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 UN SISTEMA DE MONITORIZADO ESTRUCTURAL BASADO EN DISPOSITIVOS DE BAJO COSTE

Autor: D. Pablo Pérez González
Tutor: D. Antolín Lorenzana Iban
Tutor: D. Antonio Foces Mediavilla

Valladolid, julio, 2020

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 UN SISTEMA DE MONITORIZADO ESTRUCTURAL BASADO EN DISPOSITIVOS DE BAJO COSTE

Autor: D. Pablo Pérez González
Tutor: D. Antolín Lorenzana Iban
Tutor: D. Antonio Foces Mediavilla

Valladolid, julio, 2020

Resumen

Hay gran interés en conocer el comportamiento dinámico de las estructuras con distintos factores como sus condiciones ambientales y su ocupación. También en apreciar los cambios que se producen en los sistemas estructurales debido a los cambios en los materiales y/o en sus propiedades geométricas, para poder realizar un mantenimiento y una puesta en servicio correctos. En el presente Trabajo de Fin de Máster se desarrolla un sistema de monitoreo de la salud estructural (SHM por sus siglas en inglés) basado en dispositivos de bajo coste. El sistema SHM consta de un microcontrolador, como sistema de adquisición de datos de los sensores, y de una Raspberry Pi que se encarga del procesado de los datos. Además se realiza una comparación de prestaciones entre un microcontrolador de Arduino y otro de Texas Instruments. El software desarrollado se realiza en lenguaje Python para la Raspberry Pi y con Arduino para los microcontroladores.

Palabras clave

SHM, Texas Instruments, Arduino, Raspberry Pi, bajo coste

Abstract

There is great interest in knowing the dynamic behavior of structures with different factors such as their environmental conditions and their occupation. Also, in appreciating the changes that occur in structural systems due to changes in materials and / or their geometric properties, in order to perform proper maintenance and commissioning. In this Final Master's Project, a structural health monitoring system (SHM) based on low-cost devices is developed. The SHM system consists of a microcontroller, as a sensor data acquisition system, and a Raspberry Pi that is responsible for data processing. In addition, a performance comparison is made between an Arduino microcontroller and another one from Texas Instruments. The software developed is made in Python language for the Raspberry Pi and with Arduino for the microcontrollers.

Key words

SHM, Texas Instruments, Arduino, Raspberry Pi, low cost

Índice

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS

1.1 Justificación

A la implementación de la caracterización del comportamiento y a la detección previa de daños en las estructuras de ingeniería, como puentes y edificios, se le conoce como monitoreo de salud estructural, SHM por sus siglas en inglés (Structural health monitoring). Su objetivo es intentar conocer los cambios en el material, en las propiedades geométricas, en las condiciones de contorno y en las respuestas dinámicas de las estructuras. Estos cambios pueden afectar negativamente al rendimiento del sistema estructural y pueden generar fallos catastróficos.

Hay un interés creciente en SHM y en su potencial asociado para generar importantes beneficios económicos y de seguridad. Con su uso se pueden realizar estructuras más esbeltas (ahorrando material) y evitar fallos catastróficos que pueden generar lesiones o incluso muertes.

El proceso SHM implica la observación de un sistema a lo largo del tiempo utilizando mediciones muestreadas periódicamente de una serie de sensores. Estos sensores pueden ser: acelerómetros, células de cargas, sensores de temperatura, sensores de humedad, etcétera, para determinar el estado del sistema y sus variaciones. El uso de estos sistemas puede facilitar las labores de mantenimiento. Con ellos es posible predecir fallos, que no se pueden ver a simple vista, pudiendo realizar los análisis de forma remota.

La acumulación de daños de fatiga o corrosión son factores de fallo a largo plazo. Mientras, hay otros eventos que pueden afectar en un corto periodo de tiempo, como los terremotos, las sobrecargas, los temporales o las explosiones. La inspección periódica de la infraestructura puede evitar grandes catástrofes. Un ejemplo es el colapso del puente Morandi en Génova que causó decenas de muertes el 14 de agosto de 2018, figura 1.

Figura 1. Puente de Morandi tras el derrumbe. [1]

Por último otra motivación para utilizar estos sistemas, a mayores de poder optimizar las estructuras y mejorar el mantenimiento, es poder realizar un seguimiento de la ejecución de la obra y detectar posibles desviaciones respecto del diseño proyectado. Estas desviaciones se podrían corregir antes de poner la estructura a disposición del usuario, mejorando con ello la satisfacción del cliente.

Los componentes de un sistema SHM son:

- Estructura.
- Sensores.
- Sistema de adquisición de datos.
- Mecanismo de transferencia.
- Procesado de datos.
- Interpretación de datos y diagnostico estructural.

Algunos ejemplos de sistemas SHM son:

- Puente Stonecutters es un puente atirantado de aproximadamente 1,6 kilómetros situado en Hong Kong, cuya construcción se finalizó en 2009. Incluye un sistema de monitoreo de la salud estructural de más de 1700 sensores. Se miden desplazamientos deformaciones, aceleraciones, temperaturas y condiciones del viento. Se puede ver toda la red sensorial en la figura 2.

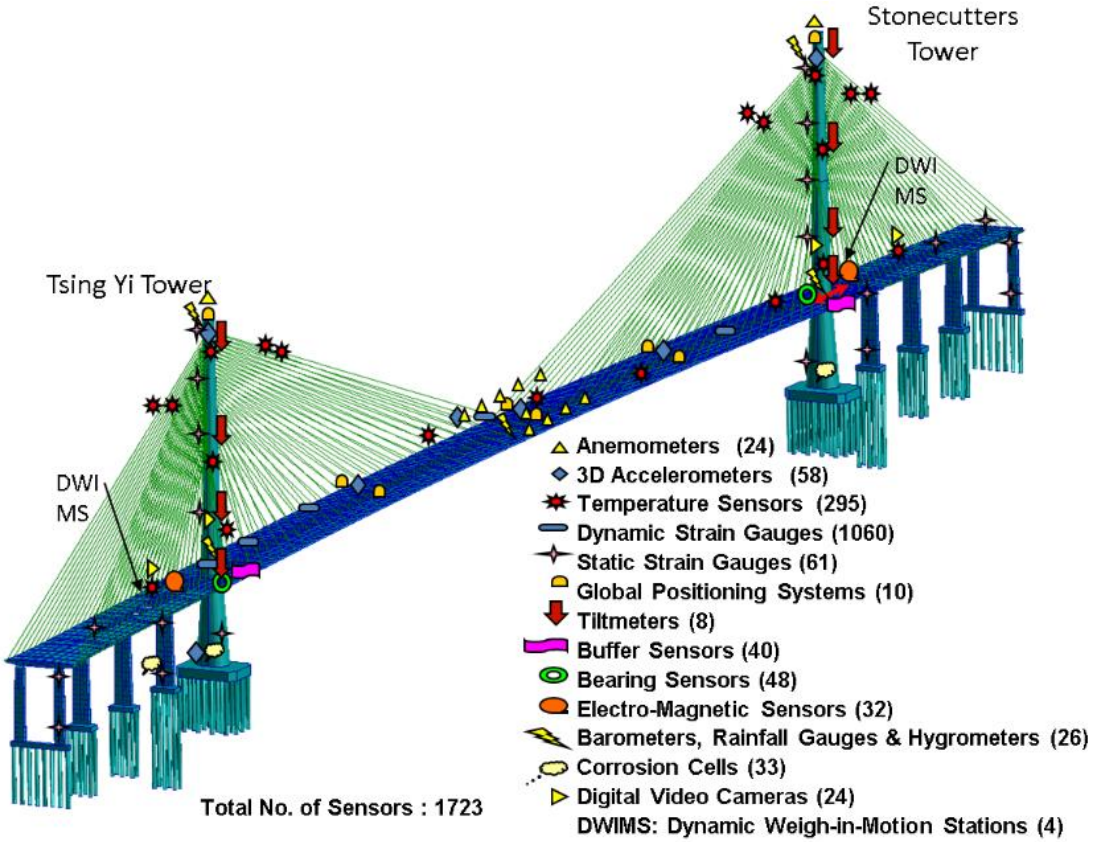

Figura 2. Puente Stonecutters. (Dascotte et al., 2013)

- El Lakhta Center, situado en San Petersburgo, es un complejo público y de negocios de 462 metros de altura, siendo el rascacielos más alto de Europa. Este edificio está equipado con un sistema SHM en sus cimientos, como se muestra en la figura 3. Este equipo dispone de más de 2800 sensores de diversos tipos, que monitorean la estructura en tiempo real.

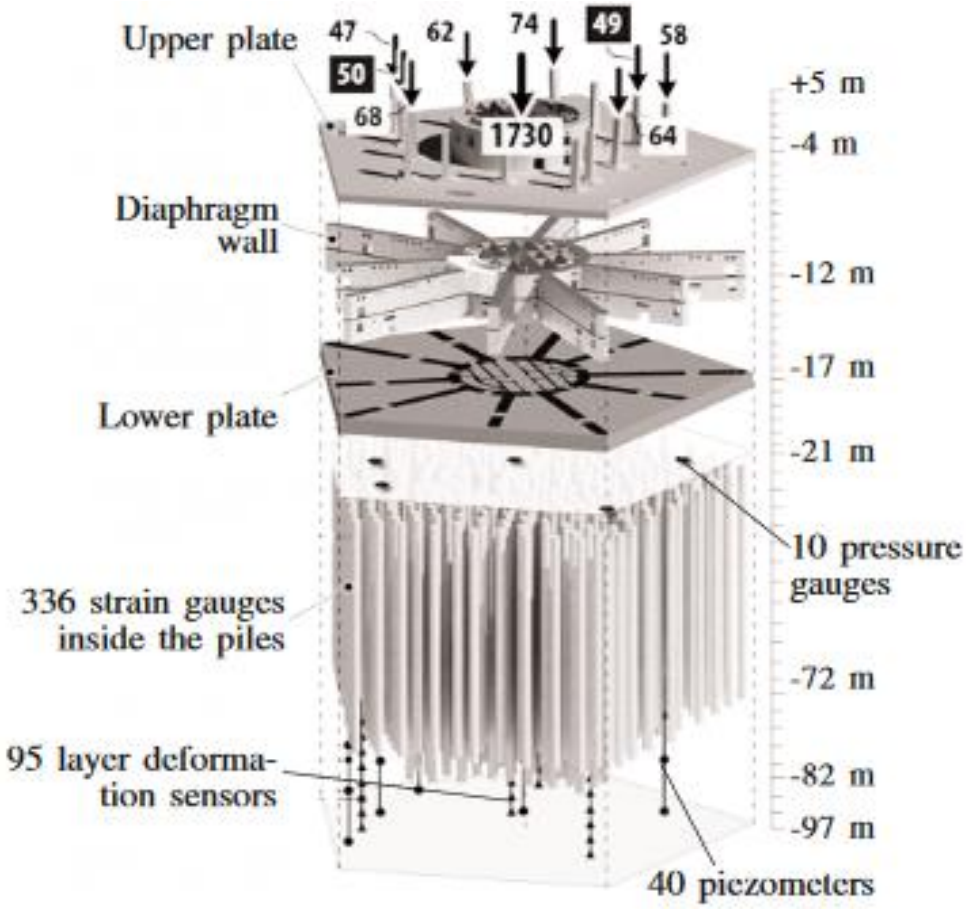

Figura 3. Lakhta Center. (Travush et al., 2019)

- El puente San Michele o puente Paderno situado en la región de Lombardía es un ejemplo de sistema SHM, que se incorpora a estructuras ya construidas. Este puente tiene más de 100 años, pues su construcción fue finalizada en 1889. En 2009, se equipó con 21 acelerómetros microelectromecánicos, 2 termopares, 7 módulos de adquisición Ethernet, 2 conmutadores Ethernet y 1 PC industrial. El esquema de la instalación aparece en la figura 4. Las aceleraciones se muestrean a 200Hz y se pueden visualizar los datos de manera online u offline.

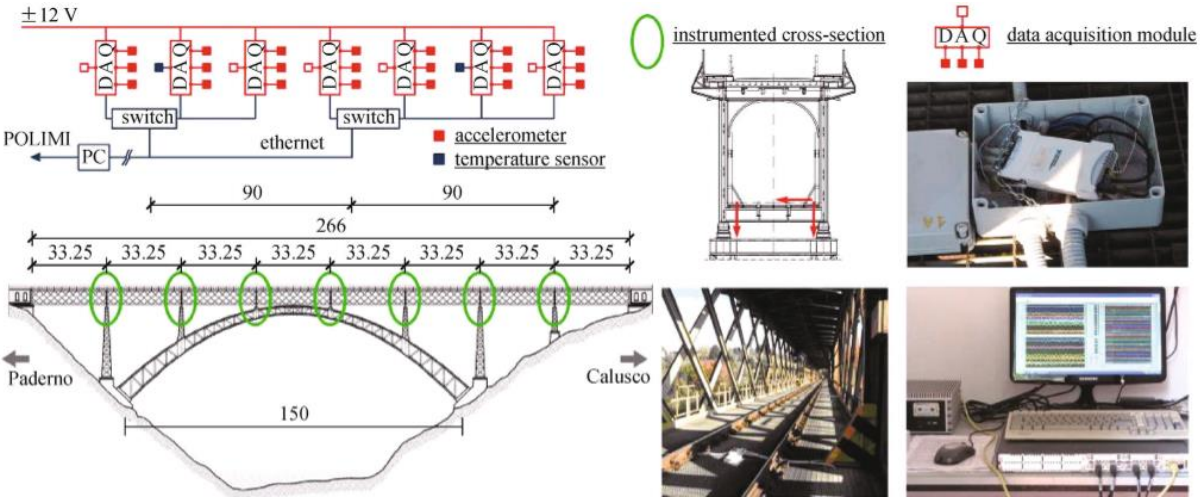

Figura 4. Puente San Michelle. (Gentile & Saisi, 2015)

- El puente de Xihoumen, es un puente colgante situado en china, posee un sistema de monitoreo de salud estructural orientado principalmente en analizar los efectos de los tifones. El sistema SHM, mostrado en la figura 5, caracteriza los parámetros ambientales, los parámetros del viento y la respuesta estructural inducidas en el puente. Estos datos se pueden utilizar para su mantenimiento, como referencia para futuros puentes de estructura similar y para prever como va a responder ante futuros tifones.

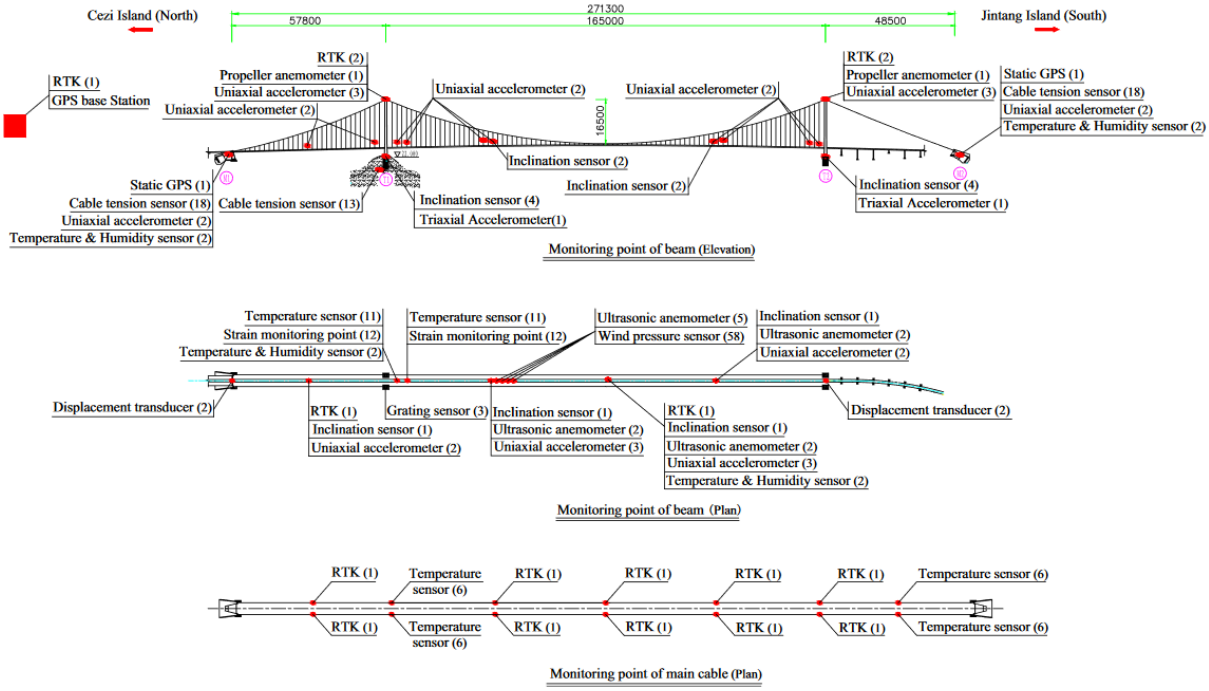

Figura 5. Puente Xihoumen. (Ye et al., 2019)

Los equipos actuales de monitorización son caros y se usan solo de forma puntual en el edificio. Hay una labor por intentan hacer equipos de adquisición de datos y de procesamiento de estos datos con materiales low cost, que permitan dejar siempre instalados los equipos en las estructuras y permitan detectar si se producen cambios en su comportamiento durante su vida útil pudiéndose predecir un fallo antes de que ocurra.

Una vez visto qué es y para qué sirve un sistema SHM, cuáles son sus componentes y cuál es su mayor inconveniente, surge este Trabajo de Fin de Máster (TFM), buscando la viabilidad de realizar estos sistemas con materiales de bajo coste, para que su coste no sea un obstáculo visto todos los beneficios que puede aportar.

1.2 Antecedentes

Previamente a este proyecto se han realizado diversos trabajos de fin de máster relacionados con dispositivos de bajo coste dentro del laboratorio de estructuras de la escuela de ingenieros industriales de la universidad de Valladolid. Pueden ser consultados en el enlace indicado.

- (Tuñón, 2016), "Sistema de monitorización de bajo coste para el registro de desplazamientos, fuerzas y aceleraciones en maquetas de estructuras". Realiza con una Raspberry Pi el monitoreo de parámetros en estructuras, como desplazamientos, fuerzas y aceleraciones.

- (Alario, 2018) "Desarrollo de una aplicación móvil para la preevaluación del comportamiento dinámico de estructuras esbeltas". Pretende realizar una herramienta para el registro del comportamiento dinámico de estructuras esbeltas basada en el uso de Smartphones y microcontroladores de bajo coste.

En el presente TFM se continúa con esta línea de trabajo, indagando con otros dispositivos comerciales y otros entornos de programación para identificar limitaciones y analizar su rendimiento.

1.3 Objetivos

Con este proyecto se pretenden alcanzar los siguientes objetivos:

- Sistema SHM de bajo coste, para estructuras con comportamiento dependiente de la temperatura y humedad. Se deben registrar, además de la temperatura y humedad, aceleraciones y fuerzas. El SHM se basa en un microcontrolador que actúa como sistema de adquisición de datos, que inalámbricamente transfiere datos a una Raspberry Pi. En este dispositivo se almacenan los datos y se realiza su post-procesado. El sistema también debe permitir exportar datos, para ser guardados para futuros análisis y compararlos con otros realizados en otras condiciones. En la figura 6, aparece en modo esquemático este sistema SHM.

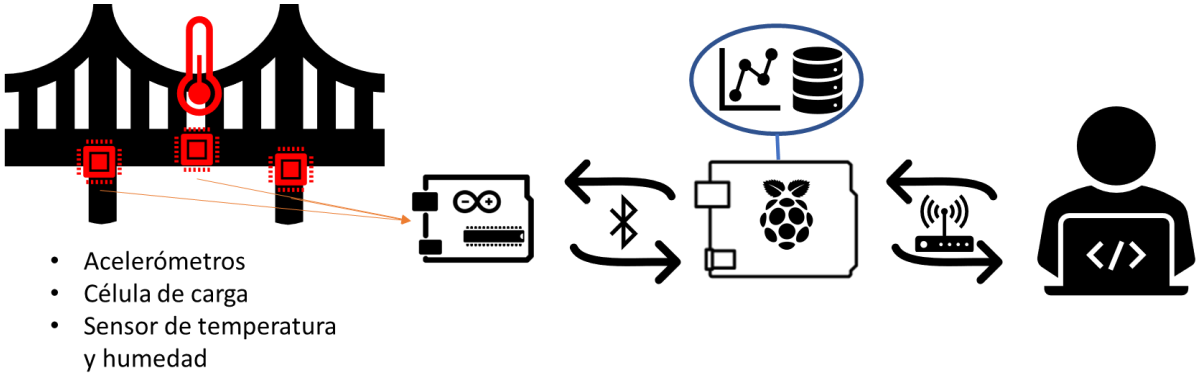

Figura 6. Esquema de sistema SHM.

- Comparar los microcontroladores de Arduino y Texas Instruments: características técnicas, facilidad de programación y uso, prestaciones conseguidas durante el funcionamiento del programa desarrollado y precio. Determinar la máxima capacidad de frecuencia de muestreo y canales de los microcontroladores empleados y cuáles son sus principales limitaciones.
- Documentar todo para futuros TFMs entroncables con las líneas de investigación del laboratorio.

1.4 Estructura de la memoria

La memoria del proyecto desarrollado se divide en cinco capítulos con la siguiente estructura:

- Capítulo 1. Introducción y objetivos. El capítulo actual donde se realiza una breve introducción del tema a desarrollar con la justificación del proyecto, se comprueba cuál es el estado del arte y se presentan los objetivos del proyecto
- Capítulo 2. Materiales. Se explican cuáles son los materiales seleccionados, cómo se realiza su montaje y configuración en el caso de que sea necesario, así como su

interacción con el resto de los elementos del proyecto. También se expone el estudio económico de los materiales.

- Capítulo 3. Software. Entornos de desarrollos utilizados, explicación de las funcionalidades de los programas y explicación de las herramientas y librerías utilizadas para el desarrollo de los programas en cada entorno.
- Capítulo 4. Análisis de resultados. Se realiza la comparación entre los microcontroladores, los resultados obtenidos del programa y el análisis del coste del desarrollo del sistema SHM.
- Capítulo 5. Conclusiones y trabajos futuros. Se expone las conclusiones obtenidas del trabajo desarrollado, así como unas líneas de trabajo futuras para complementar este estudio dentro del laboratorio, enfocándose en los puntos que se han quedado por desarrollar o viendo posibles mejoras que se pueden realizar en el sistema.
- Finalmente se expone la bibliografía consultada y en los anexos se recogen los códigos de los programas desarrollados y utilizados para la configuración de los dispositivos.

CAPÍTULO 2. MATERIALES

Para la realización de este sistema SHM se escogen materiales low cost, incluyendo un microcontrolador, un microprocesador y toda la instrumentación y elementos de comunicación necesarios para tener el sistema completamente funcional.

La jerarquía del sistema se muestra en la figura 7. Se basa en una Raspberry Pi como controlador principal, que se comunica en ambas direcciones con un microcontrolador y este a su vez, posee sensores que leen las magnitudes físicas. Se ha comprobado que con unas pequeñas modificaciones el sistema puede ser escalable, pues una Rapsberry Pi puede controlar varios microcontroladores. Para que los datos sean síncronos y se puedan procesar correctamente sin errores, debe ser el mismo microcontrolador con el mismo programa cargado.

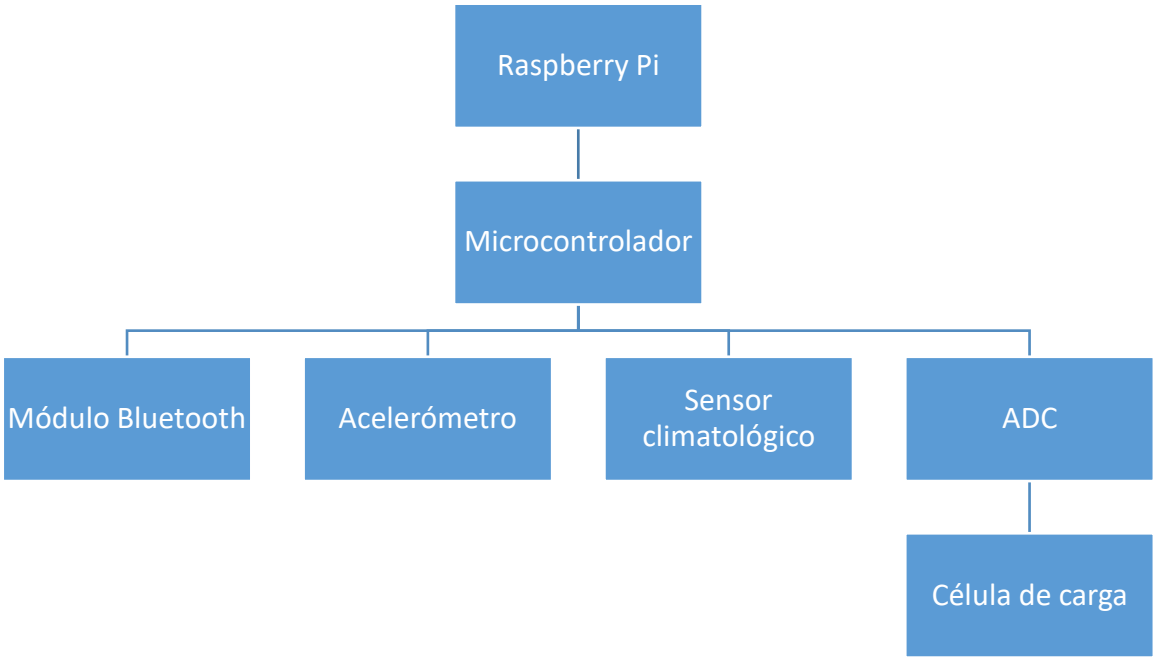

Figura 7. Jerarquía del sistema.

Por lo tanto los componentes de este sistema SHM son:

- Estructura
- Sensores → Acelerómetro, célula de carga y sensor de temperatura y humedad.
- Sistema de adquisición de datos → Microcontrolador
- Mecanismo de transferencia→ Comunicación por bluetooth.
- Procesado de datos → Raspberry Pi.
- Interpretación de datos y diagnóstico estructural→ Usuario

2.1 Raspberry PI

Como elemento principal donde se almacenan y procesan los datos, enfocados con la filosofía de bajo coste del proyecto se selecciona una Raspberry Pi Model 3 B+, como la que aparece en la figura 8. Se trata de un pequeño ordenador cuya memoria interna es una tarjeta microSD, con conexiones wifi y bluetooth, un conector HDMI, conector LAN y conector USB. Las especificaciones de esta Raspberry Pi se muestran en la tabla 1. Se ha adquirido también una caja para protegerlo principalmente de los golpes y también se le

han añadido disipadores en los microprocesadores para una correcta disipación del calor, ya que no vienen incluido en la tarjeta de fábrica.

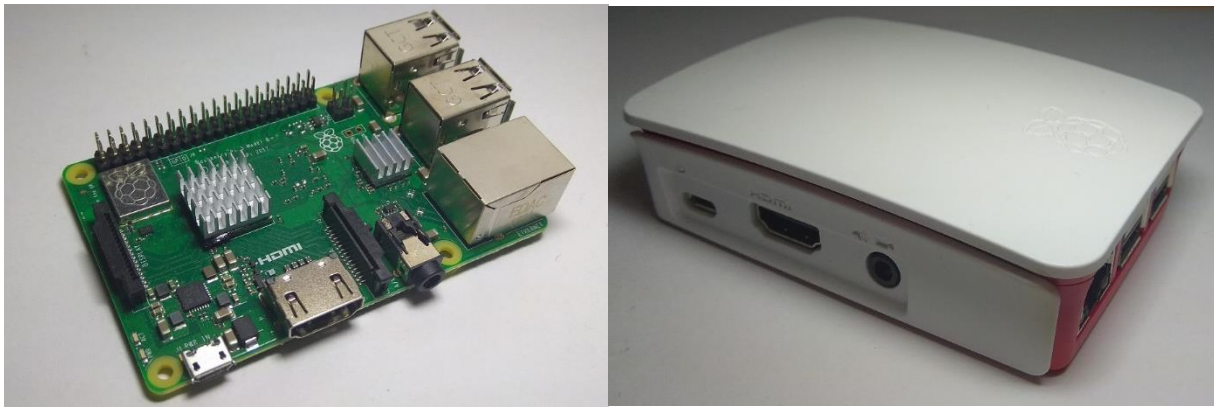

Figura 8. Raspberry Pi sin carcasa y con carcasa protectora.

Tabla 1. Especificaciones Raspberry Pi. (Raspberry Pi Foundation, 2016)

Atributo	Valor
Microcontrolador	Broadcom BCM2837B0, Cortex-A53
Bits microcontrolador	64 bits
Velocidad de reloj	1.4 GHz
RAM	1 GB SDRAM
Temperatura funcionamiento	0-50ºC
Conectividad	2,4GHz y 5GHz Wireless LAN, bluetooth 4.2, BLE
	Gigabit Ethernet (300Mbps)
	4 puertos USB
Video y sonido	1 HDMI
	Salida estéreo de 4 polos y puerto de video compuesto
	Conector de pantalla DSI y conector de cámara CSI
Multimedia	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
Memoria ROM	Micro SD
Alimentación	5V/2,5 A, a través de micro USB
	5V a través de conector GPIO
	Power over Ethernet
Tamaño	85x56x17 mm
Peso	67,3g
Precio	86,99 € (Todo incluido, alimentación, tarjeta SD, caja, disipadores). Solo placa: 39,78€

Para su correcto funcionamiento se utiliza un sistema operativo basado en Linux llamado Raspberry Pi OS, antiguamente Raspbian, que está optimizado para los recursos de este dispositivo, ya que, su potencia es limitada. Además, es un sistema operativo open source, debido a lo cual es gratuito, eliminando un gasto.

En el proyecto funciona con un software desarrollado en el lenguaje Python que permite comunicarse con el microcontrolador a través de bluetooth, siendo este programa la interfaz con el usuario y este dispositivo almacena todos los datos recibidos y los procesa en función de las peticiones del usuario.

2.2 Microcontroladores

En este sistema el microcontrolador es el encargado de la recogida de los datos y su envío a la Raspberry Pi para el procesado y almacenado de los datos. En este proyecto se han tenido en cuenta solo microcontroladores low cost para realizar esta tarea, siendo Arduino y Texas Instruments, las marcas seleccionadas para compararlas.

2.2.1 Arduino DUE

Arduino tiene una amplia gama de microcontroladores y placas de desarrollo de hardware, que venden bajo Licencia Publica General de GNU (GPL) y la Licencia Pública General Reducida de GNU (LGPL), permitiendo que cualquier empresa pueda distribuirlas. También sigue esta política de software libre para su plataforma de desarrollo. En este caso se ha utilizado, un Arduino DUE, propio de la empresa Arduino, como se aprecia en la figura 9.

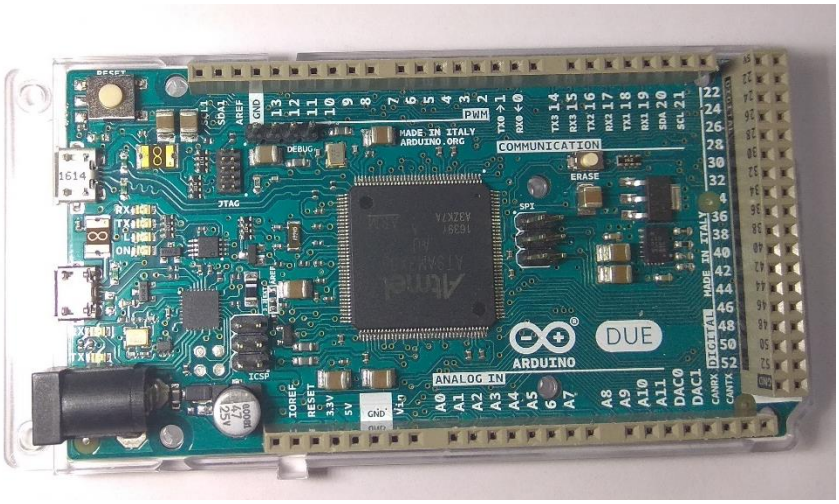

Figura 9. Arduino DUE.

En la tabla 2, se pueden apreciar las especificaciones del modelo seleccionado.

Tabla 2. Especificaciones Arduino DUE. (Arduino, 2020)

Atributo	Valor
Microcontrolador	AT91SAM3X8E
Bits microcontrolador	32 bits
Voltaje de funcionamiento	3,3V
Voltaje de alimentación	7-12V

Pin digitales I/O	54 (de las cuales 12 salidas PWM)
Entradas analógicas	12 (12bits)
Salidas analógicas	2 (DAC) (12bit)
Intensidad total de todos los pines I/O	130 mA
Intensidad a 3.3V Pin	800 mA
Intensidad a 5V Pin	800 mA
Memoria flash	512 KB toda disponible para el usuario
SRAM	96 KB (2: 64KB and 32KB)
Velocidad de reloj	84 MHz
Tamaño	101.52x53.3 mm
Peso	36 g
Precio	35 €

Se ha seleccionado el modelo DUE, porque es la placa de desarrollo más potentes de toda la gama de Arduino. Comparando unos atributos básicos del DUE con los otros modelos más vendidos como el Uno y el Mega, se puede ver claramente la diferencia de recursos, entre placas y el porqué de la selección de esta placa.

Tabla 3. Comparativa de placas de Arduino.

Atributo	Uno	Mega	Due
Bit procesador	8 bits	8 bits	32 bits
Memoria flash	32 KB (8 KB usados por bootloader)	256 KB (8 KB usados por bootloader	512 KB toda disponible para el usuario
SRAM	2 KB	8 KB	96 KB (2: 64KB and 32KB)
Velocidad de reloj	16 Mhz	16 Mhz	84 MHz

En la tabla 3 se aprecian las características que más afectan para el proyecto. La memoria flash del Arduino Uno, que permite guardar el programa desarrollado, es muy pequeña impidiendo poder realizar todas las funcionalidades que se quieren en el proyecto. Las dos características que más afectan para el desarrollo del programa que se quiere realizar son la SRAM, poder almacenar variables mientras se leen de los sensores para mandarlos posteriormente; y la velocidad de reloj, que permite la lectura de datos a mayor frecuencia y el envío de datos más rápido.

2.2.2 Texas Instruments TM4C1294XL

Texas Instruments es una de las empresas más importantes de electrónica. Dentro sus distintas gamas de negocio poseen una amplia gama de tarjetas de desarrollo, de las cuales se han seleccionado tres modelos distintos para comparar sus características y finalmente se ha elegido una para realizar el proyecto.

Para seleccionar la tarjeta de TI más adecuada se tiene en cuenta: precio, velocidad de reloj, bits del convertidor analógico digital, si incorpora o no Bluetooth, memoria Flash para el programa, memoria RAM para las variables volátiles y el entorno de desarrollo integrado (IDE). Los modelos elegidos para esta comparación son: SimpleLink CC2640R2F, C2000 Delfino F28379D, EK-TM4C1294XL o TivaC. En la tabla 4 se muestran los datos comparados.

Tabla 4. Comparación de tarjetas de desarrollo de Texas Instruments.

Atributo	SimpleLink CC2640R2F	C2000 Delfino F28379D	EK- TM4C1294XL
Precio (€)	31,24	33,76	23,39
Velocidad de reloj (MHz)	48	200	120
ADC (bits)	12	12/16	12
ADCS (Hz)	200 KHz	3,5/1,1MHz	2 MHz
Bluetooth	Si	No	No
Memoria Flash (Kb)	128	1024	1024
Memoria RAM(Kb)	28	204	256
IDE	Code Composer Studio	Code Composer Studio	Code Composer Studio y Energia

Simplelink posee peores prestaciones en todo y tiene la única ventaja del Bluetooth incluido. La C2000 tiene mayor velocidad de reloj e incluye el ADC de 16 bits, por lo que se puede eliminar el ADC externo ADS1115, pero sus desventajas son el mayor precio y que no se puede desarrollar en Energia IDE, entorno de desarrollo similar al usado por Arduino. Actualmente y no se puede hacer una comparación directa con Arduino, siendo necesario una programación propia para este modelo. El modelo elegido ha sido una tarjeta EK TM4C1294XL, como la que se muestra en la figura 10, debido a que la se puede comparar directamente con el Arduino DUE, cargando el mismo programa, debido a que se puede programar con Energia IDE, que utiliza la misma programación que Arduino.

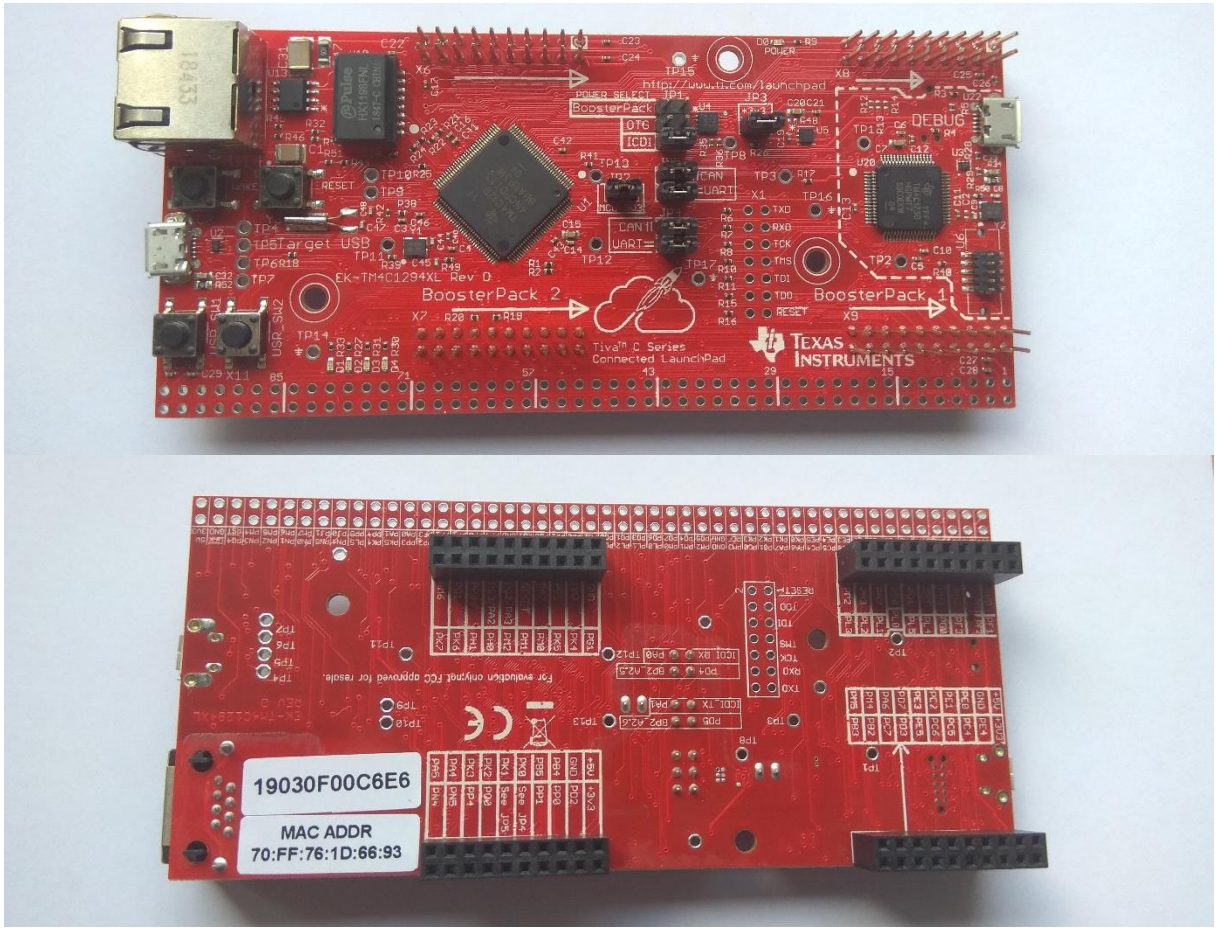

Figura 10. TM4C1294XL

Se muestran las especificaciones del modelo TM4C1294XL en la tabla 5.

Tabla 5. Especificaciones TM4C1294XL.

Atributo	Valor
Microcontrolador	ARM Cortex-M4F
Bits microcontrolador	32 bit
Voltaje de funcionamiento	5V
Voltaje de alimentación	4.75-5.35 V
Pin digitales I/O	90 (de las cuales 8 salidas PWM)
Entradas analógicas	20 (12 bits)
Salidas analógicas	0
Intensidad total de todos los pines I/O	400 mA
Intensidad a 3.3V Pin	1 A
Intensidad a 5V Pin	1 A
Memoria flash	1024 KB
SRAM	256 KB
Velocidad de reloj	120 MHz

Tamaño	124.5x56x10.8 mm
Peso	162 g
Precio	23,39 €

2.3 Resto de elementos

A continuación, se ven los sensores y el módulo de comunicación que se utilizan en este proyecto.

2.3.1 Célula de carga

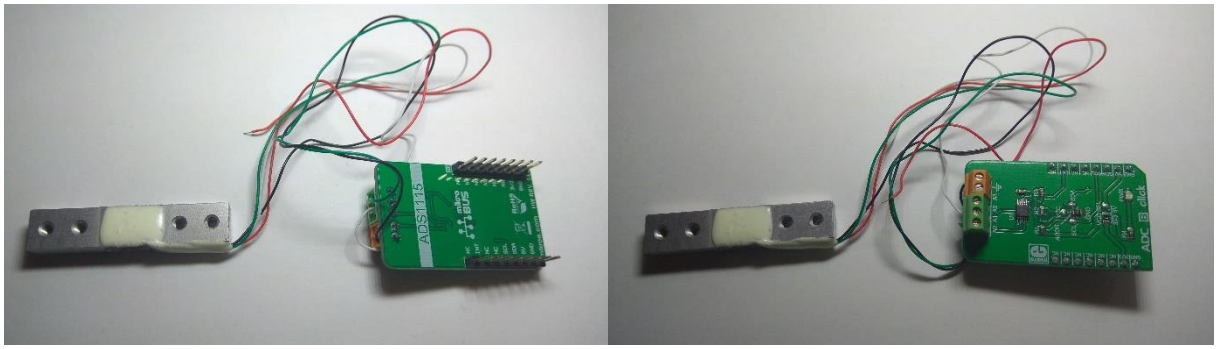

Figura 11. ADS1115 y célula de carga.

Con la intención de obtener medidas de fuerzas aplicadas en una estructura se instalan en este proyecto una célula de carga y un conversor analógico digital (ADC) externo que permite medir la señal con mejor precisión que Arduino (12 vs 16bit). En este caso se utiliza Mikroe 3394, cuyo principal componente es un ADS1115. Las especificaciones de la célula de carga y del ADC, están indicadas en las tabla 6 y tabla 7 respectivamente. Ambos elementos se han conectado como se muestra en la figura 11.

Tabla 6. Especificaciones célula de carga. (Electronics, 2015).

Atributo	Valor
Capacidad de carga	1kg
Voltaje salida	4mV/V
Salida de cero	0,05% FS
Precisión	0,02% FS
Precisión-Temperatura	0,02% FS
Temperatura de trabajo	-10-55°C
Sobrecarga de seguridad	150%
Salida	4 cables (puente Wheatstone)
Material	Aluminio

La célula de carga puede variar en función de su capacidad de carga y de la estructura en cuestión a estudiar, siendo intercambiables sin ningún problema mientras que basen su

funcionamiento en el puente Wheatstone. Los cuatro cables que disponen las células de carga son estándares teniendo cada color una función diferente:

- Rojo. Tensión de alimentación
- Negro. Toma a tierra
- Verde. Señal positiva
- Blanco. Señal negativa

Tabla 7. Especificaciones ADS1115. (Texas Instruments, 2016).

Atributo	Valor
Fabricante	Mikroe
Tamaño	42,9x25,4mm
Alimentación	3,3-5V
Frecuencia muestreo	8-860 Hz
Resolución	16bits
Interfaz de comunicación	I2C (4 direcciones)
Ganancia	±0.256, ±0.512, ±1.024, ±2.048V, ±4.096, ±6.144
Temperatura de operación	-40-125ºC
Tipo de muestreo	4 entradas únicas
	2 entradas diferenciales
	Comparador
Precio	18,9€

Antes de elegir el conversor analógico digital ADS1115 se probó un ADC Hx711, aunque tiene mayor precisión 24 bits, su gran inconveniente es que su frecuencia de muestreo máxima es 80Hz, suponiendo un 10% del máximo que permite el ADC seleccionado finalmente.

El ADS1115 dispone de 4 ADCs de 16 bits, siendo 15 para la medición y 1 para el signo. Para este proyecto se utiliza en modo diferencial, por lo que se pueden leer dos células de carga con un dispositivo. Se conecta al microcontrolador por I2C, siendo muy sencilla su lectura utilizando los pines SDA y SCL. En este ADS1115, al venir integrado en una placa de Mikroe, viene asignado ya la dirección I2C del dispositivo, asociado al pin GND por lo tanto la dirección 0x48. La tensión de alimentación también está fijada en 3,3V, no siendo posible su alimentación a 5V.

Los pines de este modelo son:

- A0. Entrada señal analógica.
- A1. Entrada señal analógica.
- A2. Entrada señal analógica.
- A3. Entrada señal analógica.
- GND. Toma a tierra
- 5V. Entrada de tensión. No se utiliza al venir predefinido en la PCB la de 3,3V
- 3,3V. Entrada de tensión.
- SDA. Trasmisión de datos, protocolo I2C.

- SCL. Señal de reloj, protocolo I2C.
- INT. Señal de alerta. Para lectura en continua.
- NC. Pines sin usos en la PCB, pensados para cuando se coloca otro modulo.

La instalación en Arduino Due se representa en la figura 12 y en la figura 13 se muestra el esquema de montaje en la TivaC. Para el correcto funcionamiento del bus I2C, es necesario la colocación de una resistencia pull up en cada conexión. En este caso se han conectado 2 resistencias de 220Ω desde la alimentación de 3,3 V.

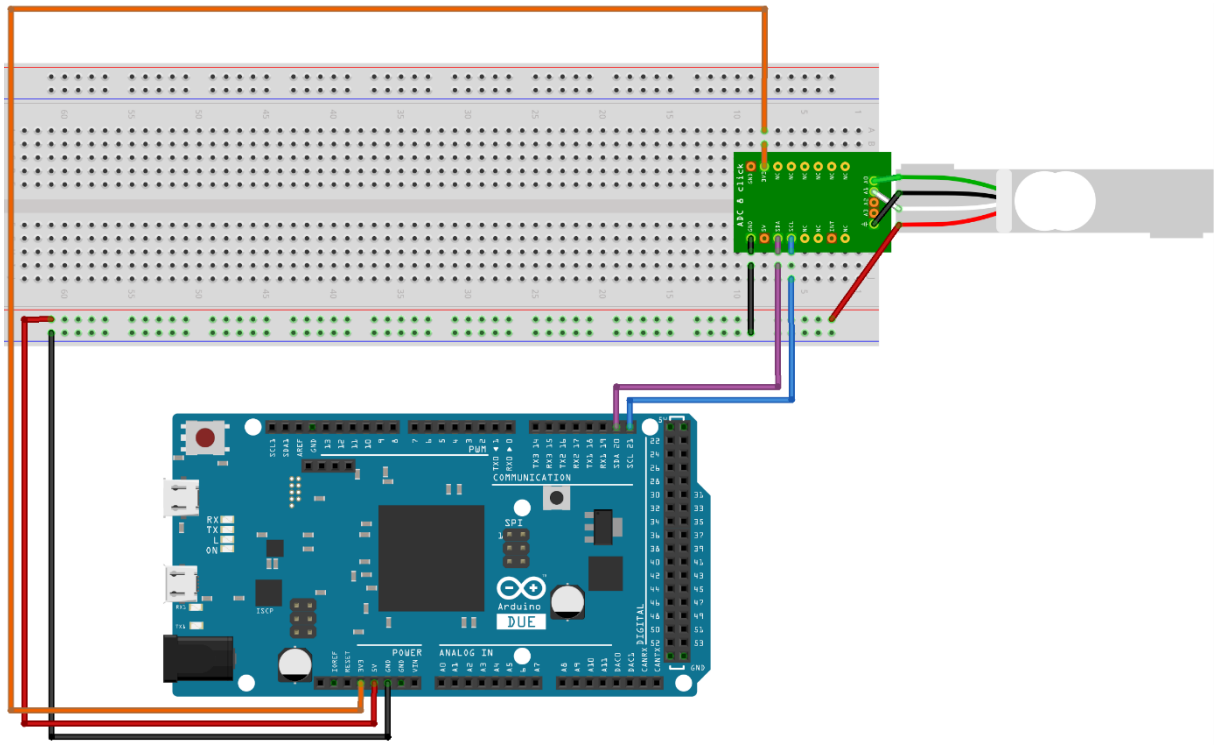

Figura 12. Esquema de montaje ADS1115 y célula de carga en Arduino.

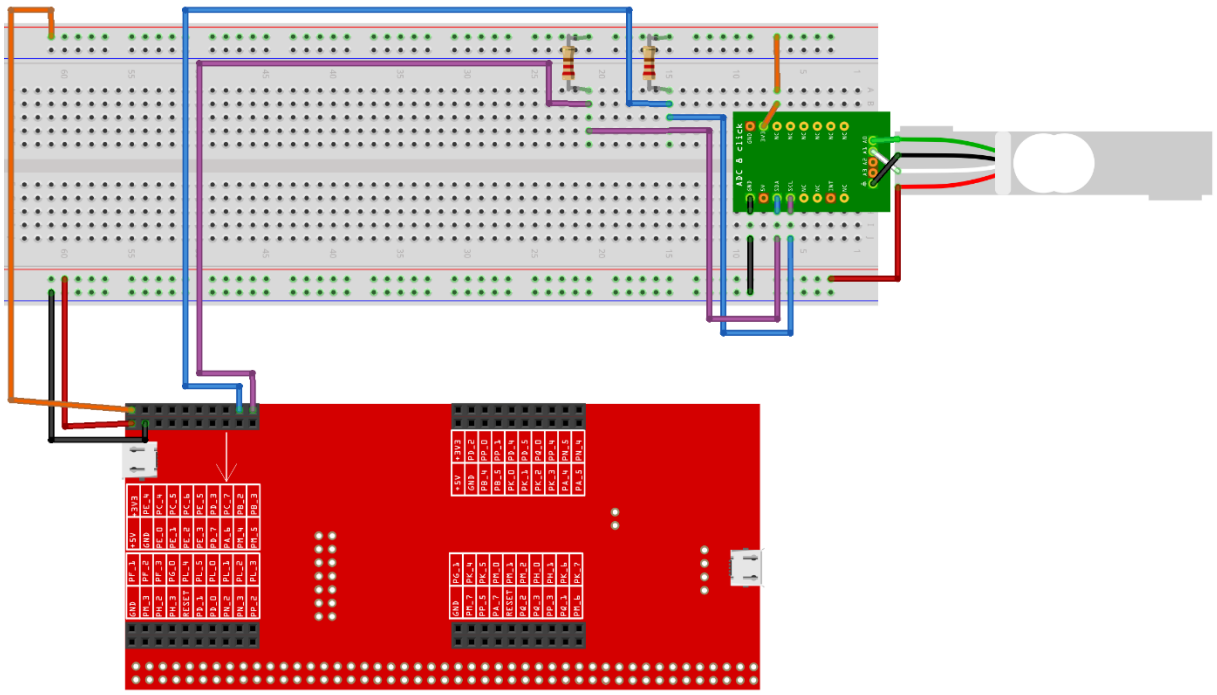

Figura 13. Esquema de montaje ADS1115 y célula de carga en TivaC.

2.3.2 Acelerómetro

Para ver cuál es la respuesta de las estructuras se ha seleccionado un acelerómetro MEMS (sistemas microelectromecánicos) de tres ejes MMA7361LC como el que se muestra en la figura 14.

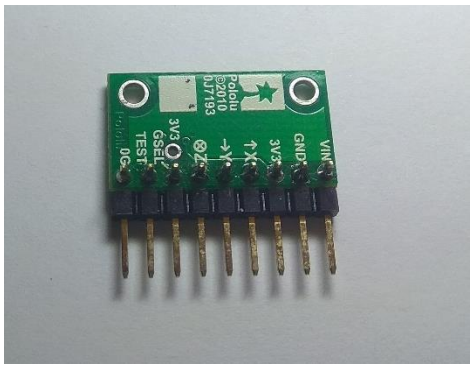

Figura 14. Acelerómetro MMA7361LC.

Las especificaciones del acelerómetro utilizado se presentan en la tabla 8.

Tabla 8. Especificaciones MMA 7361LC. (Freescale Semiconductor, 2011)

Valor	Atributo
Tamaño	23x13x2 mm (sin pines)
Interfaz	Salida analógica (3,3V)
Consumo	400 µA
Alimentación	2,2-16 V
Voltaje de operación	2,2-3,6 V
Rango de sensibilidad	±1,5g (por defecto), ±6g
Sensibilidad	800 mV/g (±1,5 g) y 200 mV/g (±6 g)
Procesado de la señal	Filtro paso bajo
Coste	2,36 €

Los pines de este modelo son:

- VIN. Alimentación.
- GND. Toma tierra
- 3,3V. Indica la tensión real para calcular el punto medio de aceleraciones positivas/negativas.
- X. Aceleraciones en el eje indicado.
- Y. Aceleraciones en el eje indicado.
- Z. Aceleraciones en el eje indicado.
- GSEL. Cambiar sensibilidad del dispositivo. Si se envía voltaje se selecciona el rango de media de 6g.
- TEST. Autoverificación de funcionamiento correcto.
- 0G. Caída libre cuando los tres ejes marcan gravedad 0.

En la figura 15 y en la figura 16 se muestran los esquemas de montaje del acelerómetro en Arduino DUE y en TivaC respectivamente.

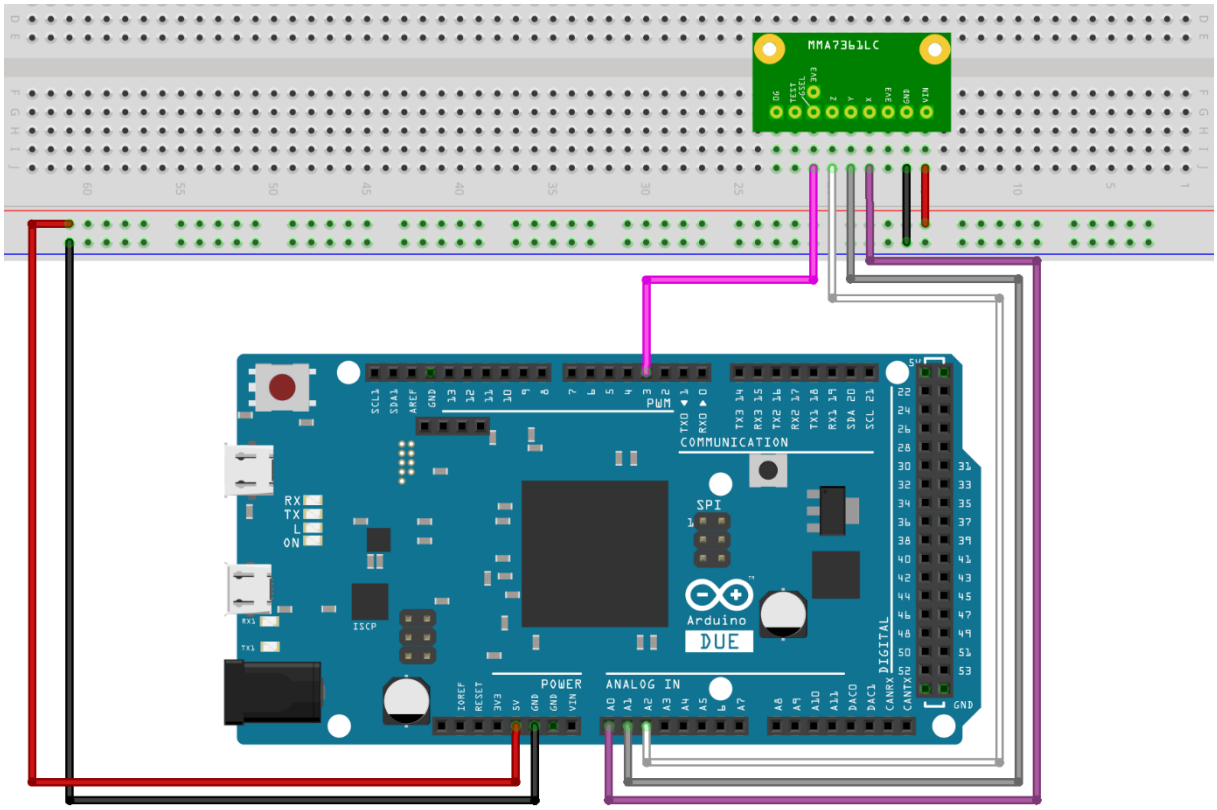

Figura 15. Esquema de montaje acelerómetro en Arduino.

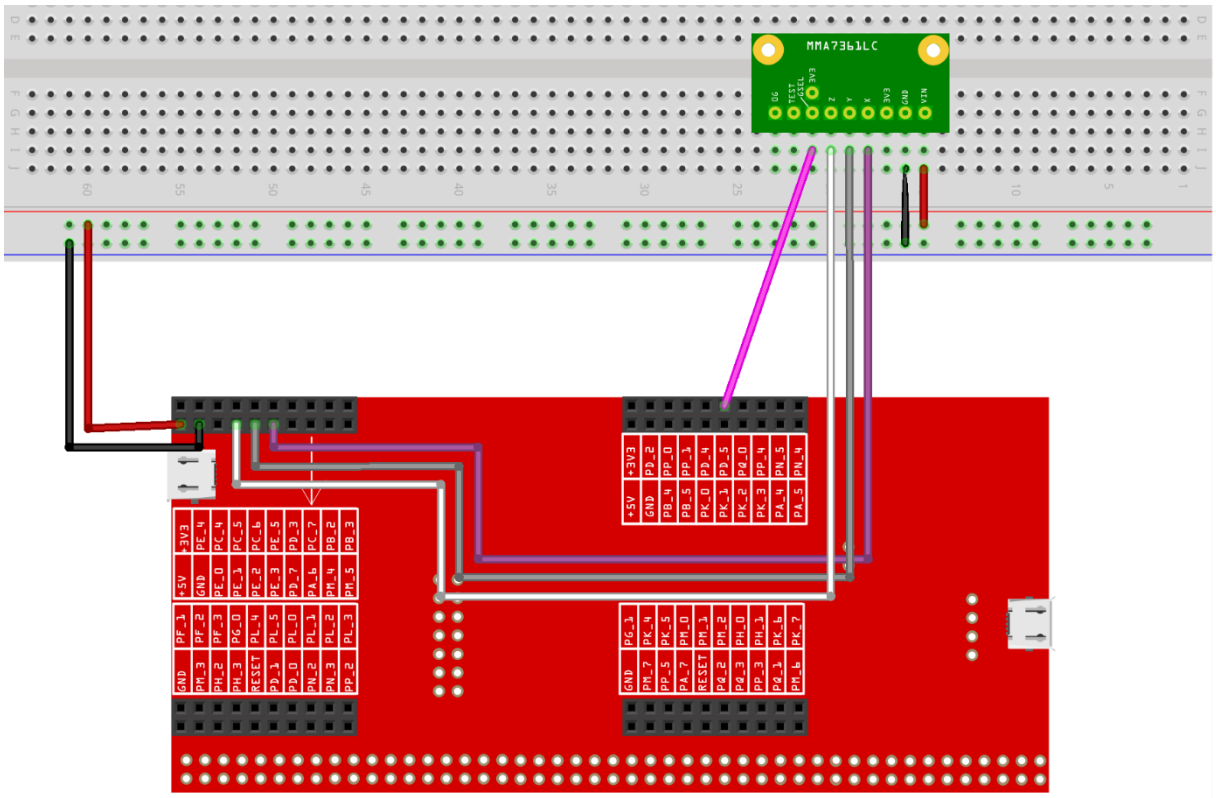

Figura 16. Esquema de montaje acelerómetro en TivaC.

2.3.3 Temperatura y humedad

Interesa conocer las condiciones ambientales a las que se realizan las mediciones por ello se ha incorporado un sensor digital de humedad y temperatura de bajo coste DHT22, como el que se muestra en la figura 17. Estos datos es importante conocerlos pues condicionan las propiedades de los materiales y con ello el comportamiento de las estructuras.

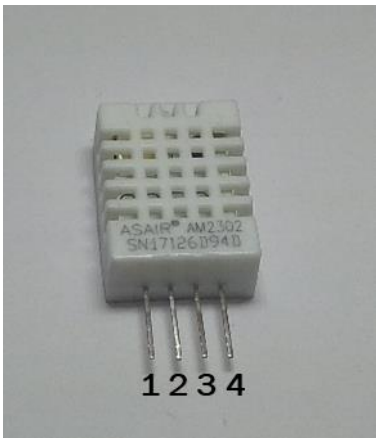

Figura 17. Sensor digital de temperatura y humedad DHT 22.

Los pines de este modelo son:

1. VCC. Voltaje de entrada
2. Datos.
3. No conectado
4. GND. Toma tierra

En la tabla 9 se recogen las especificaciones del sensor seleccionado.

Tabla 9. Especificaciones técnicas DHT 22. (Liu, 2016)

Atributo	Humedad	Temperatura
Tamaño	15,1x25,1x7,7 mm (sin pines)	
Potencia de alimentación	3,3-5,5V DC	
Señal de salida	1 pin digital	
Sensor	Condensador de polímero de humedad	
Rango operativo	0-100%HR	-40/80ºC
Precisión	±2%HR	±0,5ºC
Resolución	0,1%HR	0,1ºC
Repetibilidad	±1%HR	±0,2ºC
Precio	8,96 €	

En la figura 18 y en la figura 19 se muestran los esquemas de montaje del sensor DHT22 en Arduino DUE y en TivaC respectivamente. Para ello es necesario una resistencia de entre 4,7 kΩ y 10 kΩ.

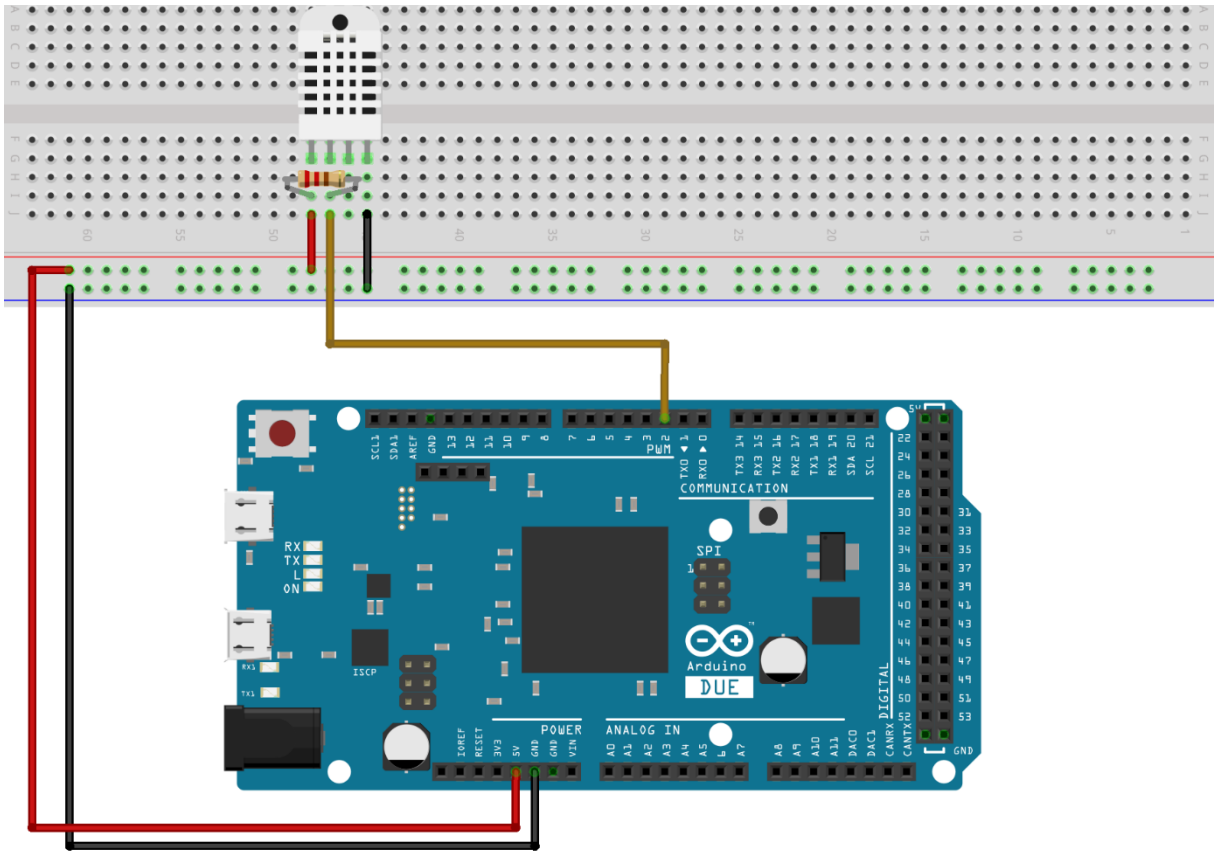

Figura 18. Esquema de montaje DHT 22 en Arduino.

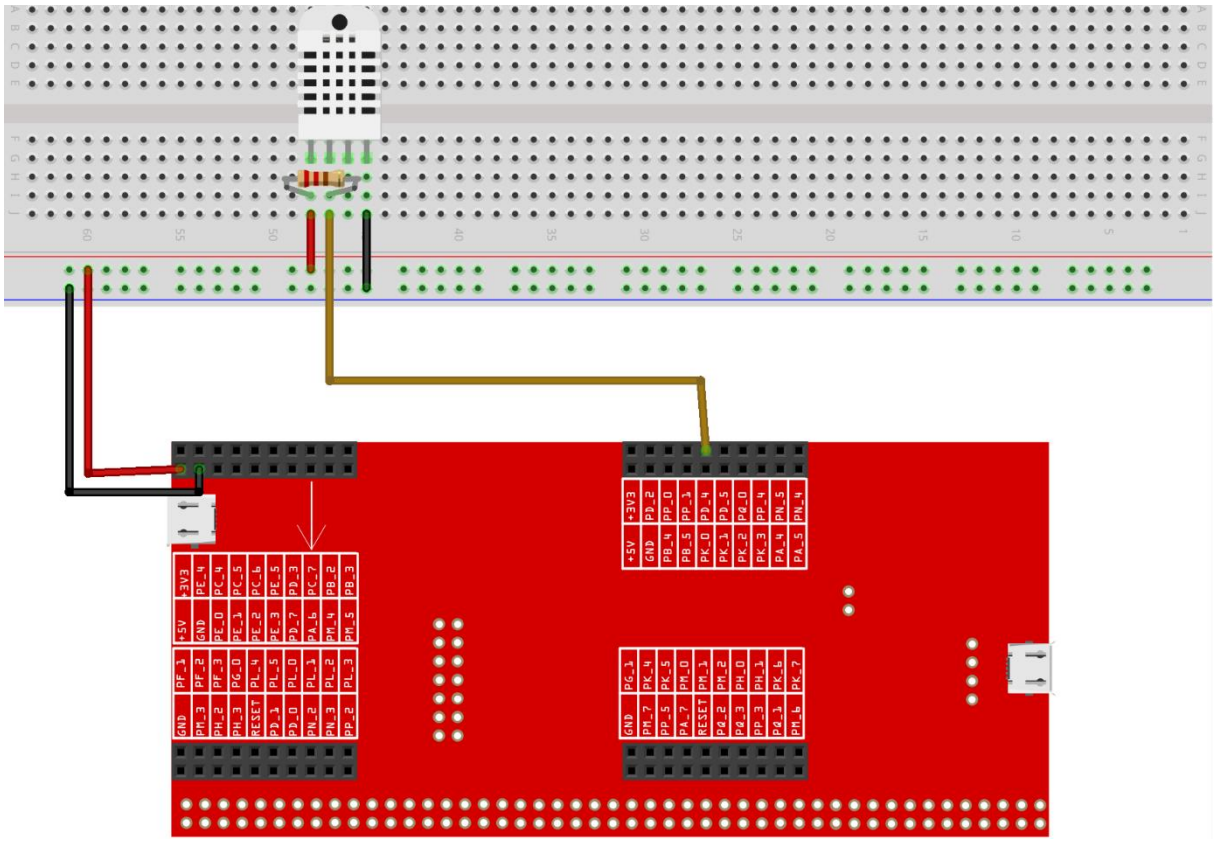

Figura 19. Esquema de montaje DHT 22 en TivaC.

2.3.4 Módulo de comunicación

La comunicación entre el microcontrolador y la Rasberry Pi se realiza mediante bluetooth. Se ha seleccionado el módulo de comunicación bluetooth de baja energía HC-08, como el que se muestra en la figura 20.

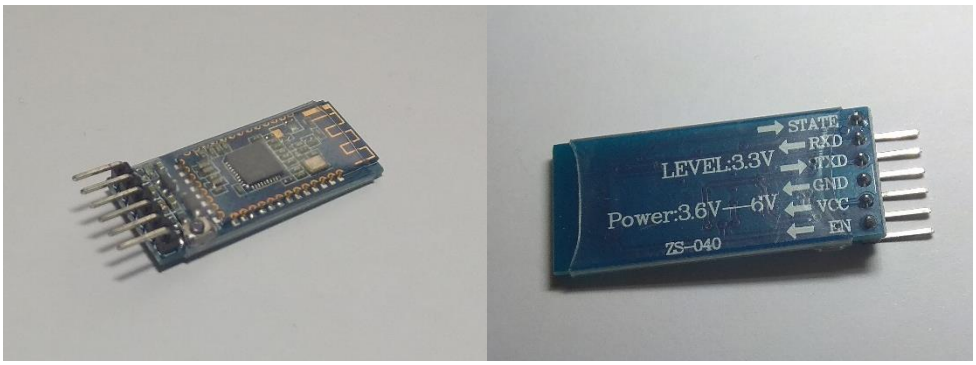

Figura 20. Módulo bluetooth HC-08.

Este dispositivo conectado al microcontrolador actúa como esclavo, solo pudiéndose conectar con un dispositivo maestro que en este caso es la Raspberry Pi. En tabla 10 se recogen las principales características de este elemento:

Tabla 10. Especificaciones modulo HC-08. (*Keyestudio HC-08 Bluetooth Module*, n.d.)

Atributo	Valor
Alimentación	5V
Banda de frecuencia	2,4G
Velocidad de transmisión	1Mbps
Interfaz de comunicación	UART 3,3 V
Antena	Incluida en la PCB
RSSI (Indicador de fuerza de la señal recibida)	No disponible
Distancia máxima	80m
Temperatura de trabajo	-25-75ºC
Humedad de trabajo	10-90%
Coste	6,59€

Los pines de este modelo son:

- EN. Habilitar configuración
- VCC. Voltaje de entrada
- GND. Toma de tierra
- TXD. Transmisión de datos
- RXD. Recepción de datos
- STATE. Situación del módulo.

Utilizando el IDE de Arduino con el dispositivo conectado según la figura 21 y mediante el uso de la librería “SoftwareSerial” para Arduino y un Arduino Mega, se realiza la configuración del dispositivo. Esta configuración no se puede realizar con el Arduino Due

ni con los microcontroladores de Texas Instruments, pues no reconoce la librería, aunque este instalada en Arduino IDE o en Energia IDE respectivamente. Para la configuración se requiere cargar el programa de configuración que viene con la librería citada anteriormente, anexo 1, y mediante el uso de unos comandos específicos, incluidos en tabla 11, se configura el dispositivo como esclavo y se cambia la velocidad de transmisión a 115200 baudios, siendo el máximo posible en este dispositivo.

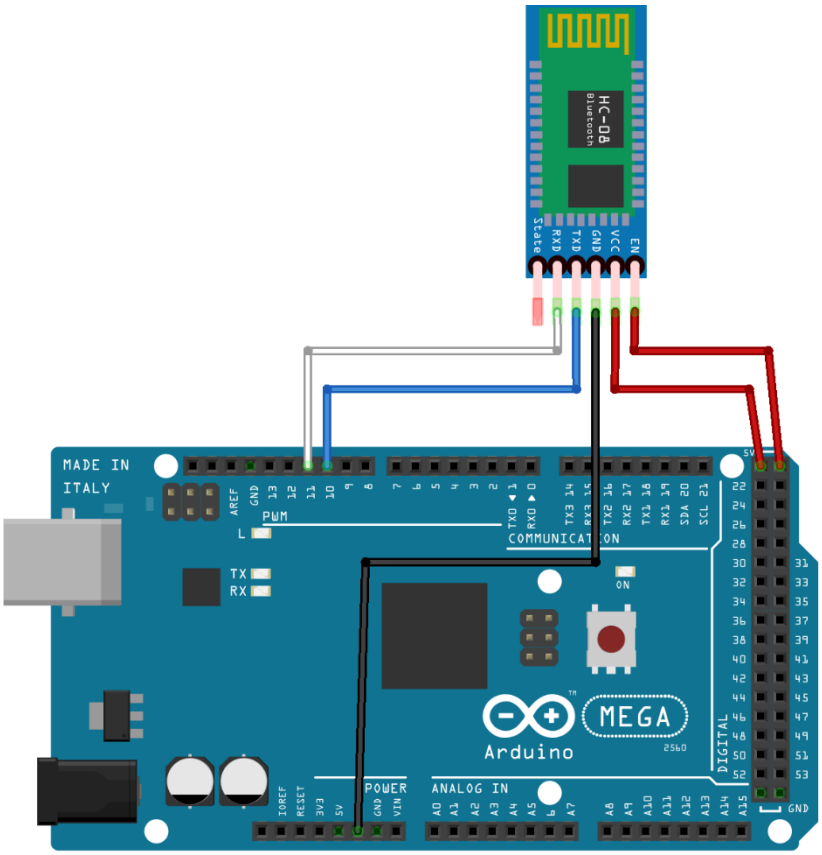

Figura 21. Esquema eléctrico configuración HC-08.

Tabla 11. Funciones configuración modulo HC-08.

Comando	Función
AT	Responde OK
AT+DEFAULT	Restaura los valores de fábrica
AT+RESET	Reinicia el módulo
AT+NAME=? AT+NAME=x	Responde con el nombre del módulo en BT o si se cambia la interrogación por un nombre cambia el nombre
AT+VERSION	Responde con la versión del firmware
AT+ROLE=? AT+ROLE=x (x=M o x=S)	Configurar el módulo con maestro o como esclavo. Viene configurado como esclavo.
AT+ADDR=? AT+ADDR=AABBCCDDEEFF	Responde con la dirección MAC, si se cambia la interrogación por una dirección la cambia

AT+BAUD=? AT+BAUD=115200	Muestra o configura la tasa de baudios. Siendo posible (1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200). Es necesario reiniciar el módulo.

Una vez realizado la configuración del módulo el esquema eléctrico para el Arduino Due en funcionamiento normal es el que se muestra en la figura 22 y en la figura 23 se muestra como es su montaje para el microcontrolador TM4C129.

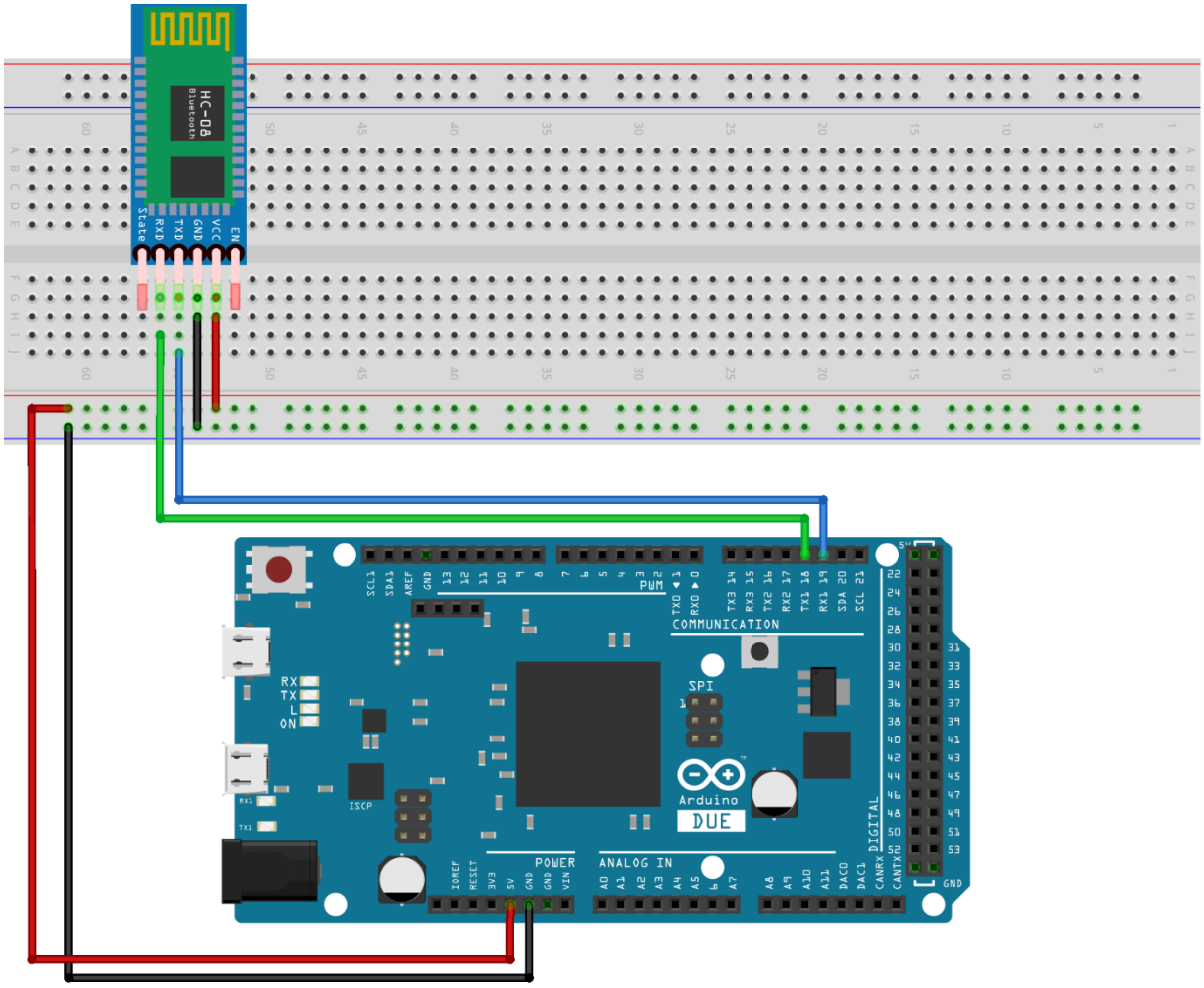

Figura 22. Esquema de montaje HC08 en Arduino.

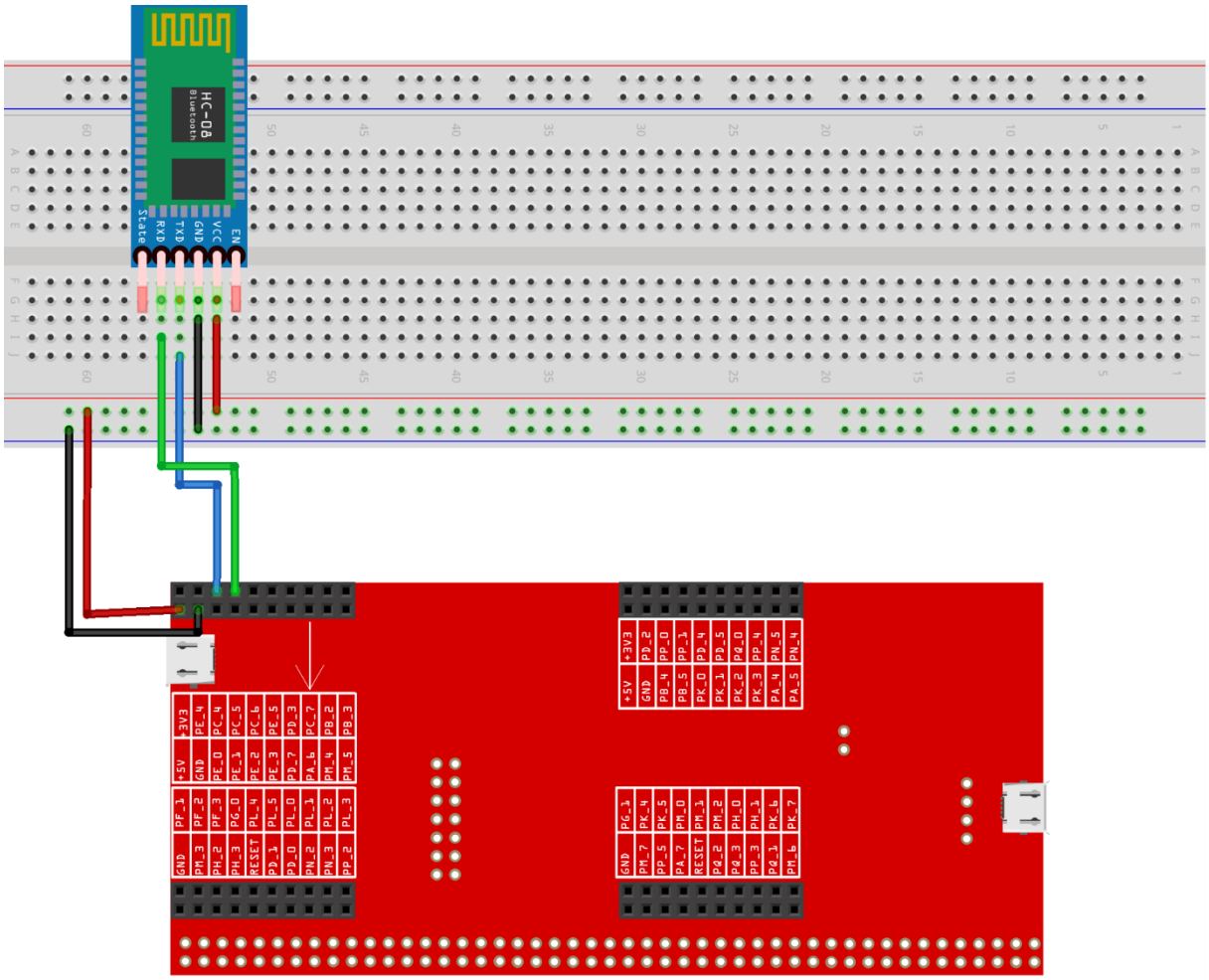

Figura 23. Esquema de montaje HC08 en TivaC.

2.4 Esquemas eléctricos

La conexión completa de todos los elementos para el Arduino Due se muestra recogida en la tabla 12 y en la figura 24. En ella se han añadido 3 leds, conectados en serie con resistencias de 220 Ω, que sirven para identificar cual es el estado del controlador en cada momento.

Tabla 12. Conexiones Arduino Due.

DUE			
Dispositivo	Conector	PIN	Sobrenombre
BLE	VCC	5V	
	GND	GND	
	TXD	19	RX1
	RXD	18	TX1
ADS1115	3V3	3V3	
	GND	GND	
	SDA	20	SDA
	SCL	21	SCL
Célula de carga	Verde	A0 (ADC)	
	Blanco	A1 (ADC)	
	Negro	GND	

	Rojo	5V	
Acelerómetro	X	A0	
	Y	A1	
	Z	A2	
	Gsel	3	
	GND	GND	
	Vin	5V	
DHT22	1	5V	
	4	GND	
	output	2	Cualquier PWM
LED Rojo	Señal	8	
LED Verde	Señal	9	
LED Amarillo	Señal	10	

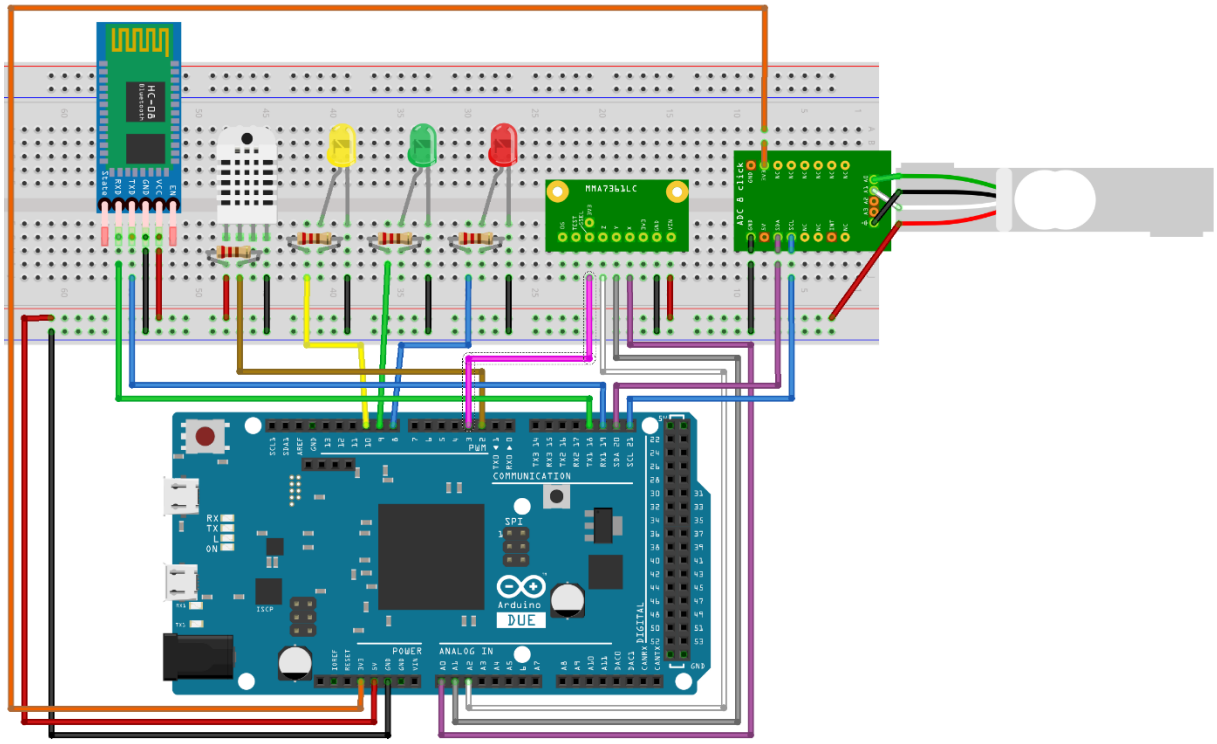

Figura 24. Montaje completo Arduino.

La conexión completa de todos los elementos para el modelo de Texas Instruments se muestra recogida en la tabla 13 y en la figura 25. En ella se han añadido 3 leds, conectados en serie con resistencias de 220 Ω, que sirven para identificar cual es el estado del controlador en cada momento.

Tabla 13. Conexiones TivaC.

TM4C1294XL			
Dispositivo	Conector	PIN	Sobrenombre
BLE	VCC	5V	
	GND	GND	
	TXD	PC_4	RX7
	RXD	PC_5	TX7

ADS1115	3V3	3V3	
	GND	GND	
	SDA	PB_3	SDA
	SCL	PB_2	SCL
Célula de carga	Verde	A0 (ADC)	
	Blanco	A1 (ADC)	
	Negro	GND	
	Rojo	5V	
Acelerómetro	X	PE_3	A0
	Y	PE_2	A1
	Z	PE_1	A2
	Gsel	PD_5	
	GND	GND	
	Vin	5V	
DHT22	1	5V	
	4	GND	
	output	PD_4	Cualquier PWM
LED Rojo	Señal	PK_3	
LED Verde	Señal	PK_2	
LED Amarillo	Señal	PK_1	

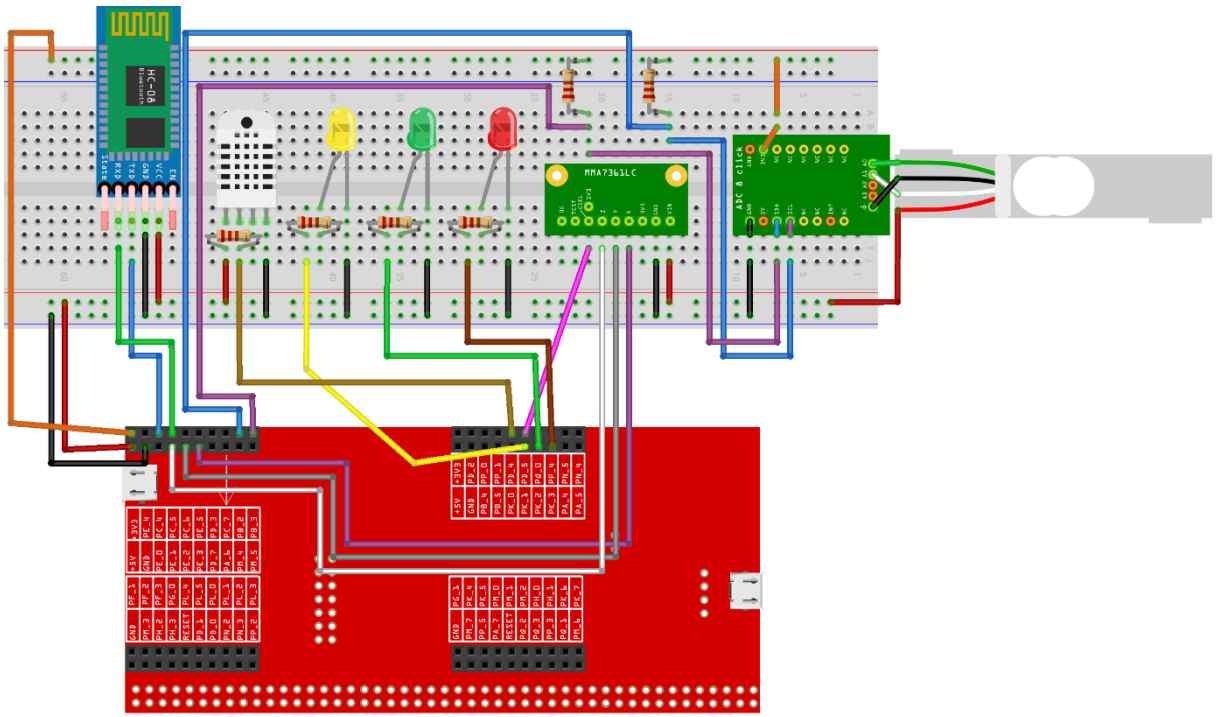

Figura 25. Montaje completo TivaC

En las figura 26 y figura 27, se muestran los montajes realizados físicamente para las dos tarjetas.

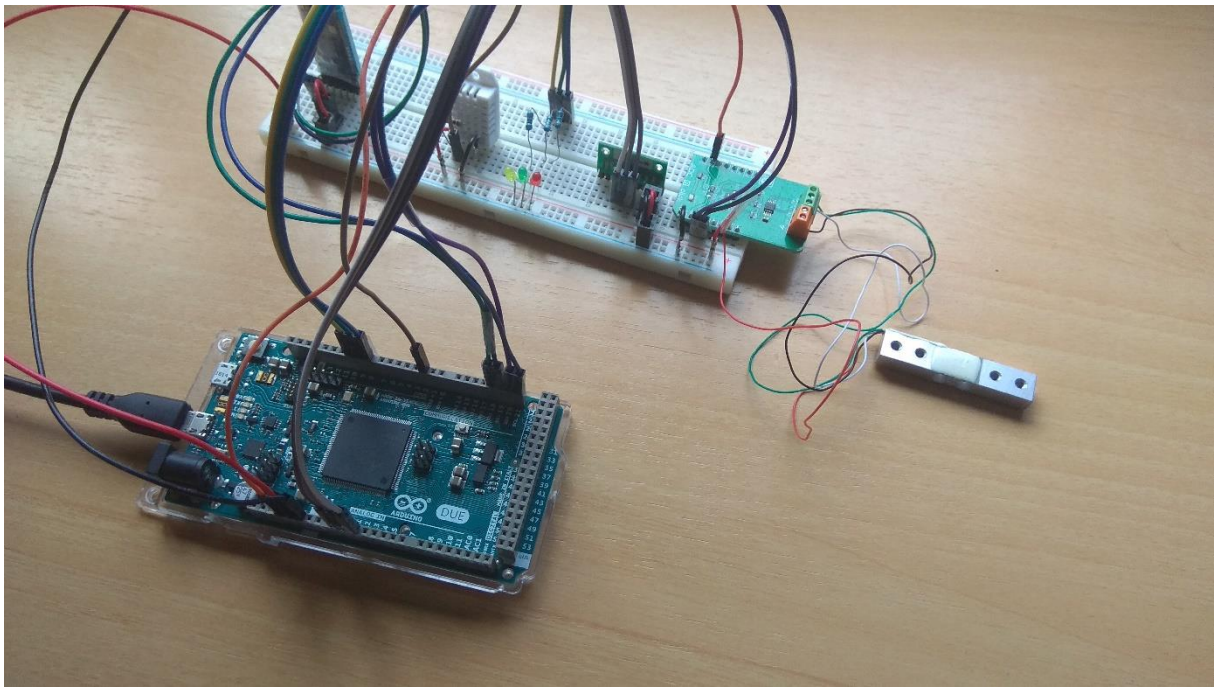

Figura 26. Montaje Arduino Due

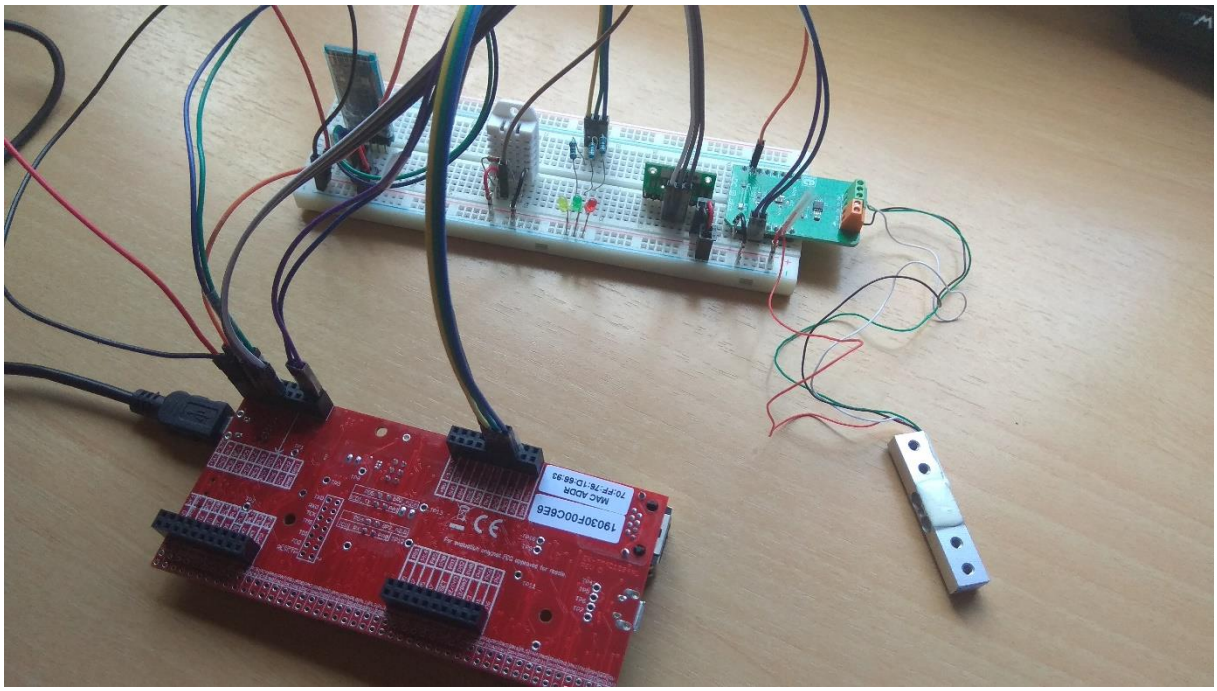

Figura 27. Montaje TivaC.

2.5 Resumen de coste

Los costes generados durante el proyecto en materiales se encuentran reflejados en la tabla 14, distinguiéndose en ella cual es el coste de realizar el sistema con un microcontrolador o con el otro.

Tabla 14. Resumen de coste.

Elemento	Coste	
Raspberry Pi	86,99€	
Microcontrolador	35€	23,39€
ADS + célula de carga	18,9€	
Acelerómetro	2,36€	
DHT22	8,96€	
HC-08	6,59€	
Cables Dupont (40 Uds.)	2,95€	
Breadboard (1)	3,17€	
Leds (50 Uds.)	1,99€	
Total	166,91€	155,3€

CAPÍTULO 3. SOFTWARE

En este capítulo se expone el software desarrollado para los distintos elementos. Para la programación de la Raspberry Pi el lenguaje seleccionado ha sido Python3 y con el propio desarrollador que trae Raspberry Pi OS, Python IDE. Para la programación de los microcontroladores se realiza con lenguaje propio de Arduino tanto para la placa de Arduino como para la placa de Texas Instruments.

Filosofía

La filosofía seguida para el desarrollo del sistema SHM es utilizar el microcontrolador como sistema de adquisición de datos exclusivamente y realizar todo el posterior procesado de los valores leídos en la Raspberry Pi. Esto se debe a la mayor potencia que tiene la Raspberry Pi, liberando al microcontrolador de tareas que puedan limitar las prestaciones obtenidas del sistema. Además al enviar los datos en bruto, facilita la comunicación bluetooth pues se trabaja solo con enteros de 16 bits, sin necesidad de utilizar variables en coma flotante de mayor tamaño, que provocan mayor tiempo de envió.

Se ha dado importancia a la comunicación en las dos direcciones, que permite seleccionar desde la Raspberry Pi las tareas que debe ejecutar el microcontrolador. Este envía, a parte de los datos adquiridos, cuál es su situación (midiendo, enviando, o finalizado), su memoria disponible si se va a hacer uso de ella, así como de la frecuencia a la que va a muestrear. Esto último es imprescindible para utilizar las dos placas de desarrollo sin necesidad de cambiar ninguna parte del software.

Se ha incluido la visualización en tiempo real las acciones sobre las estructuras y la opción realizar mediciones sin visualización para obtener muestreos a mayores frecuencias, que permitan hacer un análisis exhaustivo de la estructura.

La filosofía de adquisición de datos es la siguiente: el sensor de temperatura y humedad, se leen solo al principio, debido a que el tiempo de variación es mucho mayor que el tiempo de variación de las fuerzas y aceleraciones que afectan a la estructura. Por lo tanto, se puede considerar que no varían durante la duración de la medición. Esto permite obtener mayores frecuencias de muestreo de los sensores de fuerza y aceleración y realizar ensayos a máxima frecuencia de mayor duración.

Recomendaciones de uso

El modo lectura online es recomendable para ver cómo está funcionando el sistema a tiempo real. Es aconsejable realizar una medición a tiempo real antes de hacer mediciones offline, para comprobar que todo el equipo está funcionando correctamente. Para realizar ensayo que permitan analizar la estructura, haciendo procesado de datos, se recomienda utilizar el modo offline o el modo híbrido si es una medición de duración mayor de la que puede permitir el microcontrolador.

Antes de la realización de la FRF es aconsejable realizar algún filtro para acondicionar los registros y obtener mejores resultados. Los filtros son acumulativos, si se utilizan los datos de partida de un filtro ya realizado, por lo tanto si el usuario lo desea puede aplicar todos los filtros consecutivamente.

3.1 Python

Python es un lenguaje de programación interpretado y fácil de aprender, pues su filosofía se basa en la legibilidad de su código. Es un lenguaje de alto nivel, permitiendo varios estilos de programación: orientada a objetos, imperativa y funcional. Python es un lenguaje interpretado (se realiza la traducción del programa a código de maquina a medida que es necesaria, permitiendo así ejecutar el programa con un solo archivo fuente en sistemas

diferentes), dinámico (una variable puede tomar valores de distinto tipo) y multiplataforma (puede ejecutarse en Windows, Linux y Mac). Tiene interfaces para muchas llamadas de sistema y librerías, así como para varios sistemas de ventanas y es extensible en C o C++. También puede usarse como un lenguaje de extensión para aplicaciones que necesitan un interfaz programable. Es propiedad de Python Software Foundation que posee su licencia de código abierto, compatible con la Licencia pública general de GNU.

El entorno de desarrollo utilizado para la realización del programa para la Raspberry PI es el propio de Raspberry Pi OS Python 3 IDE. En los siguientes apartados se van a detallar las funcionalidades del programa y su desarrollo, mostrando las librerías y sus funciones utilizadas en el programa.

3.1.1 Funcionalidades

Las funcionalidades de la aplicación desarrollada para la Raspberry Pi se pueden apreciar en la interfaz gráfica mostrada en la figura 28. En ella se puede distinguir seis zonas valorando su funcionalidad:

1. Lectura. Se manda la orden para que el controlador lea y envíe los datos de los sensores. Tarado de los sensores, registro en tiempo real (online), registro a máxima frecuencia y posterior envío (offline) o registro en tiempo real sin visualización (híbrido).
2. Conexión bluetooth. Permite la conexión bluetooth y cuál es el estado de esta conexión, así como del controlador.
3. Información del ensayo realizado. Se muestra cuál es el estado del ensayo: no realizado, leyendo, enviando o finalizado. También se muestra la temperatura y humedad a la que se realiza el ensayo, el tiempo de lectura del microcontrolador, el tiempo total del ensayo, el número de muestras y la frecuencia a la que se ha realizado el muestreo.
4. Procesado de los datos. En esta sección se pueden realizar los siguientes filtros: media móvil (suavizar la señal), Detrend (eliminar la tendencia lineal), Butterworth (atenuando las señales a partir de una frecuencia determinada) y remuestreo (cambiar el número de muestras del ensayo). Se puede conocer la media cuadrática (RMS) y los máximos de las señales muestreadas. Realizar el cálculo de la respuesta en frecuencia (FRF), tomando los valores leídos o filtrados. Así mismo se realizan las gráficas de cada una en ventanas emergentes.
5. Abrir y exportar valores, en .txt, .csv y en formato para Matlab. Permite abrir registros realizados anteriormente, guardar las lecturas en online en formato bytes y guardar tanto los datos registrados, como los procesados para utilizar en otros programas
6. Gráficas de los valores leídos por el microcontrolador. En el caso de la gráfica de aceleración se puede mostrar u ocultar la línea de datos para cualquiera de los ejes.

Las gráficas mostradas a continuación para explicar las funcionalidades se corresponden con un ensayo realizado con una viga de policarbonato en voladizo, con una célula de carga y un acelerómetro situada en su extremo. El objetivo de las gráficas es ilustrar las funcionalidades, no realizar un análisis de la estructura. La explicación del ensayo y el análisis estructural se realiza en el capítulo 4.

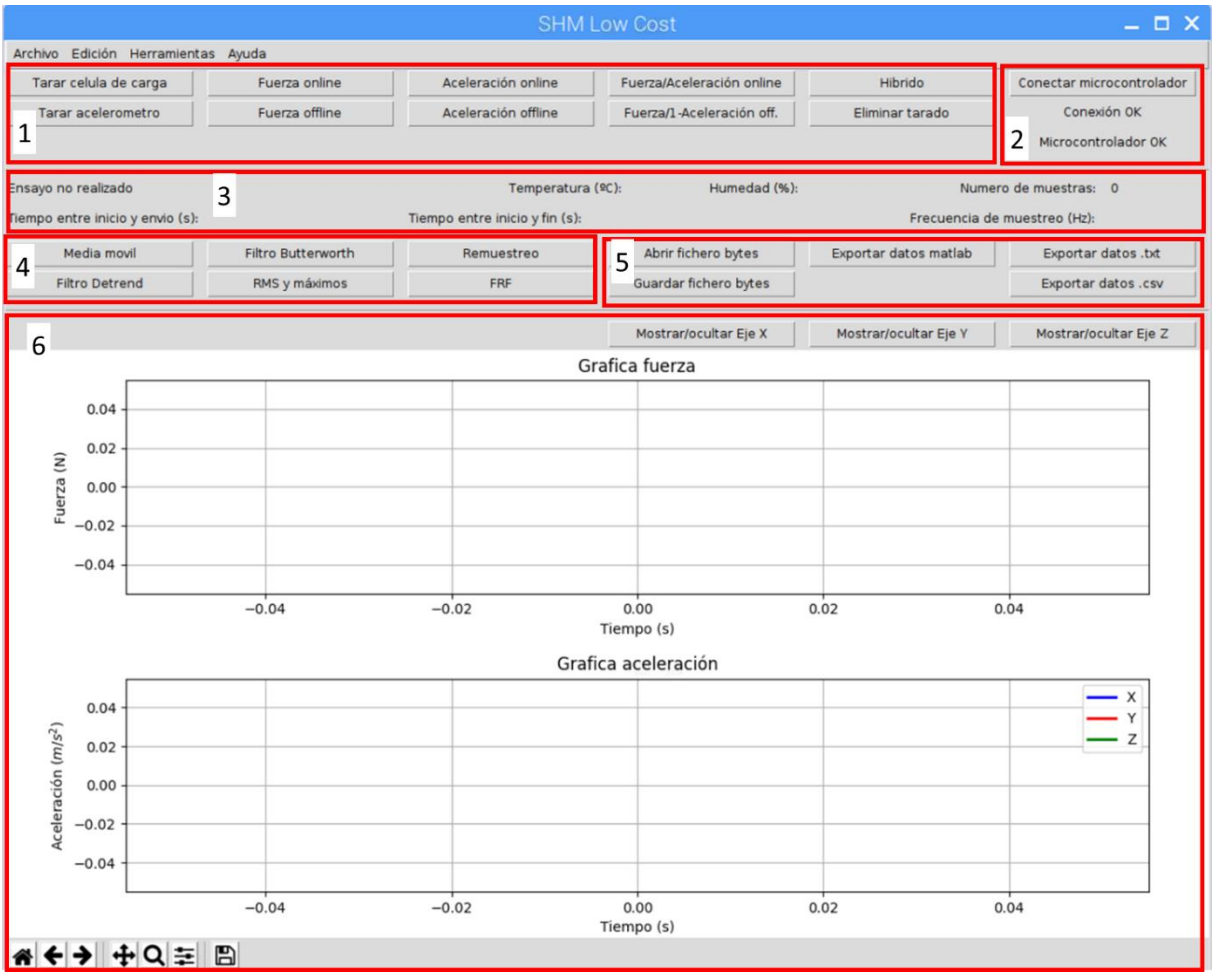

Figura 28. Interfaz de la aplicación.

Conectar Bluetooth

Al iniciar el programa se ejecuta automáticamente la conexión, pero se ha añadido está funcionalidad por si no ha sido posible establecerla o por si se quiere cambiar de dispositivo. Para ello se pide la introducción de la dirección MAC del dispositivo bluetooth al que se desea conectar por el usuario, como en la figura 29. La dirección MAC, también conocido como dirección física, es un identificador que consta de 6 bloques de dos caracteres hexadecimales, separados por dos puntos y es única de cada dispositivo.

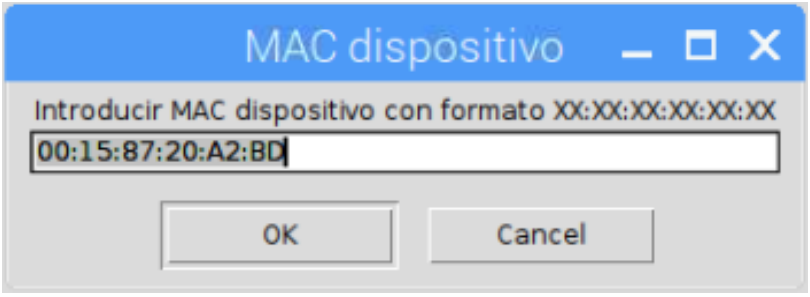

Figura 29. Introducción MAC por el usuario

Tarado

Al ejecutar está función se da una orden para que el microcontrolador haga una serie de lecturas y devuelva su media. Una vez recibido **el valor se guarda en una base de datos, que debe estar en la misma carpeta que el programa desarrollado y se llama "Datos sensores",** no siendo necesaria actualizar constantemente, solo cuando se cambien las

condiciones de la estructura. Esto permite obtener los datos centrados, eliminado las fuerzas o aceleraciones estacionarias del sistema.

Eliminar tarado

Se puede restablecer los valores por defectos de tarado, pulsando esta opción. Con ella se guardan estos valores en la base de datos, eliminando los valores guardados anteriormente.

Lectura online

Permite la lectura de los sensores y su visualización en tiempo real. Para ello, se selecciona la magnitud que se quiere medir: fuerzas, aceleraciones (permitiendo seleccionar el eje) o ambas. Una vez seleccionado que se quiere medir, se introduce la duración de la medición a realizar en segundos y ya con ello se envía la orden al microcontrolador. La duración del ensayo no está acotada, debe ser el usuario quien decida los segundos que desea medir teniendo en cuenta que el programa ejecuta esta función hasta que concluya el tiempo.

Primero se recibe la señal de que ha recibido la orden correctamente y que va a comenzar a leer y posteriormente otro código confirmando que va a empezar a enviar datos. Recibiendo los primeros datos la temperatura y humedad, que se muestran en la interfaz. A partir de aquí, ya se empiezan a leer las magnitudes físicas, que se reciben mediante bluetooth desde el microcontrolador, se guardan en listas, según la magnitud que se corresponda, y se cargan en las gráficas embebidas en la interfaz gráfica. Durante el proceso no se puede realizar nada, aparte de la visualización de los datos. Cuando la ventana de muestreo se ha completado, la Rasberry Pi recibe un código fin, que finaliza el ensayo, cargándose a su vez la frecuencia a la que ha sido ejecutado y mostrando una ventana emergente.

Una vez finalizado el ensayo, se muestra con la figura 30, ya se pueden utilizar el resto de las funciones del programa, como se expone más adelante. No se recomienda la realización del cálculo de la respuesta en frecuencia debido a que es mejor obtenerla cuando la lectura se realiza al máximo de frecuencia con el modo de lectura offline o por lo menos en el modo híbrido.

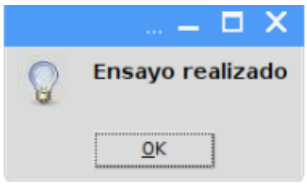

Figura 30. Aviso de lectura finalizada.

Lectura offline

La selección de la magnitud a medir y la duración de la medición es igual que al modo online, pero en este caso se pide un nombre para crear un fichero de bytes donde se almacenan los datos durante la comunicación bluetooth. **Se genera un fichero .txt que lleva incluido "_b" al final del nombre introducido por el usuario para identificar que contiene bytes.** Si se introduce un nombre ya existente, el fichero nuevo sobrescribe al anterior. Tras finalizar el ensayo estos bytes, se cargan en arrays para generar las gráficas, poder realizar su procesado o exportar los valores leídos. Durante el ensayo no se puede realizar nada, se muestran los datos climatológicos en los que se está realizando el ensayo y se actualiza el estado del microcontrolador, informando cuando está leyendo o enviando datos. El ensayo finaliza como en el modo offline, con una ventana emergente, pero justo antes carga en las gráficas de la interfaz todos los datos leídos, así como toda la información sobre el ensayo que se recoge en la ventana principal. Este modo tiene limitado la duración de la toma demuestra por la memoria del controlador que se disponga.

Si se intenta hacer un ensayo de mayor duración de la que se puede almacenar en la memoria del microcontrolador sale un aviso informando de que no se puede realizar y de cuál es la duración máxima de la medición en segundos, como aparece en la figura 31. Esto es debido a que el microcontrolador analiza su memoria RAM disponible, para evitar la saturación de la memoria y que colapse el sistema.

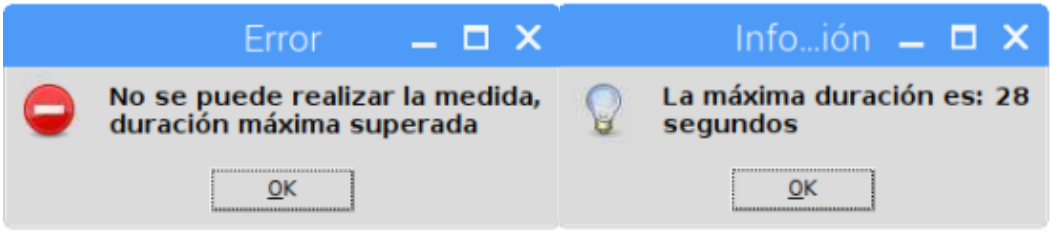

Figura 31. Aviso de duración máxima superada

Una vez finalizado el ensayo se puede utilizar el resto de las funciones del programa, como se expone más adelante. Una vez finalizado, se cargan las gráficas como se muestran el figura 32.

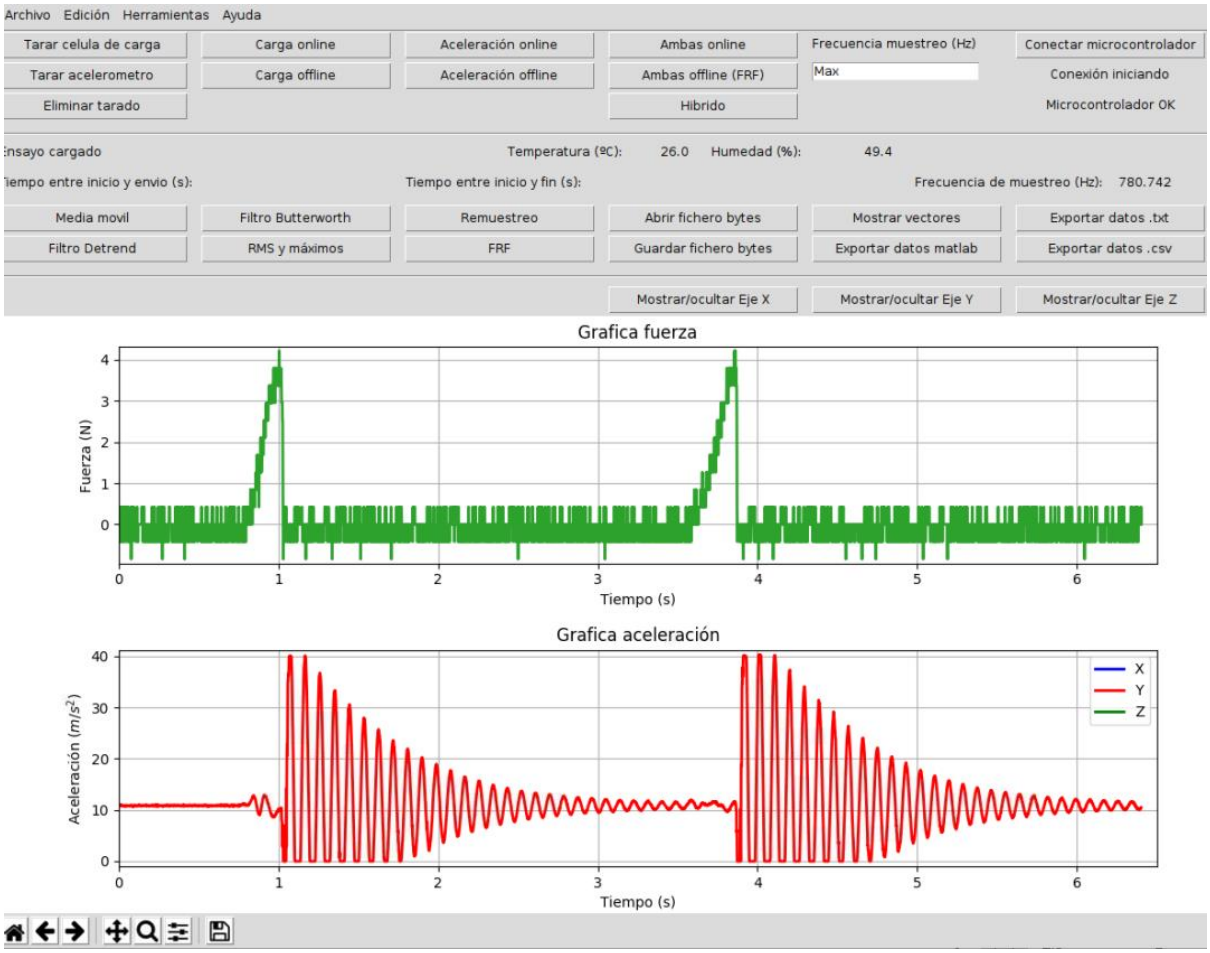

Figura 32. Ensayo realizado

Lectura online sin visualización

Se ha realizado un híbrido entre la lectura online y offline, basado en leer con el microcontrolador y enviarlos datos según se leen, pero sin realizar la visualización de los datos hasta haber concluido la toma de muestras. Dentro del programa de la Raspberry Pi, actúa como si fuera modo offline, guardando los datos en byte dentro de un fichero .txt. Como principal ventaja respecto del modo online, es obtener mayores frecuencias de

muestreo (250 Hz vs 62 Hz), no alcanzando las del modo offline (dependiendo del controlador 447 Hz o 780 Hz), pero sin el limitante del modo offline de la memoria del microcontrolador. La duración del ensayo no está acotada, debe ser el usuario, quien decida los segundos que desea medir teniendo en cuenta que el programa ejecuta esta función hasta que concluya el tiempo.

Conversión de datos registrados

En el programa de Python se reciben los datos en bruto, siendo necesario la conversión de unidades para que los valores leídos tengan significado físico. También se eliminan los offset, cargas o aceleraciones estacionarias, para que los valores estén en torno a 0. Para el caso de la célula de carga el valor de la fuerza en newton se obtiene mediante la ecuación (1).

$$Fuerza(N) = Valor(bit) \times Ganancia\left(\frac{mV}{bit}\right) \times Escala\left(\frac{N}{mV}\right) - Offset(N) \quad (1)$$

Siendo los parámetros:

- Valor, es el dato que recibe la Raspberry Pi, convertido por el ADC, Es un valor de 16bits, entre −32768 y 32767.
- Ganancia, establecida en Arduino, en función de las características de la célula de carga que esté conectada al ADS1115.
- Escala, se ha calibrado mediante la aplicación de varias fuerzas conocidas de distintos valores. Se ha estimado que su valor es igual a 226,4.
- Offset, se calcula mediante el programa con la función tarado de la célula de carga, que obtiene la lectura de 20 valores seguidos y realiza la media.

Los valores de aceleración se obtienen a partir de la ecuación (2).

$$\begin{aligned} &Aceleración\left({}^{m}/_{s^2}\right) \\ &= Valor(bit) \times Resolución\left(\frac{V}{bit}\right) \times \frac{1}{Sensibilidad}\left(\frac{g}{V}\right) \times S.I\left(\frac{{}^{m}/_{s^2}}{g}\right) \\ &- Offset({}^{m}/_{s^2}) \end{aligned} \quad (2)$$

Siendo los parámetros:

- Valor, es el dato que recibe la Raspberry Pi, convertido por el ADC, Es un valor de 12 bits, entre 0 y 4095.
- Resolución, se obtiene del rango de voltaje en el que trabaja de 0-3,3V entre el número de valores posibles
- Sensibilidad, dado por el fabricante, a 6 g de 0,2 V/g.
- Transformar el valor de unidades al sistema internacional
- Offset, se calcula mediante el programa con la función tarado del acelerómetro, que obtiene la lectura de 20 valores seguidos y realiza la media.

Si se sustituyen por los valores de cada parámetro se obtiene la ecuación (3).

$$\begin{aligned} &Aceleración({}^{m}/_{s^2}) = Valor(bit) \times \frac{3{,}3}{4096}\left(\frac{V}{bit}\right) \times \frac{1}{0{,}2}\left(\frac{g}{V}\right) \times 9{,}8\left(\frac{{}^{m}/_{s^2}}{g}\right) - \\ &Offset({}^{m}/_{s^2}) = Valor(bit) \times 0{,}0394775390625\left(\frac{{}^{m}/_{s^2}}{bit}\right) - Offset({}^{m}/_{s^2}) \end{aligned} \quad (3)$$

Los datos para la conversión de los valores leidos a magnitudes físicas, se guardan en una base de datos, llamada “Datos sensores”, que debe estar en la misma carpeta que el programa desarrollado. En esta base de datos, se almacena:

- Nombre de sensor.
- Tipo de sensor.
- Offset.
- Escala.
- Ganancia.
- Eje.

Si se sustituye alguno de los sensores utilizados, por otro equivalente, es necesario cambiar aquí los parámetros del sensor.

RMS y máximos

Interesa conocer cuáles son los valores máximos y la media cuadrática que se han producido en la lectura de datos, pues son los valores de más interés que pueden afectar a la estructura. En la figura 33, se muestran, en una ventana emergente, estos valores de una lectura de la carga ejercida sobre un sistema y sus aceleraciones en el eje Y. La media cuadrática se calcula a partir de la ecuación (4). Se evalúan los máximos y los mínimos de todas las señales que se pueden leer con sus valores en bruto, sin realizar procesado.

$$\bar{x} = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}{n}} \qquad (4)$$

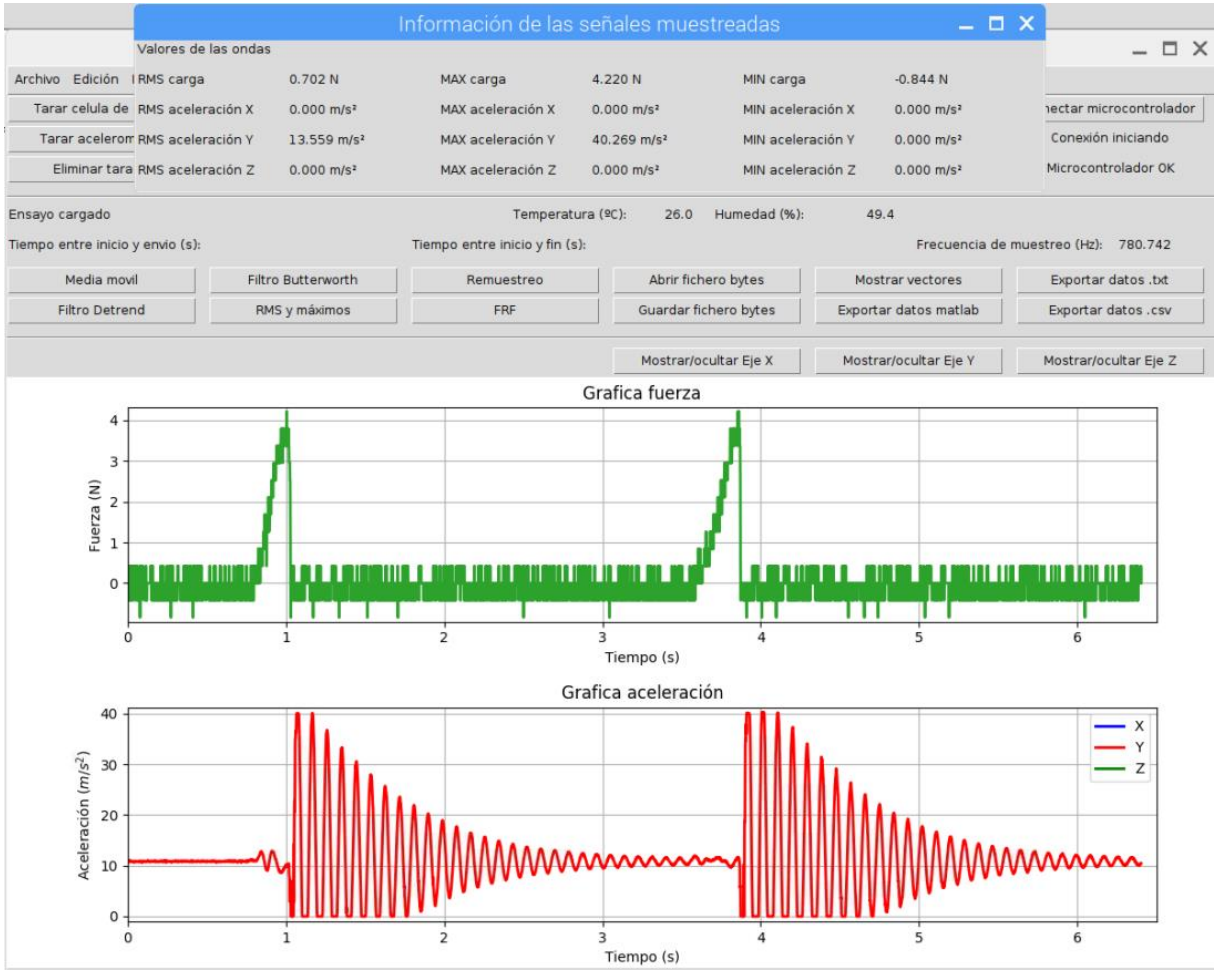

Figura 33. Ventana con información de la RMS y máximos de una señal muestreada

Media móvil

Para el cálculo de la media móvil es necesario el tamaño de la ventana, que introduce el usuario, sobre el cual se va a calcular cada valor de la media, la figura 34. **Siempre se debe introducir un número entero positivo.** Se puede utilizar algún los datos de algún filtro usado anteriormente. El objetivo de este filtro es suavizar las fluctuaciones no deseadas y mostrar la señal con menos ruido.

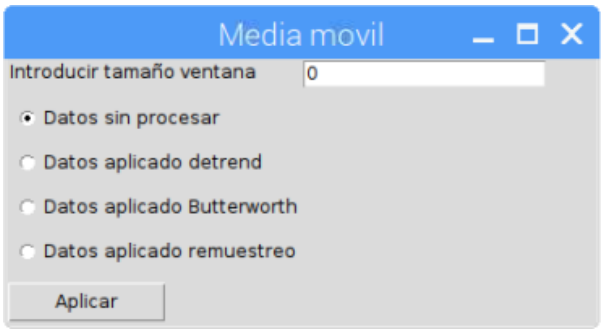

Figura 34. Petición para introducir el tamaño de ventana de la media móvil.

El tamaño de ventana se refiere al número de valores que se utilizan para hacer la media y calcular el nuevo valor. Con un valor de 1 la señal filtrada es igual que la introducida. Si el valor del tamaño de ventana es muy grande se desvirtúan los datos y se pierden los datos leídos. Los valores de tamaño de ventana correcto, que suaviza la señal registrada, sin perder información puede oscilar entre 3-10. Este valor puede ser mayor a mayores frecuencias de muestreo, debido a la menor equidistancia entre muestras.

Los n primeros valores de la media móvil se eliminan, empezando la señal en la muestra en el número n, a partir de la cual se hace el promedio de los n valores anteriores incluyendo. Para el cálculo de cada valor de la media móvil se utiliza la ecuación (5).

$$\bar{x}_i = \frac{x_{i-(n-1)} + \cdots + x_{i-1} + x_i}{n} \quad (5)$$

Estos datos se guardan en el programa para su uso posterior si se desea por el usuario y se muestran en una gráfica emergente, para que se puedan comparar con las señales en bruto como se muestra en la parte inferior de la figura 35. En ella también se comparan con los datos sin filtrar, apreciándose la disminución de ruido de las señales.

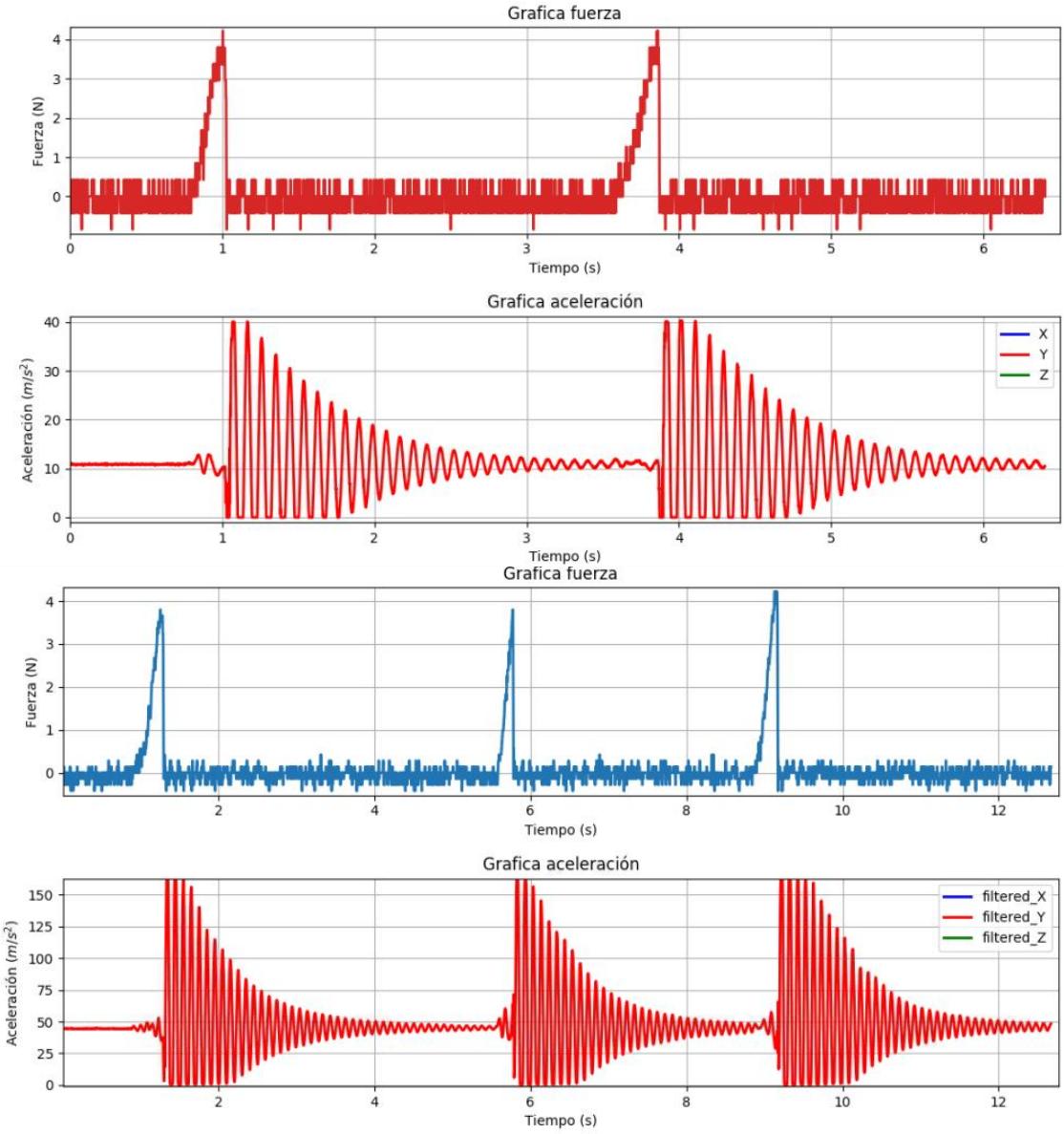

Figura 35. Datos filtrados con media móvil inferior vs datos sin procesar

Filtro Detrend

Se utiliza para eliminar la tendencia lineal del ensayo. Este filtro es útil cuando no se ha realizado el tarado de los sensores antes de realizar la medición o se ha tarado anteriormente sobre unas condiciones que han cambiado. Se ejecuta mediante el botón "Filtro Detrend" y solo es necesario elegir los datos de partida para realizar el filtro, con o sin filtrado previo. Se generan los datos y se muestra la gráfica con los resultados. Un ejemplo de cómo actúa este filtro se puede ver en la figura 36, donde se aprecia que la aceleración media de aproximadamente 10 m/s^2 se elimina.

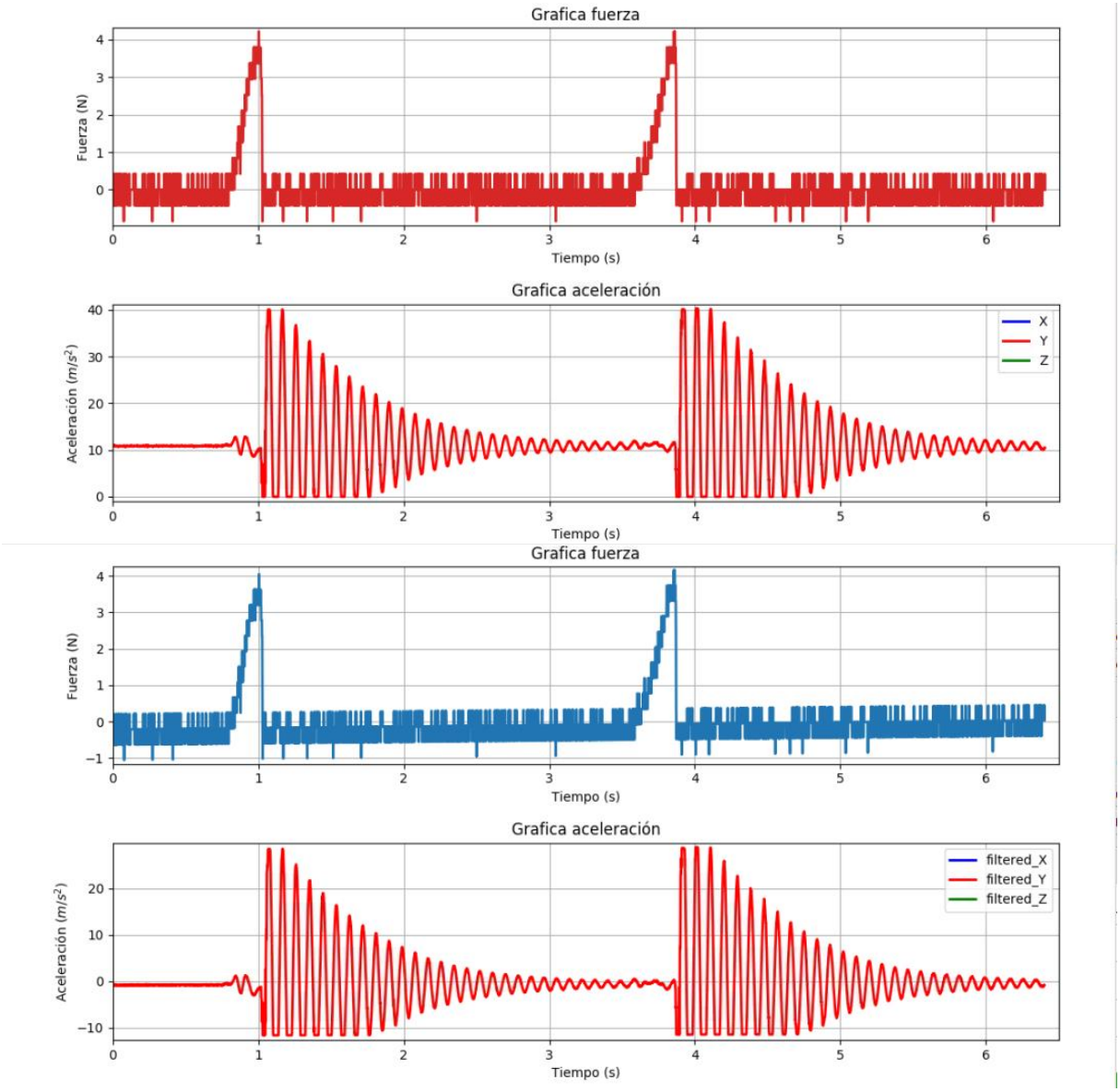

Figura 36. Datos filtrados con detrend inferior vs datos sin procesar

Filtro Butterworth

Se ha añadido un filtro digital Butterworth de paso bajo, que permite el paso de las frecuencias más bajas y atenúa las frecuencias más altas. Se ha añadido este filtro porque la salida se mantiene constante casi hasta la frecuencia de corte y no cambia en función del orden seleccionado, solo cambia la pendiente de atenuación. Esto evita que se altere la señal hasta casi la frecuencia de corte. Como inconveniente de este filtro respecto de otros, como el de Chebyschev o el de Cauer, es que su zona de transición es mayor.

Para el cálculo del filtro digital Butterworth es necesario que el usuario introduzca **el orden del filtro, que debe ser un número entero positivo, y la frecuencia de corte, que debe ser menor que la mitad de la frecuencia de muestreo,** como aparece en la figura 37. También se puede utilizar los datos de filtros previos aplicados. Si la frecuencia de corte está fuera de los valores establecido aparece un mensaje de información diciendo que no se han introducido correctamente los parámetros. La respuesta decae linealmente desde la frecuencia de corte hacia menos infinito. Está pendiente es en función del orden del filtro, siendo la pendiente mayor para ordenes mayores, por lo que la atenuación a partir de la frecuencia de corte es más rápida. Un filtro de orden 1, tiene una pendiente de -20 dB por

década, en un eje logarítmico, a partir de la frecuencia de corte. Para un filtro mayor de orden n, la pendiente es -20n dB.

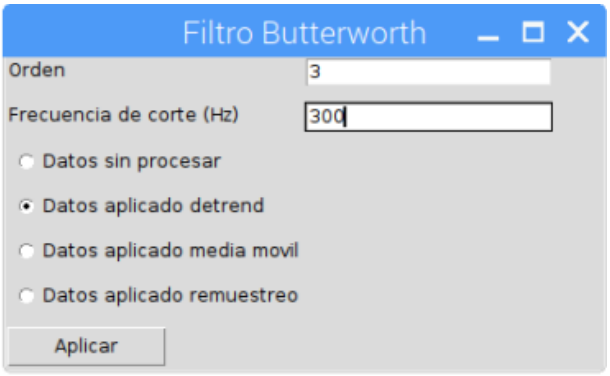

Figura 37. Introducción de parámetros de filtro Butterworth

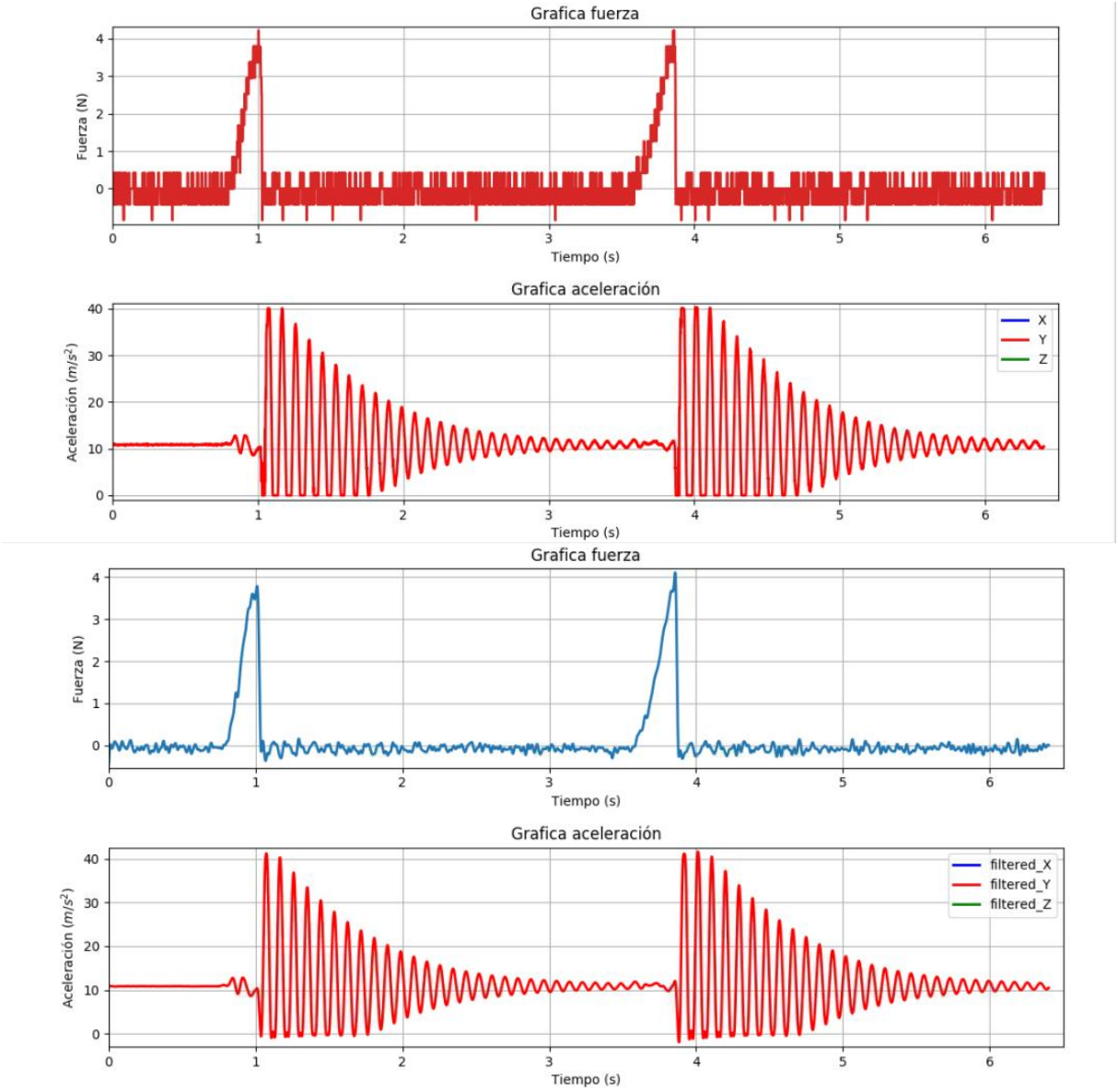

Figura 38. Señales filtradas filtro Butterworth

Los datos filtrados se guardan en el programa para su uso posterior si fuera deseado por el usuario y se muestran en una gráfica emergente, las señales filtradas para que se

puedan comparar con las señales en bruto. En la figura 38, se comparan las señales filtradas con los datos muestreados, apreciándose la disminución de ruido de las señales, principalmente en la fuerza.

Remuestreo

Está función permite volver a muestrear un número de puntos de la señal leída utilizando el método de Fourier. Es necesario introducir el número de muestras de la señal filtrada y los datos de partida sobre los que se va a ejecutar el remuestreo, figura 39. **Siempre se debe introducir un número entero positivo.**

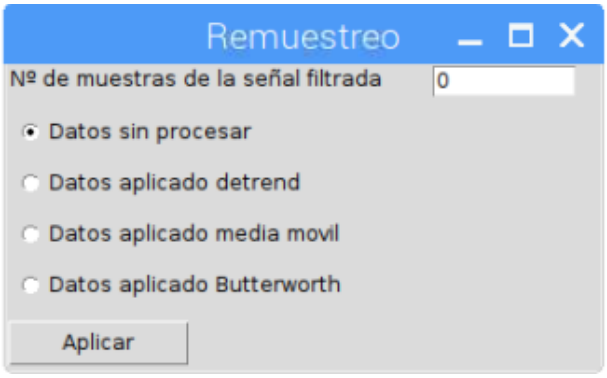

Figura 39. Parámetros remuestreo

La señal empieza con el mismo valor, pero se realiza la muestra con un espacio que se calcula con la ecuación (6), por tanto, si se desea obtener la muestra con una mayor frecuencia se aumenta el número de puntos, disminuyendo con ello el espacio entre cada muestra.

$$espaciado\ nuevo = \frac{espacido \times nº\ puntos\ muestras}{nº\ puntos\ muestras\ nuevo} \qquad (6)$$

Se ha incluido está función por la dificultad de añadir la funcionalidad de cambiar la frecuencia de muestreo por el usuario antes de la medición. Esto se debe a la no linealidad temporal del conversor analógico digital, que impide que se puede obtener un modelo matemático que ajuste automáticamente la frecuencia de muestreo. Además con esta función se pueden obtener mayores frecuencias que las obtenidas a máxima frecuencia de lectura.

Función de respuesta en frecuencia (FRF)

La función de respuesta en frecuencia (FRF) se usa para describir la relación entre la entrada y salida de un sistema, en este caso, sirve para describir la relación entre la fuerza aplicada, leída con la célula de carga, y la respuesta obtenida medida con el acelerómetro. Con el cálculo de la FRF se pueden obtener las frecuencias modales observando los picos.

Se calcula la respuesta en frecuencia del sistema introduciendo los parámetros de la función, como se puede ver en la Figura 40.

- El término “fs” o frecuencia de muestreo se iguala automaticamente con la frecuencia a la que se ha realizado la lectura.
- El término window (tipo de ventana), es la forma de dividir la señal registrada en segmentos. Se puede introducir algunos de los codigos que se muestran en la tabla 15, siendo “M” el numero de puntos de la ventana de salida o puede ser una lista.
- El número de superposición entre segmentos consecutivos. El valor introducido debe ser menor que la longitud del segmento. Si se deja en blanco la casilla se

toma el valor por defecto, que es el 50% de los puntos que forman el segmento. Se debe introducir un valor entero positivo o dejar en blanco.

- Longitud de la FFT (Fast Fourier Transform o en español Trasnformada Rápida de Fourier) utilizada. Si se deja en blanco la casilla se toma el valor por defecto, que es igual al número de puntos de cada segmento. Se debe introducir un valor entero positivo o dejar en blanco.
- Por ultimo, antes de pulsar el boton realizar FRF, se seleccionan los datos para realizar el calculo de la FRF, con o sin filtrado y el eje del acelerometro sobre el que se quiere realizar. Para utilizar señales filtradas es necesario haber realizado el filtrado previamente. La señal de entrada para el calculo de la FRF es siempre la fuerza registrada y la señal de respuesta es la aceleración en el eje seleccionado.

Tabla 15. Tipos de ventana. (The SciPy community, 2019)

Codigo
signal.boxcar(M)
signal.triang(M)
signal.blackman(M)
signal.hamming(M)
signal.hann(M)
signal.bartlett(M)
signal.flattop(M)
signal.parzen(M)
signal.bohman(M)
signal.blackmanharris(M)
signal.nuttall(M)
signal.barthann(M)
signal.kaiser (M, beta)
signal.gaussian(M,std)
signal.general_gaussian(M,p,sig)
signal.slepian(M,width)
signal.dpss(M,NW)
signal.chebwin(M,at)
signal.exponential(M,tau)
signal.tukey(M, alpha)

Figura 40. Parámetros cálculo FRF.

Se utiliza el método Welch para el cálculo de la FRF. Primero se calcula la densidad espectral de potencia "Pxx" mediante la función *"scipy signal.welch"* que solo evalúa la señal de entrada del sistema, en este caso la fuerza. En segundo término se calcula la densidad espectral de la potencia cruzada "Pxy" mediante la función *"scipy.signal.csd"*, que utiliza la señal de entrada en el sistema, fuerza, y la señal de respuesta, aceleración. Una vez calculados estos términos se obtiene la función de respuesta en frecuencia a través de la ecuación (7):

$$FRF = \frac{Pyx}{Pxx} \tag{7}$$

Los datos se muestran en una ventana emergente con la información de los parámetros seleccionados, por si se generan varias con distintos parámetros, y las gráficas de la densidad espectral de potencia "Pxx", la densidad espectral de potencia cruzada "Pxy" y por último la FRF. En la figura 41, se muestra un ejemplo de una FRF entre una entrada, la fuerza, y una salida, la aceleración en el eje Z. En el caso de que la selección de la señal de entrada no se haya generado, no se haya realizado el filtro antes de la FRF, las gráficas aparecen vacías.

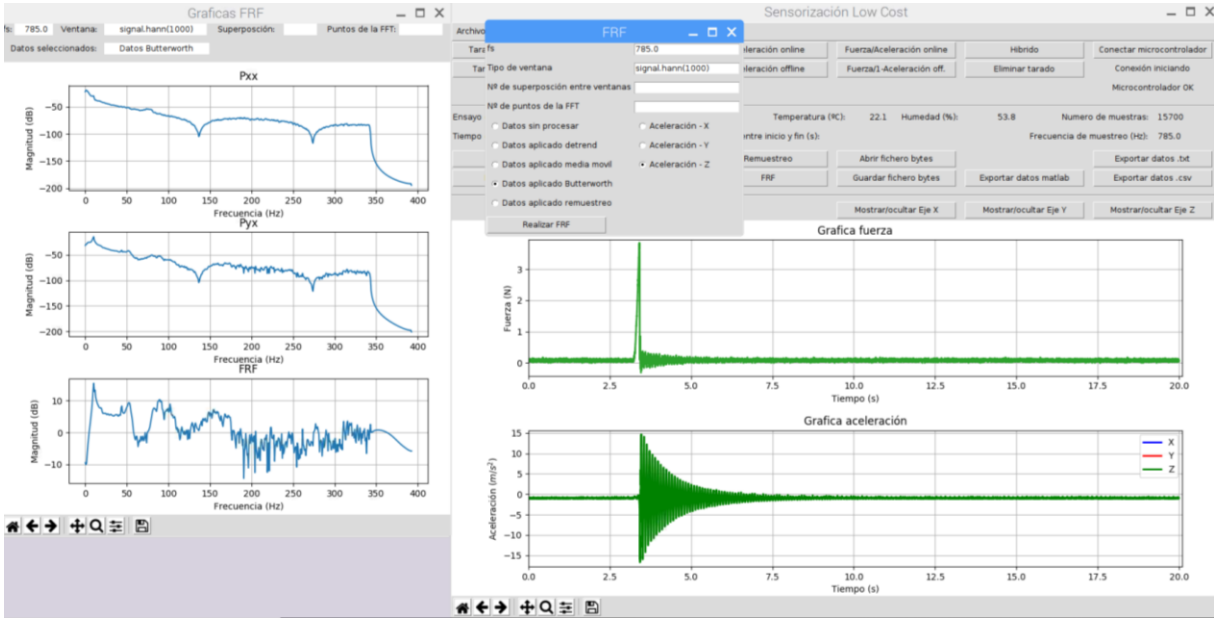

Figura 41. FRF obtenida.

Abrir

Permite abrir un archivo de bytes, generado al utilizar el modo offline, híbrido o la función guardado, que contiene al final de su nombre "_b". Solo se cargan las señales en bruto como se reciben del microcontrolador para realizar su procesado. Aparte de las señales muestreadas, carga en el interfaz, la temperatura, humedad, el número de muestras y la frecuencia a la que se realizó el muestreo. La selección del archivo se realiza como se ve en la figura 42.

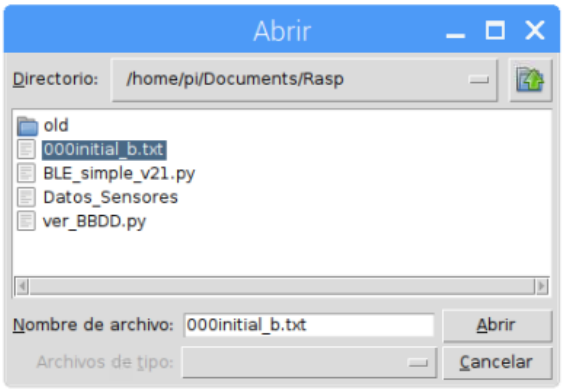

Figura 42. Abrir archivo

Guardar

Permite guardar un archivo de bytes con las señales en bruto como se reciben del microcontrolador para realizar su procesado. Aparte de las señales muestreadas guarda la temperatura, humedad y la frecuencia a la que se realizó el muestreo. Esta función se ha añadido para guardar los datos leídos en modo online en un fichero para poder realizar posteriormente algún procesado con el programa, ya que en este modo los valores no quedan guardados según se reciben. **Se genera un fichero .txt que lleva incluido "_b" al final del nombre** introducido por el usuario para identificar que contiene bytes, igual que los ficheros que se generan en modo offline.

Exportar datos .txt

Se pueden guardar los datos en un archivo de texto simple. En la primera línea se guardan las condiciones en las que se tomaron las muestras, en la segunda los nombres de cada

variable separados por comas y en las líneas posteriores se guardan los valores de cada señal separado también por comas. En la figura 43, se muestra un ejemplo de cómo quedan los datos guardados tras la exportación en este formato.

Se guardan todos los valores de las señales registradas y de las señales filtradas, que se hayan ejecutado anteriormente.

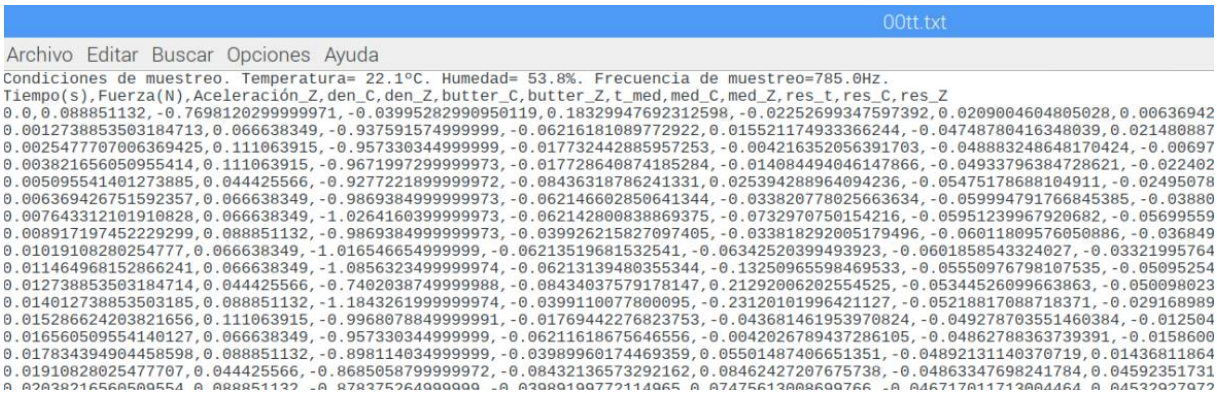

Figura 43. Ejemplo exportación .txt

Exportar datos .csv

También se ha integrado la posibilidad de guardar los datos en un archivo .csv (valores separados por comas). La exportación es similar a la del archivo .txt, pero en este caso el separador de valores es el espacio. También en la primera fila aparecen reflejado la temperatura, humedad y la frecuencia. En la Figura 44, se muestra un ejemplo de un fichero .csv que se está importando a LibreOffice Calc, donde se refleja los datos exportados y cuál es su formato.

Se guardan todos los valores de las señales registradas y de las señales filtradas, que se hayan ejecutado anteriormente.

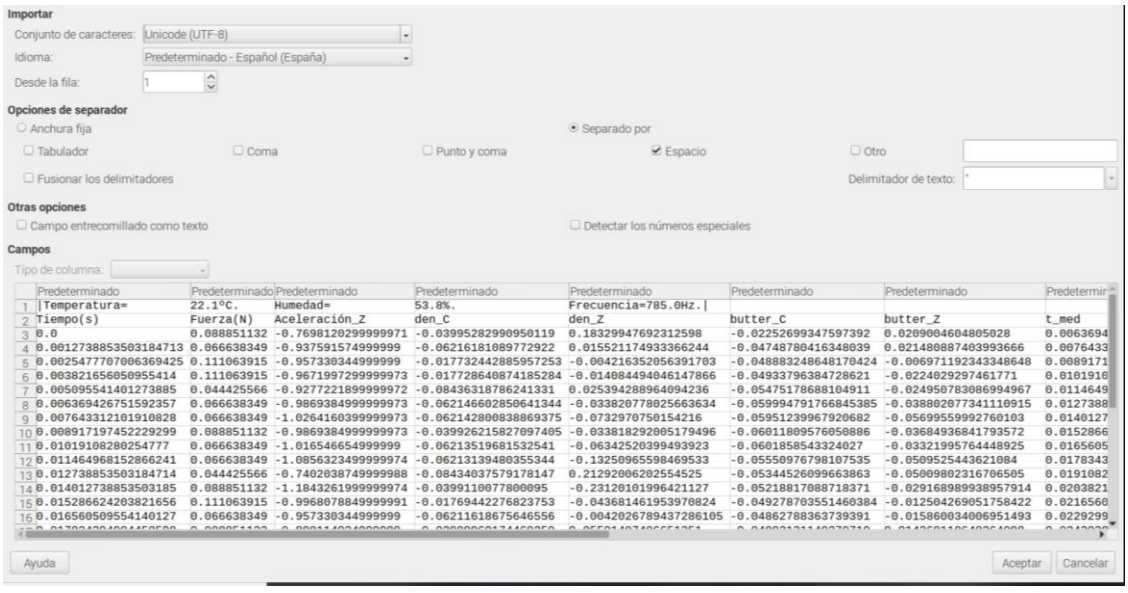

Figura 44. Ejemplo exportación .csv

Exportar datos Matlab

Guarda los ficheros en un fichero de texto, pero está vez los construye como vectores de Matlab para luego proceder a su procesado en este programa si se desea por parte del usuario. Se añaden en la primera línea como un comentario la temperatura, la humedad y la frecuencia. Se muestra un ejemplo de este formato de exportación en la figura 45.

Se guardan todos los valores de las señales registradas y de las señales filtradas, que se hayan ejecutado anteriormente.

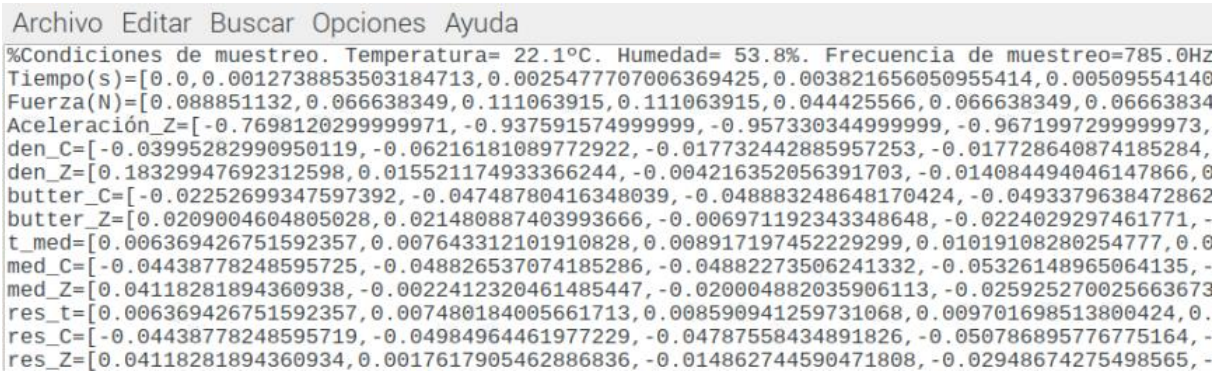

Figura 45. Ejemplo de exportación de vectores con formato Matlab

Visualizar gráficas

Todas las gráficas tienen funciones para modificar la visualización, con los botones que aparecen debajo de ellas. Se puede apreciar el uso de una ventana de zoom en la figura 46, así como el resto de los botones de los que se disponen. Estos botones de izquierda a derecha son:

- Volver a la vista inicial.
- Retroceder a la vista anterior.
- Avanzar a la vista siguiente.
- Mover los ejes de la vista.
- Hacer zoom con una ventana.
- Configurar las gráficas
- Guardar la imagen.

Aparte de estas funciones se disponen de unos botones en la parte superior de las gráficas de la interfaz gráfica para mostrar u ocultar las líneas de las señales muestreadas de los ejes del acelerómetro.

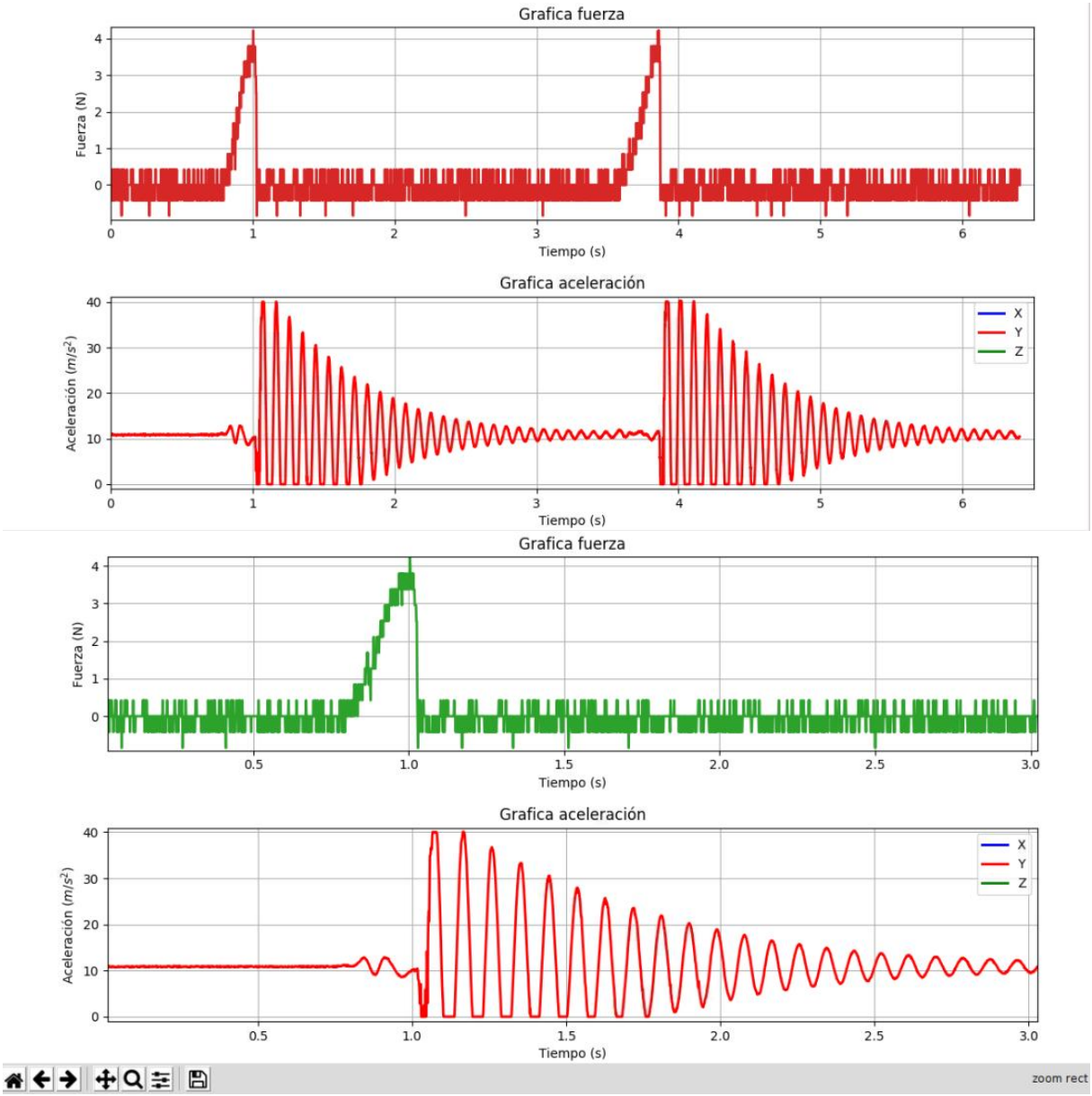

Figura 46. Ventana de zoom sobre la gráfica, con los botones de modificación en la parte inferior.

3.1.2 Desarrollo

Para el desarrollo del programa ha sido necesario instalar con los siguientes comandos todas las librerías utilizadas en la Raspberry Pi, mediante el uso del terminal.

- Sudo pip3 install numpy
- Sudo pip3 install bluepy
- Sudo apt-get install python3-pip libglib2.0-dev
- Sudo pip3 matplotlib
- python -m pip install --user scipy ipython sympy nose
- Sudo pip3 install scipy
- Sudo pip3 install ipython
- Sudo pip3 install sympy
- Sudo pip3 install nose

Nota importante es lo del pip3, pues el programa se ha desarrollado en Python 3, y si no se incluye el 3 y se utiliza solo el comando pip, se instalan para Python 2.

Bluepy

La librería Bluepy, (Ian Harvey, 2020), se utiliza para la realización de la comunicación bluetooth con el microcontrolador.

Las funciones principales de esta librería son:

- *peripheral=btle.Peripheral(mac)*
  - Permite la conexión con otro modulo bluetooth, para ello es necesario introducir la dirección MAC.
- *service=peripheral.getServiceByUUID("0000ffe0-0000-1000-8000-00805f9b34fb")*
  - *Crea el servicio para crear la característica.*
- *characteristic=service.getCharacteristics()[0]*
  - *Representa un elemento de datos cortos que se pueden leer y describir*
- *peripheral.withDelegate(sampleDelegate())*
  - Permite la comunicación bluetooth mediante notificaciones.
- *characteristic.write(datos)*
  - Envía datos por bluetooth.
- *peripheral.waitForNotifications(timeout*)
  - Espera a recibir una notificación del dispositivo y ejecuta la función handleNotification para recibir el dato.
- *class sampleDelegate(btle.DefaultDelegate):*
  def __init__(self):
  btle.DefaultDelegate.__init__(self)
  def handleNotification(self, cHandle, data):
  - Es la función que recibe los datos mediante bluetooth.

Scipy signal

Esta librería, (SciPy developers, 2020), permite realizar el procesado de las señales muestreadas del ensayo, para poder realizar los análisis del comportamiento de la estructura.

Filtro Detrend

El filtro Detrend elimina la tendencia linear a lo largo de un eje de los datos introducidos

- *scipy.signal.detrend(data, axis=-1, type='linear', bp=0, overwrite_data=False)*
  La respuesta:
  - ret → ndarray
    - Datos filtrados

  Los parámetros de las funciones son iguales para ambas funciones:
  - data → Array
    - Los datos de entrada.
  - axis → int, opcional
    - Eje a lo largo del cual se calcula la tendencia de los datos; el valor predeterminado es sobre el último eje (es decir, eje = -1).
      - '-1', no se define.
  - Type → 'linear', 'constant', opcional
    - El tipo de filtrado, Si es 'linear', se elimina el resultado de un ajuste lineal de mínimos cuadrados. Si es 'constant', solo se resta la media de los datos.
      - 'linear', para eliminar la tendencia linear

- bp → array_like of ints, opcional
  - Secuencia de puntos de interrupción. Si se define, se realiza un ajuste lineal para cada parte de los datos entre los puntos de interrupción. Se especifican como índices en los datos de entrada.
    - 0, no se define.
- overwrite_data → bool, optional
  - Si es 'True', los datos filtrados se sobrescriben y no hacen una copia. Por defecto es 'False'.
    - 'False', se hace una copia en otro array, para conservar los valores iniciales.

Se filtran todos los datos obtenidos durante el muestreo.

Filtro Butterworth

Diseño de un filtro Butterworth digital y analógico y devuelve los coeficientes de este filtro.

- *scipy.signal.butter(N, Wn, btype='low', analog=False, output='ba', fs=None)*
  La respuesta:
  - b, a → ndarray, ndarray
    - b, numerador y a, denominador del filtro IIR (Respuesta infinita al impulso)
  - z, p → kndarray, ndarray, float
    - Ceros, polos y ganancia del sistema de la función de transferencia del filtro IIR. Es la respuesta si output='zpk'.
  - Sos → ndarray
    - Representación de secciones de segundo orden del filtro IIR. Es la respuesta si output='sos'. Este es método selecionado

  Los parámetros de las funciones son iguales para ambas funciones:
  - N →int
    - El orden del filtro
      - Seleccionable por el usuario en la interfaz gráfica.
  - Wn→array_like
    - La frecuencia o frecuencias críticas. Por filtros paso bajo y paso alto, Wn es un escalar; para filtros paso banda y elimina banda, Wn es una secuencia de longitud 2.
    - Para un filtro Butterworth, este es el punto en el cual la ganancia cae a 1/raíz(2) de la banda de paso ("el punto -3dB")
    - Para filtros digitales, Wn está en las mismas unidades que fs. Por defecto, fs es 2 semiciclos/muestra, por lo que se normalizan de 0 a 1, donde 1 es la frecuencia Nyquist. (Por lo tanto, Wn está en semiciclos muestra).
    - Para filtros analógicos, Wn está en frecuencias angulares, por ejemplo, rad/s.
      - Seleccionable por el usuario en la interfaz gráfica.
  - btype→'lowpass', 'highpass', 'bandpass', 'bandstop', opcional
    - El tipo de filtros, por defecto es 'lowpass'.
      - Seleccionable por el usuario en la interfaz gráfica.
  - analog→bool, opcional
    - Devuelve un filtro analógico cuando es 'True'; de lo contrario devuelve un filtro digital.
      - 'False', se realizan filtros digitales.
  - output → 'ba', 'zpk', 'sos', opcional

        - Tipo de salida: 'ba', numerador/denominador; 'zpk', polos y ceros o 'sos' secciones de segundo orden. Por defecto es 'ba' para compatibilidad con versiones anteriores, pero 'sos' se debe usar para filtrado de propósito general.
            - 'sos' se recomiendo su uso para propósito general.
    - fs → float, opcional
        - La frecuencia de muestreo del sistema.
            - 'None', por defecto. No se modifica

Filtro Resample

Se vuelve a muestrear la señal utilizando el método de Fourier, que supone que la señal es periódica, a lo largo del eje dado.

- *scipy.signal.resample(x, num, t=None, axis=0, window=None)*
  La respuesta:
    - resampled_x or (resampled_x, resampled_t)
        - Datos filtrados y si se da t, una tupla que contiene los datos filtrados y su correspondiente posición.

  Los parámetros de las funciones son iguales para ambas funciones:
    - x → Array
        - Los datos de entrada
    - num → int
        - El número de muestras en la señal filtrada.
            - Seleccionable por el usuario en la interfaz gráfica.
    - t → array_like, opcional
        - Si t es dado, se supone que son las posiciones de muestra asociados con los datos de la señal de entrada
            - 'None', por defecto. No se modifica.
    - axis → int, opcional
        - El eje de los datos de entrada que son filtrados. Por defecto es 0.
            - 0, por defecto. No se modifica.
    - window → array_like, callable, string, float, or tuple, opcional
        - Especifica la ventana aplicada a la señal en el dominio de Fourier.
            - 'None', por defecto. No se modifica.

Calculo FRF

Para hacer la FRF de las señales obtenidas se va a utilizar las siguientes dos funciones de scipy.

- *scipy.signal.welch(x, fs=1.0, window='hann', nperseg=None, noverlap=None, nfft=None, detrend='constant', return_one --sided=True, scaling='density', axis=-1, average='mean')*
  La respuesta:
    - f → (ndarray).
        - Array de les frecuencias de muestreo
    - Pxx → (ndarray).
        - Densidad espectral cruzada o espectro de potencia cruzada de x.
- *scipy.signal.csd(x, y, fs=1.0, window='hann', nperseg=None, noverlap=None, nfft=None, detrend='constant', return_onesided=True, scaling='density', axis=-1, average='mean')*
  La respuesta:
    - f → ndarray

        - Array de les frecuencias de muestreo
    - Pxy → ndarray
        - Densidad espectral cruzada o espectro de potencia cruzada de x, y.

Los parámetros de las funciones son iguales para ambas funciones:

- x → Array
    - Señal de entrada.
        - Seleccionable por el usuario en la interfaz gráfica.
- y → Array
    - Señal de respuesta.
        - Seleccionable por el usuario en la interfaz gráfica.
- fs → Float
    - Frecuencia a la que se ha realizado el muestreo.
        - Seleccionable por el usuario en la interfaz gráfica.
- window →str o tupla
    - Tipo de ventana o segmento.
        - Seleccionable por el usuario en la interfaz gráfica.
- nperseg → int
    - Longitud de cada segmento, El valor predeterminado es None, pero si window es str o tupla, se estable en 256 y si window es un array, se establece la longitud de window
        - Seleccionable por el usuario en la interfaz gráfica.
- noverlap → int
    - Número de puntos de superposición entre segmentos, si *None,* noverlap=nperseg/2, siendo este el valor por defecto.
        - Seleccionable por el usuario en la interfaz gráfica.
- nfft → int
    - Longitud de la FFT utilizada, si se desea una FFT rellenada con cero. Si ninguno, la longitud de FFT es nperseg. Por defecto es Ninguno.
        - Seleccionable por el usuario en la interfaz gráfica.
- detrend → str o function o 'False'
    - Especifica cómo reducir la tendencia de cada segmento. Si detrend es una cadena, se pasa como argumento de tipo a la función detrend. Si es una función, toma un segmento y devuelve un segmento sin tendencia. Si la tendencia es falsa, no se realiza ninguna tendencia. El valor predeterminado es "constante".
        - 'None'. Se pueden introducir los valores filtrados.
- return_onesided → bool
    - Si es verdadero, devuelve un espectro unilateral para datos reales. Si es falso, devuelve un espectro de dos lados. El valor predeterminado es Verdadero, pero para datos complejos, siempre se devuelve un espectro de dos lados.
        - 'True', por defecto. No se modifica
- scaling → 'density', 'spectrum'
    - Selecciona entre calcular la densidad espectral cruzada ('densidad') donde Pxy tiene unidades de V ** 2 / Hz y calcular el espectro cruzado ('espectro') donde Pxy tiene unidades de V ** 2, si x e y se miden en V y fs se miden en Hz. El valor predeterminado es "densidad"
        - 'density', para realizar posteriormente la FRF
- axis → int

        - Eje a lo largo del cual se calcula el welch o CSD para ambas entradas; el valor predeterminado es sobre el último eje (es decir, eje = -1).
            - '-1', por defecto. No se modifica
    - average → 'mean','median'
        - Método de promedio si se usan periodogramas. Por defecto se usa la media.
            - 'Mean', por defecto. No se modifica

Matplotlib

La librería Matplotlib, (The Matplotlib development team, 2020), se ha usado para la realización de las gráficas de las señales muestreadas y de su posterior procesamiento.

Las gráficas de los datos recibidos desde el controlador se han embebido en la ventana principal del programa. Estás gráficas se actualizan cada cierto número de datos recibidos para no colapsar el programa y evitar así la posible pérdida de datos durante la comunicación bluetooth. Esto viene provocado por el tiempo de procesado que requiere la actualización la interfaz gráfica.

Para las gráficas embebidas ha sido necesario utilizar el módulo "FigureCanvasTkAgg", para los botones de modificación de la visualización el módulo "NavigationToolbar2Tk" y para todas las gráficas el módulo "Figure" y "plot". En este caso todas las figuras se componen de varias gráficas, normalmente, la gráfica de carga y la de aceleración excepto la de la FRF.

Numpy

Numpy, (NumPy, 2020),permite la realización de operaciones matemáticas. Se ha usado para obtener los máximos y los mínimos de las gráficas y así poder realizar el escalado automático. Para modificar los ejes de la FRF y las densidades espectrales de potencia y poderlas poner con un eje logarítmico. También se utiliza para poder realizar las raíces cuadradas en el cálculo de las RMS.

Tkinter

La librería Tkinter, (Python Software Foundation, 2020), se ha usado para la realización de la interfaz gráfica. Aparte de la ventana principal, se ha realizado de la misma manera las ventanas emergentes. Las ventanas de mensajes ha sido necesario utilizar el módulo "messagebox" y para las ventanas de guardado y apertura de ficheros el módulo "filedialog".

3.2 Arduino IDE

El entorno de desarrollo integrado de Arduino es la aplicación seleccionada para el desarrollo y carga de los programas para los microcontroladores de Arduino. Admite la programación en lenguajes C y C++ y hay una gran multitud de librerías, gracias a ser open source y a una gran comunidad de trabajo que hay detrás de ella. El entorno de desarrollo es como se muestra en la figura 47.

Figura 47. Arduino IDE

3.2.1 Funcionalidades

El papel principal que debe realizar el programa del microcontrolador es leer las variables físicas que se deseen y enviarlas al ordenador principal para su procesamiento. Para ello cumple las siguientes funcionalidades:

- Recibir órdenes mediante bluetooth de la Raspberry Pi.
- Leer datos de los sensores indicados por el usuario: aceleraciones, cargas o ambas.
- Modo de funcionamiento online, enviando el dato tras la lectura, para ver las cargas y las aceleraciones a tiempo real.
- Modo de funcionamiento offline. Se lee un paquete de datos y posteriormente se envía. Esto permite obtener los datos a la máxima frecuencia posible.
- Modo de funcionamiento híbrido. Se almacenan pequeños paquetes de datos que se envían cada cierto tiempo. La lectura de datos es síncrona, pues se espacia entre cada lectura el tiempo que tarda en mandar el paquete de datos.
- Enviar los datos muestreados por bluetooth.
- Comunicación del estado del microcontrolador a la RPI, para conocer si ha surgido algún problema o el experimento se ha realizado correctamente.
- Leer datos y enviar de temperatura y humedad a la que se han realizado las mediciones.
- Evitar que se sature la memoria de la placa.
- Calcular la frecuencia de lectura de cada sensor.

El programa de Arduino sigue su diagrama de flujo como se ve en la figura 48, en él se distingue los dos modos de funcionamiento principales: lectura y envío simultáneo o lectura del ensayo completa y su posterior envío. En este diagrama no se ha distinguido el tipo de sensor que se está leyendo, pues el funcionamiento es el mismo, a excepción de la función de lectura.

En el funcionamiento online lee unitariamente todos los sensores, envía estos datos y hace una pausa de 15 milisegundos, para que la RPI, pueda guardarlos y graficarlos correctamente sin perder datos. Esta pausa se ha establecido de modo experimental, hasta conseguir no perder ningún dato durante la comunicación bluetooth. Debido a esta

pausa, la frecuencia de muestreo disminuye hasta 62 Hz. El motivo causante es el tiempo que se mantiene leyendo el módulo bluetooth de la Raspberry Pi y que si se envían los datos sin espera, provoca que lea todos los datos a la vez, saturándose la entrada durante el proceso de guardado de datos en el programa de Python.

En el modo offline, se hace toda la lectura de todos los sensores seleccionados a máxima frecuencia, mientras se van almacenando en la memoria del microcontrolador, para posteriormente enviarlos por paquetes de 256 datos máximos dejando una pequeña pausa entre paquete y paquete para evitar la saturación de la comunicación. Para este último modo se evalua la memoria disponible para evitar la saturación, antes de empezar la lectura.

En el modo híbrido se realizan varios ciclos de lectura formando pequeños paquetes de datos y que se van enviando cada cierto número de ciclos, aunque el funcionamiento es similar al modo offline, por eso no se ha distinguido en el diagrama. La lectura de datos es síncrona, pues entre cada ciclo, se deja el mismo tiempo de espera que se tarda en enviar el paquete de datos, consiguiendo con ello tener todas las medidas equidistantes.

Aparte de estas funcionalidades, se envían los datos para tarar los sensores y código para la comprobación de la conexión bluetooth.

En el programa se han añadido código que envía a la Raspberry Pi, para que esta conozca cual es el estado del microcontrolador y pueda actuar en consecuencia. También se envían la misma serie de datos para las dos funcionalidades, los códigos de estado, la ganancia del ADS, la frecuencia, la temperatura y la humedad.

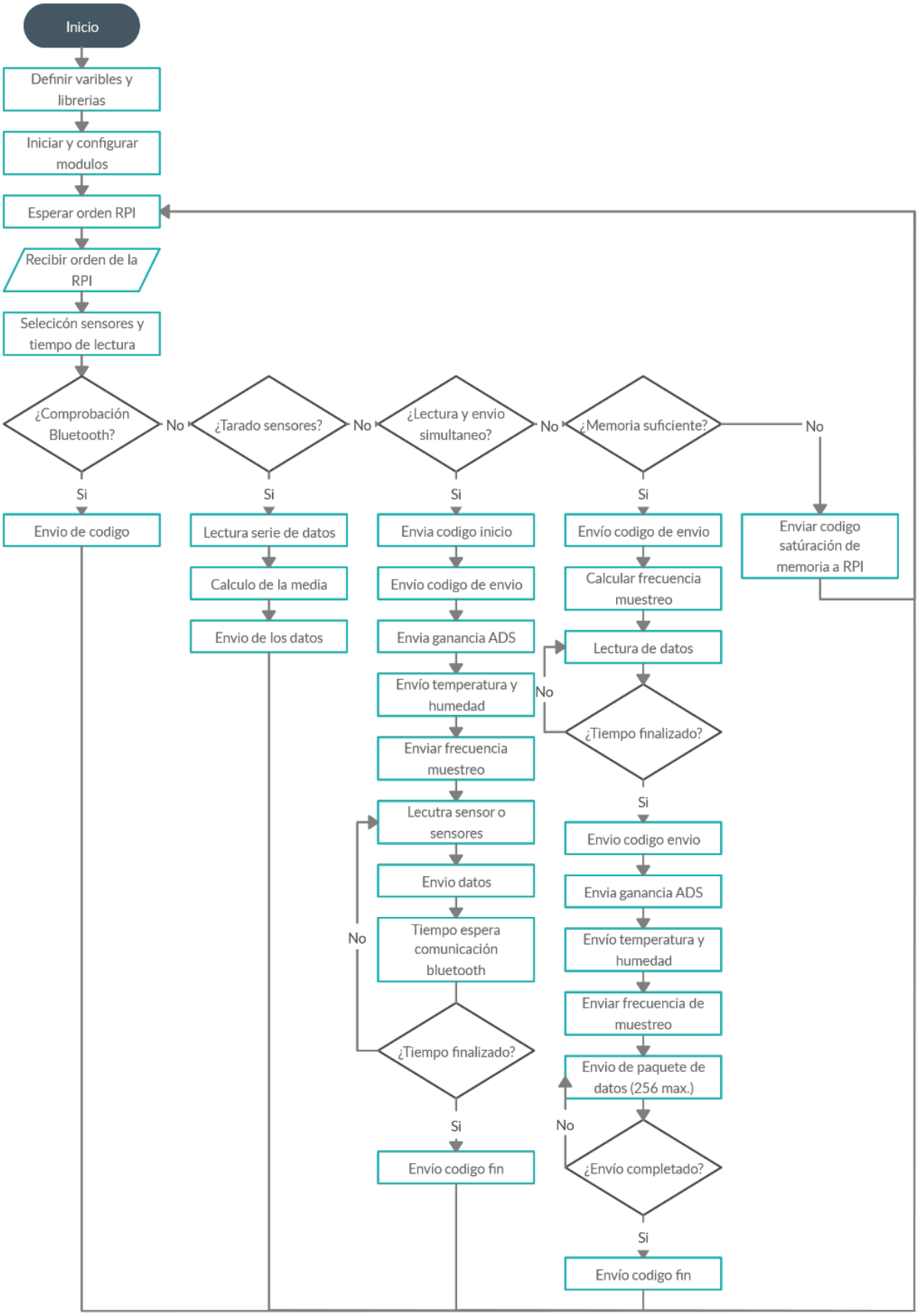

Figura 48. Diagrama de flujo microcontrolador

Un dato importante que cambia la programación en Arduino Due respecto de los modelos anteriores es que la variable tipo “int” ocupa 32 bits en lugar de los 16 bits. La tivaC actúa como el Arduino Due no siendo necesario cambiar el programa. Para enviar valores tipo int de 16 bits y así poder mandar solo 2 bytes en la comunicación por bluetooth y aprovechar mejor también la memoria de los microcontroladores se utiliza la variable “int16_t”.

En relación con el uso de memoria el programa evalúa la cantidad de memoria disponible antes de proceder a guardar datos masivamente, para evitar la saturación de la memoria RAM. De esta forma se consigue evitar el colapso del microcontrolador y de todo el sistema, pues es necesario un reinicio para que todo volviera a funcionar correctamente. En el caso que el tamaño del muestreo sea mayor que la capacidad de memoria disponible, el microcontrolador envía un código de saturación a la Raspberry Pi, para informar de que no se puede ejecutar la orden y envía cuál es el tamaño de muestreo máximo disponible.

Tanto los datos leídos del acelerómetro y de la célula de carga son de 16 bits o 2 bytes, siendo enviados mediante bytes separados byte por byte. Para el caso de modo online, el valor se separa en un byte con las 8 cifras más significativas y en otro con las 8 cifras menos significativas y se envían cada una. Para modo offline se envían por paquetes de bytes, siendo está separación de cada byte interna del programa de Arduino. Se ha limitado a 256 paquetes cada vez, para obtener un correcto funcionamiento. El modo híbrido funciona como el modo offline, pero con paquetes más pequeños, de 80 datos en el caso de Arduino. Las funciones usadas para estos envíos se recogen en la tabla 16. Se utiliza la función Serial.write(), porque lo datos se envían más rápido que con la función Seria.print(), además que permite el envío de paquetes de datos, aunque el inconveniente es trabajar con byte, que es necesario codificar antes de enviar y decodificar posteriormente en la Raspberry Pi.

Tabla 16. Envió de datos de Arduino

Código	Función	Modo
Serial1.write(lowByte(data));	Escribe byte menos significativo	Online
Serial1.write(highByte(data));	Escribe byte más significativo	
Serial1.write((byte*)aux, h*sizeof(aux[0]));	Envía un array byte a byte, el primer término es el array y el segundo el buffer.	Offline/ híbrido

Para el resto de los datos enviados por Bluetooth se utiliza la función Serial.print(), porque no es tan necesario la velocidad, ya que no son envíos masivos. Mediante esta función se envían los datos climatológicos (temperatura y humedad), la ganancia del ADC y la frecuencia, debido a que se envían 4 bytes y se complica la comunicación, sin tener casi ganancia de tiempo. Aparte de estos datos se envían códigos del estado del controlador, para saber si ha iniciado el muestreo, ha terminado el muestreo y comienza a enviar datos, o ha concluido la lectura y envío.

3.2.2 Desarrollo

En este programa no se realiza la conversión de unidades de los valores leídos por los sensores ni se ajusta el offset, ya que se realizan en la Raspberry Pi, pues tiene mucha mayor potencia de procesado y así no se limita la de los microcontroladores.

Ha sido necesario incluir dos librerías externas al programa que se presentan a continuación. Para ello se han descargado los ficheros open source de github y se han incluido mediante un fichero .rar en el program a traves de incluir librerías y añadir bibliotecas .zip. Posteriormente se han añadido estás librerías al codigo del programa en la misma sección pulsando sobre su nombre como se ve en la figura 49.

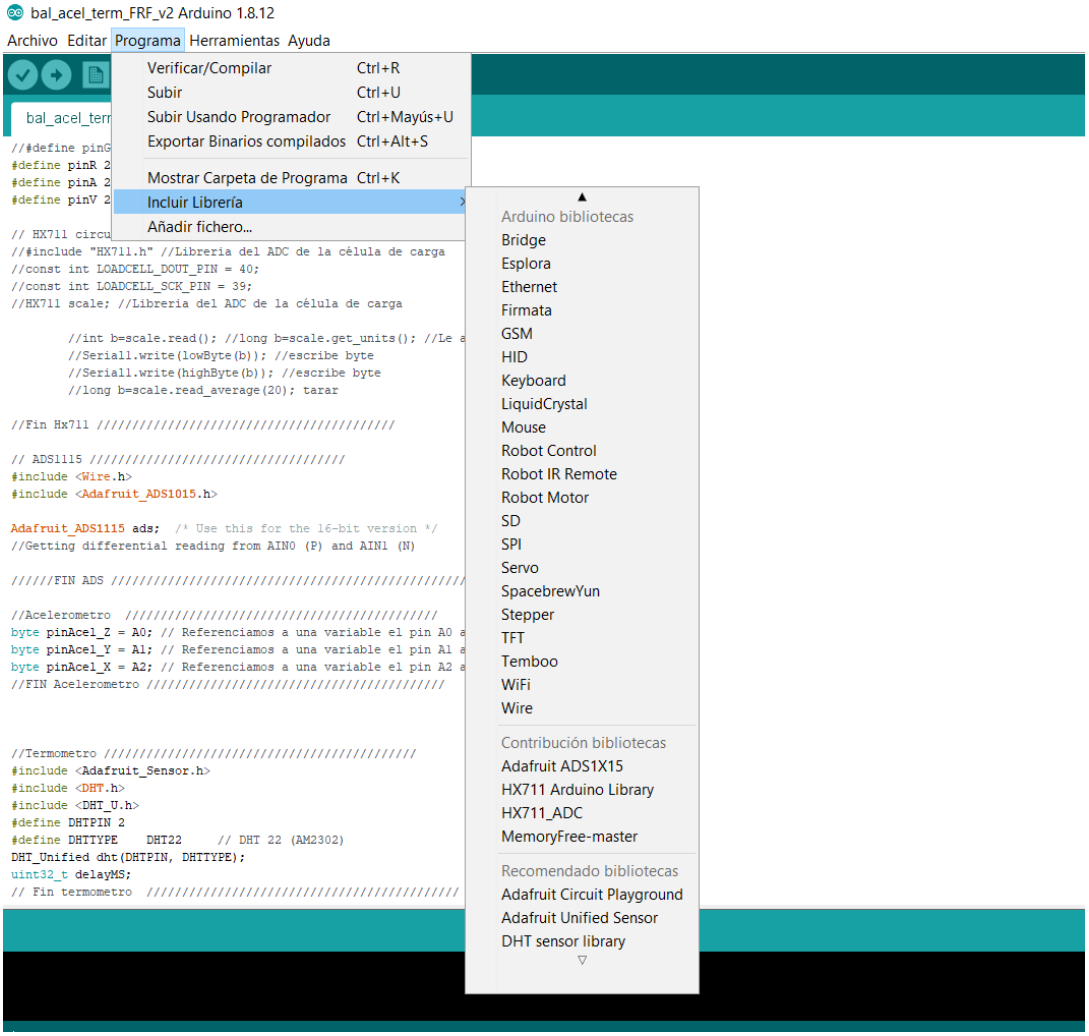

Figura 49. Librerías Arduino.

Adafruit_ADS1X15

Para la lectura de los datos del sensor ADS1115, también sirve para el sensor ADS1015, se ha utilizado la librería Adafruit_ADS1X15, disponible en: https://github.com/soligen2010/Adafruit_ADS1X15. En ella se recogen las funciones, para que la configuración y lectura del ADC, sea mucho más sencilla. Se han utilizado las funciones de esta librería que se recogen en la tabla 17 con su comentario como explicación de su uso.

Tabla 17. Funciones Adafruit ADS1115

Funciones utilizadas de la librería Adafruit_ADS1X15
#include <Adafruit_ADS1015.h> //Librería del ADC Adafruit_ADS1115 ads; //Configura uso de esta librería con ADS1115
ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V 1 bit = 0.0078125mV 0625mV Establecer ganancia

```
ads.begin();//iniciar ADS1115.
ads.setSPS(ADS1115_DR_860SPS); //Modificar la frecuencia del ADS1115.
ads.readADC_Differential_0_1(); //Lectura diferencial entre el canal 0 y el 1.
```

La ganancia admitida por esta librería está recogida en la tabla 18.

Tabla 18. Ganancia ADS1115

Código	Ganancia	Rango	Resolución
GAIN_TWOTHIRDS	2/3x	±6.144V	0.1875mV
GAIN_ONE	1x	±4.096V	0.125mV
GAIN_TWO	2x	±2.048V	0.0625mV
GAIN_FOUR	4x	±1.024V	0.03125mV
GAIN_EIGHT	8x	±0.512V	0.015625mV
GAIN_SIXTEEN	16x	±0.256V	0.0078125mV

Se ha fijado “GAIN SIXTEEN”, debido a que el rango de voltaje de la célula de carga es ±10mV y así poder tener la mayor precisión posible, sin que se pierda rango de medida. Si se cambia la célula de carga por una que soporte mayores esfuerzos y tenga otro rango de voltaje de medida, es necesario cambiar la ganancia.

Aunque no sea propia de esta librería, hay que añadir la librería Wire.h para la comunicación I2C con el conversor analógico digital. Está librería es oficial de Arduino y viene incluida en Arduino IDE de serie. Dentro de esta librería es importante cambiar la frecuencia de reloj que viene por defecto con el comando *Wire.setClock(1600000).* En la tabla 19 se puede observar cómo cambia la frecuencia de lectura del ADC cambiando la frecuencia de reloj. Para la realización de esta medición se ha usado el programa adjuntado en el anexo 2.

Tabla 19. Comparación frecuencia lectura ADS1115 Arduino.

Frecuencia teórica	Frecuencia I2C 100kHz	Frecuencia I2C 1600kHz
8	8	8
16	15	16
32	30	32
64	58	63
128	108	124
250	188	240
475	299	450
860	424	787

DHT sensor

Para mayor facilidad de la lectura del sensor DHT22, se ha utilizado la librería "DHT-sensor" disponible en el siguiente enlace: https://github.com/adafruit/DHT-sensor-library. El uso de está librería es muy sencillo y no necesita de ninguna otra para su funcionamiento como en el caso del ADS1115. Las funciones utilizadas para la lectura de las magnitudes físicas se recogen en la tabla 20.

Tabla 20. Funciones DHT

Funciones utilizadas de la librería DHT-sensor
#include "DHT.h" //Incluye la librería DHT #define DHTPIN 2 //Define el pin digital de la señal #define DHTTYPE DHT22 // Define el modelo DHT 22 (AM2302) DHT dht(DHTPIN, DHTTYPE); // Asigna el pin y el modelo.
dht.begin(); //Inicio DHT22
dht.readTemperature(); //Lee temperatura
dht.readHumidity(); //Lee Humedad

3.3 Energia IDE

Energia IDE es una plataforma que permite la programación de varias tarjetas de Texas Instruments y está basado en Arduino IDE. Se puede observar en la figura 50, que la interfaz del programa es idéntica a la de Arduino IDE, así como su uso y funciones.

Figura 50. Energia IDE

3.3.1 Código

El programa es exactamente el mismo que el desarrollado para la placa de Arduino, solo se producen cambios debido al montaje eléctrico y su correspondiente cambio de pines.

La comparativa entre los cambios para las dos placas se recogen en la tabla 21. Las librerías añadidas en el programa de Arduino también hay que cargarlas en este programa de manera similar.

Tabla 21. Cambios programa Arduino vs Energia

Arduino IDE	Energia IDE
#define pinR 8 //Led Rojo	#define pinR PK_3 //Led Rojo
#define pinV 9 //Led verde	#define pinV PK_2 //Led verde
#define pinA 10 //Led Amarillo	#define pinA PK_1 //Led Amarillo
#define pinG 3// Selecion 6g acelerometro	#define pinG PD_5//Selecion 6g acel.
#define DHTPIN 2 //Define pin DHT22	#define DHTPIN PD_4//Define pin DHT22
Wire.setClock(1600000);//frecuencia I2C	Wire.setClock(3400000); //frecuencia I2C
Serial1; //uso del puerto 1	Serial7; //uso del puerto 7
m=80 //paquete de datos modo híbrido	m=20 //paquete de datos modo híbrido

Para la tarjeta de desarrollo de Texas Instruments es necesario fijar la velocidad de reloj del bus I2C en 3400000Hz, pues solo tiene fijadas las frecuencias 100000Hz, 400000Hz, 1000000Hz y la mencionada anteriormente como máxima. Una vez fijada esta frecuencia se compara con la obtenida por defecto en la tabla 22, como se ha hecho anteriormente en Arduino.

Tabla 22. Frecuencia I2C conseguidas TivaC

Frecuencia teórica	Frecuencia TivaC I2C defecto	Frecuencia TivaC I2C 3400kHz
8	7	8
16	15	15
32	30	31
64	59	60
128	106	111
250	175	193
475	279	325
860	395	447

También es necesario cambiar el número de datos que se almacenan y envían en forma de paquete a 20 valores, respecto los 80 que se envían en Arduino en el modo híbrido. Esto se debe a que se tarda más en enviar los paquetes de datos.

Para la tarjeta de desarrollo TM4C1294XL, hay que instalar su correspondiente librería, dentro del programa de energía, llamada “Energia TivaC boards”, como se aprecia en la figura 51.

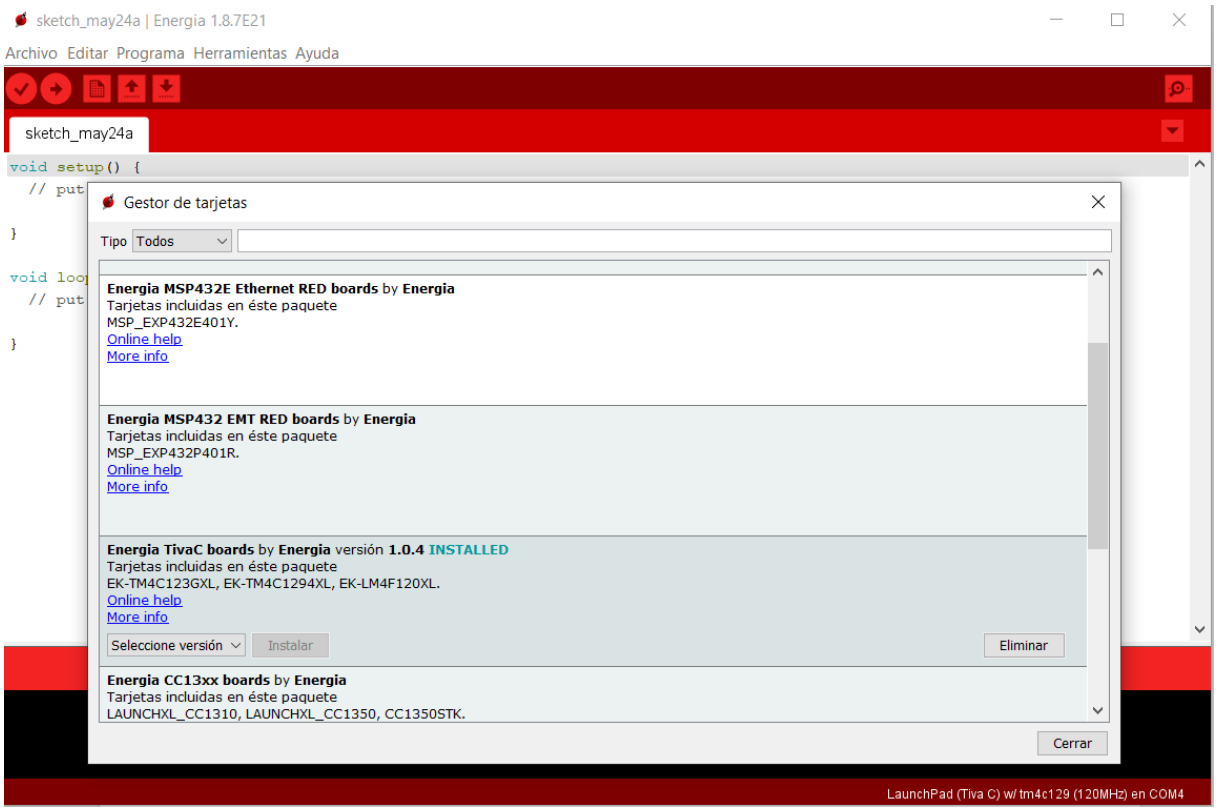

Figura 51. Gestor tarjetas Energia

Aparte de esta librería, es necesario instalar los drivers de la tarjeta en el ordenador, en este caso “Stellaris ICDI (In Circuit Debug Interface)”. En Windows 10 tiene la funcionalidad de la instalación automática de los drivers, pero para el resto de los sistemas operativos es necesario descargarlos de la página web de Texas Instruments e instalarlos en el equipo de desarrollo. En la figura 52, se muestra los drivers instalados con la tarjeta conectada. Se aprecia que está conectada a través del puerto de comunicación 4 y un par de líneas más abajo el driver que permite reconocer la tarjeta.

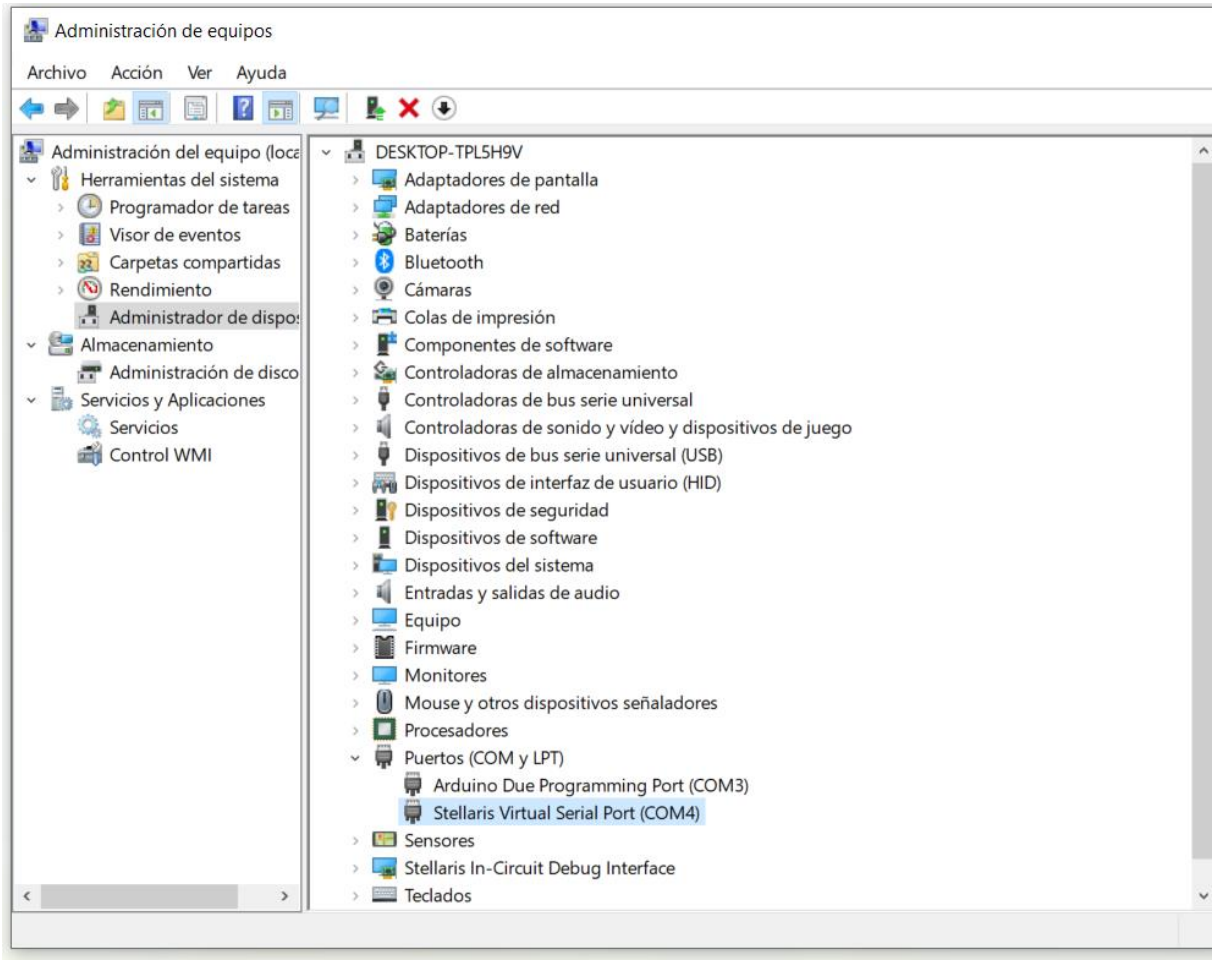

Figura 52. Drivers TivaC

3.3.2 Software Profesional

Para la programación de las tarjetas de Texas Instruments también se dispone de software profesional, que permite el desarrollo de programas utilizando el 100% de las funciones de las que dispone el hardware, estos programas se llaman Code Composer Studio y Keil ubision.

Code Composer Studio es el entorno de desarrollo integrado oficial de Texas Instruments disponible en su página web. Permite la programación de todas las tarjetas del catálogo de la empresa. Incluye un compilador optimizado de C/C++, editor de código fuente, entorno de compilación del proyecto, depurador, generador de perfiles y muchas otras características. Dispone de la posibilidad de importar proyectos de Energia IDE, aunque es necesario tener el programa Energia instalado en el mismo ordenador que CCS. Permite acceder a toda la documentación de Texas Instruments, así como ejemplos y librerías para todas las placas de desarrollo. Se dispone de la guía de usuario y de tutoriales de esta herramienta la propia web de Texas Instruments. La figura 53 muestra el programa con un sketch cargado de Energia IDE.

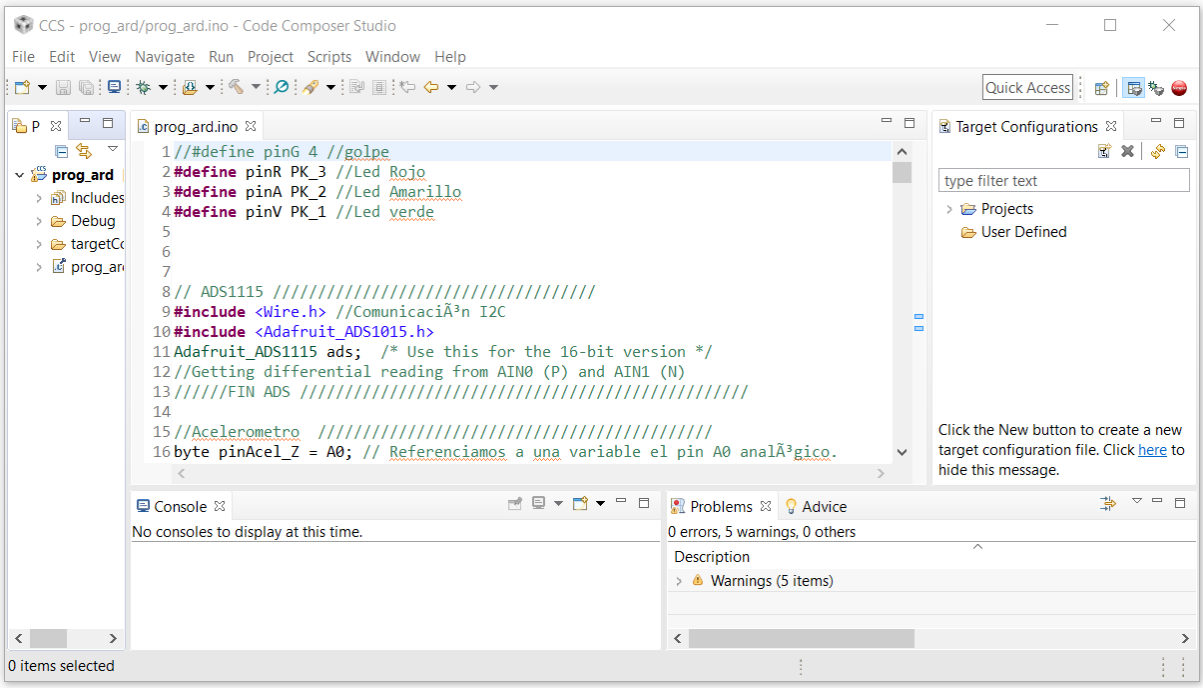

Figura 53. Code Composer Studio

Keil µbision es otro entorno de desarrollo integrado que permite la programación de tarjetas de Texas Instruments, aunque en este caso no se puede todas y solo sirve para programar la TivaC de las nombradas en este proyecto. En la web www.edx.org se disponen de cursos gratuitos para aprender su uso con estás tarjetas de desarrollo. Se muestra como es entorno de trabajo en la figura 54.

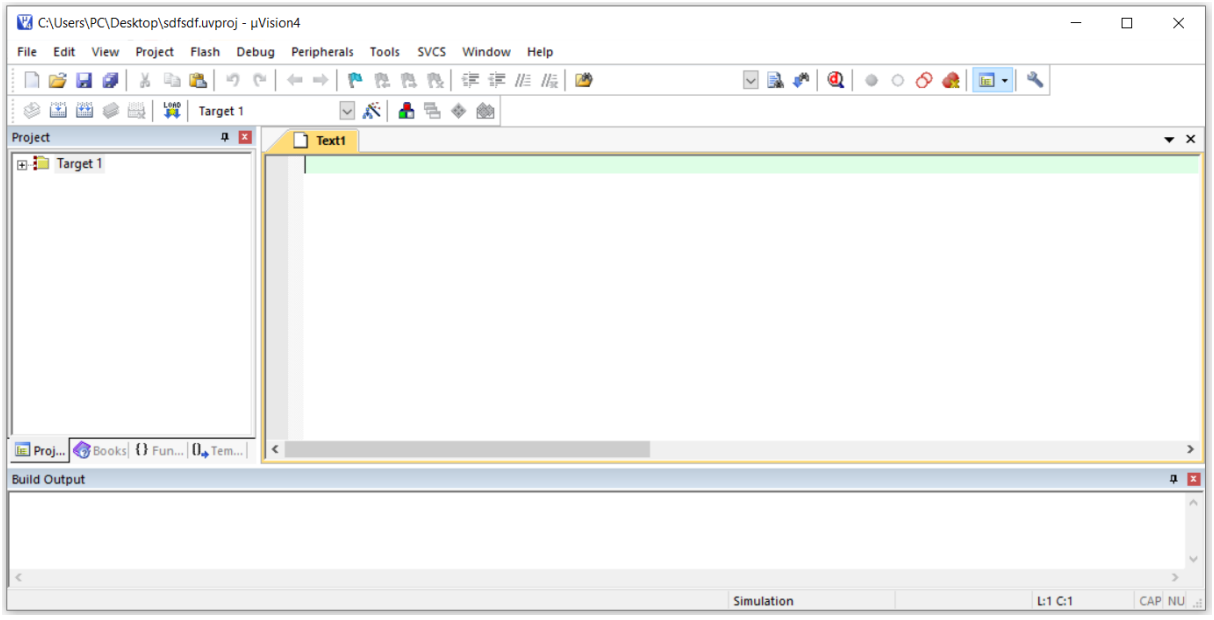

Figura 54. Keil µbision

CAPÍTULO 4. ANÁLISIS DE RESULTADOS

Sistema estructural

Para una comprobación básica, debido a las restricciones provocadas por la alerta sanitaria Covid-19, del sistema SHM desarrollado, se ha realizado una estructura de una viga en voladizo. La viga está formada por una barra de policarbonato de sección 0,065x0,0045 m y la longitud de la viga se ha fijado en 0,25 m. En su extremo se ha colocado la célula de carga y el acelerómetro, como se ve en la figura 55. Con este sistema estructural se realizan todos los análisis del sistema que se muestran a continuación.

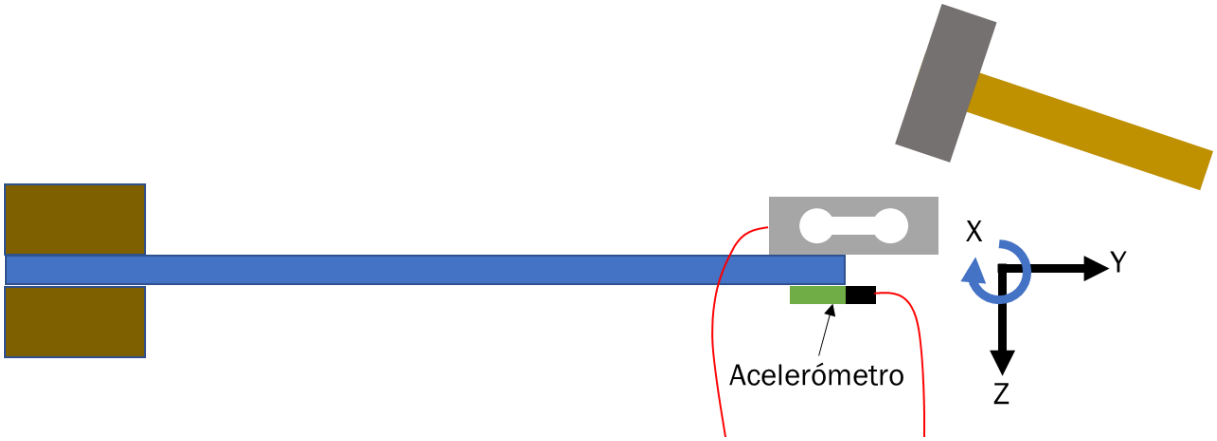

Figura 55. Esquema ensayo.

4.1 Arduino vs TivaC

En la tabla 23 se comparan las características de hardware de los dos microcontroladores seleccionados, Arduino Due Y TivaC

Tabla 23. Comparativa hardware microcontroladores

Atributo	DUE	TivaC
Microcontrolador	AT91SAM3X8E	ARM Cortex-M4F
Bits microcontrolador	32 bits	32 bit
Voltaje de funcionamiento	3,3V	5V
Voltaje de alimentación	7-12V	4.75-5.35 V
Pin digitales I/O	54 (12 salidas PWM)	90 (8 salidas PWM)
Entradas analógicas	12 (12bits)	20 (12 bits)
Salidas analógicas	2 (DAC) (12bit)	0
Intensidad total de todos los pines I/O	130 mA	400 mA
Intensidad a 3.3V Pin	800 mA	1 A
Intensidad a 5V Pin	800 mA	1 A
Memoria flash	512 KB toda disponible	1024 KB
SRAM	96 KB (2: 64KB y 32KB)	256 KB
Velocidad de reloj	84 MHz	120 MHz

Tamaño	101.52x53.3 mm	124.5x56x10.8 mm
Peso	36 g	162 g
Precio	35 €	23,39 €

Se aprecia que el modelo de Texas Instruments tiene un hardware ligeramente más potente al modelo de Arduino, mayor velocidad de reloj, más pines digitales y más memoria RAM y flash. A continuación se muestra si esta mayor potencia en la TivaC se traduce en un mejor rendimiento para hacer funcionar el programa desarrollado.

Precio

Como se vio en el capítulo 2, el microcontrolador de Texas Instruments (23,39€), es más barato que el de Arduino oficial (35€). Dicho esto, se pueden encontrar replicas, debido a su filosofía open source, en tiendas chinas donde el precio ronda los 11€, aunque las prestaciones son las mismas, la calidad de fabricación y durabilidad puede verse afectado

Prestaciones

Para comparar las prestaciones de ambos microcontroladores, se carga el programa correspondiente en cada uno. Se ejecuta el programa de Python para dar las órdenes oportunas al microcontrolador. Se realizan las comparaciones para el modo de lectura offline y el modo híbrido, pues es donde el microcontrolador opera a sus máximas prestaciones. Primero se comprueba la frecuencia de lectura conseguida en modo híbrido, posteriormente, se comprueban las frecuencias de lectura en modo offline de cada sensor. Por último se realiza la comprobación de la lectura offline de la fuerza y un eje del acelerómetro donde se analiza: la frecuencia de muestreo, la capacidad de memoria, el tiempo de ensayo máximo y la frecuencia de envió. Se analiza más esta última funcionalidad pues permite un análisis completo de la estructura al comparar las fuerzas a las que es sometida y su respuesta en aceleraciones.

No se compara el modo lectura online con visualización, pues el funcionamiento y las prestaciones obtenidas son similares para ambas placas de desarrollo. Se obtiene una frecuencia de 62 Hz, siendo limitante el bluetooth Raspberry Pi y sin límite de tiempo pues el microcontrolador no almacena datos.

En la tabla 24 se muestran las prestaciones del programa desarrollado en las dos tarjetas de desarrollo y posteriormente se comentan estos resultados, en comparación con las características del hardware que se recogen en la tabla 23. Lo valores se han obtenido según los siguientes puntos:

- La frecuencia de muestreo la obtiene el programa del microcontrolador, realizando una serie de muestreos automáticamente y la envía a la Raspberry Pi, para mostrarla en la interfaz de usuario.
- La capacidad máxima de memoria, la calcula el microcontrolador evaluando su memoria RAM disponible, antes de empezar a almacenar datos. Este dato se expresa en valores de 16 bits que se pueden almacenar en la memoria.
- Si se intenta hacer un ensayo de mayor duración del máximo para el microcontrolador se obtiene el tiempo de ensayo máximo por pantalla. Una vez evaluada la memoria RAM por parte del microcontrolador, la transforma en tiempo y la envía a la Raspberry Pi para informar al usuario.
- El tiempo de ensayo a la misma se frecuencia se ha hecho matemáticamente para comparar la capacidad de almacenaje en las mismas condiciones.
- La frecuencia de envío offline se ha obtenido a partir del número de muestras enviadas entre el tiempo que se ha tardado en enviar. El tiempo de envío se obtiene

restando el tiempo total del ensayo, el tiempo entre envió e inicio y el número de muestras aparece en la interfaz.

Tabla 24. Comparativa prestaciones Arduino vs TivaC

Atributo	Arduino Due	TM4C1294XL
Frecuencia muestreo modo híbrido	247 Hz	199 Hz*
Frecuencia muestreo – Fuerza offline	785 Hz	447 Hz
Frecuencia muestreo – 3 ejes acelerómetro offline	1000 Hz	975 Hz
Máxima frecuencia de muestreo – (Fuerza/1-Aceleracion)	783 Hz	445 Hz
Capacidad máxima de memoria	45358 valores	128254 valores
Tiempo ensayo máximo (Fuerza/1-Aceleracion)	28 s	144 s
Tiempo ensayo máximo (Fuerza/1-Aceleracion) a 447Hz	50 s	144 s
Frecuencia de envió offline. (Fuerza/1-Aceleracion)	15700 muestras / 26,98 s = 582 Hz	16020 muestras / 28,82 s = 556 Hz

Se obtiene mayores frecuencias de muestreo en todos los ensayos para el Arduino DUE. Principalmente la diferencia de frecuencia de muestreo viene dada por el reloj de la comunicación I2C, aunque también es más lento leyendo directamente las entradas analógicas. En Arduino se ha podido subir la frecuencia de reloj hasta 1600 KHz, pudiendo alcanzar con ello los 783 muestras por segundo, para la lectura de ambos sensores. En la TivaC se ha modificado su frecuencia a 3400 kHz y añadido una resistencia pull up a cada circuito del bus I2C, pero solo se han conseguido alcanzar las 445 muestras por segundo, para la lectura de ambos sensores. **Para este proyecto la mayor limitante en cuanto a la velocidad es el ADC de la célula de carga, no afectando prácticamente la diferencia de velocidad entre los procesadores.** La TivaC tiene mayor velocidad de reloj 120 MHz, contra los 84MHz del Arduino Due, pero obtiene peores resultados en todos los casos.

Para el caso del modo híbrido, aparte de obtener menor frecuencia de muestreo para la TivaC es necesario enviar paquetes de datos más pequeños (80 Arduino vs 20 TivaC) pues tarda más tiempo en enviarlos. Esto provoca que el programa no funcione correctamente, si se carga el programa de Arduino sin modificar.

El tiempo durante el que se realiza el muestreo a máxima frecuencia es muy superior en la TivaC, debido a la mayor memoria RAM, 256 Kb en la TivaC y 96 Kb en el DUE. En este caso no afecta en nada la programación pues es la misma en ambas, definiéndose el mismo tipo de variables, siendo todos los valores leídos de 16bits. A la misma frecuencia de lectura los dos microcontroladores, el Arduino DUE, solo podría realizar un ensayo del 35% de la duración del ensayo en la TivaC.

A continuación se muestran las señales registradas de un ensayo en modo híbrido de Arduino, figura 56, y de la TivaC, figura 57. En estas imágenes se puede ver que la señal registrada en Arduino tiene menos ruido principalmente en el acelerómetro. Esta tendencia también se puede percibir en el modo offline que la muestra en la figura 58 y en la figura 59, correspondiente al Arduino Due y a la TivaC respectivamente.

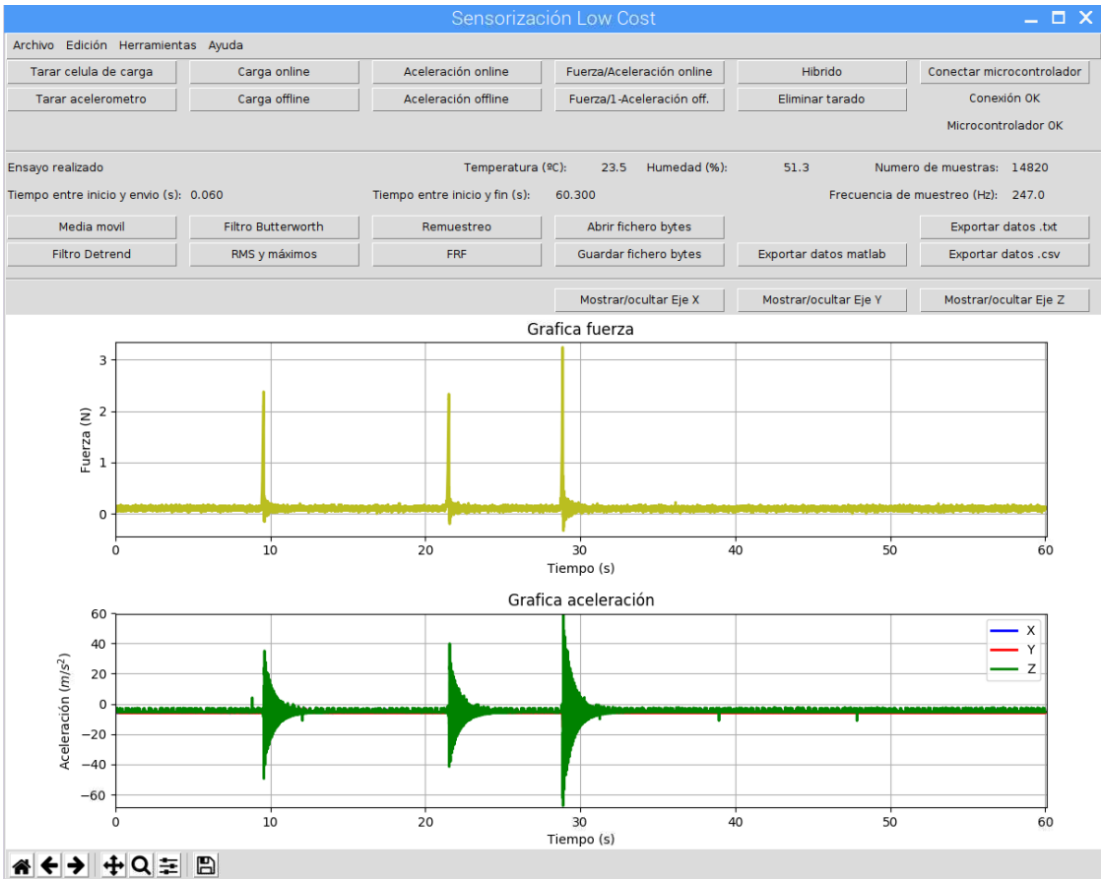

Figura 56. Modo híbrido Arduino

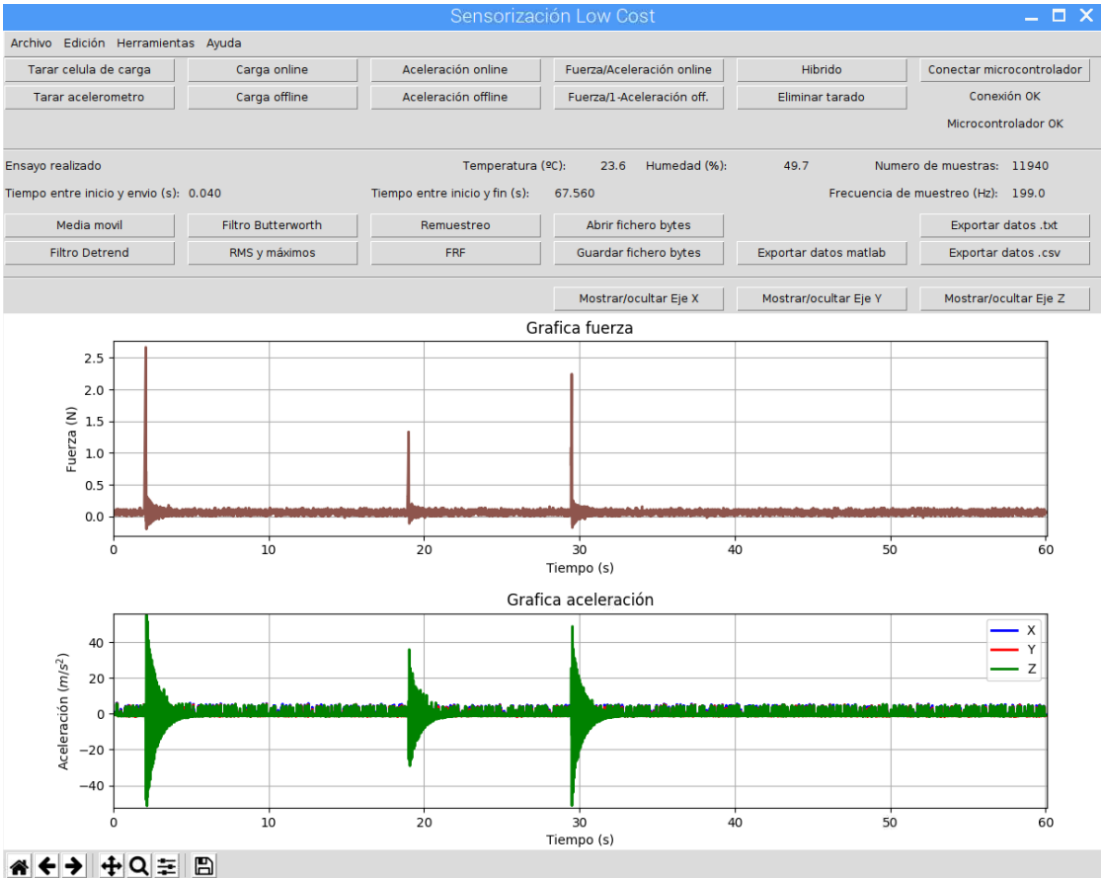

Figura 57. Modo híbrido TivaC

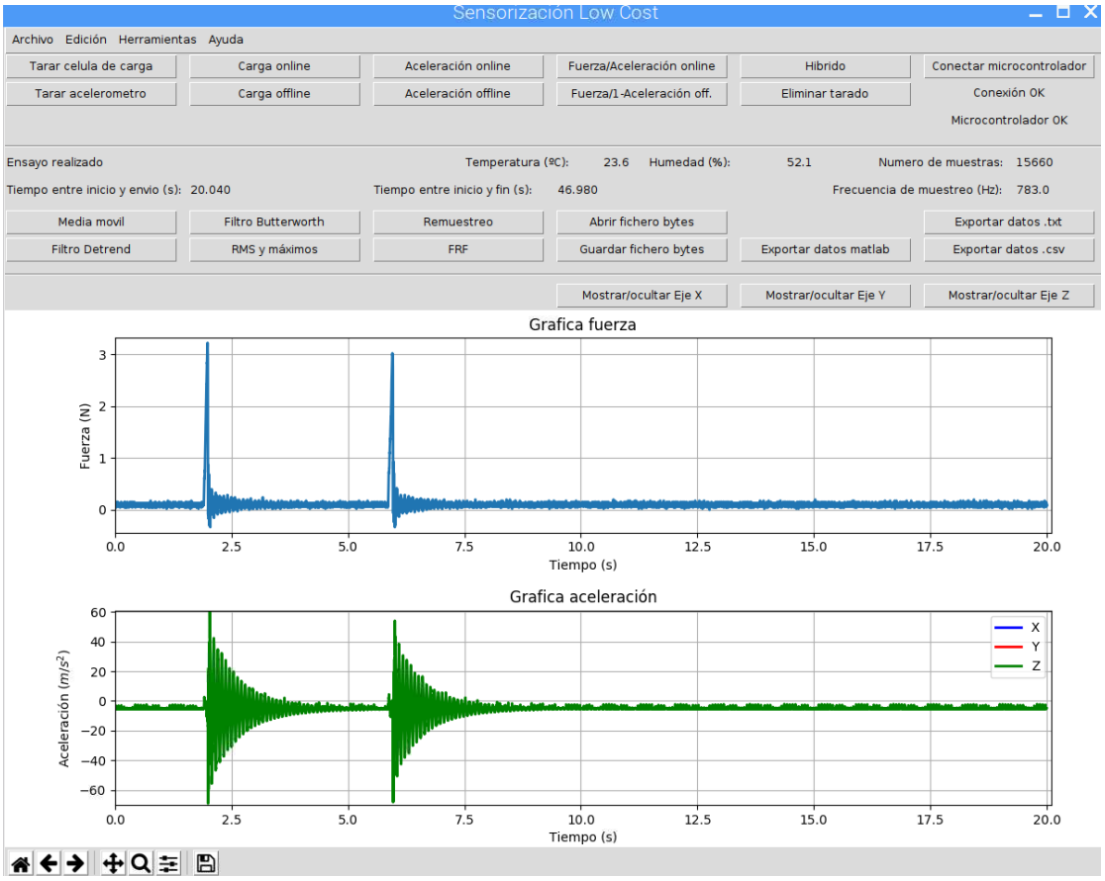

Figura 58. Modo offline Fuerza/1-Aceleracion Arduino

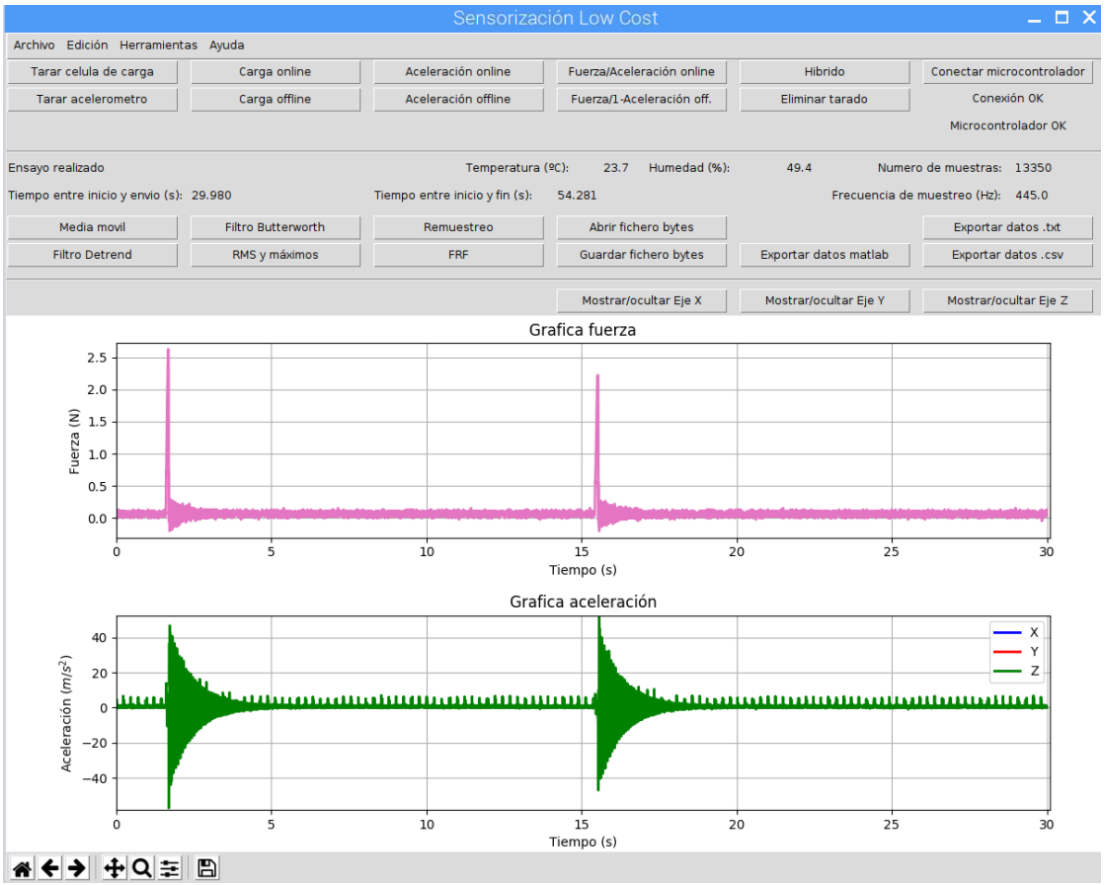

Figura 59. Modo offline Fuerza/1-Aceleracion TivaC

Facilidad de uso

Arduino tiene mayor facilidad de uso, disponiendo de una gran cantidad de librerías, tutoriales y foros exclusivos debido a que la comunidad de desarrollo que tiene detrás Arduino es enorme. Para la TivaC al disponer de Energia IDE la ventaja del entorno de desarrollo y el lenguaje se pueden igualar. Pero ante la complicación como es el caso de establecer la velocidad de reloj de la comunicación I2C, no dispone de tanta información a la que acudir como Arduino. Además algunas librerías no se pueden ejecutar en Energia IDE.

Energia IDE no está disponible para el uso en todos los microcontroladores, siendo necesario acudir a Code Composer Studio, que utiliza lenguaje C y C++, menos intuitivo y más lento de programar. Cada microcontrolador tiene sus funciones específicas, no pudiendo servir el mismo programa, para dispositivos de distintas familias. En los microcontroladores de Arduino, se puede reutilizar el mismo programa para varios dispositivos, salvo algunas excepciones, como es el caso de la configuración del módulo bluetooth HC-08 y la librería “SoftwareSerial” no estando disponible para Arduino Due.

A nivel de hardware también resulta más fácil trabajar con el modelo de Arduino, pues las resistencias pull-up del bus I2C, vienen incluidas en la propia placa, por lo menos en el bus 0. En el modelo de Texas Instruments es necesario añadirla para todos los buses I2C, pues no viene incluida en ninguno de los 3. Además en Arduino el nombre y función de cada pin viene identificado en la propia placa facilitando con ello los montajes.

En este apartado el punto favorable que disponen los microcontroladores de Texas Instruments es la extensión de la documentación técnica a disposición del usuario de cada elemento y de todas sus librerías.

4.2 Resultados obtenidos del programa

A continuación se muestra un experimento realizado con Arduino DUE, pues ha sido el controlador que mejores prestaciones ha dado, excepto para la duración del tiempo de ensayo en modo offline. Para este experimento se realiza una medición de fuerza y aceleración en el eje Z en modo offline. Con esta señal registrada se realizan los distintos filtrados y se calcula la respuesta en frecuencia. Posteriormente se exportan las señales registradas y se comparan con la respuesta en frecuencia obtenida con Matlab. Este experimento se realiza sobre la estructura antes mencionada, como se ve en la figura 60.

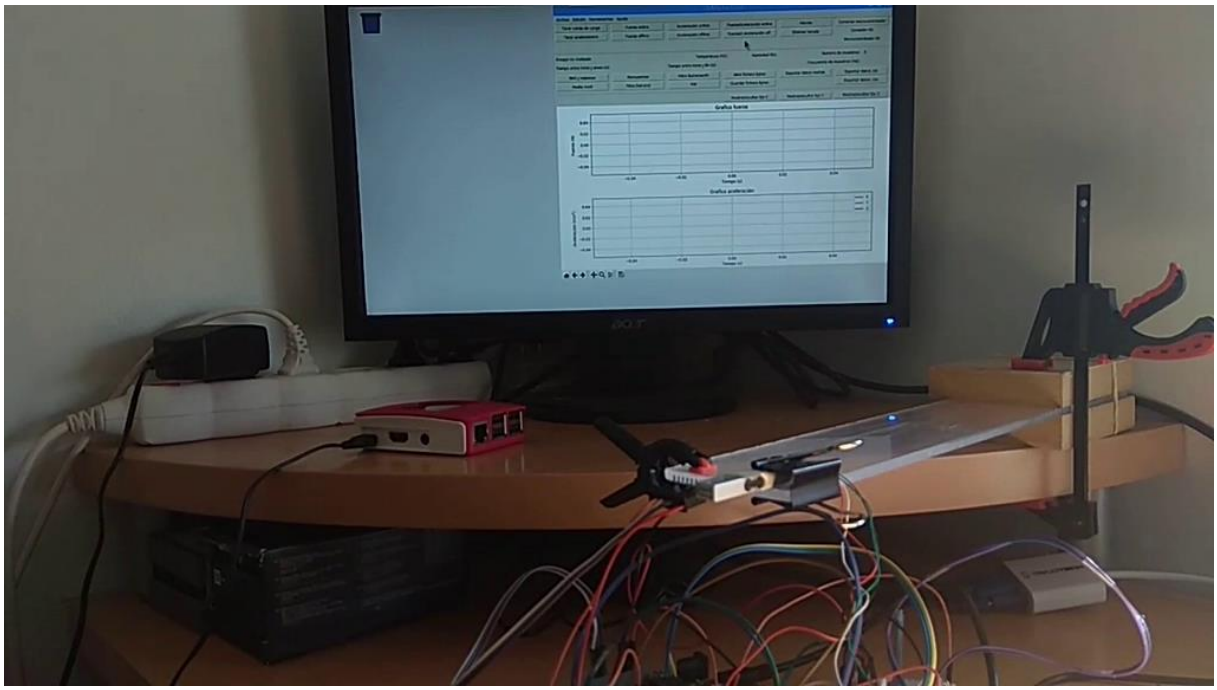

Figura 60. Montaje realizado para el ensayo.

Se realiza un ensayo de 10 segundos en modo offline de la fuerza y la aceleración en el eje Z. Durante el ensayo se realiza un impacto en la célula de carga, figura 61

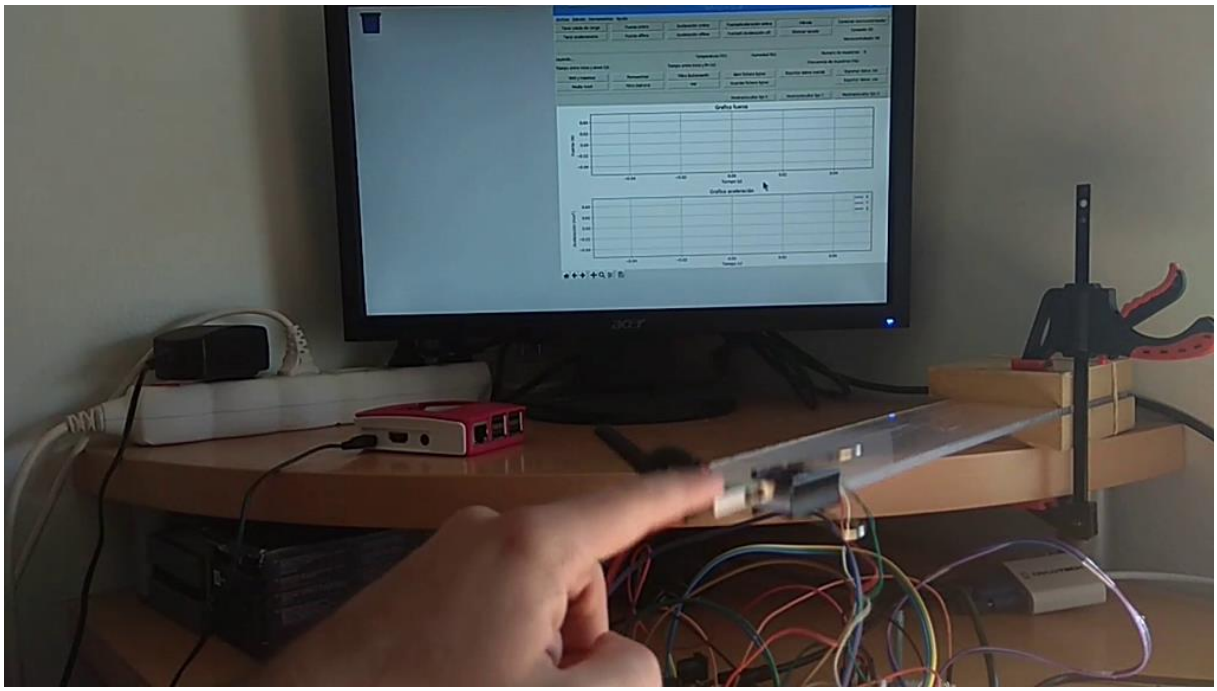

Figura 61 Impacto en la célula de carga.

Una vez finalizado la medición y recibida en la Raspberry Pi se muestran en las gráficas embebidas de la interfaz, figura 62. El ensayo se ha realizado a 28,1ºC y una humedad relativa del 49,7 %. Se han tomado 7830 muestras de cada señal, fuerza y aceleración, a una frecuencia de 783 Hz cada una.

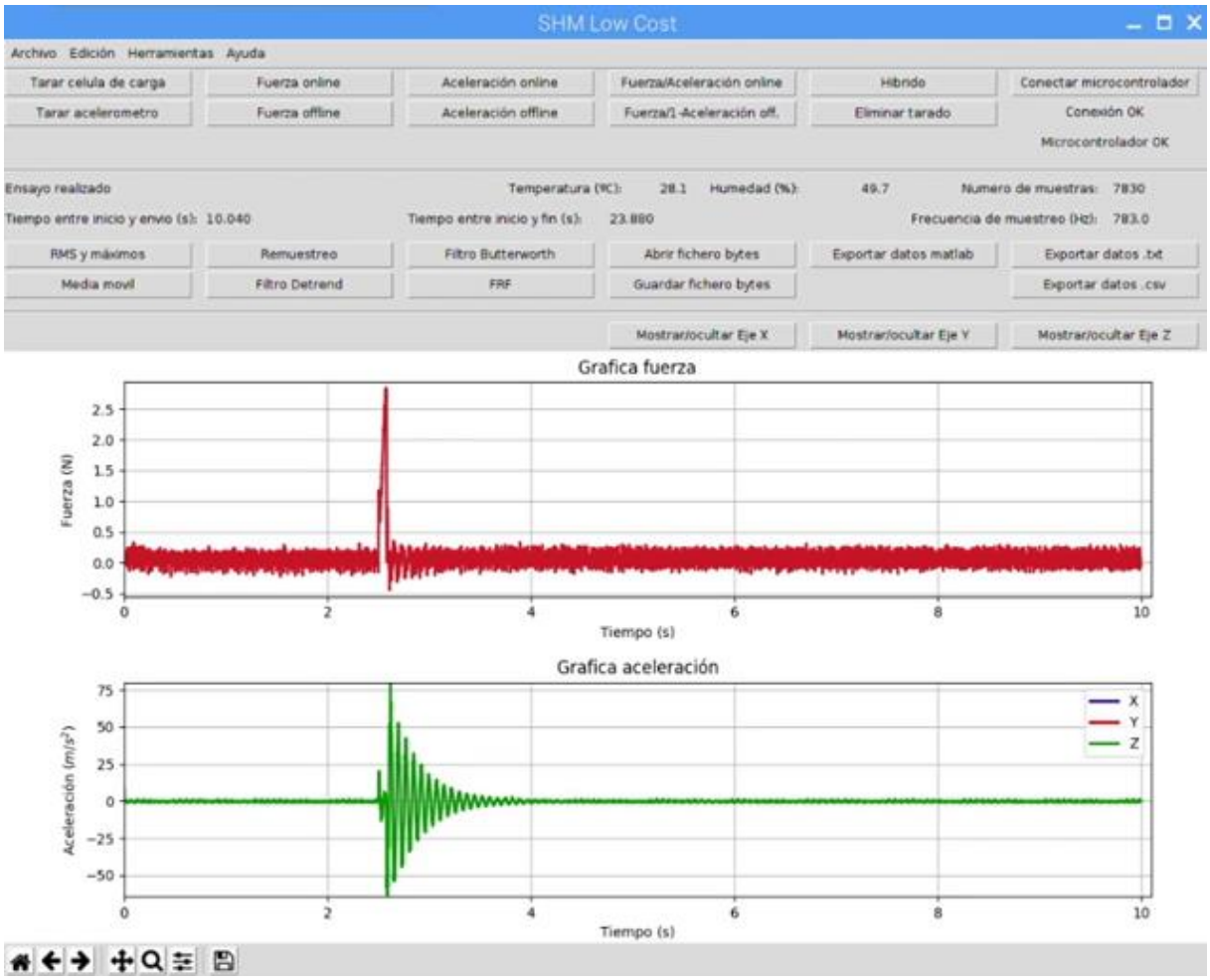

Figura 62. Datos registrados.

Primero se calculan los valores de la RMS y los máximos y mínimos obtenidos de cada señal, se muestran en la figura 63

Información de las señales muestreadas

Valores de las ondas

RMS carga	0.195 N	MAX carga	2.843 N	MIN carga	-0.444 N
RMS aceleración X	0.000 m/s²	MAX aceleración X	0.000 m/s²	MIN aceleración X	0.000 m/s²
RMS aceleración Y	0.000 m/s²	MAX aceleración Y	0.000 m/s²	MIN aceleración Y	0.000 m/s²
RMS aceleración Z	6.238 m/s²	MAX aceleración Z	79.310 m/s²	MIN aceleración Z	-63.519 m/s²

Figura 63. RMS y máximos

A continuación se empieza a realizar el procesado de las señales. Primero se realiza media móvil con tamaño de ventana 6, con ello se consigue eliminar el ruido.

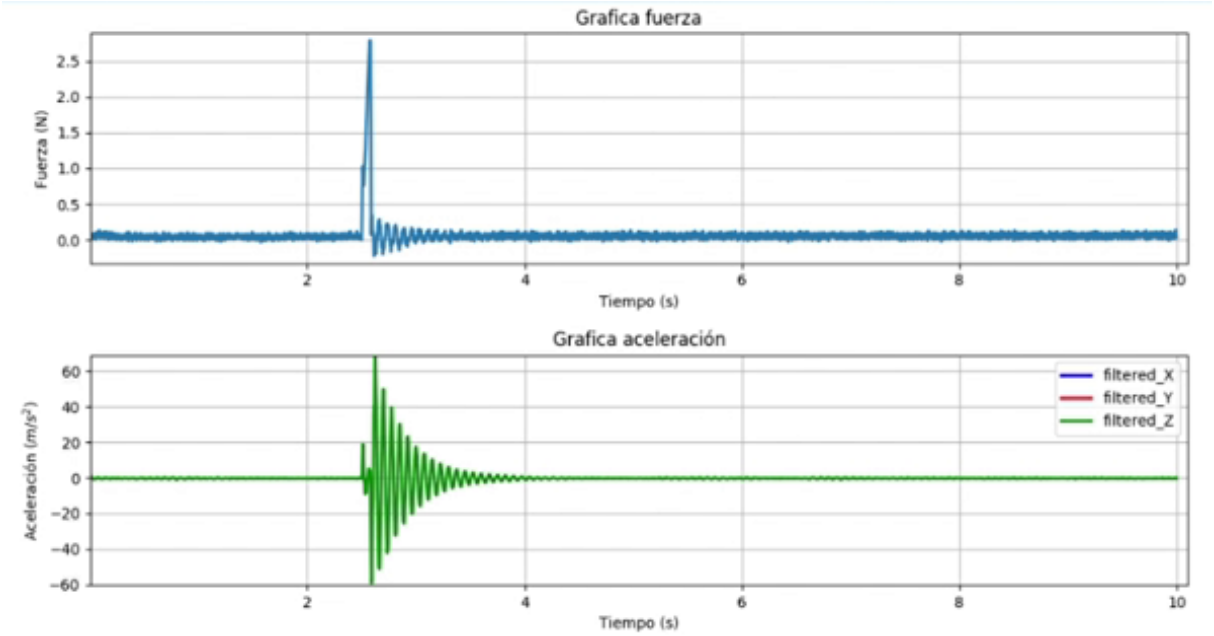

Figura 64. Media móvil realizada

Posteriormente se realiza un remuestreo tomando los datos obtenidos con la media móvil. Se seleccionan 10000 muestras simulando con ello que se ha leído la señal a 1000 Hz. El resultado obtenido se muestra en la figura 65.

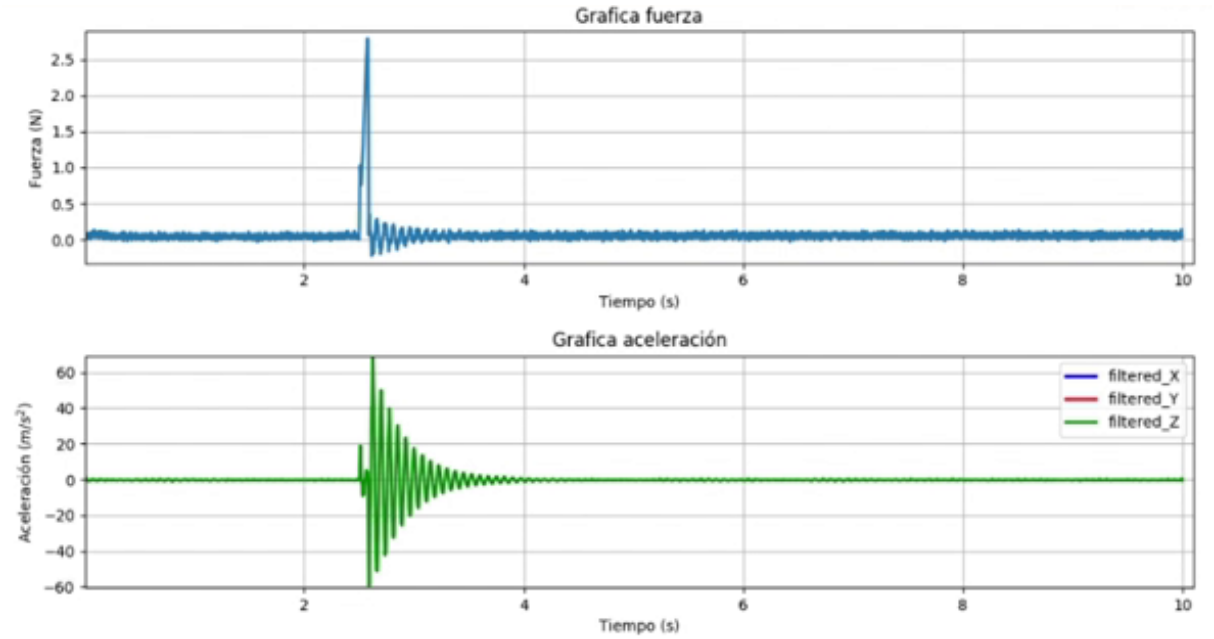

Figura 65. Remuestreo realizado

Lo siguiente que se realiza es un filtro Butterworth, con los datos obtenidos del remuestreo, de orden 4 y frecuencia de corte de 250 Hz. Una vez aplicado el filtro el resultado se muestra en la figura 66.

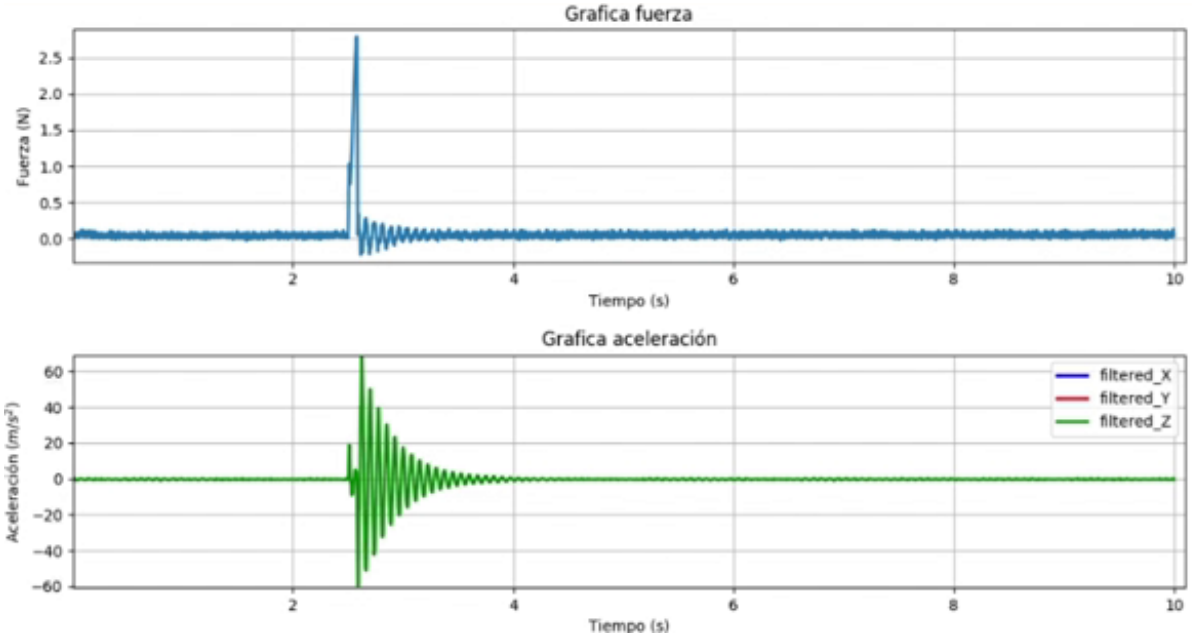

Figura 66. Aplicado filtro Butterworth.

Como ultimo procesado se calcula la FRF del sistema, para ver cómo se comporta el sistema ante la entrada aplicada. Se realiza cambiando la frecuencia de muestreo (fs) a 1000 Hz debido al remuestreo, poniendo el tipo de ventana “boxcar(1000)”, que divide el registro en segmentos de 1000 puntos en formas rectangulares. Se deja el resto de los términos en blanco, para utilizar los valores por defecto de Python, que es igual a superposición de 500 y puntos de la FFT igual a 1000. En la figura 67, se muestra primero la densidad espectral de potencia “Pxx”, evaluando la señal de entrada del sistema, en este caso la fuerza. En segundo término se calcula la densidad espectral de la potencia cruzada “Pxy”, que utiliza las señales de fuerza y aceleración. Una vez calculados estos términos se obtiene la función de respuesta en frecuencia. En la FRF se observa un pico en aproximadamente 14 Hz, que se corresponde con el primer modo de flexión de este voladizo.

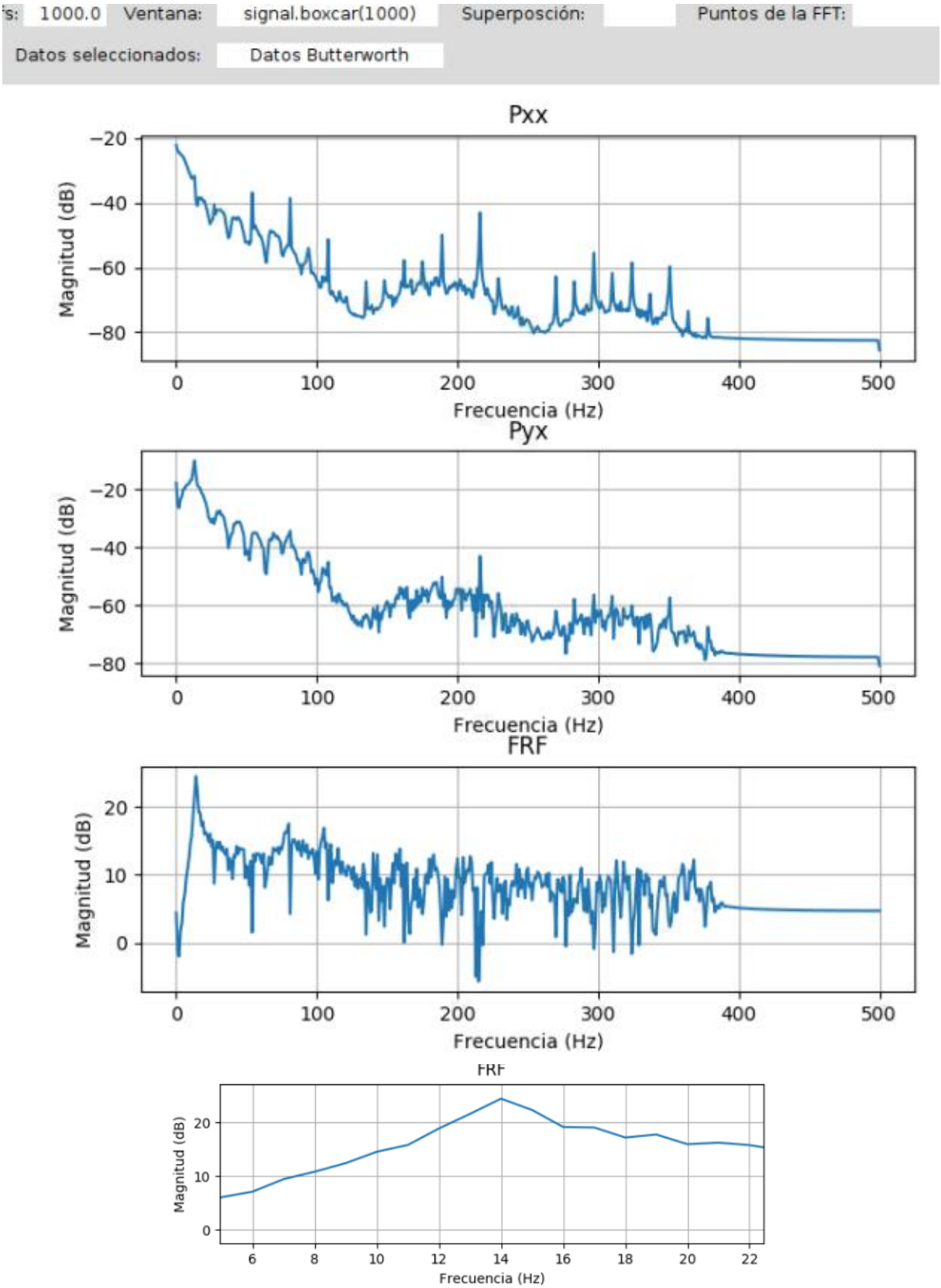

Figura 67. Cálculo FRF.

Los valores registrados y filtrados se exportan a Matlab, para realizar la comparación. Con los mismos valores con los que se calcula en Python, aplicados todos los filtros, se realiza la función *"tfestimate"* con los mismos parámetros en Matlab. En la figura 68, se puede apreciar que los resultados son los mismos que con el programa desarrollado.

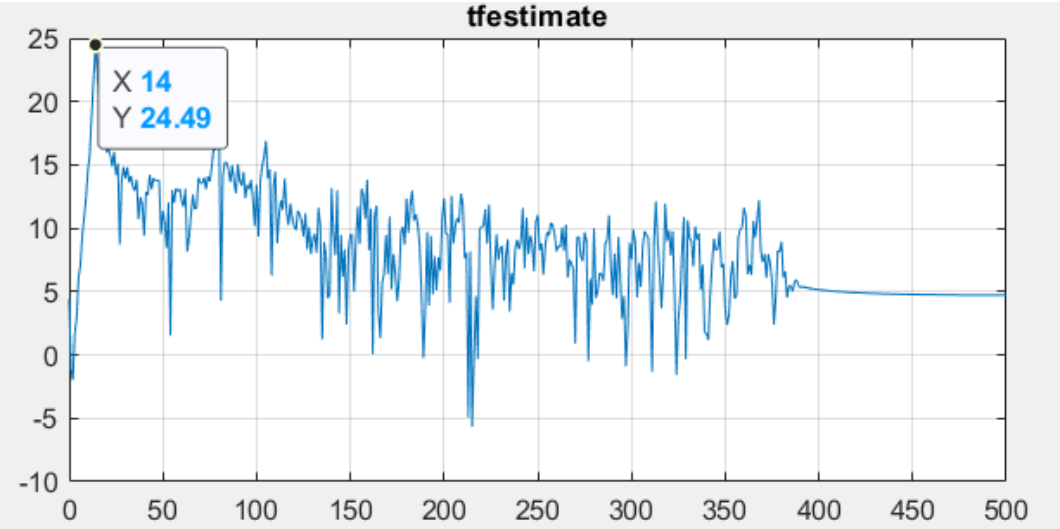

Figura 68. FRF calculada en Matlab.

4.3 Coste de desarrollo

A continuación se realiza el análisis económico completo del sistema desarrollado, para ello a parte del coste del material, mostrado en el capítulo 2, se añade el coste de personal.

Para el coste de personal se estima como salario de un ingeniero industrial de 25727,97€/año. Este salario se corresponde con mínimo establecido por el convenio del sector de empresas de ingeniería y oficinas de estudios técnicos (Trabajo & Social, 2019). Se calcula el coste por hora, considerando una jornada de 8 horas, durante 220 días al año y añadiendo la parte correspondiente de la seguridad social por parte de la empresa, obteniendo un coste de 19,44 €/h.

Tabla 25. Costes del proyecto

Concepto	Coste Ud.	Cantidad	Coste	
			Arduino	TivaC
Coste material			166,91 €	155,3 €
Selección de material	19,44 €/h	10 h	194,4 €	
Diseño y montaje hardware	19,44 €/h	5 h	97,2 €	
Desarrollo Python	19,44 €/h	250 h	4860 €	
Desarrollo Arduino	19,44 €/h	80 h	1555,2 €	
Desarrollo Energia IDE	19,44 €/h	10 h		291,6 €
Redacción de documentación	19,44 €/h	120 h	2332,8	
Total		480 h	9206,51 €	9486,5 €

En la tabla 25, se recogen los costes del proyecto con un precio total de nueve mil doscientos seis con cincuenta y un euros con el microcontrolador de Arduino. Para la TM4C1294-XL se le han imputado más horas de trabajo debido a complicaciones surgidas al I2C y a la investigación de los diversos entornos de desarrollo, ascendiendo el coste a nueve mil cuatrocientos ochenta y seis con cincuenta euros.

El coste de material supone menos del 2 % del coste total en ambos casos, siendo el coste de personal el resto debido al ser el desarrollo de un prototipo.

CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Conclusiones

Se ha conseguido el desarrollo de un sistema SHM de bajo coste basado en un microcontrolador y una Raspberry Pi comunicado inalámbricamente con las siguientes características:

- El sistema SHM realiza lecturas de fuerzas y aceleraciones, posibilitando el análisis de la dependencia de comportamiento de la estructura según la temperatura y humedad.
- Para el procesado de las señales se han incorporado los siguientes filtros:
  - Media móvil, para suavizar la señal.
  - Remuestreo, aumentar el número de muestras.
  - Butterwoth, atenuar las frecuencias a partir de una frecuencia determinada.
- Para el análisis estructural se realiza el cálculo de la respuesta en frecuencia (FRF).
- Se permite exportar las señales registradas y los valores de las señales filtradas en formatos .txt, .csv y en .txt en formato para su análisis en Matlab. También se exportan las condiciones ambientales y de las señales.

En cuanto a las limitaciones del sistema SHM obtenidas son las siguiente:

- La mayor limitación para el modo de lectura online o para el modo híbrido ha sido el tiempo de lectura del módulo bluetooth, programado con Python.
- Para el muestreo a máxima frecuencia el limitante es el ADC ADS1115 de la célula de carga. Para poder mejorar el sistema de lectura es necesario un ADC con mayores frecuencias o un microcontrolador que tuviera un ADC de por lo menos 16 bits, como el C2000 de Texas Instruments.
- La memoria RAM de los microcontroladores no permite hacer experimentos a máxima frecuencia de muestreo muy largos en el tiempo, como se expone en la comparación entre microcontroladores.

Se ha conseguido realizar una comparación entre ambos de las características técnicas, facilidad de uso y prestaciones conseguidas:

- Arduino DUE ofrece mayores frecuencias de muestreo y de envió de datos que la TivaC. La principal diferencia se debe a la lectura del ADC, siendo el limitante del sistema, dónde Arduino consigue muestrear a 785 Hz y el modelo de Texas Instruments obtiene 447 Hz. Aunque también se obtienen peores frecuencias en el muestreo del acelerómetro y en el envío de datos por Bluetooth. Arduino se impone a pesar de tener menor frecuencia de reloj siendo de 84 MHz vs 120 MHz de la TivaC.
- La mayor memoria RAM de la TivaC de 256 Kb, permite ensayos más largos que la placa de Arduino, que posee 96 Kb. La duración máxima del ensayo a máxima frecuencia es 28 segundos para Arduino DUE, mientras que para la TivaC es de 144 segundos. Aunque son conveniente microcontroladores con mayor capacidad de memoria RAM, para poder realizar ensayos más largos.
- Se ha comprobado que Energia IDE permite programar las tarjetas de Texas Instruments con la misma facilidad que se programan los microcontroladores de Arduino. Pero la facilidad de uso es mayor en Arduino, debido a la gran comunidad que hay detrás desarrollando y compartiendo librerías, programas y tutoriales. Las tarjetas presentan el inconveniente de la necesidad de instalar los drivers en el

ordenador de desarrollo, mientras que en los modelos de Arduino no es necesario. Respecto de hardware también es más fácil el uso de Arduino, pues incorpora internamente las resistencias pull-up en los bus I2C.

- El precio es menor la TivaC, pero debido a las peores prestaciones conseguidas la diferencia de precio no es destacable. Además se pueden encontrar modelos no oficiales, al ser open source, más baratos que la TivaC.

A parte de estas conclusiones se ha documentado lo que pudiera ser de interés para seguir el desarrollo de líneas de investigación relacionados con este TFM y se proponen trabajos futuros compatible con esta línea de trabajo.

Finalmente se han ampliado los estudios del máster de Ingeniería Industrial con conocimientos de vibraciones de sistemas estructurales, electrónica, acondicionamiento de señales y programación en Arduino y Python

5.2 Trabajos futuros

Se proponen las siguientes líneas de trabajos futuros, relacionados con este TFM:

- Comparar el sistema SHM de bajo coste realizado, con un sistema de adquisición de datos profesional, como el Sirius STG disponible en el laboratorio de estructuras de la Escuela de Ingeniería Industrial.
- Desarrollar un programa en el software profesional Code Composer Studio para el microcontrolador TivaC y poder comparar si al peor rendimiento se debe a la programación con Energia IDE.
- Realizar un programa para la placa de desarrollo C2000 Delfino de Texas Instruments, ya que sus prestaciones son mayores
- Desarrollar el programa de la Raspberry Pi en Django, para tener un entorno de trabajo web.
- Instalar el sistema SHM de bajo coste desarrollado en el demostrador que se está haciendo en el laboratorio de estructuras.

Algunos de estos puntos podrían haber sido abordados al menos preliminarmente de no haber limitaciones de acceso al laboratorio debido a la alerta sanitaria Covid-19.

5.3 Consideraciones adicionales

El impacto medioambiental generado durante el desarrollo de este Trabajo de Fin de Master ha sido muy bajo. La mayoría de los elementos que se han utilizado se han reutilizado de otros proyectos anteriores. Estos elementos y los adquiridos nuevos como la Raspberry Pi, y los dos microcontroladores se dejan a disposición del departamento para continuar la línea de investigación. Por lo tanto, no se ha generado ningún residuo ni se ha empleado ningún tipo de material peligroso que pudiera contaminar el medioambiente.

El consumo energético de estos dispositivos es muy bajo, siendo el consumo del ordenador de desarrollo el principal elemento de consumo. Se estima una potencia del ordenador portátil utilizado de 220 W, durante las 480 horas de desarrollo del proyecto. Se estima unas emisiones de 0,27 $KgCO_2/KWh$ que genera Endesa, según (Carbono, 2020). Con ello se obtiene una huella de carbono del sistema SHM desarrollado de 28,5 $KgCO_2$.

BIBLIOGRAFÍA

Alario, J. M. (2018). *DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA PRE-EVALUACIÓN DEL COMPORTAMIENTO DINÁMICO DE ESTRUCTURAS ESBELTAS*. http://repositorio.ucsg.edu.ec/bitstream/3317/11771/1/T-UCSG-PRE-ART-IPM-152.pdf

Arduino. (2020). *Arduino Due*. Documentation. https://store.arduino.cc/arduino-due

Carbono, R. D. E. H. D. E. (2020). *Factores de emisión*.

Dascotte, E., Strobbe, J., & Tygesen, U. T. (2013). Continuous stress monitoring of large structures. *5th International Operational Modal Analysis Conference, IOMAC 2013*, 1–10.

Electronics, M. (2015). Mouser Electronics. *Electronics*, *877*, 7–8. http://za.mouser.com/

Freescale Semiconductor. (2011). *Three Axis Low-g Micromachined Accelerometer MMA7361L*. 1–11.

Gentile, C., & Saisi, A. (2015). Continuous dynamic monitoring of a centenary iron bridge for structural modification assessment. *Frontiers of Structural and Civil Engineering*, *9*(1), 26–41. https://doi.org/10.1007/s11709-014-0284-4

Ian Harvey. (2020). *Bluepy*. https://ianharvey.github.io/bluepy-doc/

Keyestudio HC-08 Bluetooth Module. (n.d.). 1–26.

Liu, T. (2016). Digital Humidity and Temperature sensor. *Adfruit*, 1–5. https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

NumPy. (2020). *Numpy*. https://numpy.org/

Python Software Foundation. (2020). *Tkinter*. https://docs.python.org/3/library/tkinter.html

Raspberry Pi Foundation. (2016). Raspberry Pi 3 Model B. *Raspberry Pi Website*, 2837. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

SciPy developers. (2020). *SciPy*. https://www.scipy.org/

Texas Instruments. (2016). ADS1115 Datasheet. *Data Sheet of ADS1115*, 1–51.

The Matplotlib development team. (2020). *Matplolib*. https://matplotlib.org/

The SciPy community. (2019). *Scipy signal window*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.get_window.html#scipy.signal.get_window

Trabajo, M. D. E., & Social, Y. S. (2019). *Boletín Oficial del Estado: Ministerio De Trabajo , Migraciones*. 32526–32554.

Travush, V. I., Shulyat'ev, O. A., Shulyat'ev, S. O., Shakhraman'yan, A. M., & Kolotovichev, Y. A. (2019). Analysis of the Results of Geotechnical Monitoring of "Lakhta Center" Tower. *Soil Mechanics and Foundation Engineering*, *56*(2), 98–106. https://doi.org/10.1007/s11204-019-09576-9

Tuñón, D. (2016). *Sistema de monitorización de bajo coste para el registro de desplazamientos, fuerzas y aceleraciones en maquetas de estructuras*. Universidad de Valladolid.

Ye, Z., Li, N., & Zhang, F. (2019). Wind characteristics and responses of Xihoumen Bridge

during typhoons based on field monitoring. *Journal of Civil Structural Health Monitoring*, 9(1), 1–20. https://doi.org/10.1007/s13349-019-00325-y

5.4 WEBGRAFIA

[1]. Internet: https://upload.wikimedia.org/wikipedia/commons/thumb/0/06/Ponte_morandi_crollato.jpg/1200px-Ponte_morandi_crollato.jpg Ultima visita: 20/06/20.

ANEXOS

Anexo 1. Código Arduino para configurar HC-08

```
#include <SoftwareSerial.h>
#define RxD 10
#define TxD 11
SoftwareSerial BTSerial(RxD, TxD); // Recive (RD), Transmit (TxD)
void setup() {
 Serial.begin(9600);
 while (!Serial) {
 ; // wait for serial port to connect. Needed for native USB port only
 }
 BTSerial.begin(115200);
 Serial.println("Enter AT commands:");
 delay(100);
}

void loop() {
 if (BTSerial.available()) {
  Serial.write(BTSerial.read());
 }
 if (Serial.available()) {
  BTSerial.write(Serial.read());
 }
}
```

Anexo 2. Test velocidad ADS1115

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1115 ads;     // Use this for the 16-bit version
float   multiplier;



void setup(void)
```

```
{
  Wire.setClock(1600000);
  Serial.begin(9600);
  Serial.println("Hello!");

  ads.setGain(GAIN_TWOTHIRDS);
  ads.begin();
  multiplier = ads.voltsPerBit()*1000.0F;          // Gets the millivolts per bit

  // Run test for each SPS
  Serial.println("********** 8 SPS ************");
  ads.setSPS(ADS1115_DR_8SPS);
  runTest();

  Serial.println("********** 16 SPS ************");
  ads.setSPS(ADS1115_DR_16SPS);
  runTest();

  Serial.println("********** 32 SPS ************");
  ads.setSPS(ADS1115_DR_32SPS);
  runTest();

  Serial.println("********** 64 SPS ************");
  ads.setSPS(ADS1115_DR_64SPS);
  runTest();

  Serial.println("********** 128 SPS ************");
  ads.setSPS(ADS1115_DR_128SPS);
  runTest();

  Serial.println("********** 250 SPS ************");
  ads.setSPS(ADS1115_DR_250SPS);
  runTest();

  Serial.println("********** 475 SPS ************");
  ads.setSPS(ADS1115_DR_475SPS);
```

```
  runTest();

  Serial.println("********* 860 SPS ************");
  ads.setSPS(ADS1115_DR_860SPS);
  runTest();
}



void loop(void){
}

void runTest()
{
  int16_t results0_1, freq;
  delay(2000);
  uint32_t sampleTime1,sampleTime2;
  sampleTime1= micros();
  results0_1 = ads.readADC_Differential_0_1(); //1
  sampleTime2=micros();
  Serial.print("Differential 0-1: "); Serial.print(results0_1); Serial.print("(");
Serial.print(results0_1 * multiplier); Serial.println("mV)");
  Serial.print("Read Time Microseconds: ");
  Serial.println((sampleTime2 - sampleTime1));
  Serial.print("Frecuencia: ");
  Serial.println(1000000/(sampleTime2-sampleTime1));
  Serial.println();
}
```

Anexo 3. Guía SHM

Arduino

1. Descargar Arduino IDE.
2. Instalar en el equipo Arduino IDE.
3. Dentro de Arduino. En gestor de Tarjetas descargar la llamada “Arduino SAM boards (32-bits ARM Cortex-M3)”. Nota: Aparece como sugerencia al conectar el DUE con Arduino abierto
4. Descargar las librerías de DHT y ADS1115:
    a. https://github.com/adafruit/DHT-sensor-library
    b. https://github.com/soligen2010/Adafruit_ADS1X15.

5. Instalar las dos librerías en Arduino usando “incluir librerías desde .zip”
6. Abrir programa “Adquisición Arduino”
7. Compilar y subir con Arduino DUE conectado.

Energia IDE

1. Descargar en el equipo Energia IDE.
2. Descomprimir archivo en la carpeta deseada. El programa es ejecutable no se instala.
3. Descargar drivers de la tarjeta EK-TM4C1294XL.
    a. https://www.ti.com/tool/STELLARIS_ICDI_DRIVERS
4. Descomprimir e instalar drivers
5. Dentro de Energia IDE. En gestor de Tarjetas descargar la llamada “Energia TivaC boards”.
6. Descargar las librerías de DHT y ADS1115:
    a. https://github.com/adafruit/DHT-sensor-library
    b. https://github.com/soligen2010/Adafruit_ADS1X15.
7. Instalar las dos librerías en Energia usando “incluir librerías desde .zip”
8. Abrir programa “Adquisición Energia”
9. Compilar y subir con TM4C1294XL conectado.

Raspberry Pi

El modelo entregado ya dispone de todos estos pasos realizados, solo es necesario hacer estos pasos si se cambia la memoria SD

1. Descargar en el equipo Raspberry PI OS (32-bit) with desktop and recommended software.
2. Instalar en una tarjeta SD siguiendo las instrucciones.
3. Arrancar la Raspberry Pi.
4. Conectarse a internet
5. Ejecutar el terminal.
6. Instalar IDE python
7. sudo apt-get install idle3
8. Instalar la librería numpy con el comando:
    a. Sudo pip3 install numpy
9. Instalar la librería bluepy con el comando:
    a. Sudo pip3 install bluepy
10. Instalar la librería matplotlib con el comando:
    a. Sudo pip3 matplotlib
11. Instalar la librería scipy con el comando:
    a. Sudo apt-get install python3-scipy
12. Ejecutar script de Python. Dos opciones:
    a. Modificaciones: Abrir Python 3 IDLE y abrir programa
    b. Solo funncionemiento: Abrir terminal y ejecutar los siguientes comandos.
        i. cd /”Ruta del archivo”
        ii. python3 SHM.py

Nota: Para el correcto funcionamiento del programa el script de Python “SHM.py” y la base de dato de los sensores “Datos sensores” deben estar en la misma carpeta.