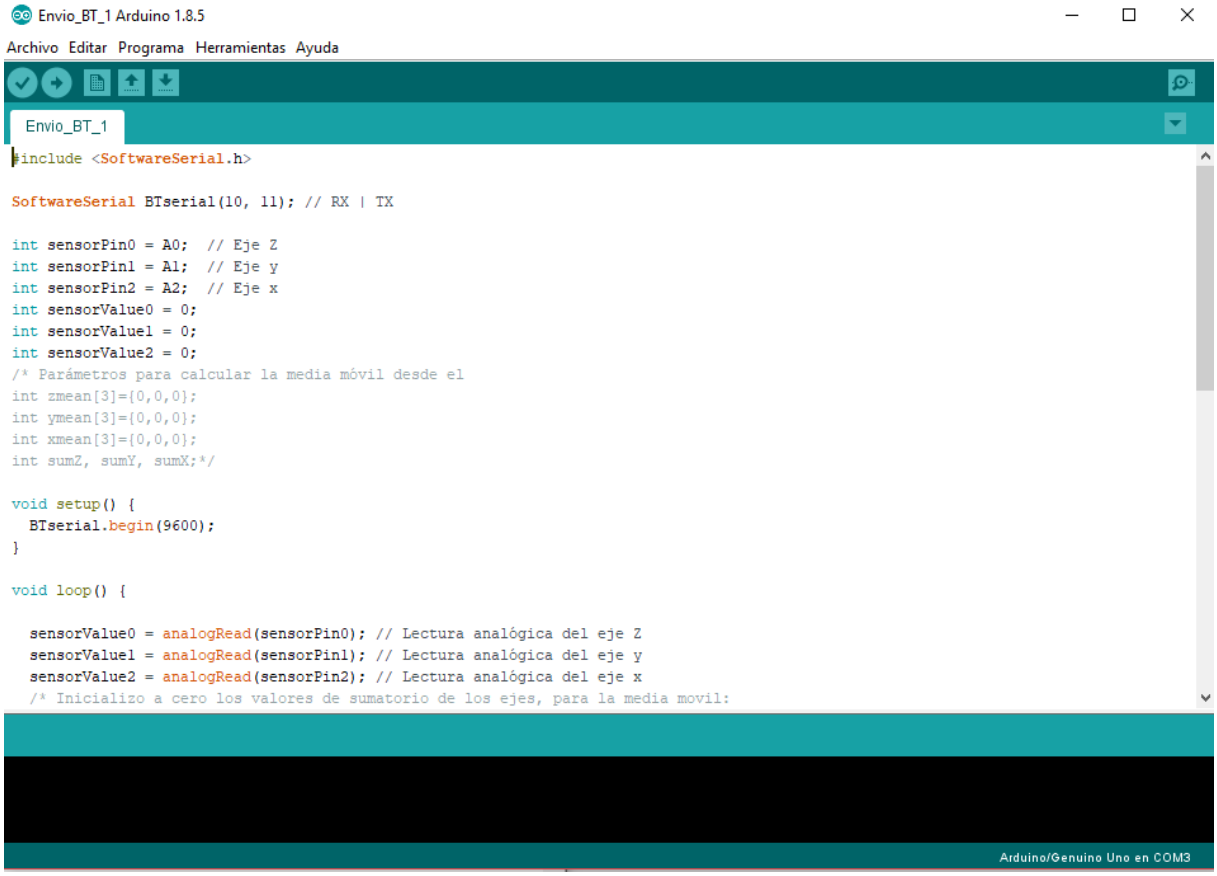


Figura 3.1.2.1 -Captura del programa en la IDE de Arduino.

Esta IDE soporta los lenguajes de programación C y C++ usando reglas especiales para la estructura del código. Además, incluye numerosas librerías procedentes de las IDEs originales del proyecto, *Processing* and *Wiring*, que habilita funciones comunes para el tratamiento de *inputs* y *outputs*.

Los programas para Arduino, denominados Sketch, consisten básicamente en dos funciones:

- **setup()**: se llama esta función al inicializar el programa, cada vez que se da corriente al Arduino o se le resetea. Su uso principal es el de instanciar variables, declara los modos de funcionamiento de los pines o declarar otras librerías de las que haga uso el programa.
- **loop()**: se ejecuta cíclicamente una vez que se finaliza *setup(),* y no finaliza hasta que se corta la corriente de la placa o se resetea.

3.2-DESCRIPCIÓN DEL HARDWARE

Una vez introducido el hardware Arduino a nivel genérico, es momento de ahondar en el específico usado para el desarrollo del presente trabajo.

Para implementar las capacidades requeridas para el cumplimiento de los objetivos iniciales, que en este caso específico son registrar y enviar por bluetooth datos de aceleración en tiempo real, ha sido necesario los siguientes elementos de hardware:

- Placa Arduino
- Módulo bluetooth
- Acelerómetro de tres ejes

Y los elementos específicos seleccionados han sido los expuestos a continuación.

3.2.1-ARDUINO UNO

Como placa microcontrolador se ha elegido uno de la gama de entrada, el Arduino UNO, por ser suficientemente completo, sencillo de utilizar, robusto y con cantidad de documentación sobre infinidad de proyectos disponibles.

Las especificaciones de este microcontrolador son las que se muestran en la figura 3.2.1.1.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figura 3.2.1.1 -Especificaciones técnicas de un Arduino UNO Rev3, análogas a las especificaciones del Arduino clónico utilizado. Fuente: https://store.arduino.cc/usa/arduino-uno-rev3

El hardware concreto utilizado para el desarrollo de la aplicación ha sido un Arduino UNO clónico, en concreto el Arduino UNO ch340 como el que se muestra en la figura 3.2.1.1, de manufactura china, pero con las mismas características que el original puesto que,

recordemos, es hardware abierto y los esquemas del original están disponibles para su uso y fabricación por cualquiera.

Figura 3.2.1.1 -Arduino UNO R3 ATmega328P CH340G. Fuente: https://www.makerlab-electronics.com/product/arduino-uno-r3-atmega328p-ch340g/

3.2.2-ACELERÓMETRO

En el caso del acelerómetro se ha optado por un MMA7361LC de 3 ejes como el que se muestra en la figura 3.2.2.1.

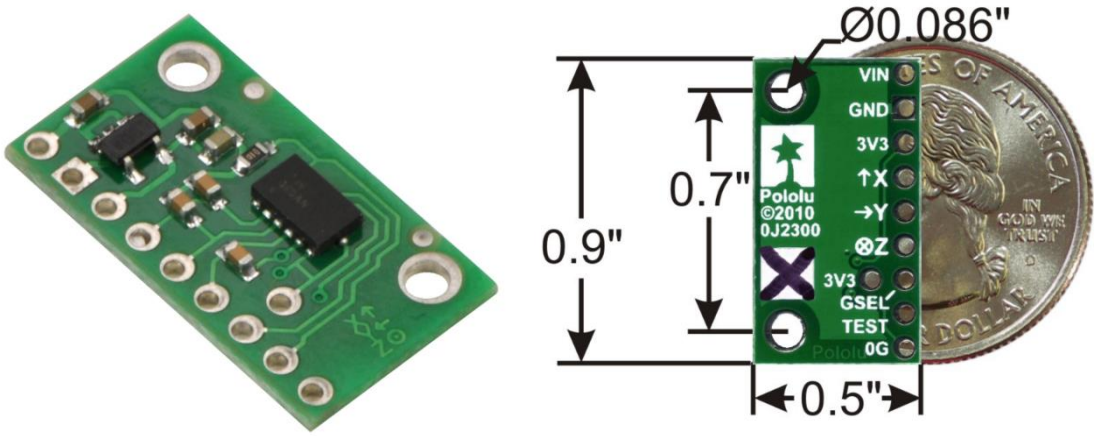


Figura 3.2.2.1 -Acelerómetro de 3 ejes MMA7361LC. Fuente: https://www.pololu.com/product/1251/specs /

Las especificaciones técnicas del mismo son las que se muestran en las siguientes tablas:

Dimensions

Size:	0.5" x 0.9" x 0.1"
Weight:	0.7 g

General specifications

Interface:	analog voltage
Minimum operating voltage:	2.2 V
Maximum operating voltage:	16 V
Measurement range:	±1.5 or ±6 g
Supply current:	0.5 mA

Tabla 3.2.2.1-Especificaciones del MMA7361LC. Fuente: https://www.pololu.com/product/1251/specs /

En cuanto a los pines del dispositivo, dispone de los siguientes:

- **Vin**: permite la alimentación del acelerómetro de manera independiente medainte una vatería o fuente de alimentación de entre 2.2 y 16V
- **GND**: conexión a tierra
- **3V3**: alimentación para los ejes
- **X,Y,Z**: lecturas de cada eje
- **g-Select**: pin para selecciónar la sesivilidad
- **Self Test**: puede dejarse desconectado
- **0g-Detect**: da señal cuando se detecta caida libre

Gracias a estas especificaciones se considera este acelerómetro suficientemente apto para la toma de medidas.

3.2.3-MÓDULO BLUETOOTH HC-06

Para cubrir la necesidad de comunicación bluetooth se ha optado por un módulo bluetooth serial HC-06.

Figura 3.2.3.1 -Módulo bluetooth HC-06. Fuente: https://www.gearbest.com/sensors/pp_241478.html

Este dispositivo funciona como esclavo, habilitando una única conexión con un dispositivo maestro, en este caso el Smartphone o el PC. El esquema de este funcionamiento es el que se muestra en la figura 3.2.3.2.

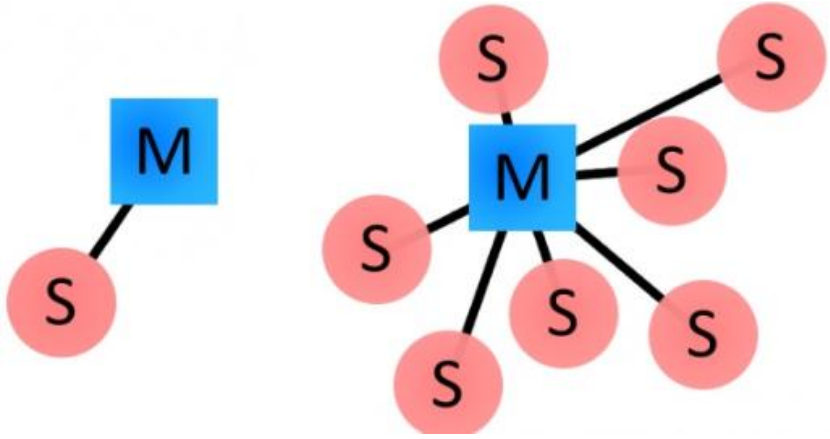


Figura 3.2.3.2-Diferencia de comportamiento entre dispositivo bluetooth esclavo (izquierda) y maestro (derecha). Fuente: https://www.prometec.net/bt-hc06/

Este dispositivo reúne las siguientes características:

Protocolo Bluetooth:	Bluetooth 2.0+ EDR standard
Protocolo USB:	USB v1.1/2.0
Frecuencia Operativa:	2.4GHz ISM frequency band
Modo de modulación:	Gauss frequency Shift Keying
Potencia de transmisión:	≤ 4dBm, second stage
Sensibilidad:	≤-84dBm at 0.1% Bit Error Rate
Velocidad de transmisión:	2.1Mbps(Max)/160 kbps(Asynchronous) ; 1Mbps/1Mbps(Synchronous)
Características de seguridad:	Authentication and encryption
Configuración permitida:	Bluetooth serial port (major and minor)
Voltaje de alimentación:	+3.3 VDC 50mA
Temperatura Operativa:	-20 to 55℃
Tamaño:	36.5*16mm
Peso:	4 g

Tabla 3.2.3.1-Características del módulo HC-06.
Fuentehttp://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06

Este módulo dispone de los siguientes pins:

- **STATE**: muestra el estado, análogo al led que ya integra el módulo
- **RXD**: terminal de recepción de la terminal serial
- **TXD**: terminal de transmisión de la terminal serial
- **GND**: puesta a tierra
- **VCC**: alimentación, entre 3.6 y 6.6 V
- **KEY**: para habilitar el modo AT

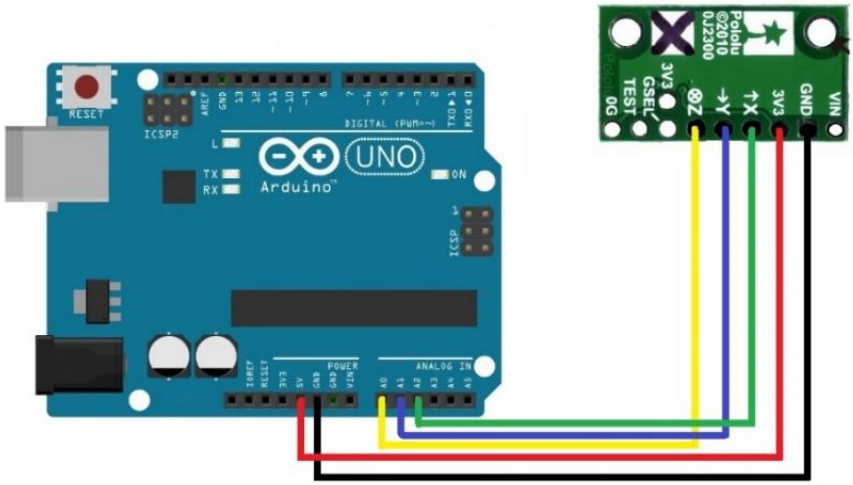
Gracias a estas especificaciones se considera este acelerómetro suficientemente apto para la toma de medidas.

3.3-ESQUEMA DE MONTAJE

Introducido el hardware empleado en el desarrollo de este trabajo, es necesario también describir su montaje.

Atendiendo a la conexión entre los módulos y el microcontrolador, los esquemas de montaje son los siguientes:

➢ **Arduino-acelerómetro:**

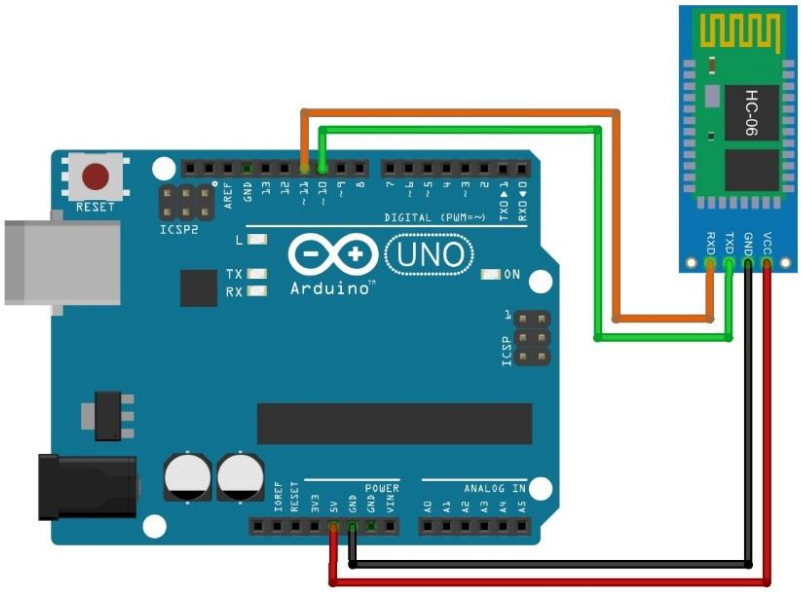


Conexiones pin acelerómetro-Arduino:

- GND-GND
- 3V3-5V
- X-A2
- Y-A1
- Z-A0

Figura 3.3.1-Conexiones entro los pines del acelerómetro y el Arduino.

➢ **Arduino-bluetooth:**



Conexiones pin bluetooth- Arduino:

- VCC-5V
- GND-GND
- TXD-10
- RXD-11

Figura 3.3.2-Conexiones entro los pines del bluetooth y el Arduino.

En cuanto al esquema general de montaje, dado que se ha requerido de una protoboard para no dar al montaje del prototipo un carácter permanente, ha quedado como se muestra en las siguientes figuras:

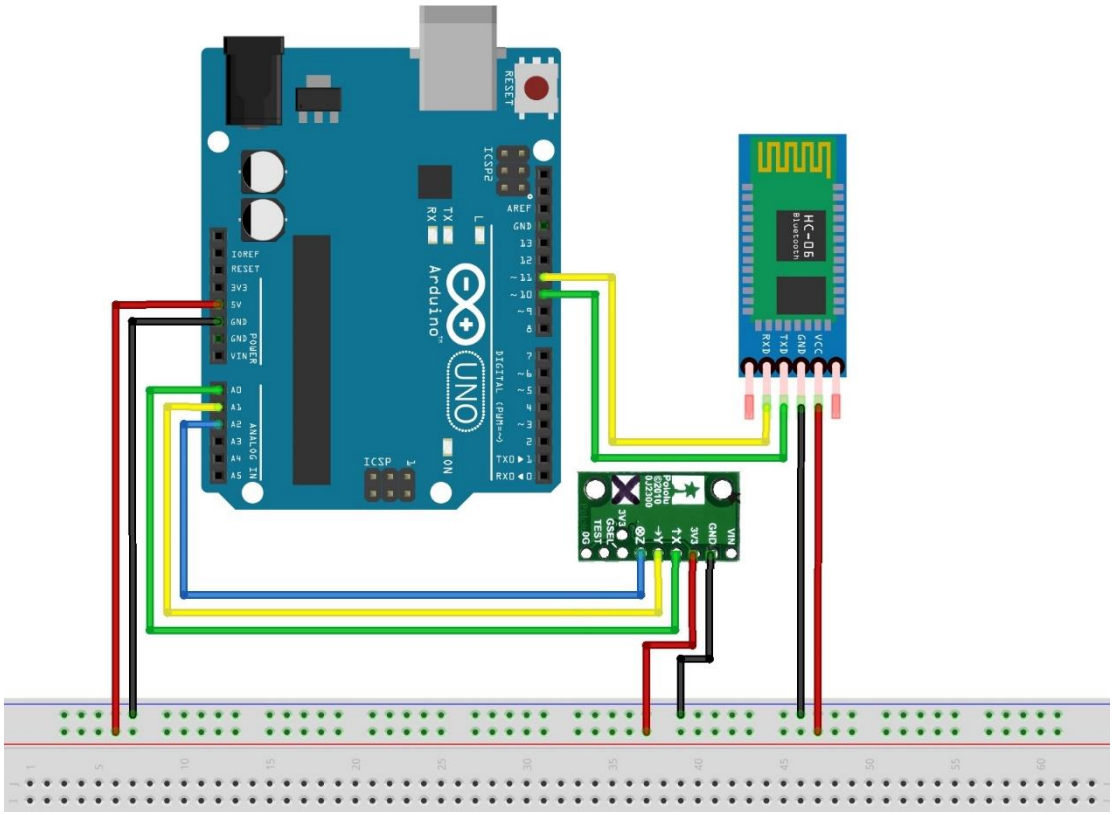


Figura 3.3.3-Esquema de montaje general del Arduino con el módulo bluetooth y el acelerómetro.

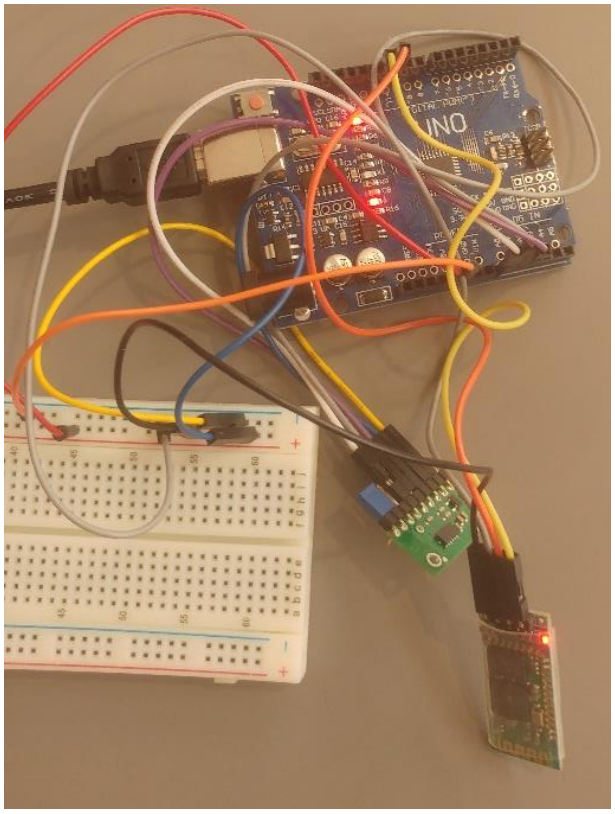


Figura 3.3.4-Foto del montaje general del Arduino con el módulo bluetooth y el acelerómetro.

3.4-IMPLEMENTACIÓN PROGRAMÁTICA DEL ENVÍO

Como ya se ha comentado, el envío de los datos obtenidos por el acelerómetro se efectúa a través de bluetooth con independencia de cuál sea el hardware al que este se conecta el Arduino. Se ha optado por esta implementación para dotar al dispositivo de autonomía y cierto rango (aproximadamente 10 metros de acuerdo a las especificaciones del módulo HC-06).

Para implementar esta solución, se ha establecido una comunicación serie a través de la librería **SoftwareSerial**.h, que nos permite crear este protocolo de comunicación fácilmente. En concreto, se ha implementado de la siguiente forma:

Implementación de la conexión serie del Arduino para la comunicación bluetooth

```
#include <SoftwareSerial.h>

SoftwareSerial BTserial(10, 11); // TX | RX


void setup() {
  BTserial.begin(9600);
}
```

De esta manera, conectamos los pines digitales 10 y 11 con los pines TX y RX del módulo HC-06 respectivamente. Se ha optado por esta configuración por ser la estándar de conexión del módulo, pero en realidad se necesitaría conectar el pin TX (transmisión), puesto que la única finalidad de este dispositivo es transmitir registro, y no va recibir nada. Esta comunicación serie se establece a 9600 dbs, que es la configuración por defecto.

En cuanto a la transmisión de estos datos, se lleva a cabo mediante el comando *print*, tal que `BTserial.print(sensorValue2)`, y el formato dado al registro de salida se corresponde a lo siguiente:

#XXXX YYYY ZZZZ;

De tal manera que se disponga de un carácter "#" para identificar el principio del registro, "**;**"para el final y espacios entre los diferente ejes para facilitar la recepción del registro. Los valores procedentes del acelerómetro serán enviados con un valor de entre 0 y 1023, correspondiente a la digitalización del valor del voltaje procedente del pin analógico al que se conecta cada uno de los ejes del acelerómetro. Las siguientes figuras (figura 3.4.1) muestran la recepción en los distintos dispositivos utilizados:

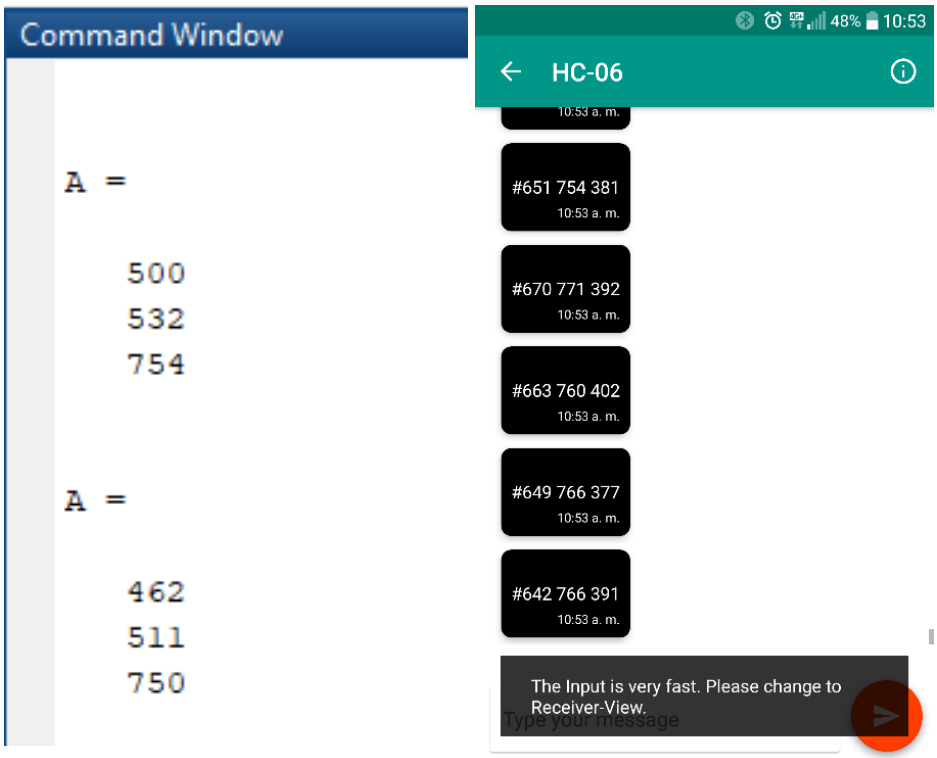


Figura 3.4.1-Capturas de pantalla de la recepción del registro enviado por bluetooth desde el Arduino. A la izquierda, envío recibido a sesión de escritorio de Matlab. A la derecha, Envío recidivo al dispositivo smartphone y visualizado a través de la aplicación ArduTooth, disponible en GooglePlay y de utilidad para visualizar conexión.

Por último, destacar que se ha optado por el envío del registro en bruto, pero se encuentra también disponible en el código del Arduino, implementada en comentario, la opción de enviar el registro tratado con una media móvil.

3.5-TRATAMIENTO DEL REGISTRO

Una vez tratado el envío y visto el formato en el que lo recibirán los diferentes dispositivos, que recordando es tal que `#XXXX YYYY ZZZZ;`, es necesario explicara como se pasa de este formato, donde los valores registrados por el acelerómetro oscilan entre 0 y 1023, a m/s^2.

Para ello, se acude a la información sobre el acelerómetro ofrecida por el fabricante, de donde extraemos, de entre toda la información que nos ofrece, la siguiente información relevante para esta transformación:

- Regulador lineal de voltaje a 3.3V, de tal manera que la conexión establecida a 5V con el Arduino se reduce a 3.3V en los pines de los ejes.
- La salida analógica de los pines correspondientes a los ejes estará siempre entre 0 y 3.3V, centrada, salvo presencia de *offset* en el hardware, en el valor medio de este rango.
- La sensibilidad por defecto es de 800mV/g, pudiéndose modificar a 440mV/g mediante el pin g-Select. En este caso, se conserva la sensibilidad por defecto, puesto que, aunque el rango de medida es menor, la sensibilidad es mayor, resultando esta configuración de mayor interés para este trabajo.

Con esta información se ha tratado el registro para darlo el formato adecuado, y dicho procedimiento ha consistido en lo siguiente:

1º- Pasar del formato de salida del Arduino (entre 0 y 1023), a voltaje, multiplicando el valor procedente del pin por 3.3/1023

2º- Pasar el voltaje a m/s^2, a través de la sensibilidad declarada en la documentación y el valor supuesto de g, de tal manera que multiplicamos el valor anterior por 9.8 m/s2 y lo dividimos por 0.8 V/g

Esto queda tal que:

$$Valor\ del\ registro\ \left({}^{m}/_{s^2}\right) = \left(\frac{Valor\ del\ registro}{1023} \cdot 3.3\ (V) - 1.65(V)\right) \cdot \frac{9.8\ \left({}^{m/s^2}/_{g}\right)}{0.8\ \left({}^{V}/_{g}\right)}$$

Cabe destacar que este registro puede tener un *offset* en cada eje, cosa que sin ir más lejos ocurre en el acelerómetro utilizado para este trabajo. Este *offset*, que será distinto para cada acelerómetro con que se monte el esquema descrito, no ha sido eliminado en la etapa de registro. Esta decisión se ha tomado puesto que ya se ha incluido la funcionalidad de eliminar la tendencia lineal en las aplicaciones desarrolladas, de cara a eliminar la gravedad, con lo que, mediante su aplicación, se eliminará también el offset.

CAPÍTULO 4: LÍNEAS FUTURAS

En el presente capítulo se van a proponer y describir posibles implementaciones futuras a las aplicaciones basadas principalmente en el trabajo realizado y en los conocimientos adquiridos durante el desarrollo del mismo. El objetivo de este capítulo es, por tanto, ofrecer alternativas a posibles futuros desarrollos basados en las aplicaciones desarrolladas, facilitando esta tarea planteado posibles implementaciones de estas líneas futuras en base a las tecnologías/aplicaciones utilizadas.

4.1- PLANTEAMIENTO SOBRE POSIBLES LÍNEAS FUTURAS

Antes de comentar las diferentes propuestas de desarrollo y su posible implementación, es conveniente introducir el posible planteamiento a adoptar para futuras revisiones o trabajos que continúen este mismo.

Una vez se ha implementado la funcionalidad de registrar y almacenar mediante un dispositivo/sensor, el paso natural es registrar con varios sensores a la vez. Esto permitiría crear una red de sensores (*sensor network*) que permitiría medir y caracterizar al mismo tiempo diferentes puntos de una estructura, u otros elementos si se desea. Esto permitiría comprobar y comparar cómo se comportan diferentes puntos del elemento afectado ante una misma perturbación.

Además de la capacidad para caracterizar el elemento medido en su conjunto y no sólo en un punto, la opción de establecer una red de sensores con estos elementos de bajo coste abre la posibilidad de registrar no solo el efecto producido sobre el elemento medido, si no también realizar un registro sobre el elemento que genera la perturbación. Esto sería, por ejemplo, en un puente peatonal, caso de estudio que ha dado pie al presente trabajo, registrar al mismo tiempo con uno o varios smartphones las aceleraciones producidas en el puente por el paso de peatones, los cuales a su vez registran parámetros referentes a su paso por el puente (como frecuencia de paso, velocidad de desplazamiento, posición, etc.). De esta forma se conseguiría medir al mismo tiempo causa y efecto, dando pie a un estudio más profundo y completo del comportamiento del elemento en cuestión.

Figura 4.1.1-Posibilidad de red de sensores que registre aceleraciones producidas en un puente por peatones, cuyo paso por el puente también se registre.

Otras posibles evidentes líneas futuras de desarrollo de las aplicaciones pasan utilizar diferentes sensores de registro más precisos o depurar y optimizar las aplicaciones ya desarrolladas, pero se ha decidido desarrollar la posibilidad de interconexión antes que el resto por considerarse de mayor interés y ofrecer mayores posibilidades.

4.2-DESARROLLOS FUTUROS

Introducido el planteamiento a seguir para desarrollar estas líneas futuras, se pasa a comentar las mismas y su posible implementación.

4.2.1-CONEXIÓN DE VARIOS DISPOSITIVOS A MATLAB

La primera opción que se plantean para establecer el sistema de sensores es la de conectar los elementos a la aplicación de Matlab en un PC. Para esto, se podrían plantear las siguientes soluciones:

CONEXIÓN CON ARDUINOS

Esta solución consistiría en establecer la red de sensores utilizando Arduinos siguiendo los esquemas correspondientes a este hardware, ya utilizados en el trabajo. Este sistema se correspondería a lo que se ve en la figura 4.2.1.1.

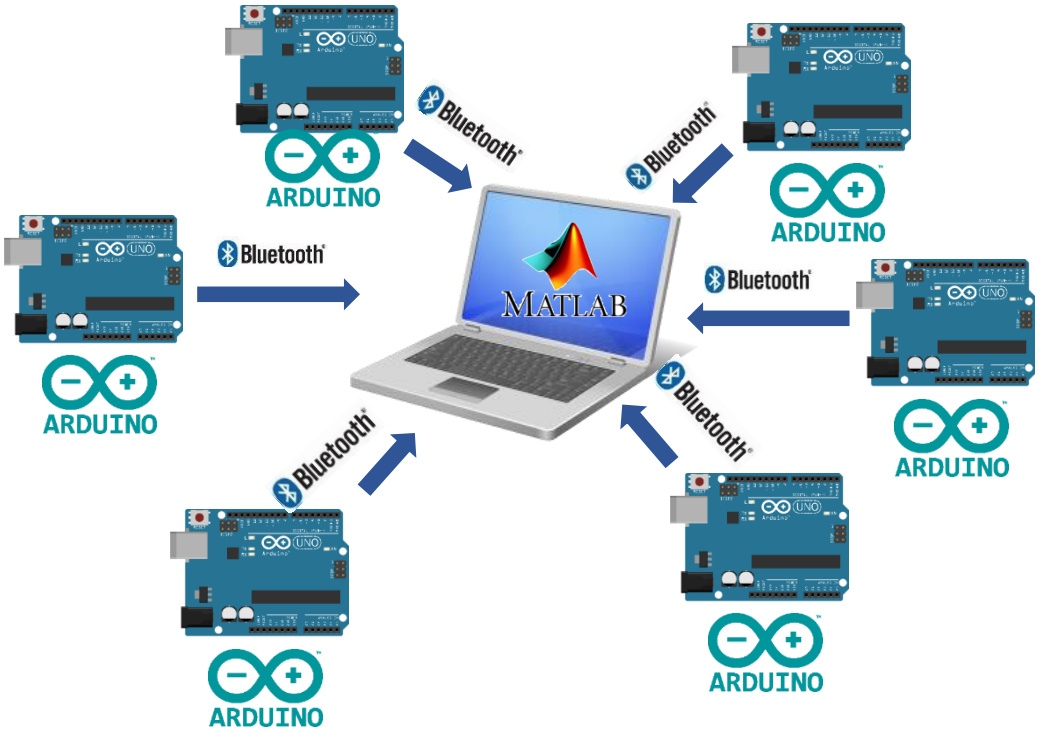


Figura 4.2.1.1-Posibilidad de red de sensores conectando Arduinos a Matlab mediante bluetooth.

POSIBLE IMPLEMENTACIÓN

Una posible forma de abordar la implementación de esto es, simple y llanamente, utilizar el objeto provisto por Matlab para conexiones bluetooth, `Bluetooth('Name',1)`. Este objeto ya se ha usado, y su implementación en el caso que se aborda parece trivial, lo único que haría falta es cambiar el nombre con el que se empareja cada dispositivo (que por defecto es HC-06, pero si queremos conectar varios no podemos hacerlo si tienen el mismo nombre). Una vez hecho esto, se establecerá el objeto creando mediante el comando visto, donde el numeró tras `'Name'` es el canal donde se establece la conexión. Se deberá definir un canal para cada dispositivo, lo que se hace simplemente cambiando el número de canal (1,2,… etc.).

```
>> a=Bluetooth('G5',1)

   Bluetooth Object : Bluetooth-G5:1

   Communication Settings
      RemoteName:          G5
      RemoteID:            btspp://5C70A3F22408
      Channel:             1
      Terminator:          'LF'

   Communication State
      Status:              closed
      RecordStatus:        off

   Read/Write State
      TransferStatus:      idle
      BytesAvailable:      0
      ValuesReceived:      0
      ValuesSent:          0
```

```
>> b=Bluetooth('HC-06',2)

   Bluetooth Object : Bluetooth-HC-06:2

   Communication Settings
      RemoteName:          HC-06
      RemoteID:            btspp://98D331601E37
      Channel:             2
      Terminator:          'LF'

   Communication State
      Status:              closed
      RecordStatus:        off

   Read/Write State
      TransferStatus:      idle
      BytesAvailable:      0
      ValuesReceived:      0
      ValuesSent:          0
```

Figura 4.2.1.2-Resultado de emparejar dos dispositivos con Matlab mediante bluetooth.

Una vez se establecen las conexiones pertinentes, la gestión de datos se implementaría programáticamente de manera análoga a lo visto en el apartado correspondiente, en Matlab, y cuyo código se encuentra disponible en los anexos.

Aparentemente esta es una solución sencilla y que no debería conllevar ningún problema en su implementación, pero no se ha podido probar en el presente trabajo por no disponerse de varios Arduinos.

CONEXIÓN CON SMARTPHONES

Otra opción es usar smartphones como sensores, simultaneando de nuevo la conexión de varios de los mismos con la aplicación de escritorio de Matlab. Sin embargo, su implementación no resulta tan trivial como en el caso anterior, con lo que podemos establecer varios planteamientos.

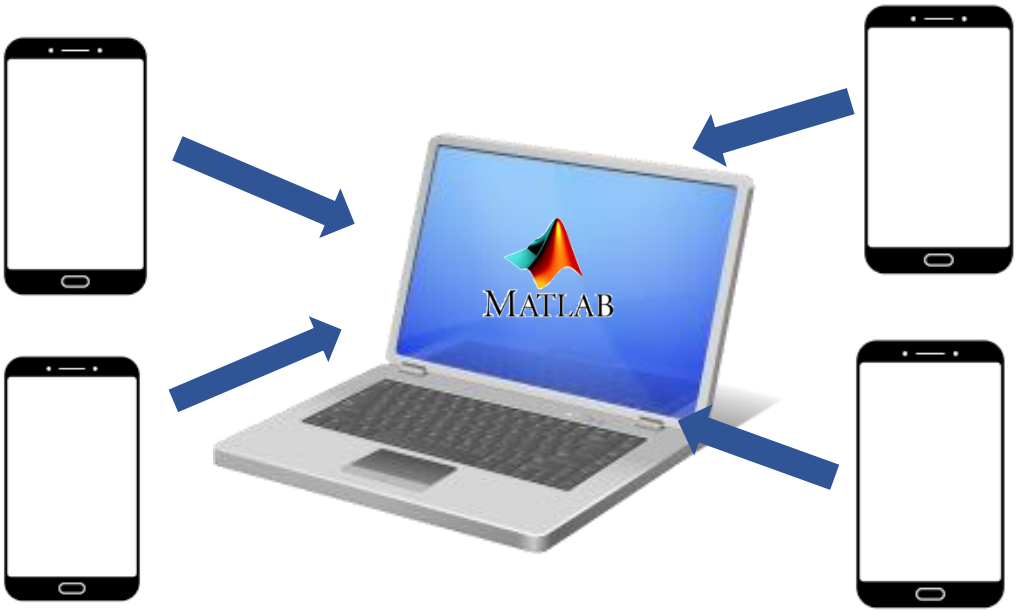


Figura 4.2.1.3- Posibilidad de red de sensores conectando Smartphones a Matlab.

POSIBLE IMPLEMENTACIÓN MEDIANTE WIFI

Al igual que en el caso anterior, se puede plantear la implementación de este sistema utilizando las herramientas ya utilizadas, en este caso el comando `connecto on`, para establecer una comunicación con la aplicación Matlab MOBILE instalada en cada dispositivo y que sea esta la que envíe los datos procedentes de los sensores embebidos.

Pero a diferencia del caso anterior, donde parece trivial, en este caso la implementación no es tan sencilla, puesto que el comando de Matlab establece una única conexión por sesión, de tal forma que, si conectamos varios dispositivos a dicha sesión y creamos los objetos con el comando `mobiledev`, nos mezclará los registros, puesto que no existe la opción de identificarlos y diferenciar los distintos dispositivos.

Aún con esto, se puede aprovechar en cierta medida esta funcionalidad implementada por Matlab, y para ello se puede plantear el siguiente sistema:

- Arrancar tantas sesiones como dispositivos se quieran conectar
- Arrancar una sesión extra que funcionará como inicializador, a través de un archivo *.ini

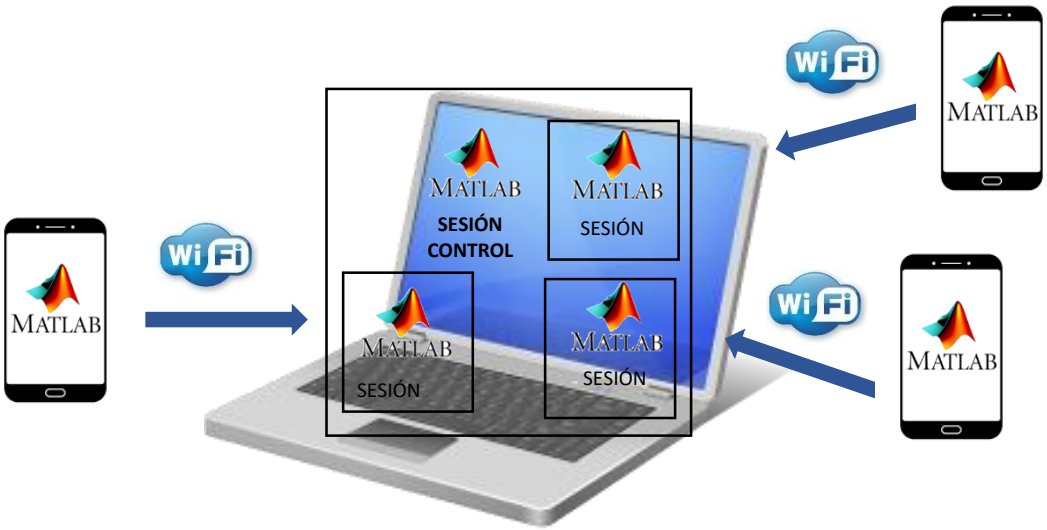


Figura 4.2.1.4- Posibilidad de red de sensores conectando Smartphones a Matlab usando las aplicaciones dadas por Matlab, mediante el uso de múltiples sesiones.

Una vez se establece este sistema, la forma de proceder sería la siguiente:

- Cada sesión asociada a los dispositivos smartphones inicia un bucle que lee el archivo *.ini, el cual contiene un número, 0 o 1, de tal manera que se lance el registro cuando este pase de 0 a 1
- Desde la sesión de control modificamos el parámetro contenido en el archivo *.ini, de tal manera que iniciamos el registro en el resto de sesiones (pasando de 0 a 1) o lo detenemos (pasando de 1 a 0)

De esta forma se conseguiría que cada sesión registre simultáneamente su correspondiente smartphone asociado.

PROBLEMAS DE LA IMPLEMENTACIÓN

Esta posible línea de desarrollo presenta, a la luz de diferentes pruebas realizadas, una serie de inconvenientes importantes, siendo la más destacada la falta de eficiencia derivada del uso de múltiples sesiones de Matlab ejecutando bucles de manera ininterrumpida.

Este problema se ha visto en pruebas realizadas conectando simultáneamente dos smartphones a sendas sesiones de Matlab, usando una tercera como inicializadora. Este sistema presentaba ralentizaciones debido a la carga de trabajo que suponía, lo cual, teniendo en cuenta que el equipo utilizado contaba con procesador Intel® Core™ i7-6820HK (4 núcleos, 2.7 GHz) y 16 Gb de RAM, da fe del pobre rendimiento de esta disposición.

Otro problema de este planteamiento es la inicialización de los registros, que no será sincronizada, puesto que cada uno ejecuta el bucle *while* independientemente.

Una posible alternativa sería usar un complemento de Matlab, el "Add-On" SHAREDMATRIX, que permite la gestión de múltiples sesiones y comunicación entre las mismas de manera más eficiente, trabajando en un espacio de memoria RAM conjunto y reservado para las distintas sesiones. Por desgracia el uso de este complemento no se ha podido probar debido a que sólo se encuentra disponible para sistemas operativos Linux.

POSIBLE IMPLEMENTACIÓN MEDIANTE BLUETOOTH

Otro posible planteamiento es el de establecer las conexiones entre los terminales smartphones y la sesión Matlab mediante comunicación bluetooth, de manera análoga a lo propuesto con Arduinos y tal y como se muestra en la figura 4.2.1.5.

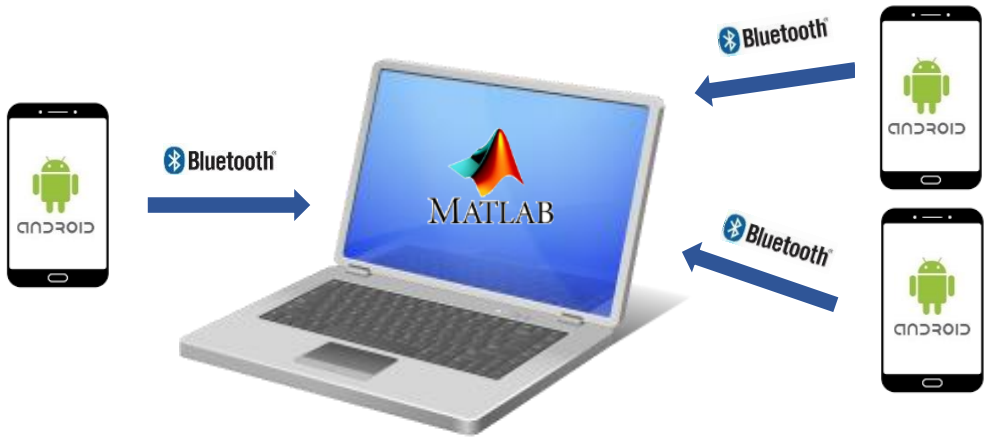


Figura 4.2.1.5- Posibilidad de red de sensores conectando Smartphones a Matlab mediante comunicación bluetooth.

En la parte referente a Matlab, el procedimiento sería análogo a lo visto en el apartado de Arduinos: creando los diferentes objetos bluetooth mediante la función dada por Matlab, cada uno con su respectivo nombre identificador y canal. Sin embargo, la parte de envío desde los smartphones sí presenta un trabajo extra.

En cuanto a esta parte de envío de datos, se deberá establecer la conexión bluetooth con el PC y enviar mediante ella los datos procedentes de los sensores, para lo cual se debería desarrollar una aplicación específica, teniendo en cuenta los siguientes aspectos:

- **Comunicación bluetooth**: esta comunicación, para que se corresponda con la comunicación que acepta Matlab, debe ser establecerse como *Serial Port Profile* (SPP), lo que es lo mismo, crear un puerto COM como el que establece el Arduino. Para ello, se debe establecer la conexión bluetooth en la aplicación como servidor, mediante las siguientes funciones:

Implementación de la conexión serie del Arduino para la comunicación bluetooth

```
    btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
    //Donde MY_UUID
    private static final UUID MY_UUID = UUID
        .fromString("00001101-0000-1000-8000-00805F9B34FB");
//Donde la base UUID es "00000000-0000-1000-8000-00805F9B34FB" y
//0x1101 es el UUID de la clase servicio "SerialPort"
//También se establece el servidor:
    serverSocket = bluetoothAdapter.listenUsingRfcommWithServiceRecord(
                "android-app-name", MY_UUID);
```

- **Habilitar la aplicación como puerto COM en el PC**: Una vez se establece la conexión bluetooth con el PC y se pone en marcha la aplicación en el dispositivo móvil, debemos especificar al PC que esa conexión bluetooth establecida es un puerto COM, para que Matlab sea capaz de abrirla como tal. En Windows 10, esto se lleva a cabo de la siguiente manera:
    1. Acceder a la configuración bluetooth: **Configuración -> Dispositivos -> Más opciones de Bluetooth -> Puertos COM**

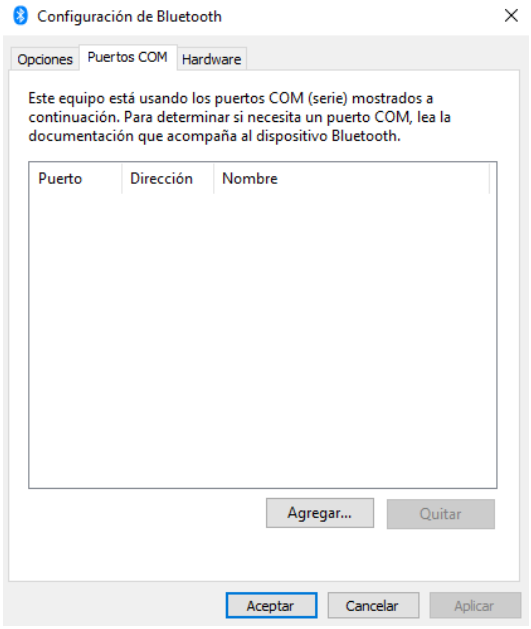


Figura 4.2.1.6- Ventana de configuración Bluetooth.

    2. Agregar aplicación android como puerto COM: desde la ventana anterior, pulsando en **Agregar**, aparece la ventana mostrada en la figura 4.2.1.7. En esta nueva ventana seleccionamos: **Dispositivo que usarán el puerto COM -> Android** y en **Servicio -> Nombre de nuestra aplicación**.

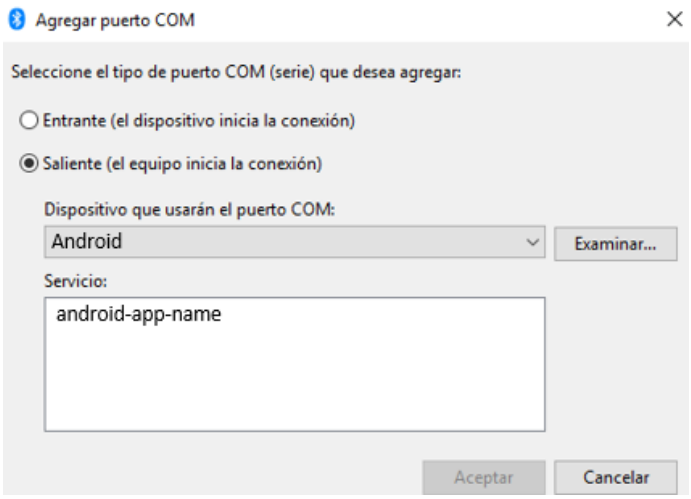


Figura 4.2.1.7- Ventana para agregar puerto COM.

Una vez se tenga la ventana tal como aparece en la figura 4.2.1.7, se selecciona la aplicación y se pulsa **Aceptar**. Con esto ya debería aparecer la aplicación conectada por bluetooth como puerto COM, tal y como aparece en la figura 4.2.1.8.

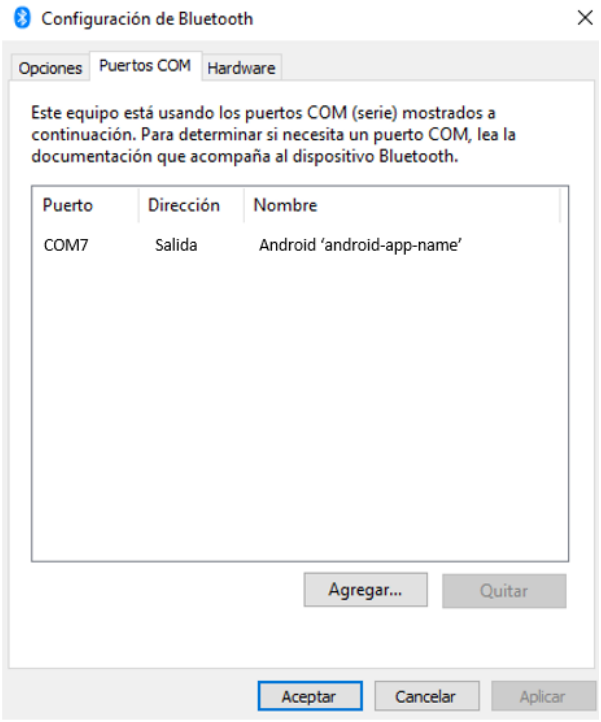


Figura 4.2.1.8- Puerto COM con la aplicación Android establecido.

Una vez creada la aplicación para enviar datos y establecida la conexión adecuada, el envío y recepción de datos es un tema ya resuelto en las aplicaciones desarrolladas, y sus respectivos códigos están disponibles en los anexos.

4.2.2-CONEXIÓN DE VARIOS DISPOSITIVOS A UN SMARTPHONE

Otra posible línea de ampliación es establecer la citada red de sensores usando un Smartphone como dispositivo maestro, en sustitución del PC con Matlab. Esta disposición aceptaría tanto Smartphones como Arduinos como sensores de entrada, puesto que estos envían sus registros de la misma forma se conecten a un dispositivo u a otro, el trabajo de esta

posible línea futura se centraría en el desarrollo de la aplicación Maestro que "escuche" a estos dispositivos esclavos y coordine la adquisición de sus respectivos registros. Esta nueva configuración se ajustaría a lo mostrado en la figura 4.2.2.1.

Figura 4.2.2.1 - Posibilidad de red de sensores conectando Smartphones o Arduinos a un Smartphone que actúe de maestro.

POSIBLE IMPLEMENTACIÓN

El trabajo de esta implementación se centraría en el desarrollo de la mencionada aplicación maestro, encargada de recibir y coordinar la entrada de datos.

Una posibilidad, y la única que se va a tratar en este trabajo, es la implementación mediante comunicación **bluetooth**. La implementación de esta comunicación sería análoga a lo visto en el apartado anterior de conexión con mediante bluetooth de aplicaciones Android con Matlab, debiendo crear un "servidor" bluetooth la aplicación maestra mediante las herramientas que se comentaron en dicha sección. Este "servidor" será al que se conecten el resto de aplicaciones mediante un *socket* de salida.

Una posibilidad sencilla para llevar a cabo esta implementación es adaptando la aplicación ejemplo disponible en la documentación online de Android para crear un **Chat Bluetooth**. En este ejemplo se realiza tanto la conexión como servidor para el maestro, que será la parte que "escucha", y las conexiones que envían los mensajes, de tal manera que sólo haría falta modificar la parte de envía para que en vez de enviar mensajes se envíen los registros del acelerómetro, y a partir de ahí programar la aplicación maestro para que registre los diferentes dispositivos a los que escucha, implementando en dicha aplicación lo que se quiera.

La otra posibilidad natural es estableciendo la conexión vía **wifi**, pero como no se ha trabajado ese aspecto en Android en el presente trabajo se deja en manos de los posibles desarrolladores de líneas futuras el explorar esta opción.

CAPÍTULO 5: COMPARATIVAS Y VERIFICACIONES

5.1-COMPARATIVA ENTRE SMARTPHONES

El mayor inconveniente que presenta el uso de smartphones se encuentra en la amplia variedad de dispositivos, cada uno con sus sensores específicos y con una gestión de dichos componentes diferente. Por lo dicho, los resultados de las aplicaciones desarrolladas y su ámbito de aplicabilidad vendrán determinados por las características del dispositivo sobre el que se monten, de tal manera que no podemos considerar los resultados y capacidades de la aplicación de forma genérica, si no que habría que analizarlas para cada dispositivo a utilizar.

Para ejemplificar este problema, se ha realizado una medición sobre la misma perturbación con dos smartphones diferentes: un LG G5 y un Hauwei Honor 7. Los sensores de cada dispositivo son los siguientes:

	LG G5	Honor 7
Sensor Vendor	BOSCH	STMicroelectronics
Sensor Name	LGE Accelerometer	LSM330 3-axis Accelerometer
Sensor Resolution	0.0023956299m/s^2; Max Range: 78.4532; Min Delay: 5000 µs	0.009576807m/sec^2; Max Range: 78.4532; Min Delay: 10000 µs

De entrada, ya se puede apreciar la diferencia en calidad de los distintos sensores: el dispositivo LG monta un sensor BOSCH específico, mientras que el Huawei monta un acelerómetro genérico, como el que se podría montar en un Arduino, con menores capacidades que el sensor BOSCH.

Realizando la prueba de medición sobre la misma perturbación a la frecuencia máxima del dispositivo, se observa lo siguiente:

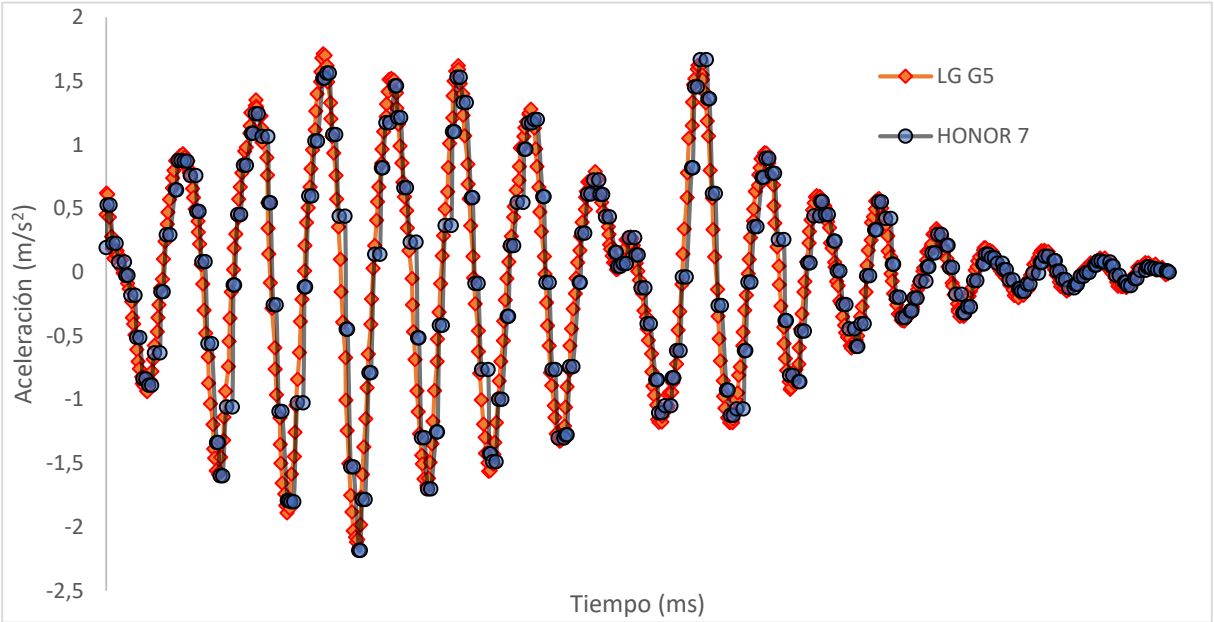


Figura 5.1.1- Resultados de las mediciones mediante la aplicación Smartphone con los dispositivos LG G5 y Honor 7. Arriba a la izquierda, con el Honor 7, a la derecha con el LG G5 y abajo los dos registros superpuestos.

A la vista de estos resultados, resulta evidente la disparidad en la calidad de los sensores. Mientras que el sensor del LG G5 muestrea a aproximadamente 200Hz, el sensor del Honor 7 lo hace a 100Hz, y además con más ruido. Esto último se ve aún mejor si se realiza una FFT sobre ambos muestreos:

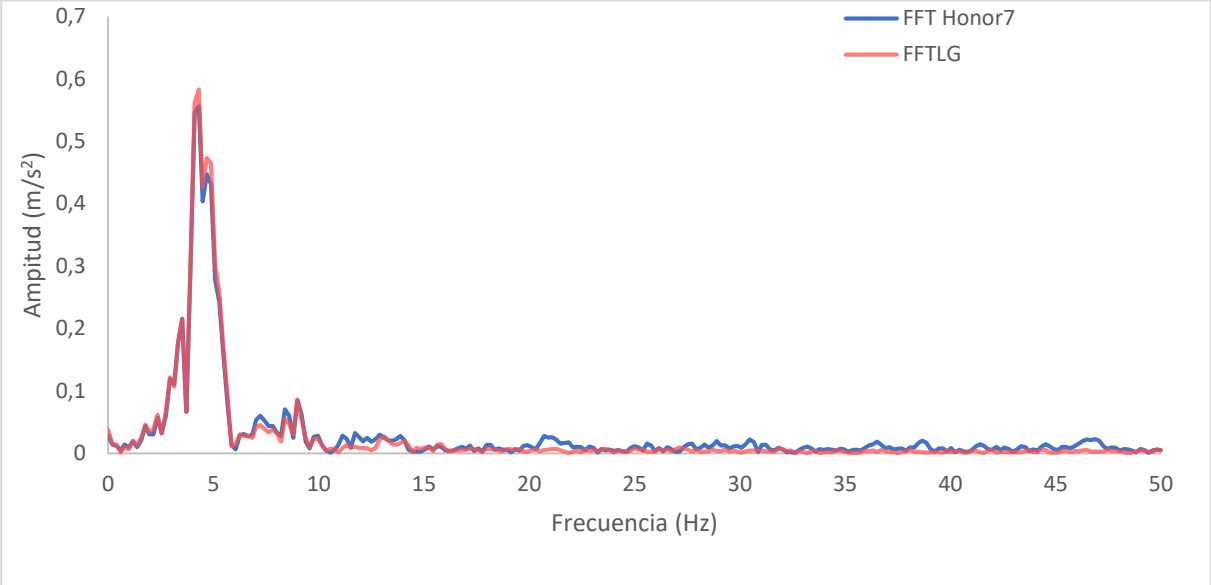


Figura 5.1.2- FFT aplicada sobre los registros mostrados en la figura 5.1.1.

En esta figura 5.1.2-, se ve más claramente la diferencia tanto en frecuencias de muestreo, que en el caso del LG G5 la operación permite analizar el doble de frecuencias, como la diferencia de ruido de cada registro, notablemente menor en el caso del LG G5. Estas medidas han sido tomadas mediante el registro sin gravedad, que como se explicó en el capítulo dedicado al desarrollo de la aplicación se lleva a cabo accediendo al sensor TYPE_LINEAR_ACCELERATION. Este elimina la gravedad y el offset de cada eje mediante el uso de uno o varios sensores (acelerómetro o acelerómetro y giróscopo) y software, con un tratamiento de datos no especificado.

Si se realiza la misma comprobación, pero con el otro sensor, TYPE_ACCELEROMETER, que no procesa el registro y ofrece la lectura directa, el resultado obtenido es el siguiente:

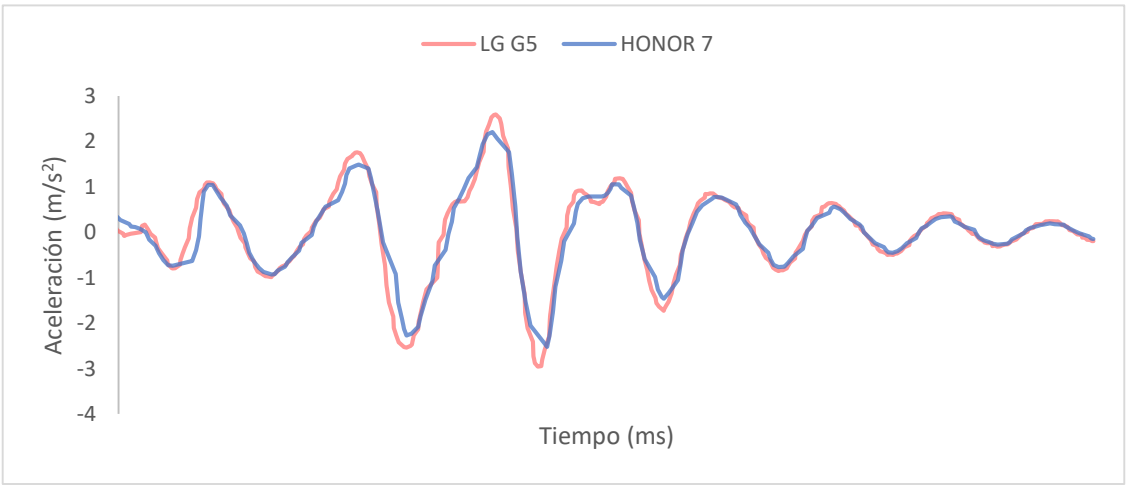


Figura 5.1.3- Superposición de los registros mostrados en la figura 5.1.1

Observamos que el caso del LG G5 es similar al anterior, mientras que el Honor 7 mejora notablemente, aunque la diferencia en las frecuencias de muestreo sigue siendo la misma en ambos casos, con el LG G5 muestreando al doble de velocidad.

Por otro lado, aplicando de nuevo la operación FFT sobre ambos registros se observa lo siguiente:

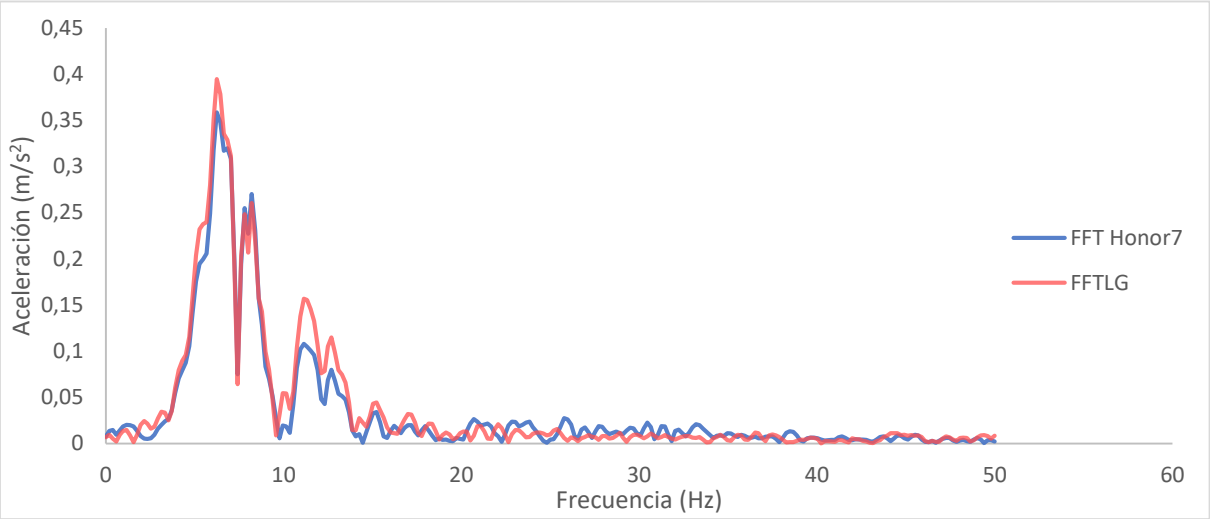


Figura 5.1.4- Aplicación da la operación FFT a los registros mostrados en la figura 5.1.1.

Al aplicar esta operación, se ve claramente como gran parte del ruido que aparecía en el caso del HONOR 7 ha desaparecido, mientras que en el caso del LG G5 parece haber aumentado, igualándose el ruido en ambos registros. Con esto se hace evidente la influencia no sólo de la capacidad del acelerómetro incluido, sino también de cómo gestiona y procesa dicho registro el propio smartphone.

Otro problema que surge de la diversidad de smartphone y de componentes electrónicos que estos montan es que, al disponer de recursos limitados a la hora de desarrollar las aplicaciones en cuestión, no se puede garantizar el funcionamiento de las aplicaciones desarrolladas en todo tipo de dispositivos, a pesar de que era este un objetivo inicial. Concretado en el caso de la aplicación desarrollada en Matlab, se ha comprobado que, mientras que su funcionamiento es correcto en el caso de utilizar como sensor el LG G5, el registro "*live*" presenta problemas si usamos el Honor 7, no funcionando en la mayoría de ocasiones. Esto ocurre, posiblemente, por la diferencia en la frecuencia de muestreo que es capaz de alcanzar cada dispositivo, que al haber utilizado como elemento de diseño y comprobación el más rápido de los dos, parece presentar problemas de compatibilidad al cambiar a un dispositivo de menor calidad.

Resumiendo, los principales problemas encontrados relativos al uso de smartphones a modo de sensor son los siguientes:

- La disparidad en los sensores incluidos en cada smartphone. Esto hace que el funcionamiento de las aplicaciones en diferentes dispositivos no sea previsible, viéndose fuertemente influenciada la capacidad de registro por el hardware.
- No conocer cómo gestiona y procesa cada smartphone los registros. Esto redunda en el problema anterior y hace del uso de el sensor TYPE_LINEAR_ACCELERATION, a priori útil para el desempeño de las tareas

objetivo de las aplicaciones, algo no recomendable si no se comprueba antes como afecta al registro.

- El disponer de solamente un smartphone durante el desarrollo de las aplicaciones ha limitado la capacidad de realizar pruebas de funcionamiento, de modo que no se puede garantizar el correcto funcionamiento de las aplicaciones en todos los dispositivos, en especial si se trata de dispositivos de menores capacidades al usado en el diseño y desarrollo, el LG G5.

5.2-COMPARATIVA ENTRE DISPOSICIONES DE ARDUINO

En lo referente al uso del Arduino, el mayor porblema que presenta el dispositvo utilizado es su baja frecuencia de muestreo, así como la mayor presencia de ruido. Este resultado no sorprende, puesto que el sensor no pasa de los siete euros y estamos comparandolos con los de los dispositivos móviles. Aún así, se han realizado algunas pruebas con el fin de identificar el o los factores que limitan la capacidad de muestreo de este hardware.

Una alternativa a la configuración propuesta es sustituir la comunicación bluetooth por comunicación mediante cable USB. Esta nueva configuración presenta un tiempo de muestreo medio de aproximadamente 27 ms, lo que es 10 ms más lento que con la comunicación bluetooth, seguramente por el tiempo que se toma la aplicación Matlab en acceder al arduino y extraer la información de los ejes. Lo que sí se observa es una aparente reducción de ruido en el registro, aunque puede deberse simplemente al menor tiempo de muestreo. El menor ruido puede obserbarse en la figura 5.2.1mostrada a continuación.

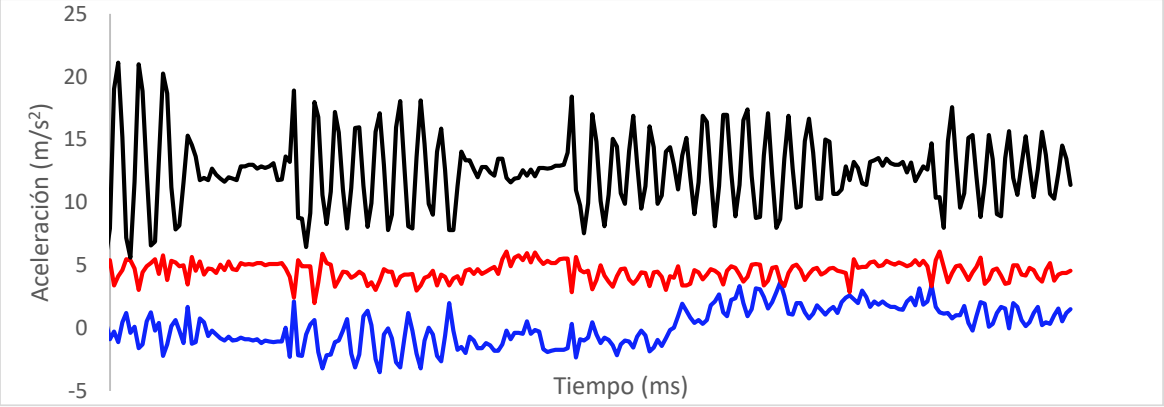


Figura 5.2.1- Muestreo procedente del Arduino mediante cable y sin aplicar media móvil.

Otra comprobación realizada ha sido la de registrar solamente un eje mediante la configuración orginal, con comunicación bluetooth. En este caso si se ha visto una reducción notable en el tiempo de muestreo, pasando de 17ms, mostrado en figura 5.2.2, a 8ms, mostrado en la figura 5.2.3, que es menos de la mitad. Esto parece indicar que el elevado tiempo de muestreo viene determinado principalmente por el tiempo que le lleva al arduino tomar un registro.

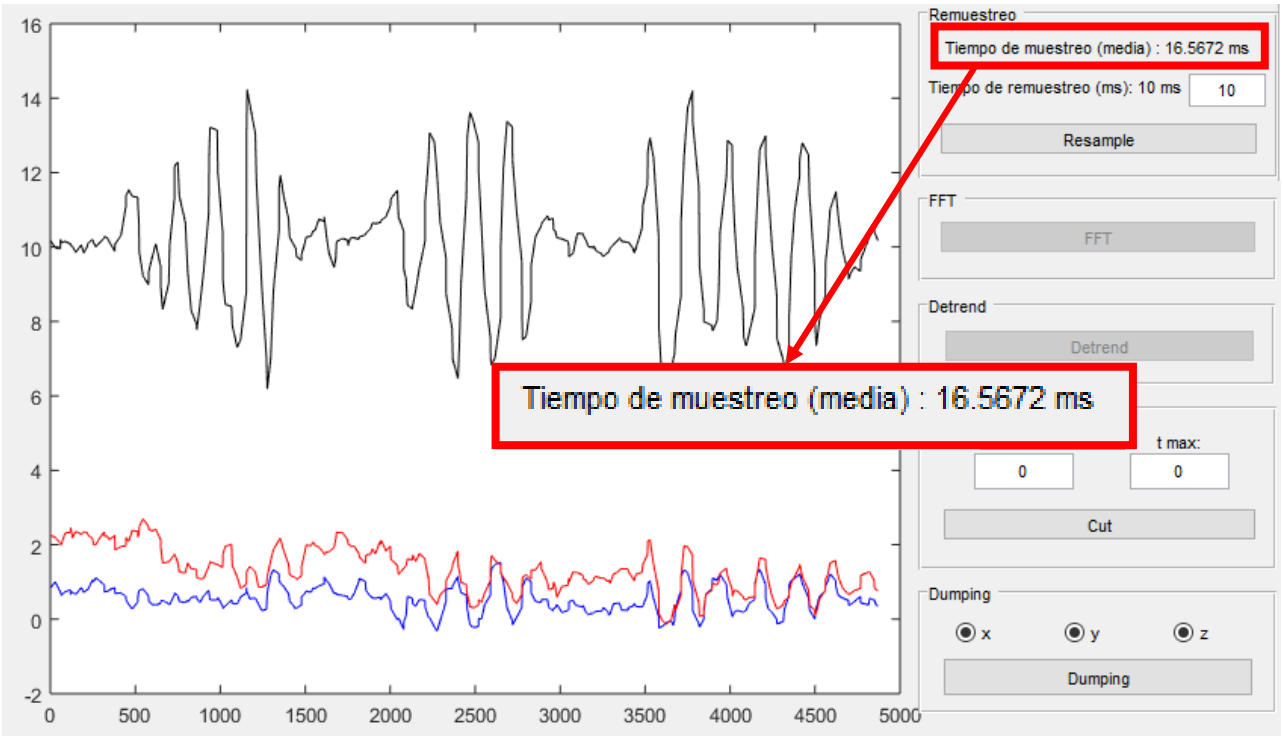


Figura 5.2.2- Registro de Arduino mediante comunicación bluetooth sobre los tres ejes. Muestra el tiempo de muestreo medio de aproximadamente 17 ms

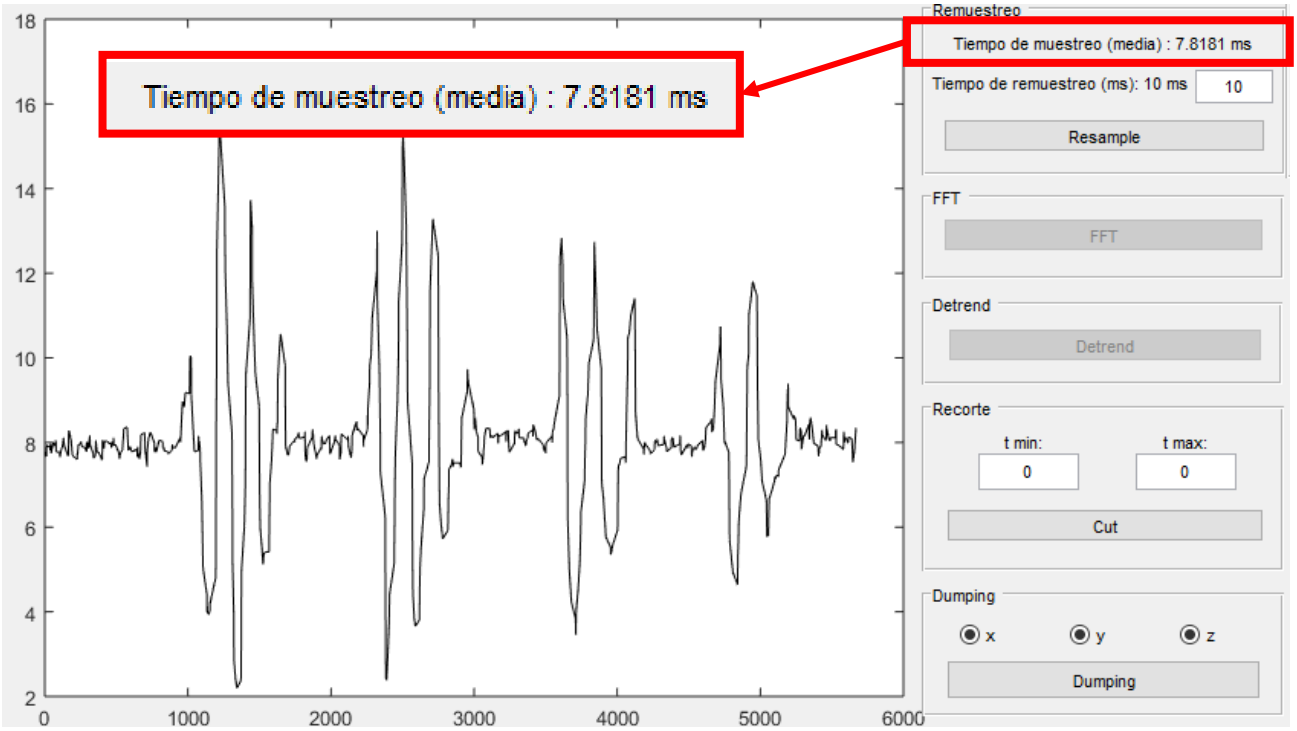


Figura 5.2.3- Registro de Arduino mediante comunicación bluetooth sobre el eje z. Muestra el tiempo de muestreo medio de aproximadamente 8 ms

5.3-VERIFICACIONES

Para terminar el capítulo y comprobar el correcto funcionamiento de las distintas disposiciones se va a comparar con un equipo profesional de adquisición de datos.

Para ello, se usará un acelerómetro piezoeléctrico de 100 mV/g de sensibilidad conectado a una tarjeta de adquisición de datos (modelo sirius STG) de 24 bits, 16 canales y frecuencias de muestreo síncrono de 20000 Hz.

El acelerómetro se dispone en el piso superior de una maqueta de edificio de 2 plantas disponible en el laboratorio, al igual que los sensores de las aplicaciones desarrolladas, tanto smartphone como arduino, tal y como se muestra en la figura 5.3.1.

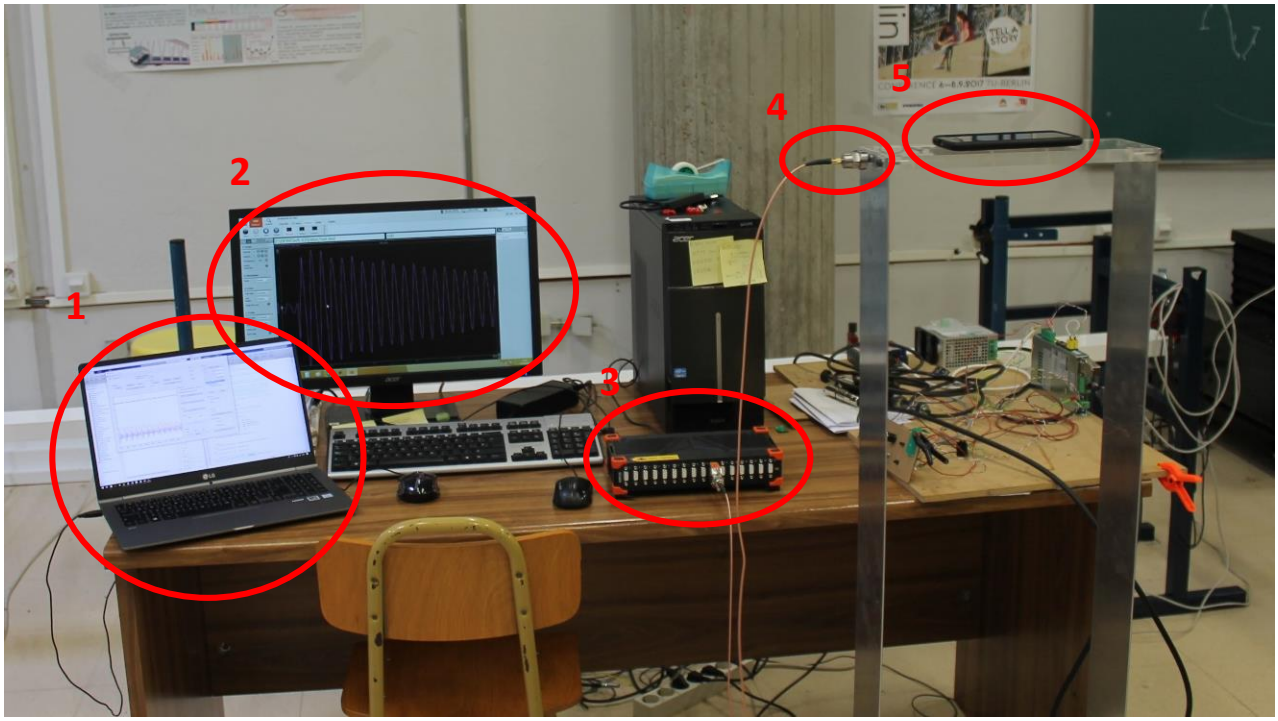


Figura 5.3.1- Disposición de los distintos elementos de medición sobre maqueta de edificio.

Con respecto a esta figura, los elementos ue se muestran en ella son:

1- Pc con la aplicación de registro de Matlab
2- Pc con el sofguar de adquisición de datos para la tarjeta Sirius
3- Tarjeta Sirius de adquisición de datos
4- Acelerómetro piezoeléctrico
5- Smartphone o Arduino

Para realizar esta comparativa se ha muestreado en paralelo la acción de una misma perturbación aplicada sobre la estructura del edificio. A continuación se expondrán los resultados en cada caso.

5.3.1- APLICACIÓN SMARTPHONE – EQUIPO DE ADQUISICIÓN DE DATOS

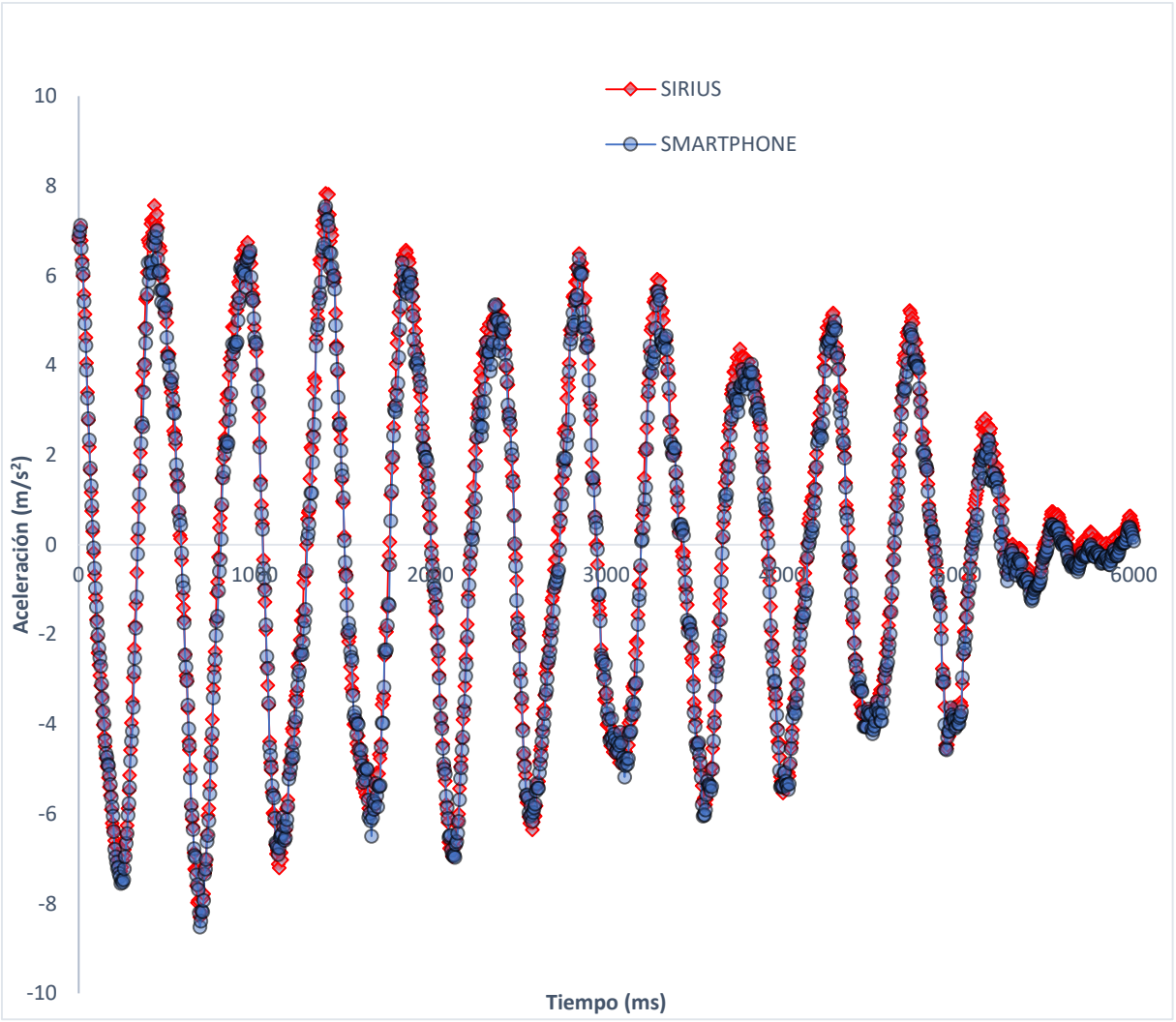


Figura 5.3.1.1- Superposición de los registros realizados con Smartphone frente a equipo profesional de adquisición de datos.

Como vemos en la figura 5.3.1.1 mostrada, los registros con ambos sistemas se superponen casi indistinguiblemente. Ambos registros se han efectuado a 200 Hz, y en el caso del registro realizado con el Smartphone no se ha aplicado ninguna operación de postproceso, con lo que se compara el registro en "bruto". A la vista de esta comparativa, se puede considerar que la capacidad de registrar con el Smartphone es muy buena en el rango de aplicación descrito, resultando en una herramienta de pre-análisis apta para llevar a cabo registros precisos.

A mayores, se puede comparar también la respuesta en frecuencia de los dos registros para terminar comparar también la capacidad de muestreo de la aplicación Smartphone en el dominio de la frecuencia.

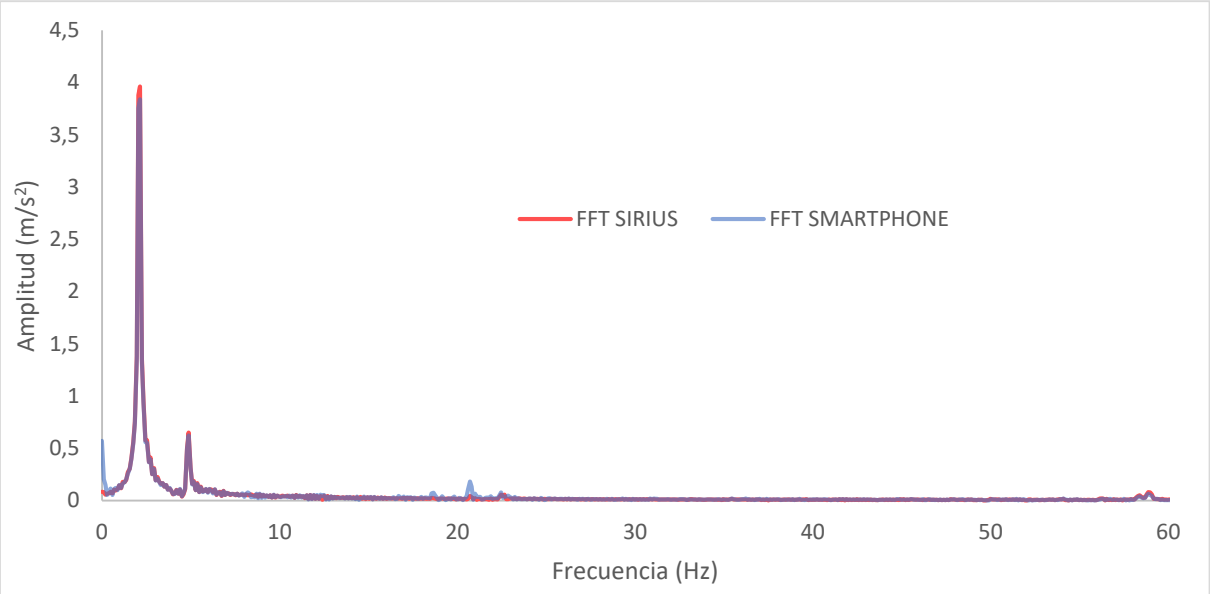


Figura 5.3.1.2- Superposición de las FFTs aplicadas sobre los registros mostrados en la figura 5.3.1.1.

Como se puede observar, la respuesta en frecuencias es también idéntica, salvo en el primer tramo donde se aprecia la presencia de la tendencia lineal presente en el registro del Smartphon, dado que no se a realizado un detrend en dicho registro.

En definitiva, a la vista de los resultados se puede considerar que la capacidad de registro mediante la aplicación smartphone a la frecuencia de 200 Hz o menores es buena.

Además, a partir de la validación de la capicidad de registro, podemos considera como buenas también las aptitudes de postprocesado de la aplicación, que mejoran aún más su capacidad de registro y uso. Algunas de estas capacidades se muestran en la figura 5.3.1.3:

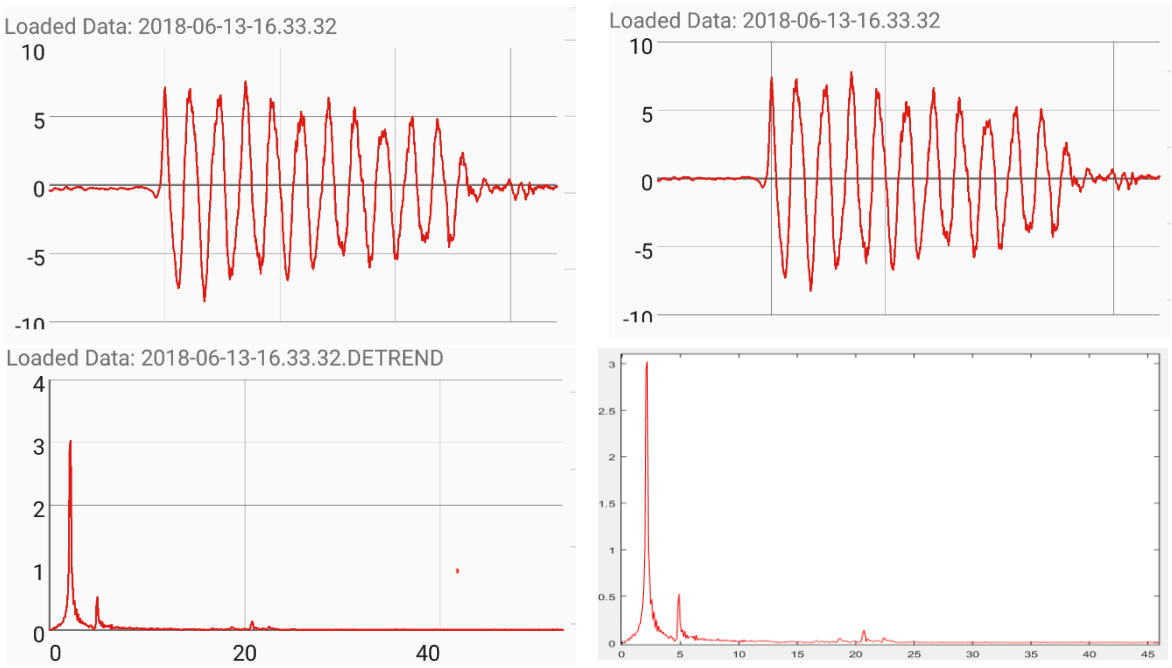


Figura 5.3.1.3- Postprocesado del registro en la aplicación Smartphone: arriba a la izquierda, registro original, a la derecha tras el detrend, abajo a la izquierda la FFT en el smartphone, a la derecha en Matlab.

5.3.2- ARDUINO – EQUIPO DE ADQUISICIÓN DE DATOS

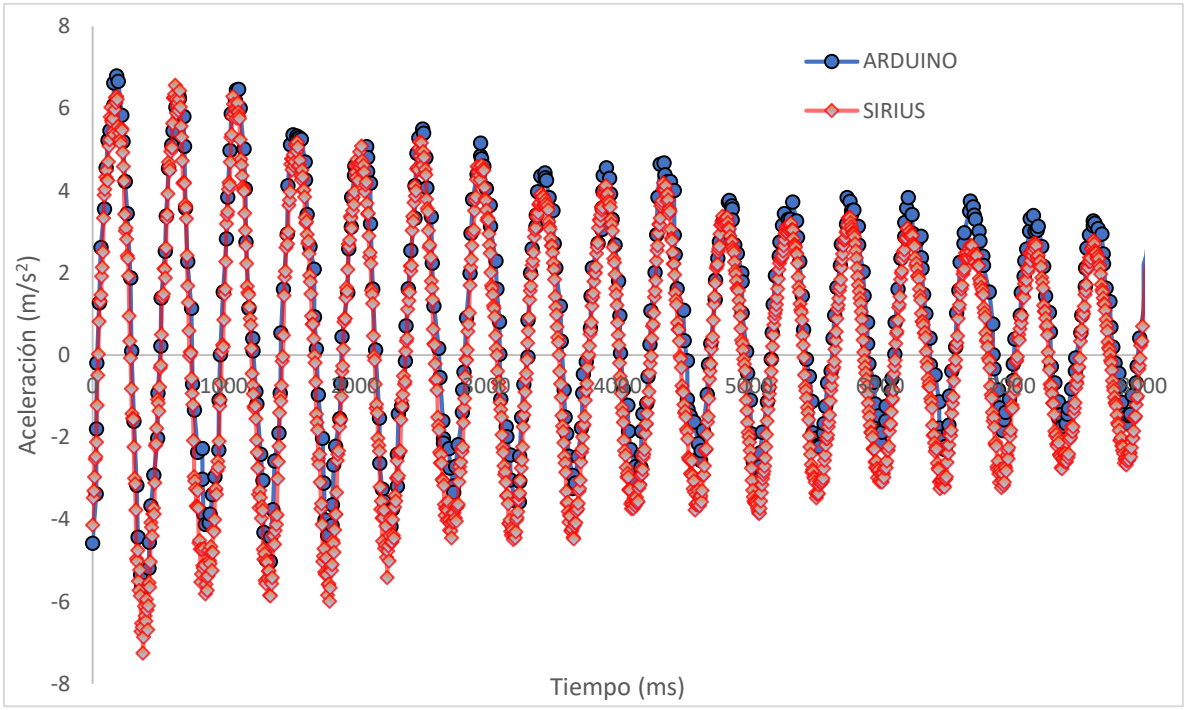


Figura 5.3.2.1- Superposición de los registros realizados con Smartphone frente a equipo profesional de adquisición de datos.

Como vemos en la figura 5.3.2.1 mostrada, los registros con ambos sistemas también se superponen. A diferencia del caso anterior, ahora ambos registros no se han efectuado a la misma frecuencia: con el equipo de adquisición de datos se ha hecho a 200 Hz mientras que el Arduino se ha hecho a su máxima capacidad, aproximadamente 17 ms por muestra, 60 Hz aproximadamente. La diferencia entre dichos registros por esto se aprecia mejor en la figura 5.3.2.2.

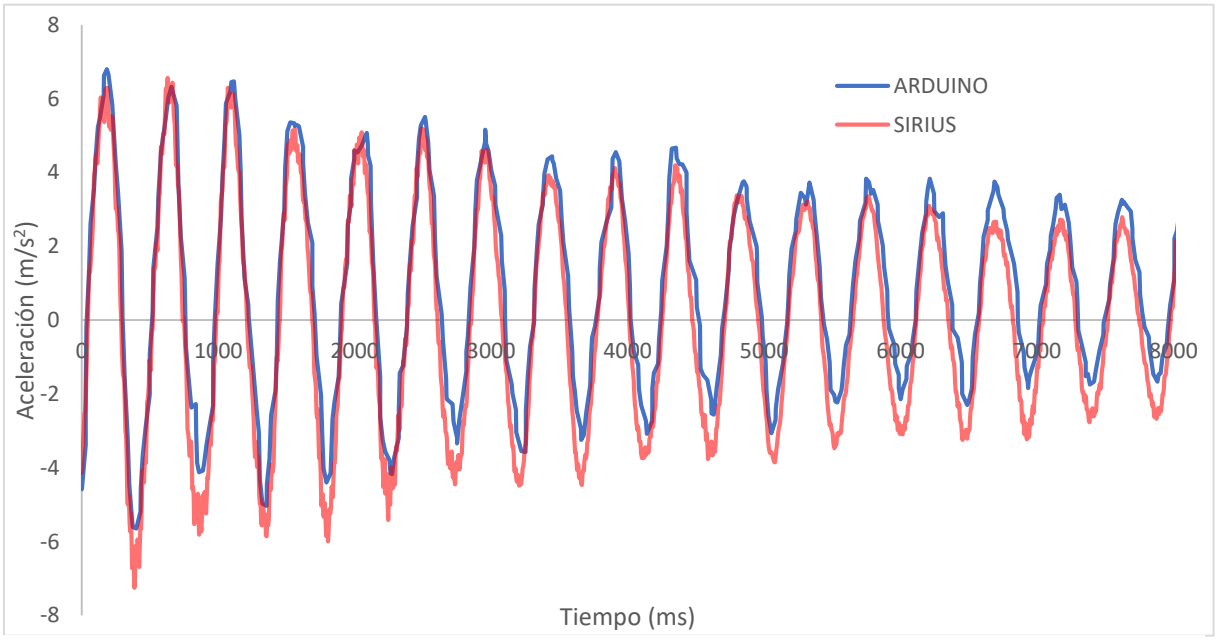


Figura 5.3.2.2- Superposición de los registros realizados con Smartphone frente a equipo profesional de adquisición de datos, sin puntos registrados remarcados.

Aún así, más allá de esta diferencia y la causada por no eliminar la tendencia lineal u offset que se produce en el Arduino, que como se ha comentado en apartados previos se deja para ser eliminada en el postproceso, los registros sí son análogos tanto en freceuncias como en amplitudes, lo que se puede ver más claramente en la FFT mostrada a continuación en la figura 5.3.2.3.

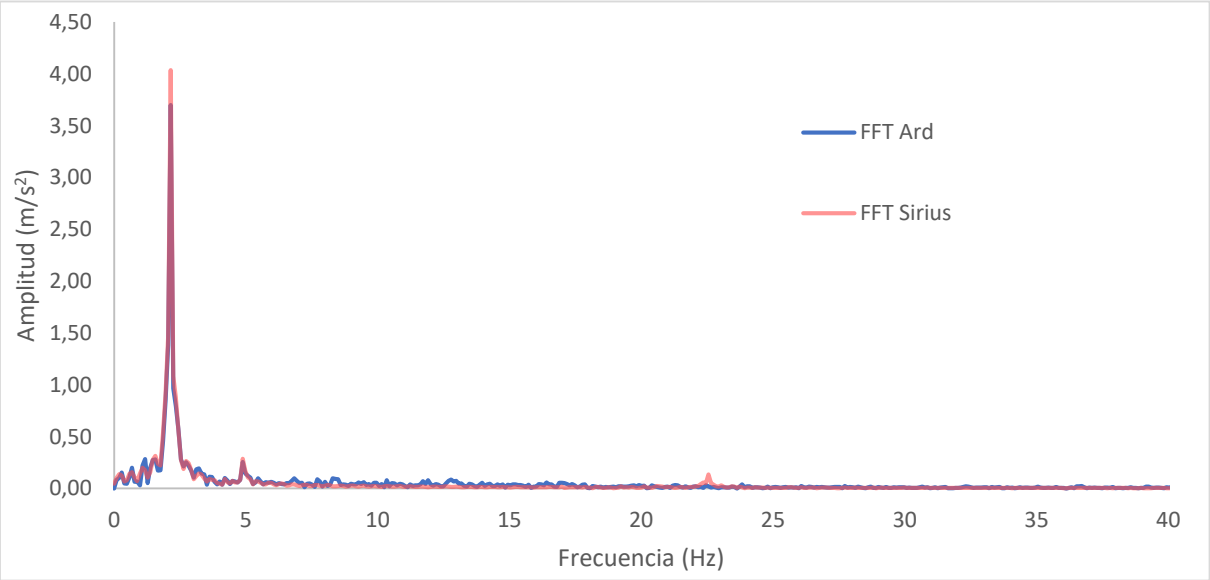


Figura 5.3.2.3- Superposición de las FFTs aplicadas sobre los registros mostrados en la figura 5.3.2.2.

A la vista de estos resultados, podemos considerar que la capacidad de registrar del Arduino también es aceptable, a pesar de las conocidas limitaciones en cuanto a su frecuencia de muestreo, que limitarían su ámbito de aplicación.

5.3.3- MATLAB + SMARTPHONE– EQUIPO DE ADQUISICIÓN DE DATOS

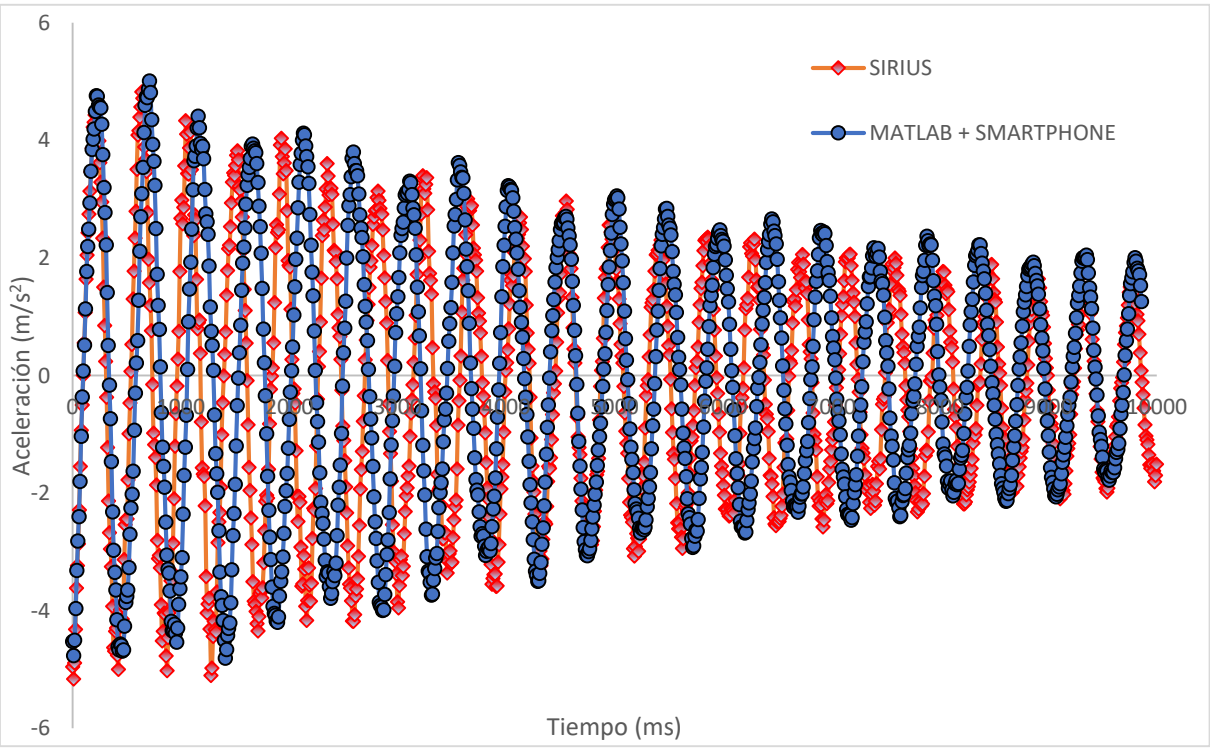


Figura 5.3.3.1- Superposición de los registros realizados con Matlab + Smartphone frente a equipo profesional de adquisición de datos.

A diferencia de los casos anteriores, con este sistema de adquisición de datos desarrollados vemos, en la figura 5.3.3.1, una discrepacia importante con respecto al profesional. En concreto, el registro con el sistema desarrollado se acelera en el tiempo con respecto al equipo profesional usado como referencia. Este efecto se aprecia mejor en la siguiente figura, figura 5.3.3.2, donde se ve claramente como la frecuencia fundamental es mayor en el caso del sistema basado en Matlab.

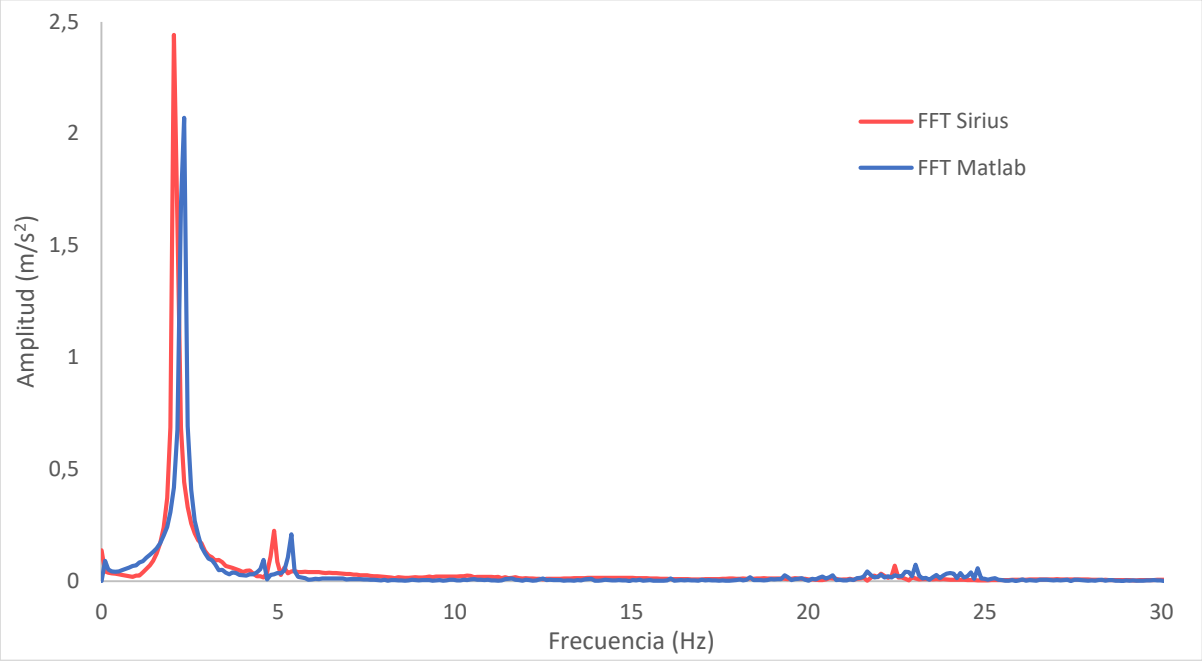


Figura 5.3.3.2- Superposición de las FFTs aplicadas sobre los registros mostrados en la figura 5.3.2.

En aras de solventar este problema, se ha identificado el fallo. Este se produce, a la vista de la comparativa realizada con la referencia del sistema profesional, en que el tiempo de muestreo arrojado por la aplicación de Matlab no se corresponde con el tiempo real en el que se produce el registro. Por desgracia, este es un error que se produce internamente en la aplicación dada por Matlab, funcionamiento interno al que no se tiene acceso y por tanto no se puede corregir.

Aún con lo dicho en el párrafo anterior, se va a tratar de ofrecer una solución a este problema asumiendo que el problema está solamente en el tiempo devuelto y calculado por la aplicación y no en el registro de la aceleración, que como se ha comprobado anteriormente los sensores del Smartphone sí tienen capacidad de llevar a cabo.

ALTERNATIVA AL USO DIRECTO DE LOS DATOS ARROJADOS POR LA APLICACIÓN MATLAB MOBILE

Las primeras consideraciones a tener en cuenta para plantear una solución, obtenidas a partir del análisis del funcionamiento de la aplicación tratando de identificar donde se produce el fallo, son las siguientes:

- Como se ha comentado, la aplicación no envía los registros efectuados de manera individual, si no que envía una matriz con los registros completos desde que se "ordena" el inicio del registro hasta que se "leen". Además, se ha comprobado mediante diversas pruebas que esto no se hace cada vez que se "apila" un nuevo registro, si no que se envía el registro del dispositivo Smartphone a la aplicación Matlab una vez se apilan un número determinado de nuevos registros.

- Considerando lo anterior, se ha visto que la actualización de la pila de registros se realiza cada 10 registros nuevos en el móvil, lo que sucede aproximadamente cada 100 milisegundos.

En base a estas consideraciones se va a modificar el algoritmo de adquisición de datos, de tal forma que se media el tiempo entre registros recibidos y se asocie a cada uno de estos registros un tiempo respecto a este periodo.

Para ver esto más claro, se comparan los dos algoritmos, el original y el nuevo con el tiempo modificado:

Modificación del algoritmo de registro de datos provenientes de la aplicación Matlab MOBILE

ALGORITMO ORIGINAL

```
if length(a)> l+1
   addpoints(hx,t(l+1:length(a)),...
         a(l+1:length(a),1))
   addpoints(hy,t(l+1:length(a)),...
         a(l+1:length(a),2))
   addpoints(hz,t(l+1:length(a)),...
         a(l+1:length(a),3))
   l=length(a);
end
```

ALGORITMO MODIFICADO

```
if length(a)> l
   t2=toc; % Tiempo final de los registros
% Construyo el vector con los tiempos entre
  envíos, equiespaciandolo con el n° de
  registroe entre envíos
   tm=linspace(t1,t2,1+(length(a)-l))';
% Alaceno los datos
   addpoints(hx,tm(2:end),...
      a(l+1:length(a),1))
   addpoints(hy,tm(2:end),...
      a(l+1:length(a),2))
   addpoints(hz,tm(2:end),...
      a(l+1:length(a),3))
% Declaro el tiempo inicial del
  siguiente registro
   t1=toc;
   l=length(a);
end
```

Como se ve, se modifica el tiempo de registro. Mientras que en el original se extraía directamente del tiempo dado por la aplicación, en el modificado primero se calcula a partir del tiempo entre registros recibidos y el número de estos y luego se asocia cada tiempo a su registro correspondiente registro.

Para implementar esta modificación se ha asumido una serie de cosas, como que los registros entrantes están equiespaciados o que el tiempo en que se realizan dichos registros se corresponde con el tiempo entre envíos. Como no podemos acceder al funcionamiento interno de la aplicación para verificar la corrección de estas consideraciones, se va a realizar otra comparativa con otro sistema de referencia para comprobar cómo se comporta esta solución.

La siguiente comparativa utiliza como sistema de referencia la aplicación Smartphone, puesto que su funcionamiento ya ha sido verificado y el montaje es más sencillo. El resultado es el que se muestra en la figura 5.3.3.3.

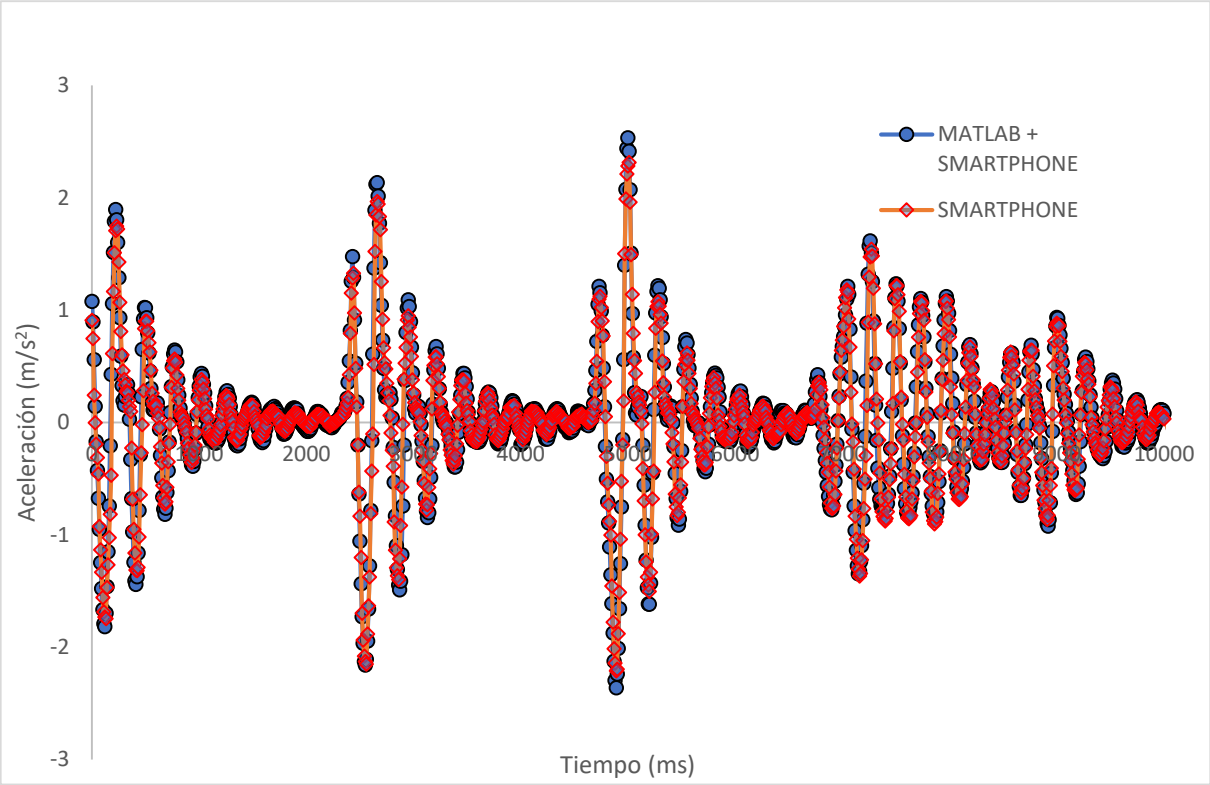


Figura 5.3.3.3- Superposición de los registros realizados con Matlab + Smartphone frente a aplicación Smartphone tras modificar el algoritmo de adquisición de datos, aplicando detrend a ambos registros para compararlos.

Como se muestra en la figura, ahora no hay desajuste en los tiempos de los registros y estos se solapan perfectamente. También se puede comprobar en la figura 5.3.3.4, donde vemos como se corresponden las frecuencias de ambos registros, aunque en esta ocasión hay mayor discrepancia entre los registros, no en frecuencia pero si en amplitud.

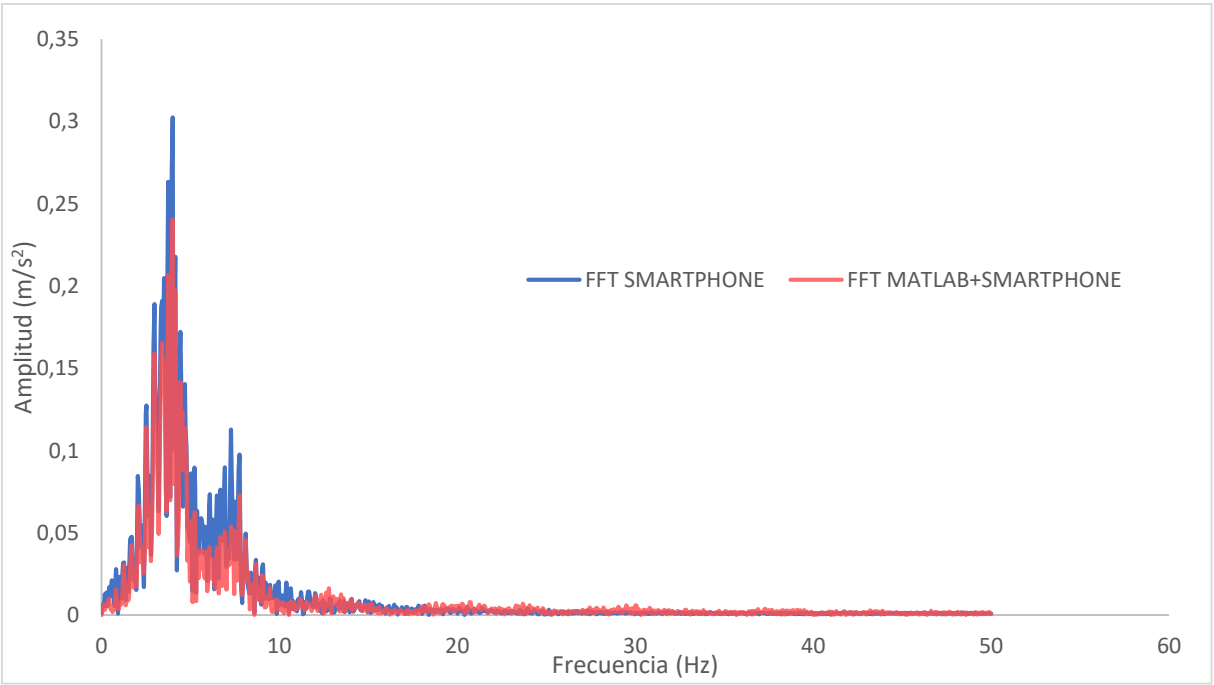


Figura 5.3.3.4- Superposición de las FFTs aplicadas sobre los registros mostrados en la figura 5.3.3.3.

A la vista de estas comparativas, se aprecia una mejora sustancial con respecto a la situación inicial, aunque sigue sin ser una solución perfecta, pudiéndose apreciar la diferencia entre los dos registros especialmente en el dominio de la frecuencia. Aún así, el resultado de la comparativa es suficientemente bueno como para considerar esta una herramienta de preanálisis válida.

CAPÍTULO 6: CONCLUSIONES

Tras los trabajos realizados se puede concluir:

- Se han conseguido los objetivos marcados tanto en el desarrollo de la aplicación autónoma para smartphone como en el entorno sobre MatLab. Concretamente:
  - Se ha habilitado el registro de aceleraciones a partir de los sensores del smartphone y de sensores externos gestionados por microcontroladores *low cost* (Arduino), solventando los problemas encontrados respecto al sistema cerrado de adquisición de datos provisto por la aplicación Matlab MOBILE
  - Se han establecido de manera satisfactoria los diferentes protocolos de comunicación para conectar los diferentes componentes del sistema de adquisición de datos. Estos protocolos de comunicación son:
    - Wifi: Smartphone – PC
    - Bluetooth: Smartphone-Arduino y PC- Arduino
  - Se ha implementado tanto el guardado en formato genérico para las distintas aplicaciones como el acceso, visualización y manipulación de dichos registros guardados
  - Se ha posibilitado el intercambio de archivos entre distintas plataformas, pudiendo estos ser postprocesados indistintamente
  - Se han implementado capacidades de postprocesado que permiten visualizar distintos indicadores y resultados útiles para el preanálisis dinámico de la tipología de estructura esbelta a tratar. Estas capacidades de postprocesado son las siguiente:
    - Picos
    - RMS o aceleraciones eficaces
    - MTVV o valor eficaz móvil de la aceleración ponderada
    - Contenido en frecuencias, a partir de la FFT
    - Filtrado de registros, tanto en el postproceso como en el registro
    - Eliminación de la tendencia lineal del registro
    - Remuestreo del registro
    - Amortiguamiento
  - Gracias a lo citado en el punto anterior, se ha dotado al conjunto de aplicaciones de las capacidades básicas de análisis que las permiten servir de herramientas de preanálisis dinámico para la tipología de estructura esbelta tratada en la introducción, gracias a que los resultados arrojados por ese postproceso se corresponden a los indicadores y parámetros relevantes a analizar, recogidos en la introducción y extraídos de las diferentes normas
  - Se ha verificado el correcto funcionamiento de las distintas configuraciones usando como referencia un equipo profesional de adquisición de datos

- Adicionalmente, se ha completado la formación recibida en el Máster con otras habilidades como: programación, tratamiento de señales, comportamiento dinámico de estructuras, tratamiento de señales, cumplimientos normativos, etc.

BIBLIOGRAFÍA

- D. E. Fomento, "Instrucción de Acero Estructural.", 2011
- D. E. Fomento, "IAP-11.",2011
- D. De Iowa, "Vibraciones al Cuerpo Entero," *Meta*, pp. 1–3
- E. Caetano, R. Barbosa, Á. Cunha, and F. Magalhães, "The Viana footbridge: construction and dynamic monitoring.," *Bridg. Eng.*, vol. 166, no. 4, pp. 273–290, 2013
- HiVoSS, "Human induced vibrations of steel structures-vibration design of floors (HiVoSS) : Background Document," no. January, pp. 1–20, 2008

WEBGRAFÍA

- "*ABC del acelerómetro*". Internet: https://www.5hertz.com/index.php?-route=tutoriales/tutorial&tutorial_id=2 , Jun. 28, 2018
- "*An efficient IIR filter library written in JAVA*", Internet: https://github.com/berndporr/iirj , Jun. 28, 2018
- "*Arduino UNO R3 ATmega328P CH340G*", Internet: https://www.makerlab-electronics.com/product/arduino-uno-r3-atmega328p-ch340g/ , Jun. 28, 2018
- "*Bluetooth*", Internet: https://developer.android.com/guide/topics/connectivity/bluetooth , Jun. 28, 2018
- "*Developers*", Internet: https://developer.android.com/ , Jun. 28, 2018
- "*FFT.java*", Internet: https://introcs.cs.princeton.edu/java/97data/FFT.java.html , Jun. 28, 2018
- "*Las redes BlueTooth*", Internet: https://www.prometec.net/bt-hc06/ , Jun. 28, 2018
- "*Learn, Share, Build*", Internet: https://stackoverflow.com/ , Jun. 28, 2018
- "*MMA7361LC 3-Axis Accelerometer ±1.5/6g with Voltage Regulator*", Internet: https://www.pololu.com/product/1251/specs / , Jun. 28, 2018
- "*Sensors Overview*", Internet: https://developer.android.com/guide/topics/sensors/sensors_overview , Jun. 28, 2018
- "*Smartphone OS*" Internet: https://www.idc.com/promo/smartphone-market-share/os, Jun. 28, 2018
- "*Using MATLAB with Other Programming Languages*", Internet: https://es.mathworks.com/solutions/matlab-and-other-programming-languages.html, Jun. 28, 2018
- Anne E. Trefethen, Vijay S. Menon , Chi-Chao Chang, Grzegorz J. Czajkowski, Chris Myers y Lloyd N. Trefethen. "*MultiMATLAB: MATLAB on Multiple Processors*". Internet: http://www.cs.cornell.edu/info/people/lnt/multimatlab.html , Jun. 28, 2018

- Binlin Wu, "*Run two Matlab functions simutaneously in parallel*". Internet: https://alenblog.wordpress.com/2011/04/21/run-two-matlab-functions-simutaneously-in-parallel/ , Jun. 28, 2018
- Jonas Gehring, "*Android Graph Library for creating zoomable and scrollable line and bar graphs*", Internet: https://github.com/jjoe64/GraphView, Jun. 28, 2018
- Mark Waldrep, "*Aliasing and Foldover: A Primer*." Internet: http://www.realhdaudio.com/?p=2983 , Jun. 28, 2018
- Steve Arar, "*An Introduction to the Fast Fourier Transform*" https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-fast-fourier-transform/ , Jun. 28, 2018
- Steve Hanly, "*Vibration Analysis: FFT, PSD, and Spectrogram Basics*". Internet: https://blog.mide.com/vibration-analysis-fft-psd-and-spectrogram , Jun. 28, 2018
- Steven W. Smith, "*The Fast Fourier Transform*". Internet: http://www.dspguide.com/ch12/2.htm , Jun. 28, 2018