Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Implementación embebida de los algoritmos de procesado de un dispositivo para la detección de caídas

Autor: Santiago Domínguez Vidal
Tutores: Luis Javier Reina Tosina
David Naranjo Hernández

Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Implementación embebida de los algoritmos de procesado de un dispositivo para la detección de caídas

Autor:
Santiago Domínguez Vidal

Tutores:
Luis Javier Reina Tosina
Catedrático de Universidad
David Naranjo Hernández
Doctor investigador en Ingeniería Biomédica

Dep. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Implementación embebida de los algoritmos de procesado de un dispositivo para la detección de caídas

Autor: Santiago Domínguez Vidal

Tutores: Luis Javier Reina Tosina
David Naranjo Hernández

El tribunal nombrado para juzgar el proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A ellas

A los buenos profesores

Agradecimientos

A las tres mujeres que me sufrieron y ayudaron, cada una en su medida y papel, a superar este tremendo reto.

A esos profesores que se preocupan verdaderamente por la transmisión de conocimiento, en especial a Carmen Sáez Agullo, José Luis Calvo Gallego, Alfredo Pérez Vega-Leal y José Ignacio León Galván, que siempre se preocupan por que sus alumnos comprendan todo y están dispuestos a repetir las veces que hagan falta las explicaciones, a Manuel Ángel Perales Esteve e Hipólito Guzmán Miranda por crear ambientes idóneos para el desarrollo de ideas y el compañerismo y, finalmente, a Pau Giménez-Gavarrell, por contar los mejores chistes de ingenieros y fomentar un ambiente en el que entender cosas como el número de Reynolds.

A Luis Javier Reina Tosina y David Naranjo Hernández, tutores del trabajo, por haberme guiado durante este proyecto y haberme dado las pautas adecuadas para llevarlo a buen puerto.

Santiago Domínguez Vidal

Escuela Técnica Superior de Ingeniería

Sevilla, 2023

Resumen

En la actualidad, las personas mayores o dependientes que viven solas se enfrentan cada día a la posibilidad de sufrir una caída sin nadie que las pueda auxiliar. Qué útil sería tener un mecanismo de detección de estos posibles accidentes y de producirse avisar a un contacto de emergencia o a los servicios de emergencias.

El diseño de este tipo de dispositivos constituye una de las líneas de investigación del Grupo de Ingeniería Biomédica de la Universidad de Sevilla. Concretamente, se trata de investigar y desarrollar soluciones basadas en sistemas inteligentes para la detección de caídas, que en caso de detectar una caída inicien una llamada a un servicio de emergencias o a una persona de contacto.

Actualmente el grupo de investigación dispone de un prototipo de laboratorio cuyo firmware está desarrollado, hasta la fecha, en lenguaje ensamblador, limitando la portabilidad y dificultando los desarrollos futuros que se puedan necesitar debidos a cambios en los requisitos del sistema.

Por tanto, es objetivo principal de este proyecto, adaptar el algoritmo de detección desarrollado por el equipo a lenguaje C, un lenguaje que permite adaptar el desarrollo fácilmente a otros microcontroladores y facilita la implementación de nuevas funcionalidades o cambios de hardware.

Se ha optado por desarrollar una librería propia para el funcionamiento del acelerómetro incluido en el prototipo, utilizando librerías existentes de Microchip para el uso de la UART y la comunicación SPI. El algoritmo principal desarrolla los cálculos de energía partiendo de las aceleraciones leídas en el mismo bucle y ejecuta las comparaciones con umbrales para detectar picos de aceleraciones y energías, que unidas darán como resultado la detección de impactos. Finalmente, tras finalizar el desarrollo se valida el algoritmo con datos experimentales de escenarios simulados obtenidos previamente con voluntarios (durante el desarrollo de la investigación), dando como resultado la viabilidad del dispositivo (y algoritmo) para dicha detección de impactos.

Abstract

At present, the elderly or dependent people who live alone face every day the possibility of suffering a fall without anyone who can help them. How useful it would be to have a mechanism to detect these possible accidents and, if they occur, notify an emergency contact or the emergency services.

That is the purpose of the work carried out by the Biomedical Engineering research team at University of Seville, which describes an intelligent fall detection system that, in case of detecting a fall, initiates a call to an emergency service or a contact person.

However, this project is implemented to date in assembly language, limiting portability and hindering future developments that may be needed due to changes in system requirements.

Therefore, the main objective of this project is to adapt the detection algorithm developed by the team to C language, a language that allows development to be easily adapted to other microcontrollers and facilitates the implementation of new functionalities or hardware changes.

It has been decided to develop a library for the operation of the accelerometer included in the prototype and get existing Microchip libraries for the use of the UART and SPI communication. The main algorithm develops the energy calculations starting from the accelerations read in the same loop and executes the comparisons with thresholds to detect peaks of accelerations and energies, which together will result in the detection of impacts. Finally, after completing the development, the algorithm is validated with experimental data from simulated scenarios previously obtained with volunteers (during the development of the research), resulting in the feasibility of the device (and algorithm) for such impact detection.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En la actualidad, las personas mayores o las personas dependientes en cierto grado, que por desgracia o voluntad viven solas o pasan la mayor parte del día solas, sin supervisión de alguien que las pueda auxiliar en caso de accidente, se enfrentan cada día a la angustia de imaginar qué pasaría si se cayeran o se desmayaran de repente y nadie se percatara de lo sucedido, al menos en un periodo de tiempo suficiente como para generar complicaciones en una posterior recuperación o incluso para que la persona fallezca.

Por ello, se hace necesario describir un sistema inteligente de detección de caídas que en caso de detectar una caída inicie una llamada a un servicio de emergencia o a una persona de contacto.

Existen diversas soluciones en este ámbito de aplicación y en la mayoría de los casos se utiliza un acelerómetro para la detección. Pero muchos de ellos cuentan con desventajas como no poder usarlos en ambientes con agua, o fuera de casa, no poder usarlos sin ropa, no contar con una autonomía suficiente como para que no suponga demasiado esfuerzo para el usuario tener que cargarlo o cambiar pilas cada poco tiempo.

En el caso que se describe en este proyecto, se parte de un prototipo investigado y desarrollado por el Grupo de Ingeniería Biomédica de la Universidad de Sevilla (código PAIDI TIC-203) [1]. Dicho sistema consta de varios dispositivos con una arquitectura de funcionamiento distribuida, la cual se compone de un dispositivo (SoM, *Sensor of Movements*) que detecta impactos, que debe portar el usuario, un software que se ejecuta en una aplicación de móvil (DAD, *Decision-Analysis Device*) que detecta si el impacto ha sido, o no, una caída, y otro software (RTC, *Remote Telehealthcare Center*) que monitoriza los datos que se envían y desde donde se pueden configurar parámetros personalizados del usuario, que se ejecuta en una computadora.

El dispositivo encargado de detectar los impactos se coloca con un parche biocompatible en la espalda del usuario, se puede usar fuera de casa, en el baño, etc. Cuenta con una autonomía del orden del tiempo de uso de un parche biomédico, reduciendo la atención al mantenimiento del dispositivo. Es muy pequeño gracias a la arquitectura distribuida, que permite que los cálculos complejos no sea necesario realizarlos en el equipo que se porta, sino en el resto de los dispositivos de la arquitectura, los cuales están conectados al SoM de forma inalámbrica.

En este trabajo se realizará una adaptación (o reimplementación) a lenguaje C de un firmware (que se ejecutará en el SoM) basado en la patente ES2378934 e implementado en lenguaje ensamblador por los investigadores del mencionado grupo de investigación, al que pertenecen los tutores, que ha desarrollado el sistema completo [1].

Ensamblador es un lenguaje de bajo nivel, muy eficiente y por tanto atractivo para sistemas con pocos recursos, pero que resulta demasiado complejo para un público menos especializado y es específico, al menos el conjunto de instrucciones, del sistema físico que se utiliza para la implementación.

El lenguaje de programación C es un lenguaje mucho más versátil a la hora de implementar nuevas funcionalidades en un sistema, mucho más portable entre diferentes microcontroladores, al tener un grado mayor de abstracción y mayor capacidad para un desarrollo modular, y más accesible a personas con conocimientos no tan especializados de sistemas microcontroladores.

Es por estas características que posee el lenguaje C por las que se decide implementar el firmware del SoM en este lenguaje.

1.1 Objetivos

El objetivo principal de este trabajo es realizar la implementación en lenguaje C del algoritmo de detección de caídas, que es el núcleo del firmware del SoM, en el microcontrolador integrado en el prototipo de este (PIC18F2431), al igual que evaluar su validez frente a distintos escenarios con caídas reales.

1.2 Resumen de metodologías y materiales

Para ello se comenzará con la familiarización del entorno de desarrollo y las particularidades del microcontrolador, configuración inicial y algún programa sencillo para comprobar que el ciclo de desarrollo, implementación y funcionamiento del hardware es claro.

Más tarde, se implementará la funcionalidad objetivo de este proyecto, que es el algoritmo de detección de caídas, realizando diversas pruebas de comunicación de datos en tiempo real, generación de gráficas con datos guardados durante varias ejecuciones del algoritmo y confirmación de la detección de caídas provocando aceleraciones que superen el umbral y que generan energías que también lo hagan.

Finalmente, una vez implementado el algoritmo y comprobado que funciona en experimentos con escenarios simulados, se realizará la validación con experimentos reales utilizando a voluntarios que porten el prototipo en la espalda y simulen diferentes escenarios con caídas y sin ellas, para dar por válido el funcionamiento del firmware del SoM en lenguaje C.

Durante las diversas fases del proyecto se han utilizado distintas herramientas y materiales con diversos propósitos.

Los materiales principales han sido:

- Prototipo del SoM con PCB adaptadora de las conexiones externas a tiras de pines 2.54 mm
- Programador PICkit™3
- Traductor USB-TTL 3.3 V

Algunos materiales secundarios que se han utilizado son:

- LED
- Resistencia
- Soldador 60 W
- Cableado 30 AWG
- Analizador lógico

1.3 Estructura de la memoria

La estructura de este documento se inicia con un capítulo que presenta una revisión del estado del arte en materia de dispositivos para la detección de caídas. Posteriormente se suceden distintos capítulos que siguen los pasos de la metodología detallada anteriormente.

1. Toma de contacto con el sistema y el entorno de desarrollo.
2. Configuración e implementación de los requisitos previos (librerías ajenas y propias).
3. Implementación del algoritmo de detección de caídas.
4. Validación del funcionamiento del algoritmo con experimentos reales.
5. Conclusiones y mejoras disponibles.

Cada uno de los capítulos de la memoria entra en más detalle en las partes que así lo requieren, por ejemplo, el desarrollo de una librería para el acelerómetro, las diversas pruebas realizadas con las lecturas de las aceleraciones y la explicación de la implementación concreta del algoritmo de detección de impacto.

2 ESTADO DEL ARTE

Tal y como reporta la OMS (Organización Mundial de la Salud), la población de avanzada edad (con 60 años o más) se incrementará hasta los 2.100 millones para el 2050, convirtiéndose en un 22% de la población mundial total y la población con 80 años o más se prevé que alcance los 426 millones [2]. El envejecimiento conlleva la pérdida de las capacidades físicas y sensoriales, lo cual aumenta el riesgo de sufrir una caída. De nuevo, la OMS reporta que entre el 28% y el 35% de la población con 60 años o más, sufre al menos una caída al año. Y que para la población con 70 años o más, el porcentaje se eleva a entre el 32% y el 42% [3]. A esto hay que añadir que los mayores de 60 años son quienes sufren más caídas mortales. La rapidez en la respuesta a las caídas sufridas por la población de avanzada edad podría disminuir la gravedad de las consecuencias.

Este tema ha sido revisado por diversos estudios, tanto internacionales como nacionales [4], para valorar cuál es la incidencia actual, cuáles son los factores de riesgo y cuáles son las medidas necesarias para reducir la incidencia de este problema.

Una de las posibles medidas, como se comentaba anteriormente, es alertar a tiempo a familiares o servicios de emergencia para reducir las consecuencias de las caídas en cuanto a gravedad de las lesiones, a la vez que otorgar a los usuarios de los sistemas de alerta la tranquilidad de que si algo les ocurre serán atendidos en cualquier momento.

El interés sobre la detección de caídas ha ido creciendo con el tiempo, especialmente desde 2018 cuando se percibe un vertiginoso incremento. La Figura 2-1 muestra el ranking de 10 países con mayor número de publicaciones al respecto desde 1945 hasta 2020, entre los que se coloca España en décima posición [5].

Figura 2-1. Top 10 países con mayor número de publicaciones sobre sistemas de detección de caídas desde 1945 hasta 2020.

Durante los últimos años, los investigadores han estudiado diferentes sistemas de detección de caídas con el objetivo de conseguir este reto. Estos sistemas utilizan múltiples variedades de sensores para obtener datos que

posteriormente podrán ser procesados por algoritmos de análisis. La mayoría de estos sistemas hacen uso de aceleraciones para la detección de un golpe causado por un impacto corporal [6]. Una de las primeras investigaciones sobre este campo [7] hace uso únicamente de un impacto con aceleraciones elevadas. Para mejorar la detección de caídas, otros usan sistemas con múltiples tipos de sensores, ya sean inerciales [8], de audio o de imagen [9], y algunos llevan los datos procesados a algoritmos de Aprendizaje Máquina (o Machine Learning) [10], para obtener los mejores resultados. Además de los diferentes sistemas de detección de caídas, otras investigaciones de temas relacionados se han llevado a cabo, como, por ejemplo, prevención de caídas, tecnologías de bajo consumo para los sistemas de detección, la selección de la mejor ubicación del sensor para una detección con alta precisión, etc. [6].

Por último, un reciente estudio, además de analizar algunos de los temas comentados previamente, proporciona un repositorio de datos (UMAFall [11]) obtenidos de experimentos realizados con un conjunto de sujetos realizando movimientos (tanto actividades de la vida diaria como caídas simuladas), los cuales portan 5 dispositivos sensorizados en diferentes partes del cuerpo, con el objetivo de ser empleado para testar diferentes algoritmos y técnicas de detección de caídas [12].

2.1 Clasificación de los sistemas de detección de caídas

La revisión de investigaciones existentes sobre sistemas de detección de caídas se puede llevar a cabo siguiendo diferentes clasificaciones. En este caso, y tal como se relata en [6], se comparan dos métodos de clasificación:

1. Clasificación por tipo de sensores utilizado.
2. Clasificación por tipo de algoritmo de análisis aplicado.

2.1.1 Clasificación por tipo de sensores utilizado

En esta clasificación se presentan cuatro tipos de sensores, como se detalla a continuación.

- Sistemas basados en sensores inerciales:

  En este tipo de sistemas, se suele colocar el sensor en cuestión en alguna parte del cuerpo del usuario para medir los cambios bruscos del cuerpo, utilizados posteriormente para detectar caídas y diferenciarlas de actividades de la vida cotidiana. La miniaturización, portabilidad, bajo coste y funcionamiento en tiempo real son características que hacen de estos sistemas una buena opción para el uso de la ciudadanía de avanzada edad [6].

  La mayoría de los sistemas usan un solo sensor, donde este suele ser un acelerómetro o giroscopio de 2/3 ejes, por ejemplo, Shahzad desarrolló un sistema de prevención de caídas que posteriormente extendió a un preciso sistema de detección basado en un acelerómetro [13] y otros como Bourke y Lyons [14] o Su [15] utilizaron giroscopios. También se puede usar un barómetro como muestra Lu [16] u otros sistemas basados en sensores de presión colocados en las plantas de los pies [17], electromiografía o inclinómetros.

- Sistemas basados en el contexto del usuario:

  En este tipo de sistemas se detecta la caída procesando información contenida en el ambiente (contexto) para monitorizar el movimiento del cuerpo humano. Se colocan sensores como micrófonos, sensores

de presión, infrarrojos, cámaras, sensores térmicos, etc. en los alrededores del usuario en los que va a permanecer, como el dormitorio, el baño, o toda una vivienda [6].

Específicamente en los sistemas ambientales, se procesa audio, vibraciones o señales de presión para monitorizar al usuario en el campo de visión del sensor. Características de tipo MFCC (*Mel-Frequency Cepstral Coefficient*) de una señal acústica pueden extraerse para capturar los movimientos del usuario, como se propone en [18] y [19]. Una señal de vibraciones puede obtenerse a partir de varios sensores de presión (piezo-resistivos o resistivos) [20] o de una alfombrilla en el suelo [21].

De estos, los sistemas acústicos son los que parecen obtener mejores resultados en la detección de caídas. Sin embargo, los sistemas ambientales presentan problemas como el hecho de que se ubican sensores sólo en interiores o en una habitación, dejando espacios muertos o puntos ciegos en la detección de caídas, es decir, presentan un rango de detección limitado. También se ven afectados por factores ambientales externos como otros elementos cayendo, el tipo de pavimento del suelo y ruidos de diferente índole [6].

- Sistemas basados en radiofrecuencia:

  Estos sistemas monitorizan fluctuaciones en las señales de radio frecuencia o de información de estado de canal inalámbrico para detectar una caída, ya que la velocidad de movimiento del cuerpo provoca cambios anormales en las señales de radio frecuencia [6].

  Para detectar caídas, en [22] se analizaron señales recolectadas de un radar de onda continua de frecuencia modulada multi antena y se extrajeron complejas características espaciotemporales para entrenar a una red neuronal convolucional (CNN, por sus siglas en inglés).

  En [23] se propone un novedoso sistema de detección de caídas basado en dispositivos WiFi. Demostraron que la información de estado de canal inalámbrico permite distinguir entre caídas y actividades similares exitosamente. Estos sistemas, que pueden ser WiFi o Bluetooth, funcionan detectando cambios repentinos en las señales inalámbricas provocados por diferentes actividades realizadas por las personas [6].

- Sistemas basados en fusión sensorial:

  Se ha demostrado que los sistemas basados en un solo sensor tienen una precisión baja y un porcentaje considerable de falsas alarmas, por lo que necesitan información extra para aumentar su precisión. Se pueden fusionar tipos de sensores de manera homogénea o heterogénea [6].

  Si se hace de manera homogénea se pueden fusionar sensores inerciales (entre sí), fusionar sensores ambientales, etc. Un ejemplo de fusión de sensores inerciales es [10], fusionando acelerómetro, giroscopio y magnetómetro, de 3 ejes ambos.

  Por otro lado, al mezclar diferentes tipos de sensores se obtiene fusión heterogénea. Un ejemplo es el ya citado [9], basado en el uso de acelerómetro (inercial), micrófono (ambiental) y cámara (por visión).

  Los sistemas basados en fusión sensorial, ya sea homogénea o heterogénea, siguen mostrando unos resultados de bajo rendimiento, además de otras limitaciones como redundancia o la dificultad para conseguir un algoritmo de fusión sensorial robusto [6].

2.1.2 Clasificación por tipo de algoritmo de análisis aplicado

En todos los sistemas de detección de caídas se miden características concretas de las que partirán los algoritmos de análisis para decidir si se produce una caída o no. En base a cómo se traten dichas características se puede

realizar una clasificación de los algoritmos en varios grupos. En este caso se dividen en tres grupos, tal como se relata en [6] y se detalla a continuación.

- Sistemas de detección mediante umbralización:

  Actualmente, la mayoría de los sistemas de detección o prevención de caídas utilizan algoritmos basados en umbralización de ciertas características extraídas previamente. En este caso se comparan las medidas obtenidas con los sensores con niveles de referencia configurados con anterioridad. La desventaja principal de estos sistemas es que ha debido elegirse correctamente el nivel de referencia con el que se comparan las medidas y esto no es una tarea sencilla, ya que un umbral elevado produciría una ausencia de caídas detectadas y un valor demasiado bajo produciría una gran cantidad de falsas alarmas [6].

  A su vez, los sistemas basados en umbralización se pueden dividir en sistemas de umbral fijo o umbral adaptativo.

  Debido a la baja complejidad computacional de los sistemas de umbral fijo, estos han sido ampliamente utilizados en la mayoría de los estudios actuales. Por ejemplo, en [24] se seleccionaron valores óptimos de umbral para la magnitud del vector suma y el ángulo de Euler para distinción de caídas frente a actividades de la vida diaria, basado en la curva de Característica Operativa del Receptor (ROC, por sus siglas en inglés) que ha sido comúnmente utilizada en estudios previos [6].

  Por otro lado, los sistemas de umbral adaptativo vienen a suplir las carencias principales de los sistemas de umbral fijo, como la baja especificidad. En [25] se propone un método de umbral adaptativo basado en gráficas de control multivariable. Este método tuvo un alto rendimiento en detección de caídas ya que consideró datos históricos individuales, es decir, fue un método específico unipersonal. En [26] se analizaron diferentes grupos de edad, género, altura y peso, para refinar los umbrales personalizados y conseguir un sistema de detección de caídas con una alta precisión [6].

- Sistemas de detección sin umbralización:

  Estos sistemas utilizan complejos algoritmos para diferenciar caídas de actividades de la vida diaria. En su mayoría utilizan algoritmos de aprendizaje máquina o de procesos estadísticos para la detección.

  En cuanto a los algoritmos de aprendizaje máquina utilizados, se puede encontrar el uso de kNN (*k-nearest neighbours*), *Support Vector Machine* (SVM), *Naïve Bayes*, *Hidden Markov Mode* (HMM), bosques aleatorios, lógica difusa, etc. En concreto, en [27] se desarrolla un algoritmo basado en HMM utilizando tan sólo un acelerómetro de 3 ejes. Se analizaron las aceleraciones en bruto usando distribuciones Gaussianas para estados ocultos para entrenar modelos HMM. En [28] se utiliza SVM para diferenciar caídas de actividades de la vida diaria. Se extrajeron 32 características a partir de información recolectada mediante un sensor Kinect para entrenar el clasificador propuesto [6].

- Sistemas de detección por fusión algorítmica:

  Recientemente, aparecen sistemas basados en la fusión de algoritmos basados en umbralización y/o algoritmos basados en otras técnicas, para aumentar la precisión del sistema y que contienen las ventajas de los métodos combinados. Se distinguen dos subcategorías, sistemas de fusión homogénea y fusión heterogénea [6].

  En fusión homogénea se pueden encontrar sistemas con varios algoritmos con distintos umbrales que votan para determinar si existe una caída como se propone en [29]. Una combinación de algoritmos de

aprendizaje máquina puede ser utilizada también para incrementar la precisión de un sistema de detección de caídas, como propone [30].

En fusión heterogénea, donde se combinan algoritmos basados en umbralización con algoritmos que no usan umbralización, se encuentran estudios como [13] que utiliza una combinación de umbralización y MKL (del inglés, *Multiple Kernel Learning*) o [31] que utiliza una combinación de umbralización y Estimación de la Densidad del Kernel.

Estos sistemas representan una prometedora línea de investigación para años próximos ya que demuestran resultados bastante alentadores en comparación con otro tipo de sistemas.

2.2 Resumen de sistemas existentes actualmente comercializados

Hoy en día existen diversos sistemas que proporcionan un mecanismo de detección y alerta ante caídas en esta parte de la población [32], que tras años de investigación y desarrollo han visto la luz en forma de productos comerciales o preparados para su incorporación al mercado.

Algunos de ellos son:

- Sense4Care-FATE (acelerómetro) (comercialmente expandido como Angel4) [33]
- Vigi'Fall (acelerómetro + detector de movimiento)(proyecto FallWatch cerrado) [34]
- Neat (acelerómetro) [35]
- Reloj Durcal (producto ampliamente comercializado) [36]
- Apple Watch (producto ampliamente comercializado) [37]

El problema de algunos de estos productos es el alto coste individual (Reloj Durcal/Apple Watch), la complejidad de instalación del sistema (Vigi'Fall) o la invasión de la privacidad.

2.3 Estructura del prototipo actual del SoM

El prototipo de laboratorio que implementa la detección de caídas desarrollada por los investigadores del Grupo de Ingeniería Biomédica de la Universidad de Sevilla, al que pertenecen los tutores, consta de un microcontrolador PIC18F2431 de Microchip, el acelerómetro LIS3LV02 de ST Electronics, el módulo de comunicación BLE RN4020 de Microchip, un oscilador de cristal de 32768 Hz y varios elementos pasivos.

Para la elección del microcontrolador (primer elemento principal) se tuvo en cuenta la necesidad del sistema, es decir, se necesitaba una MCU (*microcontroller unit*) con un consumo muy bajo, con un coste también relativamente bajo y que mantuviera una huella pequeña dentro del diseño del sistema completo.

Al respecto de esto, el PIC18F2431 tiene unas dimensiones de 6 x 6 mm en formato QFN, un muy bajo consumo con tecnología nanoWatt (en operación 900 uA reducible con modo reposo a aproximadamente 6 μA) y un coste dentro de requisitos de unos 7 € aproximadamente.

En el caso del acelerómetro (segundo elemento principal) se requería igualmente baja huella, bajo coste, un consumo suficientemente bajo en funcionamiento y una frecuencia de funcionamiento en torno a los 40-50 Hz, dado que es el rango de frecuencias que se demuestra en otras investigaciones sobre sistemas de detección de caídas que da mejores resultados.

El LIS3LV02 posee unas dimensiones en el empaquetado elegido (LGA16) de 7.5 x 4.4 mm, con un coste aproximado de 1.5 €, un consumo en operación de 650 μA y una frecuencia de funcionamiento de entre 40 y 2560 Hz (configurable).

Para poner de relieve el resultado obtenido con el uso de los componentes comentados para el diseño del prototipo del SoM, podemos analizar algunos detalles del sistema completo:

- Dimensiones: El prototipo consigue contenerse en una PCB de 29.5 x 30 mm.
- Consumo: Con el algoritmo validado en ejecución sin envío de datos tiene un consumo promedio de 5.4 mA.
- Frecuencia de funcionamiento del microcontrolador: 4 MHz.
- Frecuencia de funcionamiento del acelerómetro: 40 Hz.

3 INSTALACIÓN Y PREPARACIÓN DEL ENTORNO DE DESARROLLO

El IDE (del inglés, Integrated Development Environment) que se va a utilizar para el desarrollo del proyecto será MPLAB X IDE v5.45, un entorno de desarrollo de la empresa Microchip, fabricante del microcontrolador que utiliza el dispositivo del proyecto.

También se usará el compilador de C (lenguaje de programación) del mismo fabricante, MPLAB XC8 v1.34 (dado que el dispositivo incorpora un microcontrolador de 8 bits).

Por último, para facilitar parte del proceso de desarrollo del código del sistema, se instalará y usará una librería de periféricos genérica para microcontroladores de la familia PIC18 (PIC18F Peripheral library).

3.1 Instalación del entorno de desarrollo MPLAB X IDE v5.45

3.1.1 Descarga

Primero hay que descargarlo y para ello basta con acceder al enlace de la página de Microchip https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide y clicar sobre MPLAB X IDE Windows, en este caso, ya que se va a utilizar un ordenador con Windows. Si la descarga se realiza en un momento en el que se haya superado la versión comentada, se puede acceder a versiones anteriores clicando en el recuadro *Go to Downloads Archive* y seleccionando la versión específica.

3.1.2 Instalación

Para instalarlo se ejecuta el archivo descargado y aparecerá una ventana con los pasos a seguir para realizar la instalación.

Como referencia se puede seguir el proceso en el enlace https://microchipdeveloper.com/install:mplabx

3.2 Instalación del compilador MPLAB XC8 v1.34

3.2.1 Descarga

Para descargarlo basta con acceder al enlace https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-ecosystem-downloads-archive y en la sección *Language Tool Archives* seleccionar la versión v1.34 (WIN) (02/16/15).

Se utiliza esta versión en concreto porque es la más compatible con la librería de periféricos “PIC18F Peripheral Library” ya que, tras probar otras versiones más actuales, resultaron errores de compilación que se producían simplemente por una incompatibilidad entre estas y algunas de las funciones de la librería.

3.2.2 Instalación

Para instalarlo se ejecuta el archivo descargado y aparecerá una ventana con los pasos a seguir.

Como referencia se puede seguir el proceso en el enlace https://microchipdeveloper.com/install:xc8

El tipo de licencia utilizada o cualquier cuestión relacionada con ello será “Free”.

3.3 Instalación de la librería PIC18F Peripheral Library

3.3.1 Descarga

Para descargar la librería hay que acceder al enlace https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers, clicar en *View Downloads*, en *Compiler Downloads* > *Peripheral Libraries* > *Legacy Peripheral Libraries* > *PIC18F* y, en este caso, seleccionar la versión para Windows.

3.3.2 Instalación

Para instalar la librería hay que ejecutar el fichero descargado, indicando correctamente la ubicación de instalación, que en el caso de que el compilador esté instalado en <<C:\Program Files (x86)\Microchip>>, será <<C:\Program Files (x86)\Microchip\xc8\v1.34>>.

3.4 Creación del Proyecto

Para crear el proyecto en MPLAB X IDE, basta con seleccionar *File* > *New Project*.

En la ventana que aparece se escoge *Microchip Embedded* y *Standalone Project*.

El siguiente paso será seleccionar el microcontrolador asociado al proyecto. Para ello se elige la familia *Advanced 8-bit MCUs (PIC18)*, dispositivo *PIC18F2431* y en herramienta, por ejemplo, *Simulator*.

Luego se selecciona la versión del compilador XC8 instalado previamente, en el siguiente paso se proporciona un nombre y se clica en *Finish*.

3.5 Activación de la librería PIC18F Peripheral Library en el Proyecto creado

A continuación, para poder utilizar la librería en el proyecto, hay que acceder a las propiedades de este y dentro de la categoría *Conf: [default]*, dentro de *XC8 Global Options*, en *XC8 Linker* hay que activar la última opción visible llamada *Link in peripheral library* y ya se puede acceder a todo el conjunto de librerías de periféricos.

Para facilitar el acceso y evitar posibles errores a la hora de compilar los archivos fuente, se ha hecho una copia de los archivos fuente de la librería <<sw_spi.h>> (Software SPI) y <<usart.h>> (USART) en el directorio padre del directorio del proyecto. En el caso de la librería para comunicación SPI se han copiado todos los ficheros fuente que se encuentran en la carpeta <<SW_SPI>>, pero en el caso de la USART sólo se han copiado los elementos que no contienen un número en su nombre (ya que son los exclusivamente necesarios para el microcontrolador utilizado).

Una vez copiados, se añadirán al proyecto haciendo clic derecho en *Source Files* dentro del árbol de proyecto y seleccionando *Add Existing Items from Folder*. En la ventana que se abre (Figura 3.1) se selecciona *Add Folder* y se seleccionan las carpetas donde se encuentran los archivos fuente de cada librería (llamadas previamente, por ejemplo, USART y SW_SPI, como las carpetas originales desde las que se copiaron los ficheros fuente).

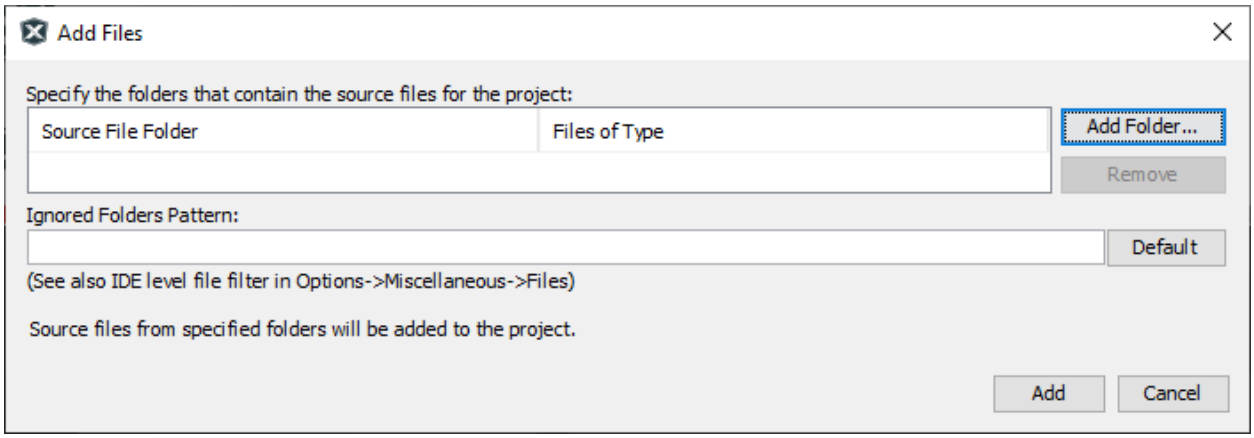


Figura 3-1. Ventana de configuración de ficheros fuente vinculados al proyecto.

4 CONFIGURACIÓN INICIAL DEL MICROCONTROLADOR PIC18F2431

Antes de comenzar a desarrollar las funcionalidades principales del proyecto, es necesario configurar los parámetros iniciales de funcionamiento del microcontrolador, referidos a: velocidades de ejecución, características como el WDT (Watch Dog Timer), las protecciones y reinicios automáticos, etc.

Para ello se va a crear un fichero independiente llamado <<configbits.c>> que se va a generar a partir de una herramienta propia del entorno de desarrollo como se muestra a continuación.

4.1 Uso de la herramienta Target Memory Views > Configuration Bits

Se accede a esta herramienta clicando en *Window > Target Memory Views > Configuration Bits* abriéndose así una pestaña nueva como si de un fichero se tratase.

Se pueden visualizar todos los parámetros pertenecientes a los registros de configuración del microcontrolador, tal y como se explica en la página 263 y sucesivas del datasheet del microcontrolador (PIC18F2431), apartado "23.1 Configuration Bits" [38].

En este caso, se establecen los siguientes parámetros, tal como están definidos en la versión en ensamblador del código del proyecto del grupo de investigación:

OSC = IRC, establecimiento del oscilador interno RC como reloj principal del sistema.

IESO = OFF, desactivación del modo internal/external switchover.

PWRTEN = ON, habilitación del Power-Up Timer.

BOREN = ON, habilitado el reinicio ante Brown-out (bajada de tensión de alimentación excesiva).

WDTEN = ON, habilitado el WDT.

WDPS = 256, configurado el post-escalado del WDT en 256.

WINEN = OFF, deshabilitada la ventana del WDT.

STVREN = OFF, deshabilitado el reinicio ante Stack Full/Underflow.

LVP = OFF, deshabilitado ICSP de baja tensión.

CP0, CP1, CPB = OFF, deshabilitado la protección de código de los bloques 0, 1 y Boot respectivamente.

WRT0, WRT1, WRTB, WRTC = OFF, deshabilitada la protección de escritura de los bloques 0, 1 y Boot y de los registros de configuración (Configuration Bits).

Una vez seleccionados los parámetros en la herramienta, se hace clic en *Generate Source Code to Output* y se abre debajo una pestaña con las configuraciones generadas. Una vez en esta pestaña, clic derecho en el texto y *Save As* para guardarlo en el directorio del proyecto junto con el fichero fuente principal como <<configbits.c>>.

A partir de ahora, cuando se compile el proyecto, este fichero se compilará también y servirá para cargar las configuraciones elegidas en el microcontrolador.

Y con esto, el proyecto está listo para empezar el desarrollo de las funcionalidades del sistema.

5 PROGRAMACIÓN Y TOMA DE CONTACTO CON EL MICROCONTROLADOR PIC18F2431

Antes de desarrollar las funcionalidades principales del sistema se comprobará la correcta programación del microcontrolador implementando un código sencillo que haga parpadear un led.

5.1 Conexionado entre el programador PICkit 3 y el microcontrolador PIC18F2431

Para la programación del PIC18F2431 se necesitará de una herramienta auxiliar llamada PICkit 3, que es un programador/emulador de depuración que se conecta al ordenador mediante un cable USB.

Este elemento debe conectarse siguiendo el esquema mostrado en la Figura 5-1.

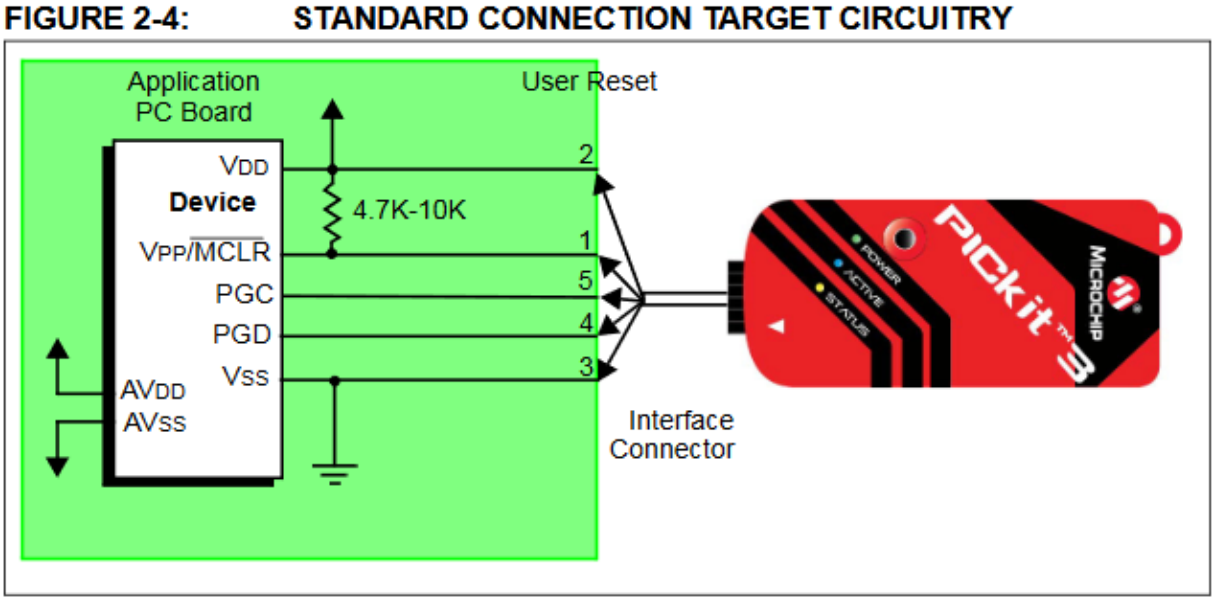


Figura 5-1. Circuito estándar de conexionado con el microcontrolador (Figura 2-4, pág. 20 en el documento "PICkit 3 User's Guide" Microchip Doc. Ref: DS51795B [39]).

En el caso de nuestro sistema, el programador irá conectado a una PCB (Printed Circuit Board) intermedia, mediante tres tiras de pines (con 3 pines cada una), utilizada sólo para la programación/depuración del microcontrolador en la que se conectará la PCB del SoM. Esta PCB intermedia tan solo hace la conexión entre las tiras de pines y un conector "MEC1-105-02" de 10 terminales, en el que se inserta la PCB del SoM.

Por consiguiente, la conexión entre programador y microcontrolador se hace en dos etapas, la primera sería similar a la mostrada en la Figura 5-1, pero limitándose a la conexión de los pines MCLR, VDD, VSS, PGD y PGC, y la segunda sería el conexionado que se puede observar en el recorte del esquemático del SoM, véase Figura 5-2, en cuanto a los terminales nombrados anteriormente.

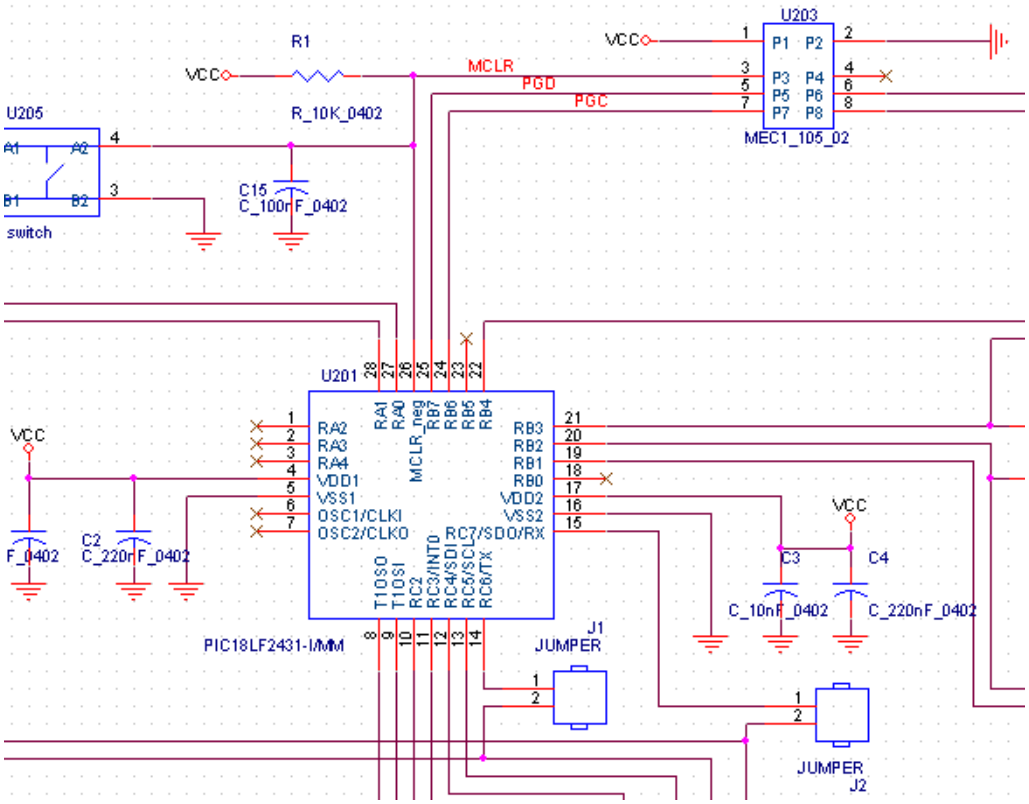


Figura 5-2. Recorte del esquemático del SoM donde se observa la conexión entre los terminales de programación en el conector U203 (esquina superior derecha) y los pines del microcontrolador destinados a la programación/depuración de este.

5.2 Configuración de MPLAB X IDE para el uso del programador PICkit 3

Con las conexiones debidamente satisfechas entre ambos dispositivos, el siguiente paso será configurar el programador en el IDE y para el proyecto en concreto.

Para ello, se debe conectar el PICkit 3 al ordenador sin permanecer conectado el SoM a la PCB intermedia, ya que el primer paso será configurar la alimentación proporcionada al sistema por el programador para evitar dañar el módulo BLE (Bluetooth Low Energy) RN4020 ya que éste tiene una limitación de 3.6 V en la tensión máxima de alimentación.

Para hacerlo, habrá que abrir las propiedades del proyecto y comprobar que en *Connected Hardware Tool* aparece el programador *PICkit3*. Tras ello, en *Categories* seleccionar *PICkit 3*, en *Option Categories* seleccionar *Power* y en *Voltage Level* la opción 3.0, asegurando previamente que esté activada la opción *Power target circuit from PICkit3*.

Y con eso ya es posible conectar el programador al SoM sin peligro de dañar el módulo BLE.

5.3 Modificación del hardware del SoM para pruebas y depuración por puerto serie

El próximo paso lógico sería implementar un ejemplo sencillo, como por ejemplo el parpadeo de un led, para comprobar que se realiza correctamente la programación del microcontrolador, pero teniendo en cuenta que el microcontrolador está integrado en el sistema SoM, no se dispone de acceso a todos los pines de éste, no haciendo posible la configuración y uso de pines no utilizados por el sistema en condiciones normales.

Existe, sin embargo, la posibilidad de utilizar dos pines (utilizados para la comunicación serie) cuya conexión está expuesta mediante jumpers (pads que permiten la conexión/desconexión, según se suelden o no, de dichos pines del MCU (Micro Controller Unit) con el módulo BLE o la conexión externa de la propia PCB del SoM).

Esto se puede hacer de dos maneras, soldando los jumpers y accediendo a dichos pines en la tira correspondiente de la PCB intermedia, lo cual se probó inicialmente y se comprobó que el módulo BLE interfería en la comunicación o uso de dichos pines, o dejando dichos jumpers sin unir y soldando dos cables a la parte conectada al MCU para tener acceso directo y no compartido a los pines RC6/TX y RC7/RX, siendo esta última la opción escogida para realizar la prueba con el led y la posterior comunicación por puerto serie del microcontrolador con la terminal (serie) del ordenador, como se observa en la Figura 5-3.

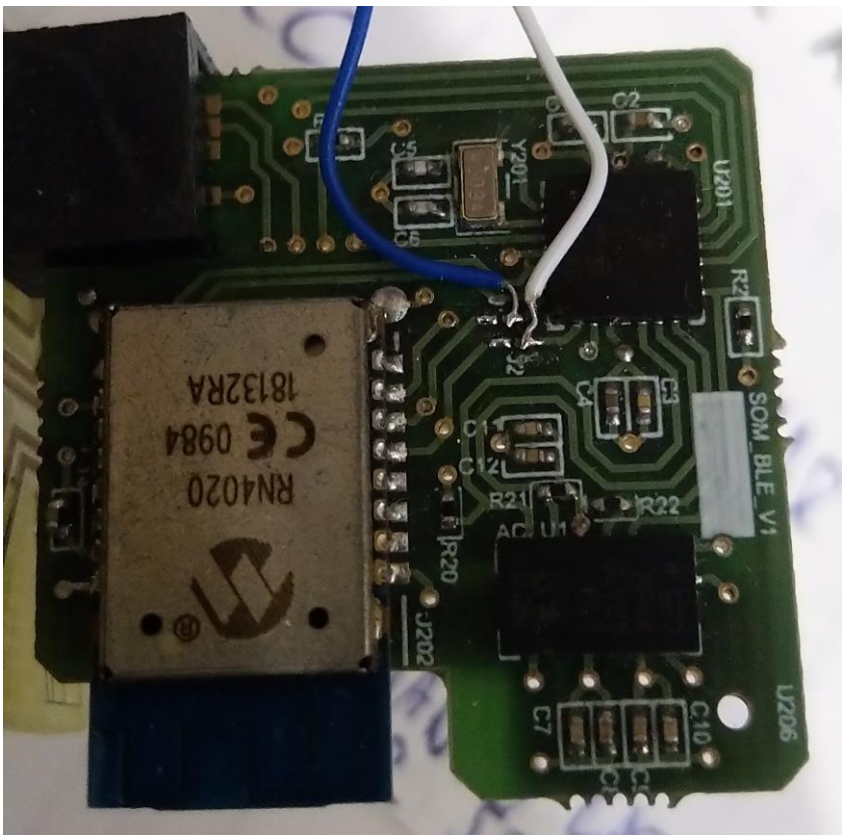


Figura 5-3. Cables soldados a los jumpers J1 y J2 para establecer conexión con los pines TX y RX del MCU.

5.4 Programación del microcontrolador PIC18F2431

Una vez se ha implementado y compilado (*Build Main Project*) sin errores un caso sencillo de parpadeo de un led que se conectara configurando el pin RC7/RX como GND "simulada" y el pin RC6/TX como pin de activación/desactivación del led (código a continuación del párrafo), basta con conectar el programador al USB del ordenador, si no lo estaba, y pulsar en la opción *Make and Program Device Main Project*, justo a la derecha de *Run* (triángulo verde). En este momento empezará el proceso de recompilado, de creación de los ficheros ejecutables (.hex) y de carga del contenido de dichos ficheros en la memoria flash del microcontrolador y, si termina correctamente, se habrá programado el código en el MCU y empezará a parpadear el led a la velocidad a la que se haya implementado.

```
#define LED LATCbits.LC6
#define SIMULATED_GND LATCbits.LC7

PORTC = 0x00; LATC = 0x00;
```

```
TRISCbits.RC6 = OUTPUT;
TRISCbits.RC7 = OUTPUT;

while(true){
        // LED TEST CODE // (blink every 1 s)
        LED = 0;
        __delay_ms(167);__delay_ms(167);__delay_ms(166);
        LED = 1;
        __delay_ms(167);__delay_ms(167);__delay_ms(166);
        ////////////////////
}
```

Este código se incluye comentado en el código del fichero *SoM_v1.c* añadido en el Anexo B.

A veces, ocurre que se visualiza un error en la consola de programación debido al conexionado del sistema completo e indica que no es posible comunicarse con el dispositivo objetivo (“target device”). En ese caso no hay otra solución que desconectar y reconectar, la PCB del SoM y el cable USB del programador PICkit 3.

6 LECTURA DE ACELERACIONES CON EL ACELERÓMETRO LIS3LV02

Una vez se ha comprobado el correcto funcionamiento de la programación del microcontrolador, el paso siguiente será conseguir hacer la lectura de aceleraciones y enviarlas por puerto serie para posteriormente realizar una gráfica para visualizar mejor y validar la lectura de aceleraciones.

El acelerómetro utilizado para extraer las aceleraciones producidas por el usuario del dispositivo SoM es el LIS3LV02 de STMicroelectronics.

Es un acelerómetro con un muy bajo consumo, característica imprescindible en el dispositivo desarrollado para conseguir una autonomía lo más elevada posible, una tasa de lecturas suficientemente grande (por encima de 40 Hz) y un tamaño pequeño (7.5 x 4.4 mm) [40].

El acelerómetro se comunicará por SPI implementado por software, ya que los pines de comunicación SPI hardware son utilizados para la comunicación por puerto serie (USART, Universal Synchronous/Asynchronous Receiver Transmitter). En este caso se utiliza la librería Software SPI incluida en la PIC18F Peripheral Library [41], comentada anteriormente en el capítulo 3, aunque habrá que realizar algunas modificaciones en tanto a correcciones y parametrización del caso concreto de uso del PIC18F2431.

6.1 Modificaciones de la librería Software SPI

Como se comentó en el capítulo 3, se ha copiado el directorio de los ficheros fuente de la librería al directorio raíz del proyecto, con lo cual se realizarán las modificaciones sobre esta copia de los ficheros. No obstante, se realizará también una pequeña modificación (fácilmente reversible) en el fichero de librería (*sw_spi.h*) original que se encuentra en el directorio <<C:\Program Files (x86)\Microchip\xc8\v1.34\include\plib>>.

Lo primero será abrir el fichero *sw_spi.h* y cambiar los pines utilizados para cada función SPI, mediante la modificación de las líneas *#define* que se encuentran al inicio de este en la parte del *else*, quedando la asignación CS=RB1, DIN=RB4, DOUT=RB3 y SCK=RB2. A continuación, sólo queda comentar la línea

```
#define MODE0
```

y descomentar o añadir al final del resto de modos una línea con

```
#define MODE3
```

Ahora se pasa a modificar el fichero fuente *opensspi.c*, que necesita una corrección de los parámetros en el modo 3. Las asignaciones deben ser SW_DOUT_PIN = 1 y SW_SCK_PIN = 1. Realmente el modo 2 también está mal implementado así que se aprovecha para corregir y poner SW_DOUT_PIN = 1. Y con esto, quedaría correctamente configurada la librería para su uso en la implementación final.

6.2 Librería propia para el LIS3LV02

Para facilitar el uso del LIS3LV02, es decir, la configuración, lecturas y escrituras en el código principal del programa, se ha decidido desarrollar una librería sencilla, sin implementar todas las funcionalidades que puede ofrecer el acelerómetro, pero con las necesarias para el proyecto que se va a desarrollar.

Se desarrollan principalmente funciones para escribir en los registros de configuración y funciones de lectura de las aceleraciones necesarias.

Esta librería hace uso a su vez de la librería Software SPI comentada anteriormente para realizar las comunicaciones por SPI entre el microcontrolador y el acelerómetro. Y también necesita la inclusión de las librerías *stdint.h* y *stdbool.h* para la declaración de variables con tipos estándar.

El fichero *lis3lv02.h* contiene principalmente macros que definen las direcciones de los registros del acelerómetro, enumeraciones con las opciones posibles de cada configuración de un registro y las declaraciones de todas las funciones que se definen en el fichero *lis3lv02.c*, junto con definiciones que simplifican los nombres de las funciones simplemente señalando el registro en el que escriben, por ejemplo, *lis3_set_ctrl_reg1*. Ambos ficheros permanecen adjuntos en el Anexo B.

Este fichero fuente contiene todas las funciones declaradas en el fichero .h que son:

- lis3_getID: implementa la lectura del registro WHO_AM_I del acelerómetro y devuelve el valor leído.
- lis3_begin: hace una llamada a la función *lis3_getID*, compara el valor devuelto con la macro LIS3_ID que está definida con el valor de la ID del acelerómetro (0x3A) y devuelve el resultado de la comparación.
- lis3_set_power_datarate: escribe la configuración deseada en el registro CTRL_REG1 del acelerómetro. En este caso, configura el encendido/apagado del acelerómetro y la frecuencia de muestreo de éste.
- lis3_set_scale_update_int_drdy_spimode_alignment: escribe la configuración deseada en el registro CTRL_REG2 del acelerómetro. Se pueden configurar la escala de aceleraciones (±2g/±6g), el modo de actualización de datos, la función del pin RDY/INT (entre Data-Ready o Interrupt), la habilitación de la generación de la señal Data-Ready, el modo de SPI (4 ó 3 conexiones) y la alineación de los datos (12 bit a la derecha/16 bit a la izquierda).
- lis3_set_filter_options: configura todas las opciones del filtro paso-alta, tales como señal de reloj, habilitación del filtro para la lectura normal de datos, para las interrupciones de caída libre o de detección de dirección y frecuencia de corte. Esta función no se utiliza a priori, ya que no se usa el filtro paso-alta, pero se ha implementado por si en un futuro el proyecto se encuentra con la necesidad de uso del filtro.
- lis3_reset_filter: función que realiza una lectura en el registro HP_FILTER_RESET provocando con ella un reinicio del filtro.
- lis3_read_status_reg: realiza la lectura del registro STATUS_REG que proporciona información sobre la existencia de datos nuevos o de sobreescritura de los datos. Esta función tampoco se utiliza.
- lis3_get_acc_x/y/z_h/l: todas las funciones de lectura de aceleraciones en cada eje (x, y, z) de un solo byte (high/low). Las funciones de byte bajo devuelven un entero sin signo y las de byte alto con signo.
- lis3_get_acc_x/y/z: lectura de los dos bytes de la aceleración de cada eje, byte a byte. Se realiza el shift necesario en cada byte para luego componer el dato a partir de cada byte (o parte de él).
- lis3_get_acc_x16/y16/z16: lectura de los dos bytes de la aceleración de cada eje, los dos bytes seguidos en una sola lectura y se compone el dato final a partir de cada byte leído.

6.3 Lectura de aceleraciones y visualización por puerto serie en tiempo real

Se pretende enviar por puerto serie el conjunto de aceleraciones leídas en los tres ejes de coordenadas, primero en un formato comprensible desde la propia terminal serie y más tarde en un formato que sirva para guardar el resultado en un fichero y poder importarlo posteriormente desde Matlab para realizar una gráfica con los datos.

Para ello primero habrá que configurar la USART mediante el uso de las funciones proporcionadas por la librería USART de la PIC18F Peripheral Library [42].

La forma de hacerlo es mediante tres variables (y dos funciones):

- config: que escribe en los registros TXSTA, RCSTA, PIR1 y PIE1, y que contendrá todas las configuraciones principales que en este caso serán interrupciones deshabilitadas en transmisión,

habilitadas en recepción, modo asíncrono, comunicación de 8 bits, recepción continua (que en modo asíncrono es simplemente activar la recepción) y selección de baudrate de alta velocidad.

- spbrg: con el valor que se debe pasar a los registros SPBRG y SPBRGH (en este caso) para definir el baudrate que se desea. En la Tabla 6.1 se puede observar una tabla extraída del datasheet del PIC18F2431 [43] con todas las opciones de configuración para conseguir un baudrate concreto y las fórmulas necesarias en cada caso.

- baudconfig: que escribirá en el registro BAUDCON las configuraciones que en este caso serán la habilitación del generador de baudrate de 16 bits y la desactivación del auto-baudrate.

Antes de proceder a utilizar las variables para configurar la USART hay que realizar una llamada a la función *CloseUSART* para asegurar que está cerrada antes de configurarla.

Con las dos primeras variables como parámetros se utiliza la función *OpenUSART* que establecerá los valores de cada uno de los registros comentados para efectuar todas las configuraciones seleccionadas. Posteriormente sólo queda llamar a la función *baudUSART* con la variable *baudconfig* como parámetro para configurar la velocidad de comunicación y tener la funcionalidad de puerto serie lista para su uso.

TABLE 20-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-Bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

Tabla 1. Configuraciones posibles y fórmulas para calcular el valor que deben tener los registros SPBRGH y SPBRG para un baudrate deseado (Tabla 20-1, pág. 221 del documento "PIC18F2431 Datasheet" Microchip Doc. Ref: DS39616D [43])

Una vez configurada la comunicación por puerto serie, se realiza la configuración del acelerómetro mediante el uso de *lis3_set_ctrl_reg1*, para encender el acelerómetro y definir una tasa de muestreo de 40 Hz, y *lis3_set_ctrl_reg2*, para configurar la escala completa en 6g, la actualización continua de datos, asignación al pin RDY/INT la señal de Data-Ready, la activación de la generación de dicha señal, el modo SPI con 4 terminales y el alineamiento de 16 bits a la izquierda.

A continuación, se realiza una espera de un segundo, encadenando varias veces la función *__delay_ms* ya que ésta tiene un límite de 167 en el valor que se le pasa como parámetro (en esta versión del compilador, puesto que se probó con versiones posteriores y funcionaba sin esa limitación).

Y se realiza la comprobación inicial de que se lee el registro WHO_AM_I del acelerómetro correctamente haciendo una llamada a *lis3_begin* haciéndola condición para salir de un bucle while. Mientras no se salga del bucle while, se envía por puerto serie el mensaje "Error initializing LIS3LV02", con las siguientes dos líneas de código:

```
while(BusyUSART());
putrsUSART("Error initializing LIS3LV02\n");
```

Si se sale del bucle (o ni siquiera se entra), se produce una espera idéntica a la recién descrita y se envía por puerto serie el mensaje "LIS3LV02 started successfully", tras lo cual se entra en el bucle principal del programa (implementado con un bucle while(true)).

Dentro del bucle principal, en este ejemplo en concreto, se realiza primero la lectura del byte más significativo de las aceleraciones de cada eje y se aplica un desplazamiento de 4 bits a la izquierda y luego se realiza la lectura del byte restante de las aceleraciones aplicando un desplazamiento de 4 bits a la derecha, ya que los 4 bits menos significativos, según dice el datasheet [40], contienen ruido. Por último, se realiza una operación OR entre ambos bytes y se reescala multiplicando por 50/17, lo cual es la fracción irreducible del factor 1000 mg / 340 LSb que indica la sensibilidad del acelerómetro, dato que aparece en la página 12/48, Tabla 4, *"Table 4. Mechanical characteristics @ Vdd=2.5 V, T=25ºC unless otherwise noted (1)"*, del datasheet del acelerómetro [40].

A continuación, sólo queda enviar por puerto serie las aceleraciones, para lo cual se usará la función *sprintf* y una variable auxiliar donde guardar el texto a enviar por puerto serie para posteriormente pasar la variable como parámetro de la función *putsUSART*.

Durante las comprobaciones de lectura de aceleraciones se ha usado un formato similar al mostrado a continuación:

acc_x: 256 acc_y: 129 acc_z: 874

Más tarde, para visualizar los cambios de aceleraciones en cada eje mediante una gráfica en Matlab (véase la Figura 6-2) se ha usado una variable extra a modo de contador/tiempo (enviada en primer lugar) y el formato siguiente:

12 256 129 874

Así, el programa de Matlab usará el primer valor como marca de tiempo y los otros tres serán las aceleraciones en el eje *X*, *Y* y *Z*.

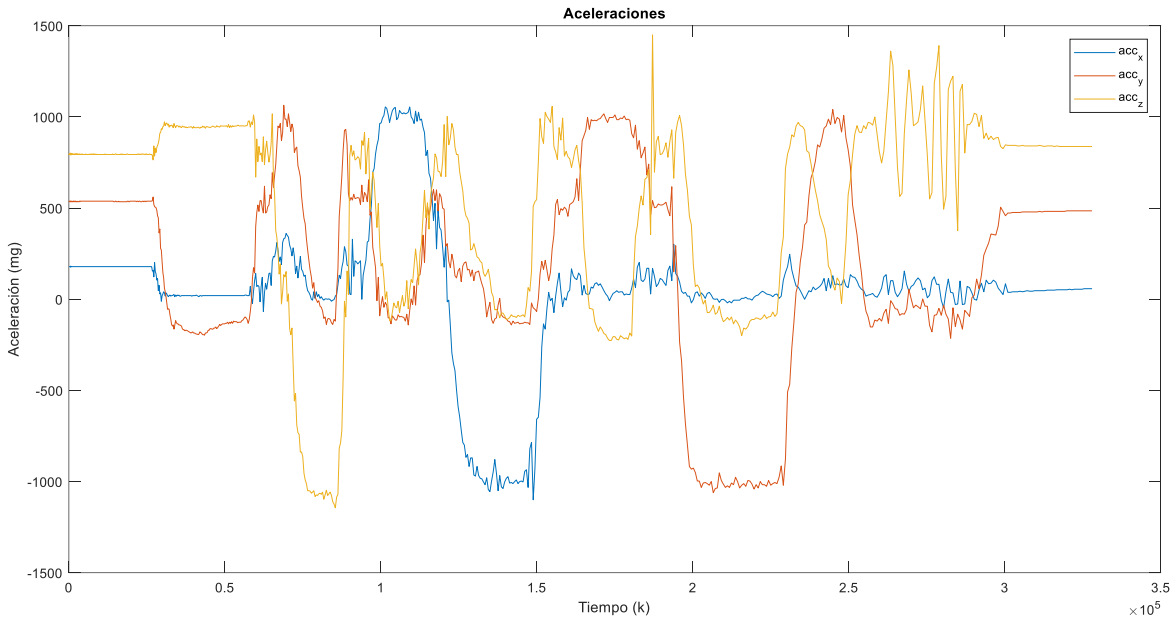


Figura 6-1. Gráfica de aceleraciones generada con Matlab a partir del fichero que contiene las lecturas de aceleraciones enviadas por puerto serie al ordenador durante un test moviendo el SoM en los tres ejes.

7 PROCESADO DE ACELERACIONES Y ALGORITMO DE DETECCIÓN DE IMPACTO

Dado que para el cometido de este proyecto no se necesita tanta resolución en las aceleraciones (debido a que las lecturas obtenidas en los impactos reales se distancian suficientemente de las obtenidas cuando estos no se producen), se puede ignorar la información que proporciona el byte menos significativo de aceleraciones (el medio byte), con lo cual se realizarán todos los cálculos utilizando únicamente el byte más significativo de aceleraciones.

Ello conlleva que, a diferencia del ejemplo anteriormente comentado, solo se realice la lectura del byte más significativo y se tome directamente como la medida de aceleración, realizando todas las operaciones sobre este dato.

Los resultados obtenidos se mostrarán en diversas gráficas generadas con dos herramientas, Matlab (MathWorks) y Arduino IDE Serial Plotter (Arduino S.r.l.).

La obtención de gráficas en Matlab se realizará usando datos obtenidos previamente mediante comunicación por puerto serie y guardados en un fichero que se cargará en Matlab mediante un script.

La herramienta Arduino IDE Serial Plotter permite realizar la visualización de los datos en tiempo real de manera muy sencilla, simplemente manteniendo un formato de datos concreto durante el envío de estos por puerto serie.

7.1 Procesado de aceleraciones

A la hora de representar las aceleraciones y cualquier otro dato calculado a partir de estas, si se quieren unidades de medida del SI (Sistema Internacional) y sus múltiplos, será necesario realizar el desplazamiento de bits oportuno (4 bits a la izquierda) y la conversión anteriormente comentada de LSb a mg (multiplicando por 1000/340 [mg/LSb]).

Pero antes de realizar las representaciones o hacer ningún otro cálculo basado en las aceleraciones, habrá que realizar un filtrado de éstas para tratar de eliminar la componente permanente de la gravedad.

Para ello se utiliza un filtro supresor de continua que tiene la siguiente formulación en el dominio Z:

$$H(z) = \frac{1}{2}\left(1 - z^{-1}\right) ,$$

siendo su implementación en el código:

$$Acc_{f,i}(n) = \frac{1}{2}\left(Acc_i(n) - Acc_i(n-1)\right) , \qquad i = x, y, z .$$

y cuya respuesta en frecuencia se muestra en la Figura 7-1

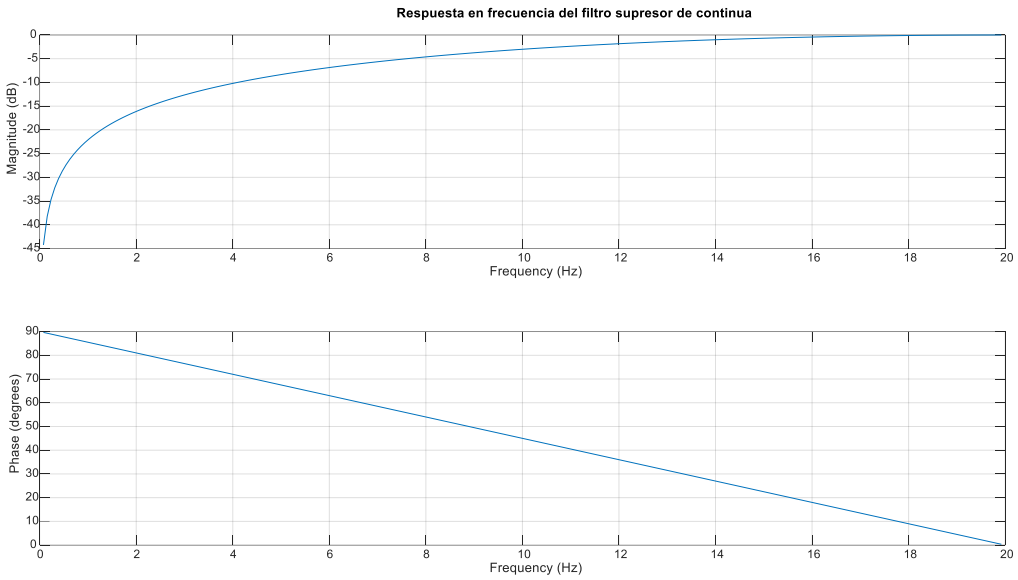


Figura 7-1. Respuesta en frecuencia del filtro supresor de continua.

Se puede comprobar el resultado del filtrado en la gráfica de la Figura 7-2:

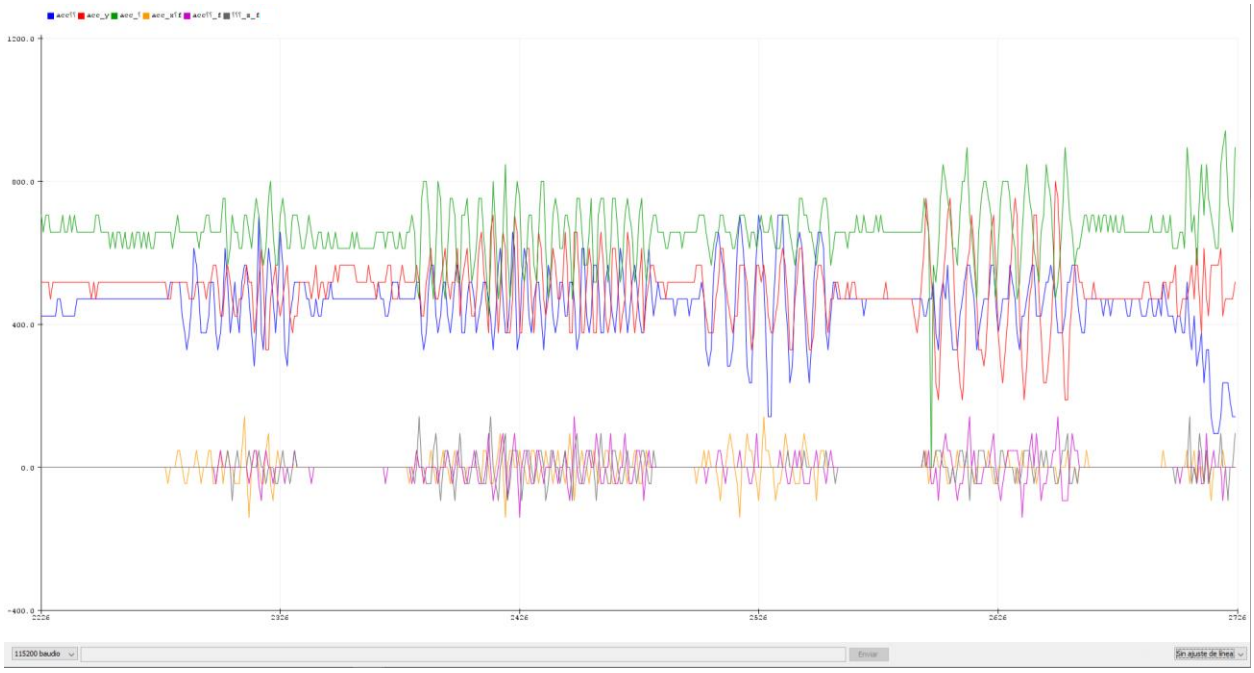


Figura 7-2. Gráfica de aceleraciones en bruto (x: azul, y: rojo, z: verde) y aceleraciones tras aplicar el filtro supresor de continua (xf: naranja, yf: violeta, zf: gris), generadas en tiempo real en la herramienta Arduino IDE Serial Plotter.

En la parte superior se encuentran las aceleraciones sin filtrar en torno a los 600 mg (se ha intentado colocar el dispositivo en una orientación en la que los tres ejes estuvieran más o menos centrados en el mismo nivel de

aceleración permanente) y en la parte inferior se visualizan los picos de aceleración filtrada, es decir, sólo los incrementos de aceleración (por eso se hayan centrados en el cero).

Una vez analizado el filtrado que se realizará a las aceleraciones, se puede describir la estructura del bucle principal del programa, incluido el algoritmo de detección de impacto implementado en éste.

7.2 Muestreo previo

En el caso del sistema a desarrollar en el proyecto, tras realizar la comprobación de que la conexión con el acelerómetro es correcta y antes de entrar en el bucle principal del programa, se realiza una lectura de 31 muestras de aceleraciones sin aplicar el algoritmo de detección (véase Figura 7-3), con el objetivo de tener datos previos suficientes para que el algoritmo realice su función correctamente y que el acelerómetro se estabilice para dar lecturas correctas de aceleración.

Esto se realiza en el siguiente bucle do-while:

```
do{
    if(acc_data_rdy){
        acc_data_rdy = false;
        SPI_data_capture();
        indexes_update();
        system_settled_counter--;
        CLRWDT();
    }
}while(system_settled_counter);
```

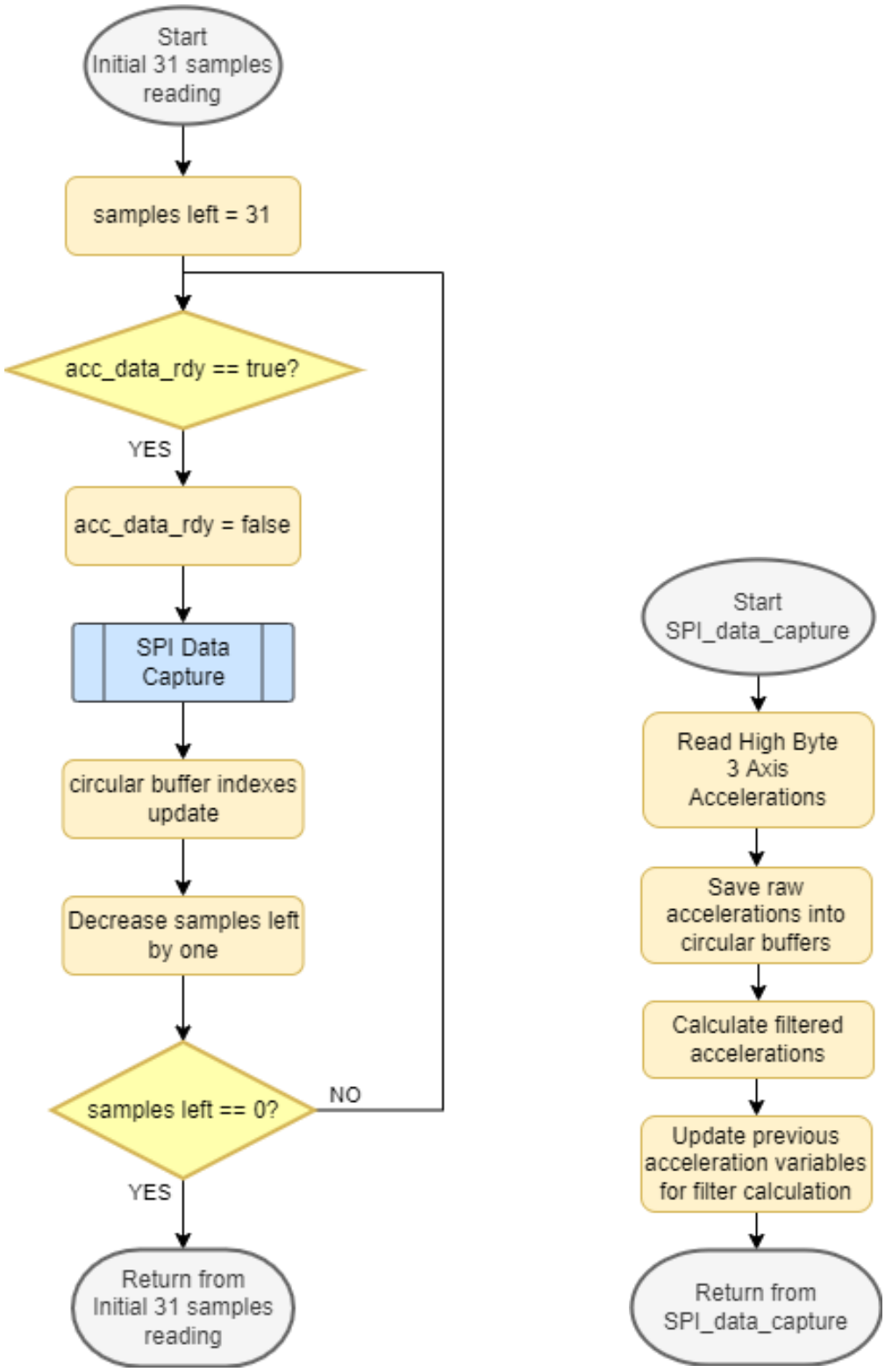


Figura 7-3. Diagramas de flujo del muestreo previo de 31 muestras (izquierda) y de la captura de aceleraciones por SPI del microcontrolador (derecha).

Donde, si hay nuevas muestras de aceleraciones pendientes de leer *(acc_data_rdy = true*), se hace la captura de aceleraciones y se disminuye en uno el contador de 31 muestras. Este bucle se ejecuta hasta que el contador llega a cero (Figura 7-3 izquierda).

En cada iteración se hace uso de la función propia *SPI_data_capture*, que realiza la lectura de aceleraciones, guarda los valores leídos en bruto en un buffer circular (por cada eje) y aplica el filtrado anteriormente analizado (Figura 7-3 derecha).

```
void SPI_data_capture(){
    //Read high byte accelerations
    acc_x = lis3_get_acc_x_h();
    acc_y = lis3_get_acc_y_h();
    acc_z = lis3_get_acc_z_h();
    //Store accelerations in the circular buffers
    acc_x_h_buffer[x_index] = acc_x;
    acc_y_h_buffer[y_index] = acc_y;
    acc_z_h_buffer[z_index] = acc_z;
    //Apply filter to accelerations
    acc_x_f = (acc_x - prev_acc_x)/2;
    prev_acc_x = acc_x;
    acc_y_f = (acc_y - prev_acc_y)/2;
    prev_acc_y = acc_y;
    acc_z_f = (acc_z - prev_acc_z)/2;
    prev_acc_z = acc_z;
}
```

Tras la lectura de dichas muestras iniciales, se procede a ejecutar el bucle principal.

7.3 Bucle principal: Algoritmo de Detección de Impacto

En el bucle principal, si existen nuevas muestras de aceleraciones, se ejecuta el algoritmo de detección de impacto, cuyo diagrama de funcionamiento se muestra en la Figura 7-4.

El algoritmo de detección de impacto consta de cinco partes diferenciadas (en orden de ejecución):

1. Lectura de las aceleraciones y filtrado de ellas mediante la función *SPI_data_capture* anteriormente comentada.

2. Comprobación de que se hayan superado los umbrales de aceleración impuestos por el algoritmo mediante las funciones *upper_acc_x/y/z*.

3. Cálculo de la energía a partir de las aceleraciones filtradas pertenecientes a distintos momentos de muestreo.

4. Comprobación de que se hayan superado los umbrales de energía impuestos por el algoritmo mediante las funciones *upper_energy_x/y/z*.

5. Determinación de si se ha detectado un impacto a partir de las condiciones de superación de los umbrales de aceleración y energía en alguno o varios de los ejes.

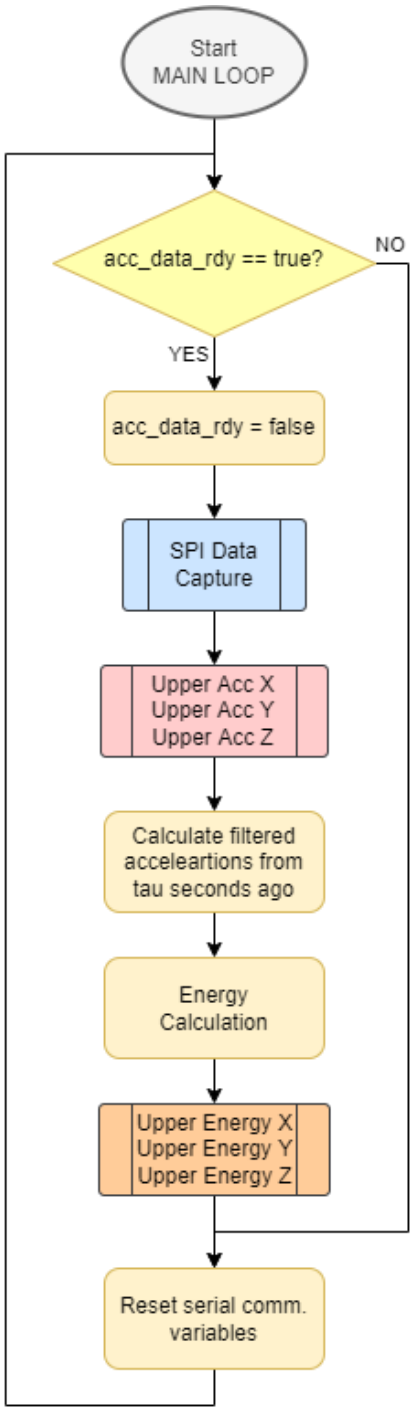


Figura 7-4. Diagrama de flujo del bucle principal del programa.

Antes de entrar en detalle en cada una de las partes, se hace necesario explicar el funcionamiento del temporizador TMR1 del microcontrolador y por qué se utiliza.

El temporizador TMR1 se utiliza para saber el momento en el que se produce un evento dentro del algoritmo y para descartar señales cuando ha pasado un periodo de tiempo suficiente para ello.

Es un temporizador de 16 bits que se incrementa desde 0000h hasta FFFFh, produciendo un *overflow* en ese momento y retornando su valor a 0000h [44]. Cuando se produce dicho *overflow*, y, gracias a la configuración de interrupciones para el TMR1 en el código, salta la interrupción, el manejador hace que cambie de valor la

variable booleana *overflow_parity,* permitiendo conocer en qué "vuelta" se encuentra la cuenta del temporizador en cualquier momento.

Como el oscilador que usa el temporizador es un cristal que funciona a 32768 Hz, la cuenta total antes de producirse el *overflow* será de 2 s. Con lo que, la variable *overflow_parity* nos permite saber en un momento dado si el valor del temporizador es del tramo 0 – 2 s o si pertenece al tramo 2 – 4 s. En realidad, no sabremos si un valor del temporizador pertenece a tramos superiores a 4 s, pero, dada la naturaleza del uso de ese valor en nuestro sistema, no se hace necesaria esa distinción, como se verá más adelante.

Dicho esto, se pasa a analizar cada una de las partes del algoritmo.

7.3.1 Comprobaciones de superación de umbrales (2 y 4)

En los casos 2 y 4 se cuenta, por cada eje, con un flag que señala si se ha superado el umbral correspondiente y con una estructura de datos para señalizar el momento en que se superó dicho umbral. Esta estructura, que define un nuevo tipo *timepoint_t*, cuenta con dos miembros, *tmr_data*, que almacena el valor del byte más significativo del temporizador TMR1, y *ov_parity* (overflow parity), que se usa para señalizar en qué "vuelta" se toma dicha medida de tiempo.

En el caso de *upper_acc_x* (y es igual para el resto de los ejes) se comprueba si el umbral de aceleración ha sido superado y si es así, el flag *acc_x_gt_th* pasa a valer *true*, se almacena el valor de TMR1H en *tmr_data* y se almacena el valor de *overflow_parity* en *ov_parity*, ambos miembros de la variable de tipo *timepoint_t acc_x_gt_th_timepoint.* En caso contrario, se revisa si ha pasado el tiempo suficiente (70.31 ms) para borrar el flag *acc_x_gt_th* y es aquí donde entra en juego el uso de *ov_parity*.

Si *acc_x_gt_th_timepoint.ov_parity* es **igual** a *overflow_parity* (caso señalado en rojo en la Figura 7-5) quiere decir que el valor de *acc_x_gt_th_timepoint.tmr_data* y el valor actual de TMR1H pueden restarse directamente para calcular el tiempo transcurrido.

Si *acc_x_gt_th_timepoint.ov_parity* es **diferente** a *overflow_parity* (caso señalado en verde) quiere decir que el valor de *acc_x_gt_th_timepoint.tmr_data* pertenece a un ciclo anterior al valor actual de TMR1H con lo que para calcular el tiempo transcurrido habrá que sumar FFh al valor actual de TMR1H antes de poder realizar la resta.

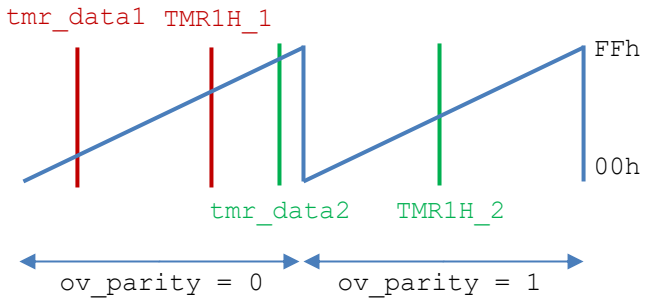


Figura 7-5. Variación del valor del registro TMR1H con el tiempo (en azul), tiempos en los que se supera el umbral (tmr_data1 y tmr_data2) y valores actuales del registro (TMR1H_1 y TMR1H_2) cuando se revisan las condiciones para borrar el flag de umbral superado.

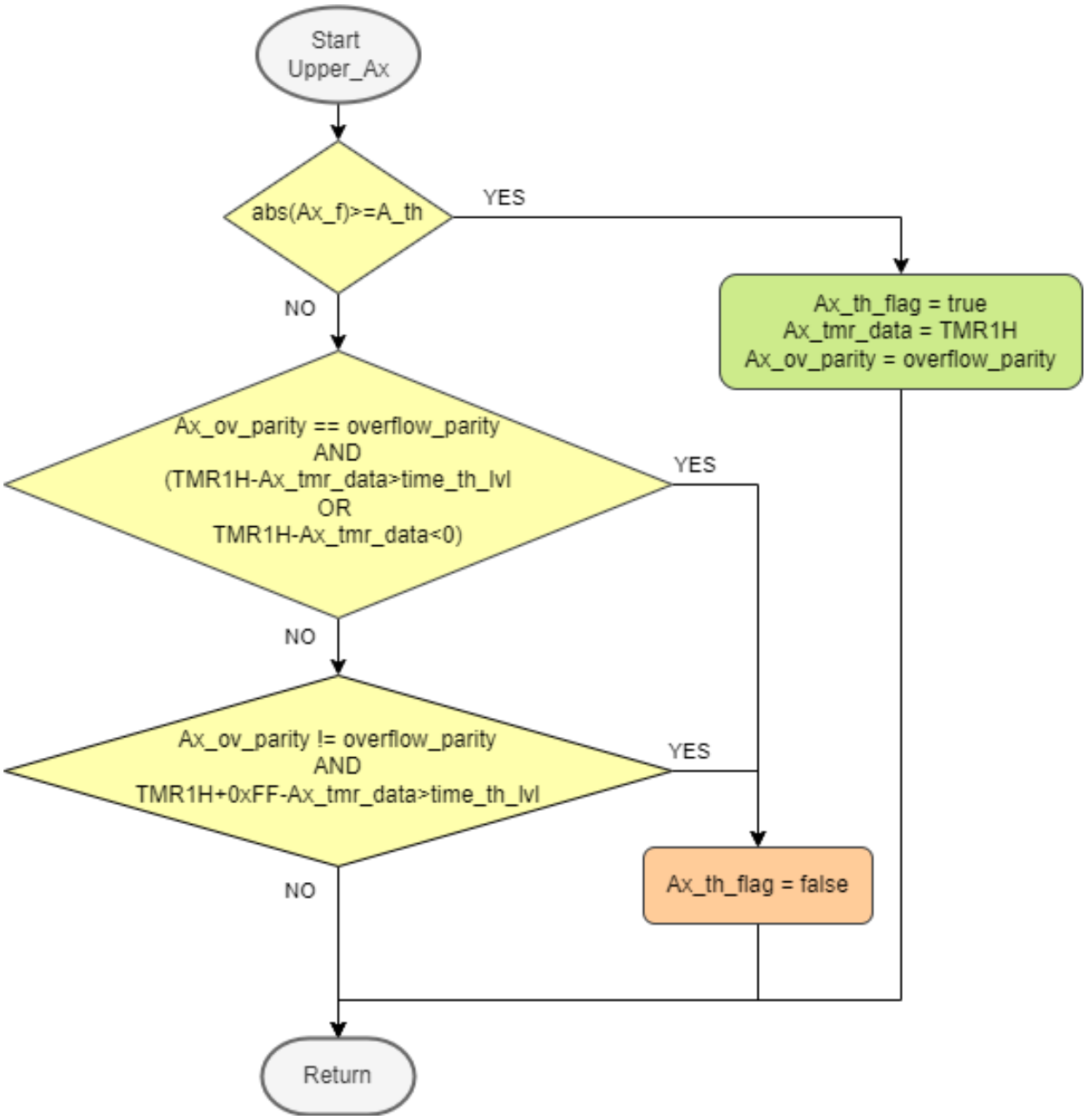


Figura 7-6. Diagrama de flujo que explica el funcionamiento de la función upper_acc_x (siendo el mismo para el resto de los ejes y para los tres ejes en el caso de la energía).

En el caso de *upper_energy_x* (y es igual para el resto de los ejes) se comprueba si el umbral de energía ha sido superado y si es así, el flag *en_x_gt_th* pasa a valer *true*, se almacena el valor de TMR1H en *tmr_data* y se almacena el valor de *overflow_parity* en *ov_parity*, sólo que en esta ocasión se realiza sobre la variable *en_x_gt_th_timepoint*. En caso contrario, se revisa si ha pasado el tiempo suficiente (70.31 ms) para borrar el flag *en_x_gt_th*.

Como el incremento de tiempo necesario para el borrado de flags es de 70.31 ms, muy inferior a los 2 s tras los cuales se produce el reinicio del TMR1 y se cambia el valor de *overflow_parity*, no es necesario diferenciar entre ciclos pares, ya que tras pasar de un ciclo a otro y finalizar ese segundo ciclo se habrá borrado el flag (y, por tanto, dejará de importar el valor de *overflow_parity,* ya que la próxima vez que se active el flag esta variable guardará un nuevo valor en *ov_parity*).

7.3.2 Cálculo de la energía (3)

Para el cálculo de la energía son necesarios previamente los datos de energía de la iteración inmediatamente anterior, la aceleración filtrada del instante actual y la aceleración filtrada del instante correspondiente a tau segundos antes.

Para ello, se calcula, antes del cálculo propio de la energía, la aceleración filtrada de hace tau segundos con los valores de aceleración en bruto del buffer circular. Como si estuviéramos calculando la aceleración filtrada en el instante actual, pero cogiendo los valores Acc(n-tau) y Acc(n-tau-1), de la forma:

$$Acc_{f,i}(n-tau) = \frac{1}{2}\left(Acc_i(n-tau) - Acc_i(n-tau-1)\right) , \qquad i = x, y, z .$$

Implementado en código, en el caso del eje X, con:

```
acc_x_f_tau = (acc_x_h_buffer[X_INDEX_MINUS_TAU] –
acc_x_h_buffer[X_INDEX_MINUS_TAU_MINUS_1])/2;
```

Con ese dato obtenido, pasamos a realizar el cálculo de la energía en el instante actual, tal como se muestra:

$$E_i(n) = E_i(n-1) + \left\|Acc_{f,i}(n)\right\|^2 - \left\|Acc_{f,i}(n-tau)\right\|^2 , \qquad i = x, y, z .$$

Implementado en código, realizando el cuadrado mediante las operaciones exp() y log() para que sea más eficiente computacionalmente (se ahorra casi 1 ms), en el caso del eje X con:

```
energy_x = energy_x + (acc_x_f * acc_x_f) – (acc_x_f_tau *
acc_x_f_tau);
```

7.3.3 Detección de impacto (5)

Finalmente, en el último paso del algoritmo se comprueba si se ha cumplido alguna (o varias) de las condiciones necesarias para alertar de la detección de un impacto.

Si se ha superado el umbral de aceleración y el de energía y se mantienen a la vez activos ambos flags de uno o más ejes (y no se borran al pasar el tiempo definido en *time_threshold_lvl*) entonces la variable *impact_detected* cambiará su valor a true, indicando que se ha detectado un impacto.

```
impact_detected = (acc_x_gt_th & en_x_gt_th) | (acc_y_gt_th &
en_y_gt_th) | (acc_z_gt_th & en_z_gt_th);
```

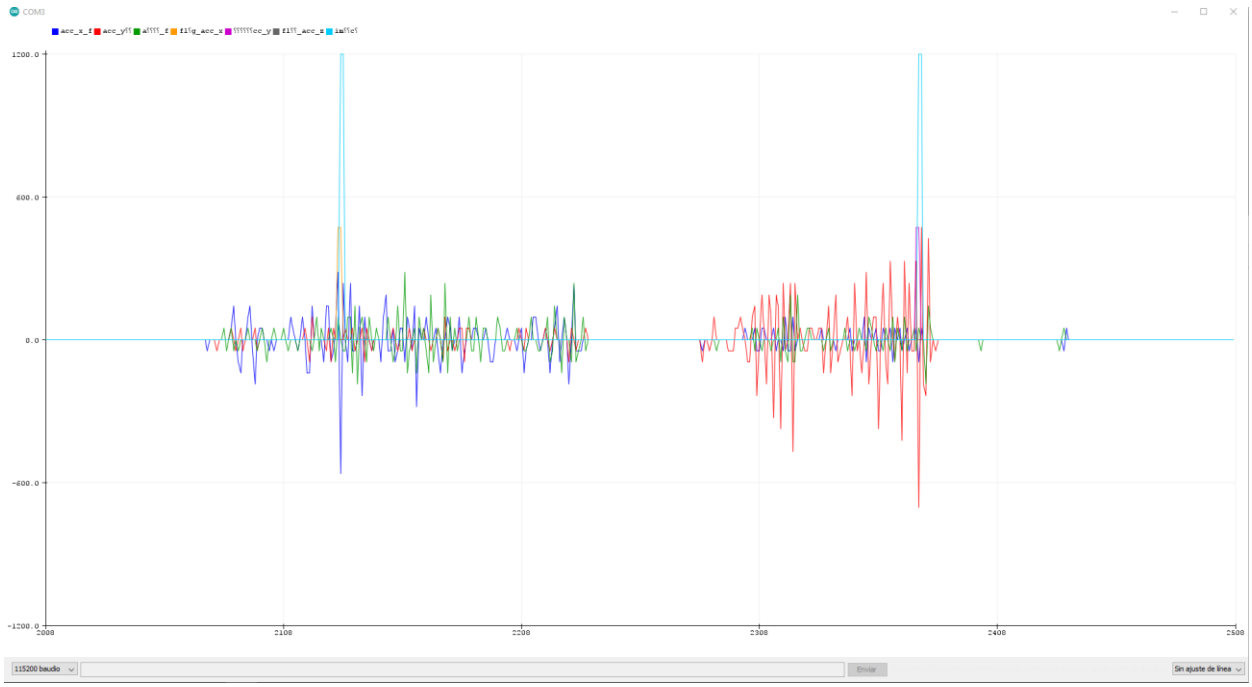


Figura 7-7. Gráfica de aceleraciones filtradas, flags de superación del umbral de aceleración y flag de detección de impacto, generada en tiempo real en la herramienta Arduino IDE Serial Plotter.

Como se aprecia en la Figura 7-7, en este ejemplo tenemos dos casos en los que se detecta un impacto:

- En el primero, se ven mayoritariamente cambios en la aceleración filtrada en el eje X (azul), acompañados de un pico perteneciente al flag de superación del umbral de aceleración en el eje X (naranja) y de otro pico perteneciente al flag de detección de impacto (cian).

- En el segundo, se ven mayoritariamente cambios en la aceleración filtrada en el eje Y (rojo), acompañados de un pico perteneciente al flag de superación del umbral de aceleración en el eje Y (violeta) y de otro pico perteneciente al flag de detección de impacto (cian).

Se puede comprobar en ambos casos que el flag de superación del umbral de aceleración vuelve a cero (false) cuando pasa el tiempo suficiente (70.3 ms) para dar por inválido este dato.

8 VALIDACIÓN DEL ALGORITMO CON DATOS EXPERIMENTALES

Para validar el algoritmo de detección de impacto se usarán datos extraídos de diferentes experimentos reales, llevados a cabo mediante la colocación de un prototipo de SoM en la espalda de voluntarios con diferentes características físicas de peso, género y edad, y realizando una serie de acciones con diferentes movimientos corporales. Algunos de estos experimentos incluyen una caída y otros no, y se comprobará que el algoritmo realiza una detección de las caídas experimentales reales correcta y no detecta como caídas los experimentos con picos de aceleración que no son caídas reales.

Se parte de datos de aceleración de 11 experimentos:

1. Andar normal sobre suelo duro.
2. Subir escaleras.
3. Bajar escaleras.
4. Agacharse y levantarse doblando las rodillas para coger un objeto.
5. Sentarse en una silla despacio.
6. Levantarse de una silla.
7. Salto vertical sobre suelo blando.
8. Caída de rodillas sobre suelo blando.
9. Caída de rodillas sobre suelo blando y tirarse al suelo.
10. Caída horizontal desde banco bajo sobre suelo blando.
11. Caída desde banco sentado a rodilla y apoya las manos (4 patas) suelo blando.

En cada experimento se cuenta con un fichero de aceleraciones “aceleraciones_X.m” con el siguiente formato:

```
10 -    ax(1)=3;
11 -    ay(1)=22;
12 -    az(1)=-3;
13 -    ax(2)=3;
14 -    ay(2)=22;
15 -    az(2)=-3;
16 -    ax(3)=3;
17 -    ay(3)=21;
18 -    az(3)=-4;
```

Figura 8-1. Formato original de las aceleraciones en bruto.

Para reorganizar las aceleraciones de manera que sea más cómodo el uso de estas para la validación en el código del SoM, se ha implementado un breve script de Matlab nombrado “accs2arrays.m” que imprime las aceleraciones en forma de vector para copiar las tres líneas y pegarlas directamente en el fichero C del SoM.

```
fprintf('ax = {')
for i = ax
    fprintf("%d,", i);
end
fprintf('\b}\n')

fprintf('ay = {')
for i = ay
    fprintf("%d,", i);
end
fprintf('\b}\n')

fprintf('az = {')
for i = az
    fprintf("%d,", i);
end
fprintf('\b}\n')
```

Figura 8-2. Script de Matlab “accs2arrays.m” para reformatear las aceleraciones en bruto en forma de arrays.

```
>> aceleraciones_1
>> accs2arrays
ax = {3,3,3,4,3,-1,-1,2,4,2,-1,-4,-5,-4,-3,-1,0,0,0,-1,-2,-2,-3,-3,-2,-2,-2,-3,-3,-3,-3,-4
ay = {22,22,21,21,24,26,23,23,25,25,22,19,20,22,24,23,21,20,20,21,21,21,21,21,21,20,19,20,
az = {-3,-3,-4,-4,-2,-1,-1,1,1,2,2,2,1,0,0,0,0,0,0,0,0,-1,-1,-2,-2,-2,-3,-3,-3,-4,-5,-3,-2
>>
```

Figura 8-3. Aceleraciones en bruto del Experimento 1 reformateadas en forma de arrays.

Adicionalmente, se implementa otro script en forma de función de Matlab, “acc_file_plot.m”, para realizar las gráficas de aceleraciones, energías y flags de umbralización y detección de impacto, a partir de cada uno de los ficheros generados con los datos recibidos por puerto serie, enviados por el microcontrolador, por cada uno de los experimentos.

Como método de comprobación extra se implementa un script en Matlab nombrado “plot_accs.m”, que incluye el algoritmo de detección de impacto, que recibe los vectores de aceleración en bruto y realiza la graficación de los resultados obtenidos tras el cálculo y ejecución del algoritmo. Esto permite comprobar que los resultados obtenidos por el algoritmo implementado en el microcontrolador PIC18F2431 son cualitativamente los mismos que los procesados por Matlab.

A continuación, se visualizan las gráficas que muestran la detección de impacto divididas en dos por cada experimento. Una primera gráfica que contiene aceleraciones en bruto en los tres ejes, aceleraciones filtradas en los tres ejes y energías en los tres ejes calculadas a partir de dichas aceleraciones. Y otra gráfica que contiene los flags de superación de umbrales de aceleración en los tres ejes, los de superación de umbrales de energía en los tres ejes y el flag de detección de impacto.

8.1 Experimento 1 – Andar normal sobre suelo duro

En este caso no se debe detectar impacto, aunque sí se detectarán superaciones del umbral de energía. Se producen variaciones de aceleración suficientes como para hacer valorables las aceleraciones filtradas y por tanto las energías calculadas a partir de éstas.

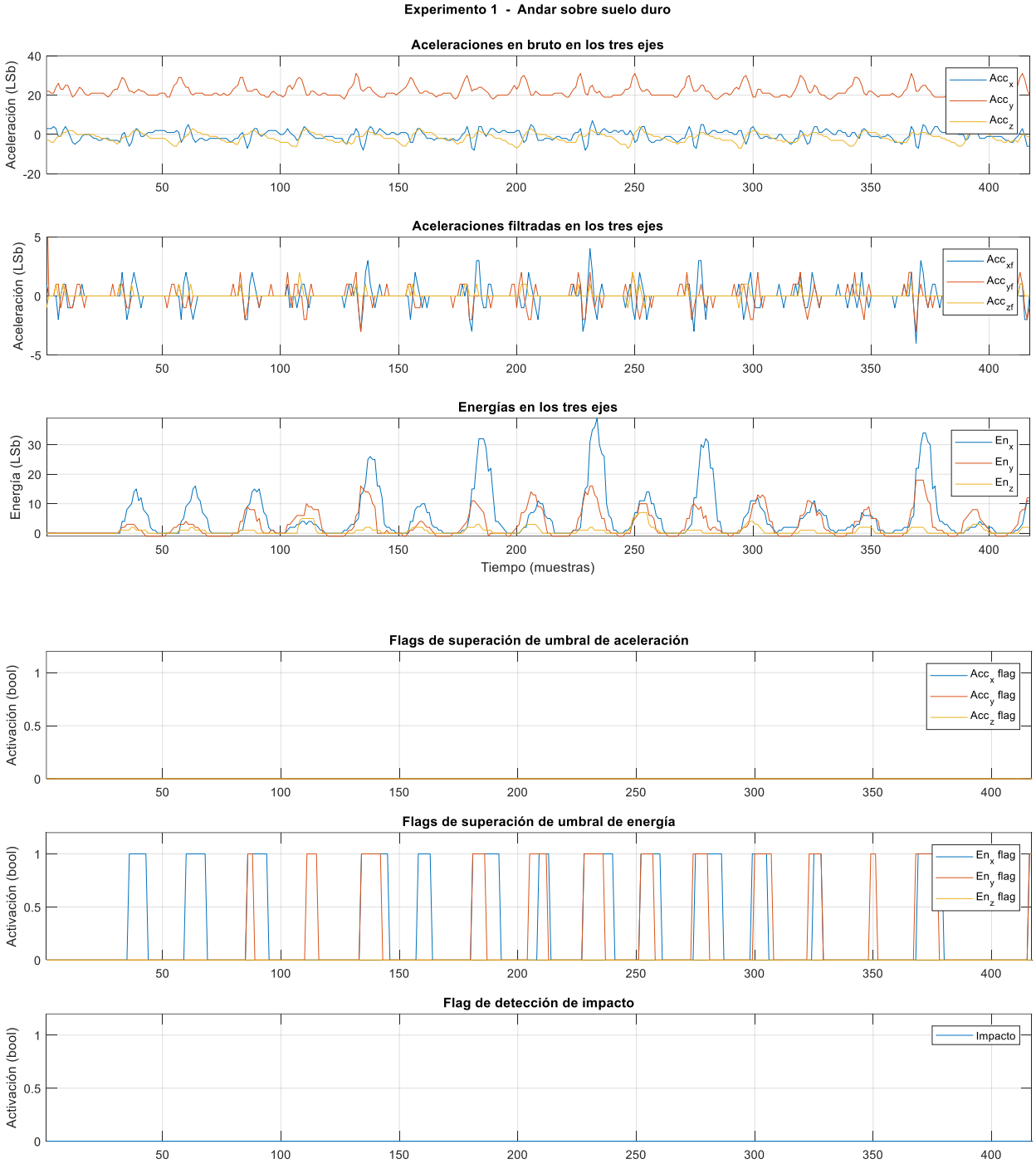


Figura 8-4. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 1.

8.2 Experimento 2 – Subir escaleras

En este caso no se debe detectar impacto, y tampoco se detectarán superaciones de los umbrales de aceleración o energía. Se producen variaciones muy leves de las aceleraciones como para resultar en aceleraciones filtradas significativas y energías significativas.

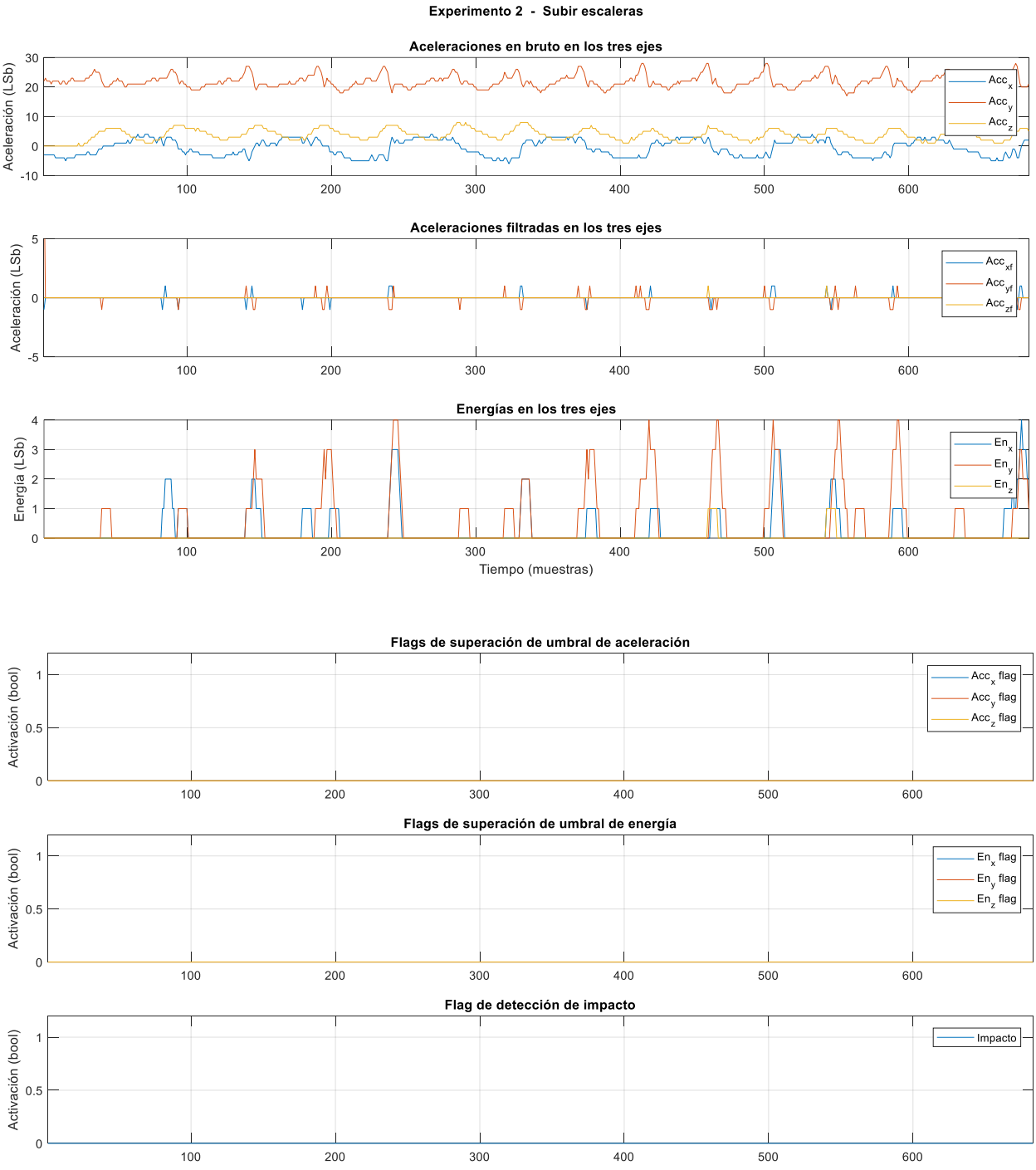


Figura 8-5. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 2.

8.3 Experimento 3 – Bajar escaleras

En este caso no se deberá detectar impacto, aunque sí superación del umbral de energía en algunas ocasiones. Se producen variaciones despreciables de aceleración en el eje Z, pero significativas en los ejes Y (por la caída del peso del cuerpo al apoyar el pie en el escalón inferior) y X. Esto deriva en aceleraciones filtradas y energías significativas en algunas ocasiones.

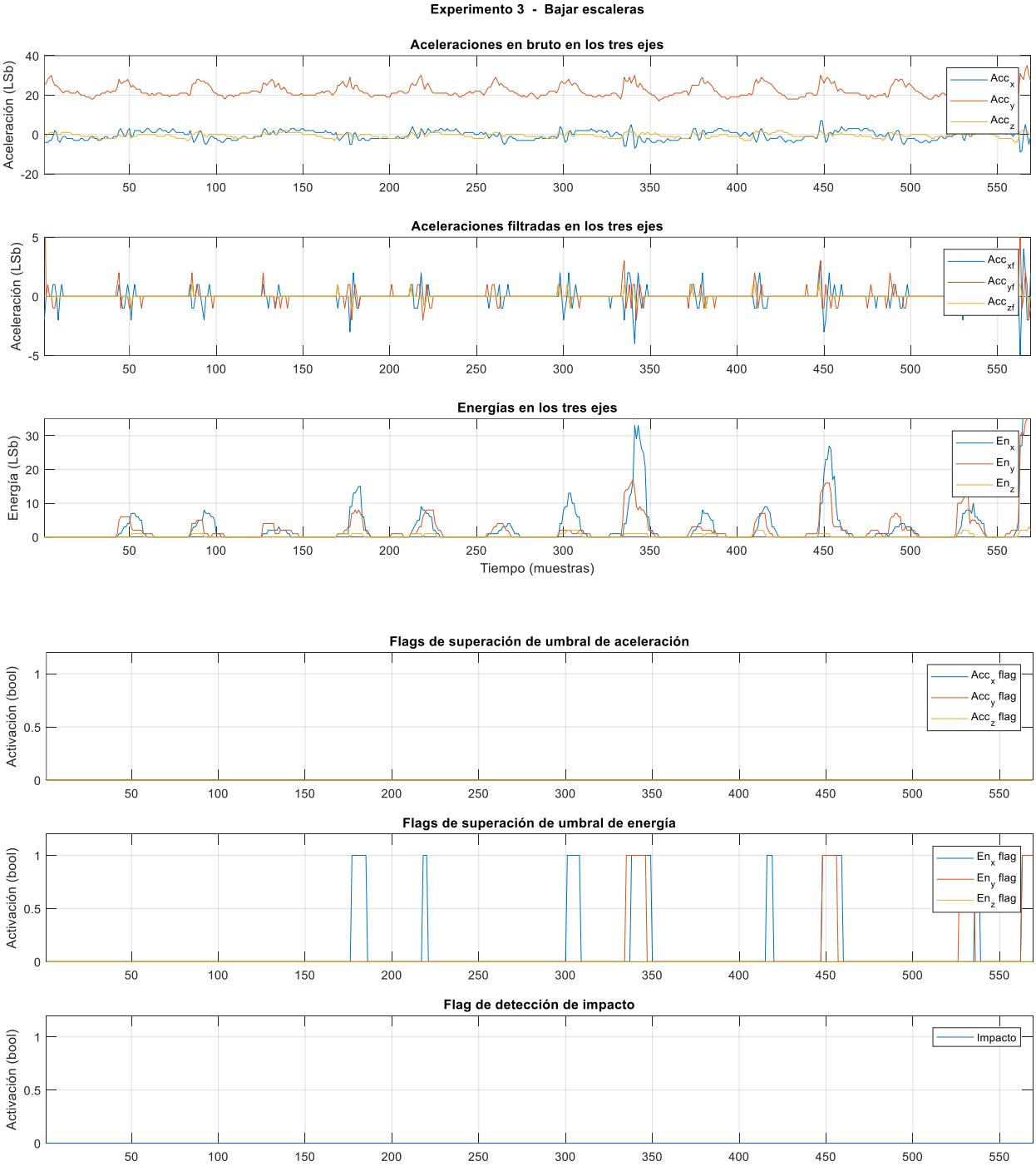


Figura 8-6. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 3.

8.4 Experimento 4 – Agacharse y levantarse doblando las rodillas para coger un objeto

En este caso no se deberá detectar impacto ni superación de umbrales de aceleración o energía. En prácticamente todo el experimento no se producen variaciones de aceleración significativas como para generar aceleraciones filtradas o energía calculada apreciable o susceptible de ser tenida en cuenta para la umbralización.

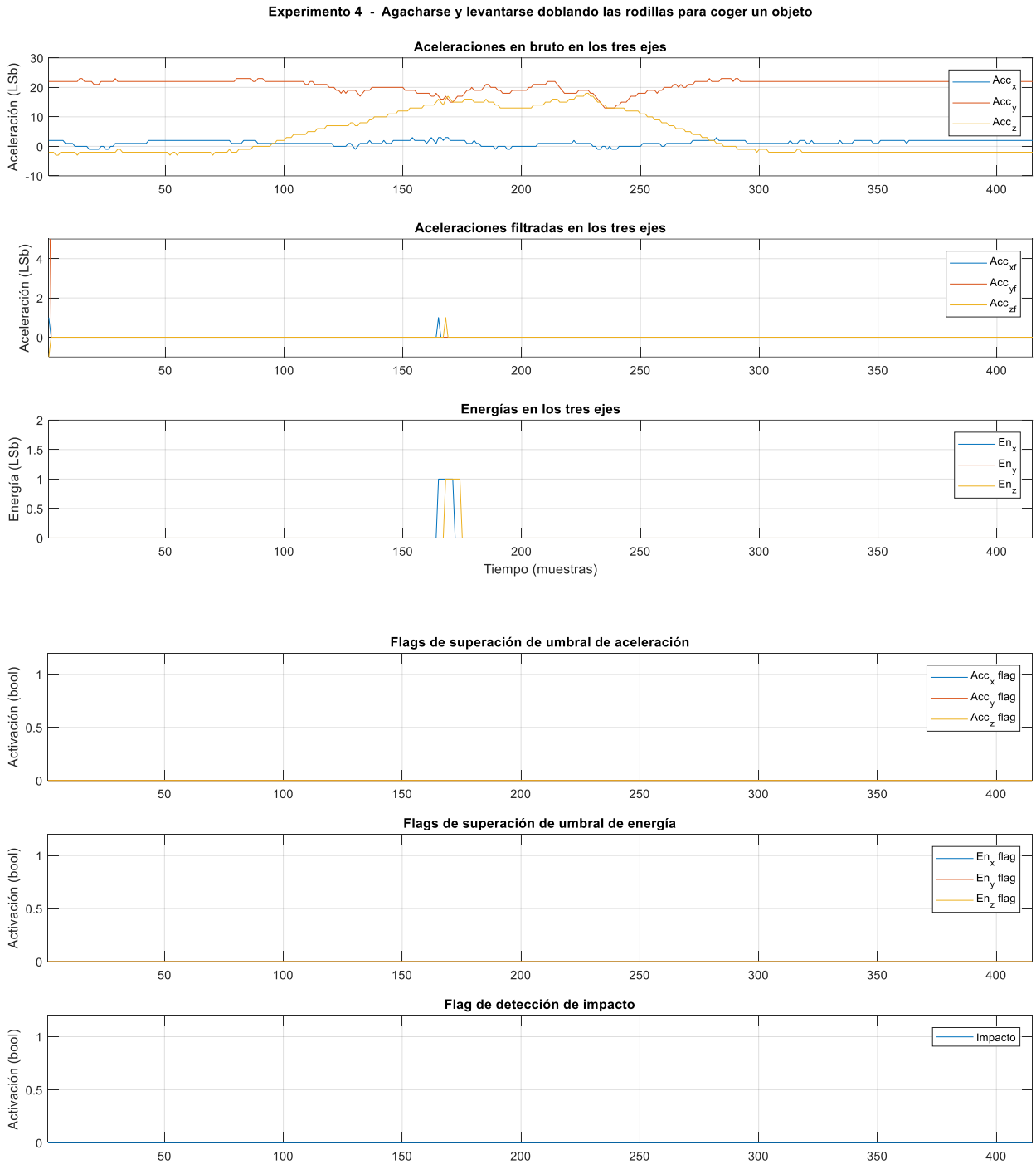


Figura 8-7. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 4.

8.5 Experimento 5 – Sentarse en una silla despacio

En este caso no se deberá detectar impacto ni superación de umbrales de aceleración o energía. En ningún momento de todo el experimento se producen variaciones de aceleración significativas como para generar aceleraciones filtradas o energía calculada apreciable o susceptible de ser tenida en cuenta para la umbralización.

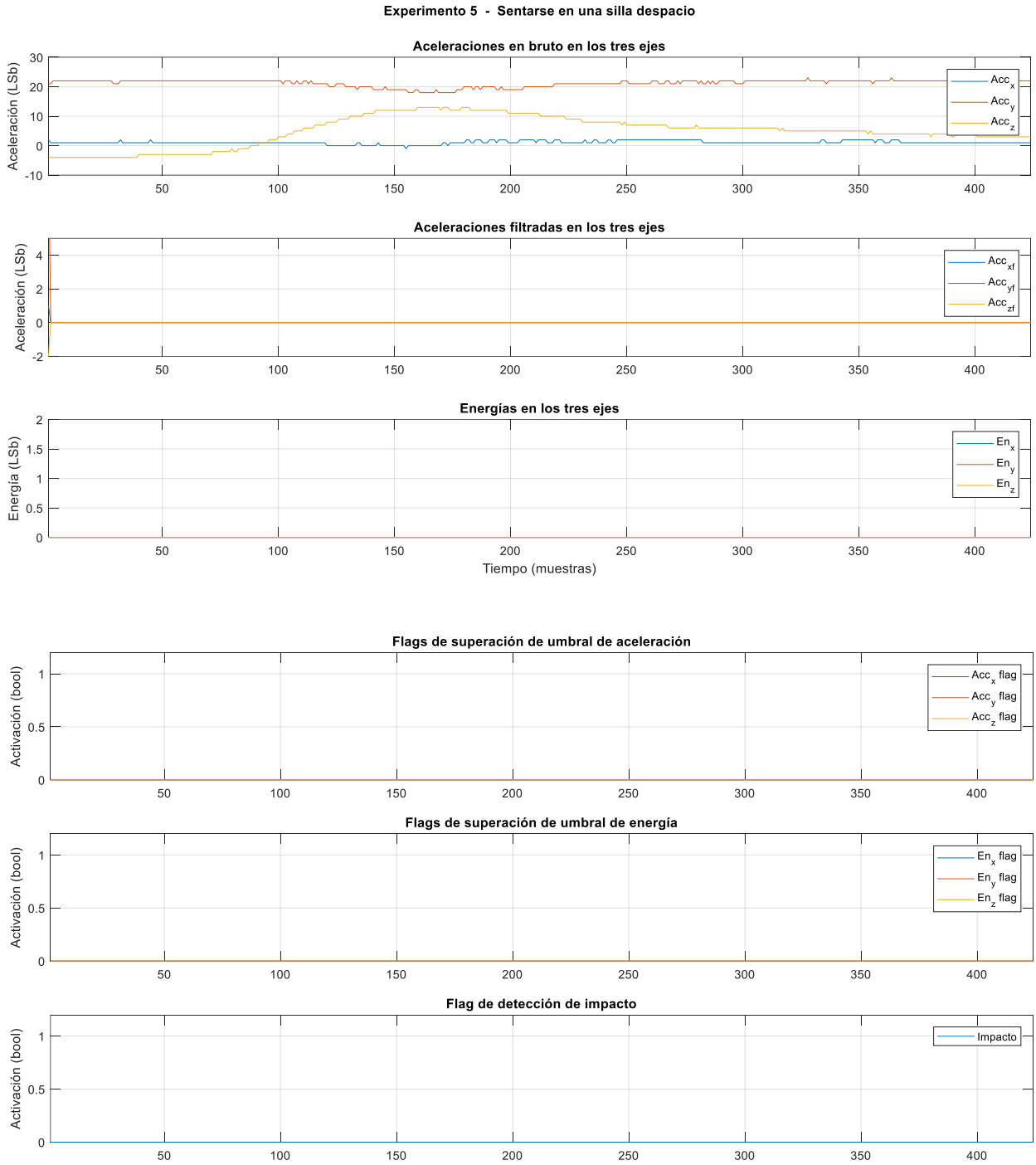


Figura 8-8. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 5.

8.6 Experimento 6 – Levantarse de una silla

En este caso no se deberá detectar impacto ni superación de umbrales de aceleración o energía. En ningún momento de todo el experimento se producen variaciones de aceleración significativas como para generar aceleraciones filtradas o energía calculada apreciable o susceptible de ser tenida en cuenta para la umbralización.

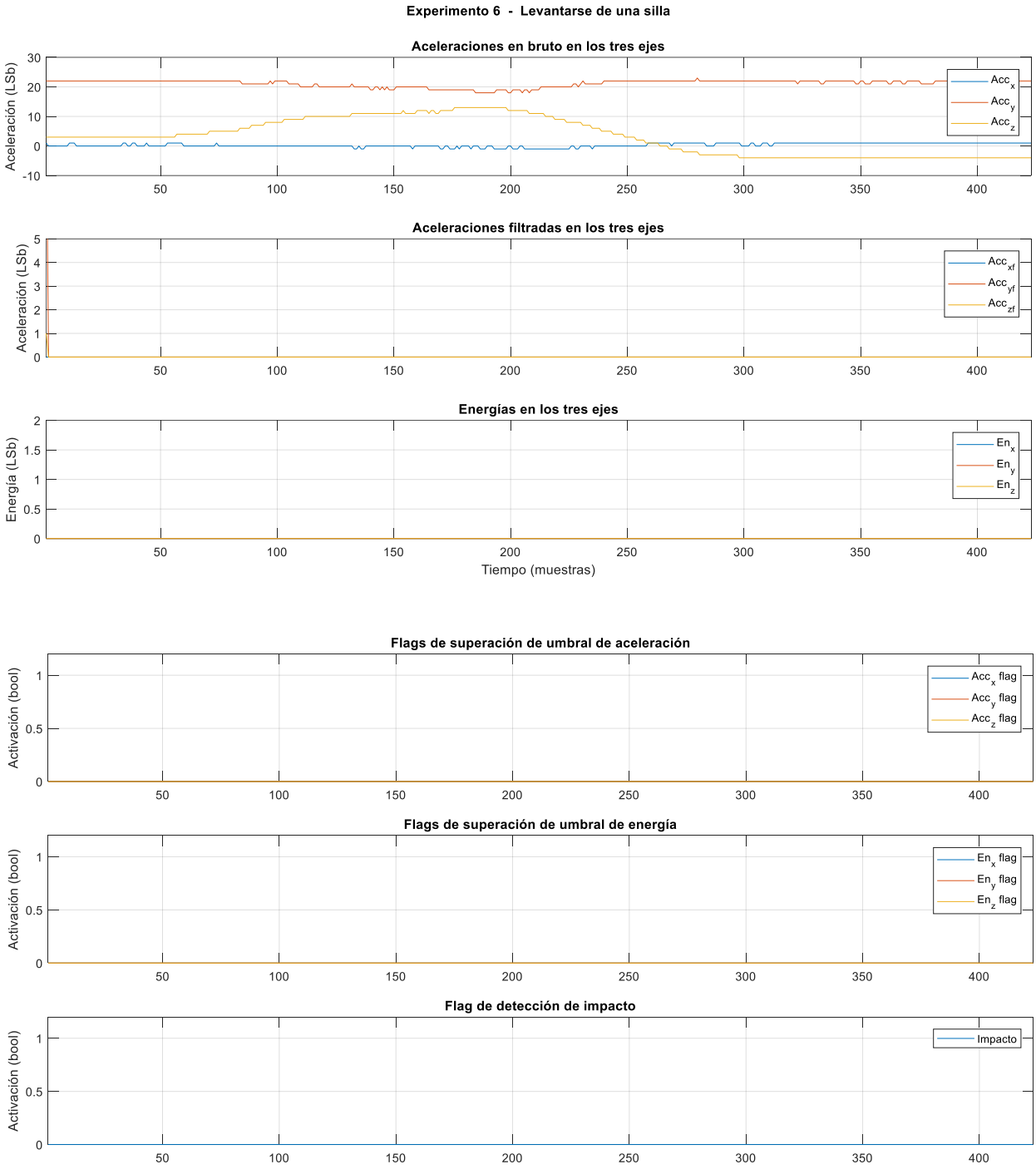


Figura 8-9. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 6.

8.7 Experimento 7 – Salto vertical sobre suelo blando

En este caso **sí** que se deberá detectar impacto y, por tanto, superación de umbrales de aceleración y energía. Se puede apreciar cómo se ha superado el umbral de aceleración en los 3 ejes debido a unos valores de aceleraciones filtradas elevados en el momento de la caída (contacto con el suelo) y un pico extraordinariamente grande en la energía en el eje Y debido a las variaciones de estas aceleraciones filtradas.

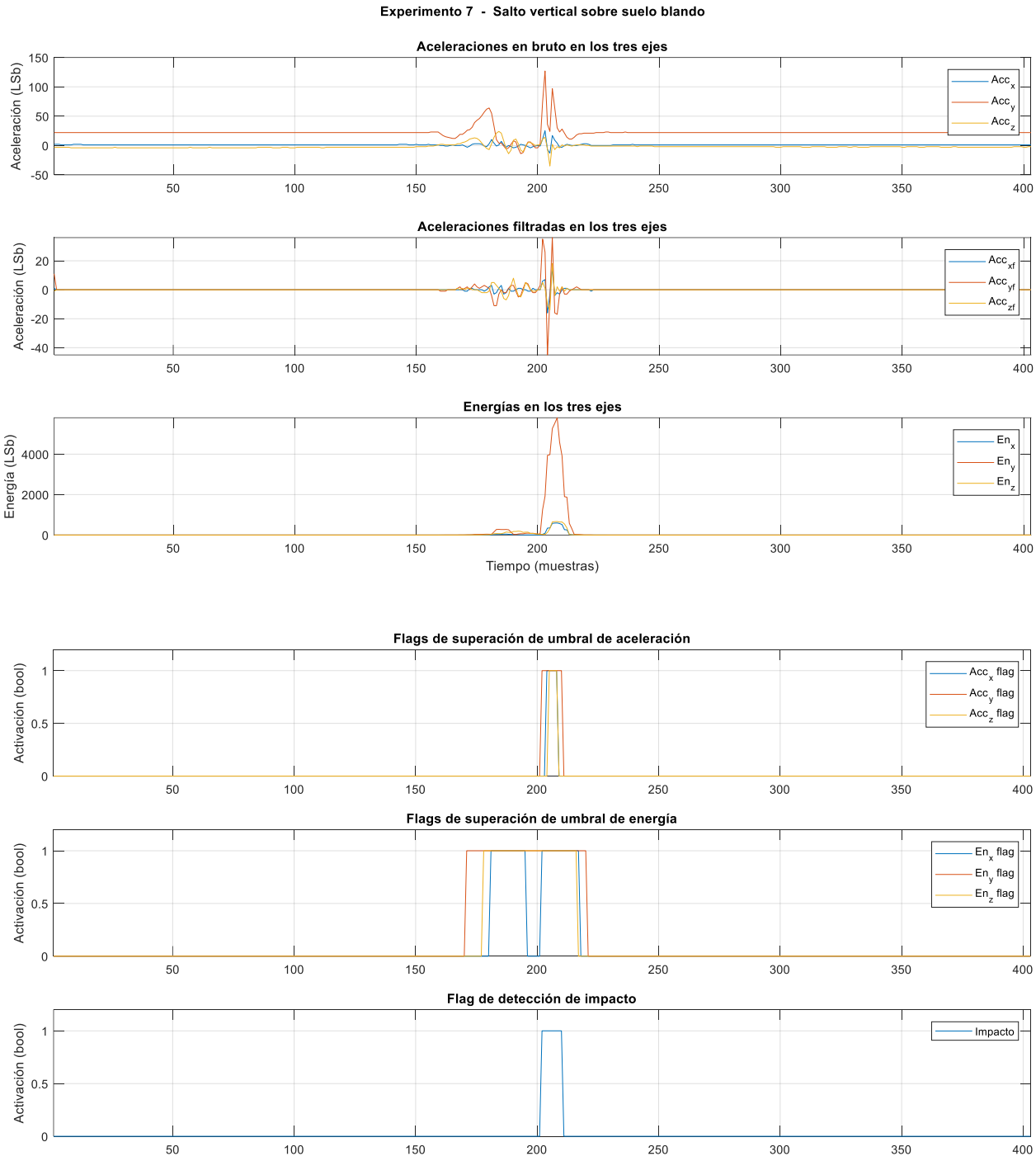


Figura 8-10. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 7.

8.8 Experimento 8 – Caída de rodillas sobre suelo blando

En este caso **sí** que se deberá detectar impacto y, por tanto, superación de umbrales de aceleración y energía. Se puede apreciar cómo se ha superado el umbral de aceleración en 2 ejes debido a unos valores de aceleraciones filtradas elevados en el momento de la caída (contacto con el suelo) y un pico extraordinariamente grande en la energía en el eje Y debido a las variaciones de estas aceleraciones filtradas.

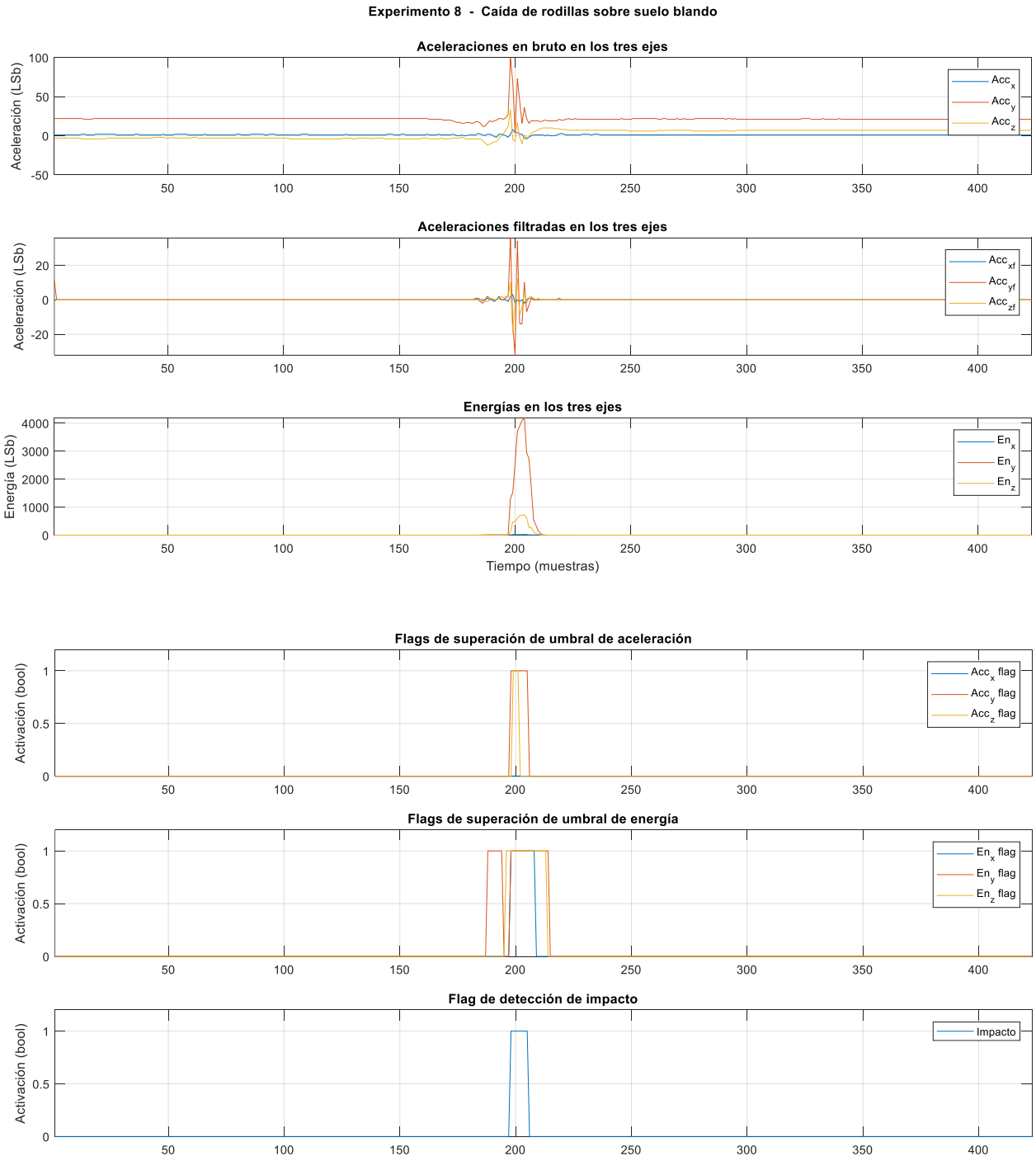


Figura 8-11. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 8.

8.9 Experimento 9 – Caída de rodillas sobre suelo blando y tirarse al suelo

En este caso se deberá detectar impacto en la caída de rodillas primero y en el contacto contra el suelo del resto del cuerpo después. Por tanto, se superarán los umbrales en dos ocasiones separadas. En la primera caída se puede apreciar cómo se ha superado el umbral de aceleración en 2 ejes, debido a unas aceleraciones filtradas elevadas en el momento de la caída, y el umbral de energía en 3 ejes, con un pico extraordinariamente grande en el eje Y. En la segunda, se supera el umbral de aceleración en el eje Y (siendo sólo un poco mayor que en el Z) y el umbral de energía en los 3 ejes, siendo mayor en el eje Y.

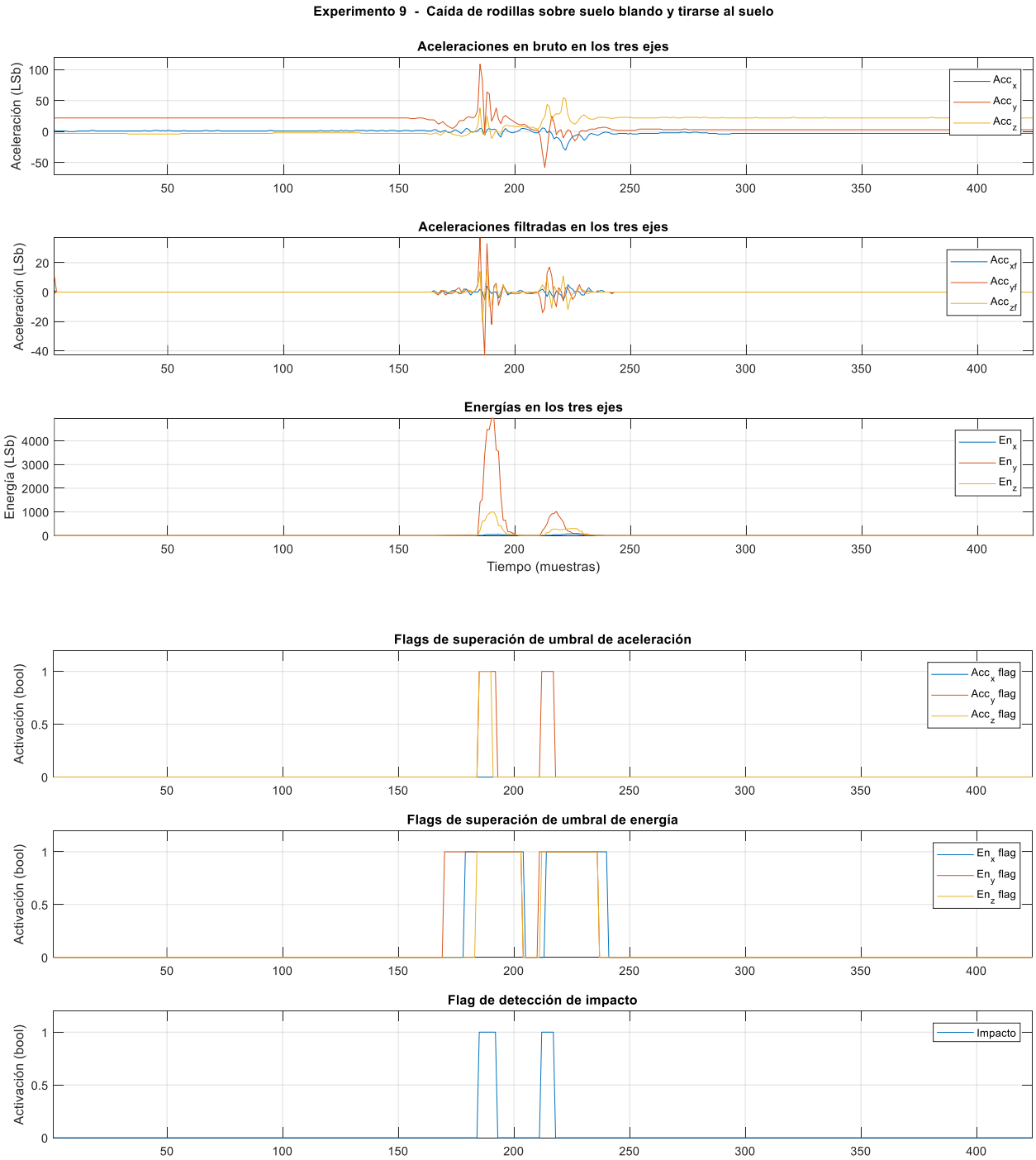


Figura 8-12. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 9.

8.10 Experimento 10 – Caída horizontal desde banco bajo sobre suelo blando

En este caso se deberá detectar impacto y, por tanto, superación de umbrales de aceleración y energía. Se puede apreciar cómo se ha superado el umbral de aceleración en los 3 ejes, debido a unos valores de aceleraciones filtradas elevados en el momento de la caída (contacto con el suelo), y un pico bastante grande en la energía en los tres ejes, debido a las variaciones de estas aceleraciones filtradas.

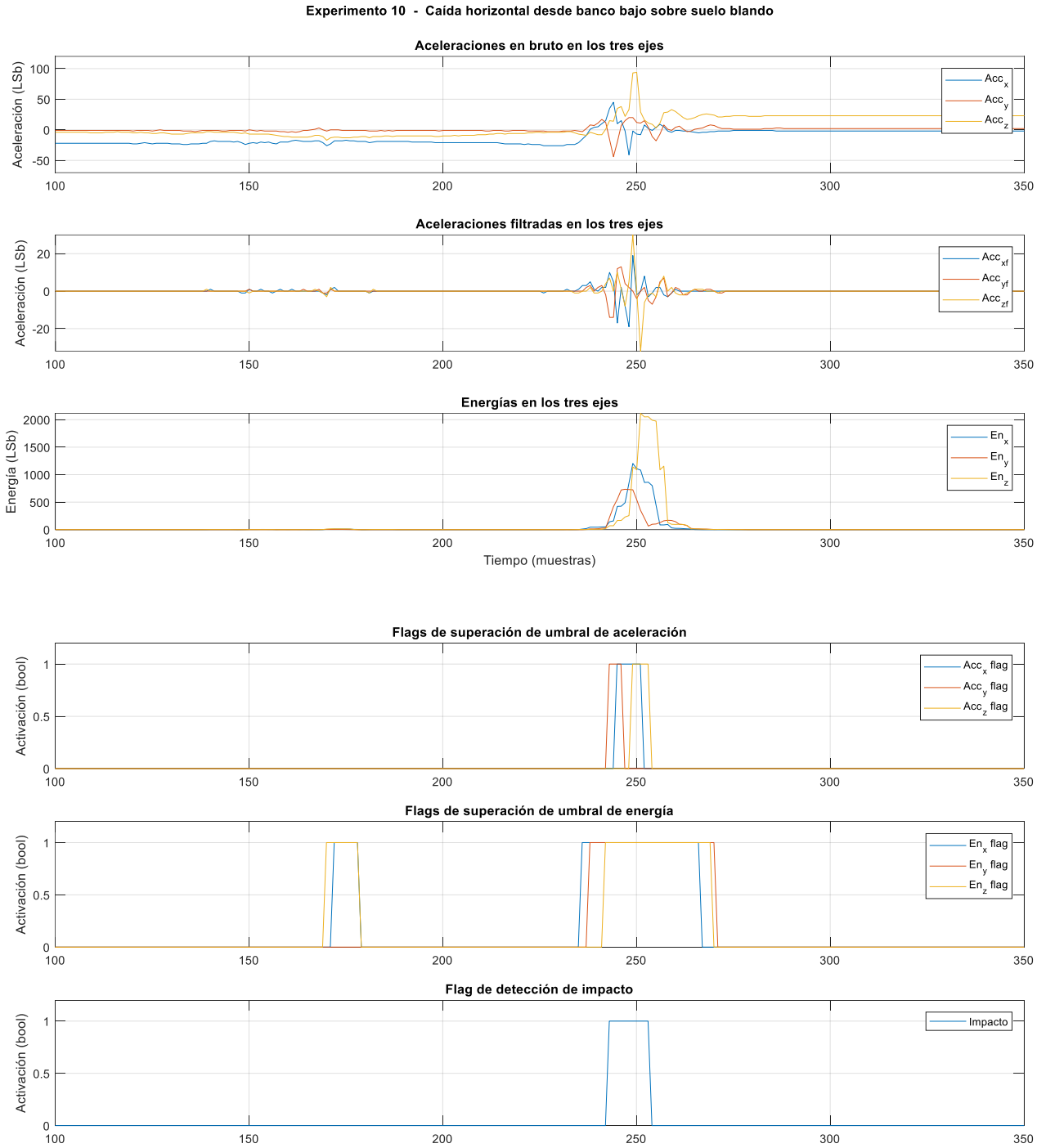


Figura 8-13. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 10.

8.11 Experimento 11 – Caída desde banco sentado a rodilla y apoya las manos (cuadrupedia) suelo blando

En este caso se deberá detectar impacto y, por tanto, superación de umbrales de aceleración y energía. Se puede apreciar cómo se ha superado el umbral de aceleración en los 3 ejes, debido a unos valores de aceleraciones filtradas elevados en el momento de la caída de rodillas, sin ser suficientemente significativos al apoyar las manos (a diferencia del Experimento 9). También se supera el umbral de energía en los 3 ejes, con un pico importante en el eje Y.

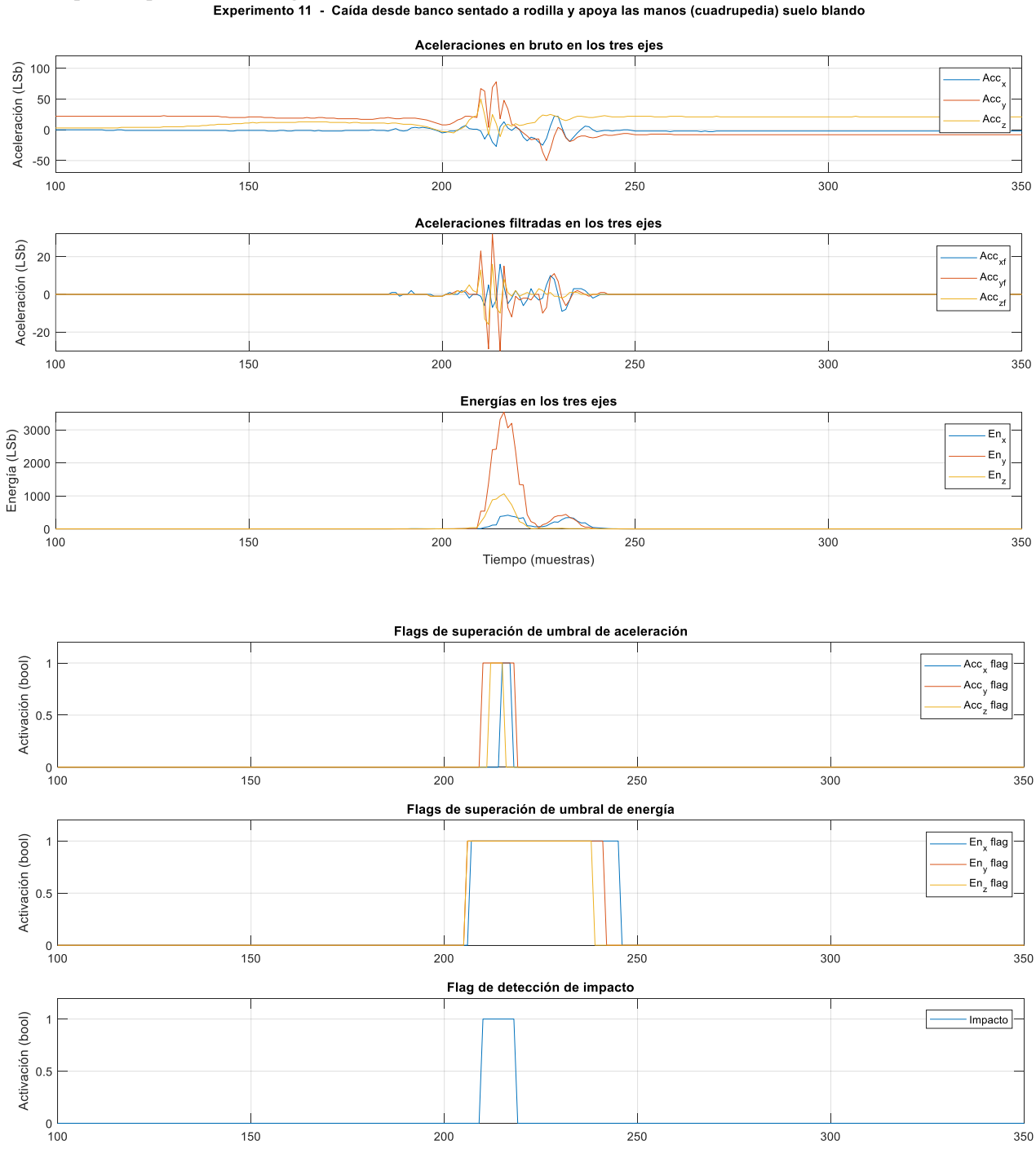


Figura 8-14. (Por orden descendente) Aceleraciones en bruto, filtradas, energía, flags de superación de umbral de aceleración, flags de superación de umbral de energía y flag de detección de impacto para el Experimento 11.

9 PROBLEMAS ENCONTRADOS

Durante el desarrollo del proyecto han ido surgiendo algunos problemas, tanto a nivel de desarrollo como a nivel de usabilidad del hardware o capacidad de testado, que se comentarán a continuación, a la vez que se explicarán las soluciones adoptadas.

Problemas encontrados durante el desarrollo del firmware:

- Para que se pueda *linkar* y compilar correctamente la librería PIC18F Peripheral Library no se pueden usar versiones del compilador XC8 recientes [45].

  Se probó inicialmente con la versión del compilador v2.32 y no fue posible compilar correctamente la librería. Tras probar la versión v1.45 se compilaban casi todas las funciones, pero quedaban algunas incompatibilidades que hacían imposible el uso de la librería. Por último, se instaló la versión v1.34 que resultó ser compatible con todas las funciones que se pretendían usar, aunque mantenía limitaciones propias de errores del compilador que se han subsanado a través del código de alguna manera (por ejemplo, la limitación de la función *__delay_ms*).

- Warning 752 – "Conversion to shorter data type".

  Esta advertenica se imprime cuando se asigna a una variable con tipo de menor tamaño un valor con tipo de mayor tamaño [46]. La idea inicial era añadir directivas *#pragma warning disable 752* en los lugares en los que se quiere eliminar esa advertencia en el código, pero se constata que las directivas *#pragma warning push* y *#pragma warning pop* no funcionan como se espera, haciendo que se aplique la directiva *#pragma warning disable 752* a todo el código posterior a la directiva, con lo que finalmente se comprueban uno a uno todos los casos de la advertencia 752 y se añade al inicio del código la directiva *#pragma warning disable 752* para evitar que llene la consola de salida y dificulte la detección de posibles problemas reales en el código [47].

- Warning 520 – "function is never called".

  Esta advertencia se imprime cuando hay funciones definidas que no se ejecutan nunca. Se ha aplicado la misma fórmula que con la advertencia anteriormente comentada, se añade la directiva *#pragma warning disable 520* al inicio del código.

- Insuficiencia en la memoria RAM.

  Inicialmente se consideró la opción de convertir las aceleraciones capturadas, realizando el desplazamiento de 4 bits a la izquierda antes de insertarlas en los buffers circulares de aceleraciones, pero esto requería de variables de 16 bits y no se podían organizar correctamente en RAM de forma que cupiesen los tres buffers de 85 posiciones, por tanto, se definieron de 8 bits de tamaño y se trabajaría siempre con el byte sin convertir. De ser necesarias las conversiones, se realizarían a posteriori una vez realizadas todas las operaciones.

Problemas encontrados mientras se realizaban pruebas / extracción de datos:

- El error de tensión al programar el microcontrolador, debido a que el Pickit no da a la salida la tensión de programación configurada. Sin una solución directa, siempre se ha comprobado el conexionado del hardware y se ha repetido el intento de programación, haciéndose desesperante a veces.

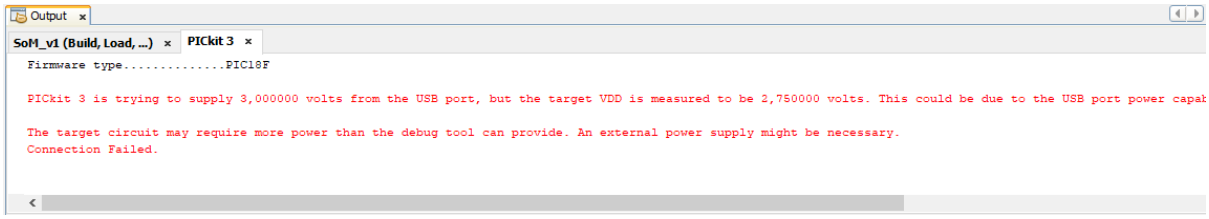


Figura 9-1. Error de tensiones en el programador Pickit 3.

- El error de configuración del Pickit desde el IDE, que obliga a cerrar y reabrir el IDE.

Figura 9-2. Error de conexión entre el IDE y el programador (necesario reiniciar el IDE).

- Bloqueo de la lectura de aceleraciones llegado a un punto de ejecución:

  Se descubrió que al intentar imprimir por puerto serie los datos generados a partir de la ejecución del algoritmo, cuando se llegaba a cierto punto la impresión no mostraba datos nuevos, se quedaba bloqueado imprimiendo el último dato obtenido correctamente de manera infinita:

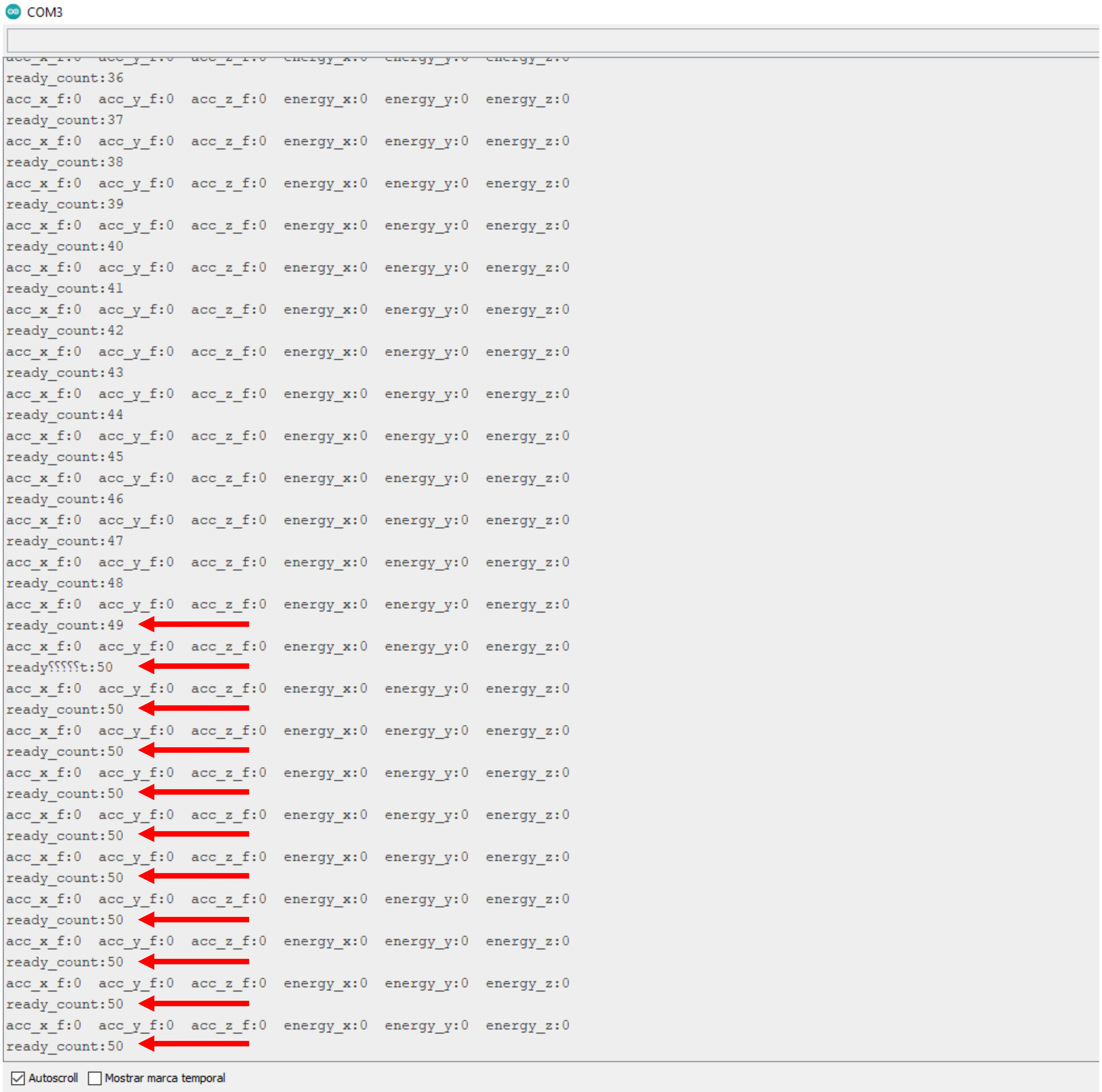


Figura 9-3. Bloqueo en la adquisición de datos de aceleración e impresión de los últimos de manera permanente.

Téngase en cuenta que en ese momento del desarrollo del proyecto se imprimía por puerto serie en cada ciclo del bucle principal, independientemente de que se obtuvieran datos nuevos o no.

Se hicieron cambios en los datos impresos y en su formato y se descubrió que conforme se simplificaban las impresiones aumentaba el tiempo de ejecución del programa sin que se ocasionara el bloqueo comentado.

Con eso en mente, debía haber un problema de tiempos y parecía que cuando se recibía la señal de data_ready por parte del acelerómetro el programa se encontraba bloqueado por la espera de la impresión por puerto serie, con lo que se mantenia a false la variable acc_data_ready haciendo que no se leyera más del registro del acelerómetro para permitir recibir la siguiente interrupción y se quedaba ahí bloqueado.

Todo esto fue analizado con un analizador lógico y se pudo visualizar el problema.

Se comprobó el tiempo que se tarda en realizar la lectura de las aceleraciones por SPI (ejes X, Y y Z). Viendo que es insignificante en comparación al periodo teórico de la señal data_ready de 25 ms.

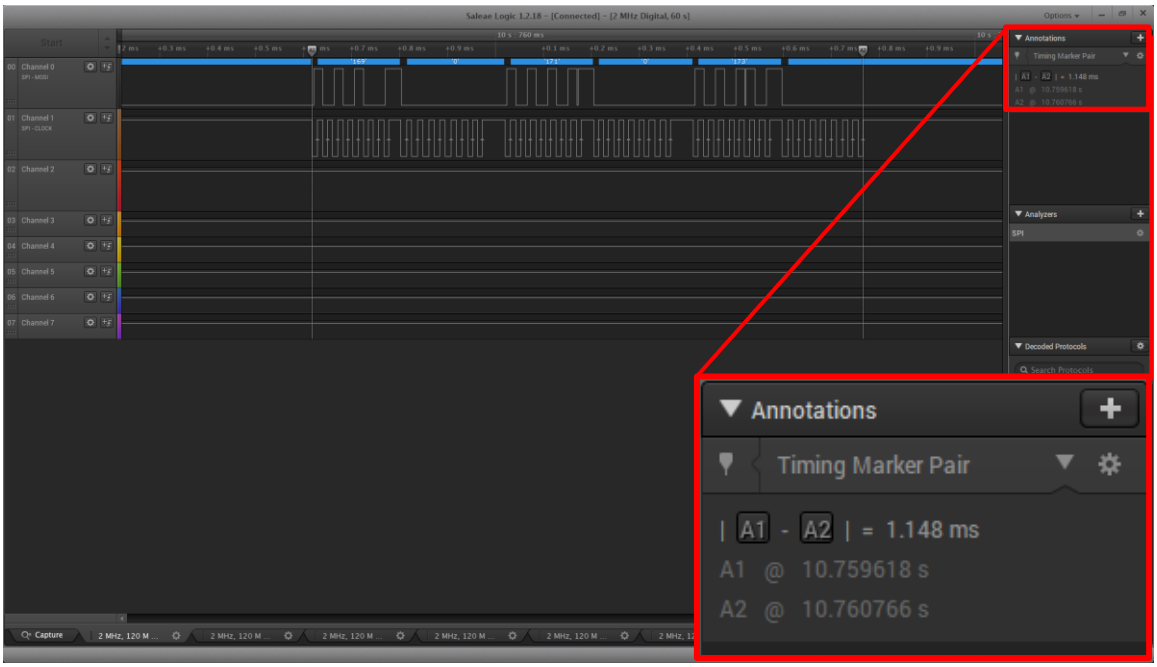


Figura 9-4. Señales de SPI MOSI y SCLK durante la lectura SPI y tiempo transcurrido.

A continuación, se midió cada cuánto tiempo se hacían esas lecturas y se puedo comprobar que el tiempo era superior al periodo de la señal data_ready, es decir, no se estaba haciendo una lectura por cada interrupción sino menos. Algo estaba ralentizando el tiempo entre lecturas.

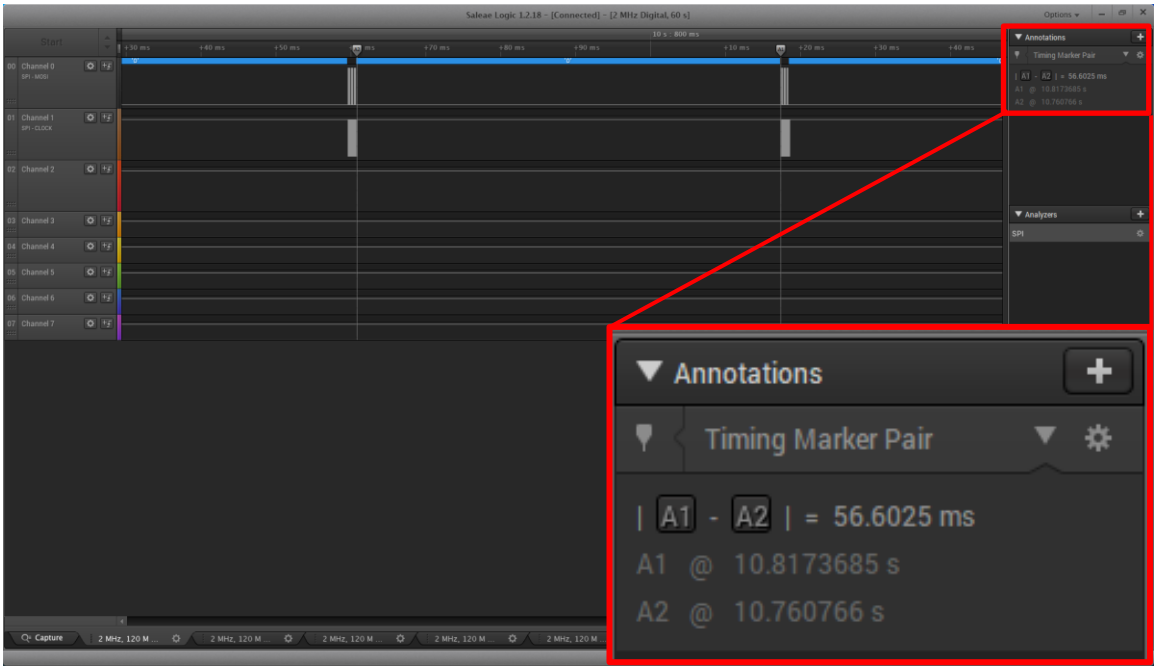


Figura 9-5. Tiempo transcurrido entre lecturas SPI.

Esto estaba haciendo que la señal data_ready del acelerómetro no fuera una señal periódica, al no realizarse las lecturas no cambiaba a nivel bajo a tiempo y a veces se unían lo que habrían sido dos interrupciones en un solo pulso.

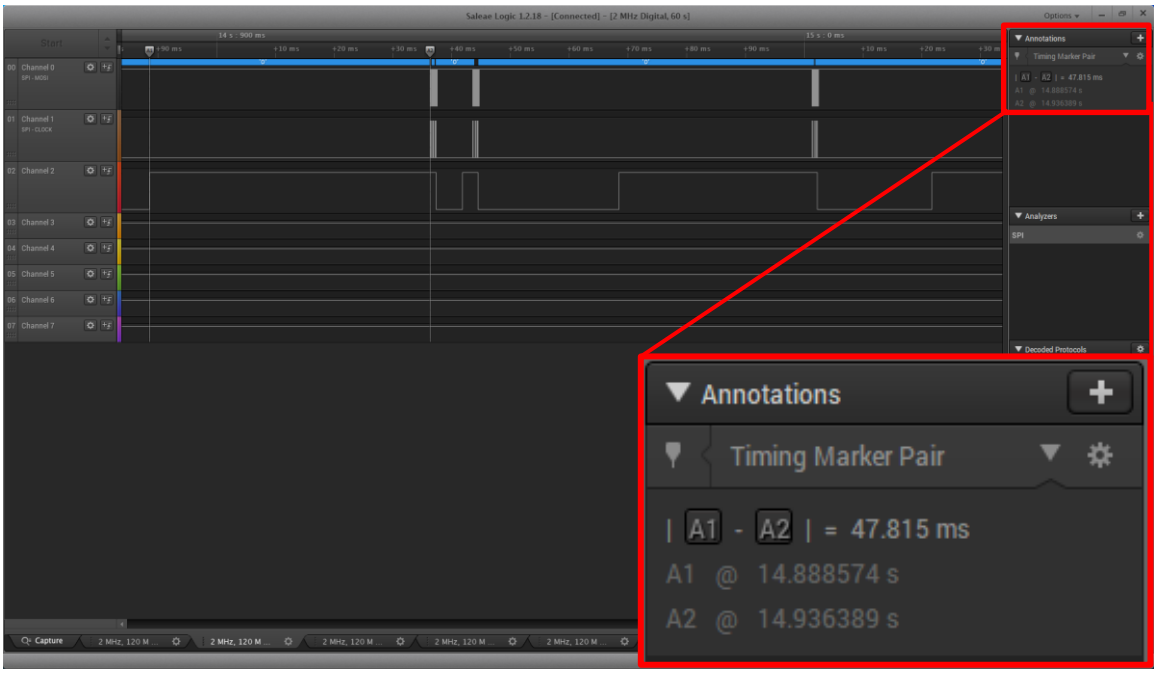


Figura 9-6. Señales SPI MOSI y SCLK, y señal data_ready del acelerómetro.

A la larga, esto estaba causando que, llegado el momento, la señal de interrupción coincidiera con la lectura por SPI haciendo que la variable acc_data_ready, que debe cambiar a true en la rutina de interrupción y a false justo antes de la lectura, se quedara a false permanentemente, impidiendo la lectura mientras la señal data_ready se encontraba en alto y a la espera de esa lectura para cambiar de estado.

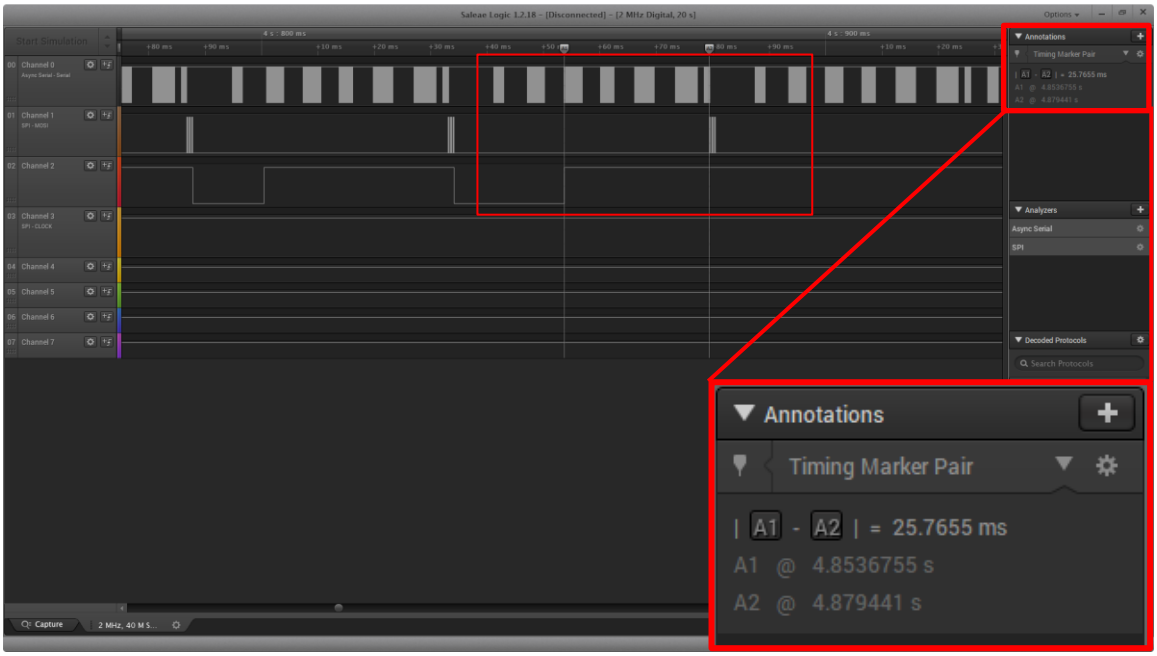


Figura 9-7. Señal USART, SPI MOSI y data_ready, indicando el momento donde la señal data_ready se mantiene en alto de forma permanente debido a la coincidencia entre la interrupción de dicha señal y la lectura SPI.

Tras las comprobaciones realizadas, se aplicaron los siguientes cambios:

- Simplificación de las cadenas enviadas por puerto serie y agrupación de los datos.
- Impresión por puerto serie únicamente ante una lectura de aceleraciones por SPI, con lo cual las

impresiones quedan sincronizadas con el resto de las señales.

Y tras aplicar los cambios comentados, la señal data_ready se mantenía periódica y no se quedaba bloqueado el sistema (con la limitación de la impresión serie).

Se comprobó la frecuencia de lectura de aceleraciones del acelerómetro sin impresiones por puerto serie más allá del contador de interrupciones (salían unos 37.5 Hz de unos 40 teóricos).

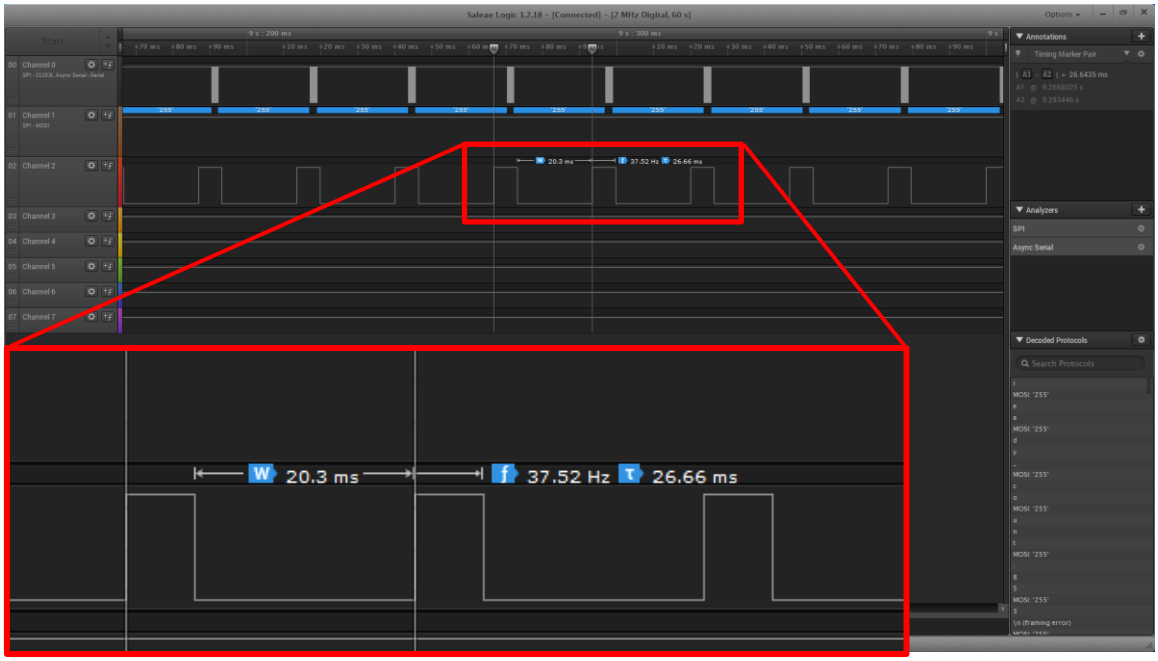


Figura 9-8. Señal USART y señal data_ready del acelerómetro y frecuencia/periodo de la señal data_ready.

Se comprobó cómo iba aumentando el valor de la cuenta de interrupciones de la señal data_ready y su correcta impresión por puerto serie.

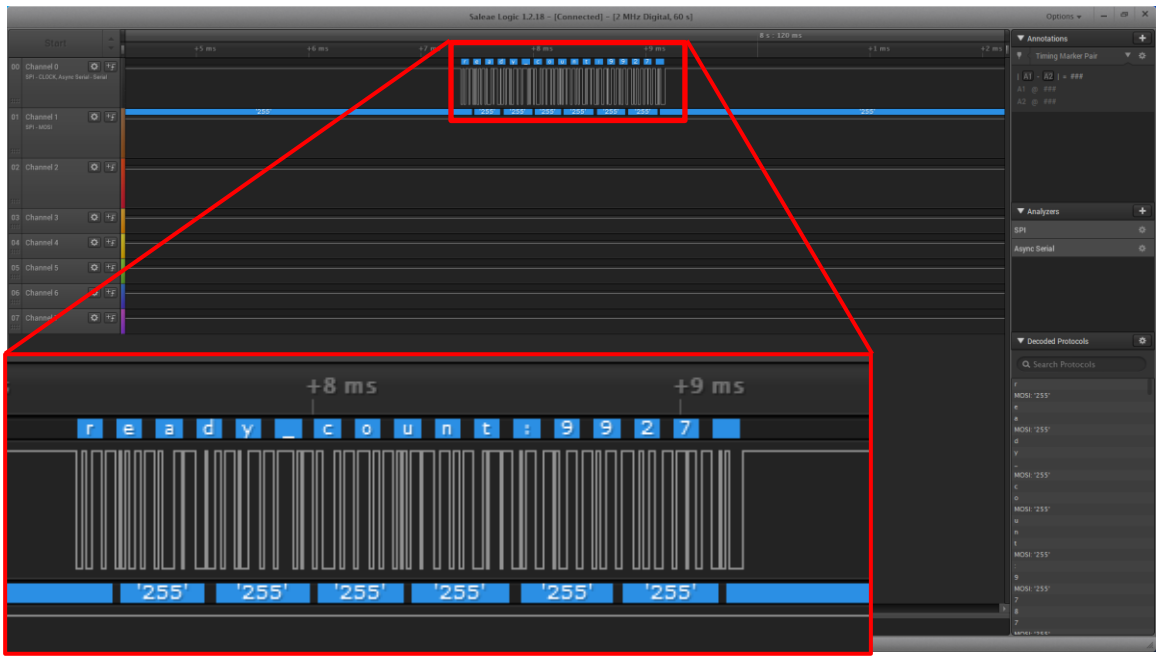


Figura 9-9. Señal USART mostrando el valor de la cuenta de interrupciones de la señal data_ready.

Con lo que el problema quedaba solucionado siempre que se tuviera en cuenta la limitación en la

impresión serie.

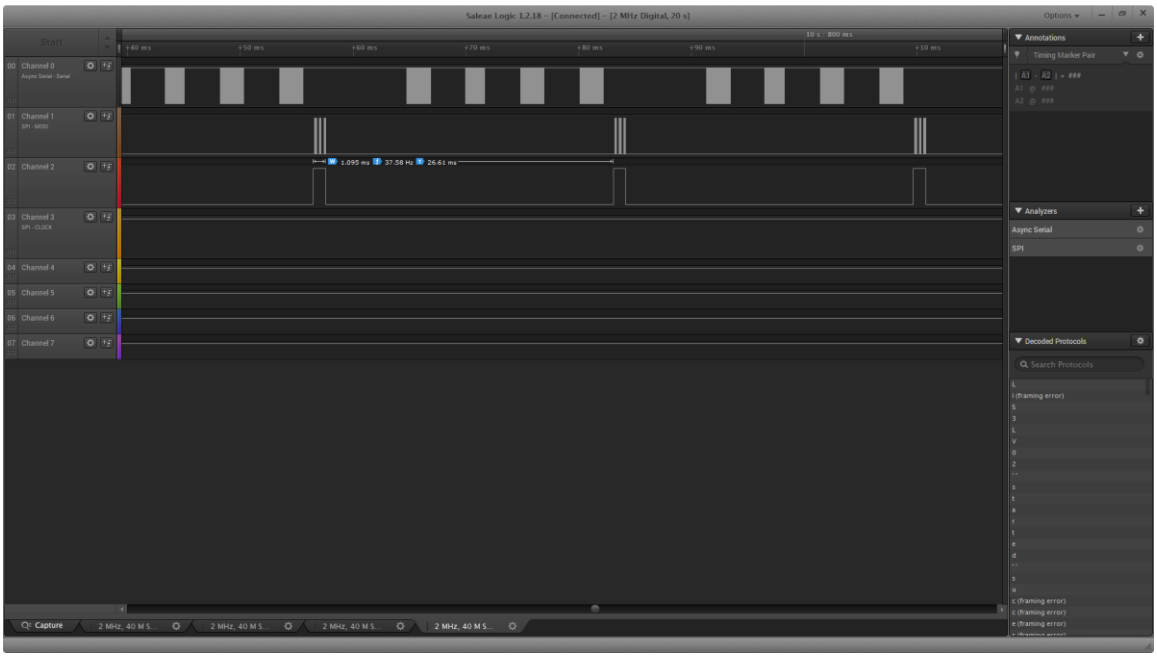


Figura 9-10. Señales USART, SPI MOSI y data_ready mostrando el funcionamiento correcto y la limitación de la impresión por puerto serie, en ese momento, a impresión de aceleraciones filtradas, energías y flags de umbralización de aceleraciones y energías, en los tres ejes.

- Los ruidos producidos en la comunicación serie al unir los jumpers J1 y J2 que obligaron a desunir los jumpers y soldar dos cables a la salida del microcontrolador.

10 CONCLUSIONES Y FUTUROS DESARROLLOS DEL PROYECTO

Tras el desarrollo del proyecto y teniendo en mente los objetivos principales y el alcance inicial de este, podemos derivar las siguientes conclusiones.

Se ha implementado el algoritmo de detección de impacto en el microcontrolador del SoM en lenguaje C, con éxito. Este algoritmo se ha validado con experimentos consistentes en diferentes escenarios con movimientos que contienen impactos, con lo que se demuestra que es totalmente funcional para su uso dentro del esquema general del sistema de aviso ante caídas.

Se han observado ciertas limitaciones en el microcontrolador, como el espacio de memoria RAM y su organización, que actualmente se encuentra cerca del 60 % de uso y que habría que tener en cuenta para futuras mejoras.

Pero en resumen, queda constancia de que es posible implementar un mecanismo de detección de caídas, de forma que no repercuta en la comodidad de los usuarios (peso y tamaño pequeños), que no tenga un coste elevado por tanto no suponga un gasto inasumible por programas de acompañamiento de la tercera edad subvencionados por los gobiernos autonómicos/estatales y que permita una fácil y rápida adaptabilidad del funcionamiento o del hardware utilizado en este gracias a la flexibilidad que aporta el lenguaje C a día de hoy.

Durante el desarrollo del proyecto el alcance ha ido viéndose modificado por diferentes motivos.

Al inicio, y tal como se hace referencia en [1], se planteaba la comunicación con agentes externos que pudieran modificar los parámetros principales del algoritmo en función de la persona para ajustarlo lo máximo posible y asegurar una detección más exacta si es que esto fuera necesario.

Por ello, parámetros como los umbrales de aceleración, de energía y de tiempo, y la separación de muestras, tau, para el cálculo de la energía, se definieron como variables (en vez de como macros), para permitir la modificación en tiempo de ejecución mediante comandos remotos. Sin embargo, más tarde esta modificación remota quedó fuera del alcance.

Además de las variables creadas para la configuración de estos parámetros se configuró la comunicación mediante el módulo BLE con un dispositivo móvil, de forma que de manera inicial se pudieran enviar las lecturas y los resultados de la ejecución del algoritmo y posteriormente se pudieran implementar los comandos mencionados para ajustar los parámetros. Todo esto siguiendo los documentos [48] y [49].

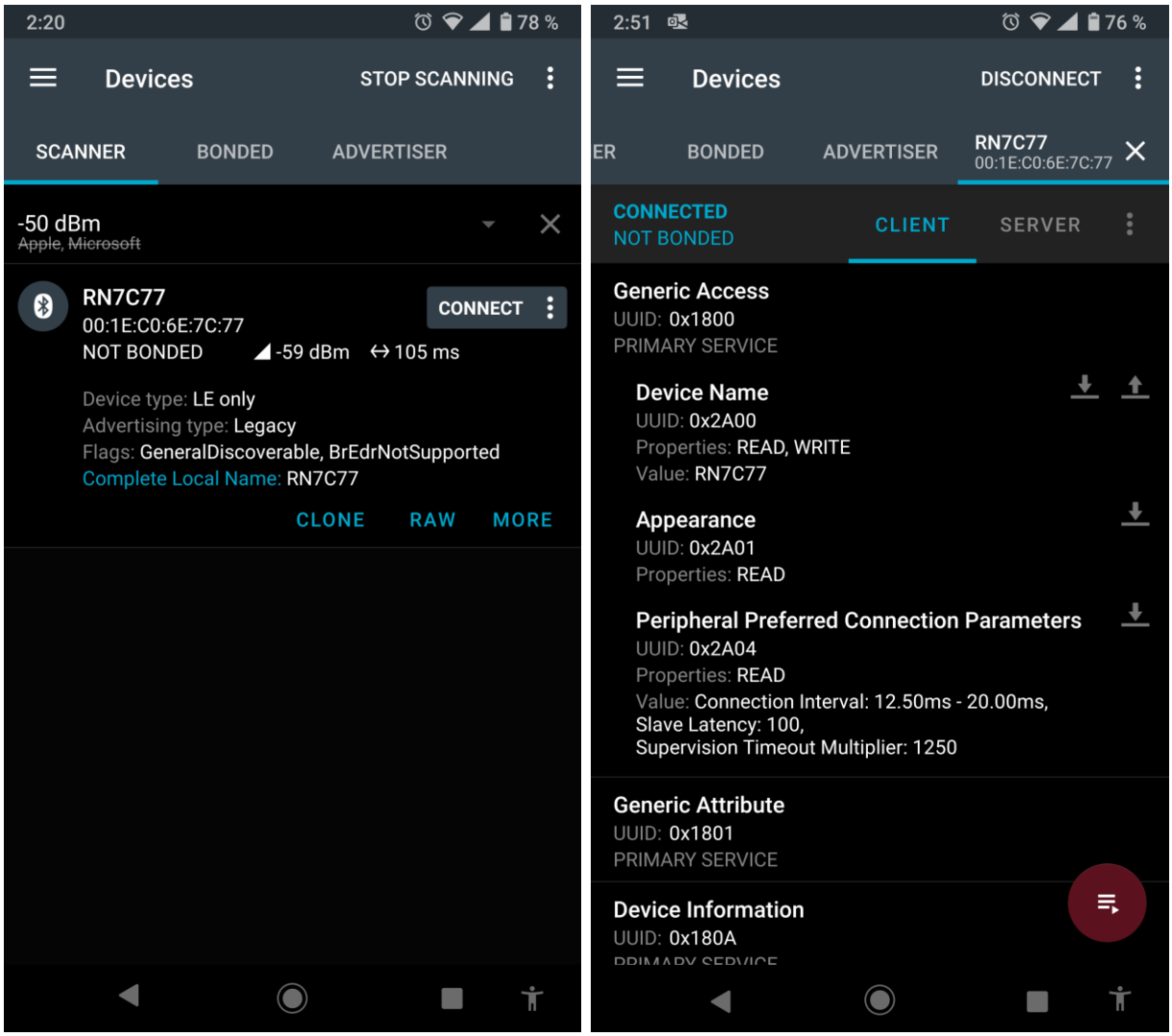


Figura 0-1. Módulo BLE RN4020 renombrado como RN7C77 con MAC 00:1E:C0:6E:7C:77 emitiendo con advertisings cada 100 ms (izquierda). Servicios habilitados en el módulo BLE RN4020 visualizados durante el estado de conexión (derecha).

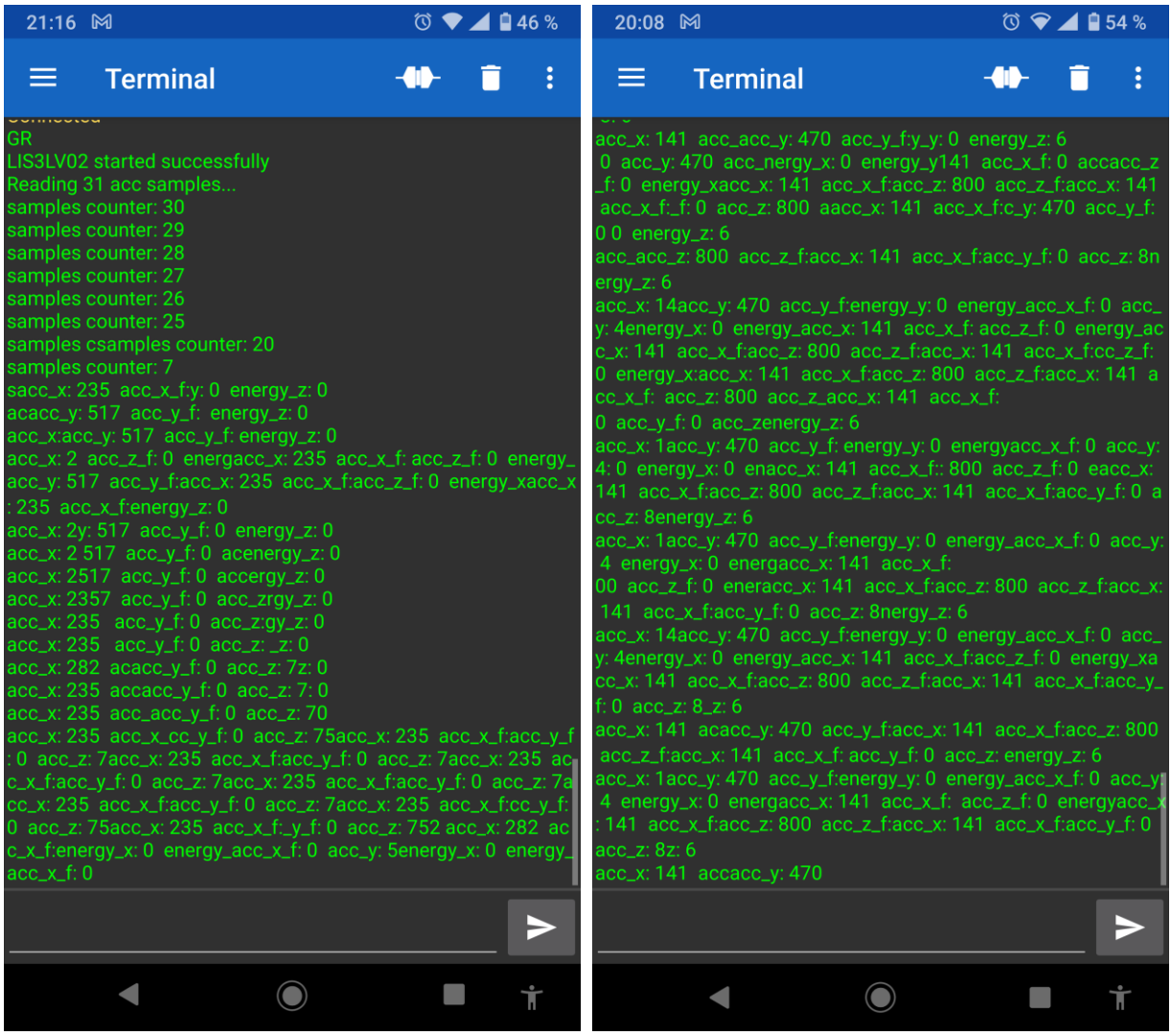


Figura 0-2. Terminal imprimiendo los datos recibidos por BLE en el dispositivo móvil, durante el inicio del programa y la lectura inicial de 31 muestras (izquierda) y durante la ejecución principal del algoritmo de detección de impacto (derecha).

REFERENCIAS

[1] D. Naranjo-Hernández, L. Roa, J. Reina-Tosina y M. Estudillo-Valderrama, «Personalization and Adaptation to the Medium and Context in a Fall Detection System,» *IEEE Transactions on Information Technology in Biomedicine,* vol. 16, nº 2, 2012.

[2] WHO, «Ageing and health,» 1 Octubre 2022. [En línea]. Available: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health. [Último acceso: 27 Abril 2023].

[3] WHO, «WHO global report on falls prevention in older age,» 2008.

[4] A. Rodríguez-Molinero, L. Narvaiza, C. Gálvez-Barrón, J. de la Cruz, J. Ruíz, N. Gonzalo, E. Valldosera y A. Yuste, «Caídas en la población anciana española: incidencia, consecuencias y factores de riesgo,» *Revista Española de Geriatría y Gerontología,* vol. 50, nº 6, pp. 274-280, 2015.

[5] X. Wang, J. Ellul y G. Azzopardi, «Elderly Fall Detection Systems: A Literature Survey,» *Frontiers in Robotics and AI,* vol. 7, 2020.

[6] L. Ren y Y. Peng, «Research of Fall Detection and Fall Prevention Technologies: A Systematic Review,» *IEEE Access,* vol. 7, pp. 77702-77722, 2019.

[7] G. Brown, «An accelerometer based fall detector: Development experimentation and analysis,» Berkeley, CA, USA, 2005.

[8] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach y G. Zhou, «Accurate fast fall detection using gyroscopes and accelerometer-derived posture information,» de *International Workshop on Wearable and Implantable Body Sensor Networks*, Berkeley, CA, USA, 2009.

[9] Q. Zhang, L. Ren y W. Shi, «HONEY: A Multimodality Fall Detection and Telecare System,» *Telemedicine and e-Health,* vol. 19, nº 5, pp. 415-429, 2013.

[10] T. de Quadros, A. E. Lazzaretti y F. K. Schneider, «A Movement Decomposition and Machine Learning-Based Fall Detection System Using Wrist Wearable Device,» *IEEE Sensors Journal,* vol. 18, nº 12, pp. 5082-5089, 2018.

[11] J. A. Santoyo Ramón y E. Casilari Pérez, «UMAFall: Fall Detection Dataset,» Universidad de Málaga, [En línea]. Available: https://figshare.com/articles/dataset/UMA_ADL_FALL_Dataset_zip/4214283. [Último acceso: 3 junio 2023].

[12] J. A. Santoyo Ramón, «Contribuciones al estudio y análisis de sistemas vestibles de detección de caídas,» Málaga, 2021.

[13] A. Shazhad y K. Kim, «FallDroid: An Automated Smart-Phone-Based Fall Detection System Using Multiple Kernel Learning,» *IEEE Transactions on Industrial Informatics,* vol. 15, nº 1, pp. 35-44, 2019.

[14] A. K. Bourke y G. M. Lyons, «A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor,» *Medical Engineering & Physics,* vol. 30, nº 1, pp. 84-90, 2008.

[15] Y. Su, D. Liu y Y. Wu, «A multi-sensor based pre-impact fall detection system with a hierarchical classifier,» de *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, Datong, China, 2016.

[16] W. Lu, C. Wang, M. C. Stevens, S. J. Redmond y N. H. Lovell, «Low-power operation of a barometric pressure sensor for use in an automatic fall detetor,» de *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Orlando, FL, USA, 2016.

[17] J. Light, S. Cha y M. Chowdhury, «Optimizing pressure sensor array data for a smart-shoe fall monitoring system,» de *2015 IEEE SENSORS*, Busan, South Korea, 2015.

[18] D. Droghini, E. Principi, S. Squartini, P. Olivetti y F. Piazza, «Human Fall Detection by Using an Innovative Floor Acoustic Sensor,» *Smart Innovation, Systems and Technologies,* vol. 69, pp. 97-107, 2017.

[19] A. Irtaza, S. M. Adnan, S. Aziz, A. Javed, M. O. Ullah y M. T. Mahmood, «A framework for fall detection of elderly people by analyzing environmental sounds through acoustic local ternary patterns,» de *2017 IEEE International Conference on Systems, Man and Cybernetics*, Banff, AB, Canada, 2017.

[20] K. Chaccour, R. Darazi, A. H. el Hassans y E. Andres, «Smart carpet using differential piezoresistive pressure sensors for elderly fall detection,» de *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications*, Abu Dhabi, United Arab Emirates, 2015.

[21] J. M. Andujar Morgado y A. König, «Low-Power concept and prototype of distributed resistive pressure sensor array for smart floor and surfaces in intelligent environments,» de *International Multi-Conference on Systems, Signals & Devices*, Chemnitz, Germany, 2012.

[22] Y. Tian, G.-H. Lee, H. He, C.-Y. Hsu y D. Katabi, «RF-Based Fall Monitoring Using Convolutional Neural Networks,» *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* vol. 2, nº 3, pp. 1-24, 2018.

[23] H. Wang, D. Zhang, Y. Wang, Y. Ma y S. Li, «RT-Fall: A Real-Time and Contactless Fall Detection System with Commodity WiFi Devices,» *IEEE Transactions on Mobile Computing,* vol. 16, nº 2, pp. 511-526, 2016.

[24] D. Razum, G. Seketa, J. Vugrin y I. Lackovic, «Optimal threshold selection for threshold-based fall detection algorithms with multiple features,» de *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 2018.

[25] Y. Wu, Y. Su, Y. Hu, N. Yu y R. Feng, «A Multi-Sensor Fall Detection System Based on Multivariate Statistical Process Analysis,» *Journal of Medical and Biological Engineering,* vol. 39, nº 3, pp. 336-351, 2018.

[26] L. Ren y W. Shi, «Chameleon: Personalised and adaptive fall detection of elderly people in home-based environments,» *International Journal of Sensor Networks,* vol. 20, nº 3, pp. 163-176, 2016.

[27] S. Yu, H. Chen y R. A. Brown, «Hidden Markov Model-Based Fall Detection With Motion Sensor Orientation Calibration: A Case for Real-Life Home Monitoring,» *IEEE Journal of Biomedical and Health Informatics,* vol. 22, nº 6, pp. 1847-1853, 2018.

[28] W. Min, L. Yao, Z. Lin y L. Liu, «Support vector machine approach to fall recognition based on simplified expression of human skeleton action and fast detection of start key frame using torso angle,» *IET Computer Vision,* vol. 12, nº 8, pp. 1133-1140, 2018.

[29] A. Poonsri y W. Chiracharit, «Improvement of fall detection using consecutive-frame voting,» de *2018 International Workshop on Advanced Image Technology (IWAIT)*, Chiang Mai, Thailand, 2018.

[30] W.-C. Cheng y D.-M. Jhan, «Triaxial Accelerometer-Based Fall Detection Method Using a Self-Constructing Cascade-AdaBoost-SVM Classifier,» *IEEE Journal of Biomedical and Health Informatics,* vol. 17, nº 2, pp. 411-419, 2013.

[31] C. Medrano, R. Igual, I. García-Magariño, I. Plaza y G. Azuara, «Combining novelty detectors to improve accelerometer-based fall detection,» *Medical & Biological Engineering & Computing,* vol. 55, nº 10, pp. 1849-1858, 2017.

[32] M. Díaz Sola, «Fundación Alzheimer España,» [En línea]. Available: http://www.alzfae.org/fundacion/461/como-elegir-un-detector-de-caidas. [Último acceso: 28 02 2023].

[33] Sense4Care, «Fall Detector for the Elderly,» [En línea]. Available: https://fate.upc.edu/. [Último acceso: 01 03 2023].

[34] Comisión Europea, «A wearable miniaturized fall detection system for the elderly,» [En línea]. Available: https://cordis.europa.eu/project/id/231485/reporting/fr. [Último acceso: 01 03 2023].

[35] Miray Consulting, «Dispositivo detector de caídas Neat,» [En línea]. Available: https://www.mirayconsulting.com/dispositivo-detector-de-caidas-neat/. [Último acceso: 01 03 2023].

[36] Durcal, «Durcal. El reloj de teleasistencia que salva vidas,» [En línea]. Available: https://durcal.com/es/. [Último acceso: 01 03 2023].

[37] Apple Inc., «Utilizar la detección de caídas con el Apple Watch,» [En línea]. Available: https://support.apple.com/es-es/HT208944. [Último acceso: 01 03 2023].

[38] Microchip Technology Inc., «23.1 Configuration Bits,» de *PIC18F2331/2431/4331/4431 Data Sheet*, DS39616D, 2010, pp. 263-273.

[39] Microchip Technology Inc., «2.4.2 Target Connection Circuitry,» de *PICkit 3 User's Guide*, DS51795B, 2010, pp. 19-20.

[40] ST Microelectronics, LIS3LV02DL, Rev 2, 2008.

[41] Microchip Technology Inc., «7.43.2.7 SPI (2431 Family),» de *PIC18F Peripheral Library Help Document*, 2012, pp. 809-810.

[42] Microchip Technology Inc., «7.43.2.12 USART (2431 Family),» de *PIC18F Peripheral Library Help Document*, 2012, pp. 811-812.

[43] Microchip Technology Inc., «20.2 EUSART Baud Rate Generator,» de *PIC18F2331/2431/4331/4431 Data Sheet*, DS39616D, 2010, p. 221.

[44] Microchip Technology Inc., «13.0 TIMER1 MODULE,» de *PIC18F2331/2431/4331/4431 Data Sheet*, DS39616D, 2010, pp. 131-135.

[45] Microchip Technology Inc., «Legacy Peripheral Libraries,» 7 January 2020. [En línea]. Available: https://microchipsupport.force.com/s/article/Legacy-Peripheral-Libraries. [Último acceso: 26 December 2022].

[46] Microchip Technology Inc., «Conversion to shorter data type,» 14 January 2020. [En línea]. Available: https://microchipsupport.force.com/s/article/Conversion-to-shorter-data-type. [Último acceso: 26 December 2022].

[47] m. (. user), «HELP: Fixing Warning (752) conversion to shorter data type,» 14 August 2014. [En línea]. Available: https://www.microchip.com/forums/m815346.aspx. [Último acceso: 26 December 2022].

[48] Microchip Technology Inc., RN4020 Bluettoth Low Energy Module User's Guide, DS70005191B, 2014.

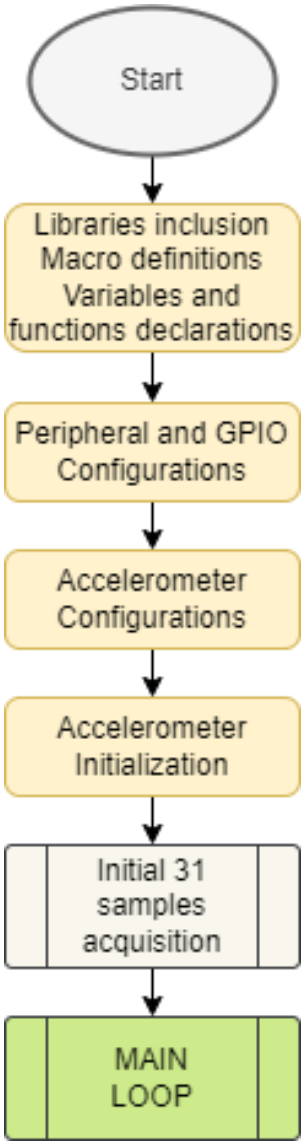
[49] Microchip Technology Inc., RN4020, DS50002279C, 2021.

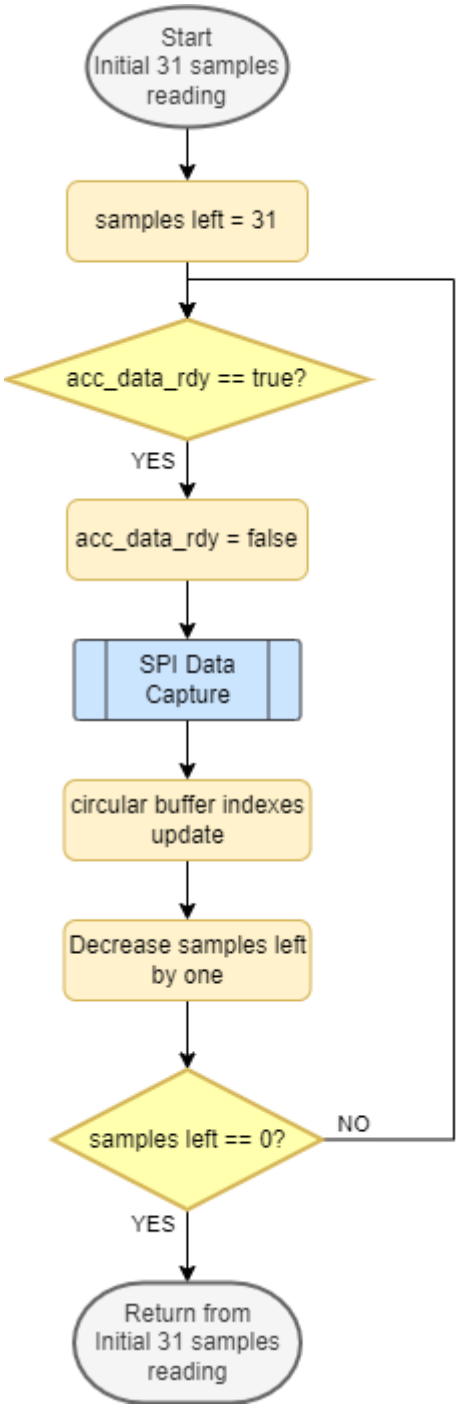
[50] J. A. Santoyo Ramón y E. Casilari Pérez, «UMAFall: Fall Detection Dataset,» Universidad de Málaga, [En línea]. Available: https://figshare.com/articles/dataset/UMA_ADL_FALL_Dataset_zip/4214283. [Último acceso: 3 junio 2023].

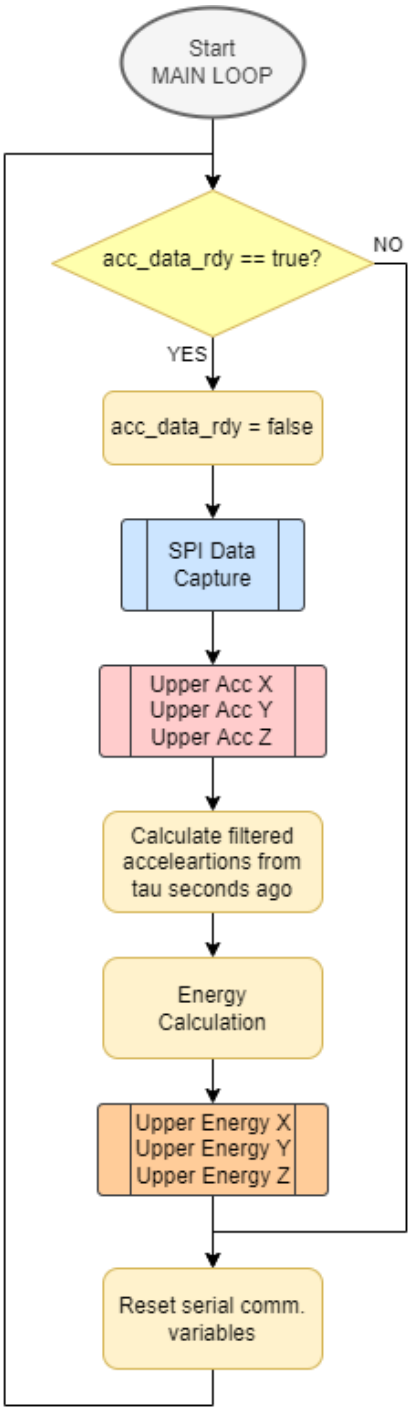
GLOSARIO

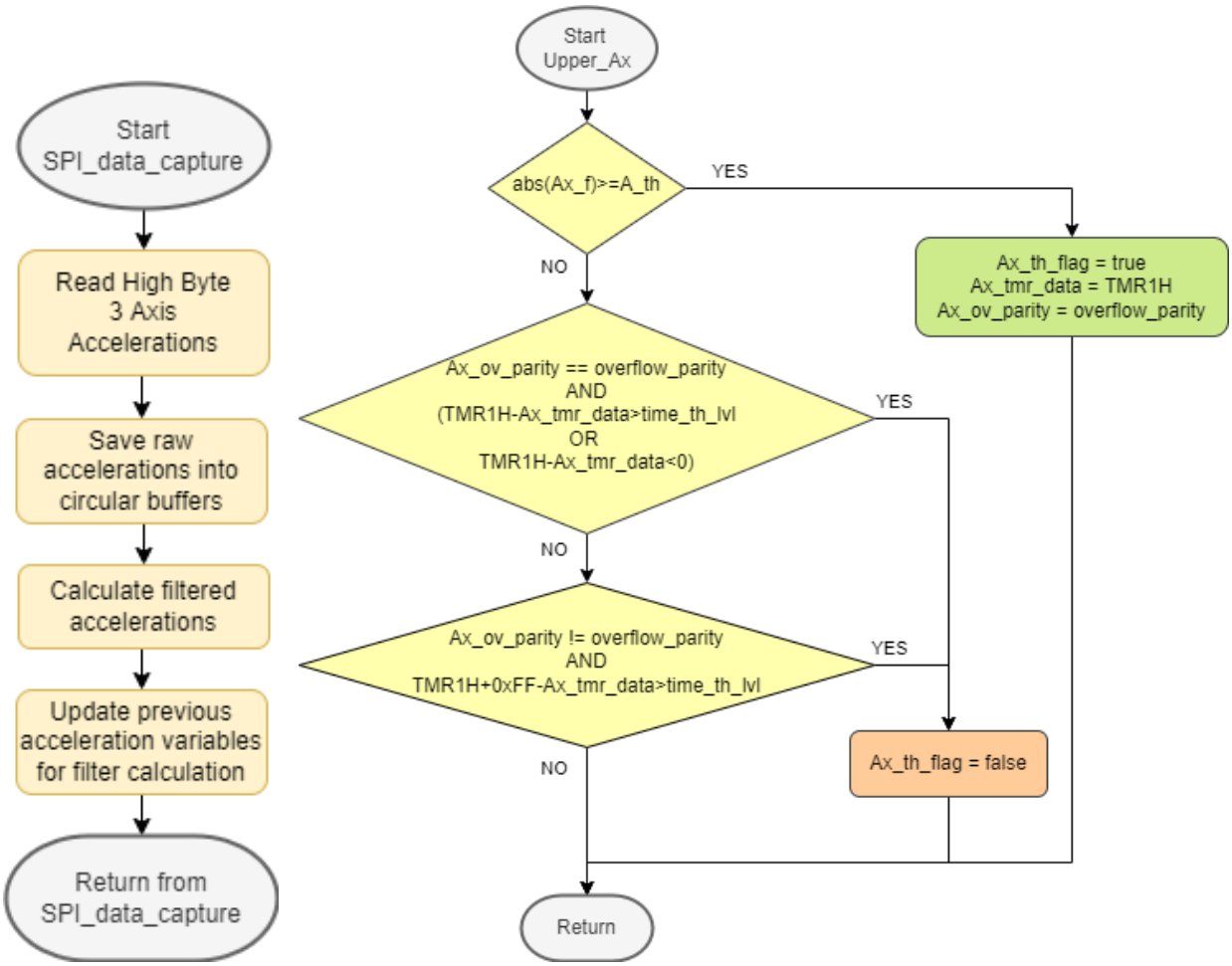
SI	Sistema Internacional
SoM	Sensor of Movements
DAD	Decision-Analysis Device
RTC	Remote Telehealthcare Center
OMS	Organización Mundial de la Salud
IDE	Integrated Development Environment
SPI	Serial Peripheral Interface
USART	Universal Synchronous-Asynchronous Receiver Transmitter
WDT	WatchDog Timer
PCB	Printed Circuit Board
BLE	Bluetooth Low Energy
MCU	Micro Controller Unit

ANEXO A – DIAGRAMAS DE FLUJO DEL SISTEMA




Start
Initial 31 samples reading
samples left = 31
acc_data_rdy == true?
YES
NO
acc_data_rdy = false
SPI Data Capture
circular buffer indexes update
Decrease samples left by one
samples left == 0?
YES
NO
Return from Initial 31 samples reading


Start
MAIN LOOP
acc_data_rdy == true?
NO
YES
acc_data_rdy = false
SPI Data Capture
Upper Acc X
Upper Acc Y
Upper Acc Z
Calculate filtered acceleartions from tau seconds ago
Energy Calculation
Upper Energy X
Upper Energy Y
Upper Energy Z
Reset serial comm. variables


Start
SPI_data_capture
Read High Byte
3 Axis
Accelerations
Save raw
accelerations into
circular buffers
Calculate filtered
accelerations
Update previous
acceleration variables
for filter calculation
Return from
SPI_data_capture
Start
Upper_Ax
abs(Ax_f)>=A_th
YES
NO
Ax_th_flag = true
Ax_tmr_data = TMR1H
Ax_ov_parity = overflow_parity
Ax_ov_parity == overflow_parity
AND
(TMR1H-Ax_tmr_data>time_th_lvl
OR
TMR1H-Ax_tmr_data<0)
YES
NO
Ax_ov_parity != overflow_parity
AND
TMR1H+0xFF-Ax_tmr_data>time_th_lvl
YES
NO
Ax_th_flag = false
Return

ANEXO B – CÓDIGO FUENTE DESARROLLADO

configbits.c

```
// PIC18F2431 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1H
#pragma config OSC = IRC        // Oscillator Selection bits (Internal
oscillator block, CLKO function on RA6 and port function on RA7)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable bit
(Fail-Safe Clock Monitor enabled)
#pragma config IESO = OFF          // Internal External Oscillator
Switchover bit (Internal External Switchover mode disabled)

// CONFIG2L
#pragma config PWRTEN = ON       // Power-up Timer Enable bit (PWRT
enabled)
#pragma config BOREN = ON       // Brown-out Reset Enable bits (Brown-
out Reset enabled)
// BORV = No Setting

// CONFIG2H
#pragma config WDTEN = ON         // Watchdog Timer Enable bit (WDT
enabled)
#pragma config WDPS = 256       // Watchdog Timer Postscale Select bits
(1:256)
#pragma config WINEN = OFF       // Watchdog Timer Window Enable bit
(WDT window disabled)

// CONFIG3L
#pragma config PWMPIN = OFF     // PWM output pins Reset state control
(PWM outputs disabled upon Reset (default))
#pragma config LPOL = HIGH      // Low-Side Transistors Polarity (PWM0,
2, 4 and 6 are active-high)
#pragma config HPOL = HIGH        // High-Side Transistors Polarity
(PWM1, 3, 5 and 7 are active-high)
#pragma config T1OSCMX = ON      // Timer1 Oscillator MUX (Low-power
Timer1 operation when microcontroller is in Sleep mode)

// CONFIG3H
#pragma config MCLRE = ON       // MCLR Pin Enable bit (Enabled)

// CONFIG4L
#pragma config STVREN = OFF      // Stack Full/Underflow Reset Enable
bit (Stack full/underflow will not cause Reset)
#pragma config LVP = OFF         // Low-Voltage ICSP Enable bit (Low-
voltage ICSP disabled)

// CONFIG5L
#pragma config CP0 = OFF       // Code Protection bit (Block 0 (000200-
000FFFh) not code-protected)
```

```
  #pragma config CP1 = OFF       // Code Protection bit (Block 1 (001000-
001FFF) not code-protected)
  #pragma config CP2 = OFF       // Code Protection bit (Block 2 (002000-
002FFFh) not code-protected)
  #pragma config CP3 = OFF       // Code Protection bit (Block 3 (003000-
003FFFh) not code-protected)

  // CONFIG5H
  #pragma config CPB = OFF       // Boot Block Code Protection bit (Boot
Block (000000-0001FFh) not code-protected)
  #pragma config CPD = OFF          // Data EEPROM Code Protection bit
(Data EEPROM not code-protected)

  // CONFIG6L
  #pragma config WRT0 = OFF         // Write Protection bit (Block 0
(000200-000FFFh) not write-protected)
  #pragma config WRT1 = OFF         // Write Protection bit (Block 1
(001000-001FFF) not write-protected)
  #pragma config WRT2 = OFF         // Write Protection bit (Block 2
(002000-002FFFh) not write-protected)
  #pragma config WRT3 = OFF         // Write Protection bit (Block 3
(003000-003FFFh) not write-protected)

  // CONFIG6H
  #pragma config WRTC = OFF          // Configuration Register Write
Protection bit (Configuration registers (300000-3000FFh) not write-
protected)
  #pragma config WRTB = OFF        // Boot Block Write Protection bit
(Boot Block (000000-0001FFh) not write-protected)
  #pragma config WRTD = OFF        // Data EEPROM Write Protection bit
(Data EEPROM not write-protected)

  // CONFIG7L
  #pragma config EBTR0 = OFF      // Table Read Protection bit (Block 0
(000200-000FFFh) not protected from table reads executed in other blocks)
  #pragma config EBTR1 = OFF      // Table Read Protection bit (Block 1
(001000-001FFF) not protected from table reads executed in other blocks)
  #pragma config EBTR2 = OFF      // Table Read Protection bit (Block 2
(002000-002FFFh) not protected from table reads executed in other blocks)
  #pragma config EBTR3 = OFF      // Table Read Protection bit (Block 3
(003000-003FFFh) not protected from table reads executed in other blocks)

  // CONFIG7H
  #pragma config EBTRB = OFF       // Boot Block Table Read Protection
bit (Boot Block (000000-0001FFh) not protected from table reads executed
in other blocks)

  // #pragma config statements should precede project file includes.
  // Use project enums instead of #define for ON and OFF.

  #include <xc.h>
```

SoM_v1.c

```
  /*
```

```
   * File:   SoM_v1.c
   * Author: Santiago Domínguez Vidal
   *
   * Created on 24th April 2021
   */

  //#define MATLAB_PRINT
  //#define DEBUG_PRINT
  #define PLOTTER_PRINT
  //#define VALIDATION

  #define _XTAL_FREQ 4000000F

  #define USE_OR_MASKS
  //#undef _PLIB
  #include  <xc.h>  //_delay(a),__delay_ms(x),__delay_us(x)->  x<179200
a<50463240
  #include <stdbool.h>
  #include <stdint.h>
  #include <stdlib.h>
  #include <stdio.h>
  #include <string.h>
  #include <math.h>
  #include "lis3lv02.h"
  //#include <EEP.h>

  #pragma warning disable 752
  #pragma warning disable 520

  #define TIME_LVL_TO_TH  (55F/5625F)

  #define TAU           0.175   //Time interval (in sec) between samples
for energy calculation
  #define ACC_SAMPLERATE  40
  #define LSBIT_TO_MG     50/17
  #define INIT_WAIT_COUNT 31

  #define P_INT_E     0
  #define G_INT_E     1
  #define PER_INT_E   1

  #define P_EEPROM    0
  #define E_EEPROM    0

  #define PRIORITY_RX 0
  #define PRIORITY_TX 0
  #define INTENABLE_RX 0
  #define INTENABLE_TX 0

  #ifndef VALIDATION          //Macro to enable validation code
      #define BUFFER_SIZE 85   //Size of the raw acceleration buffers
during normal execution
  #else
      #define BUFFER_SIZE 417 //Size of the raw acceleration buffers
during validation execution
  #endif
```

```
  #define  X_INDEX_MINUS_TAU                          ((int16_t)(x_index-
tau_samples)<0?x_index-tau_samples+BUFFER_SIZE:x_index-tau_samples)
  #define  X_INDEX_MINUS_TAU_MINUS_1     ((int16_t)(x_index-tau_samples-
1)<0?x_index-tau_samples-1+BUFFER_SIZE:x_index-tau_samples-1)
  #define  Y_INDEX_MINUS_TAU                          ((int16_t)(y_index-
tau_samples)<0?y_index-tau_samples+BUFFER_SIZE:y_index-tau_samples)
  #define  Y_INDEX_MINUS_TAU_MINUS_1     ((int16_t)(y_index-tau_samples-
1)<0?y_index-tau_samples-1+BUFFER_SIZE:y_index-tau_samples-1)
  #define  Z_INDEX_MINUS_TAU                          ((int16_t)(z_index-
tau_samples)<0?z_index-tau_samples+BUFFER_SIZE:z_index-tau_samples)
  #define  Z_INDEX_MINUS_TAU_MINUS_1     ((int16_t)(z_index-tau_samples-
1)<0?z_index-tau_samples-1+BUFFER_SIZE:z_index-tau_samples-1)

  //MACROS FOR PIN ACTIONS
  #define INPUT 1
  #define OUTPUT 0
  #define ON 1
  #define OFF 0
  #define LED LATCbits.LC6
  #define SIMULATED_GND LATCbits.LC7
  #define WAKE_HW LATCbits.LC4
  #define WAKE_SW LATAbits.LA0
  #define PIO7 LATAbits.LA1
  #define SPI_CS LATBbits.LB1
  #define SPI_SCLK LATBbits.LB2
  #define SPI_SDO LATBbits.LB3
  #define SPI_SDI PORTBbits.RB4
  #define RX_EN RCSTAbits.CREN
  #define TX_EN TXSTAbits.TXEN
  //ONLY NEEDED IF NOT USING USART LIBRARY
  //#define DataRdyUSART( ) (PIR1bits.RCIF)
  //enum{BAUD_19200=1, BAUD_38400, BAUD_57600, BAUD_115200};

  typedef struct {
      uint8_t tmr_data;
      bool    ov_parity;
  }timepoint_t;

  //COMMUNICATION VARIABLES
  char send_str[64];
  //char* receive_str;
  //char receive_char;
  //uint8_t str_index = 0;
  //bool str_complete = false;

  //**ACCEL AND IMPACT VARIABLES**//
  int8_t acc_threshold = 14;
  int8_t energy_threshold = 9;
  int8_t   time_threshold_lvl   =   9;      //Tth   =   0.0703125   s
[=(1/32768)*256*time_threshold_lvl]
  //float  time_threshold  =  9  *  TIME_LVL_TO_TH;  //NOT  USED  IN  THIS
APPROACH
  int16_t  acc_x  =  0,  prev_acc_x  =  0,  acc_x_f  =  0,  acc_x_f_tau  =  0,
energy_x = 0;
  int16_t  acc_y  =  0,  prev_acc_y  =  0,  acc_y_f  =  0,  acc_y_f_tau  =  0,
energy_y = 0;
```

```
  int16_t acc_z = 0, prev_acc_z = 0, acc_z_f = 0, acc_z_f_tau = 0,
energy_z = 0;
  int8_t acc_x_h_buffer[BUFFER_SIZE]={0} /*@ 0x100*/;
  int8_t acc_y_h_buffer[BUFFER_SIZE]={0} /*@ 0x1AA*/;
  int8_t acc_z_h_buffer[BUFFER_SIZE]={0} /*@ 0x254*/;
  //Experiment 1 - SIZE:417
  //const  int8_t  acc_x_h_buffer[BUFFER_SIZE]={3,3,3,4,3,-1,-1,2,4,2,-
1,-4,-5,-4,-3,-1,0,0,0,-1,-2,-2,-3,-3,-2,-2,-2,-3,-3,-3,-3,-4,0,1,-3,-
6,-4,1,3,2,-1,-1,0,1,1,1,2,2,2,2,2,1,1,1,1,2,1,-4,-2,3,5,2,-2,-4,-3,-
2,-2,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-3,-4,-3,-2,-2,0,1,-3,-7,-
4,1,3,3,0,-1,0,1,2,2,2,2,1,0,0,1,3,1,0,-1,-3,-2,1,4,3,1,0,-1,-2,-2,-
2,-2,-1,-1,-1,-1,-2,-2,-2,-2,-4,-5,-2,0,0,3,1,-5,-8,-
4,2,4,3,0,1,3,2,1,0,0,1,1,0,0,1,1,1,0,-4,-4,0,3,3,1,0,-2,-2,-2,-3,-2,-
2,-2,-2,-2,-3,-3,-2,-3,-3,-2,0,1,4,0,-7,-8,-2,4,4,1,-
1,1,3,3,2,1,1,2,2,1,1,1,1,2,2,-1,-4,-3,2,5,4,1,-3,-3,-3,-2,-2,-2,-3,-
2,-1,-1,-2,-2,-2,-3,-3,0,1,1,3,-3,-8,-
6,3,7,4,0,0,2,3,2,1,1,2,2,1,2,2,0,0,2,0,-4,-3,2,4,4,0,-3,-4,-4,-3,-3,-
3,-2,-1,-1,-1,-2,-3,-4,-3,-2,-2,1,4,1,-6,-7,-
1,5,5,1,0,1,2,2,1,1,1,2,2,1,1,0,0,2,2,-2,-5,-2,3,4,2,0,-2,-2,-1,-1,-
2,-2,-2,-2,0,0,-2,-3,-4,-5,-4,-2,-2,2,1,-2,-5,-
4,1,4,4,1,1,2,3,2,1,0,0,2,2,1,1,-1,-1,1,1,-1,-2,2,3,2,0,-1,-1,-1,-1,-
2,-1,-1,0,-1,-2,-4,-4,-4,-5,-3,-2,-1,4,2,-6,-
7,0,5,4,1,0,2,4,4,2,1,1,1,1,1,1,1,0,0,0,-3,-2,1,2,3,1,-2,-2,-2,-1,-1,-
1,-1,-1,-1,-1,-2,-3,-3,-4,-4,-2,-1,1,3,-1,-6,-6};
  //const                                                    int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,21,21,24,26,23,23,25,25,22,19,20,22
,24,23,21,20,20,21,21,21,21,21,21,20,19,20,22,23,23,26,29,28,25,22,22,
21,22,23,22,22,21,20,20,20,20,20,21,21,21,19,19,22,25,26,29,29,26,24,2
4,21,20,21,21,21,21,20,20,20,20,21,21,20,20,21,20,20,21,23,24,25,29,29
,25,22,22,22,23,23,21,20,20,19,19,21,22,21,20,20,19,20,24,25,23,25,28,
29,28,24,20,20,22,22,21,20,20,21,21,20,20,20,20,20,20,19,18,20,23,24,2
6,31,29,23,22,23,24,24,22,21,20,19,19,19,21,21,21,21,20,21,22,23,24,26
,29,28,25,23,21,21,22,22,21,21,20,19,20,20,21,21,21,21,19,18,19,22,25,
28,30,26,22,23,23,24,24,23,22,21,19,18,19,20,20,20,20,20,21,23,25,23,2
5,29,30,28,24,20,20,22,21,20,20,20,20,20,20,21,21,21,21,20,19,19,21,23
,24,29,31,26,21,20,24,25,22,21,22,20,19,19,19,19,20,20,20,20,21,23,23,
24,29,31,28,24,22,22,23,22,20,20,20,20,20,20,20,20,20,20,19,19,21,23,2
4,29,30,25,22,22,23,25,25,22,22,21,19,18,18,19,20,21,21,20,21,23,24,23
,26,29,30,27,23,19,19,23,23,21,21,21,21,20,19,19,19,20,20,20,19,19,21,
23,26,30,29,27,23,21,22,25,24,21,20,20,19,18,18,19,20,21,21,21,21,23,2
4,25,29,29,28,25,21,20,22,22,21,20,19,20,20,20,20,20,21,20,19,19,20,22
,23,27,31,28,22,22,24,25,25,23,22,20,19,19,19,19,19,20,21,20,21,22,24,
24,28,30,28,26,23,20,20,22,21,20,20,20,20,20,19,20,21,21,21,19,19,21,2
3,25,29,31,28,23,20};
  //const    int8_t    acc_z_h_buffer[BUFFER_SIZE]={-3,-3,-4,-4,-2,-1,-
1,1,1,2,2,2,1,0,0,0,0,0,0,0,0,-1,-1,-2,-2,-2,-3,-3,-3,-4,-5,-3,-2,-1,-
1,-1,1,2,2,2,1,0,0,-1,-2,-2,-2,-2,-2,-2,-2,-3,-4,-5,-6,-6,-4,-3,-2,-
1,0,2,3,2,1,1,0,0,0,-1,-1,-1,-1,-2,-3,-3,-4,-4,-4,-4,-5,-5,-2,-
1,0,0,1,2,2,1,0,0,0,-1,-2,-2,-2,-3,-3,-4,-4,-4,-4,-5,-6,-6,-6,-
1,2,2,3,2,1,1,0,0,0,-1,-2,-3,-3,-3,-4,-4,-4,-3,-4,-5,-4,-5,-4,-1,-1,-
1,-1,1,2,1,1,0,0,0,-1,-2,-2,-2,-2,-2,-3,-4,-5,-6,-6,-4,-
3,0,1,2,3,2,1,1,1,1,0,-1,-2,-2,-2,-3,-3,-2,-3,-4,-5,-4,-4,-2,0,-1,-2,-
1,1,2,1,1,0,0,0,-1,-2,-2,-2,-3,-3,-4,-5,-5,-6,-7,-6,-4,-
1,1,2,3,2,1,0,0,0,0,-1,-1,-2,-2,-2,-3,-3,-3,-4,-4,-4,-4,-4,-1,0,-1,-
1,0,2,2,1,1,1,0,0,-1,-1,-1,-2,-3,-4,-5,-5,-5,-7,-6,-
2,1,2,4,3,1,1,0,0,-1,-1,-1,-2,-2,-2,-3,-3,-4,-4,-5,-4,-4,-4,-2,-1,-1,-
2,-1,0,1,1,0,0,-1,-1,-2,-2,-2,-3,-3,-3,-3,-3,-4,-5,-7,-7,-4,-
```

```
1,1,1,2,2,1,0,0,0,0,-1,-2,-2,-3,-3,-3,-4,-3,-4,-4,-4,-4,-3,-1,0,0,-
1,0,1,1,1,0,0,-1,-1,-2,-3,-3,-3,-3,-3,-3,-4,-5,-6,-6,-5,-
3,0,1,2,2,1,1,1,1,0,-1,-2,-2,-2,-3,-3,-3,-4,-4,-4,-4,-4,-
2,1,1,0,0,1,1,1,0,0,-1,-1,-1,-1,-2,-2,-3,-3,-4,-4,-5,-6,-6,-5,-
3,0,0,2,2,1,1,1,1,1,0,-1,-2,-3,-3,-3,-4,-4,-3,-4,-3,-3,-4,-2,0,0,-1,-
1};
 //Experiment 2 - SIZE:683
 //const int8_t acc_x_h_buffer[BUFFER_SIZE]={-3,-3,-3,-3,-3,-3,-3,-3,-
4,-4,-4,-4,-4,-4,-4,-5,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-3,-3,-2,-
2,-3,-3,-3,-3,-3,-2,-1,-1,-
1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,2,1,1,2,3,3,3,4,3,3,3,3,4,4,4,3,3,
3,3,2,1,2,3,3,1,0,2,3,3,3,3,2,2,2,1,-1,-1,-1,-2,-2,-3,-2,-2,-1,-1,-2,-
2,-2,-2,-2,-3,-3,-3,-3,-3,-3,-3,-3,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-
3,-3,-3,-3,-2,-3,-3,-3,-2,-2,-2,-1,-1,-3,-4,-5,-4,-2,-
1,0,1,1,0,1,2,2,1,0,0,0,1,1,1,1,0,1,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,
1,2,3,2,1,0,1,2,3,3,2,2,1,0,0,0,0,0,-2,-2,-2,-3,-3,-2,-2,-2,-2,-3,-4,-
4,-4,-4,-4,-4,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-4,-3,-4,-5,-5,-4,-
3,-3,-3,-3,-4,-5,-5,-
2,1,3,2,1,2,2,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4,4,3,3,3,2,
2,2,2,3,3,2,2,2,2,3,2,1,1,1,0,-1,-1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
3,-3,-3,-4,-4,-4,-4,-4,-4,-5,-5,-5,-4,-4,-5,-5,-5,-4,-4,-5,-6,-5,-4,-
4,-4,-4,-4,-4,-
2,0,1,1,2,2,2,2,2,1,0,0,1,2,1,1,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,3,
3,3,2,1,2,3,3,3,2,2,1,-1,-1,-1,-1,-2,-2,-1,-1,-1,-1,-2,-2,-2,-2,-2,-
2,-3,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-4,-
4,-4,-4,-4,-3,-
2,0,1,1,1,2,2,1,1,0,0,1,1,1,1,1,2,2,2,3,3,2,3,3,3,3,3,3,3,3,3,3,2,3,3,
3,3,2,1,1,2,3,2,0,-1,-2,-2,-2,-3,-2,-1,-1,-1,-1,-1,-2,-3,-3,-3,-4,-4,-
4,-4,-3,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-4,-4,-4,-4,-3,-2,-3,-4,-
4,-
2,0,2,1,1,1,1,1,1,1,1,1,1,1,1,2,3,3,4,4,3,3,3,3,3,3,3,4,3,3,2,1,2,2,
2,2,1,3,3,3,1,0,-1,-1,-2,-3,-3,-3,-2,-2,-2,-2,-2,-3,-3,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-5,-4,-4,-4,-4,-4,-4,-3,-2,-2,-3,-3,-4,-
3,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,2,2,3,3,3,2,2,3,2,2,2,2,3,3,3,2,2,2,2,
2,2,1,1,1,2,1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-
2,-3,-4,-4,-4,-4,-4,-4,-4,-4,-5,-5,-5,-5,-4,-5,-5,-5,-5,-3,-2,-3,-4,-4,-
3,-1,-1,-2,-4,-4,-2,0,1,2,2,2,2};
 //const                                                    int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,23,22,22,22,21,22,22,22,22,22,21,21,22
,22,21,21,22,22,23,23,22,22,22,22,22,22,22,22,23,23,23,24,24,25,26,25,
25,25,24,22,21,20,20,20,20,21,21,22,22,22,23,22,22,22,21,20,20,21,21,2
1,21,21,21,21,21,21,21,21,21,21,22,22,22,22,23,23,23,22,22,23,23,23,23
,23,23,24,25,26,26,25,25,24,22,21,22,22,21,21,20,20,20,19,19,19,19,19,
19,19,20,20,20,21,21,21,21,21,21,21,21,21,22,22,21,21,22,22,23,23,23,2
2,22,22,22,22,22,23,23,24,25,27,27,27,26,25,23,20,19,20,21,21,21,21,21
,21,20,20,20,20,20,20,20,20,21,21,21,21,21,21,21,22,21,21,21,21,22,22,
23,23,23,22,22,23,24,24,24,24,24,26,27,27,26,25,23,20,21,23,22,22,21,2
1,20,20,19,19,18,18,18,19,19,19,19,20,20,21,21,22,21,21,21,21,21,22,23
,23,23,23,22,22,22,22,23,24,25,26,27,27,26,25,22,20,18,20,21,21,21,21,
21,21,20,20,20,19,19,19,20,20,20,21,21,21,21,21,21,21,22,22,22,23,23,2
3,22,22,22,23,23,22,23,23,23,23,24,25,26,26,25,25,24,22,21,21,22,22,22
,21,21,21,21,20,20,19,19,19,19,19,19,19,19,19,20,20,21,21,21,21,21,22,
22,23,25,24,23,23,23,23,24,25,26,26,25,23,21,20,21,22,22,21,22,21,20,2
0,20,19,19,18,19,19,19,20,20,21,21,21,21,21,22,22,22,23,23,23,23,24,23
,23,23,23,23,24,25,27,27,27,26,25,23,21,20,22,22,22,21,21,21,20,20,19,
19,18,18,19,19,19,20,20,21,21,21,21,21,21,21,21,22,22,22,23,23,22,22,2
4,24,25,27,28,28,27,25,22,20,20,21,22,21,21,21,21,21,20,20,20,20,19,19
```

```
,20,20,21,21,21,22,21,22,22,22,22,22,22,22,22,23,23,23,22,23,24,25,25,
26,27,28,28,26,24,21,22,23,21,20,20,20,19,18,18,18,18,18,18,19,19,20,2
1,21,21,21,21,21,21,21,21,22,22,22,23,23,23,23,24,25,25,27,28,28,27,24
,22,20,20,21,21,21,21,21,20,20,20,19,19,19,19,19,19,20,20,21,21,22,22,
22,22,22,22,22,23,23,24,23,23,24,25,25,26,27,27,27,26,23,21,22,24,23,2
1,21,20,19,19,18,17,18,18,18,18,18,20,20,21,21,22,22,22,22,22,21,22,22
,23,23,23,23,23,23,24,25,26,27,27,26,24,22,20,20,21,23,22,22,22,21,20,
20,19,19,19,18,19,19,20,20,21,22,22,22,22,22,22,22,22,22,23,23,24,24,2
3,23,24,24,25,26,26,26,25,24,23,21,20,21,22,22,22,22,22,21,21,20,20,20
,20,19,19,19,19,19,20,20,20,21,21,21,21,21,21,22,23,23,23,23,23,23,23,
22,23,23,24,26,27,28,27,24,21,20,20,20,20,20,21};
 //const                                                    int8_t
acc_z_h_buffer[BUFFER_SIZE]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,1,0,0,0,1,1,1,2,2,3,3,3,4,4,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,5,5
,5,4,4,4,3,3,3,3,2,2,2,2,2,2,1,1,1,1,1,1,2,2,2,2,2,3,3,3,4,5,5,6,6,6,7
,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,5,6,6,5,5,5,5,5,4,4,4,4,3,3,3,3,3,2,2,2,2
,2,2,3,3,3,3,3,3,3,3,3,4,4,4,4,5,6,6,6,6,7,7,7,7,7,7,6,6,6,5,5,5,5,5,5
,4,4,4,4,3,3,3,3,3,2,2,2,2,2,2,2,2,3,3,2,2,3,4,4,4,5,6,6,6,6,7,7,7,7,7
,7,7,6,6,6,5,5,5,5,4,4,4,4,4,4,4,3,3,3,3,3,3,3,2,2,2,2,2,2,3,3,3,3,3,3
,3,4,5,5,6,6,7,7,7,7,7,7,7,7,7,6,6,5,5,5,4,4,4,4,4,4,4,4,4,3,3,3,3,2,2
,2,2,2,2,2,2,2,2,2,3,3,3,3,4,4,4,5,6,7,7,8,8,8,7,7,7,8,7,7,7,6,6,6,5,5
,5,5,5,4,4,4,4,3,3,3,3,3,3,2,2,2,2,3,4,4,4,4,5,5,5,6,7,7,7,8,8,8,8,8,8
,7,7,6,6,5,5,5,4,4,4,4,4,3,3,3,3,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,4,4,4,4
,5,5,6,6,6,6,6,6,7,6,6,6,5,5,4,4,4,4,4,4,4,3,3,3,3,2,2,2,2,2,1,1,1,1,1
,2,3,2,2,3,3,4,4,4,5,5,5,5,5,5,6,6,5,5,5,4,4,4,3,3,3,3,2,2,2,2,2,2,1,1
,1,1,1,1,1,1,2,2,3,3,3,3,3,3,4,4,4,5,7,6,6,6,6,6,6,5,5,5,5,4,4,4,4,4
,4,3,3,3,3,2,2,2,2,2,2,2,2,1,1,2,2,2,3,2,3,3,3,4,5,5,6,6,6,6,6,6,6,6,5
,5,5,4,4,4,4,4,3,3,3,3,3,2,2,2,2,2,2,1,1,1,1,2,2,3,4,4,4,4,4,4,6,6,6,6
,6,6,6,6,5,5,5,5,4,4,4,3,3,3,3,3,2,2,2,2,2,2,1,1,1,1,1,1,1,1,2,2,3,3,3
,3,4,4,5,6,6,6,6,6,6,6,6,6,5,5,4,4,3,3,3,3,3,3,2,2,2,2,1,1,2,2,1,1,1,1
,1,1,2,2,3,4,4,4,4,5,5,6,6,6,6,6,6,6,6,7,7,7,7,6,6,5,5,4,4,4,4,3,3,3,2,2
,2,2,2,2,2,2,2,1,1,1,1,1,1,1,2,2,2,2,2,3,4,4,5,5,5,6,6,6,6,6,6,5};
 //Experiment 3 - SIZE:569
 //const  int8_t  acc_x_h_buffer[BUFFER_SIZE]={-4,-4,-4,-3,-3,-1,2,1,-
3,-4,-2,-1,-1,-2,-2,-2,-2,-3,-3,-3,-3,-2,-2,-3,-3,-3,-2,-1,-1,-2,-2,-
3,-3,-2,-2,-1,-1,-2,-2,-2,-1,-1,-1,2,3,1,-1,-1,2,3,-1,-
1,2,2,2,2,1,1,2,3,3,2,1,1,1,2,3,3,2,2,2,1,1,2,3,2,1,1,1,1,1,0,0,0,2,-
1,-4,-3,-1,1,2,0,-4,-5,-4,-2,-1,-1,-2,-2,-3,-3,-4,-4,-3,-2,-2,-2,-3,-
3,-3,-3,-2,-2,-1,-1,-1,-2,-2,-2,-2,-1,-1,-1,-1,-
1,2,3,2,0,0,1,2,0,0,2,3,2,2,1,1,2,3,3,3,2,2,2,3,3,2,2,2,2,2,2,1,1,1,1,
1,1,2,1,1,1,0,0,1,-2,-3,-2,-1,0,1,1,-5,-5,0,1,-1,-3,-3,-2,-2,-3,-3,-
3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-1,-1,-2,-2,-2,-1,-
1,0,2,4,2,0,-2,-
1,3,2,0,1,3,2,1,1,2,2,2,3,3,2,2,3,2,2,2,2,3,2,1,0,0,1,1,1,1,1,1,0,0,0,
1,1,0,1,0,-2,-1,0,0,-1,-1,1,-1,-4,-5,-4,-2,-1,-1,-2,-2,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-1,-2,-2,-2,-2,-2,-1,-1,-1,-2,-1,4,3,-1,-
4,-3,1,2,0,0,2,2,2,2,2,2,2,2,2,3,2,2,1,2,2,2,1,1,2,1,-1,-1,0,1,1,0,0,-
1,-6,-6,-1,3,5,1,-7,-6,-1,-1,-3,-4,-4,-2,-2,-3,-4,-4,-4,-3,-3,-3,-4,-
4,-3,-2,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-2,1,3,3,0,-3,-
2,2,2,0,1,1,1,1,2,2,3,3,3,2,2,2,2,1,1,1,1,1,0,-1,0,1,1,0,-1,0,1,-2,-
4,-2,2,3,1,-1,-3,-3,-2,-2,-2,-2,-2,-1,-1,-2,-3,-4,-3,-3,-3,-4,-4,-3,-
3,-3,-3,-2,-1,-1,-1,-1,-1,-1,-2,-2,1,7,7,1,-4,-
3,1,1,0,2,2,1,2,4,3,2,2,3,3,3,3,3,3,2,2,3,3,3,2,2,2,1,-1,-1,0,0,-1,-
1,0,1,-2,-3,-1,1,2,2,-1,-4,-5,-4,-2,-2,-2,-2,-2,-3,-3,-4,-4,-4,-3,-3,-
4,-4,-3,-3,-3,-3,-2,-1,-1,-2,-2,-1,-1,-1,-1,0,0,2,5,3,-2,-3,0,1,-1,-
1,3,4,3,3,3,2,2,3,3,3,3,3,3,2,1,2,3,2,1,0,1,1,0,-1,1,3,2,-9,-8,1,5,0,-
5,-1};
```

```
 //const                                                          int8_t
acc_y_h_buffer[BUFFER_SIZE]={25,26,28,29,30,27,25,25,24,23,22,21,22,22
,22,21,21,21,21,22,21,20,20,20,19,19,19,18,18,19,20,20,20,21,21,21,22,
22,21,22,22,22,24,28,26,26,27,27,28,26,24,25,24,23,23,23,21,21,21,21,2
0,20,21,20,20,21,21,21,20,20,20,20,19,20,20,20,20,20,21,21,21,21,21,20
,20,25,26,26,28,28,27,27,27,26,26,25,24,22,21,21,20,20,20,19,18,19,20,
20,20,19,20,20,20,21,21,21,21,21,21,22,22,22,22,22,21,22,26,26,25,26,2
7,28,27,24,25,26,23,23,24,23,21,21,21,20,20,20,20,20,21,20,20,20,20,20
,20,20,20,21,21,21,22,21,22,22,22,21,20,21,22,25,25,26,27,25,24,26,29,
24,23,25,25,23,22,22,21,21,21,20,19,19,20,20,20,20,20,20,20,19,19,19,2
1,21,21,21,22,22,22,21,21,21,22,25,26,25,25,27,29,30,26,23,25,26,24,22
,21,21,21,21,21,21,21,21,21,20,20,20,21,21,20,19,19,20,20,21,21,22,21,
21,22,22,22,22,21,20,21,23,24,24,26,28,29,27,25,26,25,24,24,23,22,22,2
2,21,21,20,20,19,20,19,19,20,21,21,20,19,18,19,20,20,20,20,21,22,22,22
,22,21,23,26,27,26,26,27,28,26,25,26,26,25,23,22,21,21,20,20,20,20,19,
19,19,20,20,19,19,20,20,20,20,21,21,21,20,19,18,22,29,27,27,29,26,28,3
0,25,24,26,25,23,23,22,21,21,20,20,19,18,17,18,18,19,19,19,20,20,20,21
,21,21,21,21,22,22,22,20,22,25,25,25,25,25,28,29,26,26,27,26,25,24,22,
22,21,20,19,19,19,18,18,19,19,19,19,19,19,20,20,20,20,20,21,20,21,26,2
8,26,27,29,28,27,27,26,26,25,24,23,22,21,21,20,20,19,18,18,18,18,18,18
,18,19,19,19,19,21,21,21,21,20,20,20,24,30,28,26,28,29,28,26,26,27,24,
22,23,23,22,21,21,21,21,21,21,21,21,21,20,20,20,20,18,18,20,20,19,19,2
0,20,20,20,21,19,19,23,25,26,28,28,27,28,28,26,24,25,26,25,24,23,22,22
,21,21,20,19,19,18,18,18,19,20,20,19,20,21,21,20,20,19,18,18,18,20,26,
27,26,28,31,31,28,25,27,26,23,22,23,23,22,22,21,21,20,20,19,19,18,18,1
8,19,18,18,20,21,20,18,17,17,16,21,31,29,28,33,35,30,27};
 //const            int8_t            acc_z_h_buffer[BUFFER_SIZE]={-1,-
1,0,1,1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-2,-1,-
1,-2,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-1,0,-1,-1,-1,0,1,0,-2,-1,-1,-
1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,-1,0,0,0,-1,-1,-1,-2,-2,-2,-2,-2,-
2,-3,-3,-2,0,0,1,1,2,1,0,0,-1,-1,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,-2,-
1,-2,-2,-2,-1,-1,-1,-2,-2,-2,-2,-2,-1,-1,-
1,0,1,1,0,0,1,1,1,0,0,1,1,1,2,2,2,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,-1,-
1,-1,-1,-2,-2,-2,-2,-2,-2,-3,-2,0,0,0,0,0,0,1,1,-1,-2,-1,-1,-
1,0,0,0,0,0,0,0,-1,-1,-1,-1,-2,-2,-1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-
1,-1,-1,-2,-1,1,1,0,-1,0,1,2,1,-1,-
1,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,-1,-2,-2,-2,-
2,-2,-2,-2,-1,0,0,0,0,1,1,1,0,0,-1,-1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-2,-1,-2,-2,-2,-1,1,1,0,-1,-1,1,2,0,-1,-
1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,-1,-1,-2,-1,-1,-2,-2,-2,-2,-2,-2,-2,-
2,-1,1,1,2,2,1,2,1,-1,-1,0,1,1,0,1,1,0,0,0,1,0,0,-1,-1,-1,-1,-1,-1,-
1,-1,-2,-1,-1,-1,-2,-2,-2,-2,-2,-1,1,1,0,-1,0,1,1,0,-2,-2,-2,-1,-1,-
1,-1,-1,0,0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-2,-2,-3,-3,-3,-3,-3,-
1,1,2,1,2,2,1,1,1,0,0,1,1,1,2,2,2,2,1,1,1,0,0,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-2,-1,1,2,1,0,-1,0,1,0,-1,0,0,0,0,1,0,0,0,0,0,0,0,-
1,-1,-1,0,-1,-1,-1,-1,-1,-1,-2,-2,-2,-2,-2,-2,-3,-2,-
1,0,1,1,0,1,0,0,0,0,-1,-1,-1,-1,0,0,0,0,0,0,0,0,0,0,-1,-1,-2,-2,-1,-
1,-1,-1,-1,-1,-1,0,0,-1,0,0,1,3,2,1,-1,0,1,0,-1,-
1,0,0,1,1,0,1,1,1,1,1,0,0,0,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-3,-4,-4,-
2,1,2,2,2,2,0,0};
 //Experiment 4 - SIZE:415
 //const                                                          int8_t
acc_x_h_buffer[BUFFER_SIZE]={2,2,2,2,2,2,2,1,1,1,1,0,0,0,0,0,0,-1,-1,-
1,-1,-1,0,0,-1,-
1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,2,2,2,2,2,2,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,0,-
```

```
1,0,1,1,1,1,2,1,1,1,1,1,2,1,1,1,2,2,2,2,2,2,2,2,3,2,2,2,2,2,1,2,3,2,1,
3,3,2,3,3,2,2,2,2,2,2,2,1,1,1,2,1,1,0,0,0,0,0,0,-1,0,0,0,0,-1,-
1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,
0,0,-1,-1,0,0,-1,0,-1,-1,-
1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,2,2,2,
2,2,2,2,2,2,2,3,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,2,1,1,1,2,2,2,1,1,2,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,2,2,2,2,2,
2,2,2,1,1,1,2,2,2,2,2,2,2,2,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2};
  //const                                                        int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,22,22,22,22,22,22,22,22,22,22,23
,23,22,22,22,22,21,21,21,22,22,22,22,22,22,23,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,23,23,23,23
,23,23,22,22,23,23,23,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,21,21,22,22,21,21,21,21,21,21,20,20,20,19,19,18,19,18,19,19,19,19,1
8,17,18,19,19,19,20,20,20,20,20,20,20,20,20,20,20,20,20,20,19,19,19,19
,19,18,18,18,18,18,18,17,17,18,17,16,16,17,16,15,15,16,17,17,17,18,19,
19,20,19,19,19,19,20,21,21,20,20,20,19,19,18,18,18,18,19,19,19,19,19,1
9,19,20,20,21,21,21,21,21,21,22,22,22,22,21,20,19,18,18,18,18,18,18,19
,19,19,19,19,18,18,17,16,15,14,13,13,13,13,13,14,14,15,15,15,16,17,17,
17,18,18,18,18,19,19,19,19,18,19,19,20,20,20,20,21,21,20,21,20,20,21,2
1,21,22,22,22,22,22,22,23,22,22,22,22,23,23,23,23,23,22,23,23,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22};
  //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-2,-2,-2,-3,-3,-2,-2,-2,-
2,-2,-2,-2,-3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-1,-1,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-2,-2,-
3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-2,-2,-2,-2,-2,-2,-1,-
2,-2,-2,-1,-1,-1,-1,-1,-
1,0,0,0,0,0,0,0,0,1,1,2,2,2,2,3,3,3,4,4,4,4,4,4,5,5,5,5,6,6,6,6,7,7,7,
7,7,7,7,7,7,7,8,8,7,7,8,8,8,8,9,9,10,10,10,10,10,10,11,11,11,11,12,12,
12,12,12,13,13,13,13,13,13,14,14,14,14,14,15,16,15,14,16,17,16,15,15,1
5,15,15,16,16,16,16,15,15,15,15,15,16,15,15,15,14,14,13,13,13,13,13,13
,13,13,13,13,13,13,13,13,14,14,14,14,14,15,15,15,16,16,16,15,15,15,15,
16,16,16,17,17,17,17,18,18,17,17,16,15,15,14,14,13,13,13,13,13,13,13,1
3,12,12,12,12,12,12,11,11,11,10,10,10,9,9,9,8,8,8,7,7,7,6,6,6,6,5,5,5,
4,4,4,4,3,3,3,2,2,2,1,1,1,0,0,0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,-2,-1,-
1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-1,-1,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2};
  //Experiment 5 - SIZE:424
  //const                                                        int8_t
acc_x_h_buffer[BUFFER_SIZE]={2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0
,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,1,1,1,1,1,2,2,1,1,2,2,2,1,1,
1,2,2,2,1,2,2,2,2,1,1,1,1,1,2,2,2,2,2,2,2,1,2,2,2,2,1,1,1,2,2,2,1,1,1,
```

```
1,1,1,1,1,1,1,2,1,1,1,2,2,1,1,1,1,2,2,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,2,2,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,2,2,2,1,1,1,2,
2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
 //const                                                          int8_t
acc_y_h_buffer[BUFFER_SIZE]={21,21,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,21,21,21,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,21,22,22,22,21,21,
22,21,21,22,22,21,22,21,21,21,21,21,21,21,20,20,20,21,21,21,21,20,20,2
0,20,20,19,20,20,20,20,20,20,19,19,19,19,19,20,19,19,19,19,19,19,19,19
,19,18,18,18,19,19,18,18,18,18,18,18,18,19,18,18,18,18,18,18,18,18,18,19,
19,19,20,20,20,20,19,20,20,19,20,20,20,20,20,20,19,19,20,19,19,19,19,1
9,19,19,19,19,20,20,20,20,20,20,20,20,20,20,20,20,21,21,21,21,21,21
,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,
22,22,22,21,21,21,21,21,21,21,21,21,21,22,22,22,21,21,21,22,22,21,21,2
1,22,21,22,22,22,22,22,22,22,21,22,21,21,22,21,22,21,21,22,22,22,22,22
,22,22,21,21,21,21,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,23,22,22,22,22,22,22,22,21,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,21,22,22,22,22,22,22,22,23
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22};
 //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-
2,-2,-1,-2,-2,-1,-1,-1,-1,-
1,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,5,5,5,5,6,6,6,6,6,7,7,7,7,7,8,
8,8,8,8,9,9,9,9,9,10,10,10,10,10,10,11,11,11,11,11,12,12,12,12,12,12,1
2,12,12,12,12,12,12,12,12,12,12,12,13,13,13,13,13,13,13,13,13,13,12,13
,13,13,12,12,12,12,12,13,13,13,13,12,12,12,12,12,12,12,12,12,12,12,12,12,
12,12,12,12,11,11,11,11,11,11,11,11,11,11,11,11,11,11,10,10,10,10,10,1
0,10,10,10,10,10,9,9,9,9,9,9,9,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,7,8,7
,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,7,6,6,6,6,6
,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,5,6,5,5,5
,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,4,5
,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,3,4,4,4,4,4,4,4,4,3
,3,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3};
 //Experiment 6 - SIZE:423
 //const                                                          int8_t
acc_x_h_buffer[BUFFER_SIZE]={1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,1,1
,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,-1,-1,0,-1,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,0,0,0,0,0,0,0,0,0,0,0,-1,-1,0,0,-1,-1,-1,0,-1,0,0,0,0,-1,0,0,0,-1,-
1,-1,0,0,0,-1,-1,-1,-1,-1,-1,0,0,-1,-1,-1,0,0,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,0,0,-1,-1,0,0,0,0,0,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,
0,0,0,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
```

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1};
 //const                                                          int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,21,21,21,21,21,21,21,21,21,21,21,21,22,21,22,22,22,22,22,22,21,21,21,
21,21,20,20,20,20,20,20,21,21,20,20,20,20,20,20,20,20,20,20,20,20,20,2
0,21,20,20,20,20,20,20,20,19,19,20,20,19,20,19,20,19,19,19,20,20,20,20
,20,20,20,20,20,20,20,20,20,20,19,19,19,19,19,19,19,19,19,19,19,19,19,
19,19,19,19,19,19,19,18,18,18,18,18,18,18,18,18,19,19,19,19,19,18,18,1
9,19,19,19,18,19,19,18,19,19,19,19,20,20,20,20,20,20,20,20,20,20,20,20
,20,20,21,21,20,21,22,21,21,21,21,21,21,21,21,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,23,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,21,22,22,22,22,22,22,22,22,22,21,21,22,22,22,22,22,22,2
2,22,22,22,22,22,22,21,21,22,22,21,21,21,22,22,22,22,22,22,22,21,21,22
,22,22,22,21,21,22,22,22,22,22,22,21,21,21,21,21,21,22,22,22,22,22,22,
22,22,22,22,21,21,22,22,22,22,22,21,21,21,22,22,22,21,21,21,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22};
 //const                                                          int8_t
acc_z_h_buffer[BUFFER_SIZE]={3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,7,7,7
,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,10,10,10,10,10,10,10,10,10,10
,10,10,10,10,10,10,10,10,10,10,11,11,11,11,11,11,11,11,11,11,11,11,11,
11,11,11,11,11,11,11,11,11,12,11,11,11,11,11,12,12,12,12,12,11,12,12,1
1,11,12,12,12,12,12,12,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13
,13,13,13,13,13,13,13,12,12,12,12,12,12,12,12,12,11,11,11,11,11,11,11,
10,10,10,10,9,9,9,9,9,8,8,8,8,8,8,8,7,7,7,7,6,6,6,6,5,5,5,5,5,4,4,4,4,
4,3,3,3,3,3,2,2,2,2,1,1,1,1,1,1,0,0,0,0,-1,-1,-1,-1,-1,-1,-2,-2,-2,-
2,-2,-2,-2,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4};
 //Experiment 7 - SIZE:403
 //const                                                          int8_t
acc_x_h_buffer[BUFFER_SIZE]={2,2,2,1,1,1,1,1,2,2,2,2,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,1,1,1,2,1,1,1,1,2,1,1,1,1,0,0
,-1,-1,0,1,0,0,0,1,-1,-3,-1,2,3,3,3,2,-1,-2,3,10,4,-1,1,7,1,-3,0,-2,-
4,-3,-1,2,1,0,-2,-4,-2,-1,0,-1,11,25,-7,-13,17,9,4,-3,-
3,0,2,1,0,0,0,1,2,3,3,2,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
 //const                                                          int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,22,22,22,22,22,22,22,22,22,22,22
```

```
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,23,23,23,23,20,17,15,14,13,12,12,15,19,19,22,26,27,31,39,43,46,51,
57,62,64,55,33,10,4,4,-1,-6,-4,2,8,6,-5,-14,-12,-2,6,6,1,-4,-
4,3,74,127,36,24,97,64,30,23,28,21,14,11,11,14,18,20,20,21,21,21,21,21
,22,22,22,22,23,23,22,22,22,22,22,22,23,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22};
  //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-3,-3,-3,-3,-3,-3,-3,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-3,-4,-4,-4,-4,-4,-4,-4,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-4,-4,-4,-4,-3,-3,-3,-4,-4,-4,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-4,-4,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-2,-2,-2,-2,-2,-1,-1,-
1,0,1,2,2,1,1,1,1,1,3,3,5,6,8,11,12,13,12,9,4,-1,-5,-
7,3,14,21,24,21,8,-6,-14,-8,9,11,3,-8,-11,-1,5,5,3,0,1,1,11,14,-6,-
35,2,-7,-2,-4,1,-1,1,0,-1,0,1,1,1,1,0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-1,-1,-1,-1,-1,-1,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-3,-
3,-2,-2,-2,-2,-2,-2,-2,-2,-3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-3,-3,-3,-3,-
3,-2,-2,-2,-2,-3,-3,-2,-2,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-3,-3,-2,-2,-2,-2,-2,-3,-3,-3,-2,-2,-2,-2,-2,-2,-3,-3,-3,-
3,-3,-2,-2,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-2,-2,-2,-2,-3,-3,-2,-2};
  //Experiment 8 - SIZE:423
  //const                                                       int8_t
acc_x_h_buffer[BUFFER_SIZE]={1,1,1,1,1,1,1,1,1,1,1,1,2,2,1,1,1,1,2,2,2
,2,2,2,2,2,2,1,1,1,1,1,1,1,1,2,2,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,2,2,2,2
,2,2,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,1,1,1,1,2,2,2,2,2,2
,1,1,2,2,2,1,1,1,1,1,1,1,1,1,1,1,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,1,1,1,2,2,1,1,1,1,1,1,1,1,1
,2,1,1,1,1,2,1,1,1,1,1,2,1,0,0,1,0,0,1,1,1,1,3,2,1,0,2,2,1,-1,-
2,2,2,1,0,-2,2,8,4,4,2,2,-3,-4,-
2,0,1,1,1,1,1,0,1,0,0,0,1,2,3,2,1,1,1,1,1,1,1,2,2,2,2,1,2,2,2,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1};
  //const                                                       int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,22,22,22,22,22,22,22,22,22,22,21
,21,21,21,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
```

```
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,21,21,21,21,21,21,20,20,20,20,19,18,17,17,16,
16,17,17,16,16,18,18,16,12,12,16,19,18,19,20,22,22,21,23,27,99,69,5,73
,44,16,36,22,16,19,19,19,19,18,19,20,19,19,19,19,19,21,20,20,21,22,21,
21,22,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,22,21,2
1,21,21,22,22,22,22,22,22,22,22,22,22,22,21,21,22,22,21,21,21,21,22,21
,21,22,21,21,21,21,22,22,22,22,22,22,22,22,22,22,22,22,22,21,22,22,21,
22,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,2
1,21,21,21,21,21,22,22,22,22,21,21,21,21,21,22,21,21,21,21,21,21,22,21
,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,
21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,2
1,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21
,21,21,21,21,21,21,21,21,21,21,21,21};
 //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-3,-3,-3,-2,-3,-3,-
3,-3,-3,-3,-3,-3,-2,-2,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-4,-3,-3,-3,-3,-4,-4,-3,-3,-3,-3,-3,-3,-4,-3,-3,-3,-3,-3,-4,-4,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-3,-3,-3,-4,-4,-4,-4,-4,-3,-4,-
4,-4,-4,-3,-3,-3,-3,-2,-3,-4,-4,-4,-4,-4,-4,-4,-6,-9,-12,-11,-9,-8,-
7,-4,0,3,7,12,33,-5,-7,17,-2,-10,1,-3,-
1,3,5,6,8,9,10,10,10,10,10,9,9,9,8,8,8,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
,7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,6,7,7,6,7,7,7,7,7,7,7,7,7
,7,7,7,6,6,6,6,6,6,6,6,6,6,6,6,7,6,7,6,6,6,6,7,7,7,7,7,7,7,7,7,7,7,7,7
,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
,7,7,7,7,7,7,7,7,7,7};
 //Experiment 9 - SIZE:424
 //const                                                        int8_t
acc_x_h_buffer[BUFFER_SIZE]={2,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,2,2,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,2,2,2,1,2,1,2,2,1,1,1,1,1
,2,1,1,1,1,1,1,1,1,2,2,1,1,1,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,2,1,1,1,2,1
,2,1,1,1,2,2,2,2,2,1,2,2,2,1,1,2,2,1,1,2,2,2,1,1,1,2,2,2,1,1,2,2,2,2,2
,1,1,1,3,3,0,-1,1,2,-1,-2,0,3,2,-1,-3,1,5,4,0,-1,0,1,5,4,-
6,3,5,3,4,3,-5,-9,0,5,1,-1,-2,-1,0,2,5,5,4,3,1,-1,-2,-2,1,6,5,-2,1,-
3,-12,-9,-11,-16,-26,-30,-20,-13,-8,-7,-5,-5,-9,-14,-11,-5,-3,-4,-5,-
6,-4,-2,-1,-1,-2,-4,-4,-4,-3,-3,-3,-3,-4,-3,-4,-4,-4,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-1,-1,-1,-2,-2,-1,-1,-1,-2,-2,-
2,-2,-3,-3,-3,-3,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3};
 //const                                                        int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
```

```
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,21,21,21,22,22,22,21,20,19,19,19,16,12,14,16,12,9,7,5,7,11,17,18,18,2
2,24,23,22,25,35,109,84,-
2,64,61,17,25,38,20,13,23,26,23,20,18,15,13,11,11,12,10,7,4,3,2,2,-7,-
35,-58,-31,3,25,15,-5,1,3,-10,-3,3,1,-9,-15,-11,-
1,2,1,1,3,4,4,4,6,6,7,7,6,5,3,3,2,2,2,2,2,2,2,2,2,3,4,4,4,4,4,4,4,4,4,
4,3,3,3,3,3,3,3,3,3,4,4,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3}; 
  //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-4,-
4,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-
3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-4,-4,-3,-2,-4,-6,-
3,0,0,-1,-4,-5,-6,-6,-8,-7,-6,-4,-1,-1,1,9,38,-3,-6,25,5,-11,-5,3,-3,-
4,7,10,10,9,9,8,8,8,9,8,7,6,7,7,5,4,8,19,24,44,40,17,26,29,28,32,55,52
,27,17,14,12,15,21,25,27,24,21,20,20,21,23,23,23,23,22,22,21,22,22,23,
23,23,23,23,23,22,22,22,22,22,22,22,22,22,22,22,23,23,23,23,22,23,2
3,22,22,22,23,23,23,23,23,23,23,22,23,23,23,22,22,22,22,22,22,23,22,22
,22,22,22,22,22,22,22,22,22,22,22,23,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,23,23,22,22,22,22,22,22,22,22,22,22,22,22,23,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,23,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22}; 
  //Experiment 10 - SIZE:423
  //const int8_t acc_x_h_buffer[BUFFER_SIZE]={-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-
22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-22,-23,-23,-22,-21,-22,-23,-
22,-22,-22,-22,-23,-23,-23,-24,-24,-23,-23,-23,-22,-22,-19,-18,-19,-
19,-19,-19,-20,-19,-21,-24,-22,-21,-22,-20,-21,-20,-22,-23,-20,-20,-
20,-18,-17,-18,-19,-19,-19,-18,-18,-21,-26,-23,-18,-18,-18,-17,-18,-
18,-19,-19,-19,-21,-20,-19,-19,-19,-19,-19,-19,-19,-19,-19,-20,-20,-
20,-20,-20,-20,-21,-21,-21,-21,-21,-21,-21,-21,-21,-21,-21,-21,-21,-
21,-21,-21,-21,-22,-23,-23,-23,-23,-23,-24,-23,-24,-24,-24,-26,-26,-
26,-26,-26,-26,-24,-24,-24,-21,-15,-9,1,4,5,10,15,35,45,10,15,-2,-41,-
2,-7,-8,8,2,-1,4,9,5,-2,-4,-1,-1,-2,-2,-3,-4,-5,-4,-4,-3,-3,-2,-2,-2,-
2,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2}; 
  //const int8_t acc_y_h_buffer[BUFFER_SIZE]={-1,-1,-1,-1,-1,-2,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
```

```
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-1,-1,-1,-
1,-2,-1,0,-1,-1,-1,-1,-1,-2,-2,-2,-3,-2,-1,-1,-1,-1,-1,-2,-2,-1,-1,-
1,-1,-2,0,-1,-2,-1,-2,-2,-2,-2,-3,-3,-4,-3,-4,-3,-1,-1,0,1,3,0,-
2,0,0,0,-1,-1,-1,-1,-1,-1,-1,-2,-2,-2,-1,-2,-1,-2,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-2,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-2,-2,-1,-1,-1,-2,-
2,-1,-1,-1,-1,-2,-2,-2,-2,-2,-3,-3,-3,-3,-2,-2,-2,-1,-2,-
3,2,8,7,11,17,12,-16,-44,-20,7,15,20,20,12,11,15,4,-11,-18,-7,8,1,-
1,4,6,1,-3,-
2,1,2,3,6,8,7,4,2,2,2,1,1,1,1,1,1,1,1,2,2,2,3,3,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2};
  //const int8_t acc_z_h_buffer[BUFFER_SIZE]={-4,-4,-4,-4,-5,-4,-4,-5,-
5,-4,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-
5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-
5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-
5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-5,-4,-
4,-4,-4,-4,-4,-4,-4,-4,-5,-5,-5,-5,-4,-4,-4,-3,-4,-4,-4,-5,-5,-4,-5,-
5,-6,-6,-5,-5,-6,-7,-7,-7,-7,-6,-5,-5,-5,-5,-3,-3,-4,-4,-4,-4,-4,-4,-
5,-6,-5,-7,-7,-7,-7,-7,-7,-8,-9,-10,-11,-11,-12,-12,-12,-12,-12,-11,-
9,-9,-11,-17,-13,-12,-12,-13,-13,-13,-12,-12,-11,-11,-13,-11,-11,-11,-
10,-10,-11,-10,-10,-10,-10,-10,-10,-10,-10,-10,-10,-11,-11,-10,-10,-
10,-9,-10,-9,-9,-9,-9,-8,-8,-8,-7,-7,-6,-6,-7,-6,-6,-6,-6,-6,-5,-5,-
4,-4,-5,-4,-4,-4,-4,-4,-3,-3,-5,-7,-8,-9,-4,-6,-8,-
8,1,15,14,35,38,22,33,93,94,29,16,11,9,3,11,28,29,33,30,25,20,17,18,20
,23,25,26,25,24,21,21,22,22,23,23,23,23,22,22,22,22,23,23,23,23,23,23,
23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,2
3,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23
,23,23,23,23,23,23,22,23,22,22,23,23,23,23,23,23,23,23,23,23,23,23,23,
23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,2
3,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23
,23,23,23,23,23,23,22,23,23,23,23,23,23,23,23,23,23,23};
  //Experiment 11 - SIZE:528
  //const  int8_t  acc_x_h_buffer[BUFFER_SIZE]={0,0,0,0,0,1,1,0,0,0,-1,-
1,-2,-2,-1,-1,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,0,0,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,-1,-1,-2,-2,-1,-1,-1,-1,-1,-1,-1,-1,-2,-2,-2,-1,-1,-2,-2,-1,-1,-1,-
1,-1,-2,-1,-2,-2,-2,-2,-2,-2,-1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1,-2,0,2,-
1,-2,-1,3,4,3,4,3,2,0,-2,-5,-4,-2,-2,-1,4,7,2,1,1,-2,-15,-5,-20,-
27,5,13,3,-1,4,1,-12,-18,-12,-14,-20,-25,-14,6,22,22,4,-12,-19,-12,-
5,1,6,5,0,-3,-2,-1,-1,-2,-1,-1,0,0,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-
2,-2,-2,-2,-2,-2,-2,-3,-2,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
```

```
2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-3,-3,-2,-2,-3,-3,-4,-
4,-4,-4,-4,-4,-4,-4,-5,-4,-3,-3,-2,-1,-1,-2,-2,-2,-2,-2,-2,-2,-2,-2,-
2,-2,-2,-2,-2,-2,-2,-1,0,0,1,1,1,1,1,3};
 //const                                                          int8_t
acc_y_h_buffer[BUFFER_SIZE]={22,22,22,23,23,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22
,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,
22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,22,22,2
2,22,22,22,22,22,22,22,22,22,22,22,21,21,20,20,20,20,20,21,21,21,21,20
,20,20,19,19,19,19,19,19,19,19,20,19,19,20,20,19,19,19,18,18,18,18,18,
18,17,17,17,17,18,19,19,20,19,18,18,19,19,19,19,18,17,16,14,12,10,8,8,
9,12,16,18,22,22,21,20,67,63,4,69,78,18,48,34,9,6,0,-5,-10,-16,-15,-
15,-36,-50,-32,-10,4,0,-13,-19,-17,-12,-10,-10,-12,-13,-12,-10,-8,-9,-
9,-8,-7,-6,-6,-7,-8,-8,-8,-8,-7,-7,-7,-7,-7,-7,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-
8,-8,-8,-8,-8,-8,-8,-8,-8,-8,-9,-9,-9,-9,-9,-9,-9,-9,-8,-8,-8,-9,-9,-
9,-8,-7,-6,-5,-5,-5,-5,-4,-4,-4,-5,-5,-5,-6,-5,-5,-5,-6,-6,-6,-6,-5,-
5,-5,-5,-4,-4,-4,-4,-4,-4};
 //const                                                          int8_t
acc_z_h_buffer[BUFFER_SIZE]={3,3,3,3,3,3,3,3,4,4,3,3,3,3,3,3,3,3,3,3,3
,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4
,4,5,5,5,5,5,5,6,6,6,6,7,7,8,8,9,9,9,9,10,10,10,11,11,12,11,12,12,12,1
2,12,12,12,12,12,12,13,13,13,13,13,13,13,13,12,12,12,12,12,11,12,12,11
,11,11,11,11,11,11,10,11,11,10,9,9,9,8,7,6,5,3,1,-1,-3,-3,-4,-
5,0,2,6,16,21,23,50,24,-8,25,10,-
11,6,9,6,10,7,8,10,11,12,19,24,23,25,23,21,17,15,17,20,22,22,21,20,20,
21,22,23,22,21,21,21,21,22,22,22,22,22,22,22,21,21,21,21,22,22,22,22,2
1,21,21,21,21,21,21,21,22,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21
,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,22,21,21,
21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,2
1,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21
,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,
21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,2
1,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,22,21,21,21,21,21
,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,
21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,21,2
1,22,22,22,21,21,21,21,21,22,22,22,22,21,21,22,22,22,22,22,21,21,21,21
,21,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,23,
23,23,23,23,23,23,22,22,22};

 uint16_t x_index = 0;
 uint16_t y_index = 0;
 uint16_t z_index = 0;

 volatile bool acc_data_rdy = false;
```

```
  bool acc_x_gt_th = false;       //flagA_1
  bool acc_y_gt_th = false;       //flagA_2
  bool acc_z_gt_th = false;       //flagA_3
  bool en_x_gt_th = false;        //flagE_1
  bool en_y_gt_th = false;        //flagE_2
  bool en_z_gt_th = false;        //flagE_3
  bool impact_detected = false;   //h

  uint8_t system_settled_counter = INIT_WAIT_COUNT;

  //TIME VARIABLES
  volatile bool overflow_parity = 0;
  timepoint_t acc_x_gt_th_timepoint = {0,0};
  timepoint_t acc_y_gt_th_timepoint = {0,0};
  timepoint_t acc_z_gt_th_timepoint = {0,0};
  timepoint_t en_x_gt_th_timepoint = {0,0};
  timepoint_t en_y_gt_th_timepoint = {0,0};
  timepoint_t en_z_gt_th_timepoint = {0,0};

  uint8_t tau_samples = (uint8_t)(TAU * ACC_SAMPLERATE);

  #ifndef
                                                                  _STD
LIB_
  int abs(int a){if(a < 0)return -a; return a;}
  #endif

  void delay_seconds(uint8_t seconds);
  //void set_baudrate(uint8_t);//ONLY NEEDED IF NOT USING USART LIBRARY
  void empty_buffer(int8_t buffer[]);
  void indexes_update();
  void SPI_data_capture();
  void validation_data_capture();
  void upper_acc_x();
  void upper_acc_y();
  void upper_acc_z();
  void upper_energy_x();
  void upper_energy_y();
  void upper_energy_z();

  #if !P_INT_E
  void interrupt general_isr(void){
  #else
  void interrupt high_priority accel_rdy_isr(void){
  #endif
      //HIGH_PRIORITY
      if (INTCONbits.INT0F){
          INTCONbits.INT0F = 0;
          acc_data_rdy = true;
      }
  #if P_INT_E
  }
  void interrupt low_priority peripheral_isr(void){
  #endif
      //LOW PRIORITY
      //TIMER1 INTERRUPT HANDLE
      if (PIR1bits.TMR1IF){
```

```
            overflow_parity ^= 1;
            PIR1bits.TMR1IF = 0;
        }
        //USART INTERRUPT HANDLE
    //    else if(PIR1bits.RCIF){
    //        receive_char = getcUSART();
    //        putsUSART("CHAR_RECIBIDO:\n");
    //        sprintf(send_str, "%c\n", receive_char);
    //        putsUSART(send_str);
    //        putcUSART('\n');
    //        if(receive_char=='\n')
    //            str_complete = true;
    //        else{
    //            receive_str[str_index] = receive_char;
    //            str_index++;
    //        }
    //        PIR1bits.RCIF = 0;
    //    }
    }
    void main(void) {

    #ifdef MATLAB_PRINT
        uint16_t t_counter = 1;
    #endif
    #ifdef VALIDATION
        bool validation_test = true;
    #else
        bool validation_test = false;
    #endif
        //System Oscillator configuration (internal)
        OSCCONbits.IRCF = 0b110;
        OSCCONbits.SCS = 0b10;
        OSCTUNEbits.TUN = 0;

        //PORTA ADC disable and I/O ports clear
        ANSEL0 = 0x00;
        PORTA = 0x00; LATA = 0x00;
        PORTB = 0x00; LATB = 0x00;
        PORTC = 0x00; LATC = 0x00;

        //IO_PINS direction setting
        TRISCbits.RC4 = OUTPUT;
        TRISAbits.RA0   =   OUTPUT;   //WAKE_SW//Alternative:   TRISA   &=
~(_TRISA_RA0_MASK | _TRISA_RA1_MASK);
        TRISAbits.RA1 = OUTPUT; //PIO7
        //USART_PINS/LED_TEST_PINS  direction  setting  (INPUT  FOR  USART,
OUTPUT FOR LED_TEST)
        TRISCbits.RC6    =    INPUT;       //TX//Alternative:    TRISC    &=
~(_TRISC_RC6_MASK | _TRISC_RC7_MASK);
        TRISCbits.RC7 = INPUT;  //RX

        //SPI configuration
        OpenSWSPI();

        //TIMER1 configuration
        TMR1H = 0; TMR1L = 33; //THIS COULD BE 0 BUT I HAVE TO CONFIRM
```

```
      T1CON    =    _T1CON_T1OSCEN_MASK    |    _T1CON_T1SYNC_MASK    |
_T1CON_TMR1CS_MASK | _T1CON_TMR1ON_MASK;

      //EEPROM configuration
      IPR2bits.EEIP = P_EEPROM;

      //Interrupts configuration
      RCONbits.IPEN = P_INT_E;       //Interrupt priority enable/disable
      INTCONbits.GIE = G_INT_E;       //Global interrupts enable/disable
      INTCONbits.PEIE   =   PER_INT_E;          //Peripheral   interrupts
enable/disable
      INTCONbits.INT0E = 1;           //INT0 (C3) interrupt enable
      PIE1bits.TMR1IE = 1;            //TIMER1 interrupt enable
      IPR1bits.RCIP  =  PRIORITY_RX;      //USART  RX  interrupt  priority
high/low
      IPR1bits.TXIP  =  PRIORITY_TX;      //USART  TX  interrupt  priority
high/low
      //PIE1bits.RCIE  =  INTENABLE_RX;     //USART  receiver  interrupt
enable/disable
      //PIE1bits.TXIE  =  INTENABLE_TX;    //USART transmitter interrupt
enable/disable
      PIE2bits.EEIE = E_EEPROM;       //EEPROM interrupt enable/disable

      //**USART configuration**//
      //BLE module configurations
      //WAKE_HW = 0;    //for exiting (or not) Dormant mode (wake-up the
BLE module)
      //WAKE_SW = 0;    //for entering (or not) CMD mode
      //Send "SF,1" for Factory Reset with Device Name and Info, Script
and Private Services staying the same.

      //ONLY NEEDED IF NOT USING USART LIBRARY
  //    TXSTAbits.TX9 = 0;  //8 bit transmission
  //    TXSTAbits.SYNC = 0; //Asynchronous mode
  //    set_baudrate(BAUD_38400);   //baudrate setting
  //    //USART ENABLE
  //    RCSTAbits.SPEN = 1; //USART ON(1)/OFF(0)
  //    RCSTAbits.CREN = 1; //RX ON(1)/OFF(0)
  //    TXSTAbits.TXEN = 1; //TX ON(1)/OFF(0)

      unsigned char config = 0, spbrg=0, baudconfig=0;
      CloseUSART();
      config    =    (INTENABLE_TX?USART_TX_INT_ON:USART_TX_INT_OFF)    |
(INTENABLE_RX?USART_RX_INT_ON:USART_RX_INT_OFF)  |  USART_ASYNCH_MODE  |
USART_EIGHT_BIT | USART_CONT_RX | USART_BRGH_HIGH;
      spbrg = 8;
      OpenUSART(config, spbrg);
      baudconfig = BAUD_16_BIT_RATE | BAUD_AUTO_OFF;
      baudUSART(baudconfig);           //baudrate     configured     at
38400(spbrg=25)/115200(spbrg=8)

      //LIS3LV02 configuration
      lis3_set_ctrl_reg1(LIS3LV02_POWERON,LIS3LV02_DATARATE_40HZ);

lis3_set_ctrl_reg2(LIS3LV02_SCALE_6G,LIS3LV02_CONTINUOUS,LIS3LV02_INT_
DISABLED,LIS3LV02_DRDY_ENABLED,LIS3LV02_SPI_4W,LIS3LV02_DATA_16L);
```

```
      delay_seconds(1);
      while(!lis3_begin()){
          while(BusyUSART());
          putsUSART("Error initializing LIS3LV02\n");
          CLRWDT();
      }
      while(BusyUSART());
      putsUSART("LIS3LV02 started successfully\n");
      delay_seconds(1);

      while(BusyUSART());
      putsUSART("Reading 31 acc samples...\n");
      delay_seconds(1);

      //31 SAMPLES READING LOOP//
      do{
          if(acc_data_rdy || validation_test){
  #ifdef VALIDATION
              validation_data_capture();
  #else
              acc_data_rdy = false;
              SPI_data_capture();
  #endif
              indexes_update();
              system_settled_counter--;

  #ifdef DEBUG_PRINT
              sprintf(send_str,"Samples    Left    Counter:    %d\n",
system_settled_counter);
              while(BusyUSART());
              putsUSART(send_str);
  #endif

  #ifdef MATLAB_PRINT
              sprintf(send_str,"%d ", t_counter);
              while(BusyUSART());
              putsUSART(send_str);
              t_counter++;

              sprintf(send_str,"%d %d %d ", acc_x, acc_y, acc_z);
              while(BusyUSART());
              putsUSART(send_str);
              sprintf(send_str,"%d %d %d ", acc_x_f, acc_y_f, acc_z_f);
              while(BusyUSART());
              putsUSART(send_str);
              sprintf(send_str,"%d  %d  %d  ",  energy_x,  energy_y,
energy_z);
              while(BusyUSART());
              putsUSART(send_str);
              sprintf(send_str,"%d %d %d ", acc_x_gt_th, acc_y_gt_th,
acc_z_gt_th);
              while(BusyUSART());
              putsUSART(send_str);
              sprintf(send_str,"%d  %d  %d  ",  en_x_gt_th,  en_y_gt_th,
en_z_gt_th);
              while(BusyUSART());
              putsUSART(send_str);
```

```
                sprintf(send_str,"%d\n", impact_detected);
                while(BusyUSART());
                putsUSART(send_str);

  #elif defined DEBUG_PRINT || defined PLOTTER_PRINT
                sprintf(send_str,"AccX:%d   AccY:%d   AccZ:%d   ", acc_x,
acc_y, acc_z);
                while(BusyUSART());
                putsUSART(send_str);
                sprintf(send_str,"AccXf:%d  AccYf:%d  AccZf:%d  ", acc_x_f,
acc_y_f, acc_z_f);
                while(BusyUSART());
                putsUSART(send_str);
                sprintf(send_str,"EnX:%d  EnY:%d  EnZ:%d  \n", energy_x,
energy_y, energy_z);
                while(BusyUSART());
                putsUSART(send_str);
  #endif
                CLRWDT();
            }
        }while(system_settled_counter);

        //MAIN LOOP//
        while(true){
            // LED TEST CODE // (blink every 1 s)
  //          LED = 0;
  //          __delay_ms(167);__delay_ms(167);__delay_ms(166);
  //          LED = 1;
  //          __delay_ms(167);__delay_ms(167);__delay_ms(166);
            ////////////////////

            // IMPACT DETECTION ALGORITHM CODE //
            if(acc_data_rdy  ||  validation_test){     //If  we  have  new
acceleration samples
                acc_data_rdy = false;
  #ifdef VALIDATION
                validation_data_capture();
  #else
                SPI_data_capture();
  #endif
                //CHECK FOR ACC THRESHOLD CROSSING
                upper_acc_x();
                upper_acc_y();
                upper_acc_z();

                //ENERGY  EQUATIONS  [Calculating  Acc_f(n-tau)  at  the
equation as (Acc[n-tau] - Acc[n-tau-1]) / 2]
                acc_x_f_tau    =    (acc_x_h_buffer[X_INDEX_MINUS_TAU]    -
acc_x_h_buffer[X_INDEX_MINUS_TAU_MINUS_1])/2;
                acc_y_f_tau    =    (acc_y_h_buffer[Y_INDEX_MINUS_TAU]    -
acc_y_h_buffer[Y_INDEX_MINUS_TAU_MINUS_1])/2;
                acc_z_f_tau    =    (acc_z_h_buffer[Z_INDEX_MINUS_TAU]    -
acc_z_h_buffer[Z_INDEX_MINUS_TAU_MINUS_1])/2;

                energy_x = energy_x + (acc_x_f * acc_x_f) - (acc_x_f_tau *
acc_x_f_tau);
```

```
            energy_y = energy_y + (acc_y_f * acc_y_f) - (acc_y_f_tau *
acc_y_f_tau);
            energy_z = energy_z + (acc_z_f * acc_z_f) - (acc_z_f_tau *
acc_z_f_tau);

            //CHECK FOR ENERGY THRESHOLD CROSSING
            upper_energy_x();
            upper_energy_y();
            upper_energy_z();

            //CHECK FOR IMPACT DETECTED
            impact_detected   =   (acc_x_gt_th   &   en_x_gt_th)   |
(acc_y_gt_th & en_y_gt_th) | (acc_z_gt_th & en_z_gt_th);
            /*-------------------------*/
            if(impact_detected){
                //FUTURE DEVELOPMENT
 #ifdef DEBUG_PRINT
                putsUSART("IMPACTO DETECTADO\n");
 #endif
            }

            // MATLAB/DEBUG/PLOTTER PRINTS //
 #ifdef MATLAB_PRINT
            sprintf(send_str,"%d ", t_counter);
            while(BusyUSART());
            putsUSART(send_str);
            t_counter++;

            sprintf(send_str,"%d %d %d ", acc_x, acc_y, acc_z);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"%d %d %d ", acc_x_f, acc_y_f, acc_z_f);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"%d  %d  %d  ",  energy_x,  energy_y,
energy_z);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"%d %d %d ", acc_x_gt_th, acc_y_gt_th,
acc_z_gt_th);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"%d %d %d ", en_x_gt_th, en_y_gt_th,
en_z_gt_th);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"%d\n", impact_detected);
            while(BusyUSART());
            putsUSART(send_str);

 #elif defined DEBUG_PRINT || defined PLOTTER_PRINT
            //MAXIMUM PRINTING GROUPS = 3 (4 if one is Impact Detection
Flag)
 //            sprintf(send_str,"ArX:%d ArY:%d ArZ:%d ", acc_x, acc_y,
acc_z);
 //            while(BusyUSART());
 //            putsUSART(send_str);
```

```
            sprintf(send_str,"AfX:%d  AfY:%d  AfZ:%d  ",  acc_x_f  ,
acc_y_f, acc_z_f);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"EX:%d EY:%d EZ:%d ", energy_x, energy_y,
energy_z);
            while(BusyUSART());
            putsUSART(send_str);
            sprintf(send_str,"FAX:%d FAY:%d FAZ:%d ", acc_x_gt_th*20,
acc_y_gt_th*30, acc_z_gt_th*40);
            while(BusyUSART());
            putsUSART(send_str);
  //          sprintf(send_str,"FEX:%d FEY:%d FEZ:%d\n", en_x_gt_th*50,
en_y_gt_th*60, en_z_gt_th*70);
  //            while(BusyUSART());
  //            putsUSART(send_str);
            sprintf(send_str,"Imp:%d\n", impact_detected*100);
            while(BusyUSART());
            putsUSART(send_str);
  #endif
            indexes_update();
            CLRWDT();
        }
        else{
            //FUTURE DEVELOPMENT
  //            __delay_ms(10);
        }

  //        memset(receive_str, '\0', sizeof(receive_str));
  //        str_complete = false;
    }
    return;
  }

  void delay_seconds(uint8_t seconds){
    CLRWDT();
    for (int i = 0; i < seconds; i++) {
        __delay_ms(167);__delay_ms(167);__delay_ms(166);
        CLRWDT();
        __delay_ms(167);__delay_ms(167);__delay_ms(166);
        CLRWDT();
    }
  }

  void empty_buffer(int8_t buffer[]){
    for (int i = 0; i < BUFFER_SIZE; i++) {
        buffer[i] = 0;
    }
  }

  void indexes_update(){
    //Increment index and reset if overflowed the buffer size
    if (++x_index > BUFFER_SIZE - 1) x_index = 0;
    if (++y_index > BUFFER_SIZE - 1) y_index = 0;
    if (++z_index > BUFFER_SIZE - 1) z_index = 0;
  }
```

```
#ifndef VALIDATION
void SPI_data_capture(){
    //Read high byte accelerations
    acc_x = lis3_get_acc_x_h();
    acc_y = lis3_get_acc_y_h();
    acc_z = lis3_get_acc_z_h();
    //Store raw accelerations in the circular buffers
    acc_x_h_buffer[x_index] = acc_x;
    acc_y_h_buffer[y_index] = acc_y;
    acc_z_h_buffer[z_index] = acc_z;
    //Apply filter to accelerations
    acc_x_f = (acc_x - prev_acc_x) / 2;
    prev_acc_x = acc_x;
    acc_y_f = (acc_y - prev_acc_y) / 2;
    prev_acc_y = acc_y;
    acc_z_f = (acc_z - prev_acc_z) / 2;
    prev_acc_z = acc_z;
}
#endif

void validation_data_capture(){
    //Read raw accelerations from the validation buffers
    acc_x = acc_x_h_buffer[x_index];
    acc_y = acc_y_h_buffer[y_index];
    acc_z = acc_z_h_buffer[z_index];
    //Apply filter to accelerations
    acc_x_f = (acc_x - prev_acc_x) / 2;
    prev_acc_x = acc_x;
    acc_y_f = (acc_y - prev_acc_y) / 2;
    prev_acc_y = acc_y;
    acc_z_f = (acc_z - prev_acc_z) / 2;
    prev_acc_z = acc_z;
}

void upper_acc_x(){
    if(abs(acc_x_f) >= acc_threshold){
        acc_x_gt_th = true;
        acc_x_gt_th_timepoint.tmr_data = TMR1H;
        acc_x_gt_th_timepoint.ov_parity = overflow_parity;
#ifdef DEBUG_PRINT
        putsUSART("ACC THRESHOLD EXCEEDED ON X AXIS\n");
#endif
    }
    else  if((acc_x_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
            (TMR1H      -      acc_x_gt_th_timepoint.tmr_data      >
time_threshold_lvl || \
            TMR1H    -    acc_x_gt_th_timepoint.tmr_data    <    0))
\
            || (acc_x_gt_th_timepoint.ov_parity != overflow_parity &&
\
            TMR1H   +   0xFF   -   acc_x_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
    {
        acc_x_gt_th = false;
    }
}
```

```
  void upper_acc_y(){
      if(abs(acc_y_f) >= acc_threshold){
          acc_y_gt_th = true;
          acc_y_gt_th_timepoint.tmr_data = TMR1H;
          acc_y_gt_th_timepoint.ov_parity = overflow_parity;
  #ifdef DEBUG_PRINT
          putsUSART("ACC THRESHOLD EXCEEDED ON Y AXIS\n");
  #endif
      }
      else  if((acc_y_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
              (TMR1H      -      acc_y_gt_th_timepoint.tmr_data      >
time_threshold_lvl || \
              TMR1H    -    acc_y_gt_th_timepoint.tmr_data    <    0))
\
              || (acc_y_gt_th_timepoint.ov_parity != overflow_parity &&
\
              TMR1H   +   0xFF   -   acc_y_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
      {
          acc_y_gt_th = false;
      }
  }

  void upper_acc_z(){
      if(abs(acc_z_f) >= acc_threshold){
          acc_z_gt_th = true;
          acc_z_gt_th_timepoint.tmr_data = TMR1H;
          acc_z_gt_th_timepoint.ov_parity = overflow_parity;
  #ifdef DEBUG_PRINT
          putsUSART("ACC THRESHOLD EXCEEDED ON Z AXIS\n");
  #endif
      }
      else  if((acc_z_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
              (TMR1H      -      acc_z_gt_th_timepoint.tmr_data      >
time_threshold_lvl || \
              TMR1H    -    acc_z_gt_th_timepoint.tmr_data    <    0))
\
              || (acc_z_gt_th_timepoint.ov_parity != overflow_parity &&
\
              TMR1H   +   0xFF   -   acc_z_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
      {
          acc_z_gt_th = false;
      }
  }

  void upper_energy_x(){
      if(abs(energy_x) >= energy_threshold){
          en_x_gt_th = true;
          en_x_gt_th_timepoint.tmr_data = TMR1H;
          en_x_gt_th_timepoint.ov_parity = overflow_parity;
  #ifdef DEBUG_PRINT
          putsUSART("ENERGY THRESHOLD EXCEEDED ON X AXIS\n");
  #endif
```

```
      }
      else  if((en_x_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
              (TMR1H       -       en_x_gt_th_timepoint.tmr_data       >
time_threshold_lvl || \
              TMR1H     -     en_x_gt_th_timepoint.tmr_data     <     0))
\
              || (en_x_gt_th_timepoint.ov_parity != overflow_parity &&
\
              TMR1H   +   0xFF   -   en_x_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
      {
          en_x_gt_th = false;
      }
  }

  void upper_energy_y(){
      if(abs(energy_y) >= energy_threshold){
          en_y_gt_th = true;
          en_y_gt_th_timepoint.tmr_data = TMR1H;
          en_y_gt_th_timepoint.ov_parity = overflow_parity;
  #ifdef DEBUG_PRINT
          putsUSART("ENERGY THRESHOLD EXCEEDED ON Y AXIS\n");
  #endif
      }
      else  if((en_y_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
              (TMR1H       -       en_y_gt_th_timepoint.tmr_data       >
time_threshold_lvl || \
              TMR1H     -     en_y_gt_th_timepoint.tmr_data     <     0))
\
              || (en_y_gt_th_timepoint.ov_parity != overflow_parity &&
\
              TMR1H   +   0xFF   -   en_y_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
      {
          en_y_gt_th = false;
      }
  }

  void upper_energy_z(){
      if(abs(energy_z) >= energy_threshold){
          en_z_gt_th = true;
          en_z_gt_th_timepoint.tmr_data = TMR1H;
          en_z_gt_th_timepoint.ov_parity = overflow_parity;
  #ifdef DEBUG_PRINT
          putsUSART("ENERGY THRESHOLD EXCEEDED ON Z AXIS\n");
  #endif
      }
      else  if((en_z_gt_th_timepoint.ov_parity  ==  overflow_parity  &&
\
              (TMR1H       -       en_z_gt_th_timepoint.tmr_data       >
time_threshold_lvl || \
              TMR1H     -     en_z_gt_th_timepoint.tmr_data     <     0))
\
              || (en_z_gt_th_timepoint.ov_parity != overflow_parity &&
\
```

```
             TMR1H   +   0xFF   -   en_z_gt_th_timepoint.tmr_data   >
time_threshold_lvl))
      {
          en_z_gt_th = false;
      }
  }

  //ONLY NEEDED IF NOT USING USART LIBRARY
  //void set_baudrate(uint8_t baudrate){
  //    TXSTAbits.BRGH = 1; //High baud rate
  //    switch(baudrate){
  //        case BAUD_19200:
  //            BAUDCONbits.BRG16 = 0;  //8-bit mode
  //            SPBRGH = 0x00;
  //            SPBRG = 12;
  //            break;
  //        case BAUD_38400:
  //            BAUDCONbits.BRG16 = 1;  //16-bit mode
  //            SPBRGH = 0x00;
  //            SPBRG = 25;
  //            break;
  //        case BAUD_57600:
  //            BAUDCONbits.BRG16 = 1;  //16-bit mode
  //            SPBRGH = 0x00;
  //            SPBRG = 16;
  //            break;
  //        case BAUD_115200:
  //            BAUDCONbits.BRG16 = 1;  //16-bit mode
  //            SPBRGH = 0x00;
  //            SPBRG = 8;
  //            break;
  //        default:    //BAUD_38400
  //            BAUDCONbits.BRG16 = 1;  //16-bit mode
  //            SPBRGH = 0x00;
  //            SPBRG = 25;
  //    }
  //}
```

lis3lv02.h

```
  /*
   * File:   lis3lv02.h
   * Author: Santiago Dominguez Vidal
   * Comments: simple lis3lv02 (SPI) library for basic usage
   * Revision history: 0.7 - 11/08/2021
   */

  #include <stdint.h>
  #include <stdbool.h>
  #include <sw_spi.h>

  #ifndef LIS3LV02_H
  #define LIS3LV02_H

  #define WHO_AM_I        0x0F
  #define OFFSET_X        0x16
```

```
#define OFFSET_Y        0x17
#define OFFSET_Z        0x18
#define GAIN_X          0x19
#define GAIN_Y          0x1A
#define GAIN_Z          0x1B
#define CTRL_REG1       0x20
#define CTRL_REG2       0x21
#define CTRL_REG3       0x22
#define HP_FILTER_RESET 0x23
#define STATUS_REG      0x27
#define OUTX_L          0x28
#define OUTX_H          0x29
#define OUTY_L          0x2A
#define OUTY_H          0x2B
#define OUTZ_L          0x2C
#define OUTZ_H          0x2D
#define FF_WU_CFG       0x30
#define FF_WU_SRC       0x31
#define FF_WU_ACK       0x32
#define FF_WU_THS_L     0x34
#define FF_WU_THS_H     0x35
#define FF_WU_DURATION  0x36
#define DD_CFG          0x38
#define DD_SRC          0x39
#define DD_ACK          0x3A
#define DD_THSI_L       0x3C
#define DD_THSI_H       0x3D
#define DD_THSE_L       0x3E
#define DD_THSE_H       0x3F

#define LIS3_ID  0x3A
#define BIT7 (0b1<<7)
#define BIT6 (0b1<<6)
#define BIT5 (0b1<<5)
#define BIT4 (0b1<<4)
#define BIT3 (0b1<<3)
#define BIT2 (0b1<<2)
#define BIT1 (0b1<<1)
#define BIT0 (0b1<<0)

/**
 * @brief Power setting
 */
typedef enum
{
    LIS3LV02_POWERDOWN,
    LIS3LV02_POWERON
} lis3lv02_power_t;

/**
 * @brief Output data rate setting
 */
typedef enum
{
    LIS3LV02_DATARATE_40HZ,
    LIS3LV02_DATARATE_160HZ,
    LIS3LV02_DATARATE_640HZ,
```

```
    LIS3LV02_DATARATE_2560HZ
} lis3lv02_datarate_t;

/**
 * @brief Accelerometer scale setting
 */
typedef enum
{
    LIS3LV02_SCALE_2G,
    LIS3LV02_SCALE_6G
} lis3lv02_scale_t;

/**
 * @brief Update data mode setting
 */
typedef enum
{
    LIS3LV02_CONTINUOUS,
    LIS3LV02_BLOCKED
}lis3lv02_block_update_t;

/**
 * @brief Interrupt enable/disable
 */
typedef enum
{
    LIS3LV02_INT_DISABLED,
    LIS3LV02_INT_ENABLED
}lis3lv02_int_enable_t;

/**
 * @brief Data-ready generation enable/disable
 */
typedef enum
{
    LIS3LV02_DRDY_DISABLED,
    LIS3LV02_DRDY_ENABLED
}lis3lv02_drdy_enable_t;

/**
 * @brief SPI mode selection
 */
typedef enum
{
    LIS3LV02_SPI_4W,
    LIS3LV02_SPI_3W
}lis3lv02_spimode_t;

/**
 * @brief Data alignment selection
 */
typedef enum
{
    LIS3LV02_DATA_12R,
    LIS3LV02_DATA_16L
}lis3lv02_alignment_t;
```

```
/**
 * @brief Fiter clock source selection
 */
typedef enum
{
    LIS3LV02_INTERNAL_OSC,
    LIS3LV02_EXTERNAL_PAD
}lis3lv02_clock_t;

/**
 * @brief Filter enable/disable
 */
typedef enum
{
    LIS3LV02_FILTER_BYPASSED,
    LIS3LV02_FILTER_ENABLED
}lis3lv02_filter_enable_t;

/**
 * @brief Filter frequency setting
 * f_cutoff = (0.318/HPC)*(ODRx/2)
 */
typedef enum
{
    LIS3LV02_FILTER_HPC_512,
    LIS3LV02_FILTER_HPC_1024,
    LIS3LV02_FILTER_HPC_2048,
    LIS3LV02_FILTER_HPC_4096
} lis3lv02_filter_freq_t;

uint8_t lis3_getID();
bool lis3_begin();

void         lis3_set_power_datarate(lis3lv02_power_t         power,
lis3lv02_datarate_t datarate);
#define lis3_set_ctrl_reg1 lis3_set_power_datarate

void
lis3_set_scale_update_int_drdy_spimode_alignment(lis3lv02_scale_t
scale,  lis3lv02_block_update_t  update,  lis3lv02_int_enable_t  inte,
lis3lv02_drdy_enable_t      drdy,      lis3lv02_spimode_t      spimode,
lis3lv02_alignment_t alignment);
#define                                          lis3_set_ctrl_reg2
lis3_set_scale_update_int_drdy_spimode_alignment

void         lis3_set_filter_options(lis3lv02_clock_t         clock,
lis3lv02_filter_enable_t  dirdet,  lis3lv02_filter_enable_t  freefall,
lis3lv02_filter_enable_t data, lis3lv02_filter_freq_t freq);
#define lis3_set_ctrl_reg3 lis3_set_filter_options

void lis3_reset_filter();

uint8_t lis3_read_status_reg();

int8_t lis3_get_acc_x_h();
uint8_t lis3_get_acc_x_l();
```

```
 int8_t lis3_get_acc_y_h();
 uint8_t lis3_get_acc_y_l();

 int8_t lis3_get_acc_z_h();
 uint8_t lis3_get_acc_z_l();

 //16 bit alignment functions
 int16_t lis3_get_acc_x();
 int16_t lis3_get_acc_y();
 int16_t lis3_get_acc_z();

 int16_t lis3_get_acc_x16();
 int16_t lis3_get_acc_y16();
 int16_t lis3_get_acc_z16();
 #endif
```

lis3lv02.c

```
 /*
  * File:   lis3lv02.c
  * Author: Santiago Dominguez Vidal
  *
  * Created on 19th May 2021
  */

 #include "lis3lv02.h"

 /**
  * @brief Function to get lis3lv02 device ID
  * @return accelerometer ID
  */
 uint8_t lis3_getID(){
     uint8_t id;
     ClearCSSWSPI();
     WriteSWSPI(WHO_AM_I | 0x80);
     id = WriteSWSPI(0x00);
     SetCSSWSPI();
     return id;
 }
 /**
  * @brief Function to initialize lis3lv02
  * @return true if the return of lis3_getID inside is the lis3lv02
device ID
  */
 bool lis3_begin(){
     return (lis3_getID() == LIS3_ID);
 }
 /**
  * @brief Function to set power mode, data rate and enable all axis
  * @param power
  * @param datarate
  */
 void lis3_set_power_datarate(lis3lv02_power_t power,
lis3lv02_datarate_t datarate){
     uint8_t wdata = 0x07;   //XYZ enabled
```

```
      switch(power){
          case LIS3LV02_POWERDOWN:
              break;
          case LIS3LV02_POWERON:
              wdata |= 0b01 << 6; break;
      }
      switch(datarate){
          case LIS3LV02_DATARATE_40HZ:
              break;
          case LIS3LV02_DATARATE_160HZ:
              wdata |= 0b01 << 4; break;
          case LIS3LV02_DATARATE_640HZ:
              wdata |= 0b10 << 4; break;
          case LIS3LV02_DATARATE_2560HZ:
              wdata |= 0b11 << 4; break;
      }
      ClearCSSWSPI();
      WriteSWSPI(CTRL_REG1 /*| 0x80*/);
      WriteSWSPI(wdata);
      SetCSSWSPI();
  }
  /**
   * @brief Function to select full scale, data update mode, enable
interrupt,
   *        enable data-ready generation, set spi mode and select data
alignment
   * @param scale
   * @param update
   * @param inte
   * @param drdy
   */
  void
lis3_set_scale_update_int_drdy_spimode_alignment(lis3lv02_scale_t
scale, lis3lv02_block_update_t update, lis3lv02_int_enable_t inte,
lis3lv02_drdy_enable_t drdy, lis3lv02_spimode_t spimode,
lis3lv02_alignment_t alignment){
      uint8_t wdata = 0x00;
      switch(alignment){
          case LIS3LV02_DATA_12R:
              break;
          case LIS3LV02_DATA_16L:
              wdata = 0x01;
              break;
      }
      switch(scale){
          case LIS3LV02_SCALE_2G:
              break;
          case LIS3LV02_SCALE_6G:
              wdata |= 0b1 << 7; break;
      }
      switch(update){
          case LIS3LV02_CONTINUOUS:
              break;
          case LIS3LV02_BLOCKED:
              wdata |= 0b1 << 6; break;
      }
      switch(inte){
```

```
            case LIS3LV02_INT_DISABLED:
                break;
            case LIS3LV02_INT_ENABLED:
                wdata |= 0b1 << 3; break;
        }
        switch(drdy){
            case LIS3LV02_DRDY_DISABLED:
                break;
            case LIS3LV02_DRDY_ENABLED:
                wdata |= 0b1 << 2; break;
        }
        switch(spimode){
            case LIS3LV02_SPI_4W:
                break;
            case LIS3LV02_SPI_3W:
                wdata |= 0b1 << 1; break;
        }
        ClearCSSWSPI();
        WriteSWSPI(CTRL_REG2);
        WriteSWSPI(wdata);
        SetCSSWSPI();
    }
    /**
     * @brief Function to configure filter options
     * @param clock
     * @param dirdet
     * @param freefall
     * @param data
     * @param freq
     */
    void lis3_set_filter_options(lis3lv02_clock_t clock,
lis3lv02_filter_enable_t dirdet, lis3lv02_filter_enable_t freefall,
lis3lv02_filter_enable_t data, lis3lv02_filter_freq_t freq){
        uint8_t wdata = 0x00;
        switch(clock){
            case LIS3LV02_INTERNAL_OSC:
                break;
            case LIS3LV02_EXTERNAL_PAD:
                wdata |= 0b1 << 7; break;
        }
        switch(dirdet){
            case LIS3LV02_FILTER_BYPASSED:
                break;
            case LIS3LV02_FILTER_ENABLED:
                wdata |= 0b1 << 6; break;
        }
        switch(freefall){
            case LIS3LV02_FILTER_BYPASSED:
                break;
            case LIS3LV02_FILTER_ENABLED:
                wdata |= 0b1 << 5; break;
        }
        switch(data){
            case LIS3LV02_FILTER_BYPASSED:
                break;
            case LIS3LV02_FILTER_ENABLED:
                wdata |= 0b1 << 4; break;
```

```
    }
    switch(freq){   //f_cutoff = (0.318/HPC)*(ODRx/2)
        case LIS3LV02_FILTER_HPC_512:
            break;
        case LIS3LV02_FILTER_HPC_1024:
            wdata |= 0b01; break;
        case LIS3LV02_FILTER_HPC_2048:
            wdata |= 0b10; break;
        case LIS3LV02_FILTER_HPC_4096:
            wdata |= 0b11; break;
    }
    ClearCSSWSPI();
    WriteSWSPI(CTRL_REG3);
    WriteSWSPI(wdata);
    SetCSSWSPI();
}
/**
 * @brief Reset HP_FILTER_RESET register
 */
void lis3_reset_filter(){
    ClearCSSWSPI();
    WriteSWSPI(HP_FILTER_RESET | 0x80);
    WriteSWSPI(0x00);
    SetCSSWSPI();
}
/**
 * @brief Read STATUS_REG register content
 * @return status register data
 */
uint8_t lis3_read_status_reg(){
    uint8_t rdata;
    ClearCSSWSPI();
    WriteSWSPI(STATUS_REG | 0x80);
    rdata = WriteSWSPI(0x00);
    SetCSSWSPI();
    return rdata;
}
/**
 * @brief Function to read high-byte of X-axis acceleration
 * @return x_HIGH
 */
int8_t lis3_get_acc_x_h(){
    int8_t x_h;
    ClearCSSWSPI();
    WriteSWSPI(OUTX_H | 0x80);
    x_h = WriteSWSPI(0x00);
    SetCSSWSPI();
    return x_h;
}
/**
 * @brief Function to read low-byte of X-axis acceleration
 * @return x_LOW
 */
uint8_t lis3_get_acc_x_l(){
    uint8_t x_l;
    ClearCSSWSPI();
    WriteSWSPI(OUTX_L | 0x80);
```

```
    x_l = WriteSWSPI(0x00);
    SetCSSWSPI();
    return x_l;
}
/**
 * @brief Function to read high-byte of Y-axis acceleration
 * @return y_HIGH
 */
int8_t lis3_get_acc_y_h(){
    int8_t y_h;
    ClearCSSWSPI();
    WriteSWSPI(OUTY_H | 0x80);
    y_h = WriteSWSPI(0x00);
    SetCSSWSPI();
    return y_h;
}
/**
 * @brief Function to read low-byte of Y-axis acceleration
 * @return y_LOW
 */
uint8_t lis3_get_acc_y_l(){
    uint8_t y_l;
    ClearCSSWSPI();
    WriteSWSPI(OUTY_L | 0x80);
    y_l = WriteSWSPI(0x00);
    SetCSSWSPI();
    return y_l;
}
/**
 * @brief Function to read high-byte of Z-axis acceleration
 * @return z_HIGH
 */
int8_t lis3_get_acc_z_h(){
    int8_t z_h;
    ClearCSSWSPI();
    WriteSWSPI(OUTZ_H | 0x80);
    z_h = WriteSWSPI(0x00);
    SetCSSWSPI();
    return z_h;
}
/**
 * @brief Function to read low-byte of Z-axis acceleration
 * @return z_LOW
 */
uint8_t lis3_get_acc_z_l(){
    uint8_t z_l;
    ClearCSSWSPI();
    WriteSWSPI(OUTZ_L | 0x80);
    z_l = WriteSWSPI(0x00);
    SetCSSWSPI();
    return z_l;
}
/**
 * @brief Function to read X-axis acceleration (16 bit alignment
only)
 * @return x
 */
```

```
  int16_t lis3_get_acc_x(){
      int16_t x;
      ClearCSSWSPI();
      WriteSWSPI(OUTX_H | 0x80);
      x = WriteSWSPI(0x00) << 4;
      SetCSSWSPI();

      ClearCSSWSPI();
      WriteSWSPI(OUTX_L | 0x80);
      x |= (WriteSWSPI(0x00) >> 4);
      SetCSSWSPI();

      return x;
  }
  /**
   * @brief Function to read Y-axis acceleration (16 bit alignment
only)
   * @return y
   */
  int16_t lis3_get_acc_y(){
      int16_t y;
      ClearCSSWSPI();
      WriteSWSPI(OUTY_H | 0x80);
      y = WriteSWSPI(0x00) << 4;
      SetCSSWSPI();

      ClearCSSWSPI();
      WriteSWSPI(OUTY_L | 0x80);
      y |= (WriteSWSPI(0x00) >> 4);
      SetCSSWSPI();

      return y;
  }
  /**
   * @brief Function to read Z-axis acceleration (16 bit alignment
only)
   * @return z
   */
  int16_t lis3_get_acc_z(){
      int16_t z;
      ClearCSSWSPI();
      WriteSWSPI(OUTZ_H | 0x80);
      z = WriteSWSPI(0x00) << 4;
      SetCSSWSPI();

      ClearCSSWSPI();
      WriteSWSPI(OUTZ_L | 0x80);
      z |= (WriteSWSPI(0x00) >> 4);
      SetCSSWSPI();

      return z;
  }
  /**
   * @brief Function to read X-axis acceleration in 2 bytes reading (16
bit alignment only)
   * @return x
   */
```

```
int16_t lis3_get_acc_x16(){
    int16_t x;
    ClearCSSWSPI();
    WriteSWSPI(OUTX_L | 0x80 | 0x40);
    x = WriteSWSPI(0x00) >> 4;
    x |= (WriteSWSPI(0x00) << 4);
    SetCSSWSPI();
    return x;
}
/**
 * @brief Function to read Y-axis acceleration in 2 bytes reading (16
bit alignment only)
 * @return y
 */
int16_t lis3_get_acc_y16(){
    int16_t y;
    ClearCSSWSPI();
    WriteSWSPI(OUTY_L | 0x80 | 0x40);
    y = WriteSWSPI(0x00) >> 4;
    y |= (WriteSWSPI(0x00) << 4);
    SetCSSWSPI();
    return y;
}
/**
 * @brief Function to read Z-axis acceleration in 2 bytes reading (16
bit alignment only)
 * @return z
 */
int16_t lis3_get_acc_z16(){
    int16_t z;
    ClearCSSWSPI();
    WriteSWSPI(OUTZ_L | 0x80 | 0x40);
    z = WriteSWSPI(0x00) >> 4;
    z |= (WriteSWSPI(0x00) << 4);
    SetCSSWSPI();
    return z;
}
```

accs2arrays.m

```
fprintf('ax = {')
for i = ax
    fprintf("%d,", i);
end
fprintf('\b}\n')

fprintf('ay = {')
for i = ay
    fprintf("%d,", i);
end
fprintf('\b}\n')

fprintf('az = {')
for i = az
    fprintf("%d,", i);
end
```

```
  fprintf('\b}\n')
```

acc_file_plot.m

```
  function acc_file_plot(archivo)
  %% Load data
  data=load(archivo);
  tiempo = data(:,1);
  muestras = length(data);
  minx = -inf;
  maxx = muestras;
  %% Plot accs and energies
  figure(2);
  subplot(3,1,1);
  plot(tiempo,data(:,2), tiempo,data(:,3),
tiempo,data(:,4));xlim([minx,maxx]);ylim([-70,120]);
  title({'Experimento 11  -  Caída desde banco sentado a rodilla y
apoya las manos (cuadrupedia) suelo blando';'';'Aceleraciones en bruto
en los tres ejes'});
  % xlabel('Tiempo (muestras)');
  ylabel('Aceleración (LSb)');legend('Acc_x','Acc_y','Acc_z');grid();

  subplot(3,1,2);
  plot(tiempo,data(:,5),tiempo,data(:,6),tiempo,data(:,7));ylim([-
inf,inf]);xlim([minx,maxx]);
  title('Aceleraciones filtradas en los tres ejes');
  % xlabel('Tiempo (ms)');
  ylabel('Aceleración
(LSb)');legend('Acc_{xf}','Acc_{yf}','Acc_{zf}');grid();

  subplot(3,1,3);
  plot(tiempo,data(:,8), tiempo,data(:,9), tiempo,data(:,10));ylim([-
inf,inf]);xlim([minx,maxx]);
  title('Energías en los tres ejes');
  ylabel('Energía (LSb)');legend('En_x','En_y','En_z');grid();
  xlabel('Tiempo (muestras)');
  %% Plot flags
  figure(3);
  subplot(3,1,1);
  plot(tiempo,data(:,11),tiempo,data(:,12),tiempo,data(:,13));
  title('Flags de superación de umbral de
aceleración');ylim([0,1.2]);xlim([minx,maxx]);
  ylabel('Activación (bool)');legend('Acc_x flag','Acc_y flag','Acc_z
flag');grid();

  subplot(3,1,2);
  plot(tiempo,data(:,14),tiempo,data(:,15),tiempo,data(:,16));
  title('Flags de superación de umbral de
energía');ylim([0,1.2]);xlim([minx,maxx]);
  ylabel('Activación (bool)');legend('En_x flag','En_y flag','En_z
flag');grid();

  subplot(3,1,3);
  plot(tiempo,data(:,17));
  title('Flag de detección de
impacto');ylim([0,1.2]);xlim([minx,maxx]);
```

```
ylabel('Activación (bool)');legend('Impacto');grid();
end
```

plot_accs.m

```
%% Create time array
at = zeros(size(ax));
for i = 1:size(at,2)
    at(i) = i;
end
%% Plot raw accelerations
figure(1);
subplot(3,1,1);
plot(at,ax,'b',at,ay,'r',at,az,'g');
legend("AccX","AccY","AccZ");grid();
ylabel("Aceleraciones en bruto (LSb)");
title("Aceleraciones del experimento 11: Caída desde banco sentado a
rodilla y apoya las manos (4 patas) suelo blando");
% hold on
%% Create filtered accelerations arrays
axf = ax;
axf(1)=fix(ax(1)/2);
for i=2:size(ax,2)
    axf(i)=fix((ax(i)-ax(i-1))/2);
end

ayf = ay;
ayf(1)=floor(ay(1)/2);
for i=2:size(ay,2)
    ayf(i)=fix((ay(i)-ay(i-1))/2);
end

azf = az;
azf(1)=floor(az(1)/2);
for i=2:size(az,2)
    azf(i)=fix((az(i)-az(i-1))/2);
end

%% Create energy arrays

enx = zeros(size(ax));
enx(1:31) = 0;
for i=32:size(enx,2)
    enx(i)=enx(i-1) + axf(i)*axf(i) - axf(i-7)*axf(i-7);
end

eny = zeros(size(ay));
eny(1:31) = 0;
for i=32:size(eny,2)
    eny(i)=eny(i-1) + ayf(i)*ayf(i) - ayf(i-7)*ayf(i-7);
end

enz = zeros(size(az));
enz(1:31) = 0;
for i=32:size(enz,2)
    enz(i)=enz(i-1) + azf(i)*azf(i) - azf(i-7)*azf(i-7);
```

```
end

%% Plot filtered accelerations and energies
figure(1);
subplot(3,1,2);
plot(at,axf,'b',at,ayf,'r',at,azf,'g');
legend("AccX_f","AccY_f","AccZ_f");grid();ylim([-5,5]);
ylabel("Aceleraciones filtradas (LSb)");

subplot(3,1,3);
plot(at,enx,'b',at,eny,'r',at,enz,'g');
legend("EnX","EnY","EnZ");grid();ylim([-inf,inf]);
ylabel("Energías (LSb)");
xlabel("Tiempo (muestras)");
% hold off

%% Create flags arrays
% acc_x_flag = zeros(size(at));
% acc_y_flag = acc_x_flag;
% acc_z_flag = acc_x_flag;
% en_x_flag = zeros(size(at));
% en_y_flag = en_x_flag;
% en_z_flag = en_x_flag;
% acc_threshold = 14;
% en_threshold = 9;
%
% for i=1:size(acc_x_flag,2)
%     if axf(i) >= acc_threshold
%         acc_x_flag(i) = 1;
%     end
% end
% for i=1:size(acc_y_flag,2)
%     if ayf(i) >= acc_threshold
%         acc_y_flag(i) = 1;
%     end
% end
% for i=1:size(acc_z_flag,2)
%     if azf(i) >= acc_threshold
%         acc_z_flag(i) = 1;
%     end
% end
%
% for i=1:size(en_x_flag,2)
%     if enx(i) >= en_threshold
%         en_x_flag(i) = 1;
%     end
% end
% for i=1:size(en_y_flag,2)
%     if eny(i) >= en_threshold
%         en_y_flag(i) = 1;
%     end
% end
% for i=1:size(en_z_flag,2)
%     if enz(i) >= en_threshold
%         en_z_flag(i) = 1;
%     end
% end
```

```
  %% Plot accelerations flags
  % % figure(1)
  % subplot(3,1,3);
  %
plot(at,acc_x_flag,'c',at,acc_y_flag,'m',at,acc_z_flag,'y',at,en_x_fla
g,at,en_y_flag,at,en_z_flag);
  %
legend("AccX_flag","AccY_flag","AccZ_flag","EnX_flag","EnY_flag","EnZ_
flag");grid();
  % ylabel("Flags de aceleraciones y energías (bool)");ylim([0,1.2]);
  % xlabel("Tiempo (muestras)");
  % % hold off
```