Unidad Ejecutora:

UNLaM- Departamento de Ingeniería e Investigaciones Tecnológicas

Título del proyecto de investigación:

Sistema de monitoreo y alarma para personas adultos mayores ambulantes

C182

Programa de acreditación:

PROINCE

Director del proyecto:

Dr. Daniel Alberto Giulianelli

Co-Director del proyecto:

Lic. Graciela Elisabeth De Luca

Integrantes del equipo:

Docentes-investigadores

Ing. Sebastián Barillaro; Ing. Esteban Carnuccio; Ing. Nicanor B. Casas

Ing. Gerardo Garcia; Ing. Mariano Volker; Ing. Waldo Valiente

Fecha de inicio: 01/01/2016

Fecha de finalización: 31/12/2017

Informe Final

Sistema de monitoreo y alarma para personas adultos mayores ambulantes

Resumen

Este trabajo, intenta acercar a los adultos mayores, la ayuda de un cuidador online que pueda conseguirle la asistencia necesaria en el momento oportuno. El objetivo fue desarrollar un prototipo de un sistema portable que le realice el chequeos constantes, verificando si existe un problema de caída, mediante sus datos posicionales y señales de alarma entre otros, enviando los datos mediante las redes de datos disponibles a al cuidador. Estos los reciben en el celular o en la computadora para realizar el seguimiento de la persona y recibir mensajes de alertas, siendo de bajo costo para su adquisición y utilización.

Se realizaron pruebas para generar el algoritmo de detección de caídas que incorporaron un mecanismo para la detección de falsos positivos utilizando únicamente el acelerómetro y un nuevo sistema de configuración del equipo con mayor persistencia en el tiempo que los disponibles anteriormente. Se pasó de un dispositivo cableado a uno portable con baterías de litio que permitieron la realización de pruebas empleando a tal fin el sensor MPU6050 con la placa Intel® Galileo GenI. Además se desarrolló un servidor en la nube que almacena y distribuye los mensajes de alerta, y la aplicación móvil que recibe las alertas.

Palabras clave: IoT, Detección de Caídas, Falsos Positivos, Alertas.

Keywords: IoT, Fall Detection, False Positive, Alerts.

Estructura

En este apartado se presenta la estructura del presente informe la cual toma como base la propuesta en forma general de la guía de informes finales, realizando agregados (indicándose con una –A– después del título) o quitando aquellos títulos que no aplican en la presente temática (indicándose con una –NA– después del título)

Organización del Informe Final

1. **-Introducción**:
   - 1.1. Selección del Tema
   - 1.2. Definición del Problema
   - 1.3. Justificación del Estudio
   - 1.4. Limitaciones
   - 1.5. Alcances del Trabajo
   - 1.6. Objetivos
   - 1.7. Hipótesis
2. **-Desarrollo:**
   - 2.1. Material y Métodos
   - 2.2. Lugar y Tiempo de la Investigación
   - 2.3. Descripción del Objeto de Estudio
   - 2.4. Descripción de Población y Muestra –NA-
   - 2.5. Diseño de la Investigación

1. Introducción

1.1 Selección del Tema

La nueva tendencia en la era de la informática será fuera del ámbito de la computadora de escritorio tradicional. En el paradigma de Internet de las cosas (IoT), muchos de los objetos que nos rodean estarán conectados a la red de una forma u otra. Se implementará identificación por Radio Frecuencia (RFID) y se desplegará redes de sensores para cumplir con este nuevo reto, en el que los sistemas de información y comunicación son invisibles e incrustados en el medio ambiente que nos rodea.

El término Internet de las Cosas fue acuñado por primera vez por Kevin Ashton en 1999 en el contexto de la gestión de la cadena de suministro [1]. Sin embargo, actualmente la definición ha sido más inclusiva abarcando una gran variedad de aplicaciones como el cuidado de la salud, servicios públicos, transporte, etc. Aunque la definición de "cosas" ha cambiado con la evolución de la tecnología, el objetivo principal sigue siendo el de conseguir que la información computacional tenga sentido sin la intervención humana. La evolución de Internet actual mediante los objetos interconectados en la red, no sólo recolecta la información del entorno (detección) e interactúa con el mundo físico (accionamiento / mando / control), sino que también utiliza Internet para proporcionar servicios de transferencia de información, análisis, aplicaciones y comunicaciones. Lo se incrementa con los dispositivos con la tecnología inalámbrica abierta como Bluetooth, la identificación por radiofrecuencia (RFID), WiFi, y los servicios de datos telefónicos, así como los nodos de sensores y actuadores integrados. La revolución de Internet llevó a la interconexión entre las personas en una escala y velocidad sin precedentes. La próxima revolución será la interconexión entre los objetos para crear un entorno inteligente. Sólo en 2011 hizo que el número de dispositivos interconectados en el planeta superará el número real de personas. Actualmente hay 9 mil millones de dispositivos interconectados y se espera llegar a 24 mil millones de dispositivos en 2020, generando oportunidades de ingresos en segmentos tales como la salud, automotriz, servicios públicos y la electrónica de consumo. Los que van desde usuarios individuales a organizaciones de ámbito nacional que abordan cuestiones de amplio alcance.

La combinación de las actuales tecnologías ofrece una amplia gama de posibilidades para construir sistemas que asistan a personas de grupos de interés prioritario para vivir de manera más segura y poder contar con asistencia inmediata.

1.2 Definición del Problema

Mediante un análisis previo de los dispositivos disponibles, encontramos múltiples aplicaciones de salud para dispositivos móviles inteligentes. También hallamos en el mercado actual infinidad de sensores útiles para aplicaciones médicas: aparatos para medir la presión, pulso cardíaco, glucómetros, acelerómetros, entre otros. Conjuntamente con esto la expansión de Internet y los dispositivos móviles en la mayor parte de la población, nos encontramos con la posibilidad diseñar un sistema de ayuda para personas de edad avanzada mediante monitoreo hogareño.

En el mercado existen empresas que instalan sensores de movimiento en todo el hogar conjuntamente con pastilleros inteligentes, tensiómetros conectados a la red, que entre otros servicios les permiten a los familiares del usuario saber la localización exacta de la persona que necesita ser monitoreada. También es posible encontrar camas hospitalarias con sensores complejos para pacientes con internación domiciliaria, que realizar controles y en caso de ser necesario activa alarmas para permitir una atención médica urgente. Por ejemplo, BIODATA DEVICES [2]. Luego de realizarse el relevamiento de empresas que disponen dispositivos aislados o con sistemas de monitorización, se decidió diseñar una solución que brinde características diferenciadas con aporte tecnológico y social diferente a las existentes. Ya que existe además una gran gama de sensores y dispositivos, resultado imprescindible realizar una exhaustiva revisión de calidad, consumo, disponibilidad, entre otras cosas para las aplicaciones relacionadas con el soporte para personas.
También se encontró la existencia de la necesidad de mejorar la calidad de vida de las personas mayores y de sus familiares mediante una tecnología que vaya en auxilio de sus necesidades en forma eficiente, amigable, segura y a un bajo costo, para poder ser adquirida por una gran parte de la población.

1.3 Justificación del Estudio

Mediante un análisis previo de los dispositivos disponibles, encontramos múltiples aplicaciones de salud para dispositivos móviles inteligentes, también hallamos en el mercado actualmente infinidad de sensores útiles para aplicaciones médicas, desde aparatos para medir la presión, pulso cardíaco, glucómetros, acelerómetros, entre otros. Conjuntamente con esto la expansión de internet y los dispositivos móviles en la mayor parte de la población, nos encontramos con la posibilidad diseñar un sistema de ayuda mediante el monitoreo hogareño para personas de edad avanzada. Teniendo en cuenta las características de esta población, ya que la mayoría de ellos no utilizan teléfonos inteligentes, o utilizan muy pocos prestaciones de los mismos (Aceros, Cavalcante, & Doménech, 2013). En el mercado existen empresas que instalan sensores de movimiento en todo el hogar conjuntamente con pastilleros inteligentes, tensiómetros conectados a la red, entre otros servicios que permiten a la familia saber la localización exacta de la persona a quién quieren monitorear, o camas hospitalarias con sensores complejos para pacientes con internación en su hogar, que en caso de ser necesario activa alarmas para realizar controles y permitir una atención médica urgente como BIODATA DEVICES [2]. Al realizar el relevamiento de empresas que disponen dispositivos aislados o con sistemas de monitorización, se decidió desarrollar un sistema que aporte algunas características diferenciadas con aporte tecnológico y social diferente, ya que las antes mencionadas empresas están dedicadas especialmente a pacientes sin movilidad o muy poca que necesitan control médico continuo , internados en una institución o en su hogar.
Existe además una gran gama de sensores y dispositivos, resultado imprescindible realizar una exhaustiva revisión de calidad, consumo, disponibilidad, entre otras cosas para las aplicaciones relacionadas con el soporte para personas.
También se encontró que existe una necesidad de mejorar la calidad de vida de las personas mayores y de sus familiares mediante una tecnología que vaya en auxilio de sus necesidades en forma eficiente, amigable , segura y a un bajo costo, para poder ser adquirida por una gran parte de la población [3].

1.4 Limitaciones

Son todas aquellas restricciones del diseño de esta y de los procedimientos utilizados para la recolección, procesamientos y análisis de los datos. Así como los obstáculos encontrados en la ejecución de la investigación.
El diseño del prototipo en cuanto al hardware utilizado se realizará con las placas Intel® Galileo GEN I, los sensores para Arduino disponibles en el mercado nacional (debido a las dificultades de importación), con baterías de Litio con una duración reducida, debido al consumo de las placas Galileo. Las comunicaciones se realizarán por Internet a dispositivos con el sistema Android únicamente.

1.5 Alcances del Trabajo

Mediante este trabajo se obtendrá un prototipo portable con baterías que permita detectar una caída de la persona monitoreada, eliminado los falsos positivos, enviar el mensaje a un servidor que guardará estos datos para generar un historial permitiendo futuros análisis y a su vez reenviará el mensaje recibido una aplicación Android que posea la o las personas a cargo.

1.6 Objetivos

- Diseñar un sistema de monitoreo, alerta y ubicación de personas adultas mayores ambulantes.
- Priorizar costos reducidos considerando como usuario final una población de bajos recursos, empleando tecnología existente en el mercado y adaptada para su fácil utilización.
- Permitir la transferencia y recolección de datos provistos por los sensores integrados, para que una persona a cargo pueda recibir las alertas de la persona monitoreada.
- Establecer los mecanismos de seguridad y confidencialidad de los datos.
- Desarrollar una aplicación en el sistema embebido para recolectar, realizar la comunicación y transferencia los datos que pueda ejecutan conectarse con otras aplicaciones, en un teléfono inteligente y/o en una computadora de escritorio.
- Desarrollar un prototipo para el dispositivo que llevará consigo la persona a monitorear
- Desarrollar las base de datos y el servidor web para la recepción, recolección y envío de mensajes.
- Generar documentación necesaria para que dicho sistema pueda ser reproducido y actualizado

1.7 Hipótesis

Es posible diseñar un dispositivo portable físicamente, como un accesorio, que permita monitorear, determinar la ubicación de una persona en un entorno físico cerrado o abierto, que detecte una caída o algún cambio significativo en sus signos vitales y envíe en forma automática o manual un pedido de ayuda en caso de ser necesario [4]
Los datos enviados deben ser recibidos en un dispositivo móvil o una computadora con un sistema de alarma para alertar al receptor de la situación.
Esta solución se realizará mediante la combinación de sistemas embebidos conjuntamente con la transmisión de datos por red, su procesamiento en la nube, lo que permitirá redundar en mayor ayuda y seguridad para las persona adultas mayores que lo requieran, las que contarán con un cuidador o familiar a cargo que utilizará el sistema de monitoreo remoto provisto y a un costo reducido [5].

2. Desarrollo

2.1 Material y Métodos

El desarrollo del monitor de caídas de adultos mayores está compuesto por tres partes principales que fueron diseñadas y construidas para este proyecto.

- ***Prototipo***: Se creó un Instrumento encargado de recabar la información motriz de la persona mayor. Consta de un sensor MPU6050, el algoritmo de detección de caídas y la interfaz de comunicación vía WIFI a Internet; con el fin de detectar las caídas y disparar un mensaje de notificación.

- ***Servidor en la nube***: Centraliza los avisos que producen los instrumentos monitores (prototipo) redirigiendo las notificaciones al celular correspondiente y guarda la información recibida en una base de datos.

- ***Aplicación de celular***: Da de alta al celular en el sistema de monitoreo, recibe la notificación de la caída del individuo monitoreado y permite realizar una llamada a un contacto cercano del monitoreado o a un servicio de urgencia.

Se utilizaron para la realización del prototipo placas de microcontrolador Intel® Galileo I basada en el procesador de aplicaciones Intel® Quark SoC[1] X1000, un sistema de clase Intel® Pentium de 32 bits en un chip, pin compatible con Arduino, recibidas como donación de Intel® Argentina. Las placas Galileo permiten utilizar una distribución de Linux, denominada Yocto, para su funcionamiento normal. Lo que permitió desarrollar aplicaciones en el sistema embebido utilizando los lenguajes de programación Node.Js y Wiring, las que están encargadas de detectar la caída de la persona, sensando constantemente los valores obtenidos del integrado MPU6050. Para asegurar la portabilidad y usabilidad del prototipo por la persona anciana, se adaptaron las placas Galileo para que utilicen baterías de Litio que generan 8 volts como fuente de alimentación. Además se utilizó la placa Wifi Intel® Centrinio Wireless Wimax 6150, compatible con esta placa de desarrollo, para poder realizar la transferencia de datos entre el prototipo y el servidor en la nube. De esta forma se pudo aplicar un protocolo de comunicación REST[2] en todo el sistema detector de caídas. Así mismo fue necesario implementar nuestra propia aplicación de servicios web, encargada de administrar las alertas de caídas y los datos de los individuos monitoreados, registrados en la base de datos del sistema. Dicho software fue desarrollado en un servidor en la nube, utilizando el Framework Laravel de PHP. Al mismo tiempo se empleó el servicio de notificaciones de Google, denominado Firebase, para emitir las notificaciones de caídas a los dispositivos móviles de los familiares del monitoreado que estén registrados en la base de datos del sistema y que tengan instalada en su dispositivo móvil la aplicación Android que fue desarrollada para este proyecto. Este último permite al usuario del celular ser alertado sobre las caídas que sufra el individuo que tenga a cargo y administrar sus datos en el sistema.

2.2 Lugar y Tiempo de la Investigación

La investigación se realizó en el predio de la Universidad Nacional de la Matanza, ubicada en Florencio Varela 1903 (B1754JEC) –San Justo, Buenos Aires, Argentina–. Para la construcción del prototipo y el desarrollo de los programas se utilizó del Laboratorio 266, perteneciente al Departamento de Ingeniería e Investigaciones Tecnológicas, el cual está provisto de equipos con sistemas operativos, Windows y Linux, placas Intel Galileo y sensores que forman el prototipo. Para realizar las pruebas de caídas se utilizaron las instalaciones del Gimnasio del Campus de la Universidad.

El tiempo empleado fue de dos años desde 2016 a 2017. El primer año se construyó la primera versión del prototipo y del procesamiento de los datos leídos por el sensor con un graficador para el análisis. También se desarrolló el programa para la recolección de las mediciones iniciales de caídas, para posibilitar la reproducción posterior, con el fin de poder probar en forma reiterada el algoritmo de caída. Todas las mediciones realizadas durante el proyecto de investigación se pueden observar en el **Anexo A**.

En la segunda parte, se finalizó el prototipo de caídas, transformándolo de una plaqueta cableada a una versión inalámbrica para permitir ser transportada. Se realizó la depuración del algoritmo de caídas, mediante la realización de un conjunto de pruebas reales y recolectadas en

[1] System on Chip

[2] Estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web

archivos. De esta forma se verificó la exactitud minimizando falsos positivos. Finalmente se completó el sistema mediante el desarrollo del servidor web y la aplicación móvil.

2.3 Descripción del Objeto de Estudio.

El prototipo de sistema de monitoreo y alarma para adultos mayores ambulantes posee las siguientes características:

- Es un sistema portable a batería con conexión a Internet.
- Permite la detección de eventos, en particular movimientos bruscos que puedan dar como resultado una situación de caída.
- Reconoce distintos tipos relacionados con caídas tales como:

  - ✓ **Posición inicial - PARADO**
    - Hacia adelante
    - Hacia los costados
    - Hacia atrás
    - Vertical
    - Vertical y hacia adelante – DOBLAN LAS RODILLAS –
    - Vertical y hacia atrás – DOBLAN LAS RODILLAS –

  - ✓ **Posición inicial - SENTADO**
    - Hacia los costado
    - Hacia adelante – la cabeza-

  - ✓ **Posición inicial - AGACHADO**
    - Hacia los costados
    - Hacia adelante
    - Hacia atrás
    - No se puede levantar- fractura de cadera-

  - ✓ **Posición inicial – ACOSTADO**
    - Rodar de la cama.
    - No poderse levantar

- Reconoce falsos positivos.

  - ✓ **Posición inicial - PARADO**
    - Hacia adelante
    - Hacia los costados
    - Hacia atrás
    - Vertical y hacia adelante – DOBLAN LAS RODILLAS –
    - Sentarse rápido
    - Bajar y subir rápido por una escalera

  - ✓ **Posición inicial - SENTADO**
    - Hacia el costado rápido

  - ✓ **Posición inicial - AGACHADO**
    - Hacia los costados
    - Hacia adelante

- Hacia atrás

✓ **Posición inicial - ACOSTADO**
- Rodar de la cama
- No poderse levantar
- Saltar
- Tirarse en la cama - difícil de detectar diferencias-

✓ Realiza la notificación del evento por medio de mensajes.
✓ Un servidor web almacena y distribuye los mensajes.
✓ Un dispositivo móvil recibe la información del evento producido.

2.4 Descripción de Población y Muestra.
-NA-

2.5 Diseño de la Investigación
Las tareas del proyecto se encuentran divididas por 3 etapas (Fig. 1). La etapa inicial se desarrolló durante al primer año. La etapa de avance se realizó iterando sobre ella misma a lo largo del segundo año y la tercera etapa completó el fin del proyecto.

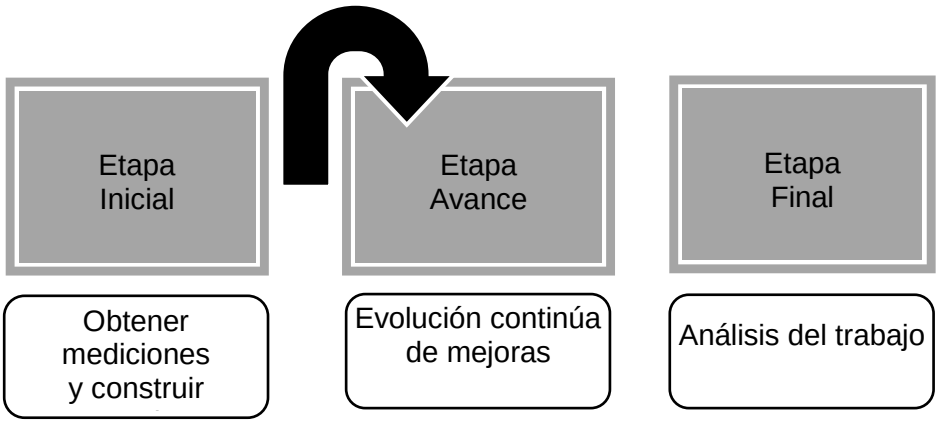


Fig. 1 - Etapas empleadas en la investigación.

Tabla 1 - Gantt primer año

Actividades / Responsables 1er Año	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
Etapa Inicial												
Elaboración inicial del proyecto	x	x	x	x	x	x						
Relevamiento de bibliografía sobre internet de las cosas	x	x	x									
Evaluación de sensores disponibles, características, tamaño, confiabiliadad y consumo de energía.	x	x	x	x	x							
Diseño de un modelo	x	x	x	x	x	x						
Puesta a punto del laboratorio de investigación	x	x	x	x	x							
Realizar cursos de computación internet de las cosas	x	x	x									
Poner en licitación los requerimientos de hardware					x	x	x					
Instalación de plataformas de trabajo para sistemas embebidos.							x	x				
Configuración del hardware existente y el adquirido							x	x	x	x		
Elaboración de casos de prueba incrementales									x	x	x	
Puesta a punto de las herramientas y entorno de medición										x	x	x
Pruebas iniciales de control para comparaciones futuras											x	x
Desarrollo de un prototipo con funcionalidades reducidas.										x	x	x

Tabla 2-Gantt segundo año.

Actividades / Responsables 2do Año	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
Etapa de Avance												
Pruebas del prototipo	x	x	x	x	x	x	x	x	x	x		
Incremento de funcionalidades al prototipo hasta la funcionalidad total	x	x	x	x	x	x	x	x	x	x	x	
Evaluación continua de mejoras	x	x	x	x	x	x	x	x	x	x	x	
Realizar pruebas mediante el uso de distintos dispositivos receptores.					x	x	x	x	x	x	x	
Realizar prueba de confiablidad de la parte de hardware.			x	x	x	x	x					
Probar el desempeño de cada optimización .del sistema.								x	x	x	x	x
Prueba del sistema en un escenario real.							x	x	x	x	x	x
Evaluación de resultados.										x	x	x
Informe de problemas y soluciones										x	x	x
Etapa Final												
Comparación de Resultados y elaboración de conclusiones										x	x	x
Divulgación de resultados y publicaciones												x

2.5.1 Etapas

2.5.1.1 Elaboración inicial del proyecto

Se analizaron distintos tipos de sistemas de monitoreo para personas, los que carecían de la función social que se desea para este proyecto. Podemos citar los que se utilizan en clínicas, para pacientes postrados con múltiples sensores de alta complejidad que reportan a una central con alarmas. Otros para pacientes que deambulan en un ámbito más amplio tales como Hogares de Ancianos o Geriátricos, que son monitoreados mediante cámaras o por el envío de señales a una central computarizada para que el personal asignado para atención lo monitoree, entre otros. Por lo que se decidió realizar un prototipo mediante un sistema de sensores, que sea portable o vestible con sólo las funcionalidades necesarias, para que resulte de muy fácil utilización por una persona mayor. Esto es importante, ya que las personas mayores en general tienen dificultades con las nuevas tecnologías y a su vez necesitan independencia.
Durante la etapa inicial del proyecto se realizó primeramente el análisis de requisitos, la selección del lenguaje de programación adecuado para este proyecto, la selección de la placa utilizada para realizar el prototipo, la elección de los sensores requeridos y se realizó la puesta a punto del laboratorio para la realización de las pruebas.

2.5.1.2 Relevamiento de bibliografía sobre internet de las cosas.

Se realizó la búsqueda durante todo el proyecto sobre bibliografía en sistemas embebidos e internet de las cosas aplicados al monitoreo. Agregándose a la búsqueda el tema referente a las dificultades físicas y los posibles seguimientos más importantes, para ello se utilizaron la

biblioteca del MINCyT conjuntamente con los buscadores académicos disponibles. Evaluación de sensores disponibles, características, tamaño, confiabilidad y consumo de energía.

2.5.1.3 Diseño de un modelo.

En esta propuesta se estableció la interacción entre los componentes de captura y reporte del sistema. La idea fue la de implementar la solución utilizando principalmente el patrón de diseño "*observer*", que nos permite el paralelismo de la implementación como así también evitar el acoplamiento entre los elementos de la solución. La principal diferencia entre los elementos *"reales*" de los "*simulados"* fue poder utilizarlos como entrada en simulaciones y en las depuraciones correspondientes de los algoritmos que posteriormente fueron implementados (Fig.)

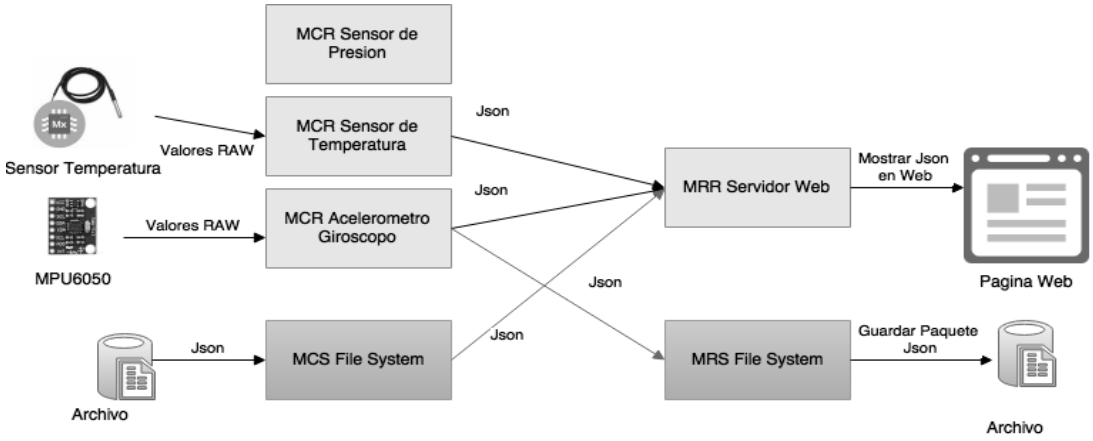


Fig. 2 - Diseño del modelo detector de caídas.

- **MCR:** El **Módulo de Captura Real** es un componente encargado de realizar las lecturas de los sensores y reportar estas lecturas al componente al cual se le suscriban los observadores.

- **MCS:** El **Módulo de Captura Simulada** es un componente encargado de realizar las lecturas del *filesystem* y reportar éstas lecturas al componente al cual se le suscriban los observadores.

- **MRR:** El **Módulo de Reporte Real** es un componente encargado de realizar los reportes de los eventos recibidos desde el **MCR/MCS**.

- **MRS:** El **Módulo de Reporte Simulada** es un componente encargado de realizar los reportes de los eventos recibidos desde el **MCR/MCS** al *filesystem* para futuras pruebas a realizar en el laboratorio y emular las acciones realizadas.

2.5.1.4 Puesta a punto del laboratorio de investigación.

Dado que Intel ® realizó una donación de una cantidad de placas Galileo I al grupo de Investigación SODIUM, se comenzó el desarrollo del prototipo con dichas placas. La ventaja de utilizar estas estuvo dada por la facilidad de contar con un simulador de Arduino y además con un sistema operativo embebido (Yocto[3]) como otra opción para el desarrollo.
Se instalaron los IDES correspondientes, Arduino 1.6.10 y el XDK[4] para Yocto.

[3] El proyecto Yocto es un grupo de trabajo de la Fundación Linux cuyo objetivo es producir herramientas y procesos que permitan la creación de distribuciones Linux para software embebido.

[4] Intel XDK es un kit de desarrollo creado por Intel para crear aplicaciones nativas para teléfonos móviles y tabletas utilizando tecnologías web como HTML5, CSS y JavaScript.

Primeramente se comenzó con Arduino, sin embargo, debido a que el que posee la placa Galileo I es un simulador, no resultaron confiables las mediciones obtenidas de los sensores, por lo que se decidió pasar a utilizar el XDK para Yocto. Además debido a que ciertas bibliotecas desarrolladas para la placa Arduino no son compatibles con la placa Galileo, y sí XDK, que está desarrollado especialmente para funcionar con la placa Galileo esto confirmó la elección del entorno de trabajo con XDK.
Por otra parte también se realizaron pruebas con diferentes placas Wifi, hasta encontrar la que posee compatibilidad con Intel Galileo I disponible en el mercado local. El modelo final utilizado fue el Intel® Centrinio Wireless-N +Wimax 6150.

2.5.1.5 Realizar cursos de computación internet de las cosas.

La información de los cursos realizados y los certificados se encuentran disponibles en el **ANEXO. C.**

2.5.1.6 Poner en licitación los requerimientos de hardware.

Se realizó un estudio de los componentes necesarios para el proyecto, debido a la falta de componentes en el mercado local, se realizaron las compras de los más necesarios o se consiguieron algunos por medio de una donación realizada por la empresa Intel®. También dentro de la donación se recibieron 4 notebooks, una placa Galileo II y un switch de redes[5].

2.5.1.7 Instalación de plataformas de trabajo para sistemas embebidos.

Existen Múltiples opciones para programar un Sistema Embebido. Intel® Galileo Gen I se trata de un dispositivo que tiene un Sistema Operativo Linux de 32 bits corriendo sobre un SoC[6] con un procesador 32 bit, Single Core, Single Thread, Pentium ISA compatible. Cualquiera sea la forma de generar un binario respetando estos parámetros es válida.
Para facilitar las cosas, existen opciones que hacen el trabajo de compilación cruzada, implantación del binario en el dispositivo y depuración de manera automatizada.
Intel brinda, además de la placa de prototipos Galileo, documentación y software para poder trabajar con ella [6]. Pero no es la única opción. Al tratarse de entornos abiertos, otras empresas adoptaron el dispositivo para brindar sus productos y servicios. Muchas veces gratuitamente. Algunas de ellas: Wyliodrin, SparkFun.com, Arduino, WindRiver, entre otros [7].
En este proyecto se decidió la utilización de:

- Arduino

Entorno abierto de desarrollo con el cuál Intel Galileo es compatible (tanto en hardware como en software). El lenguaje de programación utilizado es Wiring, muy similar a C [8].

- Intel® XDK - IoT Edition.

Entorno de desarrollo creado por Intel® para desarrollar aplicaciones multiplataforma orientadas a IoT. De las dos ediciones existentes, nosotros utilizaremos la IoT Edition, que nos permitió trabajar con Sistemas Embebidos. Aunque funciona sin conexión a Internet, los ejemplos sólo están disponibles online. El lenguaje de programación utilizado es node.js, una implementación de Javascript con repositorios de bibliotecas. Se descarga desde [7].

2.5.1.8 Configuración del hardware existente y el adquirido.

En la placa Arduino UNO, el módulo de hardware denominado TWI se encarga de administrar el uso de su bus I2C[7]. Este es controlado y utilizado a través de la biblioteca Wire.h.
Usando su función setclock(), se puede configurar la frecuencia del clock del bus, modificando el registro TWBR del módulo TWI. En Arduino UNO la velocidad del clock se configura a una

[5] Dispositivo de interconexión utilizado para conectar equipos en red formando lo que se conoce como una red de área local (LAN)

[6] System On Chip

[7] **I**nter-**I**ntegrated **C**ircuit es un bus de datos serie.

velocidad estándar de 100 khz y 400 khz como máximo, compatibles con el chip MPU6050 [9]. En cambio la plataforma Intel® Galileo Generación I, emula el comportamiento de la Arduino UNO utilizando el chip Cypress CY8C9540A [10] para administrar el uso de su bus I2C. Al conectar el MPU6050 a la Galileo se pudo determinar, a través del comando i2cdetect de Yocto, que esta plataforma mapea al sensor con la dirección 0x68.
El chip CY8C9540A es usado por la Galileo como una extensión de I/O agregando pines adicionales al QuarkX1000, multiplexando la mayoría de los pines GPIO[8], lo cual puede hacer variar la performance de los pines. Entre los números de GPIO que multiplexa se encuentra la gpio29, que en las pruebas efectuadas debió ser modificada para conseguir que el bus I2C de la Galileo sea accesible desde los pines A4 y A5, que se encuentran desactivados por defecto. Para ello fue necesario modificar en Yocto el archivo *value*, ubicado en el directorio "/sys/class/gpio/gpio29", escribiendo en su contenido el valor 1 en lugar de 0. En la Galileo todos los pines de los GPIO están conectados al CY8C9540A, menos los pines 2 y 3. Lo que restringe la velocidad máxima del I2C a 100 khz, reduciendo de esta manera el aprovechamiento máximo del MPU6050, dado que el sensor puede comunicarse a una velocidad máxima de 400 khz. En las pruebas efectuadas se debió configurar el MPU para que utilice su velocidad estándar de 100 khz y así pueda ser utilizado con la Galileo Gen 1. Por otro lado la Galileo Gen 2 utiliza el chip PCA9555 para administrar el uso de su bus I2C, resolviendo la limitación de velocidad que presentaba su predecesora, pudiendo trabajar a 400 khz.
Inicialmente uno de los problemas encontrados en la comunicación, con Galileo Gen 1 y el MPU6050, tuvo que ver con la comunicación I2C. Esta pretende que se genere un clock para el envío de datos que debe ser de 100 Khz. Pero en mediciones realizadas, a través de un osciloscopio, se generaba una salida de 110 Khz. Una diferencia de frecuencia menor no debería significar ningún problema debido a la característica sincrónica del protocolo. Pero esta diferencia es apreciable y producía diferencias de recepción de datos, pasando de un valor RAW en decimal de 58000 (o más) o de 7000 (o menos). Por ese motivo se investigaron los mecanismos existentes para variar dicha frecuencia, mejorando la recepción de las mediciones en esta plataforma. Aunque en un principio la comunicación I2C era considerado la raíz del problema, luego se descubrió a través de una calibración del giróscopo la solución del problema.
Se encontraron diferencias en la versión de la biblioteca Wire.h para Intel® Galileo con respecto a su versión para Arduino UNO. Debido a que utilizan distintos chips para manejar el bus I2C, TWI en Arduino y Cypress en Galileo, no existe la variable TWBR en este archivo. Dificultando de esta manera la posibilidad de poder modificar la frecuencia del clock usando esa biblioteca a través de los *sketchs*[9].
Un mecanismo encontrado para conseguir variar la frecuencia del clock del bus I2C, consiste en recompilar el Kernel de Yocto. Para ello primero es necesario descargar el código fuente del Sistema Operativo y sus parches. Posteriormente se deberá modificar la frecuencia del clock variando la estructura i2c_mode_info para Gen1, ubicada en el archivo 0115-mfd-intel_quark_i2c_gpio-Add-Intel-Quark-X1000-I2C-G.patch. Finalmente se tendrá que recompilar el código del kernel modificado y sus parches. Esto generará una imagen de Yocto, que modificará la frecuencia del clock I2C al utilizarse para bootear la Galileo.
El proceso de calibración del sensor MPU6050, fue bastante complejo debido a que existen diferentes mecanismos para realizar dicha tarea, pero la gran mayoría son muy pocos performantes para el correcto funcionamiento del sensor. En consecuencia en los apartados posteriores se describen de qué manera se fueron realizando los distintos procesos de calibración del sensor hasta su correcto funcionamiento.

2.5.1.9 Elaboración de casos de prueba incrementales.

Se analizaron algoritmos de detección de caídas basados en el uso del sensor MPU6050. Los estudios analizados son [11] [12] [13] y [14].

[8] **G**eneral **P**urpose **I**nput/**O**utput, Entrada/Salida de Propósito General

[9] Programa de Arduino

En todos los casos, la detección de la caída está basada en el análisis de los valores entregados por el sensor.
Debido a la naturaleza de los movimientos producidos durante una caída, la misma no se puede determinar por el estudio de una dirección en particular aislada de las demás. Por lo que se estudia la resultante de las aceleraciones en los tres ejes (1).

$$a = \sqrt{\left(a_x{}^2 + a_y{}^2 + a_z{}^2\right)} \qquad (1)$$

Una caída es detectada cuando la norma de las aceleraciones "a" supera un umbral establecido. Cabe señalar que no existe un valor que separe completamente los movimientos seguros de los movimientos producidos por una caída. Por ejemplo, acostarse bruscamente podría ser considerado una caída. Si un sensor está ubicado en una mano que es utilizada para rascarse o borrar con una goma, también pueden detectarse falsos positivos. Por estos motivos, fue necesario considerar esta problemática y diseñar una estrategia para detectar y descartar los falsos positivos.

- **Descarte de falso positivos**

Debido a que las caídas no son los únicos movimientos que pueden provocar grandes aceleraciones capaces de superar el umbral establecido para su detección, es posible que algunos movimientos (movimientos seguros) sean interpretados como una caída. En ese caso, se produciría un falso positivo. Existen varias técnicas para detectar falsos positivos. Algunas implementan un algoritmo de aprendizaje, entrenado por el mismo usuario. Otras hacen análisis de señales, estudiando las variaciones de la información brindada por el sensor. Otras más utilizan varios sensores del mismo tipo, distribuidos en el cuerpo del usuario. Por último, una técnica consiste en aprender la posición del cuerpo.
Se planifico inicialmente la técnica de entrenamiento por parte del mismo usuario. Cuando el sistema detecta una posible caída, emite un sonido durante 10 segundos. Si en ese tiempo el usuario no presiona un botón para avisar que se trata de un falso positivo, el sistema entiende que se trata de una caída real y dispara la alarma de caída. Finalmente se decidió no utilizar el botón de pánico, y se modificó el algoritmo para detectar automáticamente cuando la persona se reincorpora luego de haberse caído.

- **Alarma de caída**

La alarma de caída puede ser disparada automáticamente cuando el sistema monitor detecta un cambio brusco en las aceleraciones, de acuerdo al algoritmo de detección de caídas. En caso de confirmar que la caída es real y no se trata de un falso positivo, el sistema disparará una alarma de caída. Esta alarma es una serie de pasos prefijados, para dar aviso a los cuidadores por diferentes medios de comunicación.

- **Pruebas realizadas**

Inicialmente durante la investigación se optó por usar una biblioteca de Arduino que facilite el manejo de los registros del MPU6050 en la Galileo. Para ello se usaron varias bibliotecas de terceros y procesos de calibración del MPU que resultaron fallidos, dado que no brindaban los valores del sensor en forma satisfactoria. Finalmente después de calibrar y nivelar correctamente el MPU, se utilizó la biblioteca desarrollada por Jeff Rowberg, con algunas modificaciones para efectuar las distintas pruebas con el sensor. Para ello mediante un sketch, que se ejecutaba en la Galileo, se logró capturar los valores RAW sensados por el acelerómetro y giroscopio para posteriormente convertirlos a mediciones entendibles para el usuario. Los datos del acelerómetro a m/seg^2 y del giroscopio a °/seg. Para su conversión fue necesario configurar el rango de sensibilidad de dichos sensores. El acelerómetro se los configuró con una de aceleración límite de 2g y una sensibilidad de 16384. En cambio el giroscopio con un rango de velocidad angular de 250 °/seg y una sensibilidad de 131. Además en el programa desarrollado se utilizó el filtro de Kalman para eliminar el ruido de los valores sensados y así obtener una medición limpia. Los valores que se obtiene de las mediciones son enviados a un

programa desarrollado en *Processing* [15], para de esta forma representarlos gráficamente (Fig.) en ejes cartesianos.

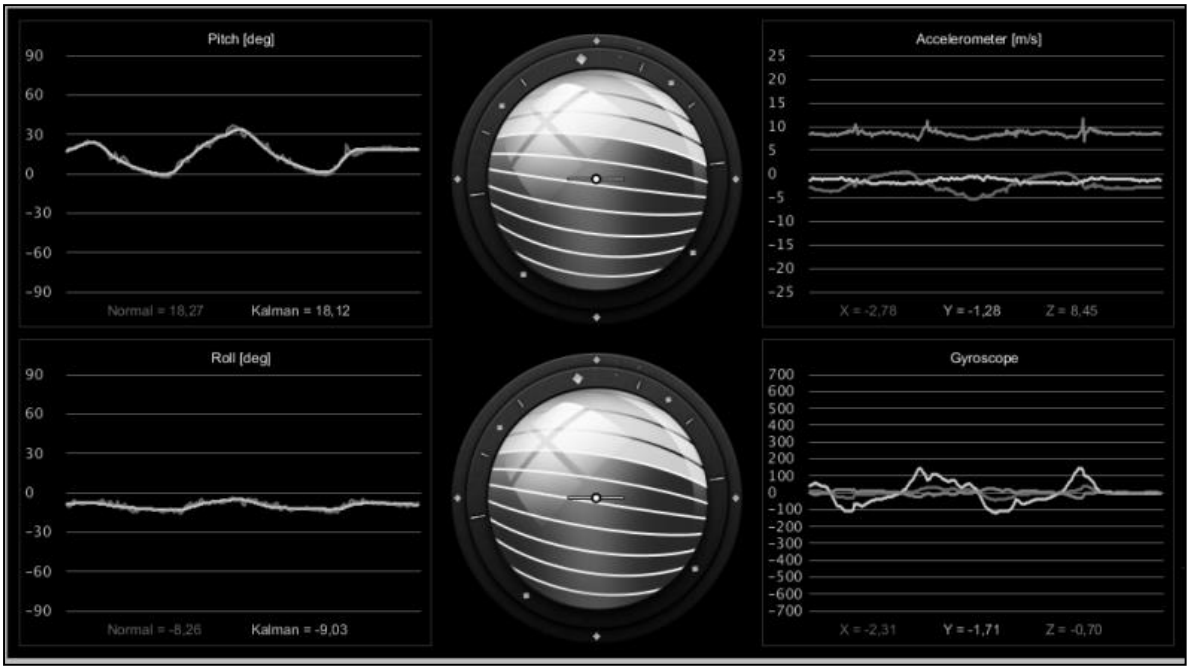


Fig. 3 - Representación gráfica de los valores detectados con el MPU6050.

Se consiguió realizar la calibración del sensor a través del sketch desarrollado por [16]. No obstante, en la documentación oficial del MPU6050 [17] se comienza a nombrar los registros desde el 13 en adelante. Pero en la definición de registros encontrada en la biblioteca utilizada para la comunicación con MPU6050, son definidos a partir del registro 0. Entre estos 13 registros iníciales está la definición del offset de los sensores por eje. Estos registros se llenan con los valores grabados de fábrica al generar la calibración luego de comprobado su funcionamiento. Pero como esos valores son cargados a estos registros desde una memoria ROM, se pueden modificar en los registros de trabajo, que es lo que se realiza con la rutina de calibración. Estos registros no son modificados por el MPU6050, solo leídos así que ante cambios de temperatura estos valores deben ser recalibrados. Como no encontramos información del fabricante respecto a este procedimiento, no se tiene información respecto a lo que hace el integrado al compensar los cambios térmicos. Pero esta biblioteca parece no usar la capacidad interna de procesamiento dada por el DMP[10].

2.5.1.10 Puesta a punto de las herramientas y entorno de medición.

Se determinó conveniente utilizar durante el desarrollo del módulo de detección de caídas al algoritmo de filtro Kalman, con la finalidad de reducir el ruido en los valores entregado por el MPU6050 durante la detección. La característica principal del filtro de Kalman, es su bajo requerimiento en el procesamiento de las señales, lo que lo hace ideal para correr en microcontroladores de bajo poder de cálculo. En este caso, el interés de utilizar este método es por el bajo consumo de recursos, ya que se pretende mantener la exigencia al mínimo del costo del producto final esperado y el bajo consumo pretendido. Estos requerimientos se deben a que se pretende un sistema portable y de gran tiempo operativo. Esto surge al proyectar que el equipo esté el mayor tiempo posible en uso por el adulto mayor objetivo. Si el equipo tuviera un consumo elevado el usuario debe quitárselo para la carga y ese tiempo queda sin

[10] Procesador Digital de Movimiento

monitorización. O si se utiliza una batería muy grande la falta de comodidad y molestias ocasionadas por este equipo, compromete el uso voluntario del usuario.
Por lo que con este filtro se obtiene una calidad de los datos muy buena respecto a su consumo de recursos.

2.5.1.11 Pruebas iniciales de control para comparaciones futuras

En primera instancia se realizaron mediciones de pruebas de caídas iniciales, realizadas por los miembros del equipo. Como se detallará en apartados posteriores, la información de estos valores y el tipo de prueba se almacenaron en archivos para realizar la segunda etapa correspondiente a la simulación para poder obtener luego el prototipo.

2.5.1.12 Desarrollo de un prototipo con funcionalidades reducidas.

En este proyecto se comenzó con el diseño y desarrollo de un prototipo de dispositivo portable a un costo accesible para personas adultas mayores.

2.5.1.13 Pruebas del prototipo.

Las pruebas iterativas se realizaron, para poder hacer un análisis exhaustivo de la ocurrencia de una caída y así obtener un patrón que nos permitiera reconocer los cambios de aceleración y de ubicación espacial que sufre el cuerpo humano durante dicho suceso. Fue necesario realizar diferentes mediciones de aceleración en las distintas actividades diarias llevadas a cabo. Por ese motivo se realizaron diversos conjuntos de tipos de caídas de personas en un ambiente controlado utilizando el prototipo. Mientras transcurrían las mediciones, los distintos valores de aceleración del cuerpo, obtenidos por el acelerómetro durante el transcurso de la actividad, se iban almacenando en archivos CSV, véase Anexo A. A partir de la observación y estudio detallado del contenido de estos archivos se pudo elaborar un patrón de caída eficiente.
Las actividades diarias que se tuvieron en cuenta para poder realizar el análisis de las caídas son las siguientes:

- **Actividades de la vida diaria:** Caminar, saltar, sentarse y levantarse de una silla, subir y bajar por una escalera.

- **Actividades de caídas:** Caerse estando de pie hacia adelante, atrás o de costado. También se realizaron mediciones cayéndose hacia adelante o de costado estando sentado.

Los gráficos resultantes de las mediciones realizadas, por cada actividad, se muestran en el Anexo A. En cada uno de ellos se intentó identificar las distintas fases asociadas a una caída, para luego a partir de estos, poder obtener un patrón de umbrales representativos de cada fase. Para eso se debieron analizar detalladamente los valores máximos y mínimos de las distintas aceleraciones y posteriormente obtener un factor común entre ellos. En base al análisis efectuado y a las experiencias de [18] y [19] , se realizó la siguiente tabla de valores de aceleración representativos de cada fase.

Tabla 3 - Valores representativos de cada fase

Pruebas		Mediciones MPU6050	
		Mínimo	Máximo
AD	Reposo Inicial	1G	3G
	Caída Libre	1G	3G
	Impacto	1G	3G
	Reposo Final	1G	3G
Caídas	Reposo Inicial	1G	3G
	Tiempo _RI		500 mseg
	Caída Libre	0G	1G
	Tiempo_CL		600 mseg
	Impacto	4.5 G	16G
	Tiempo_I		60 mseg
	Reposo_Final	1G	3G
	Tiempo_RF		100 mseg

Referencias de la tabla:

- **AD : Actividades de la vida diaria**
- **Tiempo _RI : Tiempo en Reposo Inicial**
- **Tiempo _CL: Tiempo en Caída Libre**
- **Tiempo _I : Tiempo en Impacto**
- **Tiempo_RF : Tiempo en Reposo Final**

Para comprobar el funcionamiento del algoritmo elaborado, fue necesario repetir las actividades y caídas realizadas anteriormente durante la captura de las mediciones. Por lo tanto se efectuaron dichas validaciones de dos maneras distintas: una simulada y la otra físicamente. La primera de ellas fue utilizando los archivos CSV generados previamente, con los valores de aceleración almacenados en estos. En tanto la segunda forma consistió en volver a realizar las mismas actividades físicamente usando el prototipo, mientras se ejecutaba el algoritmo de caídas.
El proceso de verificación y validación del algoritmo de caídas se debió realizar repetidas veces, debido a que fue necesario adaptar los umbrales mencionados en la Tabla 4 y agregar fases adicionales, como mejoras a las propuestas por [18] y [19] para poder detectar la caída de una persona correctamente.

2.5.1.14 Incremento de funcionalidades al prototipo hasta la funcionalidad total.

El sistema embebido, consiste en un prototipo del detector de caídas que fue evolucionando tanto parte física del aparato como el software del algoritmo. Está implementado como se mencionó en [20], utilizando principalmente una placa Intel® Galileo Generación 1, las baterías que lo alimentan, el sensor acelerómetro MPU6050 y placa de Wifi para la conexión. Este es el encargado de monitorear y controlar en tiempo real el estado de la persona, para detectar si ha sufrido una caída. En caso de haberse producido dicho suceso, realiza la notificación automática al servidor que se encuentra ubicado en la nube, enviándole distintas alertas.

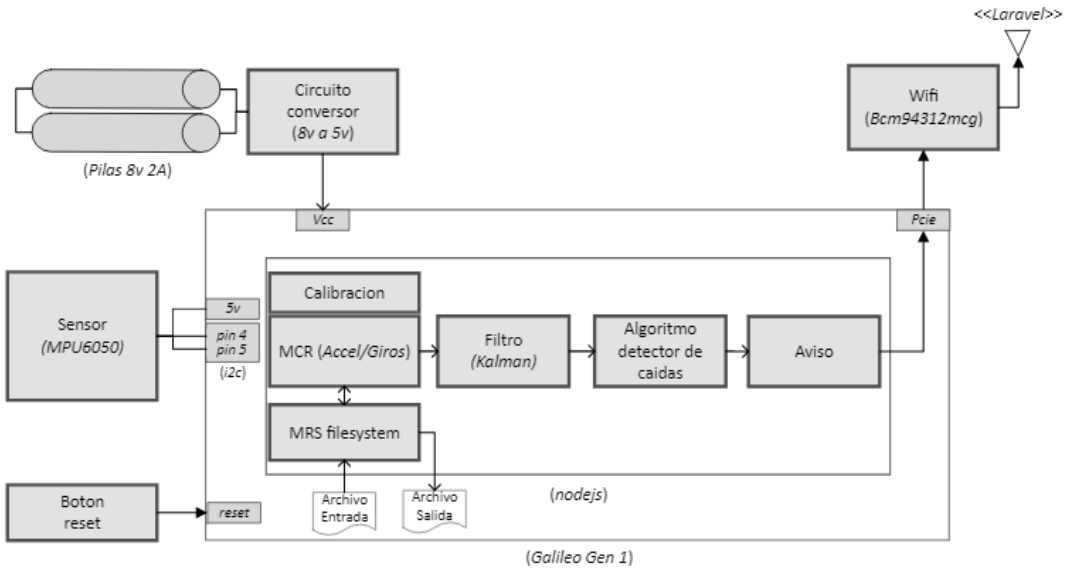


Fig. 4 - Composición del prototipo.

El corazón del prototipo está formado una placa Intel Galileo Gen I con un sistema operativo Linux Yocto. Para medir los movimientos se utiliza el integrado MPU6050, que cuenta con un acelerómetro con giroscopio. Para lograr la autonomía se utiliza una batería especial de 8v de Litio, pero debido a que la placa Galileo necesita sólo 5 volts de alimentación, se desarrolló un circuito conversor de voltaje. Además se le adicionó un botón pulsador como reset para poder reiniciar el sistema operativo en casos de necesidad. También se le incorporó una conexión por Wifi para que pueda establecer la comunicación con el servidor en la nube, emitiendo las alertas antes mencionadas.
En el gráfico (Fig. 4) se muestran los componentes de software más importante utilizados en el sistema de detección de caídas, desarrollado en lenguaje node.js. Por un lado el módulo de calibración es el encargado de realizar los ajustes iniciales del sensor MPU6050. Por otra parte el módulo del MCR es el encargado de leer los valores obtenidos por el sensor, luego el filtro de Kalman nivela el ruido de dichos datos de la medición y estos son utilizados por el algoritmo de caída, que determina si ocurrió o no una caída. En caso de que esto suceda, emite un mensaje de alerta al servidor de servicios web en la nube (desarrollado en *Laravel*[11]). Por otro lado el módulo "*MRS filesystem*" tiene una conexión configurable, utilizada para guardar las mediciones del sensor en archivos al ser requerida. Esto permitió la repetición de las pruebas sin la necesidad de volver a realizar pruebas de caída, permitiendo leer los datos sensados desde los archivos y enviarlos al algoritmo detector de caídas, pudiendo así realizar los ajustes necesarios.
A continuación se explicarán con mayor detalle los incrementos en las funcionalidades en los distintos aspectos del sistema embebido:

a. Incremento de funcionalidades de la parte física del sistema embebido.

La primera versión del prototipo (Fig.) se construyó sobre el banco de trabajo en el laboratorio, donde se conectó el acelerómetro a la placa Galileo. Para lo cual se utilizó una placa de desarrollo tipo *protoboard* y para la conexión se utilizaron cables unipolares.

[11] Framework de código abierto para desarrollar aplicaciones de servicios WEB con PHP

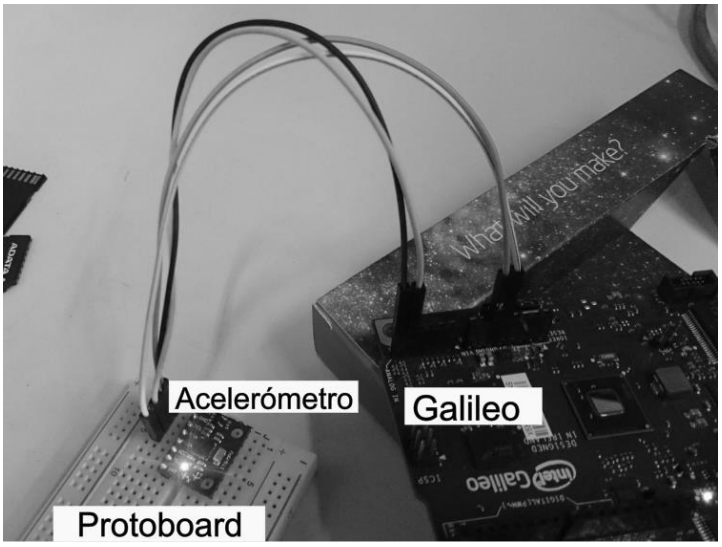


Fig. 5 - Prototipo versión inicial.

Las otras conexiones requeridas por el sistema Galileo, son el conector de alimentación al transformador, cable serie para iniciar la sesión en el dispositivo y cable del tipo de red (*ethernet*) para habilitar la consola del intérprete de comandos del sistema operativo Yocto. Con esta primera versión se logró iniciar con el desarrollo del software leyendo los primeros valores del acelerómetro.
Para poder llevar a cabo las pruebas de caídas, requirió eliminar el cableado del prototipo con que lo fijaba al banco de trabajo (Fig. 6). En la siguiente versión se lograron integrar el acelerómetro y conexión de red inalámbrica (por Wifi).

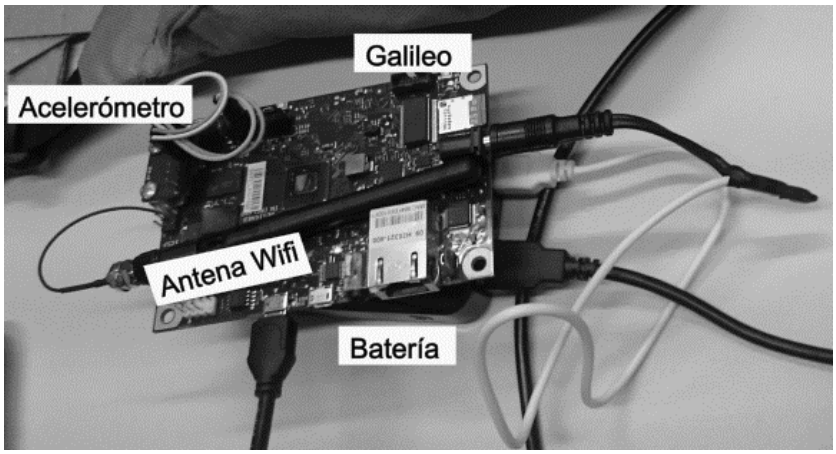


Fig. 6 - Prototipo versión integrada.

La versión final del prototipo puede verse en la (Fig.) como es llevada por una persona en el cinturón. El gabinete grande, que se sujeta al cinturón, lleva en el exterior la antena de conexión inalámbrica por Wifi y la nueva batería con mayor amperaje. El mini gabinete en la parte superior contiene el acelerómetro. En la parte interna del gabinete se encuentra la placa Intel® Galileo, un conversor de voltaje y las conexiones a los elementos antes mencionados. Puede verse en la Fig. 7 un interruptor, que es utilizado para seleccionar el tipo de energía que emplea el prototipo mejorado (batería o transformador).

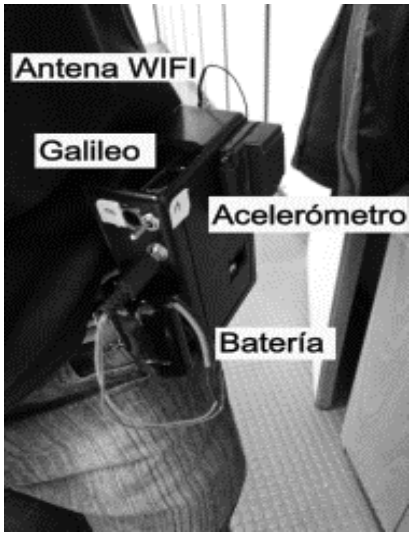


Fig. 7 - Prototipo versión final.

b. Incremento de funcionalidades de los programas que se ejecutan en el sistema embebido.

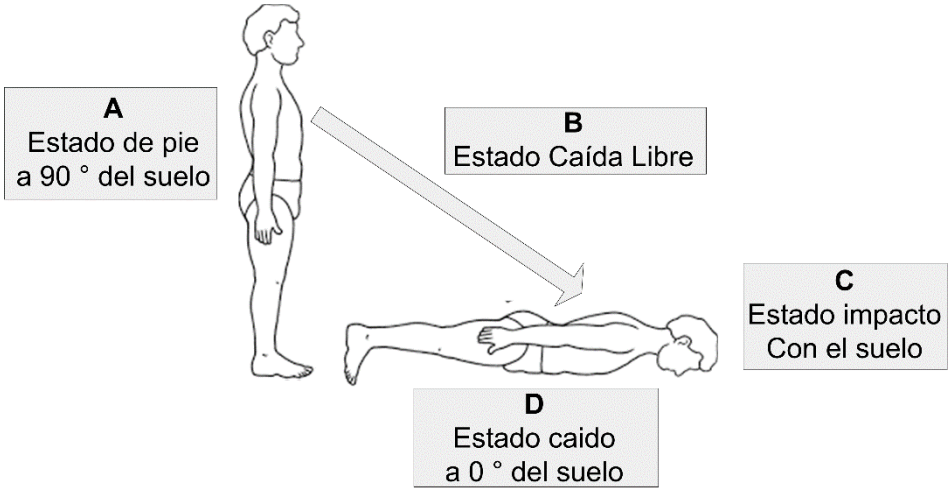
El software fue evolucionando debido a las pruebas realizadas, quedando finalmente con las mejoras en la calibración, el algoritmo de caídas realizado y el envío de datos para las comunicaciones.

- **Mejora del Prototipo**

El proceso para la obtención del algoritmo y su correcto funcionamiento fue bastante complejo, debido a distintos factores que fueron mencionados en [20] [21], tales como problemas de performance en el hardware y del software utilizado. Uno de los percances presentados consistió en obtener la adecuada calibración del sensor MPU6050 para su correcto funcionamiento. Finalmente se ha podido realizar por una única vez haciendo una calibración en todos sus ejes, mediante el desarrollo de un programa en Node.js que calcula las desviaciones de fábrica que presenta cada eje del acelerómetro, que luego son corregidos durante la ejecución del detector de caídas. El programa en Node.js que fue desarrollado para la calibración del MPU6050 se encuentra almacenado en el enlace del repositorio de GIT (VER ANEXO D).
Además el programa detector de caídas fue desarrollado utilizando el lenguaje de programación Node.js, que con sus limitaciones generó distintas dificultades en el software que debieron ser salvadas según se menciona en [12].

- **Algoritmo de caídas**

La versión resultante del algoritmo de caídas contempla los distintos estados que sufre un cuerpo en la caída [20]. Esto se puede apreciar en la Fig. 8.



.

Fig. 8 - Etapas de una caída.

En consecuencia, para poder realizar un análisis exhaustivo de como ocurre una caída y poder así obtener un patrón que permita reconocer los cambios de aceleración y de ubicación que sufre el cuerpo humano durante dicho suceso, fue necesario que el algoritmo detector de caídas, se componga de varias fases. Además de establecer sus umbrales de aceleración y tiempo delimitando cada una de ellas.
El código del algoritmo de caída presenta las siguientes fases:

- Inicialización del algoritmo
- Detección de Caída Libre
- Detección de Impacto
- Espera para el Reposo Final
- Detección de Reposo final
- Detección de recuperación de la persona.

Las mejoras al algoritmo con respecto a las mostradas en la versión inicial (Fig. 8), consistieron en la realización de una espera de varios segundos antes de evaluar la fase “Detección de Reposo Final”, y fundamentalmente detectar automáticamente la reincorporación de la persona caída una vez que haya sufrido el accidente. Para esto último se evalúa constantemente si el anciano consigue ponerse de pie. En caso de que no lo consiga, se envía un mensaje crítico al dispositivo móvil de la persona que tiene a cargo al adulto.
La figura Fig. muestra la función principal del algoritmo. Esta función es invocada cada vez que se lee el acelerómetro, recibiendo los valores sensados. Su objetivo principal consiste en determinar en qué fase de la caída se encuentra la persona de acuerdo a las mediciones capturadas.

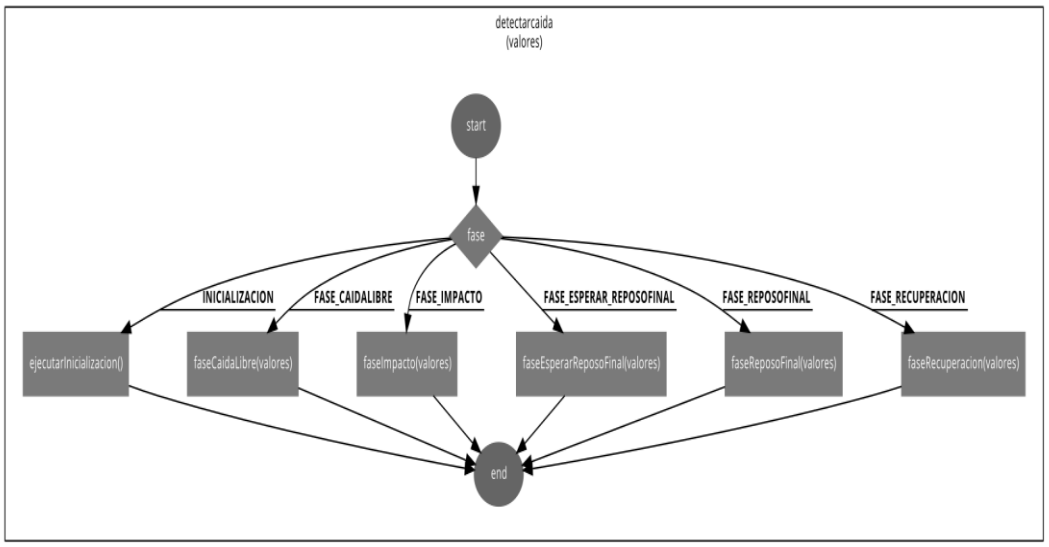


Fig. 9 - Fases del algoritmo detector de caídas.

La primera fase en ejecutarse es la fase de inicialización del algoritmo que se muestra en la Fig. 2

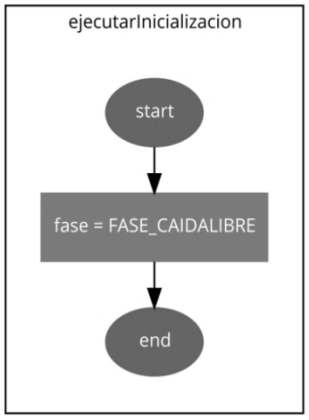


Fig. 10 - Fase de inicialización del algoritmo.

Una vez inicializado el algoritmo se procede a evaluar la detección de la fase Caída libre de la siguiente forma (Fig. 11).

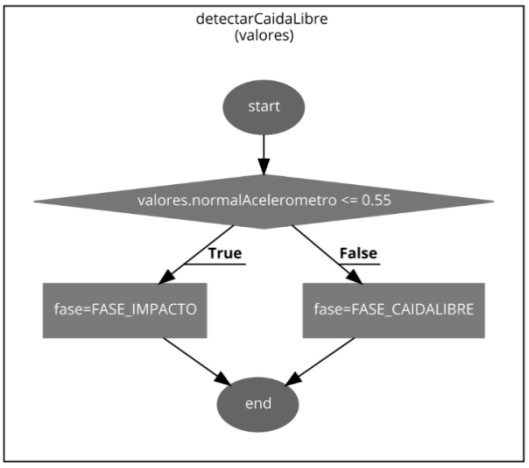


Fig. 3 - Fase de detección de caída libre.

Si algún valor de la aceleración supera el umbral 0.55 G, entonces quiere decir que se detectó una caída libre y se pasa a evaluar la fase de impacto como se muestra en la Fig. 4.

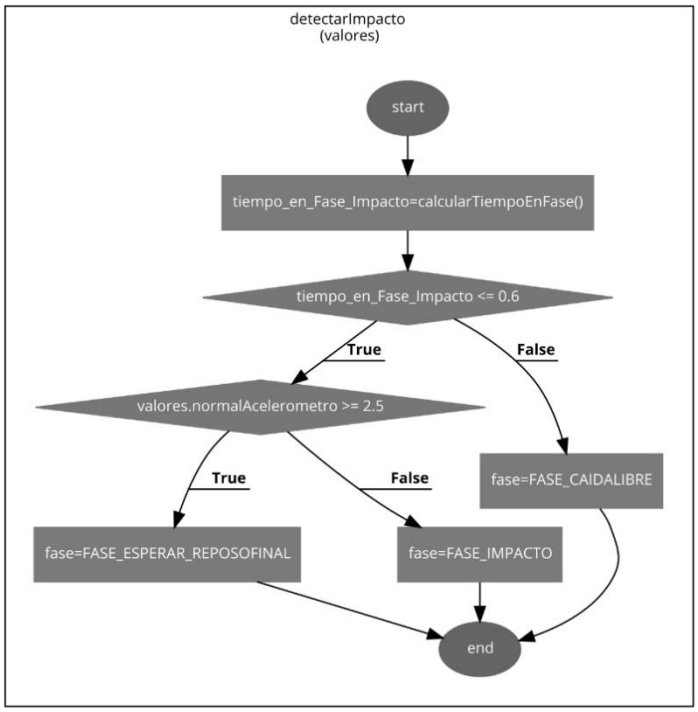


Fig. 52 - Fase de detección de impacto.

Como se observa en la Fig. 6, durante la evaluación de impacto se tiene en cuenta la cantidad de tiempo en que se está analizando dicha fase. Si dentro de ese intervalo de tiempo se detecta una aceleración mayor a 2.5 G, entonces se produjo un impacto y se pasa a la siguiente fase (de espera de la fase reposo final) Fig. 13

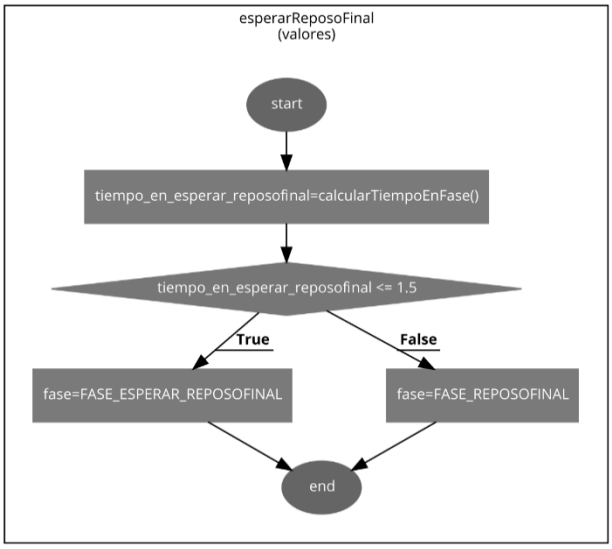


Fig. 13 - Fase de espera reposo final.

La fase de espera posee una interrupción de 1.5 segundos, para pasar a la detección de la fase de reposos final, que se muestra en la Fig. 14.

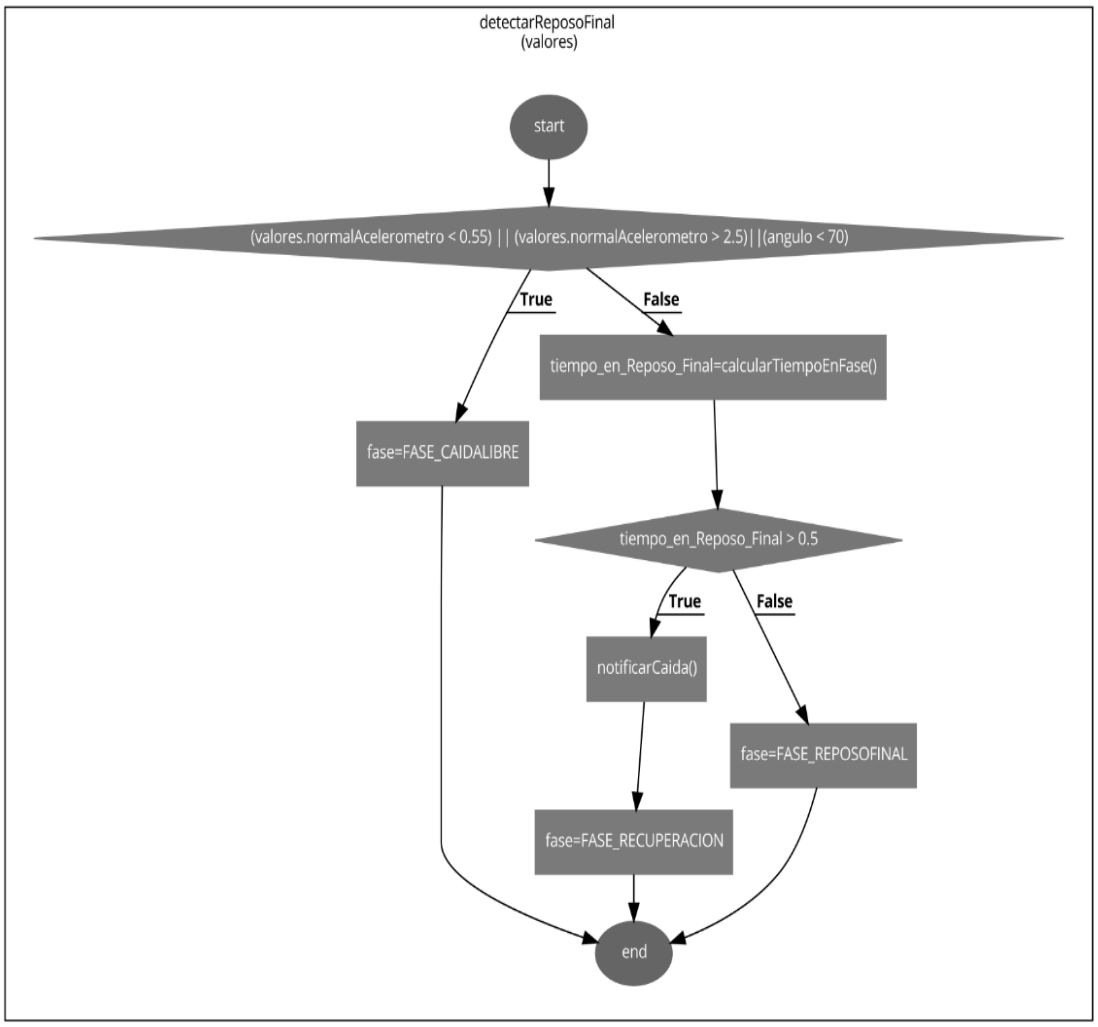


Fig. 7 - Fase de detección de reposo final.

En la fase de reposo final se evalúa si la persona se encuentra en posición horizontal, de acuerdo al ángulo que forma su cuerpo con respecto a la posición inicial y el suelo. Además se controla que la aceleración se encuentre dentro de un rango prestablecido. Al cumplirse estas condiciones se determina que se produjo una caída y se envía un alerta al dispositivo móvil del responsable. Posteriormente se pasa a la siguiente fase, la detección de la recuperación del anciano, que se muestra (Fig. 8).

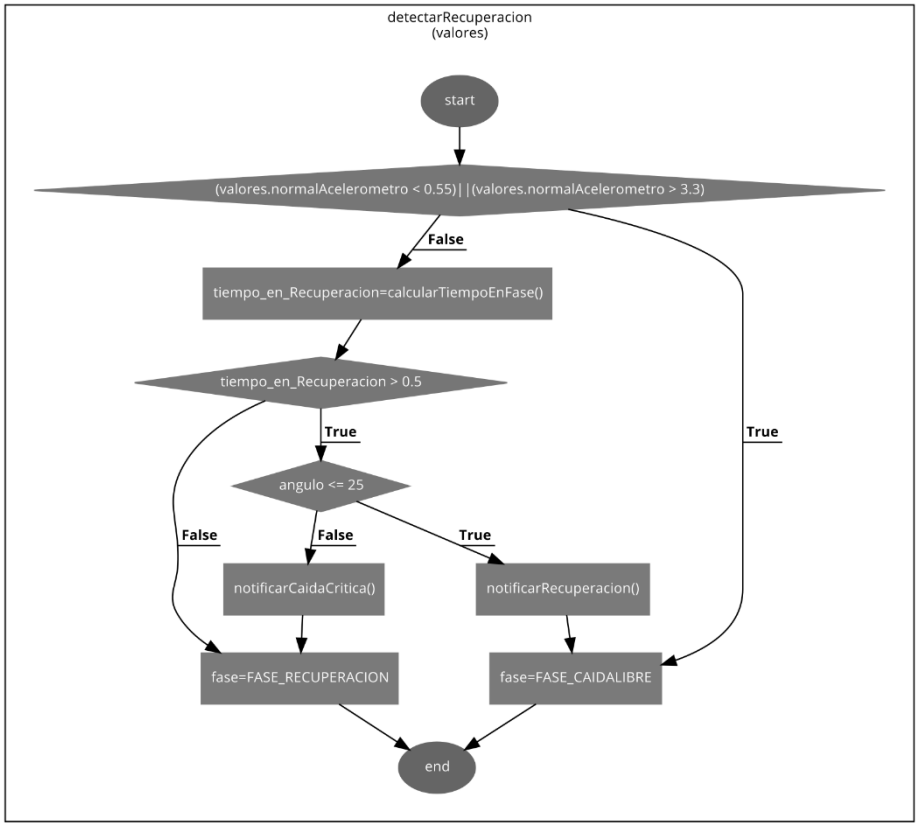


Fig. 9 - Fase de detección de recuperación de la persona.

En la fase de recuperación se determina, evaluando el ángulo y la aceleración, si luego de haberse caído la persona logra ponerse de pie. Si esto se cumple se le envía una notificación al móvil del responsable y, si luego de un tiempo no lo consigue, se le envía otro mensaje de alerta de caída crítica.
Por lo tanto el algoritmo de caída puede enviar tres alertas a la persona responsable del anciano.

- Cuando se detecte una caída
- Si luego de un tiempo sigue caído y no logra reincorporarse.
- Si consigue ponerse de pie.

2.5.1.15 **Evaluación continua de mejoras**.

El sistema de detección de caídas está compuesto por distintos programas, donde cada uno de ellos realiza una funcionalidad especifica. En la Fig. , se puede apreciar el diagrama de componentes donde se muestran las tres aplicaciones que conforman el sistema: Detector de Caídas en Sistema Embebido, Servidor en la nube y Aplicación Android.

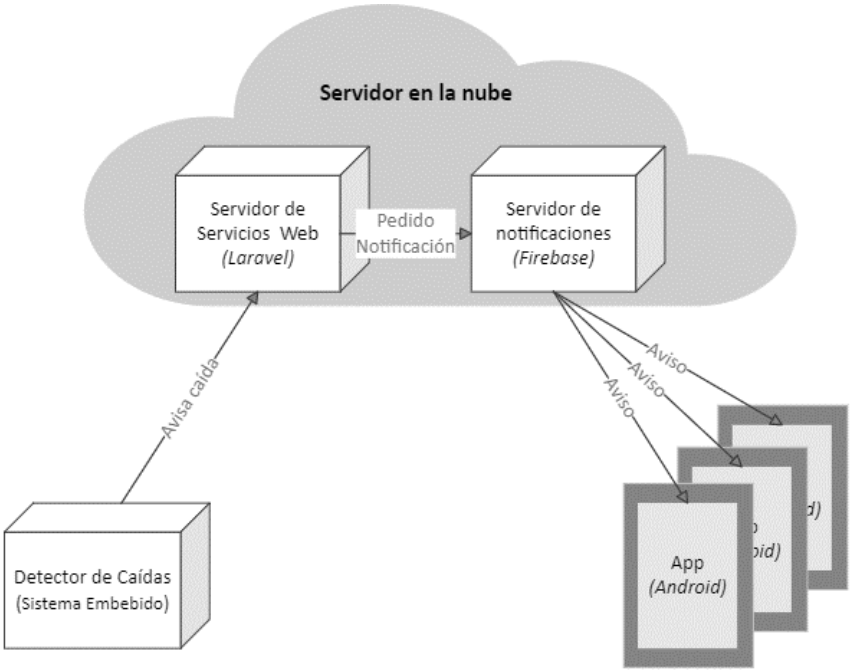


Fig. 16 - Componentes del sistema detector de caídas.

- **Detector de caídas**

Se detalló en la sección anterior.

- **Servidor en la nube**

Se encuentra conformado por un Servidor de Servicios Web y un Servidor de Notificaciones. Su funcionalidad consiste en recibir las alertas de caídas detectadas por los sistemas embebidos y en función a la información almacenada en su base de datos identifica la persona accidentada, para poder luego enviar notificaciones de alertas a los cuidadores que se registraron al sistema a través de la aplicación Android.
Para que este programa pueda intercambiar datos con la aplicación del Sistema Embebido y la Aplicación Android, dicha aplicación debía estar accesible vía internet. Para ello fue necesario implementar el programa servidor en un servicio de alojamiento de Aplicaciones Web, llamado Hostinger [22].
El Servidor de Servicios Web fue construido utilizando el Framework de PHP Laravel [23]. Esta herramienta permite desarrollar aplicaciones con funcionalidades que sean ejecutadas remotamente en el servidor, ofreciéndolas como servicios web. Para poder otorgar diferentes funcionalidades a los distintos usuarios del sistema, fue necesario implementar una base de datos con la información que necesita el sistema para poder funcionar correctamente. Laravel utiliza indirectamente SQL para trabajar con base de datos. Por medio de esta herramienta se almacena en archivos información que permiten identificar a los usuarios que utilizan determinados dispositivos embebidos, como así también a las personas responsables de las mismas, que se deberán ser notificadas ante una caída. Como se muestra en la (Fig. 6), la aplicación servidor desarrollada con Laravel, utiliza el servicio de notificaciones que ofrece Firebase [24] [25], para emitir los mensajes de avisos de caídas a la aplicación desarrollada para Android. Esta última se estará ejecutando en el dispositivo móvil de la persona que se encontrará asociado al usuario en la base de datos del servidor. En consecuencia a continuación se describen con mayor detalle el funcionamiento del Servidor de Servicios Web y el Servidor de Notificaciones.

a. Servidor de servicios web

La aplicación del Servidor de Servicios Web utiliza la arquitectura REST [26] [27], para implementar APIS que ofrecen distintas funciones. Estas pueden ser invocadas por el detector de caídas y la aplicación Android. Para realizar las llamadas a las funcionalidades el sistema

embebido y dispositivo móvil empleando peticiones “POST” y “GET”, pertenecientes al protocolo REST.
En la Fig. 10 se muestran las distintas capas que conforman la aplicación que se ejecuta en el servidor desarrollado con Laravel.

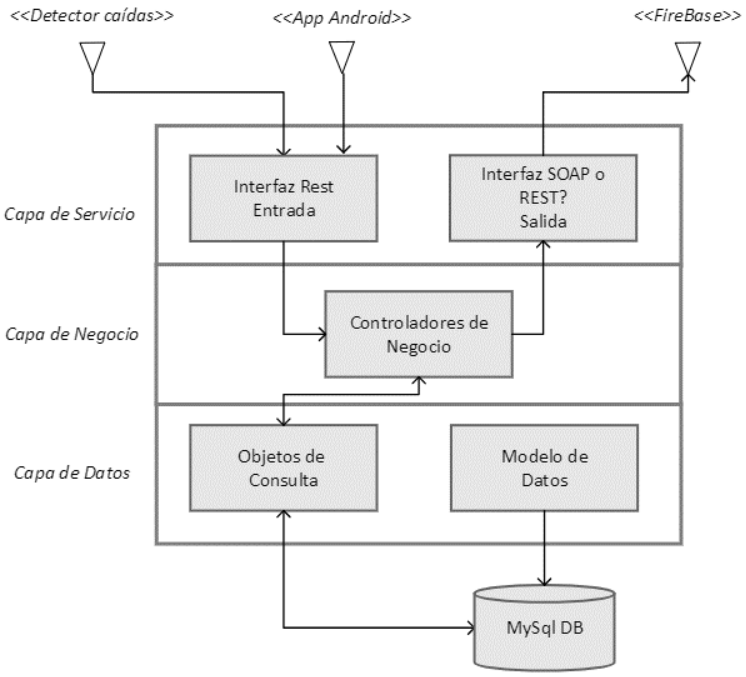


Fig. 11 - Aplicación del servidor de servicios web.

En la capa de servicio se encuentran el módulo con las interfaces REST, responsables de recibir las peticiones emitidas por el detector de caída y la aplicación Android. Además se encuentra la interfaz de salida, empleada para enviar alertas a la aplicación Android por medio de Firebase. Por otra parte, la capa de negocio contiene los módulos que realizan la lógica operacional, dependiendo de la petición REST que se haya recibido. La Tabla 5 muestra algunas de las funcionalidades que pueden realizar los módulos de negocio.

Tabla 6 – Funcionalidades del servidor de servicios web de según el tipo de petición REST recibida.

Forma de petición	Funcionalidad
mobile/send_notification	Utilizada por el sistema embebido para notificar una caída.
mobile/register	Utilizada para registrar un usuario en el sistema
mobile/login	Utilizada para que un usuario pueda acceder a la aplicación Android instalada en su dispositivo móvil
mobile/add_elder	Utilizada para registrar una anciano en el sistema y asociarlo a un usuario
mobile/delete_elder	Utilizada para borrar un anciano del sistema
mobile/update	Utilizada para actualizar los datos de un usuario en el sistema
mobile/list_all	Utilizada para mostrar todos los ancianos asociadas a un usuario

Para poder llevar a cabo sus funcionalidades, los módulos de la capa de negocio utilizan las bases de datos de MySQL, que se encuentran modelada dentro de la capa de Datos. De esta forma se mantiene información persistente en el sistema, útil durante su ejecución. Por esta

razón, a continuación se muestra un diagrama de entidad relación (DER) como el diseño de la base de datos utilizada en el Servidor de Servicios Web.

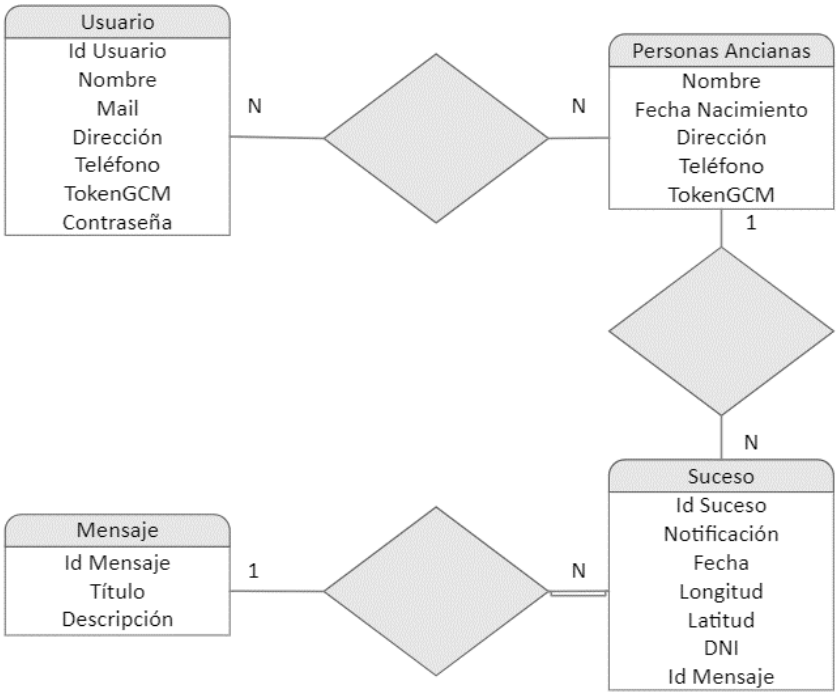


Fig. 12- DER del servidor de servicios web.

En la Fig. 13 se pueden apreciar las distintas entidades que conforman el DER del servidor. La tabla “Personas Ancianas” almacena los datos de los adultos mayores que se encuentran utilizando el dispositivo detector de caídas y están registrados en el sistema. Cada adulto mayor es identificado por un código único, denominado “Token_se”, debido a que se utiliza la dirección MAC de la placa de red de Wifi del dispositivo de detección de caída que la persona usa diariamente. Por otra parte la tabla “Usuario” contiene los datos de los usuarios de la aplicación Android, que son las personas que tienen a cargo al adulto mayor, y son quienes van a ser notificadas cuando se detecte la caída de la persona monitoreada que esta asociada a la misma. Cabe destacar que cada usuario también tiene asociado un número único, denominado TokenGCM, el cual identifica al dispositivo móvil a donde se van a enviar las notificaciones de alerta. Al utilizar el servicio de notificaciones que ofrece Firebase, el TokenGCM es otorgado por este a través de un certificado digital. El cual es registrado y actualizado en la Base de datos del servidor de Servicios Web cada vez que el usuario ingresa a la aplicación de Android con sus datos en algún dispositivo móvil. La secuencia de este proceso se muestra en la Fig. 14
Es importante mencionar, que un usuario puede estar a cargo de varios adultos monitoreados al mismo tiempo, por lo que se tuvo en cuenta esta necesidad al realizar el diseño del sistema.

b. Servidor de Notificaciones

El servidor de notificaciones está representado por el servicio que ofrece la herramienta de Google Firebase para emitir notificaciones a los dispositivos móviles Android identificados mediante un Token, como se describió en el apartado anterior.
El servidor de notificaciones es utilizado únicamente para enviar mensajes de alertas de caídas a los dispositivos móviles que tengan asociado al adulto mayor en la base de datos del Servidor de Servicios Web. Para las restantes transferencia datos hacia la aplicación de Android se envían directamente desde el servidor de servicios web empleando mensajes REST.

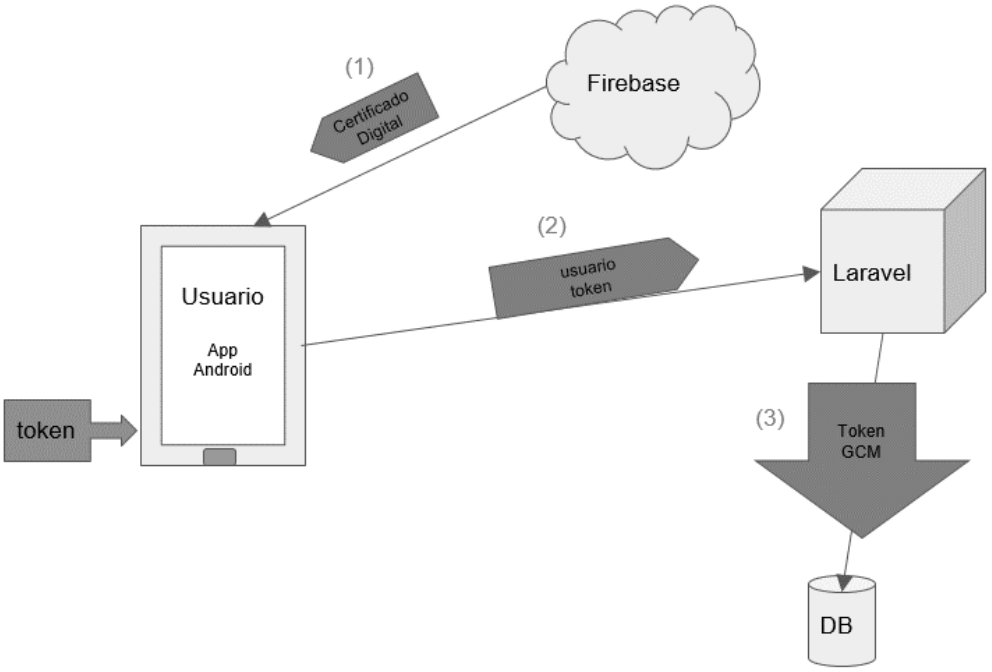


Fig. 19 - Secuencia del proceso de registración del TokenGCM en la base de datos del servidor de servicios web.

En el siguiente gráfico se muestra la secuencia del proceso de notificación de alertas de caídas, desde su detección en el sistema embebido hasta la recepción del alerta en la aplicación Android.

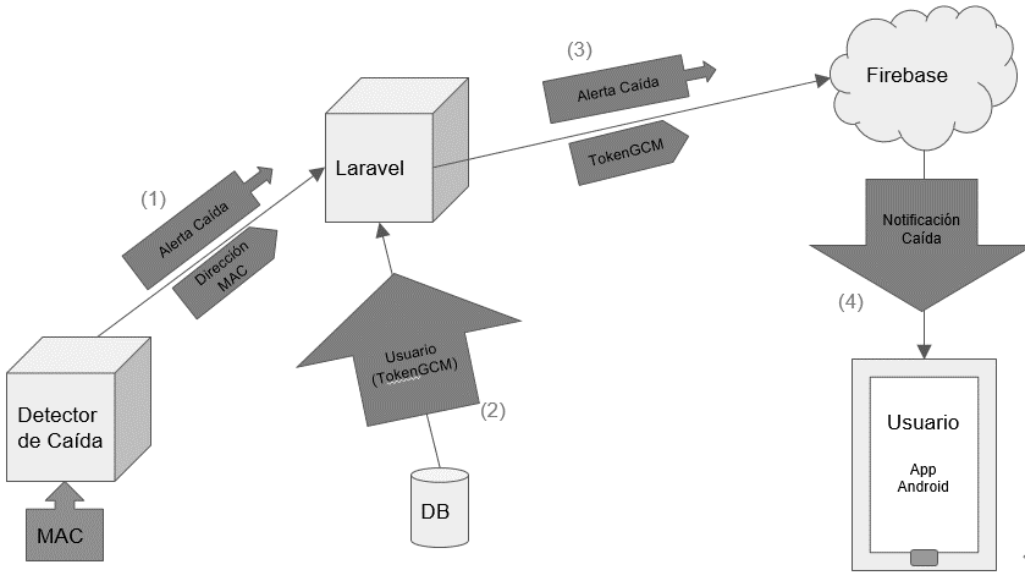


Fig. 20 - Secuencia del proceso de notificación de caídas.

- **Aplicación Android**

Programa desarrollado para funcionar en versiones Android superiores a 4.2, conocida con el nombre de Jelly Bean. Esta aplicación deberá ser instalada en los dispositivos móviles de las personas que se encuentran a cargo del adulto mayor, por ejemplo familiares. Por medio de esta aplicación, el usuario podrá realizar las siguientes acciones en el Sistema de Detección de Caídas:

- Registrarse en el sistema

- Registrar los datos de los ancianos que lleven los dispositivos detectores de caídas que desea monitorear
- Identificar a un abuelo mediante un Código QR[12], que representa la dirección MAC perteneciente al dispositivo detector de caídas.
- Modificar sus datos y los abuelos que monitorea
- Recibir notificaciones de alertas en tiempo real, en el momento en que un abuelo que monitorea el usuario, sufra una caída.
- Recibir una notificación de alerta cuando pasado un determinado tiempo un abuelo no se haya recuperado después haberse accidentado.
- Recibir una notificación de alerta cuando un abuelo se recupere, poniéndose de pie nuevamente, después haberse accidentado.

Asociar a un abuelo un listado de contactos de la agenda de su dispositivo móvil, para que al recibir una notificación de caída pueda comunicarse rápidamente con ellos.
La aplicación desarrollada para el sistema operativo Android, se la ha denominado con el nombre de *AbuCaida*. Por el momento no se encuentra disponible en la tienda online de Google Play Store[13] de Android. Por ese motivo la única forma de instalar este aplicativo, es descargando el ejecutable APK[14] desde un repositorio de Git, cuyo ubicación web se encuentra especificado en el ANEXO D.
Para poder ejecutar el APK correctamente en el dispositivo móvil, será necesario activar la opción “instalar aplicaciones de fuentes desconocidas”, que se encuentra en la configuración de Android. Esto se debe a que Android no reconoce dicha aplicación por defecto, al haber no sido descargada desde su tienda oficial. Una vez instalada el APK en el dispositivo móvil, la aplicación podrá ser accedida desde el acceso directo de AbuCaida, que es representado por la imagen de la Fig. 15.

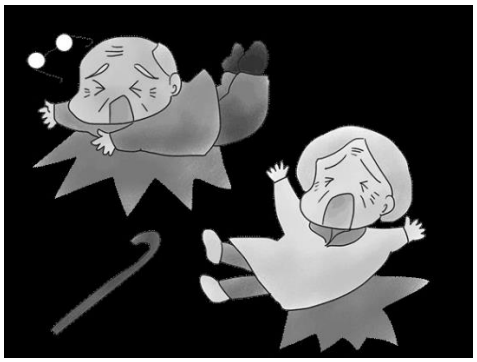


Fig. 16 - Imagen del acceso directo de la aplicación de AbuCaida.

a. Diagrama de Casos de Usos de la aplicación Android

En el siguiente diagrama se muestra el Modelo de Casos de Uso perteneciente a la aplicación Android desarrollado para el sistema detector de caídas.

[12] Un código QR es la evolución del código de barras

[13] Google *Play Store* es la tienda oficial de aplicaciones para Android.

[14] Es un paquete ejecutable para el sistema operativo Android

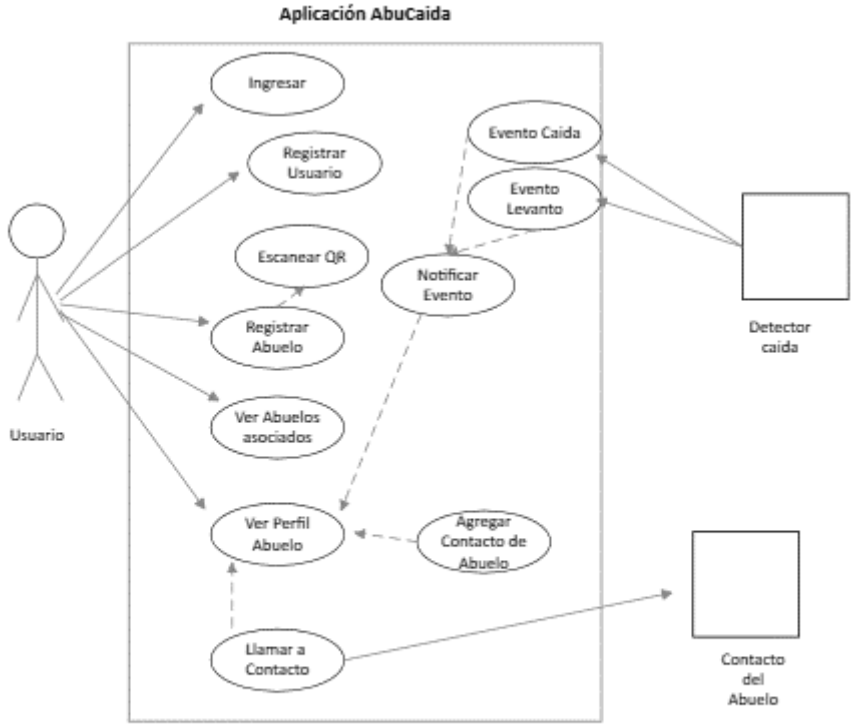


Fig. 17 Diagrama de Casos de Uso de la aplicación Android.

b. Diagrama de Clases de la aplicación Android

En la (Fig. 18) se muestra el Diagrama de Clases correspondiente a la aplicación AbuCaida.

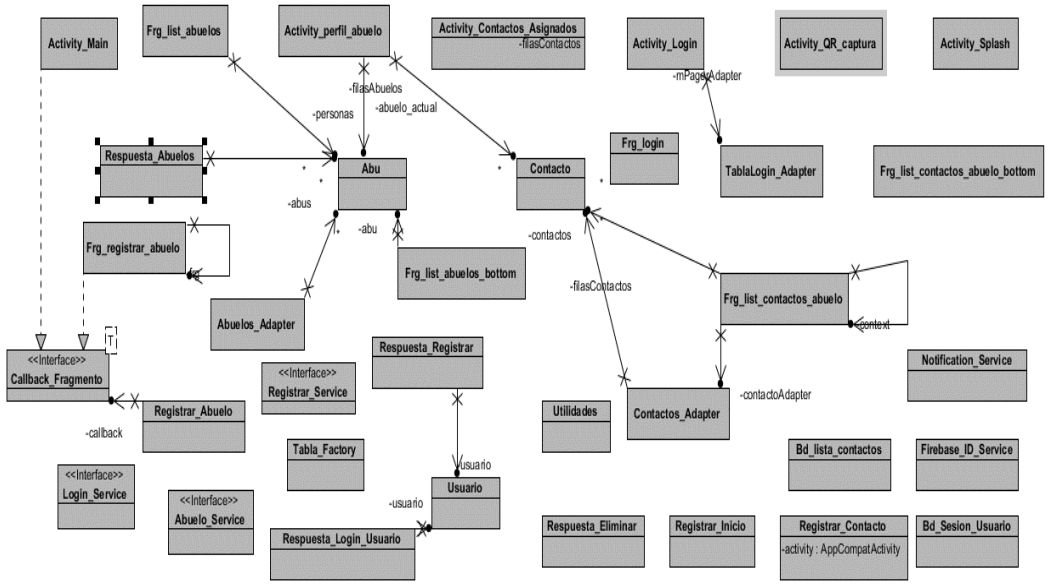


Fig. 19 - Diagrama de Clases de la aplicación Android.

c. Descripción de las distintas Interfaces Gráficas de la Aplicación Android

En este apartado se describen el funcionamiento del programa AbuCaida, conjuntamente con el detalle de la funcionalidad de cada pantalla gráfica que lo conforma.

INTERFAZ LOGIN

Al abrir la aplicación por primera vez, la pantalla inicial que se mostrará corresponderá al Login del usuario, que permite identificar a la persona que utilizará el sistema. Ver Fig. 20

Fig. 21 - Interfaz Login.

Es necesaria la registración previa del usuario en el sistema, a través de la interfaz de "Registración de usuario", para poder realizar el inicio de sesión.

INTERFAZ REGISTRAR USUARIO

Permite a la persona su registración como usuario del sistema detector de caídas, para poder utilizar la aplicación AbuCaida. Dicha pantalla está representada por la Fig. 25.

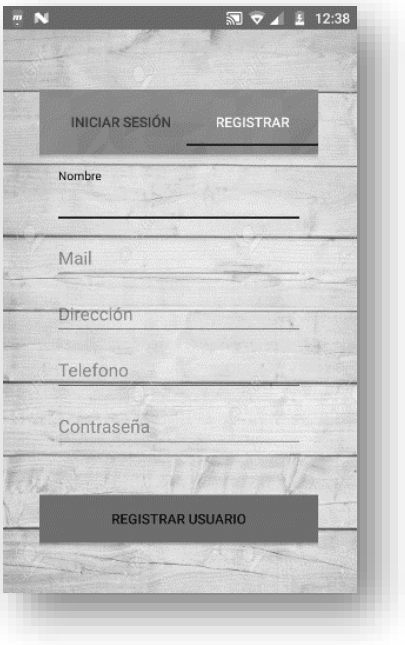


Fig. 22 - Interfaz Registrar Usuario.

INTERFAZ LISTADO DE ABUELOS ASOCIADOS

Una vez que un usuario ingresa al sistema con sus datos, a través de la pantalla de Inicio de Sesión, se mostrará la siguiente interfaz (Fig. 23) , que muestra un listado con los abuelos que están siendo monitoreados por ese usuario. Cuando algunos de esos ancianos sufran una caída, le llegará una notificación de alerta al dispositivo móvil en donde el usuario accedió por última vez a la aplicación AbuCaida.

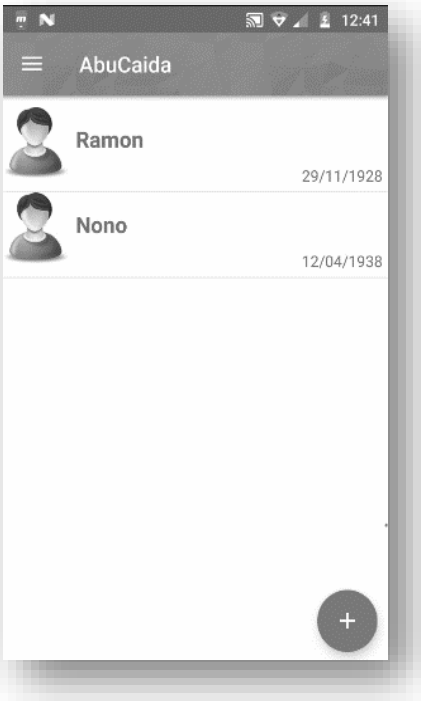


Fig. 24 - Interfaz Listado de Abuelos

MENU LATERAL

La aplicación contiene menú lateral que permite acceder a determinadas opciones, las cuales se muestran a continuación.

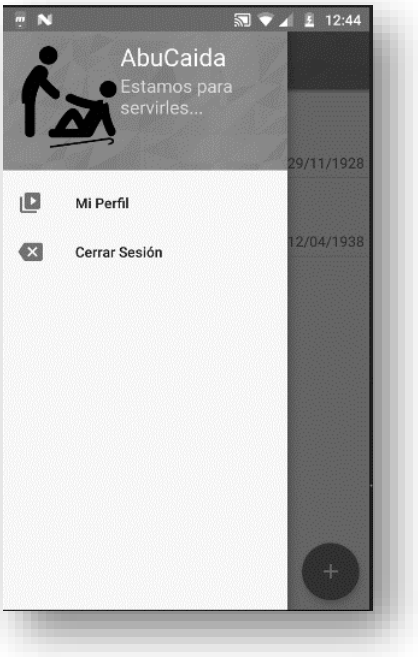


Fig. 25 - Menú lateral

INTERFAZ PERFIL DE USUARIO

Android mostrará la siguiente interfaz cuando el usuario al pulse en la opción de “Mi Perfil” del Menú Lateral (Fig. 26). Desde esta pantalla podrá modificar sus datos, que se encontrarán almacenados en la base de datos del sistema ubicado en el Servidor de Servicios Web.

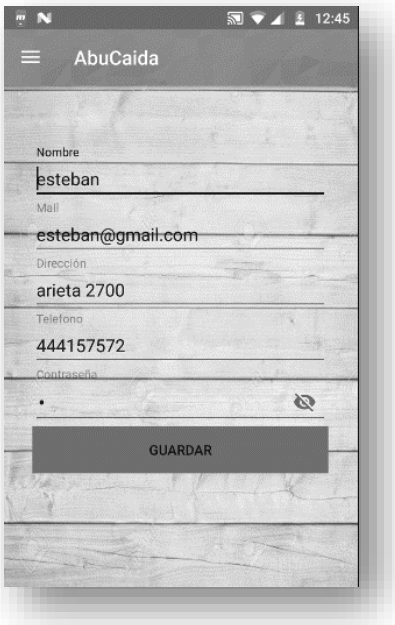


Fig. 26 - Interfaz Perfil Usuario.

INTERFAZ REGISTRAR ABUELO

A través de la siguiente pantalla gráfica, el usuario podrá registrar los datos de un anciano, que utiliza un determinado dispositivo de detección de caídas.

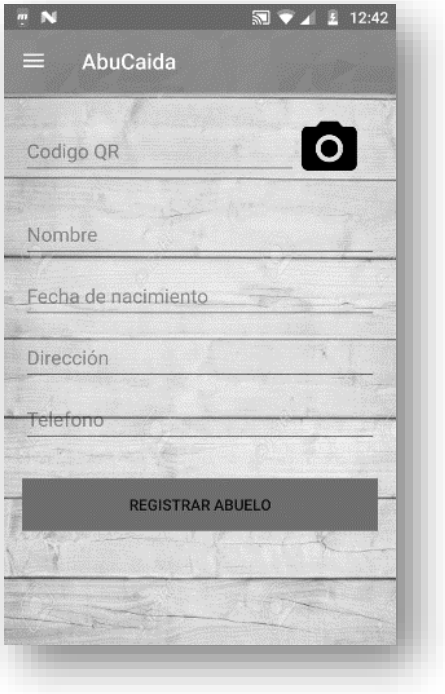


Fig. 27 - Interfaz Registrar Abuelos.

Para poder registrar un abuelo, será necesario ingresar la dirección MAC del dispositivo que utilizará el anciano. Esta dirección podrá ser registrada en forma manual o automáticamente, escaneando el código QR (Impreso en el dispositivo que realiza el sensado de datos). De esta forma el sistema podrá identificar a dicha persona. Cuando el usuario confirme la registración del abuelo, el sistema validará la dirección MAC. En caso de que ya se encuentra asignada a un anciano, la aplicación mostrará un mensaje alertando de dicha situación, otorgándole la posibilidad al usuario de monitorear al abuelo que ya tiene asignada dicha MAC.
Después de efectuarse la confirmación, el sistema, registrará al anciano en su base datos y lo asociará al usuario actual. De esta forma, cuando el abuelo sufra una caída, el sistema sabrá a que usuario notificar. Además, la aplicación Android automáticamente actualizará el listado de los abuelos mostrados en la Fig. 24.

INTERFAZ PERFIL DEL ABUELO

El usuario podrá acceder desde el listado de abuelos al perfil de un determinado anciano (Fig. 24). Para ello deberá pulsar un ítem del listado y seleccionar la opción “Ver Perfil”. Seguidamente se mostrará la siguiente interfaz con los datos del abuelo seleccionado.

Fig. 28- Interfaz Perfil Abuelo con los datos seleccionados por el usuario.

En esta interfaz el usuario podrá ver y modificar la información de un abuelo que se encuentra almacenada en la base de datos. Desde aquí también el usuario podrá asociarle al abuelo a un conjunto de contactos, de la agenda de su dispositivo móvil, con quienes podrá comunicarse rápidamente cuando reciba una notificación de caída de ese anciano. En la Fig. 29 se muestra cómo acceder al listado de contactos de un abuelo.

Fig. 30 - Agregar Contactos.

Es importante mencionar, que por cuestiones de privacidad, la lista de contactos de cada abuelo es guardado en un archivo local del dispositivo móvil. Con lo cual se eliminará el archivo con dicho listado al desinstalar la aplicación.

INTERFAZ LISTADO DE CONTACTOS

A medida que el usuario vaya asociando contactos a un abuelo de su agenda personal, se irán mostrando en un listado en la siguiente interfaz gráfica (Fig. 31).

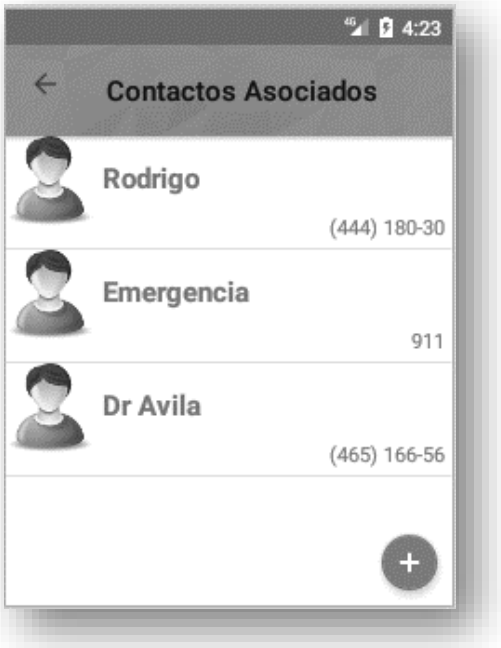


Fig. 32 - Listado de Contactos asociados a un abuelo.

NOTIFICACIÓN DE ALERTA

Cuando un abuelo asociado al usuario sufra una caída, el servidor en la nube le enviará un mensaje de alerta a la aplicación Android alertándola sobre de dicho suceso. Este mensaje se mostrará en la bandeja de notificaciones del Sistema Operativo, ver Fig. 33

Fig. 34 - Notificación de una Caída en el dispositivo móvil con el sistema operativo Android.

Del mismo modo, el usuario podrá recibir otra notificación distinta en la bandeja del S.O, si se detecta que el abuelo no se ha puesto de pie después de un determinado tiempo de haberse caído, como así también cuando se recupere.

Cuando el usuario despliegue cualquiera de estas notificaciones se mostrará una descripción del alerta, con la siguiente información.

- **Tipo de Alerta:** Caida_Detectada, Sigue_Caido, Se_Recupero
- **Breve descripción del alerta**
- **Nombre del abuelo**
- **Fecha y hora del suceso**

En la Fig. 35 se muestra un ejemplo de una notificación de caída desplegada.

Fig. 36 - Notificación de caída desplegada.

Si el usuario abre una de estas notificaciones, pulsándola, automáticamente se mostrará el perfil del abuelo accidentado con sus datos (**¡Error! No se encuentra el origen de la r eferencia.**0). Con lo cual el usuario tendrá la posibilidad de acceder al listado de contactos que tiene asociado ese anciano, para poder comunicarse con ellos. Para acceder al listado se deberá pulsar en la opción “Contactos Asociados”.
Cuando el usuario seleccione un contacto de ese listado, la aplicación mostrará la siguiente opción para poder comunicarse con esa persona.

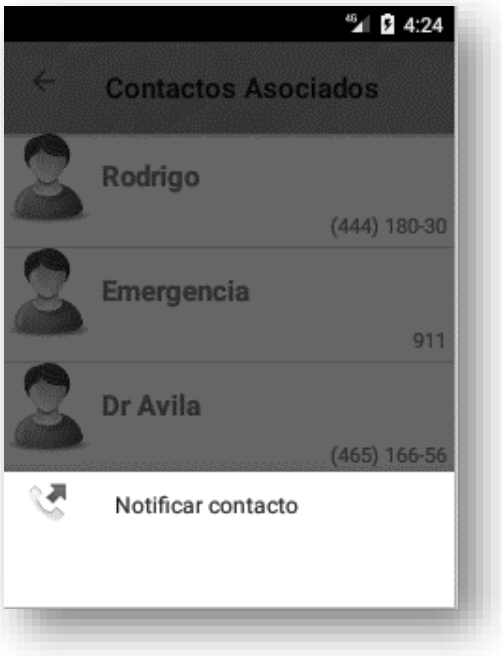


Fig. 37 - Opción para comunicarse con el contacto seleccionado.

2.5.1.16 Realizar pruebas mediante el uso de distintos dispositivos receptores.

Al comenzar la investigación se realizaron pruebas del funcionamiento del sensor MPU6050. Como se describió en apartados anteriores, al principio se crearon programas Arduino que mostraban los valores sensados por acelerómetro y giroscopio en un gráfico de coordenadas, utilizando Processing como se muestra en la Fig. .

Más adelante en la investigación fue necesario migrar a Node.js. Por lo tanto se creó un programa empleando ese lenguaje de programación que almacenaba en archivos CSV las valores sensados por el MPU6050 y además otro programa que mostraba estos valores en distintos gráficos de coordenadas mediante una página web hecha en java script. Por consiguiente, utilizando las herramientas creadas, se realizaron las primeras mediciones de caídas en el laboratorio grabando las mediciones en archivos. El análisis de estos datos no resultó como se esperaba, debido a que presentaban valores inconsistentes, dificultando obtener un patrón claro. Este inconveniente surgió como consecuencia de que el dispositivo embebido, utilizado en ese momento, no poseía un eje de referencia fijo. Además el sensor presentaba problemas de configuración y el dispositivo no utilizaba baterías, sino que se encontraba conectado a la corriente eléctrica durante las mediciones.

Luego de haber corregido los ejes de referencia en el programa de Node.js y adaptado el prototipo para que sea portable e inalámbrico, se procedieron a realizar nuevas pruebas de caídas en el gimnasio de la Universidad. Para ello se utilizó el dispositivo ubicado en la cintura con un cinturón. Mientras se realizaban las pruebas, las mediciones fueron almacenadas en archivos CSV y al mismo tiempo desde una computadora se pudo visualizar las valores sensados en los gráficos de coordenadas de las páginas web.

A partir de la información obtenida en estos archivos, se procedió a realizar el análisis de las mediciones. Posteriormente, se pudieron determinar patrones preliminares de caída y de actividades de la vida diarias de una persona. Permitiendo así elaborar la base del algoritmo. Durante el desarrollo de la versión inicial del programa detector de caída, se hicieron pruebas para comprobar su funcionamiento mediante la reproducción de los datos almacenados previamente en los archivos CSV.

La finalidad de la versión inicial del programa detector era mostrar mensajes de alertas cuando detectara una caída por la consola de una computadora, también cuando la persona haya estado mucho tiempo caído y luego en el momento en que se pusiera de pie.

Gracias estas pruebas se descubrió que el algoritmo que no estaba detectando todos los estados de una caída y era necesario generar nuevas fases para ello. Con lo cual fue necesario modificar la lógica del programa. Luego, de aplicar las mejoras necesarias, se volvió a repetir nuevamente la ejecución del algoritmo comprobando su funcionamiento con la reproducción de los archivos CSV.
Una vez que se comprobó el correcto del funcionamiento del algoritmo reproduciendo los datos de las mediciones en los archivos. Se procedió ejecutar el programa detector mientras una persona usaba el prototipo repitiendo las mismas caídas y actividades realizadas en la captura de los CSV. Como resultado de estas pruebas se pudo observar que no eran adecuados los umbrales y patrones utilizados en ese momento, por más que hayan funcionado adecuadamente con la reproducción de los archivos CSV. Con lo cual se tuvo que rehacer varias veces las pruebas de caídas almacenándolas nuevamente en archivos CSV. Así como también, repetir el análisis de los datos, modificar la sensibilidad del sensor, mejorar el algoritmo, ejecutarlo utilizando la reproducción de los datos que fueron almacenados y finalmente repetir las pruebas mientas una persona usaba el prototipo.
Cuando el programa detector superó satisfactoriamente las pruebas anteriores. Se procedió a probar el dispositivo ejecutando el algoritmo en el gimnasio de la Universidad realizando distintas actividades, controlando por consola si se detectaba correctamente una caída. Luego, cuando se finalizó el desarrolló del servidor en la nube y el programa de Android, se realizaron pruebas con el algoritmo controlando si enviaba correctamente alertas de caída a los teléfonos móviles correspondientes.

2.5.1.17 Realizar prueba de confiablidad de la parte de hardware.

Las pruebas de confiabilidad del hardware fueron llevadas a cabo en las distintas etapas del proyecto, comprobando el funcionamiento del sistema embebido. Para ello se debieron tener en cuenta las limitaciones del hardware utilizado, que fueron presentándose durante el desarrollo del proyecto. Primeramente el tamaño del prototipo resultó bastante considerable, el cual debería ser un dispositivo portable y cómodo de usar por el usuario. Esto es como consciencia de las dimensiones físicas de la placa Intel Galileo. Otro factor que se debió tener cuenta fue el consumo energético que requieren estas placas para poder funcionar. Por ese motivo se emplearon un conjunto de baterías de Litio de 8 Volt, cuya duración de operabilidad en el sistema embebido estando con carga máxima es de aproximadamente dos horas. Además requieren de aproximadamente entre 3 y 5 horas cargarlas completamente utilizando el Cargador Inteligente Imax B6 12v, que fue adquirido para este proyecto. Otro componente que presentó ciertas limitaciones de hardware fue el MPU6050, debido a que el sensor es sensible a los cambios bruscos de temperatura, pudiendo afectar las mediciones. Su hoja de datos [17] especifica que el desvió de las mediciones del acelerómetro es de +/-0.02% por grado centígrado y el giroscopio de +/- 20% por grados/segundos. Por consiguiente para poder solucionar esta dificultad se determinó conveniente aislar el sensor y colocarlo en la parte exterior del prototipo, como se detalló en [20]. Para que de esta forma no se vea afectado por el calor generado por el microprocesador de la placa Galileo. Luego de esta mejora se puedo determinar que son imperceptibles las desviaciones producidas por el cambio de temperatura que pueden afectar al sensor.

2.5.1.18 Probar el desempeño de cada optimización del sistema.

Como se mencionó en apartados anteriores el sistema detector de caídas fue sufriendo distintas mejoras sucesivas, por consiguiente al finalizar cada una de ellas se verificó su correcto funcionamiento, inicialmente en forma independiente y luego en forma integral. Para esto último primero se hicieron distintas pruebas con la repetición de la reproducción de datos de mediciones que habían sido recolectados en archivos CSV y posteriormente mediante mediciones de caída y actividades de la vida diaria en tiempo real en un ambiente controlado. De esta forma se pudo comprobar el correcto funcionamiento del sistema, desde el algoritmo detector de caídas ejecutándose en el sistema embebido, hasta el Servidor en la nube y la Aplicación desarrollada para Android. En el Anexo A se encuentran mencionadas los resultados de las distintas mediciones realizadas a lo largo de toda la investigación, utilizadas para estas pruebas.

2.5.1.19 Prueba del sistema en un escenario real.

Debido a que la carga de la batería permite utilizar el sistema embebido sin interrupción durante aproximadamente dos horas, la realización de las pruebas en un escenario real tuvieron que ser acotadas a ese intervalo de tiempo para poder verificar el funcionamiento del prototipo sin detener en ningún momento la ejecución del programa embebido. Por ese motivo se determinó conveniente realizar este tipo de pruebas utilizando el prototipo durante dos horas en el gimnasio de la Universidad Nacional de La Matanza. En el transcurso de ese tiempo se comprobó el funcionamiento integral del sistema detector de caídas mediante un sujeto de prueba que realizó distintos movimientos, incluyendo la reiteración de todas las caídas y actividades diarias que se habían planificado previamente, que fueron mencionadas en apartados anteriores. En los casos testeados el sistema de detector de caída, generó las alertas correspondientes. La primera en el momento que se produjo una caída, la segunda cuando luego de transcurrido un determinado tiempo de haberse caído no se puso de pie la persona y finalmente un tercer mensaje cuando se reincorporó. Además se comprobaron las distintas funcionalidades desarrolladas que ofrecen la Aplicación Android y el Servidor en la nube.

2.5.1.20 Evaluación de resultados.

Se ha conseguido desarrollar el prototipo de un detector de caídas IoT, que avisa al celular de las personas responsables del individuo monitoreado, cuando sufra un accidente de este tipo. El sistema desarrollado es capaz de detectar distintos eventos de caídas, que fueron mencionados en este informe, pudiendo diferenciar los movimientos rutinarios de las personas con respecto a las caídas que puedan sufrir. Utilizando el concepto de Internet de las Cosas, se ha construido un sistema embebido que se comunica con otros dispositivos haciendo uso del protocolo REST. De esta forma se consigue la comunicación entre este con el servidor en la nube y los teléfonos inteligentes que utilicen la aplicación Android desarrollada para este proyecto. El último software mencionado genera otras alertas en forma automática a las personas que estén asociadas al anciano, registradas en la base de datos del servidor. Las restantes notificaciones son generadas cuando el abuelo sigua postrado en el piso, después de un tiempo de haberse caído, y también en el momento en que logre ponerse de pie. Esta es una gran diferencia con respecto a otros trabajos estudiados, en donde esta situación la realiza manualmente la persona que haya sufrido el accidente a través de la pulsación de un botón anti pánico. Además el programa del celular les ofrece a los responsables del abuelo, la posibilidad de administrar sus datos en el sistema de detección de caída. Permitiendo así registrarlos en sistema identificándolos por la dirección MAC del sistema embebido que esté utilizando el anciano.
A partir de las distintas pruebas realizadas, se pudo determinar que existen determinadas situaciones en que el algoritmo de caída desarrollado puede que no llegue a detectar una caída. Esto se debe a los siguientes motivos.

- Si se detecta una caída y luego la persona se levanta y cae nuevamente muy rápidamente, entonces la segunda caída puede que no se llegue a detectar. Debido a que no transcurrió el tiempo suficiente (1,5 seg) para detectar la fase de reposo final.

Si el umbral de la fase de Impacto es muy grande, mayor a 3G, puede que no llegue a detectar una caída en donde el anciano aminore su velocidad al caer, por ejemplo atajándose con las manos. En cambio si el umbral de Impacto es muy bajo, menor a 2G, puede ser que se generen falsos positivos, por ejemplo al acostarse en una cama muy rápidamente, ya que fácilmente superará el umbral de 2G y al estar a 90 grados detecte una falsa caída. Por ese motivo se consideró adecuado dejar a dicho umbral en 2.5G. En consecuencia pueden existir situaciones en que el anciano aminore demasiado su velocidad de caída y el algoritmo no consiga detectarla, debido a que al producirse el impacto la aceleración no logre superar el umbral de impacto de 2.5G.

2.5.1.21 Informe de problemas y soluciones.

En el transcurso del proyecto de investigación se presentaron distintos problemas que no habían sido previstos en la planificación inicial de la investigación. Cuestión que produjo ciertos atrasos en la finalización de determinadas tareas, afectando de esta forma el cumplimiento de determinadas metas del proyecto. El principal factor de dificultad resultó entorno al hardware y herramientas de software escogidos para realizar este proyecto. Como se explicó en apartados anteriores, esta investigación se llevó a cabo utilizando las placas Intel Galileo Generación I y el sensor MPU6050.

Inicialmente se comenzó a desarrollar el software embebido en las placas Galileo Gen I empleando el lenguaje Wiring, pero estas al emular el comportamiento de las Arduino, presentaron varias diferencias de conectividad y de performance con el sensor MPU6050 con respecto a la empleada con Arduino UNO. Estos contrastes se debieron principalmente a la frecuencia clock del bus I2C existente entre ambas placas. Así como también entre sus bibliotecas Wire.h y la variable TWBR, que únicamente existe para el código de Arduino UNO. Por ese motivo se determinó que resultaba conveniente continuar desarrollando el programa utilizando el lenguaje Node.js, siendo uno de los lenguajes base disponibles para las placas Galileo. Sin embargo, posteriormente, se descubrieron distintos problemas durante el análisis de las primeras mediciones de caídas realizadas [21], con el software que se había desarrollado en Node.js. La razón de dichos percances se debió a problemas de performance que poseen los programas construidos con este lenguaje. Los inconvenientes presentados con Node.js durante la investigación fueron explicados detalladamente en [20]. Una de las causas principales de estos inconvenientes consistió en que al tiempo de muestreo de los datos del sensor MPU6050 no era suficiente para poder determinar patrones de caídas a partir del análisis de las mediciones. Con lo cual fue necesario buscar una solución, paralelizando la lectura del sensor desde el programa de Node.js. La particularidad de este lenguaje es que los programas se ejecutan en único hilo de ejecución. Sin embargo existen bibliotecas que permiten paralelizar el código de los programas desarrollados. Para conseguir el paralelismo, primeramente se hicieron pruebas con la biblioteca webworkers-thread [28], que permite crear hilos en Node.js. Sin embargo los resultados de rendimiento obtenidos en el tiempo de muestreo no resultaron ser los óptimos, y se alejaban demasiado de lo esperado. Finalmente para resolver este problema se utilizó la biblioteca child_process [29], que permite realizar el paralelismo en este lenguaje utilizando procesos. Esta biblioteca nos permitió obtener los tiempos de muestreo adecuados para poder determinar patrones de caídas a partir de los datos muestreados. A pesar de ello se pudo advertir que los programas que realizan paralelismo con node.js ejecutándose en las placas Intel Galileo Generación I, consumen demasiados recursos de hardware. Debido a que principalmente utilizan un gran porcentaje de la CPU y de la memoria durante su ejecución. La razón de este rendimiento es que Node.js si bien el lenguaje poseía las funciones que se necesitaban, no está preparado para hacer este tipo de procesamiento en tiempo real en sistemas embebidos, ya que necesita una cantidad importante de recursos hardware disponibles.

No obstante se decidió conveniente desarrollar el programa detector de caídas en la placa Intel Galileo Gen I utilizando el lenguaje Node.js junto a Wiring, dado a que puede funcionar correctamente en esta plataforma. A causa, de que las placas Intel Galileo se asemejan a una computadora personal de bajo rendimiento, en cuanto a cantidad de recursos disponibles.

Otro de los inconvenientes presentados durante la investigación consistió en el armado del prototipo de dispositivo hardware del sistema embebido que detecta la caída de la persona mayor. Dichos problemas ya fueron mencionados en apartados anteriores. Principalmente se debieron al tamaño del dispositivo resultante y su alto consumo de energía al utilizar las placas Intel Galileo Generación I. Como se ve en la Fig. el tamaño resultante del prototipo es 10*13*20 cm, y debe ser colocado en la cintura de la persona anciana para su correcto funcionamiento. Situación que puede llegar a ser un poco incómoda para la persona mayor durante su vida diaria. Además como se mencionó, el consumo energético del dispositivo para poder hacerlo vestible y portable utilizando las Galileo es muy alto. Requiriendo dos baterías de litio de 8 Volts para su funcionamiento normal. Por otro lado, el tiempo de la carga útil de las pilas utilizadas es de aproximadamente 2 horas, y es necesario alrededor de entre 3 y 5 horas

para realizar su carga complemente utilizando un cargador especial. De esta forma se hace muy dificultoso el uso diario del dispositivo por parte de una persona anciana. El tamaño resultante del dispositivo también se incrementó debido a que las baterías y el sensor MPU6050 tuvieron que ser colocados en forma externa al gabinete, ver Fig. . El acelerómetro hubo que colocarlo en ese sitio por cuestiones de temperatura, razones que fueron detalladas previamente.

Un punto importante en la investigación, es que se requirió mucho más tiempo del que se había planificado inicialmente para hacer funcionar correctamente el acelerómetro y el giróscopo del sensor MPU6050. Hubo demoras en su correcta calibración, debido a que fue necesario probar distintos mecanismos para conseguir una calibración persistente en el tiempo. En la Etapa Inicial de la investigación se estuvo utilizando un programa Arduino que empleaba la biblioteca desarrollada por Jeff Rowberg para calibrar el sensor. Pero luego en el transcurso de la etapa de avance se descubrió, que al utilizar este mecanismo de calibración, el sensor se mantenía calibrado durante algunos meses únicamente y también se veía afectado por cambios bruscos de temperatura. Con lo cual se investigaron otras alternativas de calibración. Finalmente se solucionó este inconveniente realizando un método propio de calibración por medio de un programa que se desarrolló en Node.js. En su programación se realiza la calibración teniendo en cuenta todos los ejes del acelerómetro y del giróscopo, para el cálculo de la desviación particular de cada uno de ellos. Para eso se tienen en cuenta todas las posiciones en que puede estar el MPU6050 en todo momento. A diferencia del programa Arduino que utiliza la biblioteca de Jeff Rowberg, que únicamente emplea la posición horizontal en que se encuentra el sensor para realizar la calibración. La ubicación del programa en Node.js que fue desarrollado para la calibración del MPU6050, se encuentra especificado en el **Anexo D.**

A lo largo del proyecto, tuvimos inconvenientes en cuanto a la cantidad de los elementos de hardware que fueron planificados y adquiridos inicialmente en la investigación. Por distintos motivos se fueron dañando, como consecuencia del corto tiempo de vida de los mismos y mal funcionamientos, debido a la calidad de componentes disponibles en el mercado local al inicio del proyecto. Todas estas cuestiones retrasaron el desarrollo y las pruebas, debido a que se demoró un tiempo en conseguir los repuestos adecuados de los mismos. Además la placa Intel Galileo, posee incompatibilidad con ciertos componentes que funcionan en Arduino, como se expuso anteriormente. Por este motivo y debido a que Intel® descontinuó la producción de las placas Galileo Gen I y Gen II, así como también las Edison se decidió no implementar sobre este prototipo el resto de las funcionalidades tales como, pulsómetro y GPS entre otras, ya que los que se adquirieran resultarían incompatibles con el producto final a desarrollar.

2.5.1.22 Comparación de Resultados y elaboración de conclusiones.

La resolución de los distintos obstáculos presentados con el sensor MPU6050, nos sirvieron para obtener mediciones de caídas confiables, luego a partir de ellas determinar los patrones adecuados y posteriormente desarrollar un algoritmo de detección de caída eficiente. Por ende, para poder realizar el análisis de las mediciones eficientemente, se debieron repetir varias veces las capturas de los datos de las pruebas de las caídas y actividades de la vida diaria que realiza una persona. Esto fue debió no se podía conseguir hacer que funcione el algoritmo en con mediciones en tiempo real a partir de los patrones de umbrales detectados previamente. Cuestión que finalmente se pudo lograr con éxito luego varias modificaciones en el algoritmo.

2.5.1.23 Divulgación de resultados y publicaciones.

Durante el trascurso del proyecto se presentaron avances de la investigación en los principales congresos nacionales, ver Anexo B.

- XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016)
- IEEE Congreso Argentino de Ciencias de La Informática y Desarrollos de Investigación (CACIDI 2016)
- XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017)
- XXIII Congreso Argentino de Ciencia de la Computación (CACIC 2017)

Por otro lado dejamos a disposición de la comunidad científica, el código fuente de las aplicaciones que conforman el sistema de detector de caídas que se encuentran almacenados en un repositorio de GIT público, de libre acceso para cualquier persona. Su ubicación Web se encuentra especificado en el ANEXO D.

Además las distintas mediciones realizadas se durante todo la investigación se encuentran en el Anexo A.

2.6 Instrumentos de Recolección y Medición de Datos

El instrumento utilizado para recolectar los datos sobre la movilidad del adulto mayor es el prototipo construido. Él lee los datos medidos desde el acelerómetro (MPU 5060) (más detalle ver sección **¡Error! No se encuentra el origen de la referencia.**). Luego de filtrar los datos se p rocesan con algoritmo que detecta la caída. Para asegurarse que realmente detecta la caída, está dividido en etapas y cada uno poseen sus configuraciones de límites permitidos, la etapa final sirve para verificar los posibles falsos positivos (para más detalle de cómo funciona el algoritmo ver el apartado **¡Error! No se encuentra el origen de la referencia.**).

2.7 Confiabilidad y Validez de la Medición

La confiabilidad del algoritmo se verifica en las pruebas realizadas. Para logarlo se agregó especialmente al algoritmo como última fase, la lógica para descartar los falsos positivos. Estos podían ser producido por movimientos rápidos como sentarse o acostarse. El funcionamiento de esta etapa radica en comparar la posición del giroscopio en el estado inicial --estando parado-- y al momento de estar caído, donde el ángulo de referencia con el eje “Y” varia alrededor de 90 grados (más detalle ver la última etapa en la sección **¡Error! No se encuentra el origen de la r eferencia.**).

Con respecto a la confiabilidad de acelerómetro se verifica en la hoja de fabricante [17] que se utilizó el producto con los valores establecidos de amperaje y temperatura. Sin embrago en la práctica al calentar el circuito se verificaron cambios en los resultados aproximadamente 1% por hora. (ver Anexo A paper WICC2017).

Con respecto al voltaje que es suministrada por la batería, a medida que se usa va disminuyendo la energía entregada. Las mediciones del acelerómetro se corrigen por programa aplicando filtro de Kallman, para descartar valores picos que son producidos por el ruido electroestático del ambiente.

2.8 Métodos de Análisis Estadísticos

No Aplica

2.9 Resultados

Como resultado final de la investigación se desarrolló un prototipo de dispositivo detector de caídas IoT, cuya funcionalidad consiste en detectar accidentes de este tipo notificándole en tiempo real a la persona a cargo del mayor en su dispositivo móvil. Para el sistema embebido se utilizaron las placas de desarrollo Intel Galileo ® Generación 1 y el sensor MPU6050.

2.10 Discusión

Al principio del proyecto se había planificado emplear el botón de antipático, para que avise la persona mayor en caso de emergencia. Finalmente se decidió descartar esta idea, ya que el algoritmo detecta automáticamente cuando la persona se cae y en el momento en que se reincorpora luego de haberse caído.

3. Conclusiones

En este proyecto se ha podido realizar un prototipo que permite al portador generar un alerta a un cuidador online los que le permite recibir atención adecuada en el menor tiempo posible.
La idea de este dispositivo es reducir los costos producidos por un acompañante personal, para de esta forma llegar a las personas con recursos económicos muy limitados.
Para esto se construyó este prototipo como versión inicial, que permitirá construir un dispositivo pequeño para su comodidad de uso, con baterías de larga duración de carga y con sensores económicos. Esto se diferencia de una aplicación de telefonía celular en el costo reducido y la facilidad de uso para las personas mayores.
Se desarrolló un algoritmo de caídas, que se ejecutará utilizando un sensor únicamente como el MPU6050 disponible en el mercado y de bajo costo, eliminado falsos positivos sin necesidad de utilizar sensores adicionales.
Debido a los inconvenientes producidos por la placa GALILEO GEN I se generó un sistema que permite conectar distintos sensores, sin embargo en el próximo proyecto serán incorporados en diseño del dispositivo final, ya que se generó software modular al que se puede perfeccionar.
También se desarrolló un servidor, en la nube que envía mensajes y guarda la información para un posible uso posterior y una aplicación Android que puede incorporar nuevos tipos de mensajes.

4. Bibliografía

[1] Ashton, «That 'Internet of Things ',» *Thing. RFID Journal,* 2009.

[2] Biodatadevices, «http://www.biodatadevices.com/index.php,» 2014. [En línea].

[3] Aquino, «La tecnología como apoyo para alertas y ubicación de grupos de interés prioritario,» *Revista Científica y Tecnológica UPSE,* 2015.

[4] Wherton, Procter, Sugarhood y Hinder, Co-production in practice: how people with assisted living needs can help design and evolve technologies and services, 2015.

[5] Atkinson y Karimi, What the Internet of Things (IoT) Needs to Become Reality, Texas,USA, 2014.

[6] Intel ®, "Pagina Oficial de Intel ®," [Online]. Available: https://software.intel.com/en-us/iot/documentation.

[7] Intel, «https://software.intel.com/en-us/iot/software/ide,» [En línea].

[8] Intel, «https://software.intel.com/en-us/get-started-arduino,» [En línea].

[9] Arduino, «http://playground.arduino.cc/Main/WireLibraryDetailedReference,» [En línea].

[10] C. Monroe, Intel Galileo and Intel Galileo Gen 2, Apress Open, 2014.

[11] J. Lou, B. Zhong y D. Lv, «Fall Monitoring Device for old People based on Tri-Axial Accelerometer,» vol. 6, 2015.

[12] B. Banjanin, L. Zhang y C. Wang, «Fall Detection Monitoriong,» 2013.

[13] G. Prince Kanman, R. Hemamalini y I. Rajkumar, «LabVIEW based Abnormal Muscular Movement and Fall Detection using MEMS Accelerometer during the Occurrence of Seizure,» *Indian Journal of Science and Technology,* vol. 7, Octubre 2014.

[14] Y. Sui, Development of a low False-Alarm-Rate Fall-Down Detection System Based on Machine Learning for Senior Health Care,, University of Cincinnati, 2015.

[15] K. Jarzebski, ««https://github.com/jarzebski/Arduino-KalmanFilter,» [En línea].

[16] naylampmechatronics.com, «http://www.naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html,» [En línea].

[17] I. InvenSense, *MPU-6000 and MPU 6050 Product Specification Revision 3.4.,* Sunnyvale: Inc.InvenSense, 2013.

[18] R. Blanco, «Sistema de detección de caída en personas de la tercera edad para uso en centro geriatricos,» Bogota, 2010.

[19] E. Oporto Díaz, «Diseño de un sistema inalámbrico de detección de caídas aplicado a personas de la tercera edad basado en acelerómetro y teléfono móvil,» Lima-Peru, 214.

[20] E. Carnuccio, W. Valiente, M. Volker, G. De Luca, G. Garcia, D. Guilianelli y S. Barillaro, «Desarrollo de un Prototipo detector de caídas utilizando la placa Intel Galileo Generación I y el sensor MPU6050,» de *XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017).* , La Plata, Buenos Aires, 2017, pp. 954-964.

[21] S. Barillaro, W. Valiente, G. De Luca, E. Carnuccio, G. Gracía, M. Volker, D. Guilianelli y N. Casas, «Diseño de sistema IoT de monitoreo para personas mayores,» de *XIX Workshop de Investigadores en Ciencias de la Computación(WICC 2017)*, Buenos Aires, 2017, pp. 782-784.

[22] «Hostinger,» [En línea]. Available: https://www.hostinger.com/.

[23] M. Bean, Laravel 5 Essentials, Liverly Place: Pack Publishing LTd, 2015.

[24] L. Moroney, The Definitive Guide to Firebase:Build Android Apps on Google's Mobile Plataform, Apress, 2017.

[25] [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/concept-options?hl=es-419.

[26] J. O. Bastiga, Aplicación Android y Servicios REST para compartir Libros Electronicos, Alicante, España: Escuela Politenica Superior, 2017.

[27] F. Doglio, Pro REST API Develeopment with Node.js, Uruguay: Apress, 2015.

[28] «https://www.npmjs.com/package/webworker-threads,» [En línea].

[29] «https://nodejs.org/api/child_process.html,» [En línea].

[30] A. Ali, H. F. y M. Hannaford, «"Integrated Standard Environment for the Teaching and Learning of Operating Systems Algorithms Using Visualizations,» de *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, Valencia, España, 2010.

5. Producción científico-tecnológica PAPERS ANEXO

Se incluirán todas las producciones que se hayan producido en el marco de desarrollo del proyecto de investigación a saber:

5.1 Publicaciones

Presentar originales, copias o certificaciones que avalen la producción. (Se recuerda la importancia de citar en cada publicación a la Universidad Nacional de La Matanza, a la Unidad Académica en donde fue acreditado el proyecto, así como también el Programa de origen de la Investigación -Programa PROINCE-UNLaM- y su código de identificación).Indicar las publicaciones de acuerdo con el orden que a continuación se detalla:

5.2 Artículos

AÑO 2016:

- **AUTOR (ES):** Barillaro Sebastián, De Luca Graciela, Valiente Waldo, Carnuccio Esteban, García Gerardo, Volker Mariano, Giulianelli Daniel Alberto, Casas Nicanor, Pérez Maximiliano
  TÍTULO: Diseño de sistema IoT de monitoreo y alarma para personas mayores
  FUENTE de la publicación: XVIII Workshop de Investigadores en Ciencias de la Computación
  PÁGINAS: 712-716
  NUMERO: 1er Ed.
  EDITORIAL: RedUNCI, 2016.
  LUGAR: Buenos Aires. UNNOBA; La Plata.
  FECHA: abril 2016
  ISBN: 978-950-698-377-2

- **AUTOR (ES):** Graciela E. De Luca ; Esteban A. Carnuccio ; Gerardo G. Garcia ; Sebastián Barillaro
  TÍTULO: IoT fall detection system for the elderly using Intel Galileo development boards generation I
  FUENTE de la publicación: IEEE CACIDI 2016 - IEEE Conference on Computer Sciences
  DOI: 10.1109/CACIDI.2016.7785.997
  EDITORIAL: IEE EXPLORER
  FECHA: 22/12/2016
  Electronic ISBN: 978-1-5090-2938-9
  Print on Demand (PoD) ISBN: 978-1-5090-2939-6

AÑO 2017:

- **AUTOR (ES):** Carnuccio Esteban, Valiente Waldo, Volker Mariano, De Luca Graciela, García, Gerardo, Giulianelli Daniel Alberto, Barillaro Sebastián
  TÍTULO: Desarrollo de un prototipo detector de caídas utilizando la placa Intel Galileo Generación I y el sensor MPU6050
  FUENTE de la publicación: Libro de Actas CACIC2017
  PÁGINAS: 954-964
  NUMERO: 1er Ed.
  EDITORIAL: RedUNCI, 2017.
  LUGAR: Buenos Aires. UNNOBA; La Plata.
  FECHA: 2017
  ISBN: 978-950-34-1539-9

5.3. Capítulos de Libros.

-NA-

5.4. Libros

-NA-

6. Congresos Internacionales, Nacionales, Simposios, Jornadas, otros

AÑO 2016:

- **AUTOR (ES):** Barillaro Sebastián, De Luca Graciela, Valiente Waldo, Carnuccio Esteban, García Gerardo, Volker Mariano, Giulianelli Daniel Alberto, Casas Nicanor, Pérez Maximiliano
  TÍTULO: Diseño de sistema IoT de monitoreo y alarma para personas mayores
  TIPO: Ponencia
  REUNIÓN: XVIII Workshop de Investigadores en Ciencias de la Computación
  LUGAR: Universidad Nacional de Entre Ríos
  FECHA REUNIÓN: 14 y 15 de Abril del 2016
  RESPONSABLE: Universidad Nacional de Entre Ríos
  TIPO DE TRABAJO: artículo completo
  FUENTE: Proccedings XVII Workshop de Investigadores en Ciencias de la Computación
  EDITORIAL: RedUNCI, 2016
  ISBN: 978-950-698-377-2

- **AUTOR (ES):** Graciela E. De Luca ; Esteban A. Carnuccio ; Gerardo G. Garcia ; Sebastián Barillaro
  TÍTULO: IoT fall detection system for the elderly using Intel Galileo development boards generation I
  TIPO: Ponencia
  REUNIÓN: Congreso Argentino de Ciencias de la Informática y Desarrollo de la Investigación (CACIDI 2016)
  LUGAR: Facultad de Ciencias Económicas de la UBA; CABA, Buenos Aires, Argentina
  FECHA REUNIÓN: 30 de noviembre, 1 y 2 de diciembre de 2016
  RESPONSABLE: Universidad Nacional de San Martín, CAECE y Universidad Nacional de Chile
  TIPO DE TRABAJO: artículo completo
  FUENTE: IEEE CACIDI 2016 - IEEE Conference on Computer Sciences
  EDITORIAL: IEE EXPLORER
  ISBN: 978-1-5090-2938-9

AÑO 2017:

- **AUTOR (ES):** Giulianelli Daniel Alberto, De Luca Graciela García Gerardo, Carnuccio Esteban, Valiente Waldo, Volker Mariano
  TÍTULO: Diseño de sistema iot de monitoreo y alarma para personas mayores
  TIPO: Presentación de Poster
  REUNIÓN: XIX Wokshop de Investigadores en Ciencias de la Computación (WICC2017)
  LUGAR: Instituto Tecnológico Buenos Aires (ITBA); CABA, Buenos Aires.
  FECHA REUNIÓN: 27 y 28 de Abril del 2017
  RESPONSABLE: Instituto Tecnológico Buenos Aires
  TIPO DE TRABAJO: artículo completo
  FUENTE: Proccedings XVIII Workshop de Investigadores en Ciencias de la Computación
  EDITORIAL: RedUNCI, 2017
  ISBN: 978-987-42-5143-5

- **AUTOR (ES):** Carnuccio Esteban, Valiente Waldo, Volker Mariano, De Luca Graciela, García, Gerardo, Giulianelli Daniel Alberto, Barillaro Sebastián
  TÍTULO: Desarrollo de un prototipo detector de caídas utilizando la placa Intel Galileo Generación I y el sensor MPU6050
  TIPO: Ponencia
  REUNIÓN: XXIII Congreso Argentino de Ciencia de la Computación (CACIC)
  LUGAR: Facultad de Informática, UNLP. La Plata, Buenos Aires
  FECHA REUNIÓN: del 9 a 13 de Noviembre del 2017

RESPONSABLE: Universidad Nacional de San Martín, CAECE y Universidad Nacional de Chile
TIPO DE TRABAJO: artículo completo
FUENTE: Libro de Actas CACIC2017
EDITORIAL: RedUNCI, 2017
ISBN: 978-987-42-5143-5

ANEXO A: MEDICIONES REALIZADAS

Contenido

INTRODUCCION

Por simplicidad en este documento se muestran únicamente los gráficos resultantes que fueron obtenidos de las últimas mediciones realizadas con el prototipo del detector de caídas. El resultado de las restantes mediciones se encuentra almacenadas en la siguiente dirección web.

- **Mediciones efectuadas:** https://gitlab.com/soa-unlam/monitoreo/tree/Pruebas/Investigaci%C3%B3n/Mediciones%20Caidas

En consecuencia a continuación se muestran los resultados de las siguientes mediciones.

- **Actividades de la vida diaria:**
  - Sin movimiento estando de pie
  - Caminando
  - Saltando
  - Sentándose y levantándose de una silla

- Subiendo y bajando por una escalera.
- Acostándose y levantándose en una cama

- **Actividades de caídas:**
  - Caerse hacia adelante estando de pie
  - Caerse hacia atrás estando de pie
  - Caerse de costado estando de pie.
  - Caerse hacia adelante estando sentado
  - Caerse de costado estando sentado
  - Caerse saltando

MEDICIONES REALIZADAS

Sin movimiento estando de pie

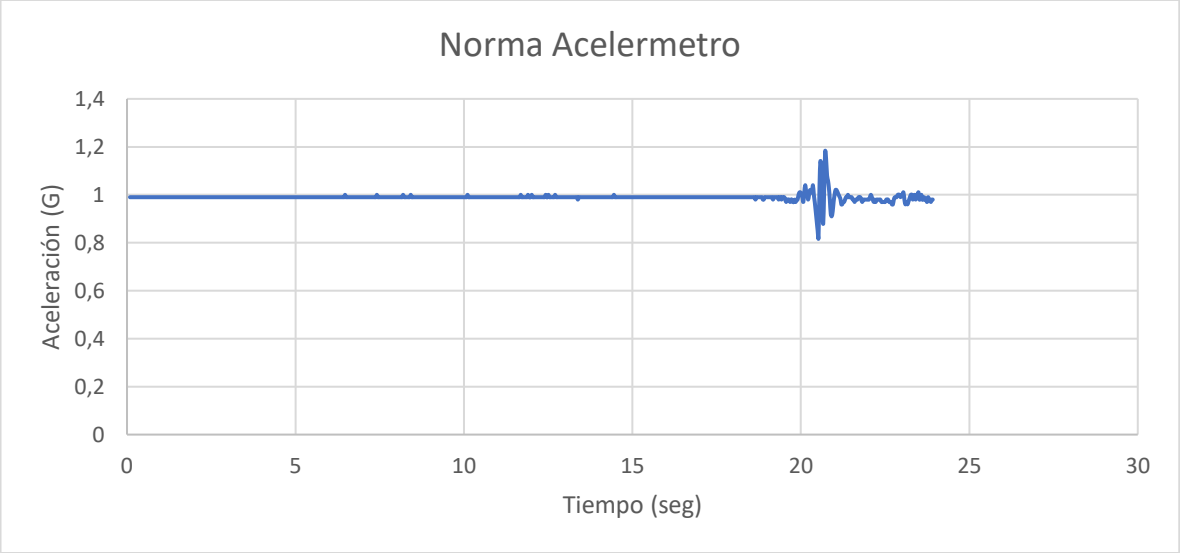


Ilustración 1 Norma Acelerómetro estando estático de pie

Caminando

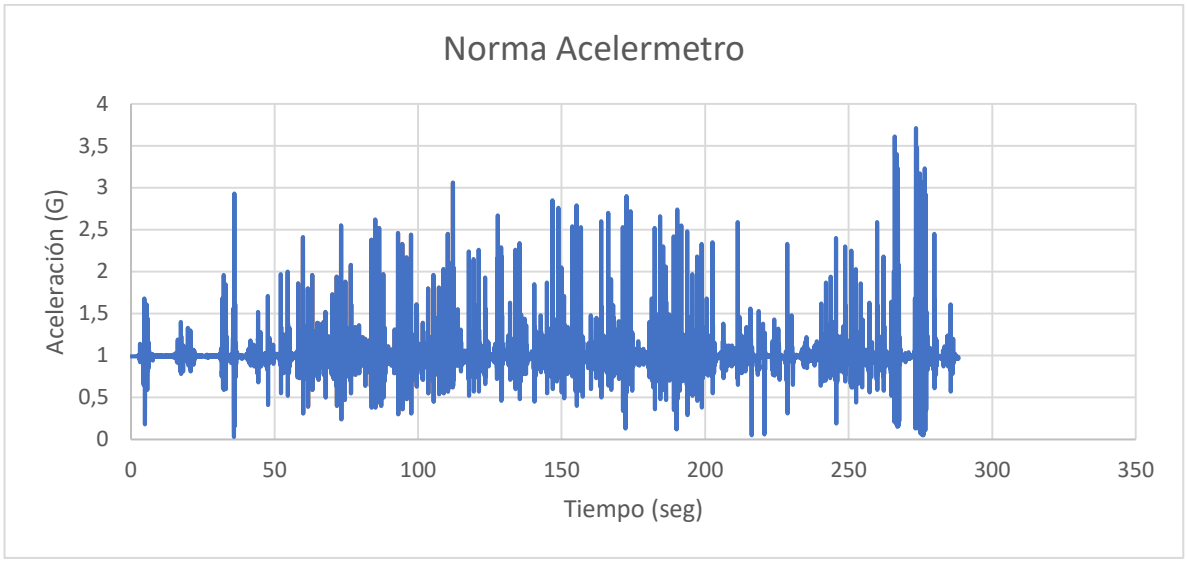


Ilustración 2 Norma Acelerómetro Caminando

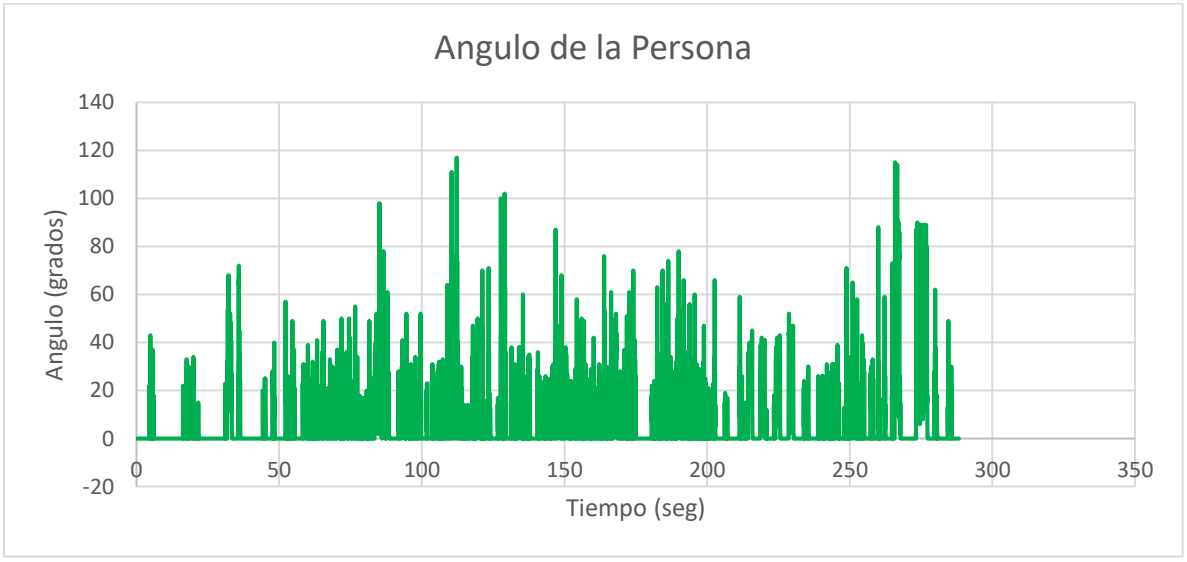


Ilustración 3 Angulo de la Persona Caminando

Sentarse y Levantadose de una Silla

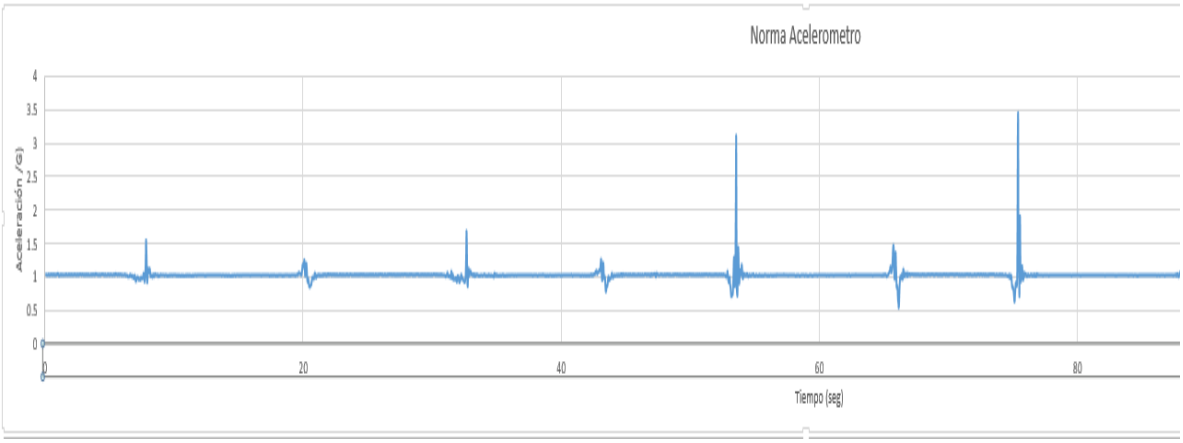


Ilustración 4 Norma Acelerómetro Sentándose y Levantadose de una silla

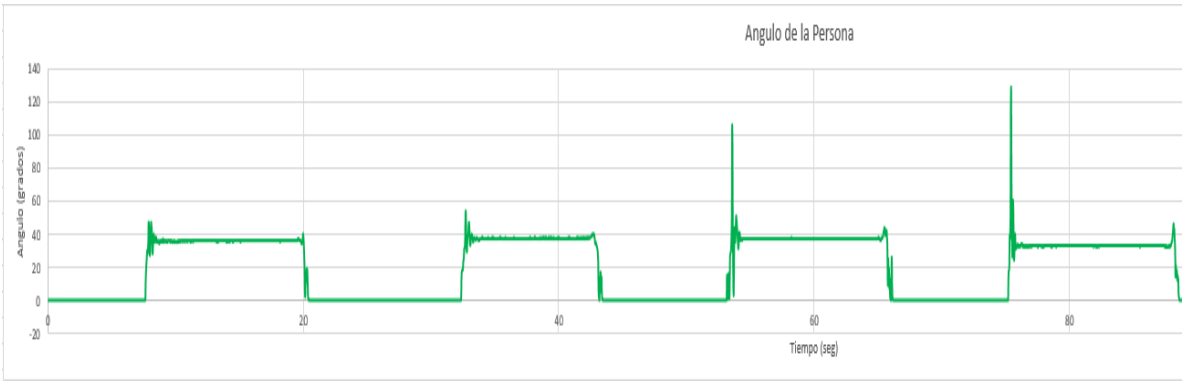


Ilustración 5 Angulo de la Persona Sentándose y Parándose de una silla

Subiendo y bajando por una escalera

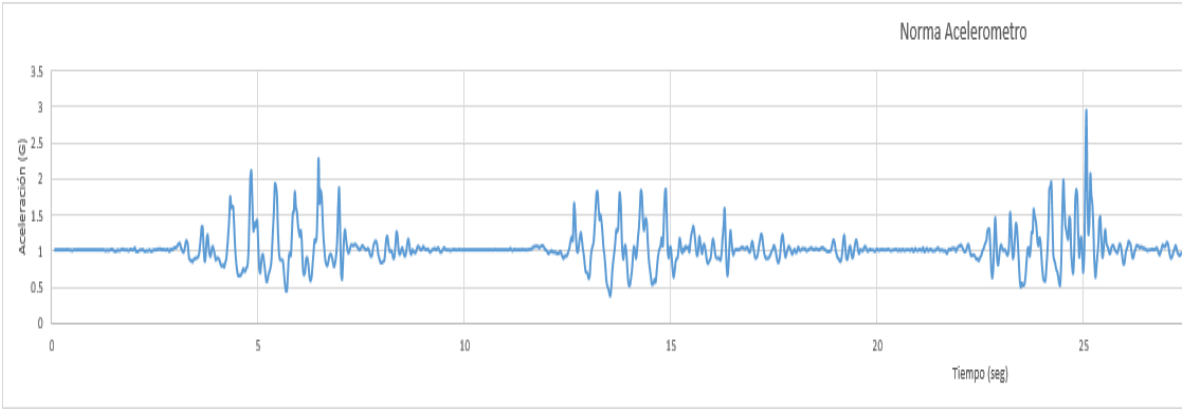


Ilustración 6 Norma Acelerómetro subiendo y bajado por una escalera

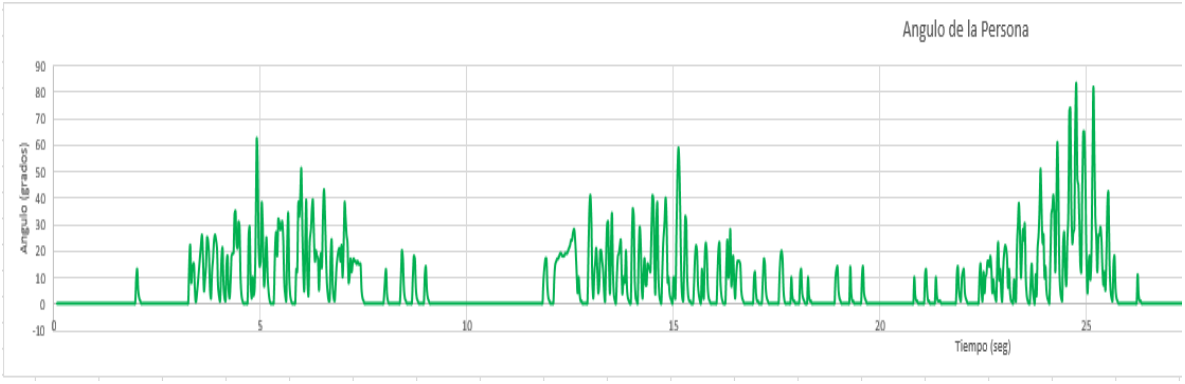


Ilustración 7 Angulo de Persona subiendo y bajando por una escalera

Acostandose y Levantadose de una cama

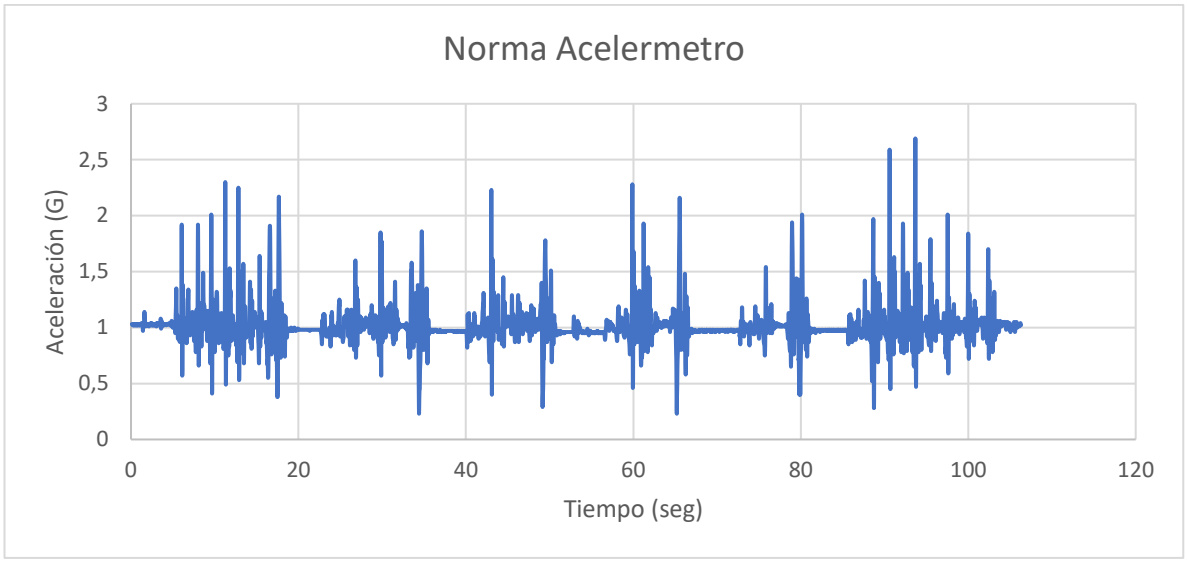


Ilustración 8 Norma Acelerómetro Acostándose y levantándose de una cama

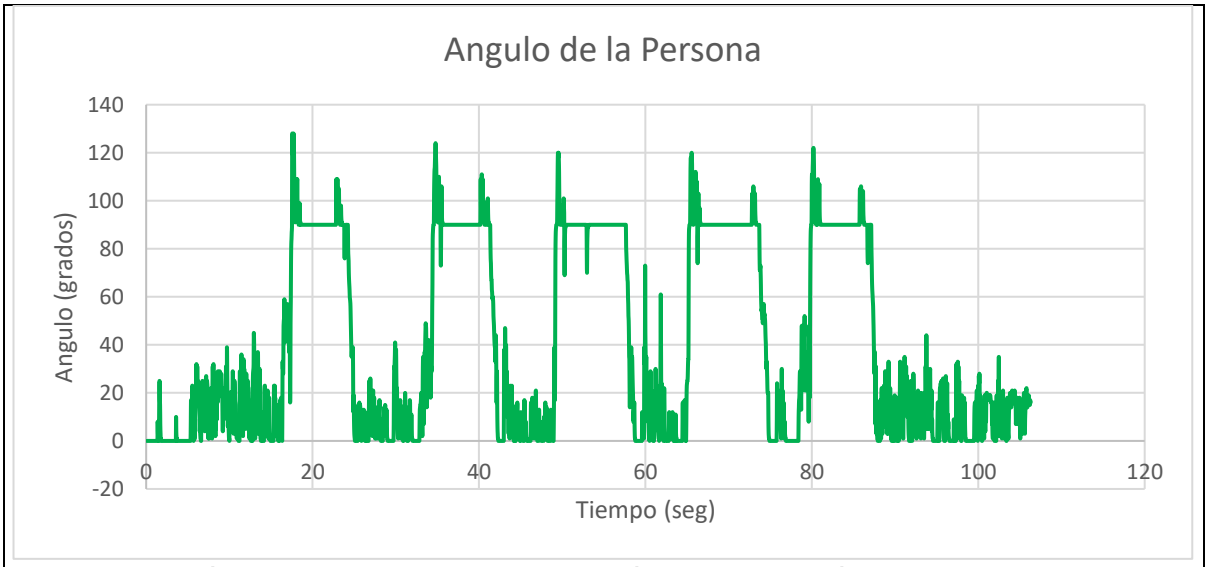


Ilustración 9 Angulo de la Persona acostándose y levantándose de una cama

Cayéndose hacia adelante estando de pie

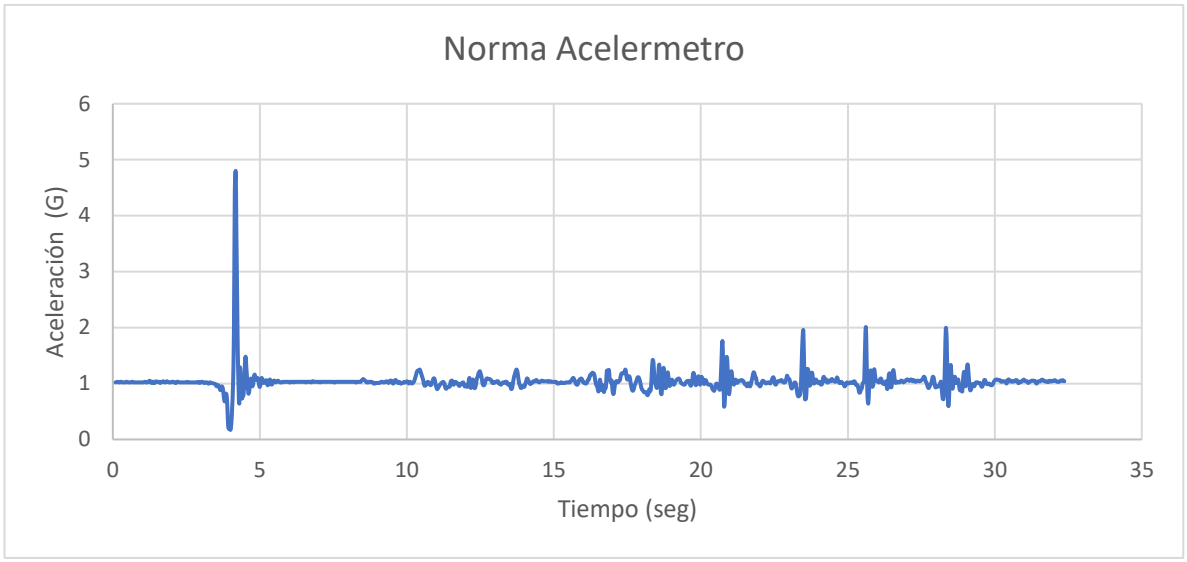


Ilustración 10 Norma Acelerómetro Cayéndose hacia adelante estando de pie

Ilustración 11 Angulo de la persona Cayéndose hacia adelante estando de pie

Cayéndose hacia atrás estando de pie

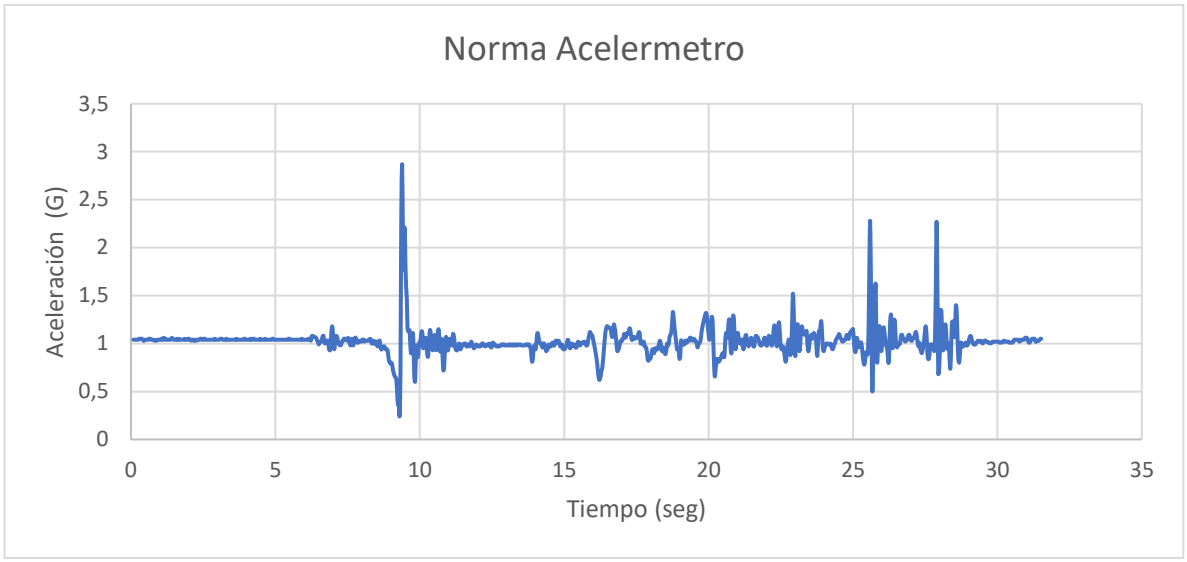


Ilustración 12 Norma Acelerómetro cayéndose hacia atrás estando de pie

Ilustración 13 Angulo de la Persona Cayéndose hacia atrás estando de pie

Cayéndose de costado estando de pie

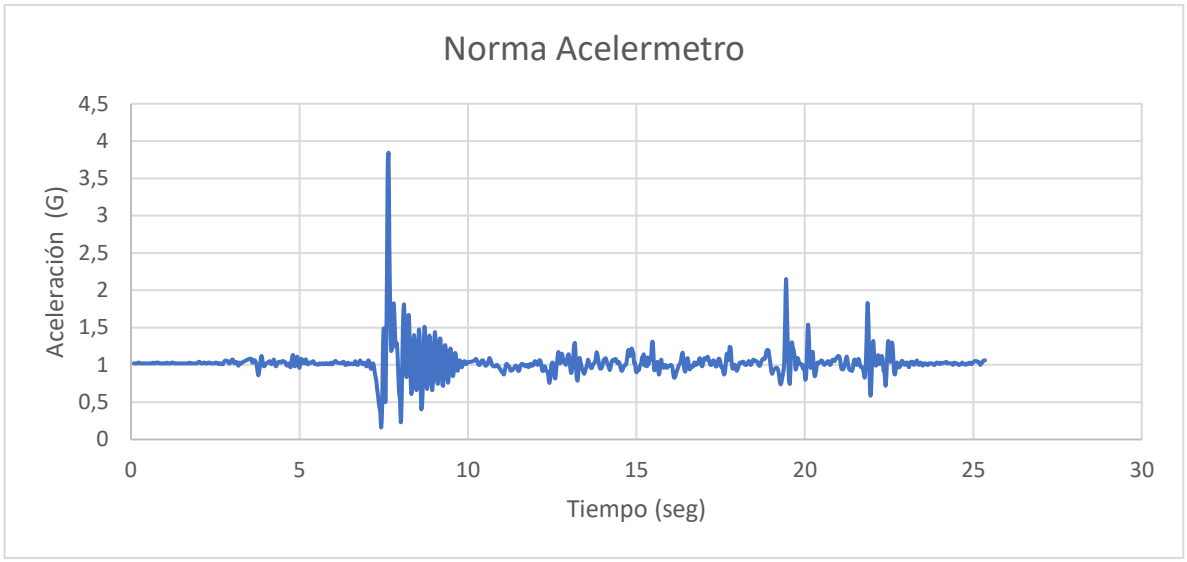


Ilustración 14 Norma Acelerómetro cayéndose de costado estando de pie

Ilustración 15 Angulo de la Persona Cayéndose de costado estando de pie

Cayéndose hacia adelante estando sentado

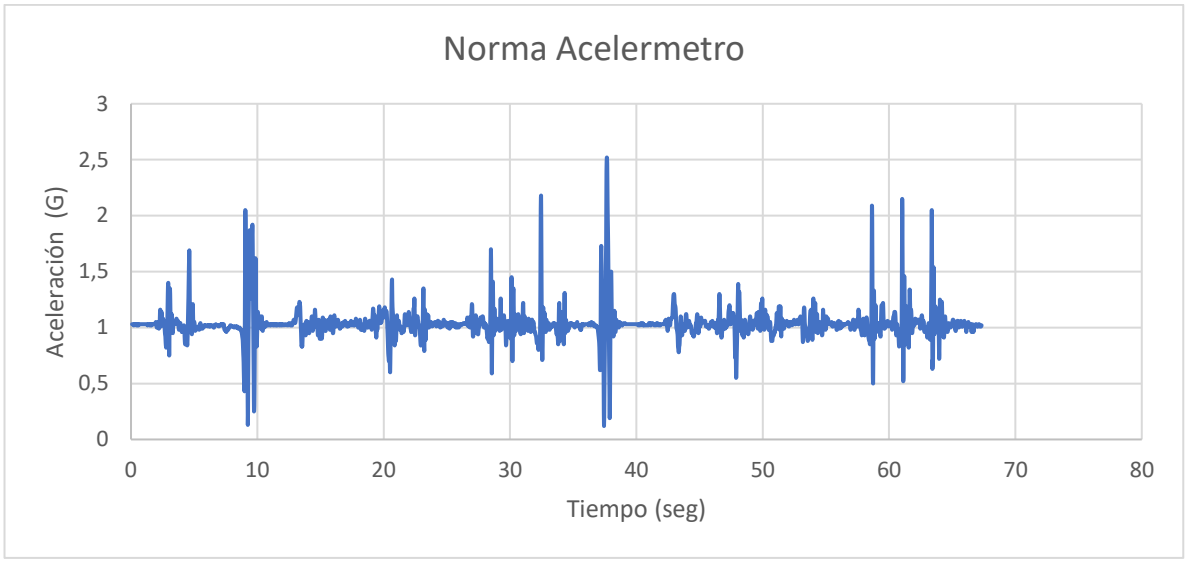


Ilustración 16 Norma Acelerómetro cayéndose hacia adelante estando sentado

Ilustración 17 Angulo de la Persona Cayéndose hacia adelante estando sentado

Cayéndose de costado estando sentado

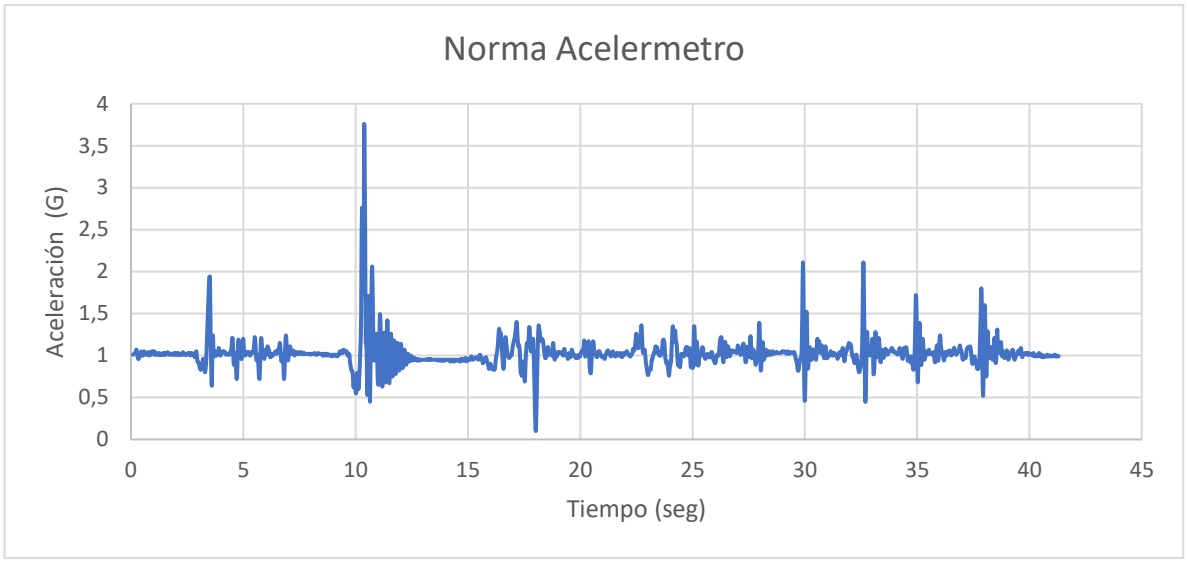


Ilustración 18 Norma Acelerómetro cayéndose de costado estando sentado

Ilustración 19 Angulo de la Persona Cayéndose de costado estando sentado

Cayéndose Saltando

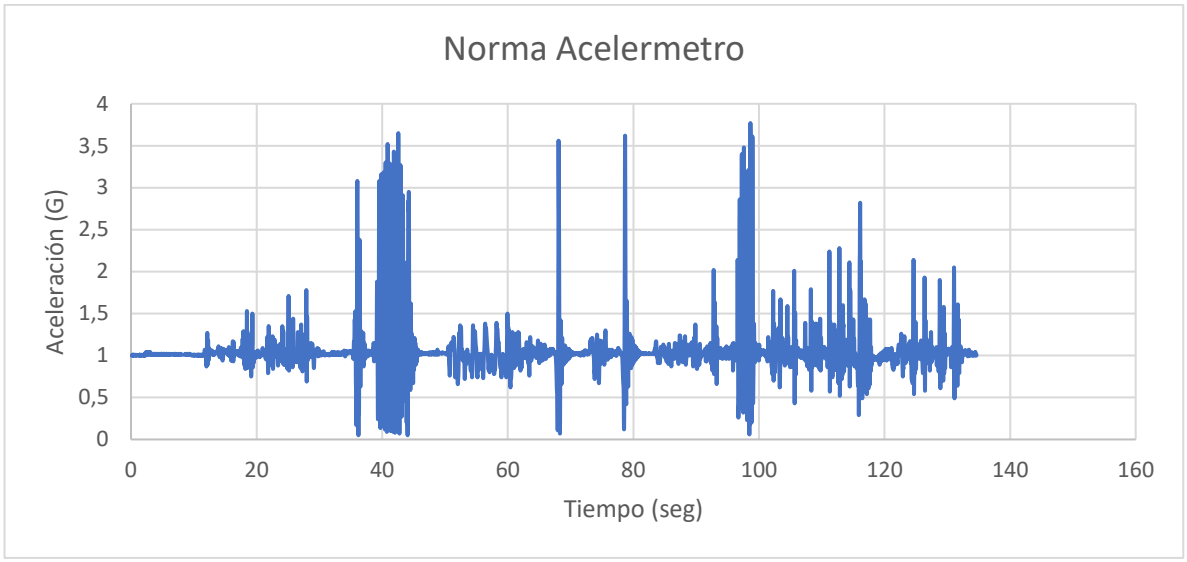


Ilustración 20 Norma Acelerómetro cayéndose saltando

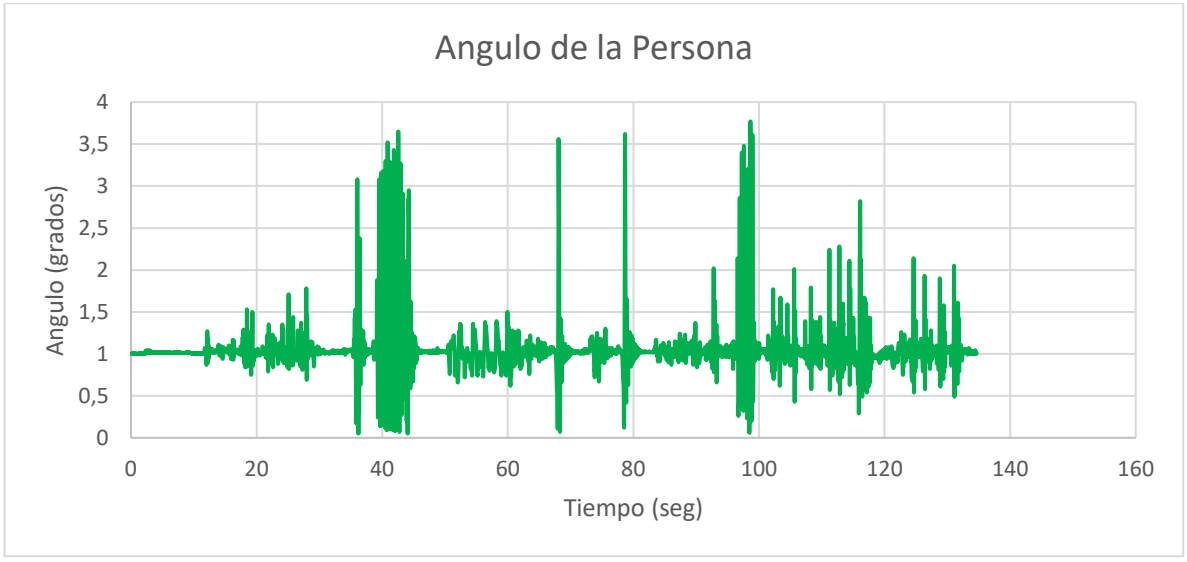


Ilustración 21 Angulo de la persona cayéndose saltando