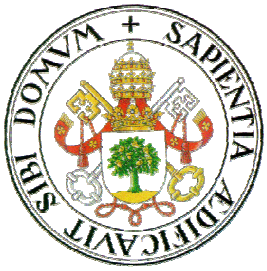


Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica en Informática de Gestión

Aplicación para la detección de caídas

Alumna: María Jesús Tirado Núñez

Tutor: Miguel Ángel Laguna Serrano.

Agradecimientos

A mis padres Félix y Elena que con su sacrificio

han permitido que hoy me encuentre aquí.

Aplicación para la detección de caídas.

Contenido

Introducción al proyecto 9
1. Introducción 10
2. Objetivos planteados 12
3. Estructura de la memoria. 13
El wiimote y sus ventajas como acelerómetro. 15
1. Características mando wiimote. 16
1.1. Diseño 16
1.2. Funcionalidad 16
1.3. Memoria 18
1.4. Alimentación 18
1.5. Bluetooth 18
2. Algunas aplicaciones de investigación que utilizan el wiimote. 19
2.1. Aplicación para la rehabilitación física y neurológica de pacientes. 19
2.2. Pizarra electrónica 19
2.3. WiiAirBoard 20
2.4. WiiHome 20
Algoritmo de detección de caídas. 23
1. Introducción 24
2. Dispositivos anteriores en la detección de caídas basados en acelerómetros. 24
2.1. SPEEDY: Un detector de caídas en un reloj de muñeca. 24
2.2. Tunstall detector de caídas comercial. 24
3. Estudio de las aceleraciones del cuerpo humano medidas con un acelerómetro triaxial en personas mayores. 25
3.1. Observación del estudio, cálculo de umbrales. 32
4. Algoritmo final de detección de caídas 33

4.1. Medidas de aceleración con el wiimote. 35
Análisis 41
1. Introducción. 42
2. Requisitos Software 42
2.1. Descripción general de la aplicación. 42
2.2. Contexto de aplicación. 43
3. Roles 45
4. Requisitos no funcionales 45
5. Funcionalidad de la aplicación. 47
5.1. Requisitos Funcionales 47
6. Casos de uso. 50
6.1. Actores 50
6.2. Casos de uso 51
6.3. Descripción detallada de los casos de uso. 54
7. Modelo de Análisis 72
7.1. Introducción 72
7.2. Modelo inicial de clases 72
Diseño 74
1. Introducción. 75
2. Diagramas de secuencia en diseño. 75
2.1. Diagrama de secuencia UC-001 Elegir sensor. 75
2.2. Diagrama de secuencia UC-002 Comprobar Calibración 76
2.3. Diagrama de secuencia UC-003 Calibrar Sensor. 78
2.4. Diagrama de secuencia UC-004 Asociar paciente a sensor 80
2.5. Diagrama de secuencia UC-005 Iniciar detección 82
2.6. Diagrama de secuencia UC-006 Parar detección 83
2.7. Diagrama de secuencia UC-007 Consultar estado sensor 84
2.8. Diagrama de secuencia UC-008 Consultar estado paciente. 86
2.9. Diagrama de secuencia UC-009 Llamada de emergencia 87

2.10. Diagrama de secuencia UC-010 Detener notificación de emergencia. 88
2.11. Diagrama de secuencia UC-012 Activar Alarma. 89
2.12. Diagrama de secuencia UC-012 Aplicar algoritmo detección de caídas 91
2.13. Diagrama de secuencia UC-013 Enviar alarma servidor 93
3. Diagrama final de clases. 94
3.1. Especificaciones de las clases 97
3. Arquitectura del sistema. 101
3.2. Proyecto completo de detección de caídas. 101
3.3. Diagrama de paquetes para MonitorizacionCaídas 102
4. Diseño de la Capa de Presentación. 103
4.1. Diagrama de clases 103
5. Diseño de la capa lógica de negocio. 106
6. Diseño de la capa de persistencia. 106
Implementación 108
1. Introducción 109
2. Software utilizado 109
3. Hardware empleado. 109
4. Clases presentes en la el paquete WiimoteLib 110
4.1. Struct ButtonState 113
4.2. Struct AccelCalibratioInfo 113
4.3. Struct LEDState 114
4.4. Struct AccelState 114
5. Algunas funciones presentes en el proyecto de detección de caídas 117
5.1. Conexión y búsqueda de wiimotes 117
5.2. Actualización de los valores del wiimote. 117
5.3. Implementación Función aplicarAlgoritmo() 118
5.4. Como conocemos la orientación que tiene en todo momento el wiimote 121

5.5. Implementación de la función LlamadaEmergencia 122
Pruebas 126
1. Introducción 127
2. Casos de prueba aplicación. 127
3. Casos de prueba del algoritmo de detección. 131
3.1. Pruebas diferentes movimientos. 132
3.2. Pruebas simulando caídas 132
4. Conclusiones pruebas realizadas en la detección del algoritmo 132
Manual de usuario 135
1. Instalación 136
2. Conexión del wiimote al PC. 139
2.1. Conexión del wiimote con la aplicación Bluesoleil 139
2.2. Conectar el wiimote con la aplicación bluetooth de Windows 143
3. Inicio de la aplicación 149
4. Asignar un paciente al wiimote. 151
5. Calibración del wiimote. 152
6. Ver la información detallada de los sensores. 154
7. Realizar una llamada de emergencia 155
8. Desactivar una alarma. 156
9. Notificación de una alarma 158
Conclusiones 161
1. Objetivos alcanzados 162
2. Conocimientos Adquiridos 163
3. Líneas de trabajo futuras. 163
Bibliografia. 165
Apéndice A: Calibración del Wiimote. 169
Apéndice B: Conversión del proyecto de detección de caídas en una línea de producto. 173
1. Introducción. 174

2. Finalidad de la línea de producto monitorización de pacientes. 174
3. Características del proyecto anterior. 174
3.1. Tecnología disponible en la línea de producto monitorización de pacientes. 175
3.2. Problemas con el bluetooth del mando wiimote y PDA con Windows Mobile. 175
3.3. Solución propuesta para la integración de caídas en la línea de productos 176
4. Descripción de la línea de producto 176
4.1. Diagrama de características 176
4.2. Características de los paquetes 177
5. Conclusiones paquete detector de caídas en dispositivo móvil. 178
Apéndice C. Microsoft .Net y Net Compact Framework 181
1. Introducción. 182
2. .NET Framework 183
3. Objetivos del .NET Framework 184
4. Common Language Runtime (CLR) 186
5. Base Class Library (BCL). 187
6. .NET Compact Framework 188
7. Desarrollo bajo .NET: Visual Studio. 190
8. C#. 190
9. ASP.NET 191
Apéndice D contenido del CD-ROM 193
1. Contenido CD-ROM 194

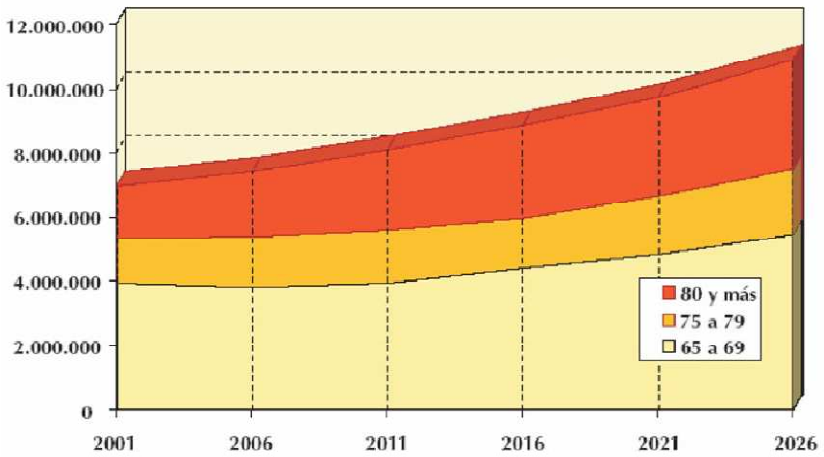
Introducción al proyecto

1. Introducción

Este proyecto nace junto con la Universidad de Valladolid con un objetivo claro, ayudar a personas dependientes a tener una mayor calidad de vida e independencia. Este proyecto es uno de los primeros pasos para la consecución de este objetivo.

El Consejo de Europa define la dependencia como "la necesidad de ayuda o asistencia importante para las actividades de la vida cotidiana", o, más concretamente, como "un estado en el que se encuentran las personas que por razones ligadas a la falta o la pérdida de autonomía física, psíquica o intelectual, tienen necesidad de asistencia y/o ayudas importantes a fin de realizar los actos corrientes de la vida diaria y, de modo particular, los referentes al cuidado personal". Por otra parte, aunque los datos son ya algo antiguos, la Encuesta sobre Discapacidades, Deficiencias y Estado de Salud de 1999 (EDDES 99) cifraba en 3.528.221 el número total de personas con alguna discapacidad o con limitaciones que han causado o pueden llegar a causar discapacidades, lo cual representa aproximadamente un 9% de la población española.

El envejecimiento de la población española es un factor que incrementará significativamente en el futuro el porcentaje de población dependiente, ya que existe una estrecha relación entre dependencia y edad. Según las proyecciones de población del INE, el número de personas mayores de 65 años supondrá en veinte años más del 20% de la población total española, y dentro de ese grupo de población se producirá un fuerte aumento de las personas mayores de 80 años, edad a partir de la cual la tasa de prevalencia de dependencia crece notablemente.



Fuente: INE, Proyecciones de población calculadas a partir del Censo de 2001.

Figura 1

Aplicación para la detección de caídas.

Hasta ahora han sido principalmente las familias, especialmente las mujeres, las que se han encargado del cuidado de las personas dependientes. Sin embargo, los cambios que han ido aconteciendo en la sociedad, tales como la incorporación masiva de la mujer al mercado de trabajo, la caída de la fecundidad, la movilidad geográfica o las transformaciones de las relaciones familiares hacen que este modelo de "asistencia informal" se vuelva insostenible a medio plazo.

El papel de las tecnologías de la información y las comunicaciones (TIC) como mecanismo de integración social de las personas mayores, discapacitadas o dependientes en general se está imponiendo rápidamente entre capas cada vez más amplias de la población. Estas tecnologías están generando enormes expectativas y se debe prestar especial atención al carácter de asequibilidad de las mismas para que sean adoptadas por el mayor número de individuos. El incremento de los costes de asistencia y la dispersión geográfica de una población cada vez más envejecida, hacen necesario el despliegue de tecnologías que puedan prestar servicios personalizados basados en sistemas distribuidos de bajo coste con herramientas de computación ubicuas. En consecuencia, una de las ramas más activas de la Telemedicina es la monitorización y suministro de servicios remotos. Las opciones crecientemente disponibles pretenden facilitar la vida diaria del paciente. Los requisitos incluyen una monitorización mínimamente invasiva (inalámbrica, si es posible), con sensores dinámicos (envío frecuente de la información), estándares (para facilitar el envío de información desde países diferentes), inteligentes (capaces de alertar al paciente en situaciones de riesgo) y adaptativos (para proporcionar soporte para la toma de decisiones de forma remota), necesitamos con urgencia:

- reducir costes en respuestas rápidas a problemas de salud
- extender los cuidados médicos para que sean accesibles a más personas
- ofrecer cuidados personalizados

Además de los enfermos propiamente dichos, otras personas con diverso grado de dependencia pueden ver mejorado su grado de autonomía: personas con distintas discapacidades físicas o psíquicas, ancianos que viven solos o en residencias, etc. Los tipos posibles de monitorización son variados: ritmo cardiaco, saturación de oxígeno en sangre, temperatura, niveles de glucosa, etc. Otro tipo de situación frecuente que requiere solución, muy demandada sobre todo en personas ancianas, es la detección de caídas fortuitas ya que:

- El 30% de las personas mayores de 65 años se cae al menos una vez al año.
- El miedo, la ansiedad y la depresión aumentan debido al riesgo de caídas.
- Las caídas son las responsables de 70% de los accidentes mortales de las personas mayores de 75 años.
- Una caída en un anciano incluso siendo muy suave puede provocarle daños irreversibles incluso la muerte.

Por todos estos puntos resulta indispensable el desarrollo de un dispositivo capaz de detectar caídas de forma automática.

Las redes inalámbricas facilitan servicios móviles adaptables a diferentes escalas incluyendo áreas dispersas ó entornos más restringidos servicios Web basados en redes locales inalámbricas (Wi-Fi), y comunicaciones de corto alcance como Bluetooth proporcionan un soporte global para sistemas de e-Salud encargados de obtener, evaluar, transmitir y analizar los datos obtenidos de forma remota.

2. Objetivos planteados

Los objetivos planteados en este proyecto han sido los siguientes:

- Desarrollar un algoritmo de detección de caídas lo más fiable y robusto posible el cual minimice los falsos positivos y sea capaz de detectar una amplia gama de caídas.
- Imponer las características técnicas y físicas que debe poseer el sensor elegido para poder implementar el algoritmo desarrollado.
- Implementar dicho algoritmo en una aplicación informática real y comprobar su funcionamiento a través de caídas simuladas y movimientos bruscos.
- Implementar un modulo móvil con base una PDA Windows mobile de la línea de producto monitorización de caídas desarrollada por el grupo de investigación GIRO.

Los Objetivos secundarios que derivan de este proyecto son los siguientes:

- Aumentar la seguridad de las personas mayores.
- Proporcionar tranquilidad a las familias y a los cuidadores.
- Reducir los costes derivados a la sociedad derivados del cuidado de personas dependientes así como dar la posibilidad de independencia a muchas personas que podrán vivir solos en sus hogares.

Para poder llevar a cabo estos objetivos hemos tenido que ampliar los conocimientos en los siguientes puntos.

- Aprender acerca de los movimientos producidos por el cuerpo humano así como los diferentes dispositivos que existen para medirlos de forma cuantitativa.
- Aprender acerca de estudios y métodos anteriores en la detección de caídas en personas.
- Ampliar los conocimientos en comunicaciones inalámbricas especialmente en comunicación Bluetooth.
- Ampliar los conocimientos en desarrollo de aplicaciones Visual Studio 2008 y en general sobre la arquitectura Microsoft. Net.
- Aprender a desarrollar una aplicación en un dispositivo móvil con sistemas operativos Windows Mobile 6 y 6.1.

Aplicación para la detección de caídas.

3. Estructura de la memoria.

- *Mando wiimote.*

En esta sección se explicará la tecnología de la que disponemos para desarrollar la aplicación planteada así como algunos ejemplos de proyectos desarrollados utilizando un wiimote.

- *Algoritmo de detección planteado*

A partir de estudios anteriores se explicará cómo es el algoritmo de detección sobre el que funciona la aplicación.

- *Análisis*

Este documento describe la especificación de requisitos obtenida a partir del punto 2, y así comenzar a describir los diferentes diagramas de análisis hasta obtener el modelo de dominio del sistema.

- *Diseño*

Este documento permite, a partir del documento de análisis, detallar el funcionamiento de la aplicación y adaptarlo a la tecnología escogida en el punto 3.

- *Implementación*

Se trata del documento de despliegue de la aplicación, en el cual se pueden comprobar los diferentes elementos que intervienen en el funcionamiento

- *Pruebas*

Observaciones de las pruebas de caja negra realizadas en la aplicación y las caídas probadas en la detección de caídas.

- *Manual de Usuario*

Documento que ilustra cómo funciona la aplicación.

- *Conclusiones*

Las conclusiones obtenidas de dicho estudio además de posibles proyectos futuros que complementen a este o le mejoren.

- *Bibliografía*

- *Apéndices.*

Aplicación para la detección de caídas.

El wiimote y sus ventajas como acelerómetro.

1. Características mando wiimote.

Wiimote es el mando principal de la consola Wii de Nintendo. Sus características más destacables son la capacidad de detección de movimiento en el espacio y la habilidad de apuntar hacia objetos en la pantalla.

1.1. Diseño

El diseño del Wiimote no se basa en los tradicionales mandos para los videojuegos. A diferencia de ellos, es similar a un control remoto de televisión, creado para ser utilizado en una sola mano y de la manera más intuitiva posible.

En su cara frontal, el Wiimote presenta los botones "A", "1", "2", "+", "-", "HOME" y "POWER", junto a una cruz direccional. En la parte anterior sólo presenta el botón "B", en un formato similar a un gatillo.

Adicionalmente, en su parte frontal incluye un altavoz y cuatro luces numeradas que indican el número de jugador al que corresponde tal mando y ver el tiempo de vida de la batería. En la parte inferior viene unida una correa de seguridad que se amarra a la muñeca para evitar soltar y dañar el control de forma accidental.

1.2. Funcionalidad

Barra de sensores con LEDs Infrarrojos y detección de movimientos

Figura 3. Barra de sensores con LEDs Infrarrojos

El Wiimote tiene la capacidad de detectar la aceleración a lo largo de tres ejes mediante la utilización de un acelerómetro ADXL330. El Wiimote también cuenta con un sensor óptico PixArt, lo que le permite determinar el lugar en el Wiimote está apuntando.

Aplicación para la detección de caídas.

A diferencia de un mando que detecta la luz de una pantalla de televisión, el Wiimote detecta la luz de la Barra sensor de la consola (número de modelo RVL-014), lo que permite el uso coherente, independientemente del tipo o tamaño de la televisión. Esta barra mide aproximadamente 20 cm de longitud y cuenta con diez LED infrarrojos, con cinco LED dispuestos en cada extremo de la barra. En cada grupo de cinco LED, el LED más lejano fuera del centro apunta ligeramente lejos del centro, el LED más cercano al centro apunta ligeramente hacia el centro, mientras que los tres LED entre ellos están apuntando directamente hacia adelante y agrupados. El cable de la barra sensor mide 353 cm de longitud. La barra puede ser colocada por encima o por debajo de la televisión, y debe centrarse. Si está colocada por encima, el sensor debe estar alineado con la parte delantera de la televisión, y si coloca en la parte inferior, debe alinearse con la parte delantera de la superficie de la televisión en la que se coloca. No es necesario señalar directamente a la barra sensor, pero apuntar significativamente fuera de la barra de posición perturbará la capacidad de detección debido al limitado ángulo de visión del Wiimote.

El uso de la Barra de Sensores permite al Wiimote ser utilizado como un dispositivo de señalamiento preciso de hasta 5 metros (aprox. 16 pies) de distancia de la barra. El sensor de imagen del Wiimote se utiliza para localizar los puntos de luz de la Barra con respecto al campo de visión del Wiimote. La luz emitida desde cada extremo de la Barra de Sensores se centra en el sensor de imagen que ve la luz brillante como dos puntos separados por una distancia de "mi" en el sensor de imagen. La segunda distancia "m" entre los dos grupos de emisores de luz de la barra sensor es una distancia fija. A partir de estas dos distancias y mi m, el procesador de la consola Wii calcula la distancia entre el Wiimote y la barra de sensores utilizando la triangulación. Además, la rotación del Wiimote con respecto al suelo también puede ser calculada a partir del ángulo relativo de los dos puntos de luz en el sensor de imagen. Los juegos pueden ser programados para detectar si el sensor de imagen está cubierto, lo que fue demostrado en un mini juego de *Smooth Moves*, donde si el jugador no descubre el sensor, la botella de champán que el mando a distancia representa no se abre.

La Barra Sensor es necesaria cuando el Wiimote está controlando movimientos arriba-abajo, izquierda-derecha de un cursor en la pantalla del televisor para apuntar a las opciones de menú u objetos como los enemigos en un juego. Debido a que la Barra de sensores también permite calcular la distancia entre el Wiimote y la sensibilidad de la Barra, el Wiimote también puede controlar el lento movimiento adelante-retroceso hacia un objeto en un juego en 3 dimensiones. El movimiento rápido hacia delante o hacia atrás, como los puñetazos en un juego de boxeo, está controlado por los sensores de aceleración. Usando estos sensores de aceleración (que actúan como sensores de inclinación), el Wiimote también puede controlar la rotación de un cursor u otros objetos.

El uso de un sensor infrarrojo para detectar posición puede causar algunos problemas cuando otras fuentes de infrarrojos se encuentran alrededor, como bombillas incandescentes o velas. Esto puede ser fácilmente mitigado por el uso de luces fluorescentes alrededor de la Wii, ya que emiten poca o ninguna luz infrarroja. Usuarios innovadores han utilizado otras fuentes de luz IR como sustitutos Sensor Bar, como un par de linternas y un par de velas. Tales sustitutos de la Barra de Sensores ilustran el hecho de que un par de luces estáticas

proporcionan una calibración continua de la dirección que el Wiimote está apuntando y su ubicación física en relación con las fuentes luminosas. No hay manera de calibrar la posición del cursor en relación con la que el usuario está señalando con el controlador sin las dos fuentes estables de referencia de la luz proporcionada por la Barra de Sensores o sustitutos.

Los LEDs pueden verse a través de algunas cámaras y otros dispositivos con un mayor espectro visible que el ojo humano.

La posición y seguimiento del movimiento del Wiimote permite al jugador imitar las acciones reales de juego, como blandir una espada o una pistola con objetivo, en lugar de simplemente pulsando los botones. Uno de los primeros videos de marketing mostró actores mimetizando acciones como la pesca, cocina, tocar la batería, dirigiendo un cuarteto de cuerda, disparando un arma de fuego, luchando con espada, y la realización de cirugía dental.

Feedback del controlador

El Wiimote dispone de funcionalidad de audio y vibración. El volumen se puede cambiar o silenciarse con el botón "Home" y seleccionar el correspondiente icono de control en la parte inferior de la pantalla. Cuando el altavoz está en posición de silencio, los efectos de sonido que desempeña a través de él serán desviados a través de los altavoces de la televisión. La característica de vibración también puede ser encendida o apagada utilizando el Menú Home.

1.3. Memoria

El Wiimote contiene un chip de 16 KB EEPROM donde una sección de 6 kilobytes puede ser libremente leída y escrita por el host.

1.4. Alimentación

El Wiimote utiliza dos pilas AA como fuente de energía, que pueden alimentar el Wiimote durante 60 horas usando sólo la función de acelerómetro y 25 horas utilizando acelerómetro y puntero.

1.5. Bluetooth

El mando usa bluetooth para comunicarse con la consola. Además también es detectado por otros dispositivos con bluetooth, como ordenadores personales.

Aplicación para la detección de caídas.

2. Algunas aplicaciones de investigación que utilizan el wiimote.

Debido al bajo coste del wiimote y a todo su potencial son varias las empresas que le están utilizando para algunas investigaciones, los siguientes puntos describen algunas aplicaciones que se están desarrollando gracias al wiimote.

2.1. Aplicación para la rehabilitación física y neurológica de pacientes.

El centro de investigación CEDETEL [2]se encuentra desarrollando una serie de aplicaciones para rehabilitación y aumento de las capacidades cognitivas en personas discapacitadas física o neurológicamente.

Estas aplicaciones permitirán al personal médico disponer de otras técnicas de evaluación y control de pacientes. El Wiimote se emplea como dispositivo para capturar y monitorizar movimientos, ofreciendo al paciente una terapia nueva, intuitiva y atractiva, con la que se podrá monitorizar los ejercicios y evaluar el grado de mejora de forma automática.

Una de estas aplicaciones consiste en una serie de juegos que provocan en los niños un alto nivel de concentración, al mismo tiempo que desarrollan una actividad lúdica, lo que se traduce en una terapia muy productiva y divertida la cual permite a los pacientes más pequeños realizar su rehabilitación mientras se divierten.

Otro de los proyectos también muy interesante de este mismo centro de investigación, es la aplicación contra fobias, su filosofía: la única manera de superar una fobia es afrontar los miedos.

El proyecto se apoya tan sólo en la cámara del mando y en unas gafas que lleva puestas la persona en cuestión. La cámara captura el movimiento de las gafas de tal forma que el mando sabe en todo momento en que posición está la cabeza, y con ello puede ir adaptando la pantalla que está viendo el usuario. Lo que intenta, en definitiva, es crear un entorno inmersivo en el que más que mirar una pantalla parezca que se está mirando una ventana, que tenga profundidad.

Si el proyecto da los resultados que se esperan, se podría tratar las fobias de forma segura y virtualmente, simulando un entorno real que obligara a las personas a enfrentarse a sus miedos pero sin poner en peligro su seguridad, pues, por ejemplo, a una persona que tenga miedo a las alturas no haría falta subirla a un quinto piso.

2.2. Pizarra electrónica

Aplicación para la detección de caídas.

Las aulas también pueden acabar albergando un Wiimote, pues, tal y como se está investigando la cámara del mando, un lápiz infrarrojo, un cañón y un ordenador son suficientes para convertir una pizarra normal en una pizarra electrónica con todas las oportunidades que ésta ofrece. Y es que, es de sobra conocido que la pizarra electrónica permite innovar en las prácticas docentes y mejorar la motivación y la atención de los alumnos y ofrece nuevas herramientas para atender a la diversidad de los escolares, especialmente a aquellos con dificultades severas o moderadas para el aprendizaje. Y todo por un precio inferior al de las pizarras electrónicas propiamente dichas.

En definitiva, la cámara del mando captura la luz infrarroja del lápiz y va siguiendo su movimiento en la pizarra. Las imágenes que recoge las manda al ordenador, el cual las proyecta a través del cañón. Si se combina con otros programas, como el Simulador de Física que han creado, la sensación es que el ordenador se controla desde la pizarra sin que haya nada más que el cañón, el mando y el lapicero.

Figura 4 Pizarra electrónica con wiimote referencia [9].

2.3. WiiAirBoard

Se trata de realizar trazados en el aire y trasladarlos a un programa de dibujo (como el Paint de Windows), de forma que de la sensación de estar “pintado en el aire”, por decirlo de alguna manera. Basta con mantener pulsado el botón del guante o boli para dibujar (se enciende la luz infrarroja) o soltarlo para no hacerlo.

2.4. WiiHome

WiiHome creado por Jonnhy Lee [9], está pensado para controlar la casa mediante dispositivos domóticos. Ejemplos: encender y apagar una luz, la televisión, activar o desactivar una alarma, etc. Para ello se ha diseñado un interfaz con botones grandes y que permite recorrer virtualmente la casa a través de un mapa para elegir habitación, y una serie

de fotos con las que poder ver las distintas perspectivas de cada una de ellas. En una de estas perspectivas es posible tocar uno o varios elementos para activarlo. Por ejemplo, podemos “tocar” (que he implementado como apagar y encender la luz del guante rápidamente dos veces - una especie de doble clic - utilizando el pulsador) una lámpara y se encenderá, quedando reflejado su estado en la propia perspectiva. Si la volvemos a tocar, se apagará. Además, disponemos de un mapa para elegir habitación, y unas flechas para elegir la perspectiva

Aplicación para la detección de caídas.

Algoritmo de detección de caídas.

1. Introducción

En este capítulo se procederá a explicar las fuentes y los métodos que se han empleado en la elaboración del procedimiento de detección de caídas el cual ha sido nuestro primer objetivo del proyecto.

En la elaboración de este proyecto fin de carrera no dispusimos de medios tecnológicos y sobre todo humanos para realizar las pruebas necesarias y apreciar la fiabilidad de nuestro algoritmo de detección, por lo que se opto por investigar los trabajos y publicaciones que ya estuvieran probadas.

Como primera parte se explicaran las aceleraciones producidas en el cuerpo humano medidas científicamente así como una breve descripción de los detectores de caídas que podemos encontrar en el mercado.

Este capítulo ha sido uno de los fundamentales en el proyecto debido a la falta de información disponible y de investigación en este tema, si bien haciendo una búsqueda intensa se ha podido conseguir información y estudios de gran interés.

Dado los estudios previos detalladlos en este capítulo se vio como mejor opción el uso del mando wiimote, cuyas características se detallan en el capítulo 2.

2. Dispositivos anteriores en la detección de caídas basados en acelerómetros.

2.1. SPEEDY: Un detector de caídas en un reloj de muñeca.

T. Degen, H. Jaeckel, M. Rufer [4], crearon “Speedy” un detector de caídas que funciona gracias a un acelerómetro que se sitúa en un reloj de muñeca.

Utiliza un algoritmo multi-etapa, durante la primera etapa se buscaba una gran velocidad hacia el suelo y esta velocidad seguida de un impacto. Una vez detectado el impacto se esperaba 60 segundos y si después de estos había un periodo de inactividad mayor de 40 segundos se activaba una alarma.

Este dispositivo fue un éxito en cuanto no producía falsas alarmas pero un desastre en cuanto a que solo detectaba caídas de frente, y no era capaz de detectar otras caídas como laterales o marcha atrás.

2.2. Tunstall detector de caídas comercial.

Aplicación para la detección de caídas.

La referencia [5] escrita por K. Doughty, R. Lewis, y A. McIntosh documenta el diseño de un detector de caídas comercial llamado Tunstall, el cual usa un patentado algoritmo de detección de caídas con 2 estados.

Se utilizaron dos sensores el primero de ellos detectaba impactos, mientras que el segundo estimaba la orientación, en resumen cuando se detecta un impacto se comprueba la orientación del individuo durante un periodo anterior al impacto y durante un periodo posterior al impacto si existe un cambio de orientación se activa la alarma.

3. Estudio de las aceleraciones del cuerpo humano medidas con un acelerómetro triaxial en personas mayores.

Para poder encontrar un algoritmo fiable de detección de caídas utilizando un acelerómetro, primero debíamos investigar las aceleraciones presentes en el cuerpo humano de forma natural, el siguiente estudio nos ayuda a comprender los diferentes valores de aceleraciones que se producen en el cuerpo humano dependiendo de la actividad que este realice y de la condición física de las personas.

Cuando una persona se cae y entra en contacto con el suelo el cuerpo sufre unas fuerzas superiores a las que se producen cuando se encuentra realizando una actividad normal. Se predijo que la aceleración de una caída iba a ser mayor que las demás aceleraciones presentes en el cuerpo humano durante actividades normales, se puede ver esta diferencia observando las siguientes gráficas:

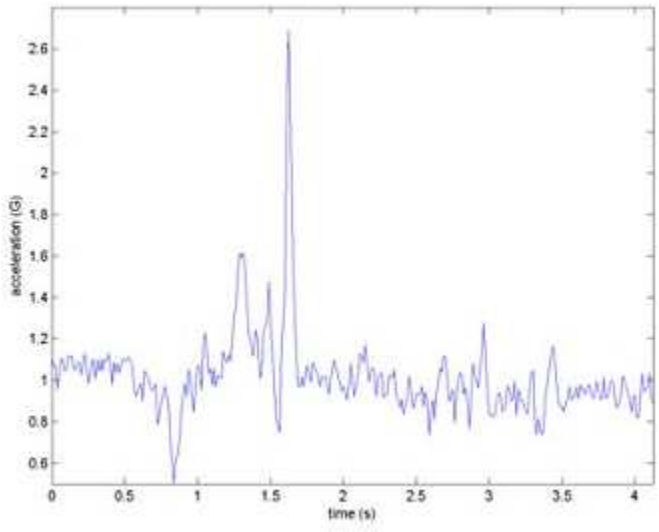


Figura 5. Aceleraciones medidas con un acelerómetro triaxial mientras una persona se sienta en una silla. Gráfica perteneciente a [3]

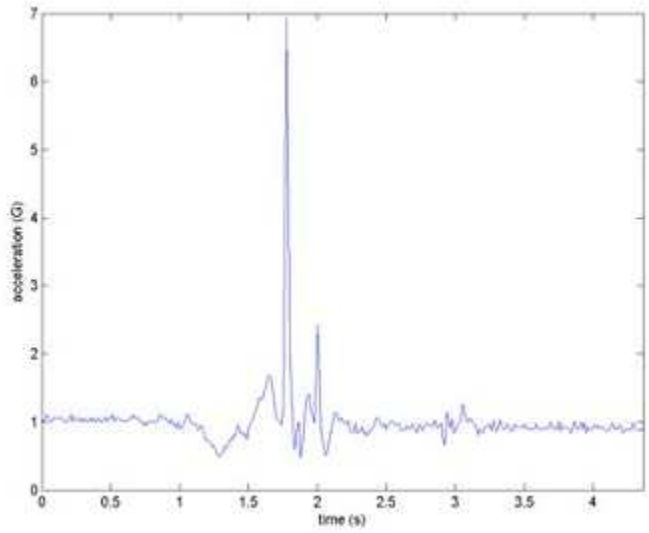


Figura 6. Aceleraciones medidas con un acelerómetro triaxial durante una simulación de caída marcha atrás. Gráfica perteneciente a [3]

Si comparamos la gráfica (a) con la gráfica (b) sin tener en cuenta la escala podemos observar que las gráficas son prácticamente iguales, pero viéndolas teniendo en cuenta la escala se aprecia que durante la caída simulada la aceleración más alta llega hasta casi 7g mientras que las aceleraciones sufridas por el cuerpo cuando una persona se sienta llega hasta 2,6g como vemos las diferencias son considerables.

Observando la gráfica (b), hay varios puntos importantes a destacar: 1) Entre el tiempo t = 1s y t = 1.5s, hay una pequeña depresión, lo que indica un corto período de caída libre, 2) En el tiempo t = 1.75s, hay una gran pico, que indica el impacto en el suelo; 3) Después del pico de impacto, hay una efecto negativo, como la fuerza impacto es absorbida por la cuerpo y el suelo, 4) La inicial y final de acelerómetros lectura se mantiene constante en alrededor de 1 G, como se esperaba. Resultados similares se observaron para caer de lado.

Una vez conocida esta observación de las aceleraciones, lo siguiente que se pensó fue buscar un umbral de aceleración para determinar cuáles de las aceleraciones se podían considerar como peligrosas y cuales estaban dentro del rango normal en de movimientos.

Las personas mayores a menudo se mueven de forma diferente que los más jóvenes y suelen tener menos control sobre la velocidad y movimientos de su cuerpo debido a la reducción de la fuerza muscular debida a la vejez.

Aplicación para la detección de caídas.

Se consideró apropiado utilizar ancianos para realizar las mediciones de la aceleración y así incrementar la robustez de la prueba métodología.

Este artículo describe el desarrollo y ensayo de un umbral basado capaz de discriminar automáticamente entre un movimiento normal del cuerpo y un movimiento "extraño" como pudiese ser una caída utilizando acelerómetros triaxiales.

Los valores de las aceleraciones se adquirirán a partir de caídas simuladas realizadas por personas jóvenes y sanas, y de actividades de la vida cotidiana realizada por personas ancianas en sus propios hogares.

El umbral de aceleraciones permitidas se dedujo a partir de datos empíricos, nos hemos basado en un estudio realizado por A.K. Bourke , J.V. O'Brien , G.M. Lyons para la universidad de Irlanda [1].

En esta investigación se realizaron 2 experimentos uno de ellos con personas ancianas y otro de ellos con personas jóvenes.

En el primer experimento en el que participaron 10 ancianos de edades comprendidas entre 70 y 83 años a cada persona se les mando realizar unas actividades y repetirlas 3 veces, cada uno de ellos llevaba un acelerómetro triaxial colocado primero en el tronco y luego en el muslo. Las actividades realizadas fueron las siguientes:

- sentarse y levantarse de un sillón;
- sentarse y levantarse de una silla de cocina;
- sentarse y levantarse de un asiento de inodoro;
- sentarse y levantarse de un taburete bajo;
- sentarse y levantarse de un asiento de seguridad como puede ser el de un coche;
- sentarse y levantarse de una cama;
- acostarse y levantarse de una cama;
- caminar 10 m;

El valor de aceleración del estudio fue el resultado del cálculo de la norma de las tres aceleraciones en cada eje, esto permitió que se tomaran en cuenta las caídas en todas las posibles direcciones.

El segundo experimento implicó a otros 10 jóvenes de edades comprendidas entre 21 -29 años y pesos entre 67.6 y 85.3 kg, ellos simularon 6 tipos de caídas diferentes.

Las caídas simuladas fueron seleccionadas a partir de una investigación sobre las caídas típicas en ancianos, el estudio realizado por Lord SR, Ward JA, Williams P, Anstey KJ [6], encontró que 82% de las caídas se produjeron cuando las personas se encontraban en posición vertical. La mayoría de causas comunes de caídas se produjeron mientras un anciano se encuentra caminado, se resbala y cae.

Aplicación para la detección de caídas.

Otra estadística en este caso llevada a cabo por O'Neill TW, Varlow J, Silman AJ, Reeve J, Reid DM, Todd C, referencia [7] comprobó que de 180 caídas registradas 160 se produjeron hacia adelante y en el 60% de estas el sujeto se encontraba dando un paso hacia delante con una de las rodillas flexionadas y un pie en el aire, típico movimiento de dar un paso cuando caminamos.
No obstante aunque las caídas más típicas sean hacia delante también se simulo las caídas laterales ya que estas suelen producir un gran impacto en el tronco que suele derivar en fractura cada vez que ocurre.

Teniendo en cuenta todas estas investigaciones las caídas simuladas por estos 10 jóvenes fueron las siguientes:

- Caída hacia delante, con ambas piernas estiradas
- Caída hacia delante, una pierna flexionada.
- Caída hacia atrás, con piernas estiradas.
- Caída hacia atrás, una pierna flexionada.
- Caída lateral derecho, con piernas estiradas
- Caída lateral derecho, con una pierna flexionada.
- Caída lateral izquierda, con piernas estiradas y pierna flexionada.
- Caída lateral izquierda, con una pierna flexionada.

Como era de esperar el estudio comprobó que las aceleraciones registradas durante actividades comunes en la vida diaria fueron más bajas que las registradas durante la simulación de las caídas.

En las gráficas de abajo podemos observar la notable diferencia de aceleraciones dependiendo de la actividad realizada.

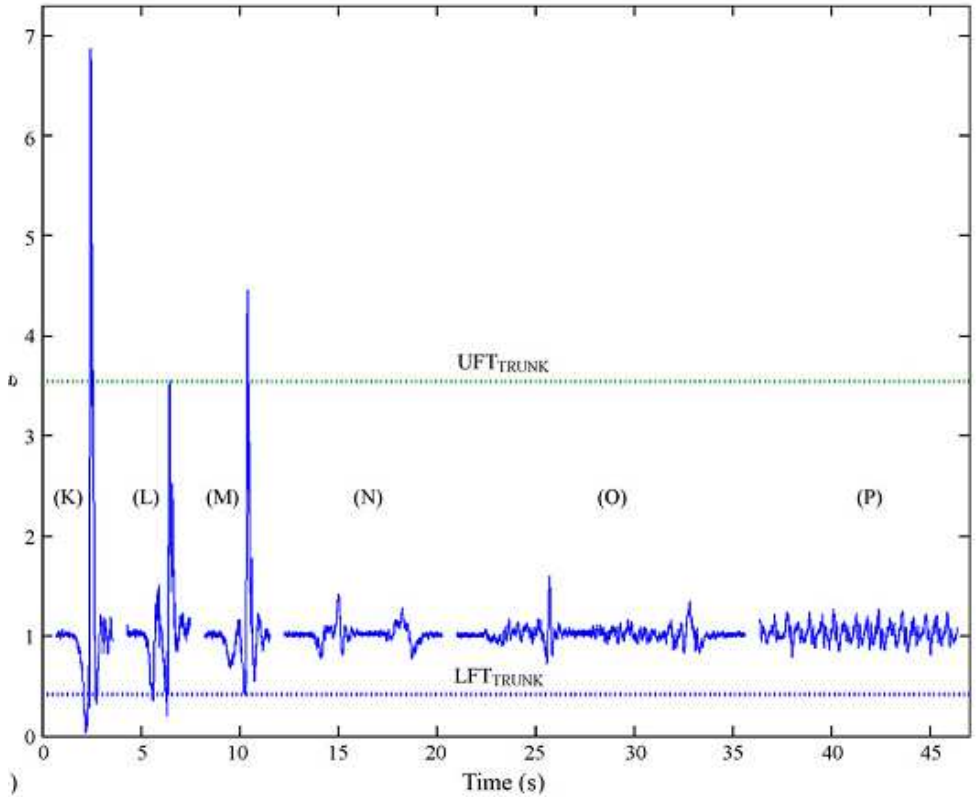


Figura 7. Imagen gráfica perteneciente [1].

(K) Resultado de una típica caída

(L) Resultado de una caída en baja magnitud

(N) Sentarse y levantarse de una silla

(0) Sentarse y levantarse de un asiento de seguridad.

(P) Andando 10 metros.

De los picos más altos de aceleraciones registrados durante la simulación de caídas llamados UPVs, se tomo el valor más bajo, a este valor se le llamó UFT (umbral superior de caída), mientras que de los picos más pequeños de aceleración se tomo el mayor FPV (umbral inferior de caída), ambos umbrales fueron registrados con el acelerómetro situado en el tronco y en el muslo.

También se registraron los puntos de mayor y menor aceleración registrados por cada tarea en particular.

Aplicación para la detección de caídas.

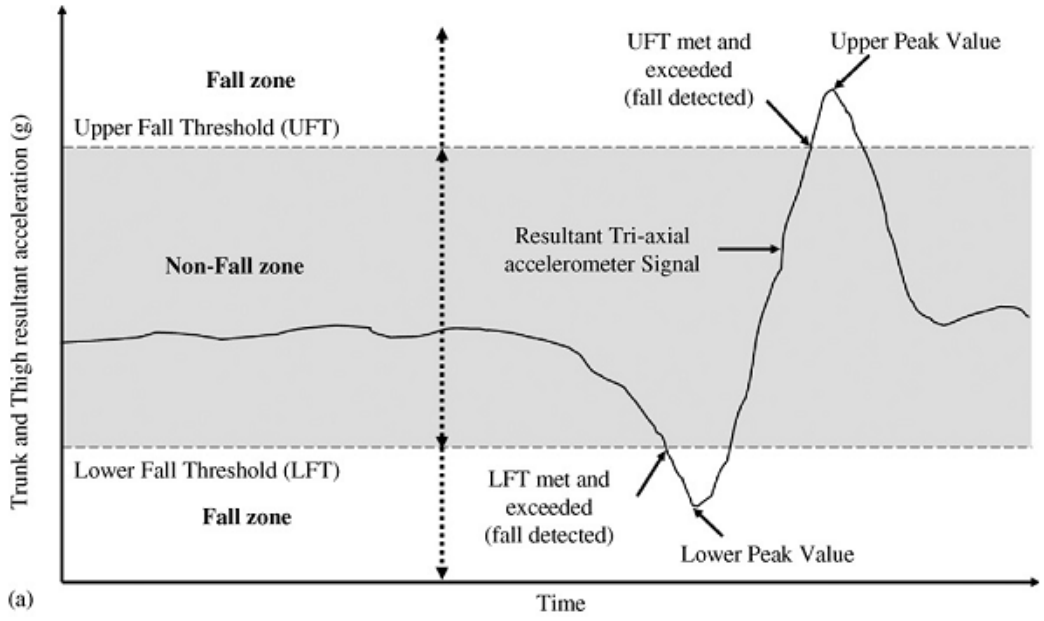


Figura 8 aceleraciones típicas durante una caída. Gráfica perteneciente a [1].

En la imagen (a) podemos observar la función de aceleración típica durante una caída y como esta supera los umbrales de zona de no caída calculados.

Los valores registrados durante el estudio se encuentran resumidos en la siguiente tabla.

	Tronco		**Muslo**	
	UFT	**LFT**	**UFT**	**LFT**
Umbral (g)	3.52	0.41	2.74	0.60

	Tronco		Muslo	
	Mayor UPV (g)	Mayor LPV (g)	Mayor UPV (g)	Mayor LPV (g)
Sentarse en un sillón.	2.60	0.11	3.23	0.36
Sentarse en una silla de la cocina.	3.16	0.21	3.75	0.31
Sentarse en el retrete.	2.41	0.30	3.76	0.30
Salir y entrar de un coche.	3.02	0.16	7.19	0.11
Sentarse en un taburete	2.95	0.24	2.50	0.55
Sentarse en la cama.	2.06	0.39	3.09	0.28
Recostarse en la cama.	2.38	0.16	4.07	0.43
Andar 10 metros.	1.99	0.61	6.61	0.06

Tabla valores registrados realizando ciertas actividades en el estudio referencia [1]

Una vez procesados estos datos se comprobó el porcentaje de acierto del algoritmo, es decir tras una serie de pruebas conocer qué porcentaje de acciones no peligrosas era capaz de

identificar y haber cuantas caídas podía procesar. Ambas pruebas fueron realizadas para el tronco y el muslo.

Para el tronco el umbral LFT consiguió identificar el 91,25% de las actividades comunes como no caídas mientras que el UFT consiguió identificar como no caídas el 100 de las pruebas. El porcentaje de acierto se resumen en la tabla de abajo en ellos se puede observar que el lugar optimo de colocación del acelerómetro es el tronco.

	Tronco		**Muslo**	
	UFT	**LFT**	**UFT**	**LFT**
%Acierto	100	91.25	83.33	67.08

	Tronco		Muslo	
	%acierto UPV	%acierto LPV	%acierto UPV	%acierto LPV
Sentarse en un sillón.	100	83.3	86.7	80
Sentarse en una silla de la cocina.	100	86.7	90	90
Sentarse en el retrete.	100	93.3	96.7	70
Salir y entrar de un coche.	100	86.7	60	36.7
Sentarse en un taburete	100	93.3	100	93.3
Sentarse en la cama.	100	96.7	93.3	83.3
Recostarse en la cama.	100	90	90.0	60
Andar 10 metros.	100	100	50.0	23.3

La caída que produjo el valor más pequeño en el tronco de UPV fue la caída lateral con la rodilla flexionada este valor fue de 3.52g este valor fue tomado como UFT del tronco. La actividad común que produjo un UPV mayor fue sentarse en la silla de la cocina, en el grafico de abajo podemos observar el rango de magnitudes en g obtenidos durante el estudio tanto en caídas como en actividades comunes.

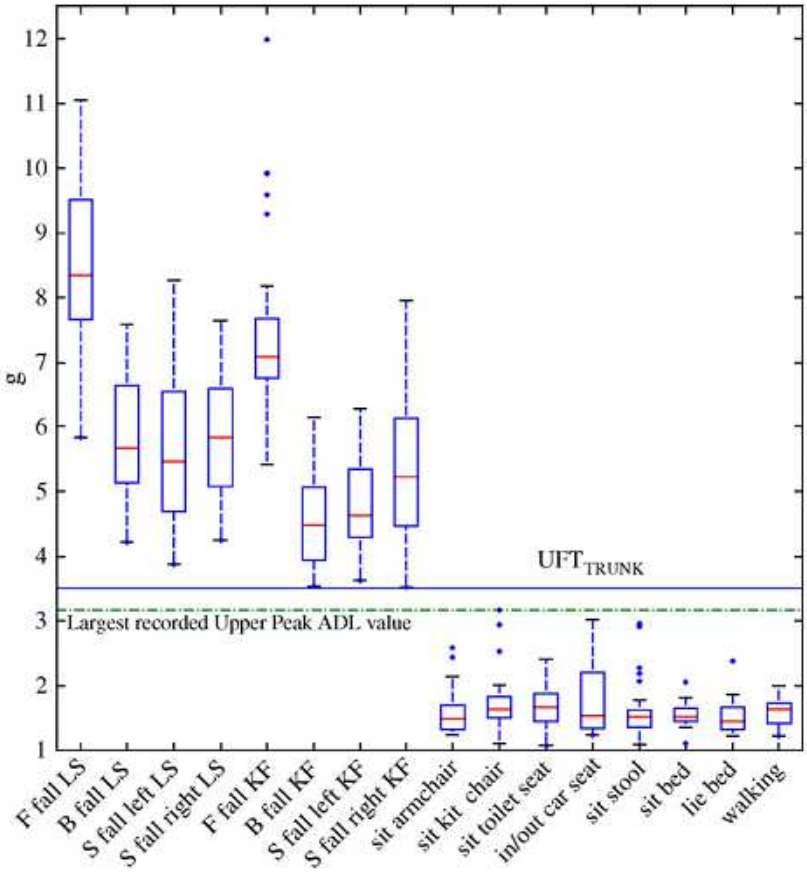


Figura 9 Gráfica perteneciente a [1],

3.1. Observación del estudio, cálculo de umbrales.

Hay que tener en cuenta que para este estudio se realizaron caídas simuladas utilizando personas jóvenes y que en la simulación los participantes no evitaron caerse, en una caída real es de esperar que el sujeto realice movimientos para evitar la caída.

3.1.1. Umbrales registrados.

Como conclusión del estudio podemos decir que las aceleraciones más pequeñas durante una caída fueron en torno al 3,5G pero otras fueron de varios G mayores mientras que las actividades normales suelen producir aceleraciones de 1 a 2,5G aunque a veces existen actividades como correr o sentarse que pueden supérados.

Al finalizar el estudio se dedujo que el umbral de movimientos normales en el cuerpo humano debía estar entre [3.52g, 0.41g] los valores de aceleración fuera de este intervalo se podían considerar como posibles caídas.

Para conseguir un detector de caídas lo más fiable posibles debemos eliminar todos los posibles los falsos positivos, si solamente se tomara como referencia los valores del umbral de aceleración se producirían falsos positivos debido a movimientos como bajar escaleras, saltar, o sentarse en el sofá dejándose caer, por ello se necesita otra manera de distinguir posibles caídas que únicamente utilizar un umbral de aceleración.

4. Algoritmo final de detección de caídas

Como algoritmo de detección de caídas hemos tomado el algoritmo creado por [referencia 6] para la universidad de California y presentado en la conferencia anual de Ingeniería para la medicina y la biología en Shanghái (septiembre de 2005).

Decidimos utilizar este algoritmo ya que utilizaba un acelerómetro triaxial y aportaba las mismas ideas de los umbrales de aceleración del punto anterior. Pero además de todo esto lo realmente innovador este algoritmo es que tiene en cuenta la orientación del cuerpo.

El diagrama de flujo de este algoritmo de detección de caídas es el siguiente:

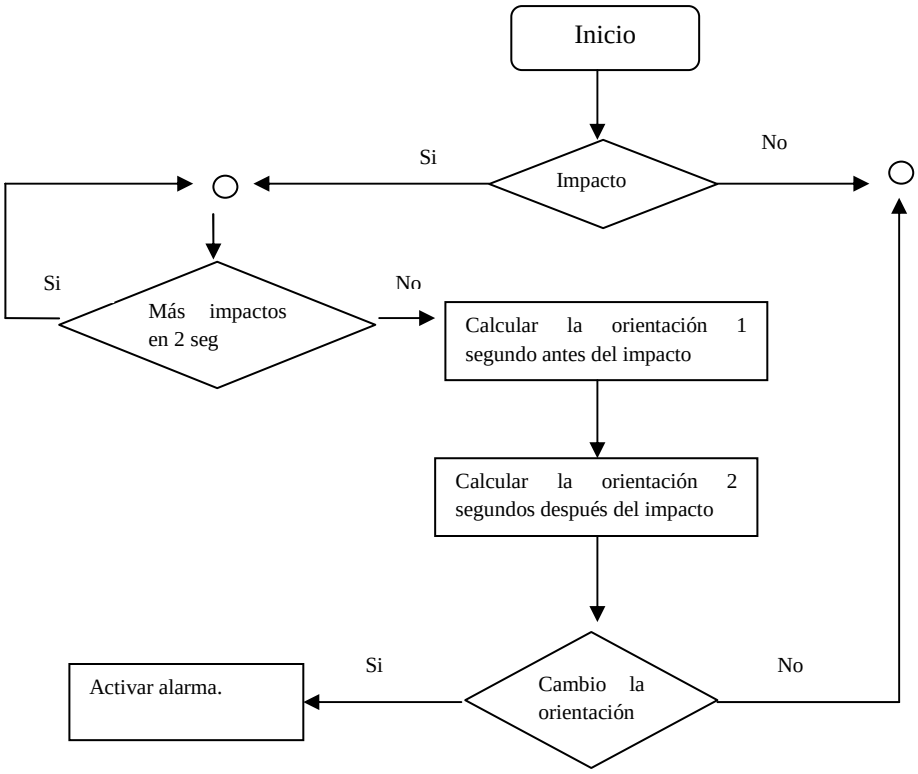


Figura 10. Algoritmo de detección de caídas [3]

Como podemos ver en el diagrama de flujo lo primero que se comprueba es si hay impactos, en este punto entra en juego los umbrales de aceleración, se considera que se ha producido un impacto si en algún momento se han superado los umbrales máximo o mínimo de aceleración (UFV, LFV). Una vez identificado un impacto, se comprueban que no se produzcan más impactos en 2 segundos, gracias a esta observación se podrían identificar caídas donde se sufrieran más de un impacto en un periodo corto de tiempo como puede ser el caso de caerse por las escaleras.

Después se tiene en cuenta la orientación que tenía el sujeto un segundo antes de sufrir el primer impacto y la orientación del sujeto 2 segundos después del impacto, si estas 2 orientaciones son diferentes entonces se considera que se ha producido una caída.

En el algoritmo original los valores de umbrales de aceleración se tomaron de forma aproximada, la modificación que se ha hecho de este algoritmo para este proyecto ha sido cambiar los umbrales propuestos por el algoritmo original, y usar los umbrales calculados por el estudio [1], es decir en nuestro algoritmo toda aquella aceleración que sean mayor que 3,52g o aquellas aceleraciones menores que 0,41g serán considerados como un impacto.

Aplicación para la detección de caídas.

En estos momentos podemos responder a 2 de los objetivos que nos planteábamos, las características físicas y técnicas que debe cumplir nuestro sensor de movimiento para poder desarrollar el algoritmo aquí descrito, estas características son las siguientes:

- Para llevar a cabo el objetivo anterior y poder implementar el algoritmo antes descrito necesitamos un **acelerómetro triaxial**.
- **Tiene que ser capaz de detectar aceleraciones en un rango de [0.41,3.56]g** ya que han sido los valores umbrales de aceleración.
- Tecnología inalámbrica, el acelerómetro debe funcionar sin ningún tipo de cables.
- Duración de la batería, el acelerómetro para que sea viable debe funcionar de manera independiente varias horas.
-

Además el acelerómetro debe cumplir las siguientes características físicas.

- El tamaño y el peso, este acelerómetro va ser colocado en el cuerpo de los pacientes por lo que su peso y tamaño deben ser reducidos, si es posible poder integrar el sensor en la ropa.

4.1. Medidas de aceleración con el wiimote.

En este punto hemos aplicado las investigaciones de los puntos anteriores al acelerómetro de un mando wiimote, tenemos que comprobar es que los valores de las aceleraciones medidas realizando las mismas actividades que las investigaciones nos dan valores próximos a los de la tabla del punto 3.

En nuestro caso no disponemos de ancianos para hacerles las pruebas oportunas, pero se hizo las pruebas a tres personas de edades comprendidos entre 23 y 50 años. Estas pruebas han sido andar, y sentarse y levantase, los valores obtenidos con un wiimote situado en tronco han sido las siguientes:

Andando	UPV	LPV	UPV	LPV	UPV	LPV
Sujeto 1	1,70	0,60	2,036	0,574	1,76	0,65
Sujeto 2	1,80	0,61	1,648	0,665	1,698	0,6269
Sujeto 3	1,72	0,62	1,739	0,58	1,927	0,541

Sentarse y levantarse	UPV	LPV	UPV	LPV	UPV	LPV
Sujeto 1	2,63	0,65	1,60	0,73	1,997	0,5728
Sujeto 2	1,3613	0,829	1,33	0,797	1,363	0,789
Sujeto 3	1,477	0,587	1,486	0,723	1,363	0,723

El UFT obtenido en la prueba de andar utilizando el wiimote ha sido 2,036 en el estudió fue de 1,99 g, el LPV más alto de la prueba de andar con el wiimote ha sido 0,665g mientras que

en el estudio fue de 0,62g. Observamos los valores son muy próximos entre sí hay que tener en cuenta que los datos medidos por el wiimote han sido a personas más jóvenes y con mayor movilidad por lo que cabe un error en la forma de moverse, pero teniendo en cuenta esto los valores son muy próximos entre sí por lo que podemos utilizar este algoritmo con el acelerómetro triaxial que trae incorporado el mando wiimote.

Veamos ahora las gráficas que presenta el mando wiimote representando la norma.

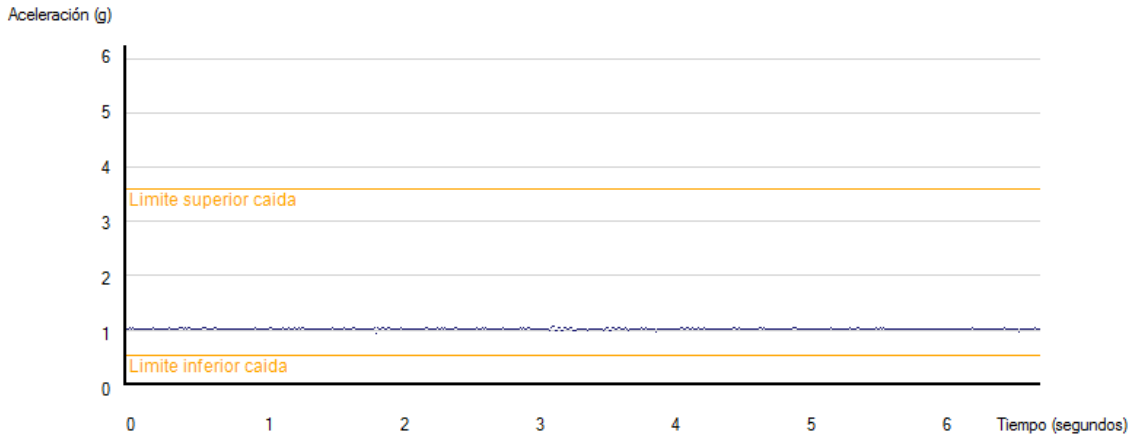


Figura 11. Gráfica de la norma de las aceleraciones de un wiimote en estado estacionario.

En esta caso se ha dibujado la gráfica mientras el mando se encuentra quieto como vemos el valor de la norma es 1, también quedan reflejados los límites de zona de no caída calculados anteriormente.

Ejemplos de gráficas de aceleración producidas con el wiimote realizando distintas actividades.

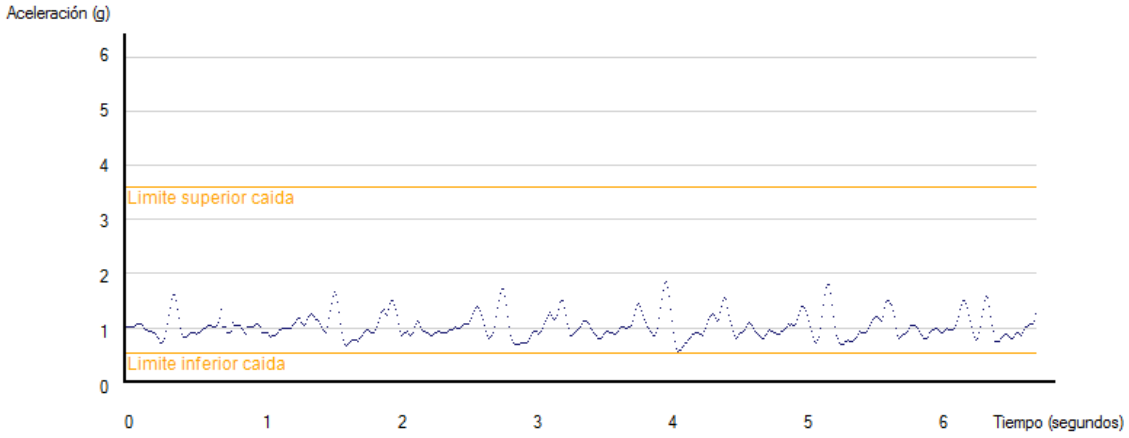


Figura 12. Gráfica aceleraciones wiimote persona andando 10 metros.

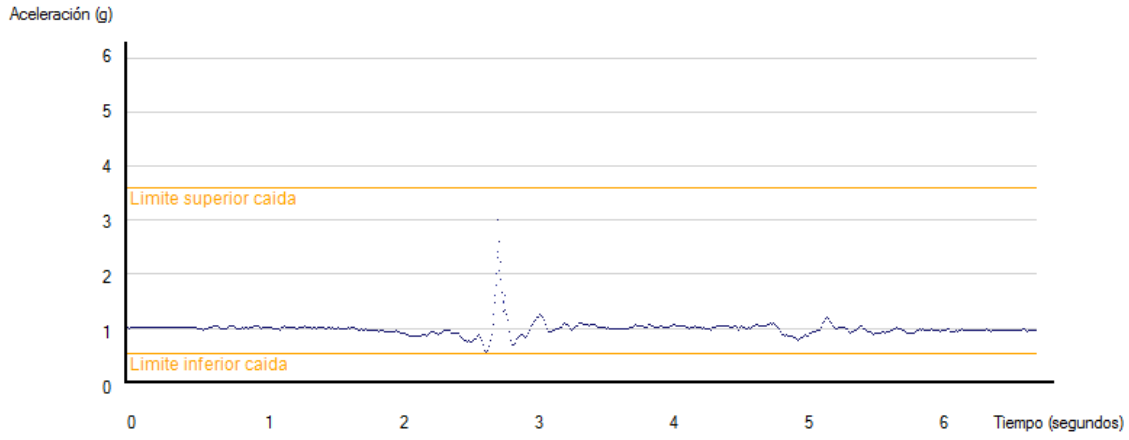


Figura 13. Gráfica aceleraciones wiimote situado en el tronco sentándose en una silla.

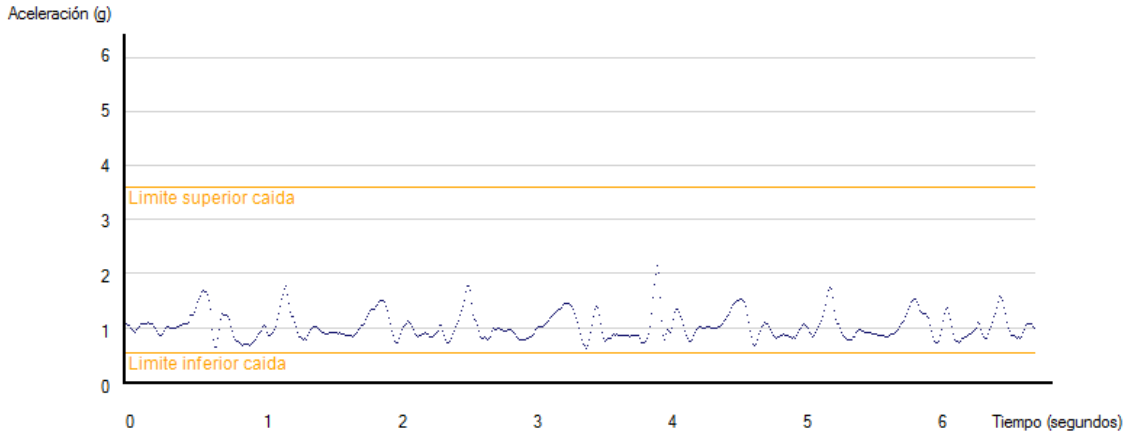


Figura 14. Gráfica aceleraciones wiimote situado en el tronco subiendo escaleras.

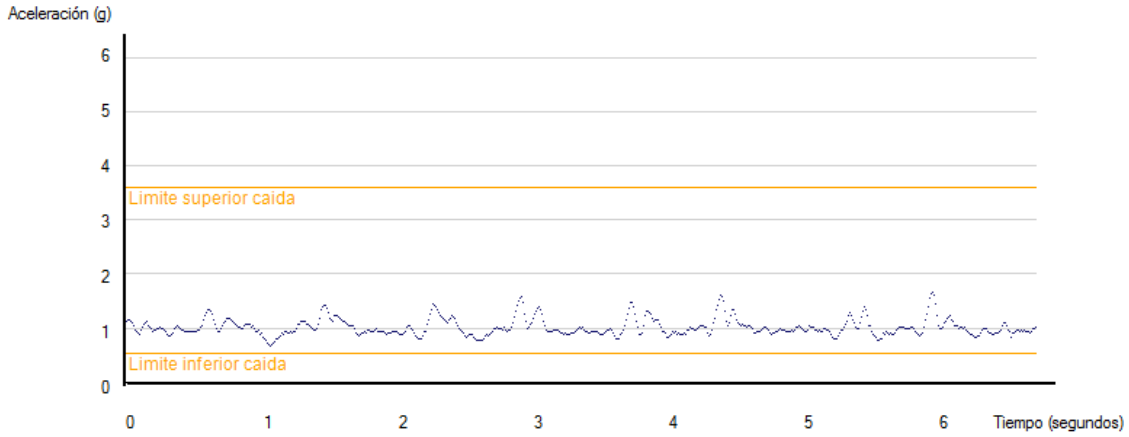


Figura 15. Gráfica aceleraciones wiimote bajando escaleras.

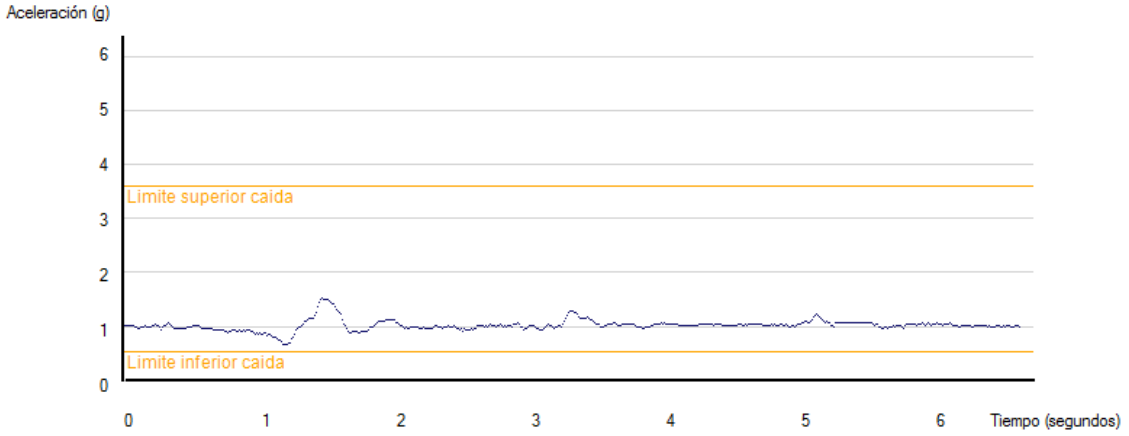


Figura 16. Gráfica aceleraciones wiimote tumbándose en una cama.

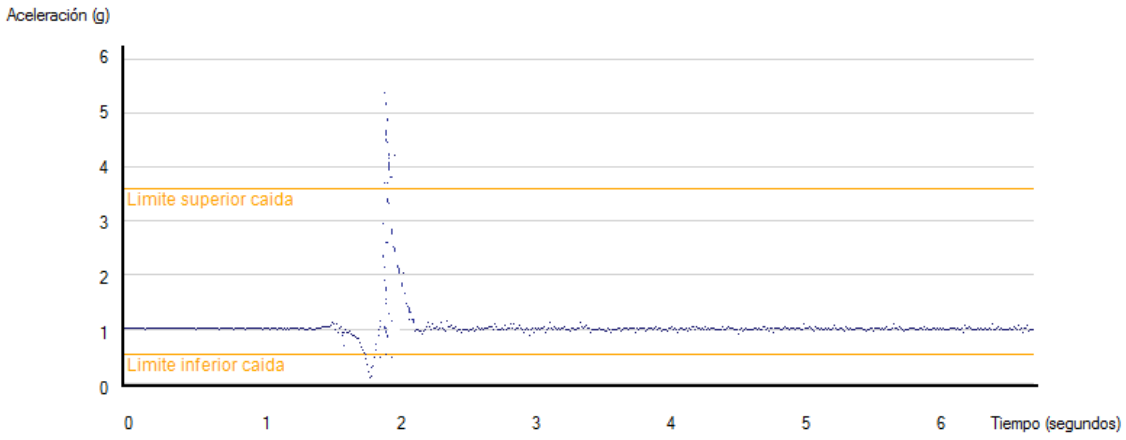


Figura 17. Gráfica de la norma de las aceleraciones de un wiimote durante una simulación caída de frente.

En esta última gráfica se aprecia cómo se han superado los umbrales de aceleración esto indica que se ha detectado un impacto después de esto si las orientaciones varían tendremos una caída.

Aplicación para la detección de caídas.

Análisis.

1. Introducción.

El análisis consiste en la comprensión y modelado de la aplicación y del dominio en el cual se desempeña. La entrada inicial de la fase de análisis es una descripción del problema que hay que resolver y proporciona una descripción general del sistema propuesto.

Inicialmente se estudia la especificación del sistema, teniendo en cuenta el contexto en el cual será aplicado.

El objetivo de este apartado es reconocer los elementos básicos del programa tal como lo percibe el usuario. La salida del análisis es un modelo formal, que muestra de una forma abstracta y resumida, los diferentes elementos del dominio, y que sirve como base para la fase de diseño del sistema.

2. Requisitos Software

El objeto de este apartado es el de identificar la funcionalidad de la aplicación que vamos a desarrollar. Para ello se deben obtener los requisitos que esta debe cumplir, obtenidos estos a través de las indicaciones de tutor del proyecto.

El objeto del PFC es obtener una aplicación que permita la detección de caídas en pacientes de forma que sea lo menos intrusiva y molesta posible.

El siguiente apartado proporciona una descripción general del proyecto que nos permite introducirnos en la funcionalidad que debe desempeñar y por tanto obtener la especificación de requisitos. Se define también el contexto de uso, así como perfil de usuario final de la aplicación y perfil de los pacientes que se van a monitorizar.

Posteriormente, comprendida la problemática a resolver, pasaremos a describir uno a uno los requisitos funcionales y no funcionales, actores y casos de uso, hasta obtener un modelo conceptual de la aplicación a desarrollar.

2.1. Descripción general de la aplicación.

Se han realizado entrevistas tanto al tutor del proyecto como a miembros médicos de una residencia de ancianos, de ellas se han podido extraer multitud de datos que permitieron definir la funcionalidad de la aplicación. A continuación realizaré una explicación general de la aplicación.

La aplicación realizada esta orientada a la monitorización pasiva de pacientes, podrá ser utilizada en complejos residenciales como en las propias casas de los pacientes.

Aplicación para la detección de caídas.

Los pacientes llevaran adosado en su cuerpo un sensor, en el caso de este proyecto será un mando wiimote, este mando debe estar situado en la cintura del paciente.

El mando wiimote leerá los valores de la aceleración del cuerpo, esta información será recogida a través de tecnología inalámbrica bluetooth y evaluada por la aplicación que estará situada en un PC.

La aplicación tendrá la capacidad de a partir de los datos recibidos por parte del wiimote reconocer una caída, y cuando esto se produzca enviará una notificación al un servidor a través de internet. Gracias al servidor el personal médico podrá conocer el estado de cada uno de los pacientes.

Como se ha indicado anteriormente, esta PFC es originaria del proyecto AIVI, el cual nos debía proporcionar la especificación de la interfaz de la Central. Finalmente por retardos en la entrega de la documentación y de la Central en sí, decidimos realizar una aplicación que nos resolviera esta necesidad. De la misma forma, los diccionarios de datos sobre la configuración de los pacientes, el registro de datos de los sensores, y los cambios en las notificaciones también son simulados, no cumpliendo la especificación del proyecto AIVI.

A continuación mostramos una ilustración que define toda la infraestructura de la solución, desde los sensores del paciente, hasta el personal médico que recibe la información.

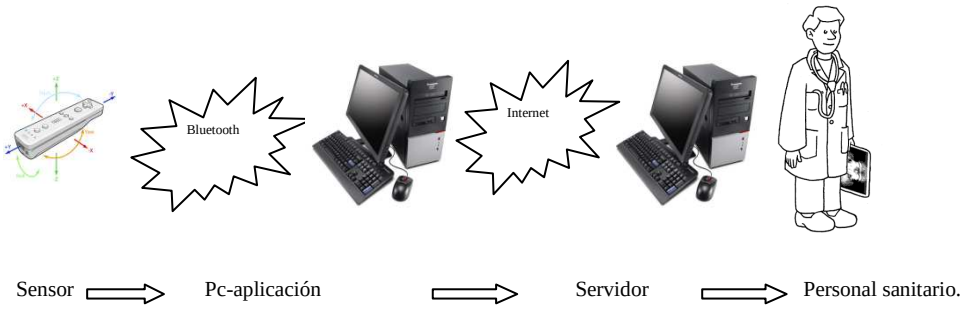


Figura 18. Tecnología disponible para el proyecto.

2.2. Contexto de aplicación.

La detección de caídas en pacientes se podrá realizar tanto en las casas particulares como en una residencia. Cuando la aplicación funcione en las casas particulares y esta detecte una alarma se enviará vía internet una notificación a un servidor central donde las personas encargadas actuarán en consecuencia.

Mientras que si la aplicación funciona en una residencia esta puede monitorizar a la vez a varios pacientes, basta con entregarles a cada uno de ellos un mando wiimote, en este caso no hará falta el envío de notificación vía internet ya que los propios trabajadores de la

residencia podrán saber en todo momento el estado de sus pacientes mirándolo en el PC-aplicación.

La aplicación además de la detección de caídas tiene una funcionalidad añadida que es la llamada de emergencia. Un paciente puede pulsar un botón en el mando Wiimote y de este modo se enviará una alarma al personal médico.

La aplicación también tendrá la posibilidad de eliminar falsas alarmas ya sean producidas por una falsa caída o bien por pulsar de forma involuntaria el botón de llamada de emergencia.

2.2.1. Servidor

El servidor está compuesto por un servicio web que se encargara principalmente de dos tareas. Por una parte dispone de la información de todos los sensores y de todos los pacientes, y proporcionará la configuración de cada uno de los pacientes. Esta configuración se registrará en un fichero XML, que dispone de la información del paciente, con los sensores que le van a monitorizar Por otra parte, se encargará de recoger las posibles alarmas que le llegan de los PC que controlan a cada paciente.

De esta forma el personal médico estado real del paciente en cada momento.

2.2.2. PC-aplicación.

Será el PC que sobre el que esté funcionando la aplicación, estará conectado vía bluetooth a los diferentes wiimotes y si detecta una caída o una llamada de emergencia será él quien envíe la notificación al servidor.

El PC-aplicación debe disponer de dos tipos de conectividad la primera bluetooth para poder conectarse a al wiimote y la segunda debe estar siempre conectado a internet para enviar las notificaciones al servidor, esta segunda conectividad solo es necesaria en el caso de que estemos monitorizando a pacientes en su propia casa ya que si es en un recinto residencial el propio PC-aplicación puede mostrar las notificaciones la Personal Médico.

La diferencia entre PC-Aplicación y servidor esta clara el servidor simplemente recibe las alarmas mientras que el PC-Aplicación será el encargado de detectar las posibles caídas y crear y enviar las alarmas.

El PC aplicación debe estar siempre a una distancia comprendida entre 10 y 200 metros del wiimote dependiendo de la capacidad de la antena bluetooth instalada en el PC aplicación ya que si no es así se perderá la conexión.

Aplicación para la detección de caídas.

2.2.3. Sensor Caídas

El Sensor Caídas que disponemos es un mando Wiimote.

3. Roles

De las entrevistas con los miembros de la residencia y con el tutor se extrajeron dos roles principales que son el paciente y el personal sanitario.

El paciente es el que será monitorizado quien llevará consigo el wiimote. Si este va ser monitorizado en su propia casa deberá de recibir una pequeña formación de cómo colocarse el dispositivo y como activar y desactivar la aplicación. Si el paciente va estar en una residencia no es necesario que conozca el uso de la aplicación ya que el personal médico será el encargado de suminístrasela.

Personal médico será quien configure las características del los pacientes en el PC y quien evalué la correcta colocación de los dispositivos. No será personal altamente cualificado pero se supone que recibirán la formación específica para el manejo de la aplicación.

4. Requisitos no funcionales.

NFR-001	**Confidencialidad datos paciente**
Dependencias	Ninguno
Descripción	El sistema deberá *permitir que solo el personal sanitario autorizado pueda acceder a los datos del servidor central.*

NFR-002	**Tecnología Bluetooth e internet.**
Dependencias	Ninguno
Descripción	El sistema deberá *disponer de las tecnologías de bluetooth para la conexión con los sensores e internet para la comunicación con el servidor central.*

NFR-003	Ergonomía
Dependencias	Ninguno
Descripción	El sistema deberá *facilitar la ergonomía en el uso de la aplicación, con el objeto que este pueda ser utilizado por personal no especializado.*

NFR-004	Leds wiimote
Descripción	El sistema deberá *permitir conocer el estado de la detección de caídas mirando los leds encendidos del wiimote siguiendo estos patrones:* *-Un solo led encendido: aplicación funcionado correctamente.* *-Dos leds encendidos: se ha enviado una alarma al servidor central* *-Cuatro leds parpadeando: el wiimote está en tiempo de espera para desactivar una alarma.* *-Ningún led encendido: el wiimote no se encuentra conectado.*

NFR-005	Vibración Wiimote
Dependencias	Ninguno
Descripción	El sistema deberá *hacer vibrar el wiimote durante unos segundos cuando haya detectado una posible caída, de este modo el paciente podrá saber que se ha activado una alarma.*

NFR-006	Botones Wiimote
Descripción	El sistema deberá *permitir tener cierta funcionalidad en algunos botones del mando wiimote, que serán los siguientes:* *-Botón A al pulsar este botón se eliminaran las alarmas.* *-Botón cruceta, al pulsar este botón se producirá una llamada de*

	emergencia.

NFR-007	**Colocación del Wiimote**
Descripción	El wiimote debe estar colocado en la cintura del paciente.

5. Funcionalidad de la aplicación.

Analizadas los condicionantes externos que intervienen en la aplicación, vamos a realizar una descripción detallada del funcionamiento de esta, con el fin de obtener la especificación de requisitos.

En un comienzo el personal Sanitario deberá realizar la conexión bluetooth de cada uno de los wiimotes caídas al PC-Aplicación. Una vez conectados los wiimotes el personal sanitario iniciará la aplicación.

Con la aplicación iniciada esta mostrará todos los wiimotes que se encuentran conectados, el siguiente paso será asociar cada uno de los wiimotes a un paciente concreto. Una vez que los wiimotes estén asociados a cada paciente, se les colocará el dispositivo y estarán listos para iniciar la detección de caídas.

Cuando este iniciada la detección de caídas el PC-Aplicación estará leyendo datos continuamente del Wiimote, aplicará con ellos el algoritmo de detección y si este detecta alguna posible caída el PC-Aplicación enviará una alerta la servidor.

Como no existe ningún algoritmo de caídas perfecto se da la funcionalidad de poder eliminar una alarma si el paciente no considera necesario ninguna atención. Para ello si se produce una alarma y el paciente quiere desactivarla podrá hacerlo pulsando un botón en el mando Wiimote.

5.1. Requisitos Funcionales

FRQ-001	**Información Caída**
Dependencias	Ninguno

Descripción	El sistema deberá *permitir conocer al personal médico que paciente ha sufrido la caída en tiempo real.*

FRQ-002	**Alarma Caída**
Dependencias	FRQ-0008 Detección de caídas
Descripción	El sistema deberá *enviar una alarma al servidor central en cuanto una posible caída sea detectada.*

FRQ-003	**Desactivar alarma**
Dependencias	FRQ-0003 Alarma Caída
Descripción	El sistema deberá *permitir al paciente desactivar la alarma si él considera que no necesita ninguna atención.*

FRQ-004	**Llamada de emergencia**
Dependencias	Ninguno
Descripción	El sistema deberá *permitir al paciente realizar una llamada de emergencia cuando él considere que necesita atención médica.*

FRQ-005	**Estado Sensores**
Dependencias	Ninguno
Descripción	El sistema deberá *permitir conocer en cualquier momento el estado actual de todos los sensores conectados, incluyendo estado de la batería del sensor.*

FRQ-006	**Asociar paciente**
Dependencias	Ninguno
Descripción	El sistema deberá *asociar un paciente a cada Wiimote para conocer que Wiimote corresponde a que paciente.*

FRQ-007	**Detección de caídas**
Dependencias	Ninguno
Descripción	El sistema deberá *aplicar un algoritmo de detección de caídas con los datos leídos del wiimote.*

FRQ-008	**Parar Aplicación**
Dependencias	Ninguno
Descripción	El sistema deberá *iniciar o detener la aplicación cuando los usuarios lo consideren oportuno.*

FRQ-009	**Información envió servidor alarma.**
Dependencias	Ninguno
Descripción	El sistema deberá *permitir al paciente conocer cuando ha sido enviada una alarma al servidor central.*

FRQ-010	**Información alarma paciente**

Aplicación para la detección de caídas.

Dependencias	**FRQ-002 Alarma Caída**
Descripción	El sistema deberá *poseer algún mecanismo para que un paciente pueda saber que una alarma ha sido activada y darle cierto tiempo para desactivarlo antes que la alarma llegue al servidor central.*

6. Casos de uso.

En este apartado explicaremos los diferentes casos de uso que hemos obtenido de los requisitos. Para explicar esto primero nos centraremos en los diferentes actores que están descritos en la aplicación y sus principales roles y características

6.1. Actores.

Los actores participantes en los casos de uso son los siguientes:

ACT-001	**Personal Medico**
Descripción	Este actor representa *el usuario quien se encargará de arrancar la aplicación y comprobar su funcionamiento a través del estado de cada uno de sus sensores así como actuar ante las diferentes alarmas.*

ACT-002	**Central**
Descripción	Este actor representa *a la Central donde serán recogidas todas las notificaciones de posibles alarmas que se hayan producido.*

ACT-003	**Paciente**

Descripción	Este actor representa *a la persona que se quiere monitorizar, es decir la persona física que llevará con él el wiimote.*

ACT-004	**Reloj**
Descripción	Este actor representa *a la parte correspondiente del sistema que se encargará de marcar la temporalidad de todos los casos de uso automáticos (sin intervención directa del paciente o el personal médico), y por tanto el que desencadenará éstos.*

ACT-005	**Sensor Caídas**
Descripción	Este actor representa *cada uno de los wiimotes que dispondremos en la aplicación y de los cuales obtendremos la información para su tratamiento posterior*.

6.2. Casos de uso

Presentamos a continuación los casos de uso en función del actor que los desencadena y una pequeña explicación de cada uno de ellos.

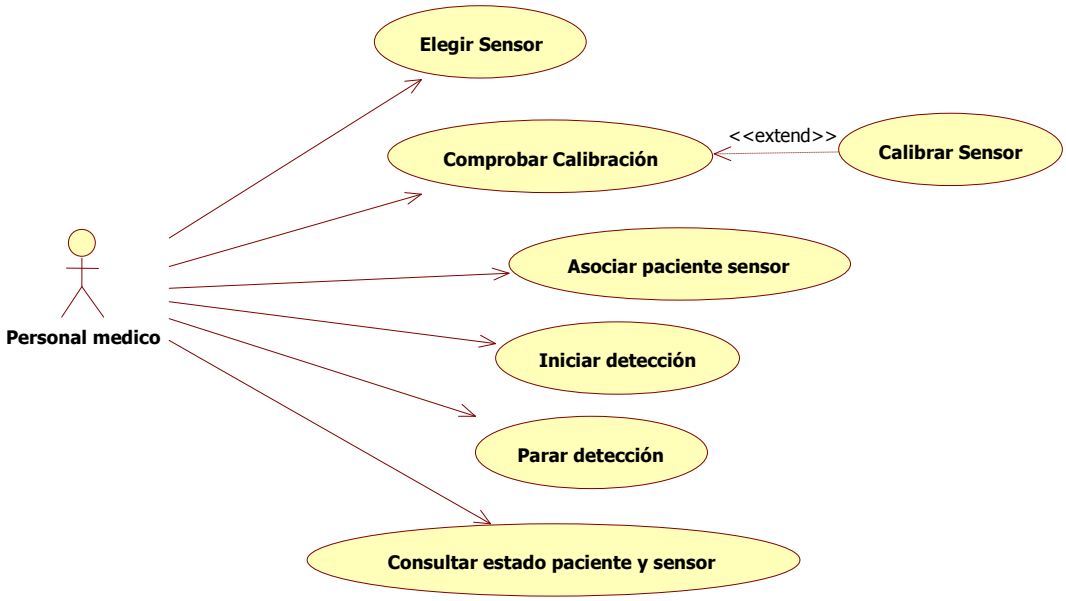


Figura 19. Casos de uso del actor Personal médico.

- UC-001: Elegir sensor.

Este caso de uso representa el hecho de elegir con cuál de los sensores disponibles se quiere trabajar. La aplicación permite disponer de varios wiimotes pero es el usuario Personal Medico quien elegirá uno de ellos entre los disponibles para realizar las opciones oportunas.

- UC-002 Comprobar Calibración.

Este caso de uso se utiliza para que el personal sanitario pueda comprobar la correcta calibración del Wiimote, si este no está bien calibrado la detección no será fiable.

- UC-003 Calibrar Sensor.

Este caso de uso se ocurre cuando se ha comprobado la calibración de un wiimote y no ha sido satisfactoria entonces la aplicación calibra el wiimote.

- UC-004 Asociar paciente con sensor.

Este caso de uso se utiliza para que cada Wiimote quede representado por un paciente y no haya equivocaciones a la hora de conocer que sensor esta monitorizando a que paciente.

Aplicación para la detección de caídas.

- UC-005 Iniciar detección

Este caso de uso sirve para que el Personal médico una vez que haya colocado el wiimote al paciente, le haya asociado con algún paciente inicie la detección de caídas. Cuando se inicia la aplicación la detección de caídas se encuentra desactivada ya que para evitar falsas alarmas en sensor se debe colocar en el cuerpo con la detección parada.

- UC-006 Parar detección

Este caso de uso es el contrario al anterior se encarga de desactivar la detección de caídas.

- UC-007 Consultar estado Paciente

Este caso de uso representa el interés por parte del personal médico de conocer el estado del paciente.

- UC-008 Consultar estado Sensores.

Este caso de uso representa el interés en conocer el estado del funcionamiento de los diferentes sensores así como el estado de la batería de estos.

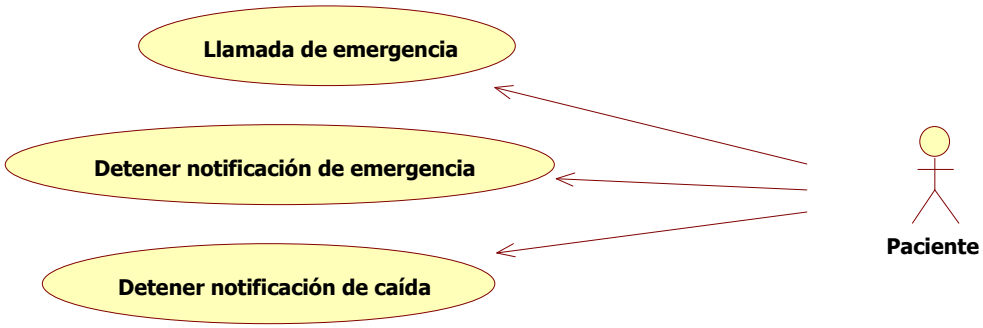


Figura 20. Casos de uso del actor Paciente monitorizado.

- UC-009 Llamada de emergencia

Este caso de uso representa la opción que tiene un paciente monitorizado de poder pedir ayuda a los servicios de asistencia.

- UC-010 Detener notificación de emergencia

Este caso de uso se podrá utilizar por parte del paciente cuando se active una falsa alarma porque este ha pulsado el botón de llamada de emergencia por error.

- UC-011 Detener notificación de caída.

Este caso de uso se podrá utilizar por parte del paciente cuando se active una falsa alarma por haberse detectado una caída errónea.

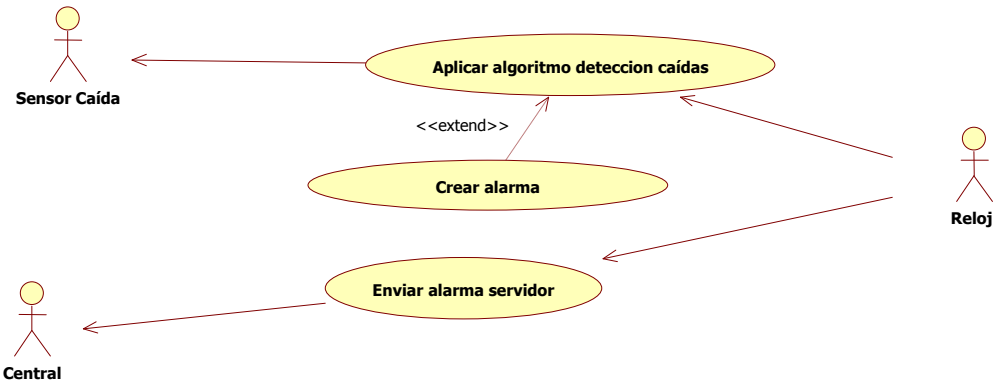


Figura 21. Casos de uso del actor Reloj.

- UC-012 Activar alarma

Este caso de uso representa la creación de una alarma por parte del sistema cuando este averigüe si se ha producido una posible caída.

- UC- 013 Aplicar Algoritmo Detección de caídas.

Este caso de uso representa el tratamiento de los datos de los sensores que realiza el sistema para buscar posibles caídas.

- UC- 014 Enviar alarma servidor

Este caso de uso representa el envío de una notificación por parte del sistema al servidor central. Esta notificación se producirá cuando el sistema haya detectado una caída, haya creado una alarma y el paciente no la haya desactivado.

6.3. Descripción detallada de los casos de uso.

UC-001	**Elegir Sensor**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El actor Personal médico quiere realizar alguna operación con un sensor para ello elige uno entre los disponibles,* o durante la realización de los siguientes casos de uso: [UC-003] Consultar estado del Wiimote., [UC-005] Aplicar algoritmo detección de caídas	
Precondición	Existe al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema *muestra cuando se inicia la aplicación un formulario con todos los sensores de caídas disponibles.*
	2	El actor Personal Sanitario (ACT-001) *elige en la opción de menú correspondiente al sensor con el que quiera trabajar.*
Postcondición	wiimote elegido.	

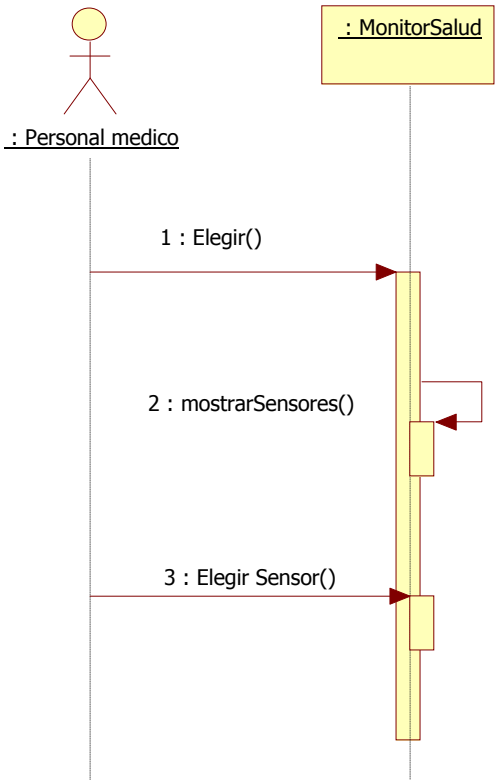


Figura 22. Diagrama de secuencia UC-001

UC-002	**Comprobar calibración**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando, *El actor personal médico desea comprobar la calibración de algún mando Wiimote.*	
Precondición	Existe al menos un Wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)

	2	El actor Personal Sanitario (ACT-001) *Selecciona la opción de menú comprobar calibración para el sensor elegido.*
	3	El sistema *Realiza la una comprobación de los datos de las aceleraciones leídas si son correctas envía un mensaje por pantalla de que el wiimote está bien calibrado.*
Postcondición	Calibración wiimote comprobada.	

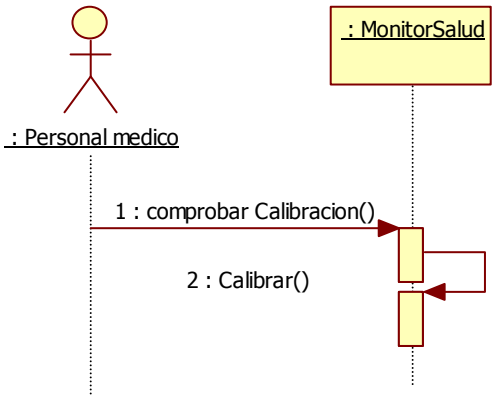


Figura 23. Diagrama de secuencia UC-002

UC-003	**Calibrar Sensor**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso durante la realización del siguiente caso de uso: [UC-002] Comprobar Calibración	
Precondición	Existe algún wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Personal Sanitario (ACT-001) *desea comprobar la calibración de un mando wiimote.*

	2	El sistema *Indica al usuario en un formulario como debe colocar el sensor primero con el teclado hacia arriba.*
	3	El actor Personal Sanitario (ACT-001) *coloca el mando como indica la foto y pulsa siguiente.*
	4	El sistema *muestra una foto del cómo debe colocar el usuario el mando.*
	5	El actor Personal Sanitario (ACT-001) *coloca el mando en posición indicada y pulsa siguiente.*
	6	El sistema *muestra una foto de como se debe colocar el mando.*
	7	El actor Personal Sanitario (ACT-001) *coloca el mando en posición indicada y pulsa siguiente.*
	8	El sistema *calibra el sensor.*
Postcondición	Sensor calibrado.	

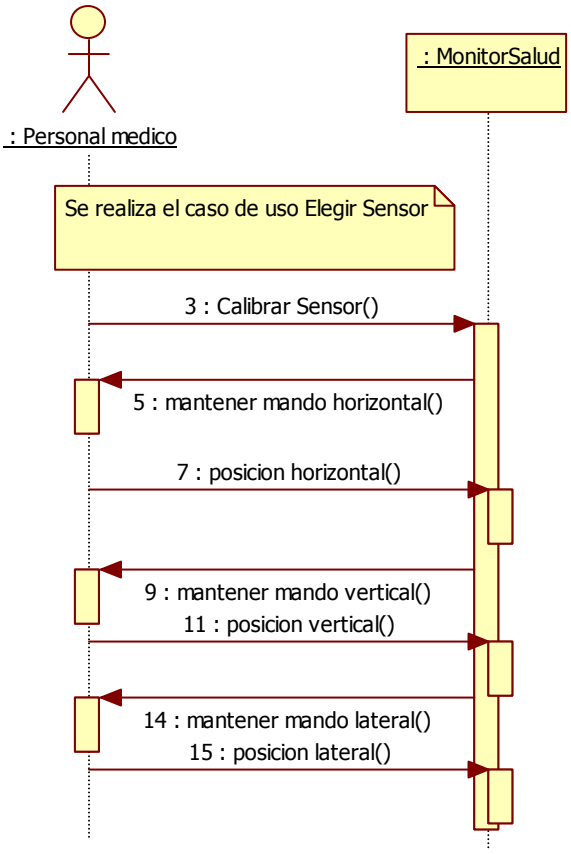


Figura 24. Diagrama de secuencia UC-003

UC-004	**Asociar sensor a paciente**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *que se quiere conectar y configurar un nuevo sensor al sistema.*	
Precondición	Requiere existan al menos un wiimote conectado.	
Secuencia	**Paso**	**Acción**

normal	1	El actor Personal Sanitario (ACT-001) *desea asociar un sensor a un paciente para ello elige el wiimote que quiere asociar entre los disponibles.*
	2	El sistema *para el wiimote seleccionado indicará que se escriba el nombre del paciente.*
	3	El actor Personal Sanitario (ACT-001) *introduce el nombre del paciente.*
	4	El sistema *crea el nuevo paciente y lo deja asociado con el wiimote.*
Postcondición	Nueva asociación paciente wiimote.	

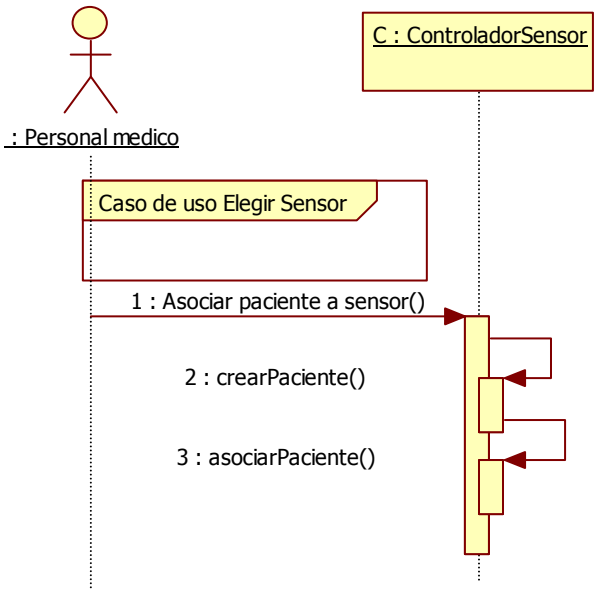


Figura 25. Diagrama de secuencia UC-004

UC-005	**Iniciar detección**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor personal médico indica que quiere activar la detección de caídas.*	
Precondición	Debe existir al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El sistema *Inicia la detección de caídas*
Postcondición	Aplicando algoritmo de detección de caídas.	

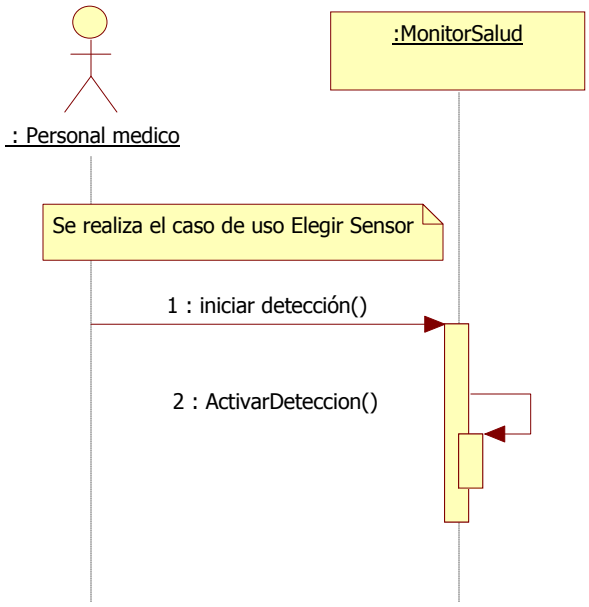


Figura 26. Diagrama de secuencia UC-005

UC-006	**Parar detección**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando: *El usuario personal médico desea parar la detección de caídas.*	
Precondición	Debe existir al menos un wiimote con el detector de caídas iniciado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El actor Personal Sanitario (ACT-001) *pulsa el botón desactivar detección para el wiimote que ha elegido anteriormente en el paso 1.*
	3	El sistema *desactiva la detección de caídas en el wiimote seleccionado.*
Postcondición	Se ha parado la detección de caídas para un wiimote.	

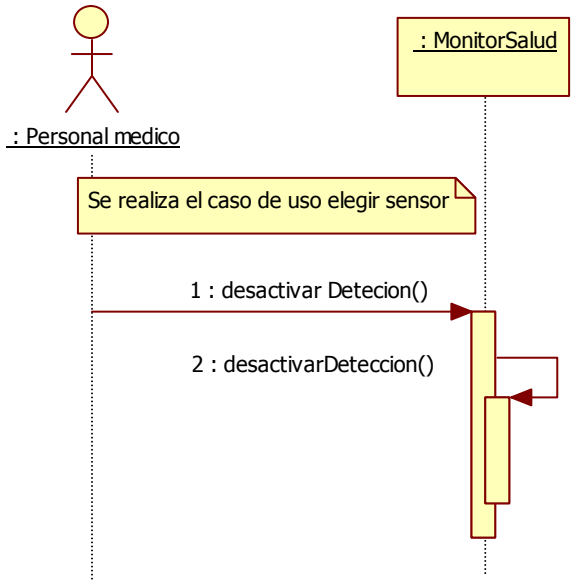


Figura 27. Diagrama de secuencia UC-006

UC-007	**Consultar estado del sensor caída.**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiere saber si el sensor está funcionando de forma correcta, es decir no se ha perdido la conexión o tiene suficiente batería.*	
Precondición	Debe existir al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El actor Personal Sanitario (ACT-0001) *Indica que quiere consultar el estado del wiimote seleccionado para ello pincha en el botón Ver Características.*
	3	El sistema *muestra para el sensor seleccionado los siguientes datos: estado detección, estado batería y aceleraciones actuales leídas del acelerómetro del sensor.*

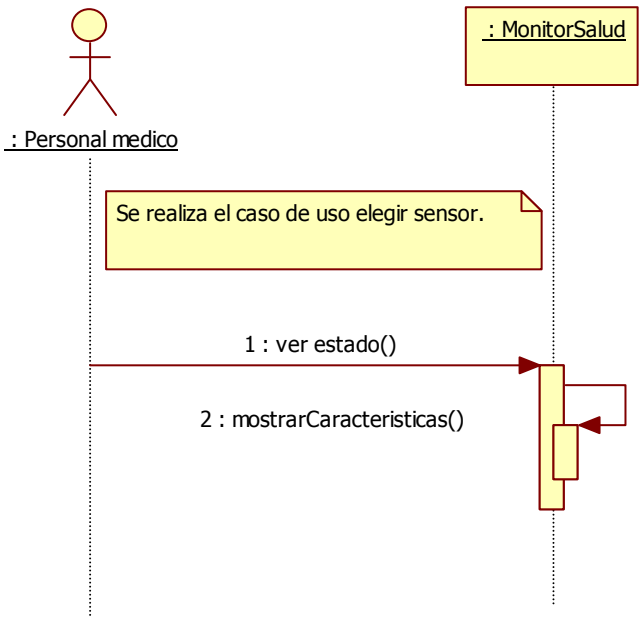


Figura 28. Diagrama de secuencia UC-007

UC-008	**Consultar estado paciente**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor personal médico desee conocer el estado de un paciente concreto.*	
Precondición	Debe existir al menos un wiimote conectado	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El sistema *muestra para el sensor seleccionado los siguientes datos: estado detección, paciente asociado, estado de alarma, orientación actual del sensor*

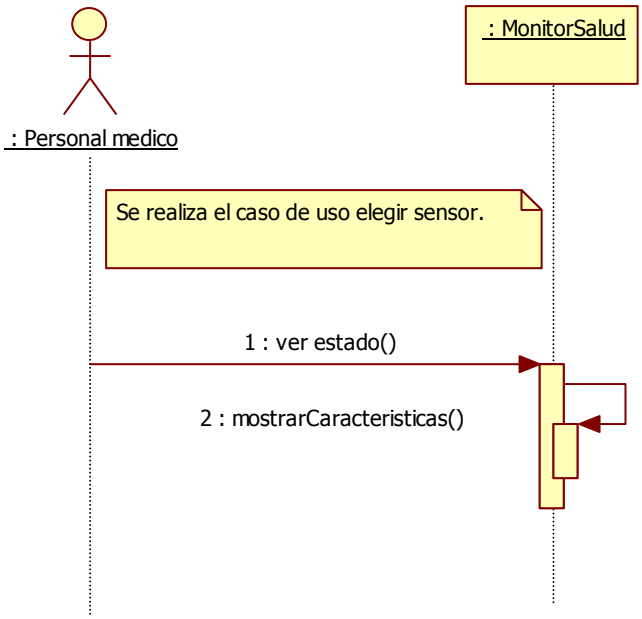


Figura 29. Diagrama de secuencia UC-008

UC-009	**Llamada de emergencia**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor paciente desea activar una llamada de emergencia*	
Precondición	Existe al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Paciente (ACT-009) *pulsa el botón cruceta en el mando wiimote.*
	2	Se realiza el caso de uso Activar alarma (UC-012)
Postcondición	Se crea una nueva alarma.	

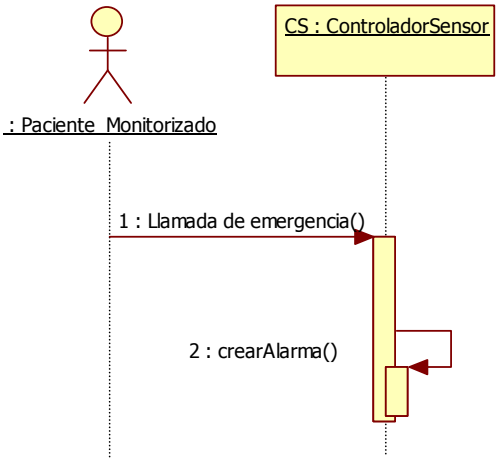


Figura 30. Diagrama de secuencia UC-009

UC-0010	**Detener notificación de emergencia**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando El usuario paciente haya pulsado por confusión el botón de emergencia del wiimote.	
Precondición	Existe al menos una alarma en estado activado que no ha sido enviada aun al servidor Central.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Paciente (ACT-009) *desea desactivar una falsa alarma para ello presiona en el botón A del mando Wiimote.*
	2	El sistema *recoge la petición elimina la alarma*
Postcondición	Alarma eliminada.	

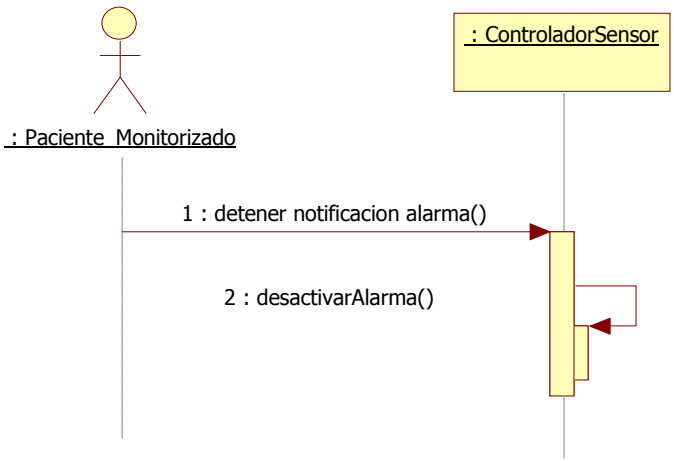


Figura 31. Diagrama de secuencia UC-010

UC-0011	**Detener notificación de caída**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando la aplicación haya creado una alarma por una falsa caída.	
Precondición	Existe al menos una alarma en estado activado que no ha sido enviada aun al servidor Central.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Paciente (ACT-0009) *desea desactivar una falsa alarma para ello presiona en el botón A del mando Wiimote.*
	2	El sistema *recoge la petición elimina la alarma*
Postcondición	Alarma eliminada.	

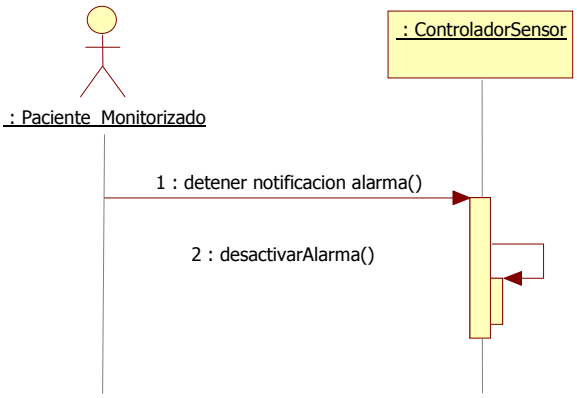


Figura 32. Diagrama de secuencia UC-0011

UC-012	**Activar alarma**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El actor reloj ha detectado una caída o bien el actor paciente ha pulsado el botón de llamada de emergencia en el mando wiimote.*	
Precondición	Se ha detectado una caída o se activado la llamada de emergencia por parte del Paciente.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Activar alarma (UC-012)
	2	El actor Reloj (ACT-004) *Espera para ver si realiza el caso de uso* Detener notificación de caída (UC-011).
Postcondición	Alarma generada	

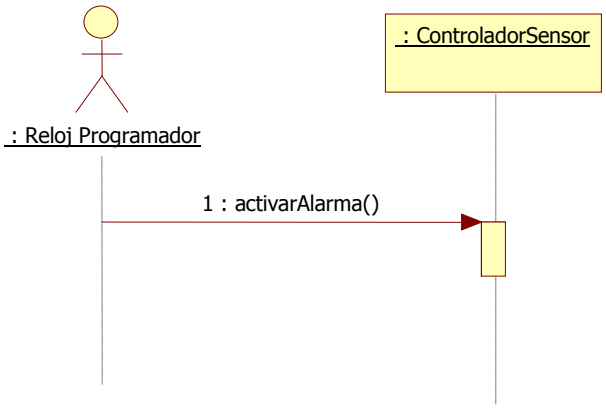


Figura 33 Diagrama de secuencia UC-012.

UC-013	**Aplicar algoritmo detección de caídas.**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se ha activado el caso de uso* UC-005 Iniciar detección.	
Precondición	Se ha iniciado el caso de uso UC-005 Iniciar detección.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Reloj (ACT-006) *Inicia el algoritmo de detección de caídas.*
Postcondición	Realizando algoritmo de detección de caídas.	

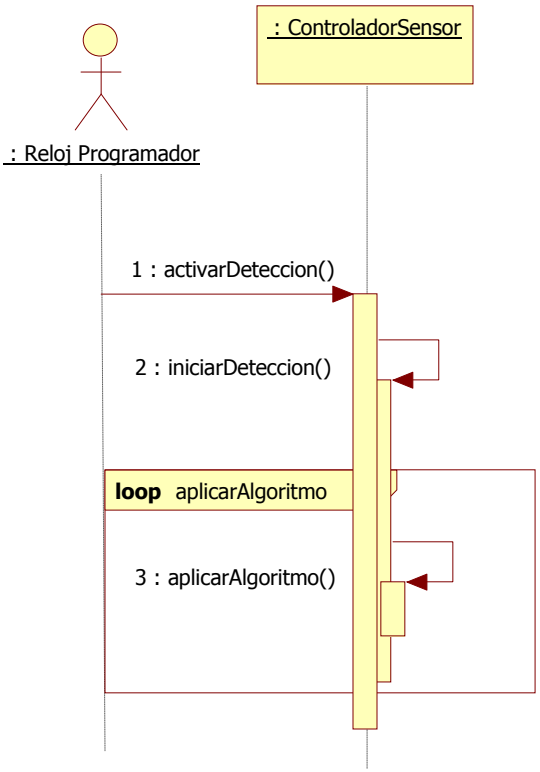


Figura 34. Diagrama de secuencia UC-0013

UC-014	**Enviar alarma servidor**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando l*a aplicación ha detectado automáticamente una posible caída o el actor paciente ha iniciado una llamada de emergencia.*	
Precondición	Existe al menos un wiimote conectado.	
Secuencia	**Paso**	**Acción**

normal	1	El actor Reloj (ACT-004) *ha detectado una alarma no desactivada y quiere enviar la notificación.*
	2	El sistema *Envía la notificación con información del paciente y el tipo de alarma al servidor central.*
Postcondición	Se ha enviado una alarma al servidor.	

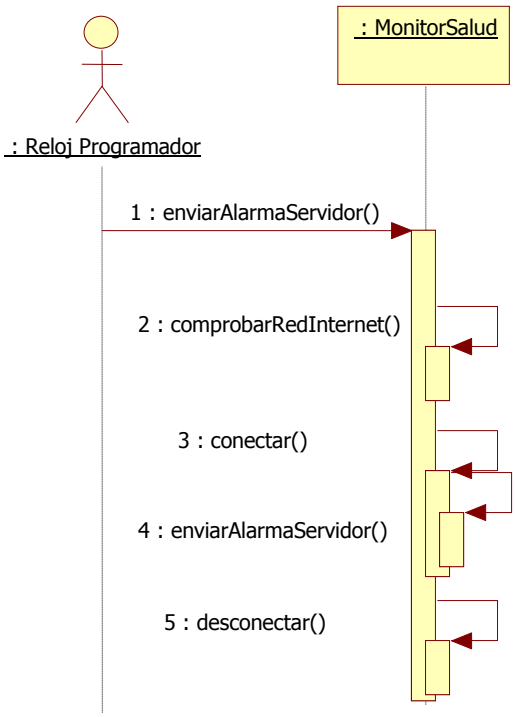


Figura 35. Diagrama de secuencia UC-0012

7. Modelo de Análisis

7.1. Introducción

Esta sección describe el diagrama inicial de clases obtenido a partir de los requisitos del sistema y de los casos de uso. Se ofrece una breve descripción de las clases detectadas en el análisis.

Este documento se limita a dar una descripción de la funcionalidad de sistema, de tal forma que se satisfagan todos y cada uno de los requisitos software desarrollados en los apartados anteriores. Se describen las clases que componen la aplicación y las relaciones entre ellas, así como una breve descripción de la responsabilidad de cada una de ellas.

7.2. Modelo inicial de clases.

En función de los requisitos de sistema y casos de uso presentamos el diagrama de clases inicial.

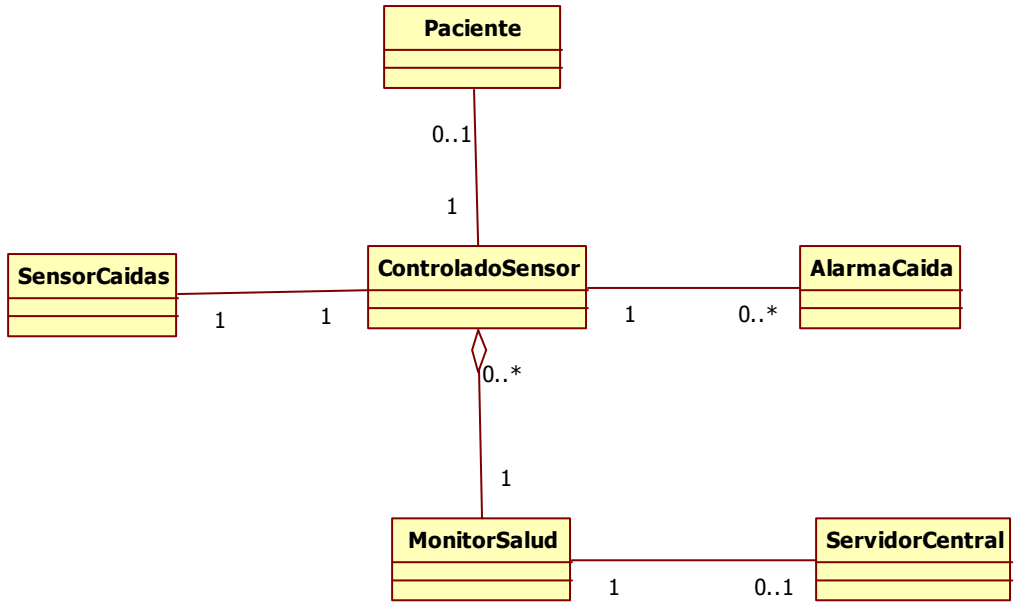


Figura 36. Diagrama inicial de clases.

Aplicación para la detección de caídas.

Diseño

1. Introducción.

El diseño es el proceso por el cual se traducen las especificaciones de los requisitos en una representación del software que se desea construir.

El diseño representa un puente entre el análisis del problema y la implementación de la solución a ese problema.

El presente documento se divide en los siguientes puntos

- Diagramas de secuencia en diseño.
- Diagrama final de clases.
- Arquitectura del sistema.
- Diseño de la capa Interfaz
- Diseño de la capa Lógica de Negocio
- Diagrama Detallado de Clases
- Modelo de Datos

2. Diagramas de secuencia en diseño.

En esta parte, vamos a desarrollar los diagramas de secuencia de los casos de uso, y por cada caso de uso mostrando el orden de las llamadas del sistema en funcionamiento, aquellos casos de usos que necesiten alguna secuencia alternativa escribiremos los pasos necesarios.

2.1. Diagrama de secuencia UC-001 Elegir sensor.

UC-001	**Elegir Sensor**	
Dependencias	Ninguno	
Precondición	Existe al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema *muestra en todo momento un formulario con los diferentes sensores que se encuentran conectados, para cada uno de ellos existe un menú con las siguientes opciones:*

		o Activar Detección. *o Desactivar Detección* *o Calibrar Sensor.* *o Asociar con Paciente* *o Ver Características.*
	2	El actor Personal Sanitario (ACT-001) *elige en la opción de menú correspondiente al sensor con el que quiera trabajar.*
	3	El sistema *procesa la petición del usuario.*
Postcondición	Wiimote seleccionado.	

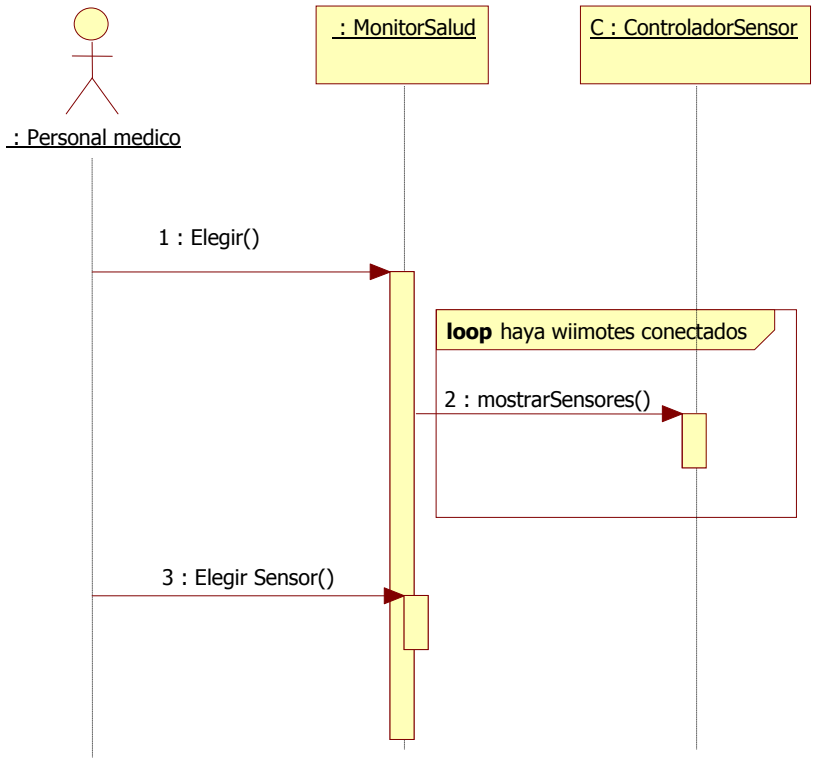


Figura 37 Diagrama de secuencia UC-001 Elegir sensor.

2.2. Diagrama de secuencia UC-002 Comprobar Calibración

UC-002	**Comprobar calibración**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El actor personal médico desea comprobar la calibración de algún mando Wiimote.*	
Precondición	Existe al menos un Wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El actor Personal Sanitario (ACT-001) *Selecciona la opción de menú comprobar calibración para el sensor elegido.*
	3	El sistema *Realiza la una comprobación de los datos de las aceleraciones leídas si son correctas envía un mensaje por pantalla de que el wiimote está bien calibrado.*
Postcondición	Calibración wiimote comprobada.	
Excepciones	**Paso**	**Acción**
	3	Si *los datos de calibración no son correctos,* se realiza el caso de uso Calibrar Sensor (UC-003)

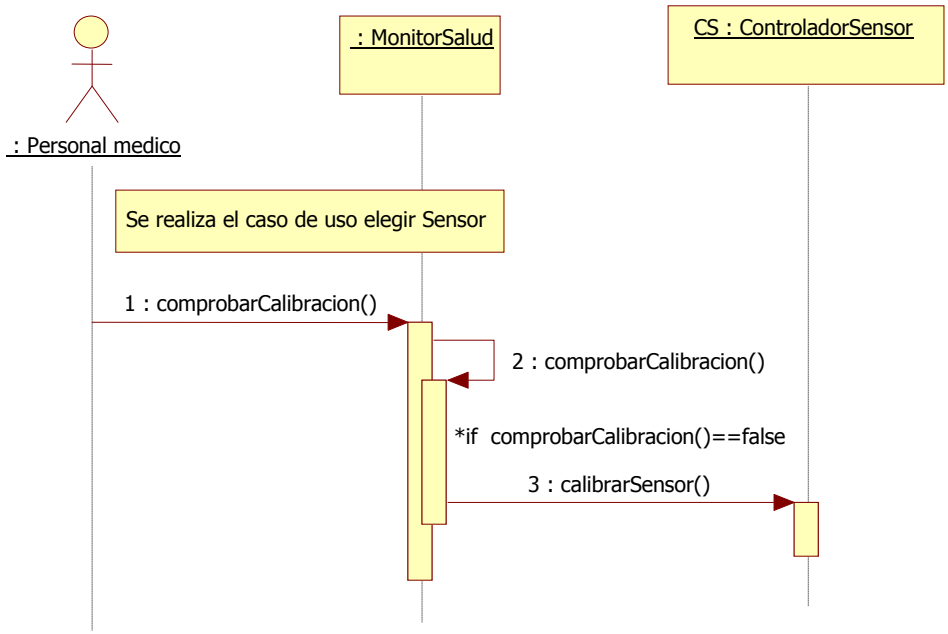


Figura 38 Diagrama de secuencia UC-002 Comprobar Calibración.

2.3. Diagrama de secuencia UC-003 Calibrar Sensor.

UC-003	**Calibrar Sensor**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor Personal Medico desea calibrar un wiimote*	
Precondición	Existe algún sensor conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Personal Sanitario (ACT-001) *desea calibrar un wiimote.*

	2	El sistema *muestra al usuario en con una foto como debe colocar el wiimote.*
	3	El actor Personal Sanitario (ACT-001) *coloca el mando como indica la foto y pulsa el botón siguiente en el formulario.*
	4	El sistema *recoge los datos de las aceleraciones del wiimote en la primera posición, e indica al usuario como es la siguiente posición en la que debe colocar el wiimote.*
	5	El actor Personal Sanitario (ACT-001) *coloca el mando en posición indicada y pulsa el botón siguiente.*
	6	El sistema *recoge los datos de las aceleraciones del wiimote en la segunda posición, e indica al usuario como es la siguiente posición en la que debe colocar el wiimote.*
	7	El actor Personal Sanitario (ACT-001) *coloca el mando en posición indicada y pulsa el botón siguiente.*
	8	El sistema *recoge los datos de las aceleraciones del wiimote en la tercera posición, calibra el wiimote y guarda los datos asociados a la calibración.*
Postcondición	Wiimote calibrado	

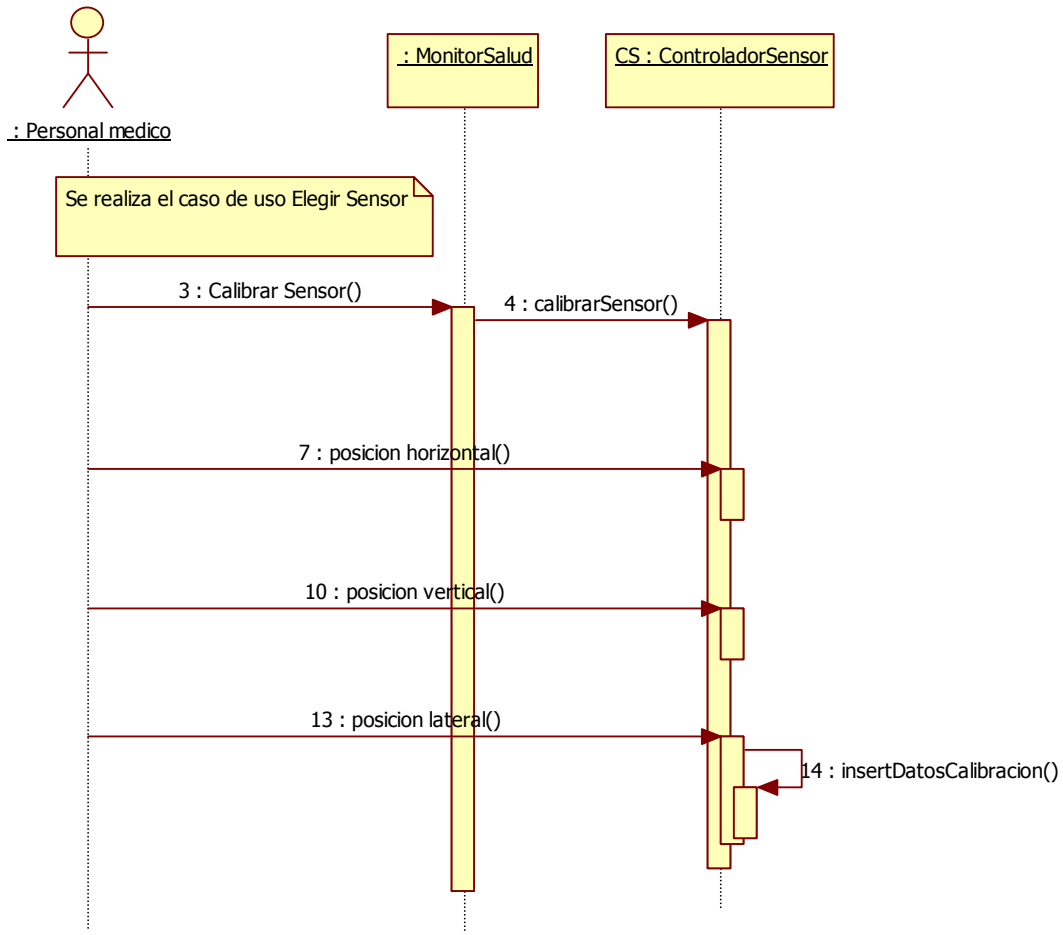


Figura 39 Diagrama de secuencia UC-003 Calibrar Sensor.

2.4. Diagrama de secuencia UC-004 Asociar paciente a sensor

UC-004	**Asociar Sensor a Paciente**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *que se quiere conectar y configurar un nuevo sensor al sistema.*	
Precondición	Requiere existan al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Personal Sanitario (ACT-001) *desea asociar un sensor a un paciente para ello elige la opción de menú asociar paciente para uno de los sensores disponibles.*
	2	El sistema *solicita al usuario el nombre del paciente a asociar.*
	3	El actor Personal Sanitario (ACT-001) *introduce el nombre del paciente y pulsa el botón Asociar Paciente.*
	4	El sistema *recoge la información proporcionada por el usuario crea un nuevo paciente y se le asocia al Wiimote correspondiente.*
Postcondición	Nueva asociación paciente sensor.	

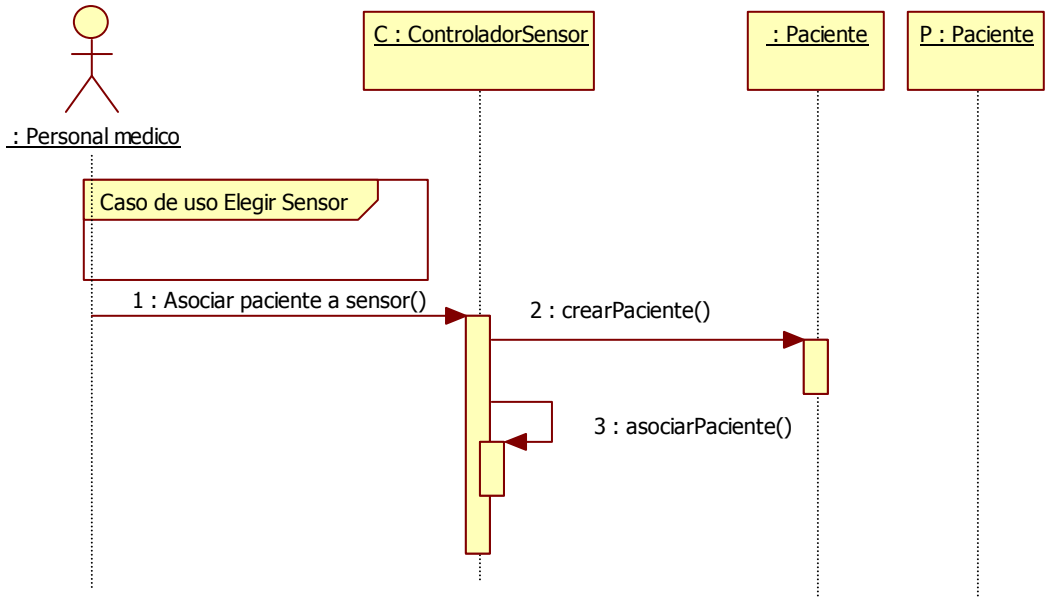


Figura 40 Diagrama de secuencia UC-004 Asociar paciente a sensor

.

2.5. Diagrama de secuencia UC-005 Iniciar detección

UC-005	**Iniciar Detección**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El actor Personal médico indica que quiere activar el algoritmo de detección de caídas.*	
Precondición	Existe al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Personal Sanitario (ACT-001) *selecciona para uno de los wiimotes conectados la opción de menú, iniciar detección.*

	2	El sistema *recoge la petición del usuario y crea un objeto DetectorCaídas para el sensor elegido.*
	3	Se realiza el caso de uso Aplicar algoritmo detección de caídas (UC-013)
Postcondición	Para el sensor elegido se está aplicando la detección de caídas.	

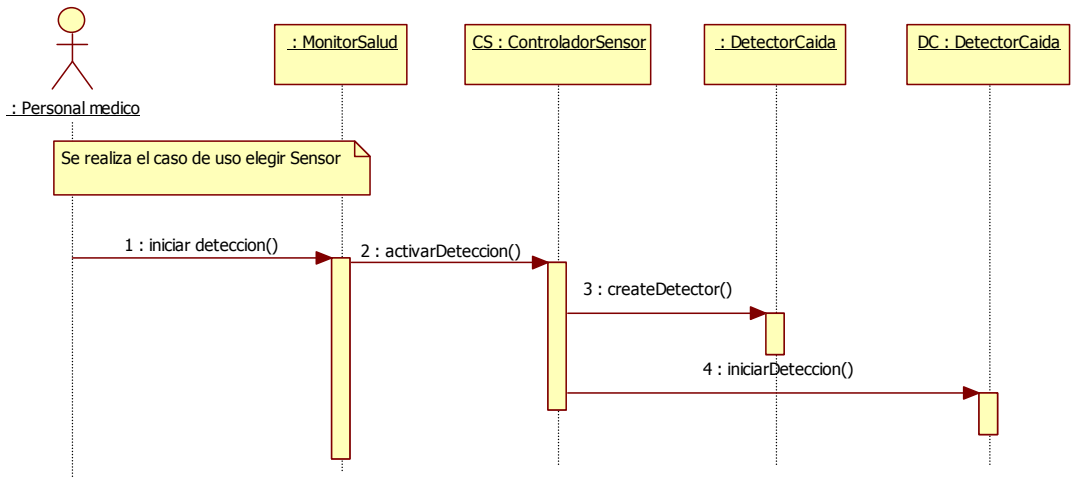


Figura 41. Diagrama de secuencia caso de uso UC-005 Iniciar detección

2.6. Diagrama de secuencia UC-006 Parar detección

UC-006	**Parar Detección**
Dependencias	Ninguno
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El usuario personal médico desea parar la detección de caídas.*

Precondición	Debe existir al menos un wiimote con el detector de caídas iniciado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El actor Personal Sanitario (ACT-001) *Elige la opción de menú desactivar detección para el sensor.*
	3	El sistema *recoge la petición y para el sensor elegido elimina el objeto DetectorCaídas*
Postcondición	Se ha parado la detección de caídas para algún sensor	

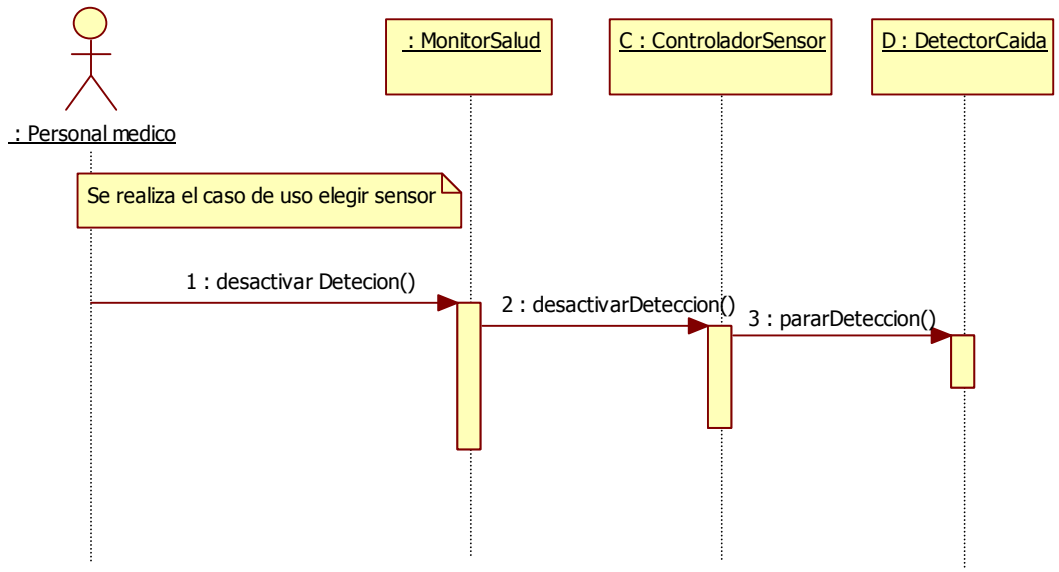


Figura 42. Diagrama de secuencia caso de uso UC-006 Parar detección.

2.7. Diagrama de secuencia UC-007 Consultar estado sensor

UC-007	**Consultar estado del sensor.**

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiere saber si el sensor está funcionando de forma correcta, es decir no se ha perdido la conexión o tiene suficiente batería.*	
Precondición	Debe existir al menos un wiimote conectado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El actor Personal Sanitario (ACT-0001) *Indica que quiere consultar el estado del wiimote seleccionado para ello pincha en el botón Ver Características.*
	3	El sistema *muestra para el sensor seleccionado los siguientes datos: estado detección, estado batería y aceleraciones actuales leídas del acelerómetro del wiimote.*

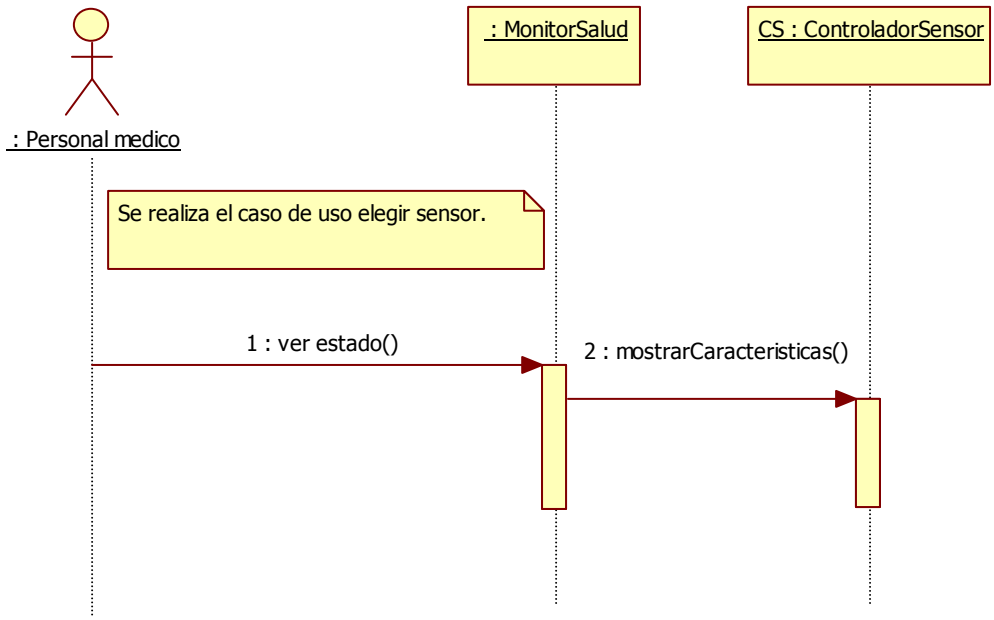


Figura 43 Diagramas de secuencia UC-007 consultar estado sensor

.

2.8. Diagrama de secuencia UC-008 Consultar estado paciente.

UC-008	**Consultar estado paciente**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor personal médico desee conocer el estado de un paciente concreto.*	
Precondición	Debe existir al menos un wiimote conectado	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Elegir Sensor (UC-001)
	2	El sistema *muestra para el sensor seleccionado los siguientes datos: estado detección, paciente asociado, estado de alarma, orientación actual del paciente.*

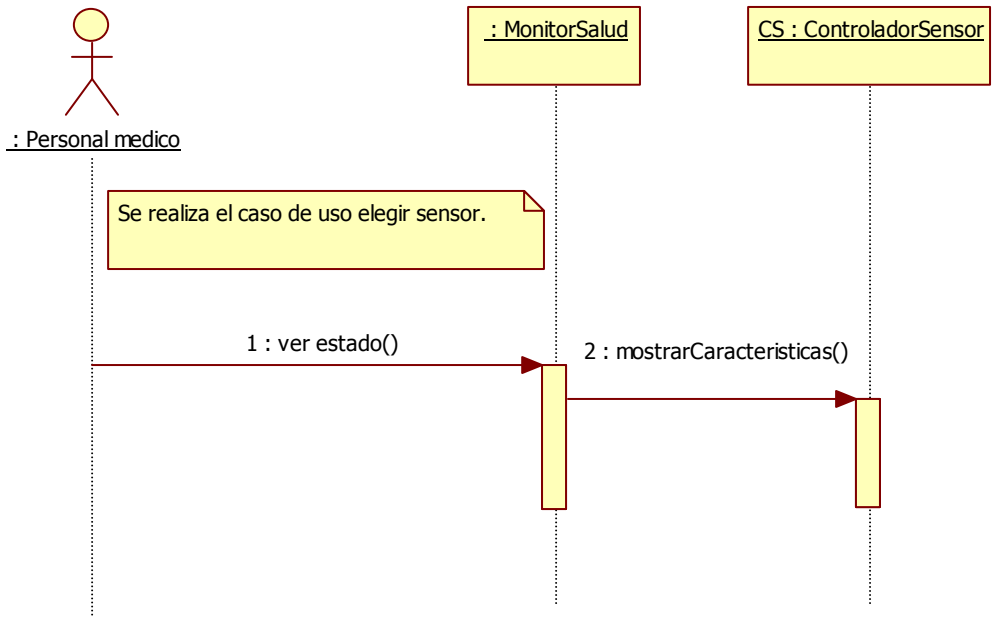


Figura 44. Diagrama de secuencia UC-008 consultar estado paciente

2.9. Diagrama de secuencia UC-009 Llamada de emergencia

UC-009	**Llamada de emergencia diseño.**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor paciente desea activar una llamada de emergencia*	
Precondición	Existe al menos un wiimote conectado funcionando.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Paciente (ACT-003) *pulsa el botón con forma de cruz en el mando Wiimote.*
	2	El sistema *crea un nuevo objeto de tipo AlarmaCaída*
	3	Se realiza el caso de uso Activar alarma (UC-012)
Postcondición	Se crea una nueva alarma.	

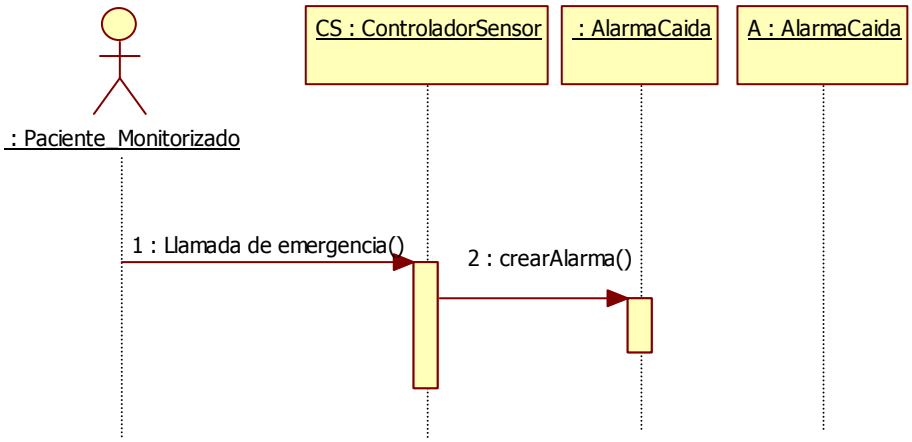


Figura 45. Diagrama de secuencia caso de uso UC-009 Llamada de emergencia

2.10. Diagrama de secuencia UC-010 Detener notificación de emergencia.

UC-010	**Detener notificación de emergencia.**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se haya producido una falsa alarma de caída y se desee eliminar la notificación.	
Precondición	Existe al menos una alarma en estado activado que no ha sido enviada aún al servidor Central.	
Secuencia	**Paso**	**Acción**

normal	1	El actor Paciente (ACT-003) *desea desactivar una falsa alarma para ello presiona en el botón A del mando Wiimote.*
	2	El sistema *recoge la petición, y elimina el objeto AlarmaCaída para el wiimote seleccionado.*
Postcondición	Alarma eliminada.	
Excepciones	**Paso**	**Acción**
	2	Si *no hay ninguna alarma activa* este caso de uso *queda sin efecto*

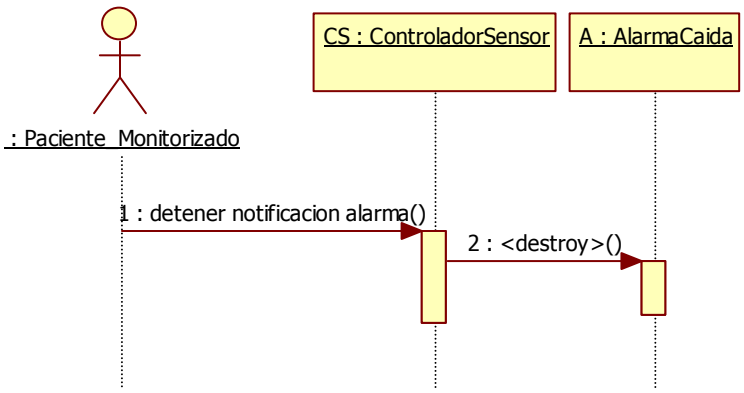


Figura 46. UC-010 Desactivar notificación de emergencia.

2.11. Diagrama de secuencia UC-012 Activar Alarma.

UC-012	**Activar alarma**
Dependencias	Ninguno
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *El actor reloj ha detectado una caída o bien el*

	actor paciente ha pulsado el botón de llamada de emergencia en el mando wiimote.	
Precondición	Se ha detectado una caída o se activado la llamada de emergencia por parte del Paciente.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Activar alarma (UC-012)
	2	El actor Reloj (ACT-004) *Espera para ver si realiza el caso de uso* Detener Notificación de caída (UC-011) o Detener Notificación de emergencia (UC-010).
Postcondición	Alarma generada	
Excepciones	**Paso**	**Acción**
	2	Si *tras 10 segundos no se realiza el caso de uso* UC-010 Detener Notificación emergencia *o* Detener Notificación de caída (UC-011) *se realiza el caso de uso* UC-014 Enviar alarma servidor

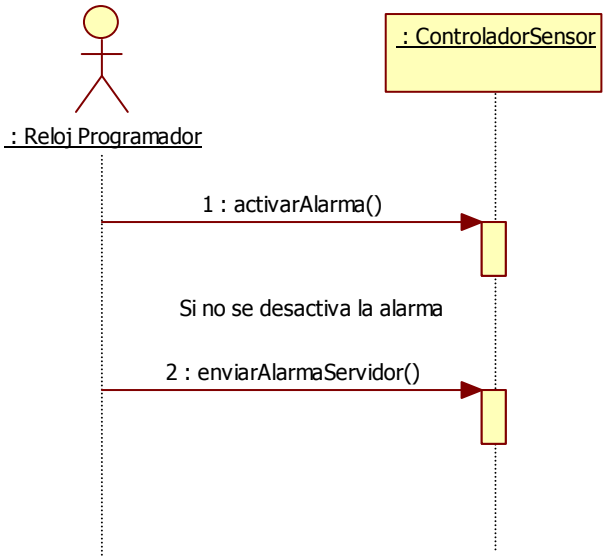


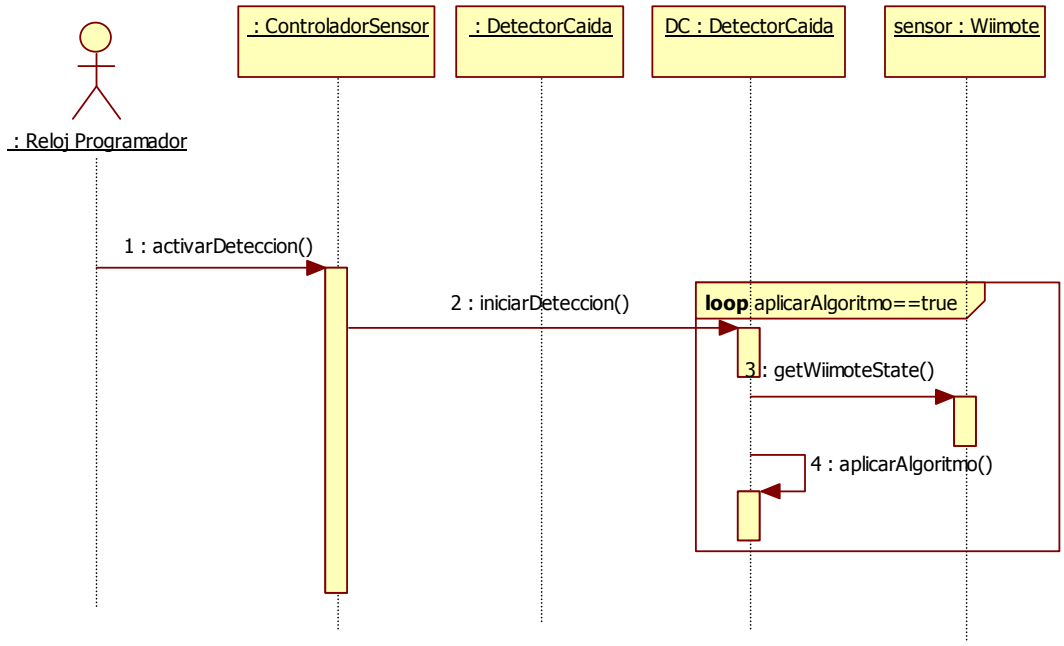
Figura 47 Diagrama de secuencia UC-011 Activar Alarma.

2.12. Diagrama de secuencia UC-012 Aplicar algoritmo detección de caídas

UC-013	**Aplicar algoritmo detección de caídas**
Dependencias	Ninguno
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se ha activado el caso de uso UC-005 Iniciar detección.*
Precondición	Se ha iniciado el caso de uso UC-005 Iniciar detección.

Secuencia normal	**Paso**	**Acción**
	1	El actor Reloj (ACT-004) *lee los datos de las aceleraciones del mando Wiimote.*
	2	El actor Reloj (ACT-004) *procesa la información de las aceleraciones y comprueba que estén dentro de unos umbrales adecuados, si todo es correcto vuelve al paso 1*
Postcondición	Realizando algoritmo de detección de caídas.	
Excepciones	**Paso**	**Acción**
	2	Si *los datos están fuera de los umbrales* , se realiza el caso de uso Activar alarma (UC-006)
	2	Si *se realiza el caso de* Parar detección (UC-006), este caso de uso finaliza.

Secuencia normal.



Aplicación para la detección de caídas.

Figura 48.

2.13. Diagrama de secuencia UC-013 Enviar alarma servidor

UC-013	**Enviar alarma servidor**	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *La aplicación ha detectado automáticamente una posible caída o el actor paciente ha iniciado una llamada de emergencia.*	
Precondición	Existe al menos un sensor conectado y asociado a un paciente.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Reloj (ACT-0004) *ha detectado una alarma no desactivada y quiere enviar la notificación.*
	2	El sistema *Envía la notificación con información del paciente y el tipo de alarma al servidor central.*
Postcondición	Se ha enviado una alarma al servidor.	

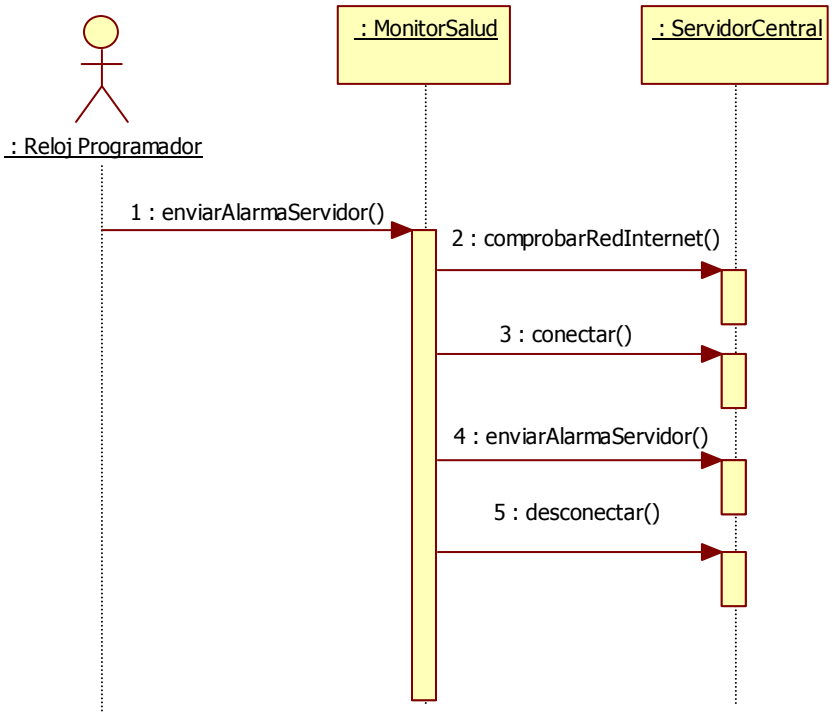


Figura 49. Diagrama de secuencia caso de uso UC-013 Enviar Alarma Servidor secuencia normal.

3. Diagrama final de clases.

Una vez desarrollados los diagramas de secuencia para todos los casos de uso podemos describir de forma detallada todos los atributos y operaciones que van a necesitar nuestras clases.

Con respecto al diagrama inicial de clases [Figura 36] aparece una nueva clase llamada DetectorCaídas cuya funcionalidad explicamos más abajo.

En el diagrama se han incluido las clases que cubren los diferentes conceptos que nos hemos encontrado en los casos de uso, y que van a permitir al sistema desarrollar la funcionalidad requerida en estos.

Se ha centralizado toda la funcionalidad y gestión de la aplicación sobre dos clases fundamentalmente, la clase MonitorSalud y la clase ControladorSensor.

La clase MonitorSalud será la encargada de buscar y conectarse a los diferentes Wiimotes, y de controlar el funcionamiento de la clase ControladorSensor, así mismo será la clase encargada de enviar las notificaciones al ServidorCentral si fuese necesario.

La clase ControladorSensor es la clase encargada de vigilar el estado de los sensores, crear las alarmas y desactivar la notificación de alarma si el usuario lo considera oportuno para cada uno de los sensores de caídas que se dispongan.

La clase DetectorCaídas es la clase encargada de realizar el algoritmo de detección de caídas, en caso de que detecte una posible caída será la encargada de comunicárselo a la clase ControladorSensor para que esta realice las acciones oportunas.

La clase ServidorCentral, será el interfaz de comunicación para la recepción de configuración de pacientes, envío de datos y notificaciones.

La clase Wiimote nos va a proporcionar una interfaz para la lectura de datos del sensor de caídas (Wiimote).

Como se observa en el diagrama detallado de clases se han pintado 3 clases con fondo gris, estas clases son las pertenecientes al paquete WiimoteLib de Brian Peek [8] la cual transforma los datos leídos en el mando Wiimote en unas estructuras de datos fáciles de utilizar.

Mientras que la clase Enumeration Datos de la que no se ha explicado nada representa una estructura de datos que nos va permitir realizar el algoritmo de detección de caídas, más adelante en esta memoria se explicará la utilidad de esta estructura.

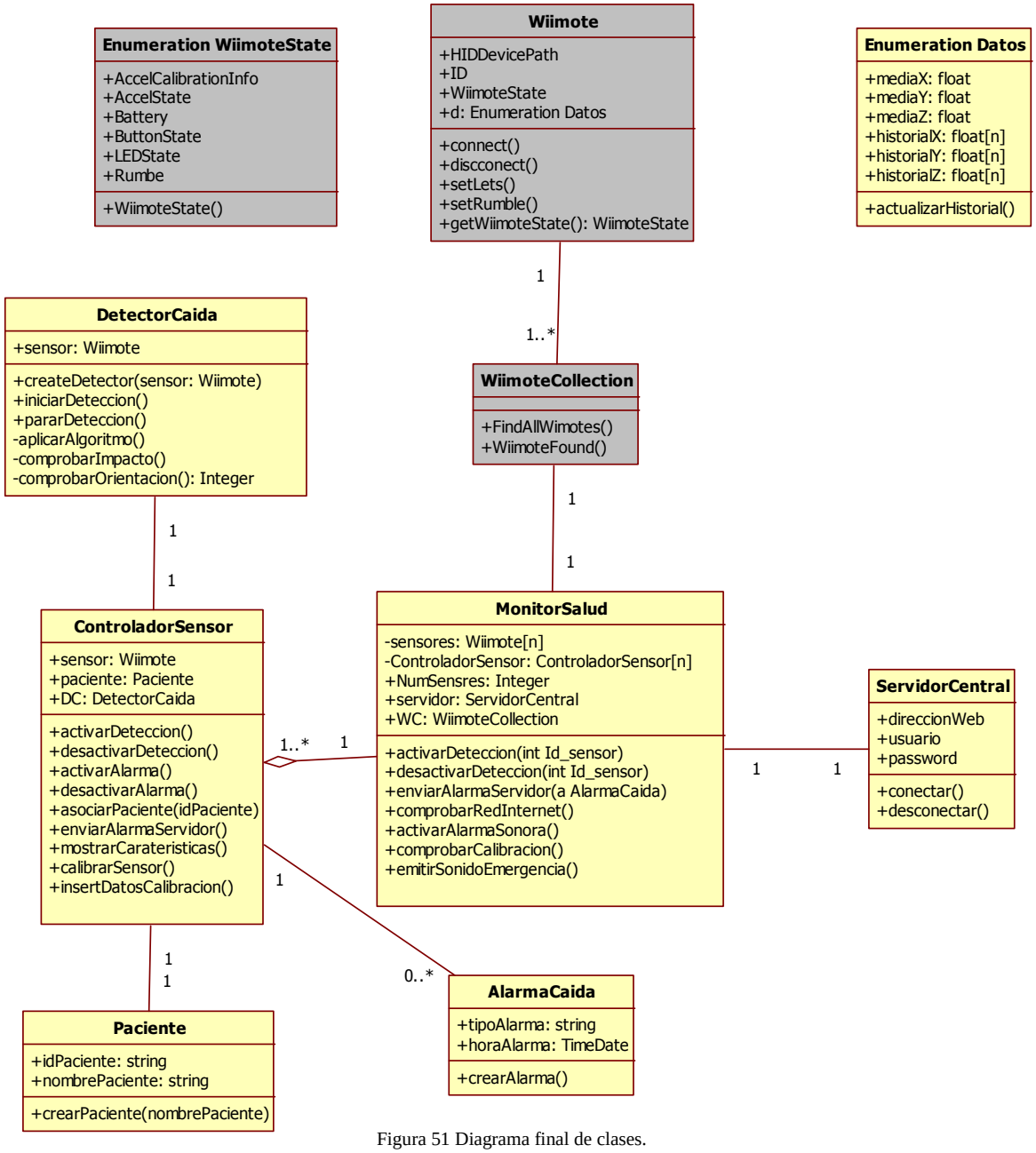


Figura 51 Diagrama final de clases.

3.1. Especificaciones de las clases

En este punto se van a explicar los diferentes atributos y métodos de cada clase.

3.1.1. MonitorSalud

Atributos:

- Sensores: Wiimote[n] representa una lista con todos los mandos wiimotes que tenemos conectados en la aplicación, en este proyecto n=2 ya que solo disponemos de 2 mandos para realizar pruebas, pero la forma en que esta estructurado el código de la aplicación permitirá que en un futuro se puedan modificar y añadir más mandos wiimotes.
- La clase MonitorSalud con la ayuda de la biblioteca WiimoteLib es la encargada de buscar y conectarse a cada Wiimote una vez que se conecta de forma correcta se guardan los objetos Wiimote en la lista llamada sensores.
- ControladorSensor: ControladorSensor[n] representa una lista que contiene los controladores de cada Wiimote en este caso como ya hemos dicho antes n=2 puesto que habrá un ControladorSensor por cada Wiimote que se encuentre en el sistema.
- NumSensores: Integer representa el número de wiimotes que se encuentras conectados en la aplicación.
- WC es un objeto de tipo WiimoteCollection que proporciona la biblioteca WimoteLib que da las operaciones para buscar y conectarse a los Wiimotes.

Métodos:

- activarDeteccion(int idSensor) activa el algoritmo de detección de caídas para el Wiimote con numero idSensor.
- desactivaDeteccion(int idSensor) desactiva el algoritmo de detección de caídas para el Wiimote con numero idSensor.
- enviarAlarmaServidor(a: AlarmaCaída) método encargado de conectarse al servidor central y enviar la notificación de alarmas.

3.1.2. ControladorSensor

Atributos:

- Sensor:Wiimote objeto mediante el cual podemos leer las aceleraciones del mando Wiimote.
- Paciente: Paciente objeto que representa al paciente asociado al objeto Sensor.
- DC:DetectorCaídas objeto con los procedimientos necesarios para averiguar a partir de los datos del objeto Sensor si se ha producido una caída.

Métodos:

- activarDeteccion() activa el algoritmo de detección de caídas.
- desactivaDeteccion() desactiva el algoritmo de detección de caídas.
- activarAlarma() método que crea un objeto de tipo AlarmaCaída, también es el método encargado de esperar a que el paciente pueda desactivar la alarma en caso de ser esta errónea.
- desactivarAlarma() método que elimina alarmas.
- enviarAlarmaServidor() método encargado de avisar a la clase MonitorSalud de que debe enviar una alarma a servidor.
- mostrarCaracteristicas() método que crea un formulario para informar al personal médico sobre posibles estados del Sensor como son:
    - Estado de la batería.
    - Estado detección de caídas.
    - Estado llamada de emergencia.
    - Estado notificación de emergencia.
    - Valores de los datos leídos del acelerómetro.
    - Paciente asociado a sensor.
    - calibrarSensor() método que crea un formulario con los pasos necesarios para calibrar un mando Wiimote.

3.1.3. DetectorCaída

Atributos

Sensor representa el objeto de tipo Wiimote del que vamos a leer los datos de las aceleraciones.

Métodos

- createDetector(sensor:Wiimote) método creador de un objeto de tipo DetectorCaídas para ello se le envía como parámetro un objeto de tipo Wiimote.
- iniciarDeteccion() inicia el método aplicarAlgoritmo().
- detenerDeteccion() detiene el método aplicarAlgoritmo().
- aplicarAlgoritmo() método con los datos leídos del objeto sensor busca posibles caídas.
- comprobarImpacto():Boolean método auxiliar que utiliza el método aplicarAlgoritmo() que devuelve true en caso de que se haya producido un impacto y false en caso que no haya impacto.
- comprobarOrientación():Integer método auxiliar que utiliza el método aplicarAlgoritmo() que devuelve un numero entero según la posición del mando wiimote puede ser vertical, lateral u horizontal.

3.1.4. Paciente

Atributos

- idPaciente:String variable que representa a un paciente.
- nombrePaciente variable que contiene el nombre del paciente monitorizado.

Métodos

- crearPaciente(nombrePaciente) método creador de un objeto tipo Paciente.

3.1.5. AlarmaCaída

Atributos

- tipoAlarma:String atributo con el que diferenciamos el origen de la alarma ya que puede ser por detección de una caída o por una llamada de emergencia.
- horaAlarma:DateTime variable que representa la hora a la que se activo la alarma.

Métodos

- crearAlarma() método creador de un objeto tipo AlarmaCaída.

3.1.6. ServidorCentral

Atributos

- direccionWeb:String dirección IP del servidor
- usuario:String identificador de usuario para poder acceder al servidor.

Métodos.

- Conectar() método para conectarse al servidor.
- Desconectar() método para desconectarse del servidor.

3.1.7. EnumerationDatos

Los atributos y métodos de esta clase serán explicados en el capitulo Algoritmo de Implementación.

Ahora vamos a explicar los atributos y métodos que nos han resultado más útiles en las clases de la biblioteca WimoteLib.

3.1.8. Wiimote

Atributos

- ID identificador del Wiimote, cada Wiimote no viene con un identificador propio de fabrica por lo que se le asigna una MAC de forma automática y aleatoria, esta MAC se guarda en el atributo ID.

- WiimoteState es una estructura de datos que nos permite acceder a las diferentes capacidades del mando Wiimote.
- d: Datos este atributo no esta en la biblioteca WiimoteLib original, se le hemos añadido para facilitar la programación de la aplicación.

Métodos

- Connect() método que conecta el mando Wiimote con el PC
- Disconnect() método que desconecta el mando Wiimote del PC
- setLeds(Boolean,Boolean,Boolean,Boolean) método para controlar los leds del Wiimote por ejemplo una llamada a la setLeds(true,false,false,true) de esta forma provocará que se iluminen los leds 1 y 4 del Wiimote.
- setRumble(Boolean) método para activar y desactivar la vibración del mando Wiimote setRumble(true) el Wiimote vibraría mientras que con false no.

3.1.9. WiimoteCollection

Métodos

- FindAllWiimotes() método que busca todos los wiimotes que se encuentren sincronizados con el PC de la aplicación y después con cada uno de ellos hace una llamada Connect para conectarlos.
- WiimoteFound() método para comprobar si la conexión con los wiimotes se ha realizado de forma correcta o incorrecta.

3.1.10. Enumeration WiimoteState

Estructura en la que son trasformados los datos leídos del Wiimote.

Atributos

- AccelCalibrationInfo contiene los datos necesarios para la calibración de un mando Wiimote.
- AccelState contiene los datos de las aceleraciones en cada momento y en cada uno de los tres ejes cartesianos del mando Wiimote.
- BatteryState información del estado de las pilas del Wiimote.
- ButtonState contiene la información correspondiente a los diferentes botones del mando Wiimote
- Rumble contiene la información referente a la vibración del Wiimote.

3. Arquitectura del sistema.

3.2. Proyecto completo de detección de caídas.

El proyecto de detección de caídas se encuentra dividido en 3 paquetes claramente diferenciados e independientes que se presentan a continuación.

- MonitorizacionCaídas

Se trata de la aplicación que se encarga de recoger la información de los sensores, evaluarla y si se produce alguna llamada de emergencia o posible caída enviar una notificación al servidor.

- ServicioWeb

Se trata del conjunto de servicios web que permiten recibir las notificaciones de alarma.

- WiimoteLib

Es un paquete donde se encuentra la biblioteca creada por Brian Peek [8] que nos ayuda a leer los datos proporcionados por el Wiimote, la creación de esta biblioteca ha sido de gran utilidad para esta y muchas otras aplicaciones que utilizan el mando Wiimote.

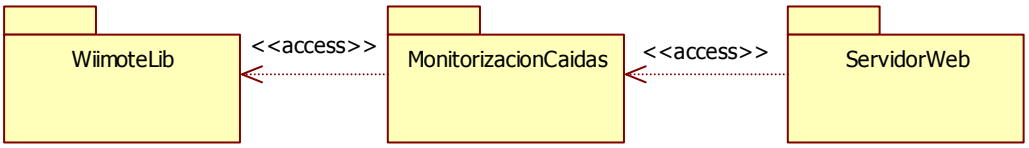


Figura 52. Estructura General del proyecto.

En el presente documento se va a detallar el paquete MonitorizacionCaídas puesto que es el principal y donde se encuentra la funcionalidad de la aplicación. El paquete WiimoteLib es una biblioteca en código abierto para la utilización del wiimote.

Para desarrollar la aplicación se han creado estructuras de datos nuevas dentro de clases de la biblioteca WiimoteLib.

Mientras que el paquete ServicioWeb esta creado como forma de presentación del proyecto ya que otros estudiantes están desarrollado un servidor que podrán utilizar simultáneamente tanto esta aplicación como otras dedicadas a la monitorización de pacientes.

3.3. Diagrama de paquetes para MonitorizacionCaídas

A continuación mostramos el diagrama general de clases para el paquete MonitorizacionCaídas, se ha dividido en más paquetes para seguir las coherencias con el modelo de diseño por capas, y en concreto el modelo de 3 capas: interfaz, lógica de negocio y persistencia.

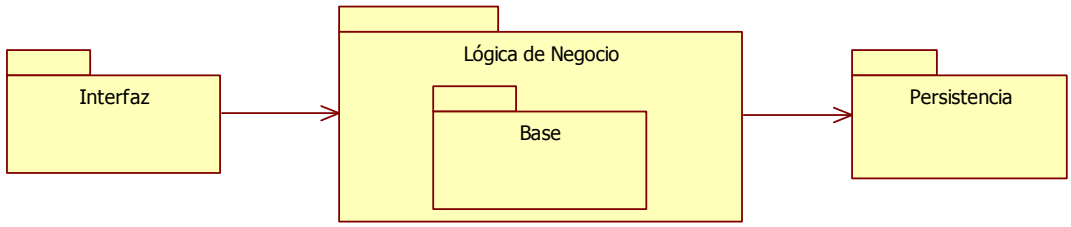


Figura 53 diagrama de paquetes.

La principal ventaja de utilizar el modelo de diseño por capas es que cada una de las tres capas puede evolucionar de manera independiente del resto.

En este proyecto la mayor parte de las clases desarrolladas están en la Lógica de Negocio puesto que es una aplicación que interactúa poco con el usuario y tiene un pequeño conjunto de datos.

Ahora daremos una pequeña explicación de las tres capas diseñadas.

- Capa de presentación.

Conocida como la interfaz de usuario, es la capa que utiliza el usuario para interactuar con la aplicación. Se compone de los formularios que dispone la aplicación para visualizar y configurar la aplicación. En este proyecto existen algunos formularios cuya única función es mostrar los diferentes cambios en la aplicación que se van produciendo, mientras que otros formularios permiten elegir opciones que intervienen en el funcionamiento de la aplicación.

- Capa de negocio.

Esta capa soporta toda la lógica de negocio de la aplicación, y es donde se coordinaran todas las lecturas de los sensores y se realizara el algoritmo de detección de caídas. Esta capa será la que proporcionará a la capa de presentación el estado de todos los sensores, y el estado de los pacientes. También interactuará con la capa de datos para la notificación de posibles alarmas.

- Capa de persistencia.

Es la capa encargada de interactuar entra la capa de negocio y los diferentes gestores de bases de datos, proporcionado independencia entre la aplicación y el gestor de la base de datos utilizado.

4. Diseño de la Capa de Presentación.

La interfaz de usuario es la presentación de la aplicación para los usuarios, y es la que se va a encargar de trasladar todo lo que ocurre al sistema. En toda aplicación es importante que esta capa se encuentre bien diseñada, ya que es la que realmente va a informa de lo que ocurre al usuario.

En nuestro caso la mayoría del tiempo la aplicación esta funcionado en modo pasivo, recogiendo información pero aun así no se ha descuidado la interfaz ya que esta va ser la encargada de hacer llegar al personal sanitario el estado de los sensores y de los pacientes.

4.1. Diagrama de clases

Presentamos a continuación las clases correspondientes a la capa de interfaz:

Se trata del conjunto de clases que nos permiten principalmente dos funciones, por una parte comprobar el estado de la aplicación y por otra comprobar el estado de los pacientes monitorizados.

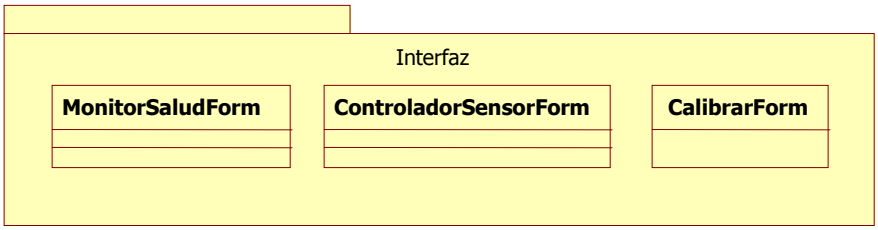


Figura 54 Clases en interfaz.

Podemos dividir las clases presentes en la interfaz en dos tipos las primera MonitorSaludForm y ControladorSensorForm nos van a mostrar información acerca del estado de los sensores y de los pacientes, mientras que la clase CalibrarForm nos sirven para configurar la aplicación. Ahora daremos una información detallada de las siguientes clases.

MonitorSaludForm está controlado por la clase MonitorSalud va ser el primer formulario que se va a ver cuando se inicie la aplicación, en el podemos observar el estado general para 2 sensores de caídas, este formulario nos va a permitir elegir las opciones siguientes para cada uno de los dos sensores:

Aplicación para la detección de caídas.

- Activar Detección.
- Desactivar Detección
- Calibrar Sensor.
- Asociar con Paciente
- Ver Características.

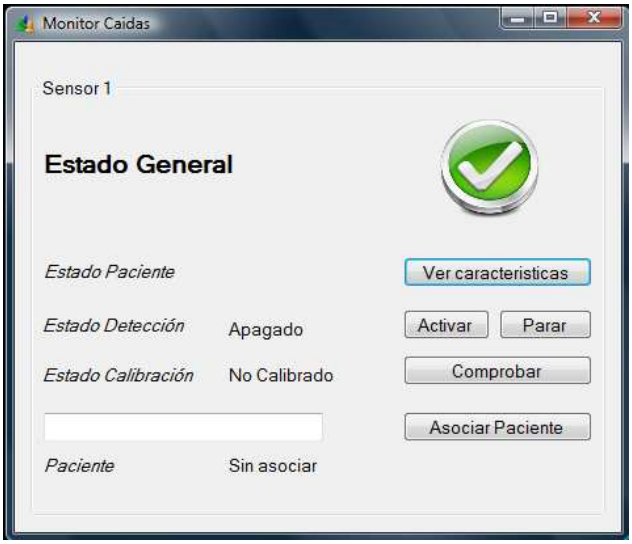


Figura 55 pantalla inicial aplicación.

ControladorSensorForm es el formulario que nos muestra la información detallada de cada Wiimote en particular, las características principales que se pueden ver en este formulario son las siguientes.

- Estado de la batería.
- Estado detección de caídas.
- Estado llamada de emergencia.
- Estado notificación de emergencia.
- Valores de los datos leídos del acelerómetro.
- Paciente asociado al wiimote.

Aplicación para la detección de caídas.

Figura 56 pantalla opción Ver Características.

CalibrarSensorForm es el formulario que indica al usuario como debe colocar el Wiimote y los pasos que debe seguir para calíbralo.

Figura 57 pantalla calibrar sensor.

5. Diseño de la capa lógica de negocio.

Esta es la capa más compleja de las tres que tenemos, ya que es la encargada de tener toda la funcionalidad del proyecto, que resumimos en:

- Lectura de datos
- Evaluación
- Envío datos y notificaciones.

También es la capa que se nos va a permitir configurar la aplicación, y sobre todo configurar los sensores correspondientes al paciente. Presentamos a continuación el Diagrama de Clases que concentra toda la funcionalidad de la lógica de negocio.

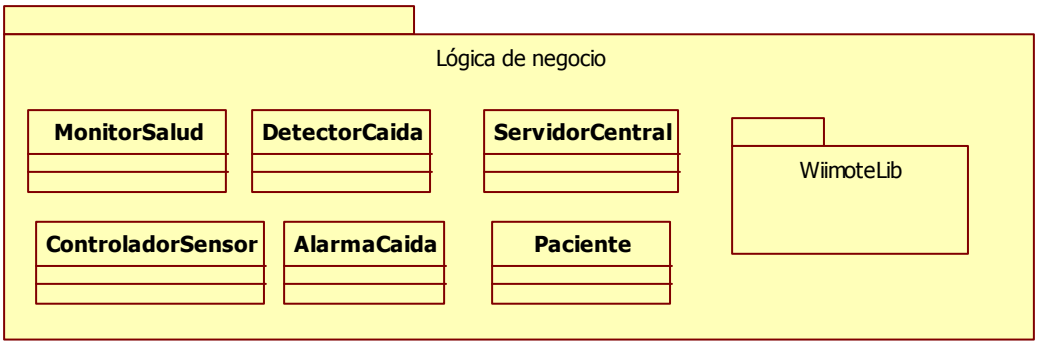


Figura 58 Clases en la lógica de negocio.

El paquete WiimoteLib se encuentra en la lógica de negocio ya que a partir de él se van a poder acceder a los datos de los Wiimotes conectados, lo hemos dejado como paquete para si algún día en vez de usar un Wiimote como sensor de caídas quisiéramos utilizar otro acelerómetro triaxial, de este modo solo tendríamos que cambiar un paquete por otro.

6. Diseño de la capa de persistencia.

En este proyecto la capa de persistencia no va tener casi importancia puesto que apenas se ha necesitado ninguna tabla para guardar los datos, solamente es necesaria una tabla que se utiliza para guardar los datos de calibración correspondientes a cada Wiimote.

Aplicación para la detección de caídas.

Implementación

1. Introducción

En este apartado se convertirán todas las especificaciones descritas hasta ahora en un sistema software real. Para conseguirlo se deben tomar las siguientes decisiones:

• Tecnología a utilizar.
• Entorno de desarrollo.
• Lenguaje de programación.
•Mecanismo de almacenamiento de información a utilizar para almacenar los datos de las configuraciones.
• Forma de acceder a la fuente de información para el manejo de los datos.

Debido al actual auge de la tecnología .NET se ha decidido desarrollar este proyecto ayudándose de esta tecnología. El entorno de desarrollo utilizado será Visual Studio.NET 2008, que es el entorno de desarrollo que ofrece .NET y se utilizará el lenguaje de programación C# que es el lenguaje nativo de .NET.

2. Software utilizado

A continuación mencionamos las aplicaciones que hemos usado:

- Microsoft Windows Vista Home Edition.
- Microsoft Visual Studio.NET 2008.
- Microsoft Word 2007
- Microsoft PowerPoint 2007
- REM 1.2.2
- Start UML
- BlueSoleil
- Adobe Reader 9.0
- Photoshop
- Microsoft .NET Framework SDK v. 3.0

Software en la edición móvil

- Microsoft .NET Compact framework 3.5

3. Hardware empleado.

Para la detección de caídas con Wiimote

- 2 Mandos Wiimote.
- PC sobremesa AMD Athlon 3200 Gh, 1GHz de RAM
- Portatil HP Pavilion 6000 Windows Vista Home edition.

Para la línea de productos monitorización de pacientes:

Aplicación para la detección de caídas.

- PDA HTC Touch Sistema Operativo Windows Mobile 6.0
- PDA HTC Diamond Sistema Operativo Windows Mobile 6.1
- Movil Sansumg Onmia Sistema Operativo Windows Mobile 6.1

En este apartado vamos a explicar cómo se ha utilizado la biblioteca WiimoteLib para conseguir programar la aplicación. Nos vamos a detener prestando atención en las funciones que se consideran más importantes así como las estructuras de datos que proporciona WiimoteLib.

4. Clases presentes en la el paquete WiimoteLib

El paquete wiimoteLib esta formado por la siguiente estructura de clases:

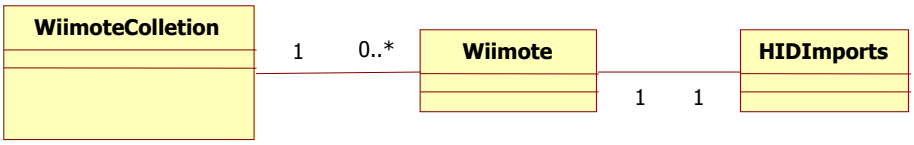


Figura 59.

Estas clases están tienen la siguiente utilidad.

La clase WiimoteCollection se utiliza para buscar todos los wiimotes que tenemos conectados en nuestro PC, sus tareas fundamentales son la búsqueda y conexión de objetos de tipo Wiimote.

La clase HIDImports es una clase que está programada a bajo nivel, permite reconocer el wiimote como un dispositivo hardware es decir reconocer el wiimote como si fuese un componente más de nuestro PC (un ratón o un teclado).

Wiimote es la clase más importante y fundamental en la biblioteca WiimoteLib, gracias a ella podemos acceder a las diferentes estructuras de datos, y funcionalidades que proporciona el wiimote.

Estas tres clases tienen los siguientes atributos y métodos:

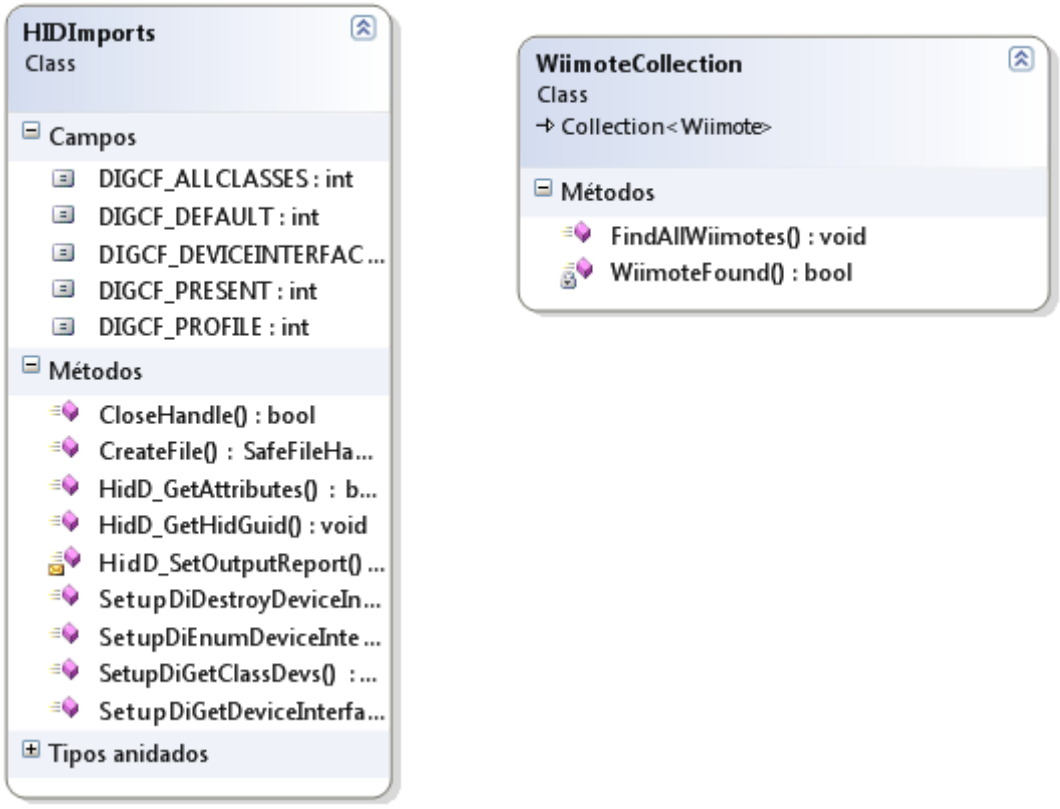


Figura 60 métodos biblioteca WiimoteLib.

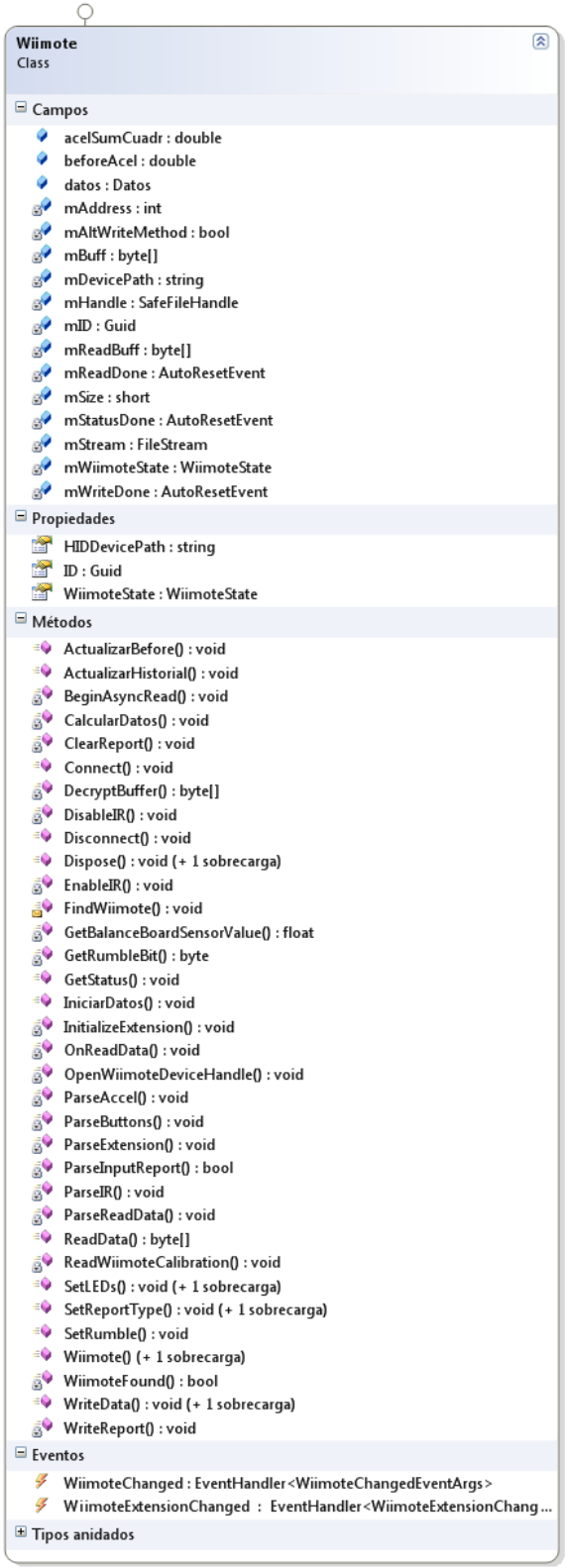


Figura 61 Wiimote Class.

Las estructuras de datos que están presente en el la clase wiimote y las cuales nos proporcionan los datos utiles del wiimote son las siguientes:

4.1. Struct ButtonState

Como su nombre indica es la estructura que representa a los diferentes botones que tenemos disponibles en nuestro mando wiimote, atributos son de tipo bool devuelve false si el botón no está pulsado y true si el botón esta pulsado.

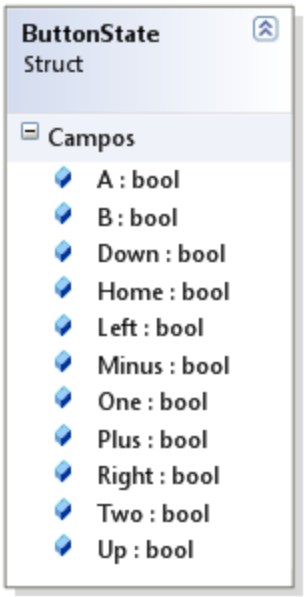
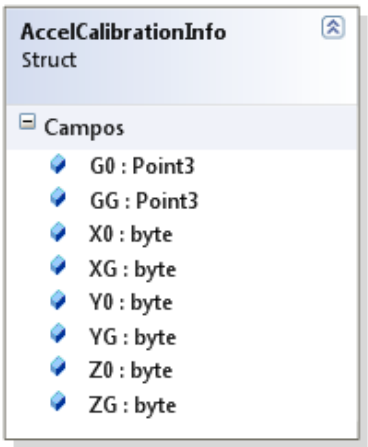


Figura 62 Struct ButtonStage.

4.2. Struct AccelCalibratioInfo

Esta estructura de datos representa los valores de calibración del wiimote, la forma de asignar estos valores los podemos ver detalladamente en el apéndice B del proyecto.



Aplicación para la detección de caídas.

Figura 63.

4.3. Struct LEDState

Esta estructura de datos nos permite conocer y cambiar el estado de los leds del wiimote, como ya hemos visto el mando wiimote tiene 4 leds representados por LED1, LED2, LED3 y LED4. Si el primer led esta luciendo su valor LED1 será true si por el contrario esta apagado el valor será false lo mismo ocurre con los demás leds.

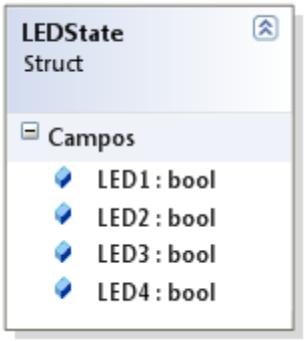


Figura 64.

4.4. Struct AccelState

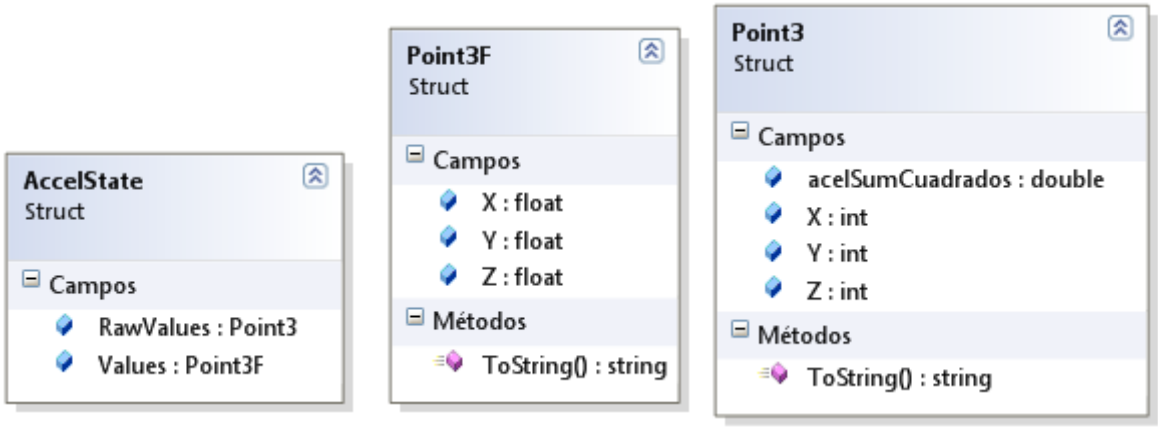


Figura 65.

La estructura AccelState nos permite acceder a los datos del acelerómetro del wiimote.

El método más importante que tiene la clase Wiimote es WiimoteState ya que gracias a él conocemos los valores actuales del mando Wiimote

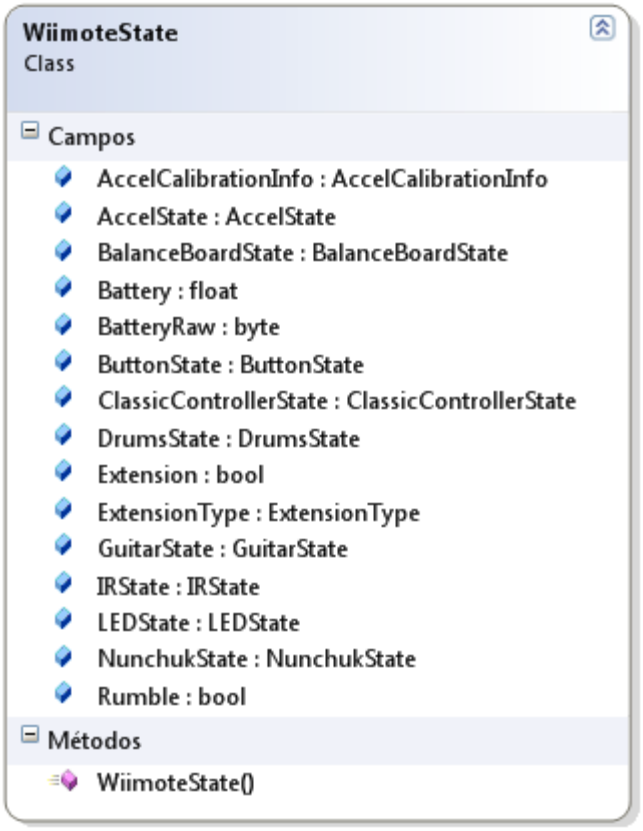


Figura 66.

Además de estas estructuras de datos para la implementación del algoritmo de detección de caídas se ha necesitado crear la siguiente estructura de datos

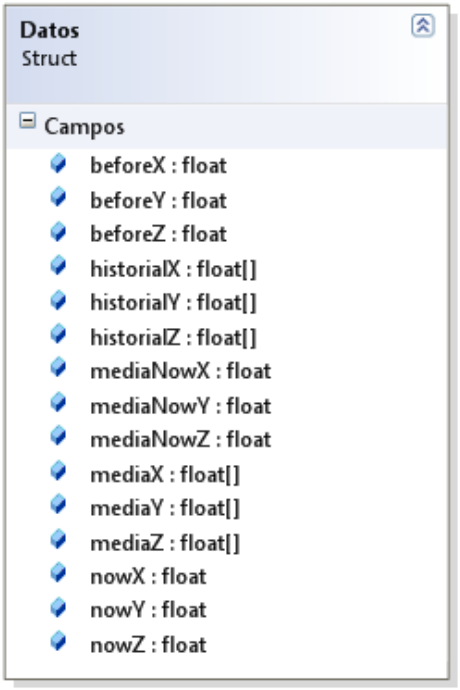


Figura 67.

Daremos una explicación de que representan estos campos,

- Los campos beforeX, beforeY, beforeZ representan la aceleración anterior a la aceleración en cada uno de los ejes cartesianos respectivamente.
- Los campos historialX[],historialY[], historialZ[] son una array de datos cuyo significado es el siguiente:
    - HistorialX[1] representa el valor de la aceración un segundo antes de la aceleración actual.
    - HistorialX[2] representa el valor de la aceleración en el eje X dos segundos antes de la aceleración actual.
    - Los valores para los ejes Y e Z son similares.
- Los campos mediaX[], mediaY[], mediaZ[] representan la media aritmética de las últimas aceleraciones leidas en cada una de los ejes mediaX[1] = media aritmética aceleraciones en el eje un segundo anterior al actual.
- Los campos mediaNowX, mediaNowY, mediaNowZ representan la media aritmética de las últimas 150 aceleraciones leidas en cada una de los ejes.

5. Algunas funciones presentes en el proyecto de detección de caídas

Una vez que hemos nombrados las clases, atributos y métodos vamos a explicar cuál ha sido su utilidad.

5.1. Conexión y búsqueda de wiimotes.

Es la primera función que se realiza cuando se inicia la aplicación, esta presente en la clase MonitorCaídasForm() vamos a explicar brevemente como funciona.

```
private void MonitorCaídasForm_Load(object sender, EventArgs e)
      {
          mWC = new WiimoteCollection();
          int index = 1;

          try
          {
              mWC.FindAllWiimotes(); //Buscar wiimotes

          }
          catch (WiimoteNotFoundException ex)
          {
              MessageBox.Show(ex.Message,            }
          catch (WiimoteException ex)
          {
              MessageBox.Show(ex.Message);           }
          catch (Exception ex)
          {
              MessageBox.Show(ex.Message);            }

          foreach (Wiimote wm in mWC)
          {


              monitor.addSensor(wm);
              wm.Connect(); //Conectar Wiimote
              wm.SetLEDs(index++); //Iluminar led index
          }

      }
```

Esta función busca todos los wiimotes conectados a nuestro PC gracias al método FindAllWiimotes una vez que hace esto por cada wiimote que encuentra conectado realiza la conexión con el método Connect( ) e ilumina el leds del wiimote que corresponda, es decir cada wiimote conectado tendrá un led diferente iluminado.

5.2. Actualización de los valores del wiimote.

Aplicación para la detección de caídas.

En todo momento existe en la aplicación un evento se encarga de actualizar los datos del wiimote (aceleraciones, botones pulsados, estado batería etc) y mostrar los resultados en un formulario:

```
public void UpdateState(WiimoteChangedEventArgs args)
        {
            BeginInvoke(new UpdateWiimoteStateDelegate(UpdateWiimoteChanged),
args);
        }
private void UpdateWiimoteChanged(WiimoteChangedEventArgs args)
        {
            //Función que actualiza los datos y la interfaz

            ws = args.WiimoteState; //Recojida de datos del wiimote
            AccelState at = ws.AccelState; //Creación de la nueva estrutura
de datos

            ori = ComprobarOrientacion(mWiimote.datos.mediaNowX,
            mWiimote.datos.mediaNowY, mWiimote.datos.mediaNowZ);

            if (ori != -1)
            {
                label27.Text = ComprobarOrientacionString(ori);
            }
               //Escritura de los datos leidos en el formulario.
            lblAccel.Text = ws.AccelState.Values.ToString();

            Label lx = new Label();
            lx.Text = at.RawValues.X.ToString();

            labelEstadoAlarma.Text = TextoLabelAlarma;
            EstadoCaída.Text = TextoLabelCaída;
            LabelAlarmaEnviada.Text = TextoLabelServidor;
            Apagado.Text = TextoLabel1;

            labelBotonPanico.Text = TextoLabelBotonPanico;
            pbBattery.Value = (ws.Battery > 0xc8 ? 0xc8 : (int)ws.Battery);
            float f = (((100.0f * 48.0f * (float)(ws.Battery / 48.0f))) /
192.0f);
            lblBattery.Text = f.ToString("F");

            progressBar1.Value = ValorBarra;

        }
```

5.3. Implementación Función aplicarAlgoritmo()

En la implementación del algoritmo de detección de caídas se han utilizado 2 hilos diferentes uno de ellos es el encargado de actualizar los datos y guardarlos en el historial de datos, el otro hilo se encarga de realizar el algoritmo de detección de caídas,

//Creación del hilo de detección de caídas

```
hiloDeteccion = new Thread(new ThreadStart(ActivarDeteccion));
```

```
public bool aplicarAlgoritmo()
        {
            try
            {
                if (nLlamadas == 0)
                {
                    //Actualizacion historial de datos cada medio segundo
                    aTimer = new System.Timers.Timer(10);
                    aTimer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
                    //Actualizacion historial de datos cada medio segundo
                    aTimer.Interval = 500;
                    aTimer.Enabled = true;
                    nLlamadas++;
                }
                if (ComprobarImpacto(mWiimote.acelSumCuadr) == true)
                {
                     /* Guardamos  la orientacion 2 segundo antes del primer
impacto*/
                orientacion2 = ComprobarOrientacion(mWiimote.datos.mediaX[2],
mWiimote.datos.mediaY[2], mWiimote.datos.mediaZ[2]);

                    int numeroImpactos = 1;
                    dt = DateTime.Now;
       /*Bucle que se queda esperando a que se produzcan 2 segundos seguidos
sin que exista ningun impacto*/
                    while (DateTime.Now < dt.AddSeconds(2))
                    {

                        if (mWiimote.acelSumCuadr != mWiimote.beforeAcel)
                        {
                            if (ComprobarImpacto(mWiimote.acelSumCuadr) ==
true)
                            {
                                dt = DateTime.Now;
                                numeroImpactos++;
                            }
                        }
                    }
                    numeroImpactos = 0;
       //Comprobamos la orientacion 2 segundo despues del ultimo impacto

         orientacionNow = ComprobarOrientacion(mWiimote.datos.mediaNowX,
       mWiimote.datos.mediaNowY, mWiimote.datos.mediaNowZ);
                    if (orientacionNow != orientacion2)
                    {
                    //Si la orientación es diferente devolvemos falso se ha
producido una caída
                        return false;
                    }

                }

            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }
            return true;
        }
```

5.3.1. Función comprobar impacto

La función aplicarAlgoritmo antes descrita realiza sucesivas llamadas a la función ComprobarImpacto(mWiimote.acelSumCuadr), la implementación de esta función es la siguiente:

```
private bool ComprobarImpacto(double sumCuadrados)
        {

            //impactos debidos a valores fuera de los umbrales establecidos
            bool impacto = false;

            if (sumCuadrados < 0.41 || sumCuadrados > 3.54)
            {
               //umbrales establecidos en el Capitulo 2.

                        impacto = true;
            }

            return impacto;
        }
```

5.3.2. Función comprobar orientación.

Otra de las funciones a las que llama el algoritmo de detección de caídas es la función `ComprobarOrientacion (float x, float y, float z)`

```
private int ComprobarOrientacion(float x, float y, float z)
        {
            int posicion = -1;
            if ((y < 1.2) && (y > 0.7)) /* de 90 a 45 grados*/
            {
                posicion = 0;
            }
            else if ((z < 1.2) && (z > 0.7))
            {
                posicion = 3;
            }
            else if ((z < -0.7) && (z > -1.2))
            {
                posicion = 4;
            }
            else if ((x < 1.2) && (x > 0.7))
            {
                posicion = 1;
            }

            else if ((x < -0.7) && (x > -1.2))
            {
                posicion = 2;
            }
            else if ((y < -0.7) && (y > -1.2))
            {
                posicion = 5;
            }
            return posicion;
        }
```

Aplicación para la detección de caídas.

Los parámetros que recibe esta función son una media aritmética de 150 valores leídos

Aquí aparece la estructura de datos que hemos creado donde mediaNowX es la media aritmética de los últimos 150 valores leídos en el acelerómetro del eje X, lo mismo ocurre con los ejes Y, y Z.

5.4. Como conocemos la orientación que tiene en todo momento el wiimote.

Para calcular la orientación del wiimote utilizamos los valores de la aceleración en cada eje por separado podemos apreciar la diferencia de estos valores según la orientación en la siguientes gráficas:

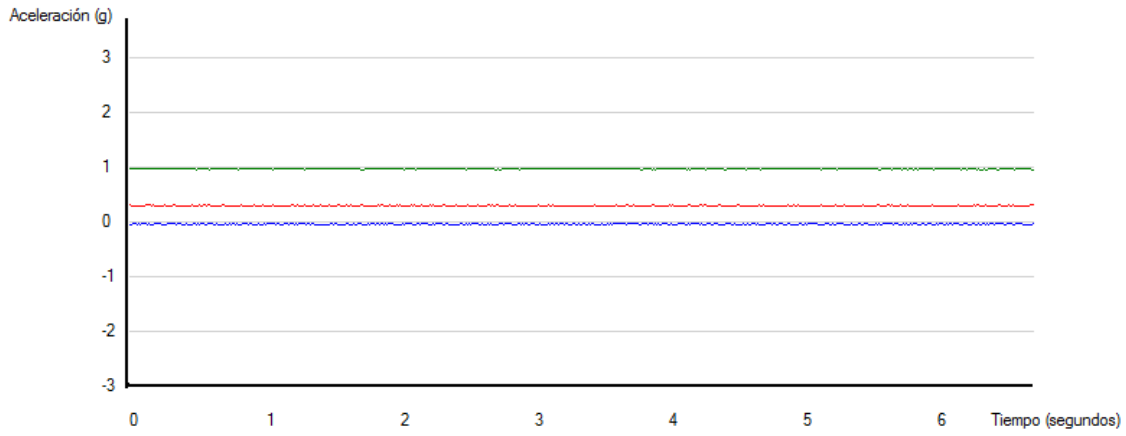


Figura 68. Gráfica aceleraciones en los 3 ejes del mando wiimote situado horizontalmente sobre una superficie plana.

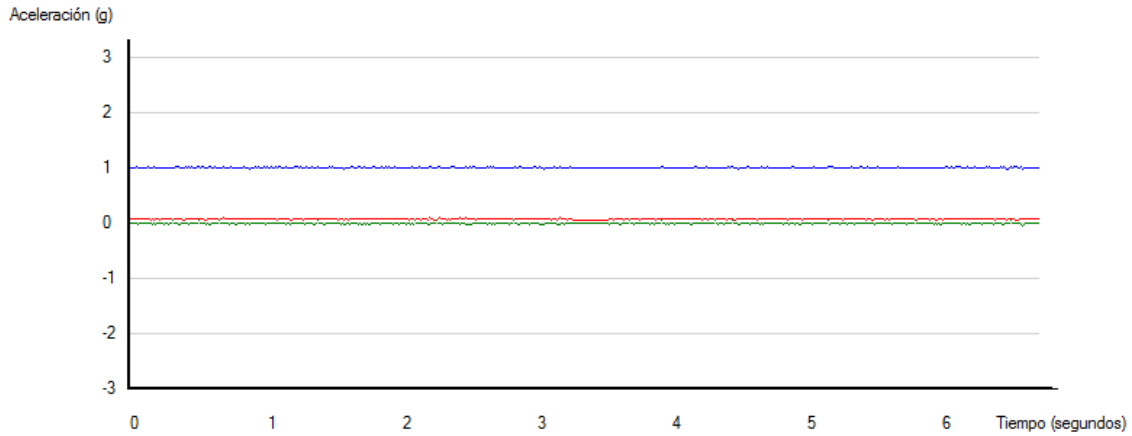


Figura 69. Gráfica aceleraciones en los 3 ejes del mando wiimote colocado en posición vertical sobre una superficie plana.

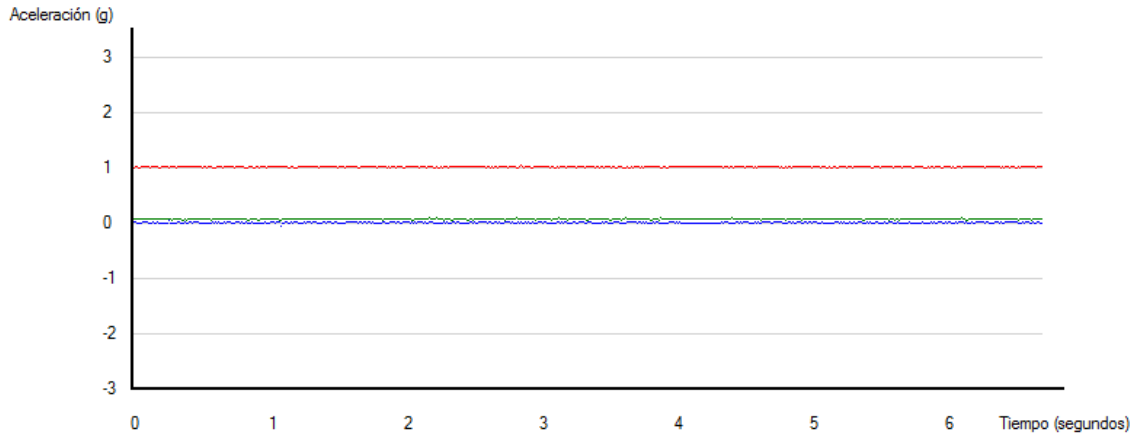


Figura 70. Gráfica aceleraciones en los 3 ejes del mando wiimote colocado en posición lateral sobre una superficie plana.

El color verde pertenece al eje Z como hemos visto cuando el mando se encuentra en posición horizontal su valor es próximo a 1, el color azul es el correspondiente al eje Y su valor es próximo a 1 cuando el mando está en posición vertical, mientras que el eje X está representado por el color rojo su valor es próximo a uno cuando se sitúa el mando sobre uno de sus laterales.

5.5. Implementación de la función LlamadaEmergencia

Otra de las funciones que se han considerado importantes ha sido la llamada de emergencia esta función se realiza mediante el hilo llamado hiloEmergencia.

```
hiloEmergencia = new Thread(LlamadaEmergencia);

public void LlamadaEmergencia()
        {
            AlarmaCaída ac;

                while (pararEmergencia == false)
                {
//Esta pendiente en todo momento de si se pulsa el botón con forma de cruz
del mando
                    if (sensor.WiimoteState.ButtonState.Up == true)
                    {
                        ac = new AlarmaCaída();
                        formulario1.estadoLlamadaRealizada();
                        pararEmergencia = true;
                        activarAlarma();
                    }
                }
```

```
        }
```

Aplicación para la detección de caídas.

Aplicación para la detección de caídas.

Pruebas

1. Introducción

Este documento recoge los casos de prueba diseñados para detectar posibles errores en la aplicación desarrollada en el proyecto, por tanto quedan excluidas de las pruebas las aplicaciones que no intervienen en la aplicación.

Todas las pruebas que presentamos en el documento son de caja negra, diseñadas para encontrar errores funcionales en el proyecto, y por tanto errores sobre los casos de uso planteados en el análisis. Se realizaran pruebas sobre cada caso de uso, comprobando que la aplicación cumple con los requisitos especificados y que funciona de la manera esperada.
Las pruebas de caja blanca no se han incluido debido a su complejidad. Estas fueron realizadas en
la etapa de programación de cada clase del proyecto, aprovechando la funcionalidad de Visual
Studio 2008 para comprobar el buen funcionamiento de cada método del proyecto.

2. Casos de prueba aplicación.

Organizaremos los casos de prueba en función de cada uno de los casos de uso. Utilizaremos la siguiente plantilla para definir la prueba

Prueba	Tipo	Resultado	Valoración
Descripción detallada de la prueba a realizar	Funcional (F), Interfaz(I)	Resultado observado de la prueba	Correcto (OK), Incorrecto(NOK)

UC-001 Elegir Sensor.

Este caso de uso se utiliza para que el personal sanitario pueda comprobar la correcta calibración del Wiimote, si este no está bien calibrado la detección no será fiable.

Prueba	Tipo	Resultado	Valoración
Con 2 wiimotes conectados pinchar la opción de menú Ver características en el primero de ellos.	Funcional (F), Interfaz(I)	La aplicación responde con los datos del wiimote seleccionado.	OK
Con 2 wiimotes conectados pinchar la opción de menú Activar en el primero de ellos.	Funcional (F) Interfaz(I)	La aplicación solo activa la detección de caídas en el wiimote seleccionado.	OK
Con 2 wiimotes conectados pinchar la opción de menú	Funcional (F) Interfaz(I)	La aplicación solo desactiva la detección de caídas en el wiimote seleccionado.	OK

Desactivar en el primero de ellos.			
Con 2 wiimotes conectados pinchar la opción de menú Asociar Paciente en el primero de ellos.	Funcional (F) Interfaz(I)	La aplicación asocia paciente en el wiimote seleccionado.	OK
Repetir las pruebas anteriores en el en el menú del segundo wiimote.	Funcional (F) Interfaz(I)		OK

UC-002 Comprobar Calibración.

Este caso de uso se utiliza para que el personal sanitario pueda comprobar la correcta calibración del Wiimote, si este no está bien calibrado la detección no será fiable.

Prueba	Tipo	Resultado	Valoración
Comprobar Calibración pulsando el botón de interfaz Comprobar.	Funcional (F), Interfaz(I)	La aplicación comprueba la Calibración del mando si no la encuentra lanza el caso de uso Calibrar Sensor	OK
Comprobar Calibración al iniciar la detección.	Funcional (F)	La aplicación comprueba la calibración correcta del mando wiimote siempre que se inicia la detección de caídas.	OK

UC-003 Calibrar Sensor.

Este caso de uso se utiliza cuando el caso de uso Comprobar Calibración no ha encontrado una Calibración correcta.

Prueba	Tipo	Resultado	Valoración
Calibrar Sensor	Funcional(F) Interfaz (I)	La aplicación calibra el Wiimote con ayuda del usuario.	OK

UC-004 Asociar paciente con wiimote.

Este caso de uso se lleva a cabo cuando el usuario de la aplicación pulsa el botón asociar paciente

Prueba	Tipo	Resultado	Valoración

Si en el usuario ha escrito un nombre de paciente vacio.	Funcional(F) Interfaz (I)	La aplicación lanza un mensaje pidiendo un nombre no vacio	OK
Si en el usuario ha escrito un nombre de paciente no vacio.	Funcional(F) Interfaz (I)	La aplicación asocia el nombre de paciente con el wiimote correspondiente.	OK

UC-005 Iniciar detección

Este caso de uso sirve para iniciar la detección de caídas.

Prueba	Tipo	Resultado	Valoración
Pulsar el botón iniciar Detección.	Funcional(F) Interfaz (I)	Comienza a ejecutarse el algoritmo de detección de caídas y no para hasta que, se pulse el botón Parar	OK

UC-006 Parar detección

Este caso de uso es el contrario al anterior se encarga de desactivar la detección de caídas.

Prueba	Tipo	Resultado	Valoración
Pulsar el botón iniciar el Detección cuando la detección esta desactivada	Funcional(F) Interfaz (I)	Comienza a ejecutarse el algoritmo de detección de caídas y no para hasta que, se pulse el botón Parar	OK
Pulsar el botón iniciar Detección cuando la detección de caídas se activo anteriormente	Funcional(F) Interfaz (I)	Queda sin efecto	OK

UC-007 Consultar estado Paciente UC-008 Consultar estado Sensores.

Prueba	Tipo	Resultado	Valoración

Pulsar el botón Ver características con la ventana Controlador Sensor no visible.	Funcional(F) Interfaz (I)	Se visualiza la ventana Controlador Sensor donde se pueden ver las siguientes características: Nombre del paciente Orientación actual en cada momento del paciente. Estado de las alarma Estado de las caídas Estado de la llamada de emergencia Última alarma enviada a servidor Porcentaje de la batería. Numero de Wiimote Aceleraciones producidas por el wiimote	OK
Pulsar el botón Ver características con la ventana Controlador Sensor ya visible.	Funcional(F) Interfaz (I)	Queda sin efecto ya que la información ya esta visible.	OK

UC-009 Llamada de emergencia

Prueba	Tipo	Resultado	Valoración
Comprobar si el mando vibra cuando se pulsa el botón de emergencia.	Funcional(F)	El mando vibra durante 2 segundos	OK
Comprobar si los leds del mando parpadean cuando se pulsa el botón de emergencia.	Funcional(F)	Los leds del mando parpadean durante 10 segundos.	OK
Comprobar que se produce una alarma cuando se pulsa el botón de emergencia.	Funcional(F) Interfaz (I)	La aplicación crea una alarma que advierte de la existencia de una emergencia.	OK
Pulsar el botón de emergencia cuando ya se ha pulsado este con una anterioridad menor a	Funcional(F) Interfaz (I)	No se crea otra alarma ya que ya existe una en proceso.	OK

10 segundos.			

UC-010 Detener notificación de emergencia UC-011 Detener notificación de caída.

Prueba	Tipo	Resultado	Valoración
Pulsar el botón A del wiimote cuando hay una alarma en proceso.	Funcional(F) Interfaz (I)	Se elimina la alarma.	OK
Pulsar el botón A del wiimote cuando no hay ninguna alarma en proceso.	Funcional(F)	Queda sin efecto	OK

UC-012 Activar alarma

Prueba	Tipo	Resultado	Valoración
Pulsar el botón de emergencia.	Funcional(F) Interfaz (I)	Se crea una alarma	OK
Tirar el wiimote al suelo desde una altura de 30 cm.	Funcional(F)	Se crea una alarma.	OK

UC- 013 Aplicar Algoritmo Detección de caídas.

Prueba	Tipo	Resultado	Valoración
Pulsar el botón iniciar detección.	Funcional(F) Interfaz (I)	Se aplica el algoritmo de detección de caídas.	OK

3. Casos de prueba del algoritmo de detección.

Con las siguientes pruebas se va intentar comprobar la eficacia del algoritmo de detección de caídas creado, hay que tener en cuenta que estas pruebas han sido simuladas, no se ha podido probar el algoritmo con caídas reales. Se van a dividir estas pruebas en 2 fases la primera de ellas se realizaran una serie de movimientos y se comprobará cuantas falsas alarmas se producen, mientras que en la segunda fase se simularan una serie de caídas a ver cuantas es capaz de detectar de forma correcta.

3.1. Pruebas diferentes movimientos.

Vamos a realizar un total de 10 pruebas de cada uno de los siguientes movimientos

- Andar 20 metros. 10 pruebas 0 falsos negativos
- Bajar escaleras. 10 pruebas 0 falsos negativos
- Sentarse en una silla. 10 pruebas 0 falsos negativos
- Sentarse en el sofá. 10 pruebas 0 falsos negativos
- Subir 7 escaleras. 10 pruebas 0 falsos negativos
- Tumbarse en la cama. 10 pruebas 5 falsos positivos.
- Levantarse de la cama 10 pruebas 5 falsos positivos
- Correr 20 metros. 10 pruebas 0 falsos negativos

3.2. Pruebas simulando caídas

Caídas simuladas por personas.

- Caída de frente 10 pruebas, 10 reconocimientos de caídas.
- Caída marcha atrás 10 pruebas, 10 reconocimientos de caídas.
- Caída de lado 10 pruebas, 10 reconocimientos de caídas.
- Caída marcha atrás manteniendo el tronco recto después de la caída. 10 pruebas ninguna detección de caídas.
- Pruebas situando el mando en un palo de escoba y tirándolo al suelo. 25 pruebas 25 aciertos.

4. Conclusiones pruebas realizadas en la detección del algoritmo.

Las pruebas que hemos realizado han sido bastante satisfactorias, en total se han realizado 65 pruebas simulando caídas de las cuales la aplicación a logrado identificar 55 pruebas con una tasa de acierto del 84.6%.

Se podría considerar esta tasa de acierto baja, pero hay que tener en cuenta otra observación, las caídas simuladas que la aplicación no ha sido capaz de reconocer, se han producido todas en un mismo tipo de caída, la caída marcha atrás manteniendo el tronco recto después de la caída (“caída de culo”). Que no reconozca este tipo de caída era de esperar si observamos la forma en que está diseñado el algoritmo de detección de caídas. Este algoritmo busca un impacto y un cambio de orientación en el caso de este tipo de caída se produce el impacto, el cual localiza pero no el cambio de orientación puesto que el wiimote continua estando vertical al igual que el tronco de la persona.

Aplicación para la detección de caídas.

Durante las pruebas en los movimientos comunes la tasa de acierto fue de 88,8% la aplicación detecto 10 falsos positivos de 90 pruebas realizadas.

En este caso nos paso una situación similar a la anterior ya que los falsos positivos se han producido en un tipo de movimiento especifico, tumbarse y levantarse de la cama.

Al tumbarse en la cama se produjeron falsos positivos cuando se realizó el movimiento de tumbarse sin haberse sentado antes primero. En las pruebas donde el sujeto primero se sentaba en la cama y luego se tumbaba no hubo falsos positivos.

Lo mismo sucedió al levantarse, en las pruebas levantándose dando un "salto" se produce falso positivo pero si pasamos de estar recostados a sentarse y después levantarse no hubo falsos positivos.

El detector de caídas está pensado para personas mayores cuya movilidad es limitada por lo tanto sería raro el caso en que estas personas se acostaran y levantaran de forma brusca, en las caídas marcha atrás con el tronco vertical habría que pensar otro modo de identificarlas pero también podemos pensar que si la persona se queda en esta postura es consciente para pulsar la llamada de emergencia.

Como conclusión el algoritmo ha funcionado de manera exitosa en las pruebas realizadas, nos queda probarle en la vida real para obtener unas medidas más reales.

Aplicación para la detección de caídas.

Manual de usuario.

Aplicación para la detección de caídas.

1. Instalación

Abrimos el disco de instalación doble click en setup

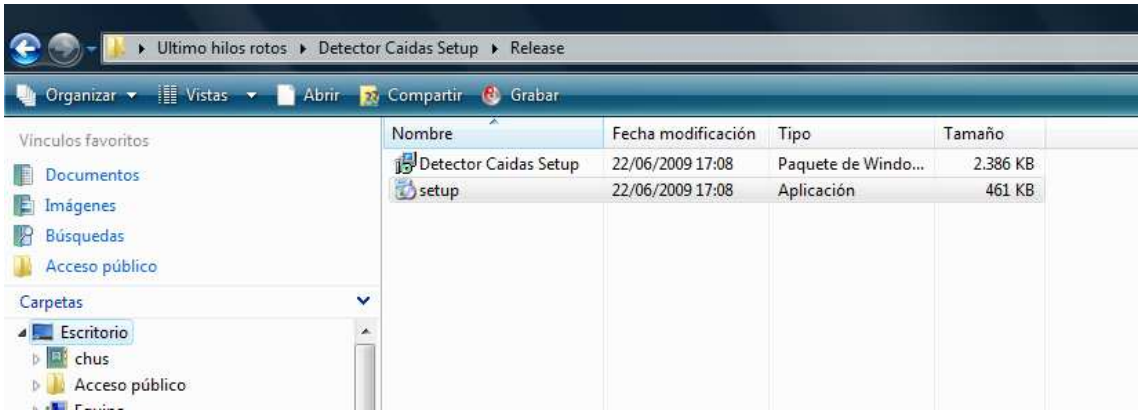


Figura 71.

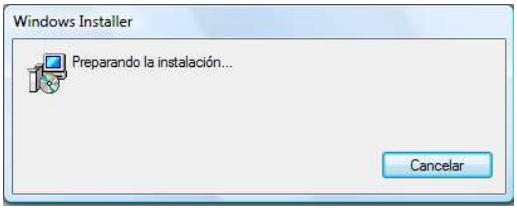


Figura 72.

Veremos la siguiente ventana y segundos después aparecerá el asistente de instalación

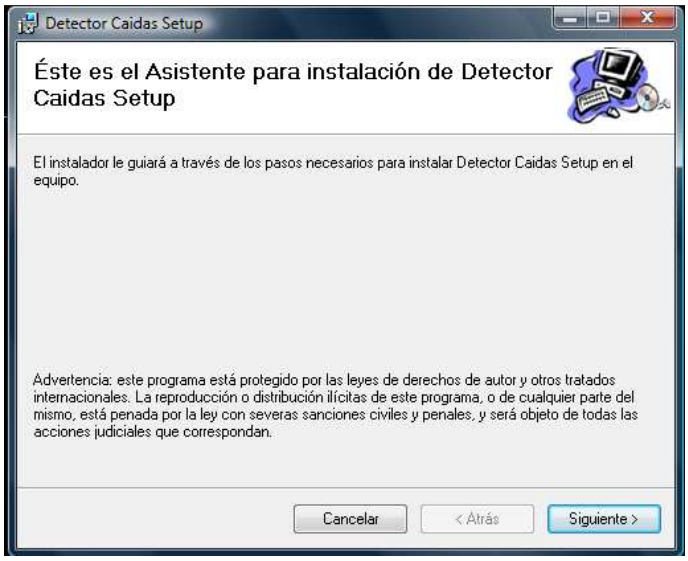


Figura 73.

Figura 74.

Elegimos la carpeta donde queremos que se instale el programa

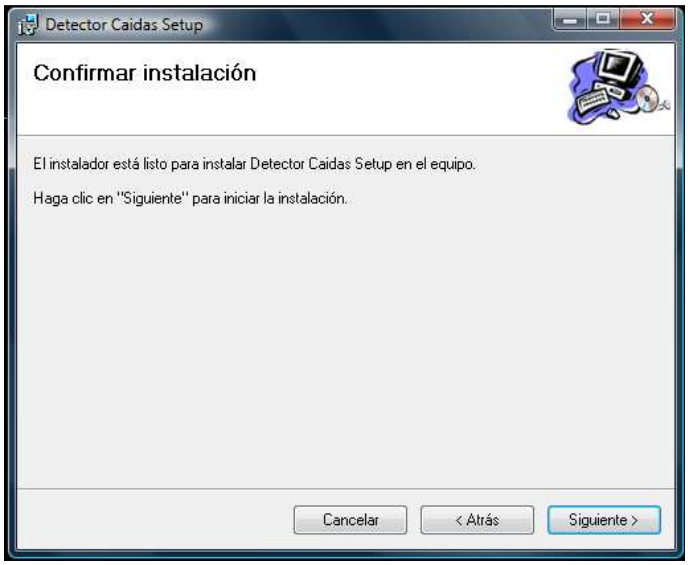


Figura 75.

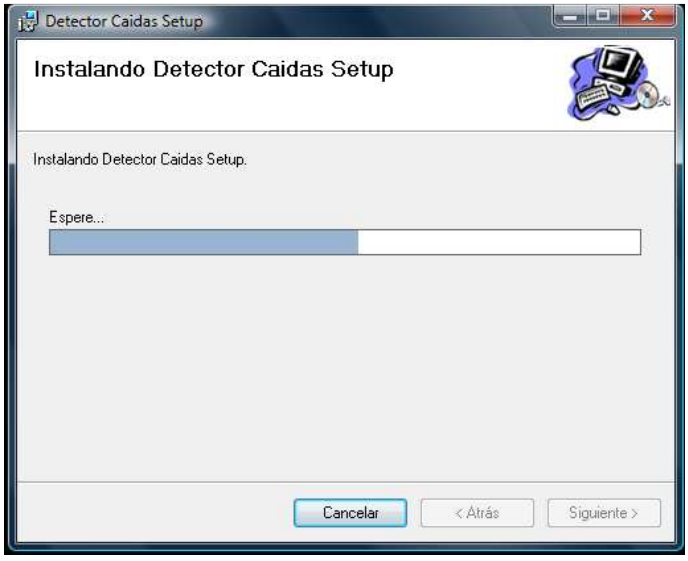


Figura 76.

Aplicación para la detección de caídas.

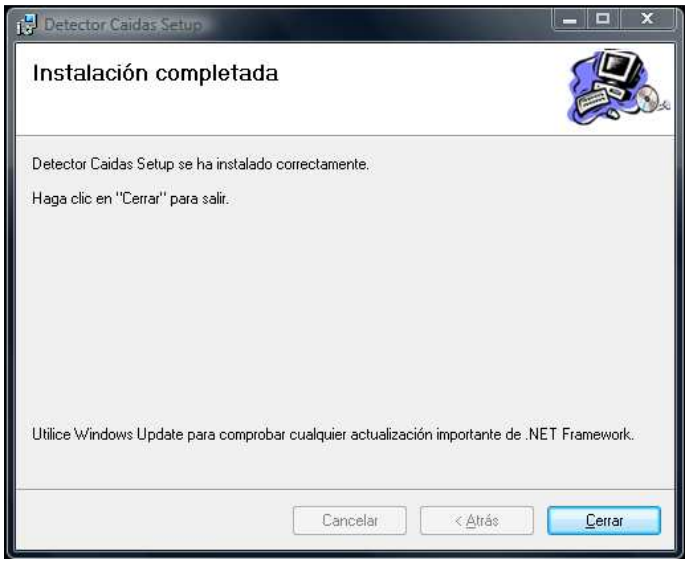


Figura 77.

2. Conexión del wiimote al PC.

Para poder iniciar nuestra aplicación lo primero que tenemos que hacer es conectar el mando wiimote vía bluetooth con nuestro PC. Es necesario que nuestro PC tenga instalado una antena bluetooth.

Para hacer esto podemos seguir dos caminos, el primero de ellos es utilizar la aplicación Bluesoleil.

Advertencia para poder realizarse esta conexión necesitamos que nuestro PC tenga instalado un adaptador bluetooth. Una vez conectado el PC y el wiimote estos no pueden alejarse más de la distancia que venga especificada por el fabricante de nuestro adaptador, esta distancia variará entre 10 y 200 metros.

2.1. Conexión del wiimote con la aplicación Bluesoleil.

Antes de empezar veamos lo que necesitaremos:

BlueSoleil XP: Programa de conexión Bluetooth que permitirá crear una conexión Mando_wii/PC sin que se pierda al pasar un tiempo.

BlueSoleil Vista y XP:. En realidad hace lo mismo que la versión anterior, pero es más moderna, mayor velocidad y mejor conectividad.

Conexión del Wiimote al PC:

Aplicación para la detección de caídas.

Para este paso necesitaremos el BlueSoleil. Este es un programa que requiere de instalación, una vez que ya lo hayamos instalado,y reiniciado el PC, nos aparecerá una ventana con los datos del administrador, la conexión Bluetooth y un botón para activar la protección media. Ponerlo sin protección media, y al aceptar todo, veremos lo siguiente:

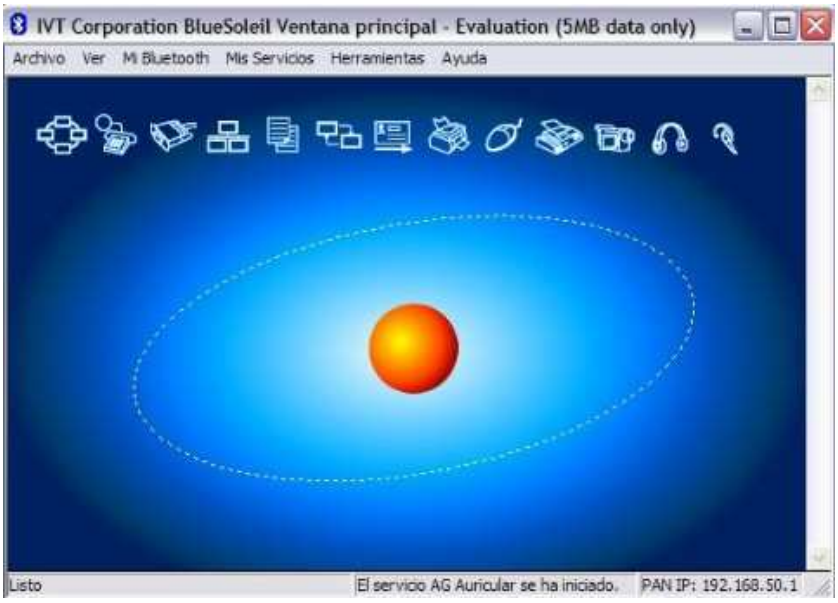


Figura 78. Imagen ventana principal aplicación Bluesoleil

Ahora iremos a Mi Bluetooth, y pulsaremos Búsqueda de dispositivo Bluetooth:

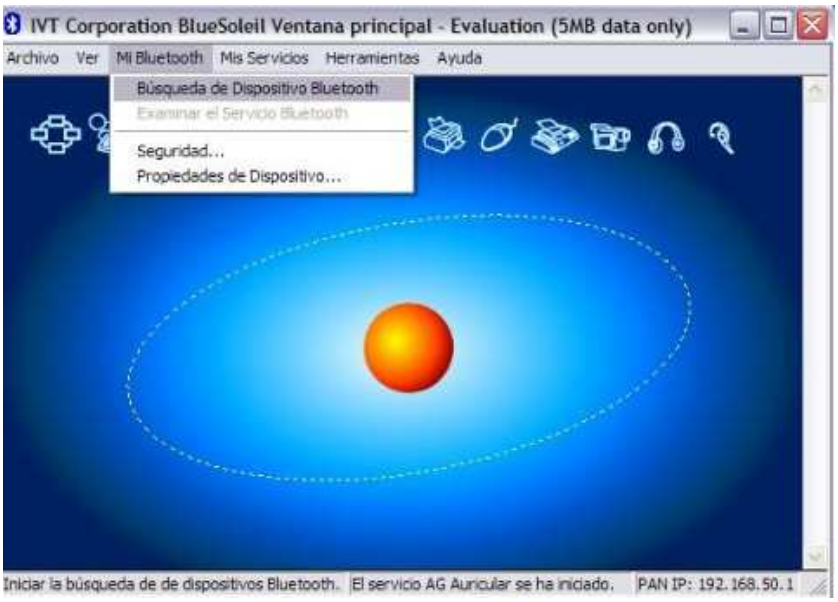


Figura 79. Imagen menú Mi bluetooth aplicación Bluesoleil

Antes de iniciar la búsqueda hay que tener el mando de Wii sin la tapadera de las pilas, ya que ahí se encuentra el botón de sincronización. Inmediatamente después de pulsar la Búsqueda de dispositivos Bluetooth presionaremos el botón de sincronización del mando. Tras la primera sincronización las siguientes se pueden hacer pulsado el **botón 1 y 2** simultáneamente.

Figura 80 Wiimote sin tapa.

Una vez hecho esto el programa nos reconocerá el mando como un dispositivo Bluetooth, con el nombre de Nintendo RVL-CNT 01. Aún así, todavía no tenemos el mando conectado a BlueSoleil, solo nos lo ha reconocido. Para ello pulsamos con el botón derecho encima del

wiimote, y pulsamos a Actualizar Servicios, e inmediatamente después el botón de sincronización del mando:

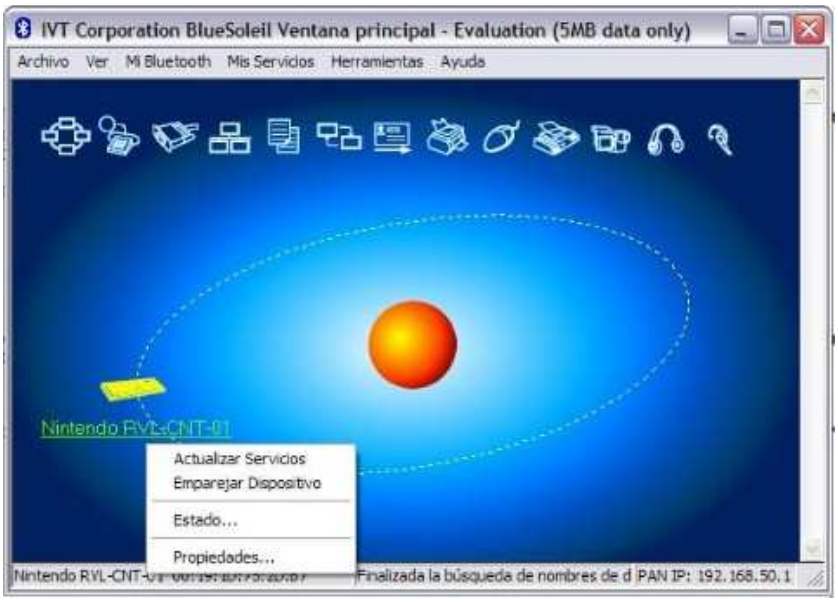


Figura 81. Imagen menú actualizar servicios aplicación Bluesoleil

Volvemos a pulsar botón derecho sobre el Wiimote, y ahora pulsamos Conectar. Si hemos hecho todo correctamente, el mando se pondrá en verde y se verá una línea que une el mando y el "sol", además tiene que aparecer en la barra inferior derecha de Windows un letrero de "Nuevo hardware encontrado", debemos esperar a que lo instale:

Aplicación para la detección de caídas.

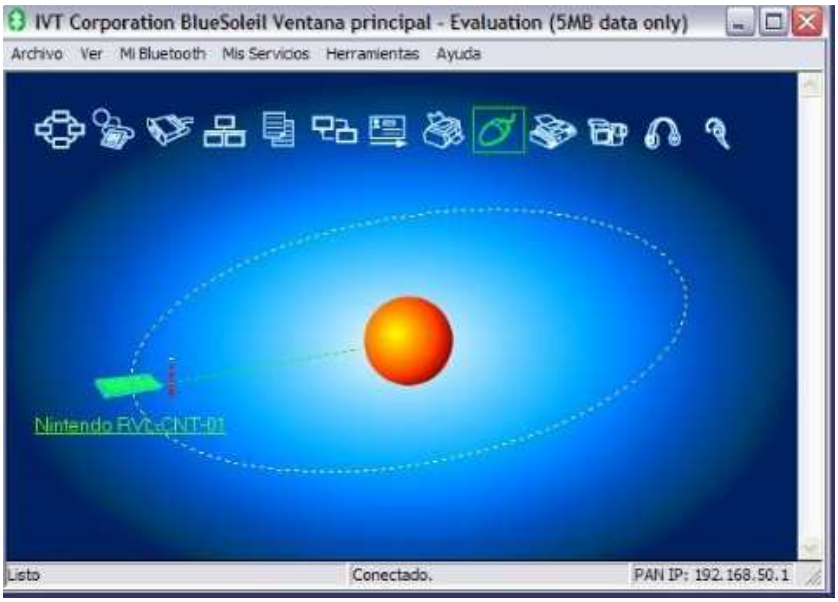


Figura 82 Wiimote conectado a Bluesoleil.

Cada vez que queramos conectar el mando, después de haber cerrado el programa, tendremos que Actualizar servicios y luego conectar. Para dejar de usar el mando sin cerrar el programa, simplemente botón derecho sobre el mando de wii, y después desconectar.

2.2. Conectar el wiimote con la aplicación bluetooth de Windows.

El segundo método de conectar el PC al wiimote es usar el driver bluetooth genérico de Windows para ello seguiremos los siguientes pasos: Menú Inicio → Panel de control → Dispositivos Bluetooth

Nos aparecerá la siguiente ventana

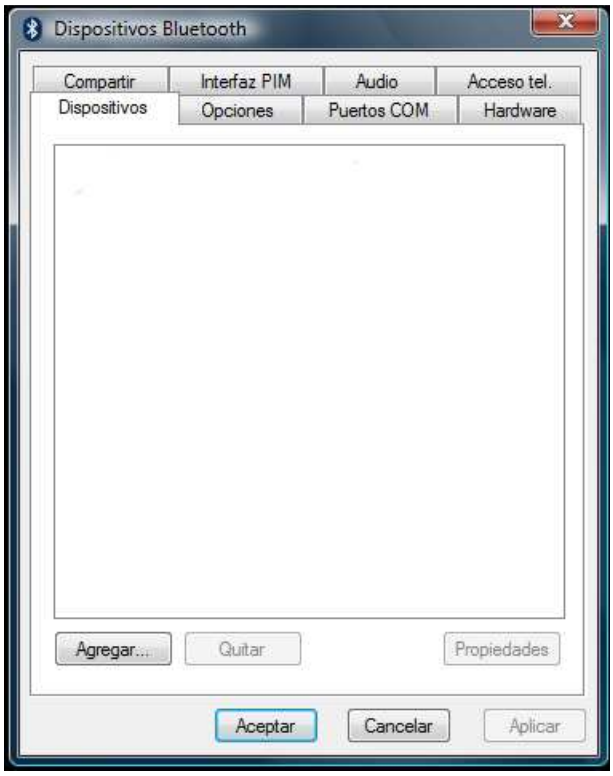


Figura 83. Imagen ventana Dispositivos Bluetooth de Windows

Pulsamos al botón Agregar y simultáneamente los botones 1 y 2 del wiimote a la vez

Figura 84. Imagen ventana Dispositivos Bluetooth de Windows

Marcamos la opción Mi dispositivo está configurado y listo para ser detectado y pulsamos siguiente, continuamos con los botones 1 y 2 del mando pulsados simultáneamente mientras la aplicación busca dispositivos bluetooth una vez que encuentre el wiimote veremos la siguiente ventana

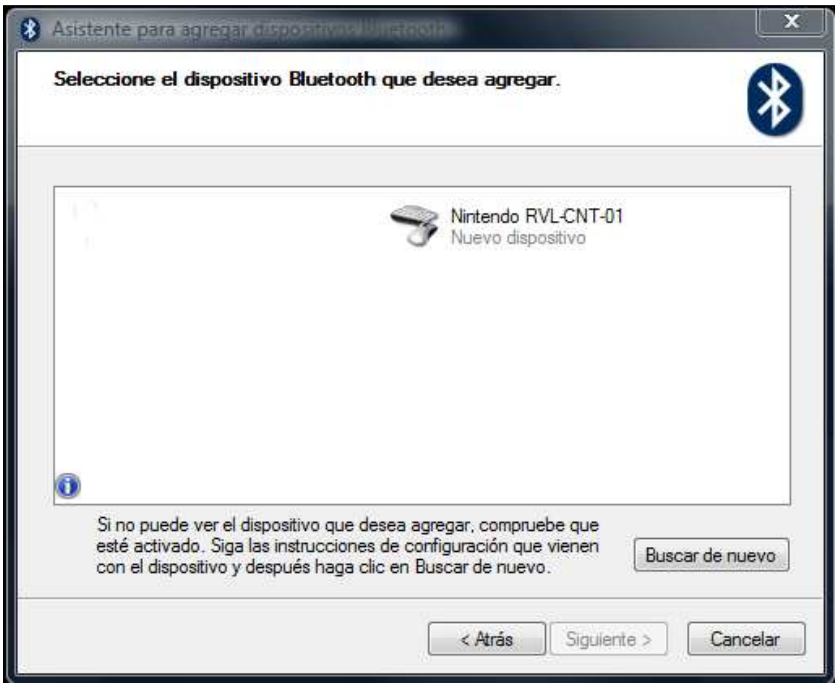


Figura 85. Imagen ventana Dispositivos Bluetooth de Windows

Seleccionamos Nintendo RVL-CNT-01 y pulsamos Siguiente en este paso continuamos con los botones 1 y 2 del mando pulsados

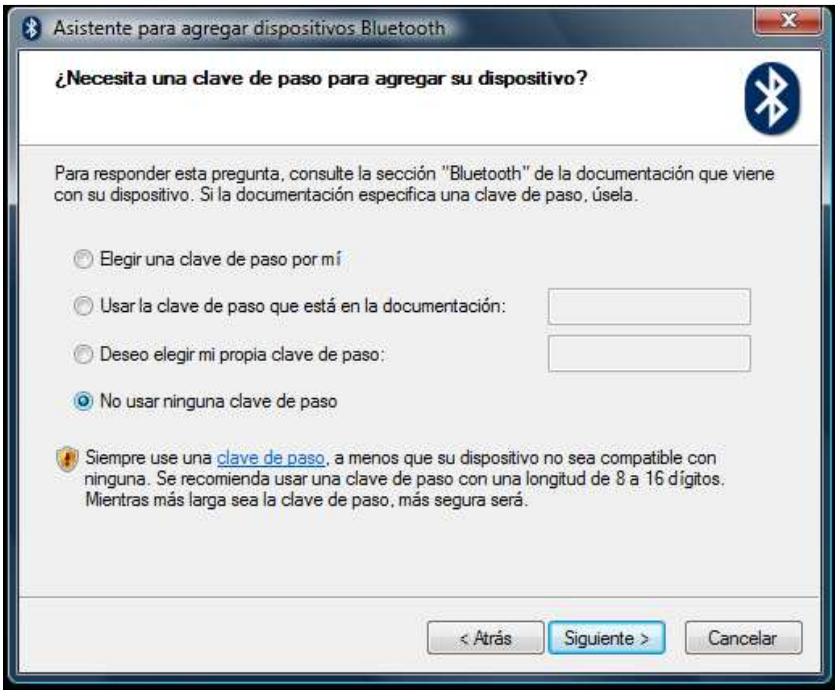


Figura 86. Imagen ventana Dispositivos Bluetooth de Windows

Seleccionamos la opción no usar ninguna clave de paso y pulsamos Siguiente

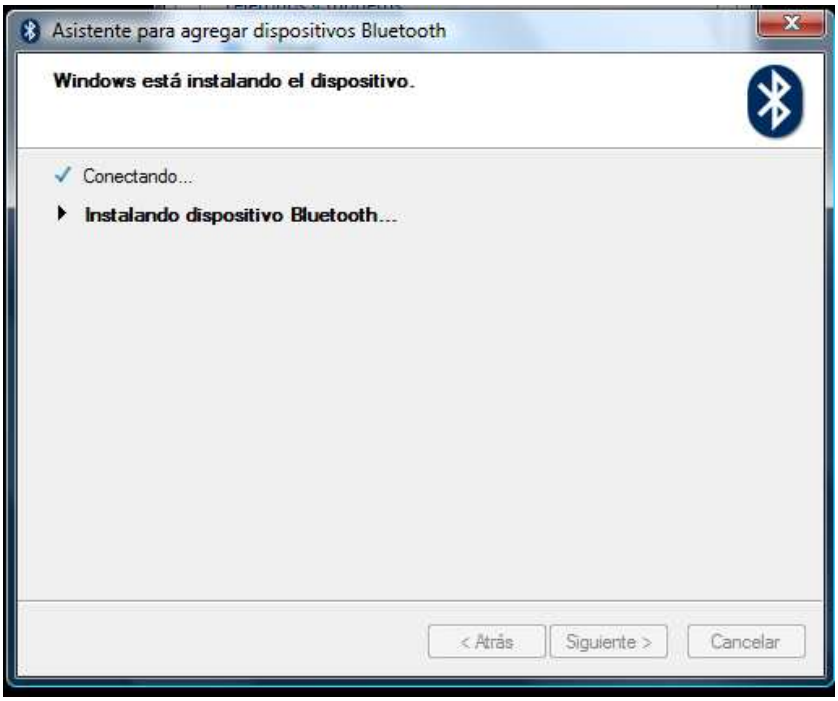


Figura 87. Imagen ventana Dispositivos Bluetooth de Windows

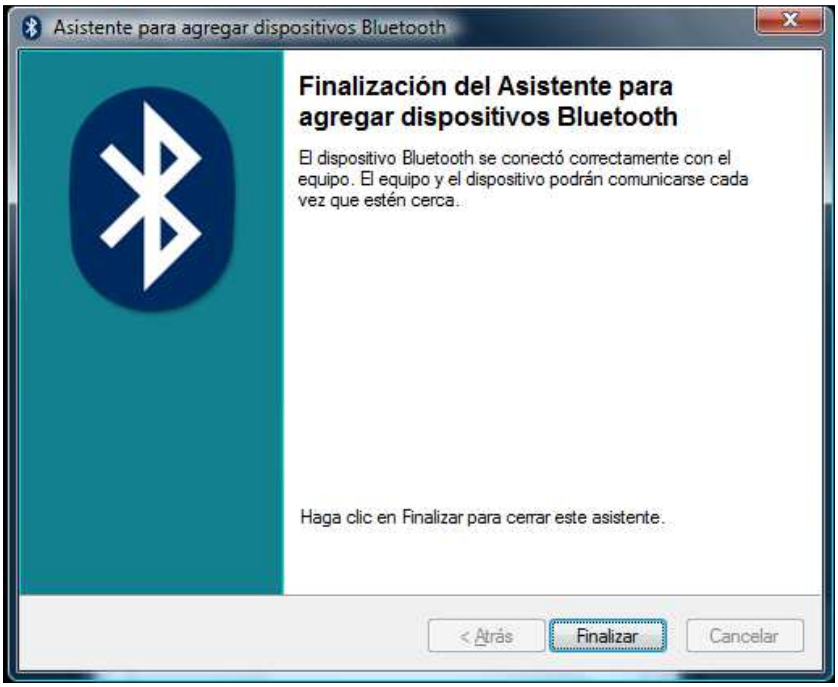


Figura 88. Imagen ventana Dispositivos Bluetooth de Windows

Es en este punto donde podemos dejar de pulsar los botones 1 y 2 del wiimote , si todo ha ido correcto veremos la siguiente ventana con el Nintendo RVL-CNT-01 conectado.

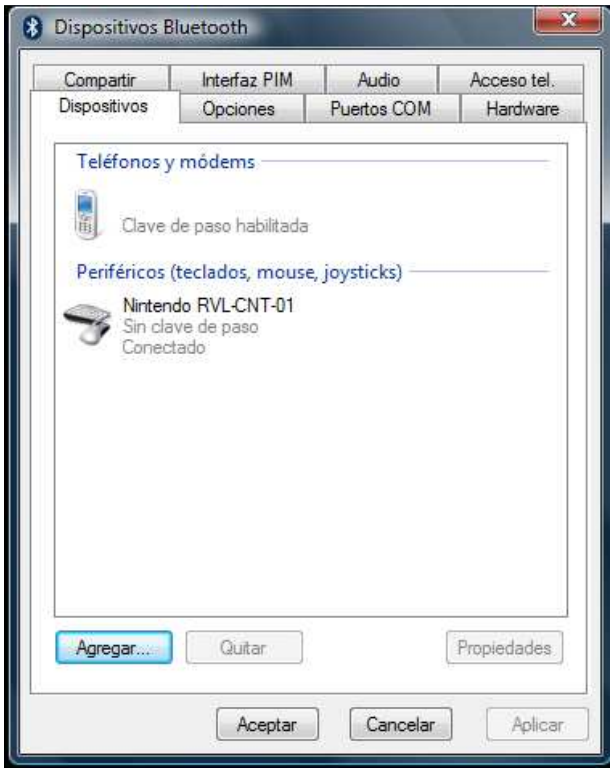


Figura 89. Imagen ventana Dispositivos Bluetooth de Windows con un wiimote conectado correctamente.

3. Inicio de la aplicación

Una vez conectado el wiimote al PC vía bluetooth ya podemos iniciar la aplicación. Pulsamos el siguiente icono para iniciar la aplicación

Figura 90, logo principal de la aplicación.

Aparecerá la siguiente ventana si tenemos conectado un solo mando wiimote

Aplicación para la detección de caídas.

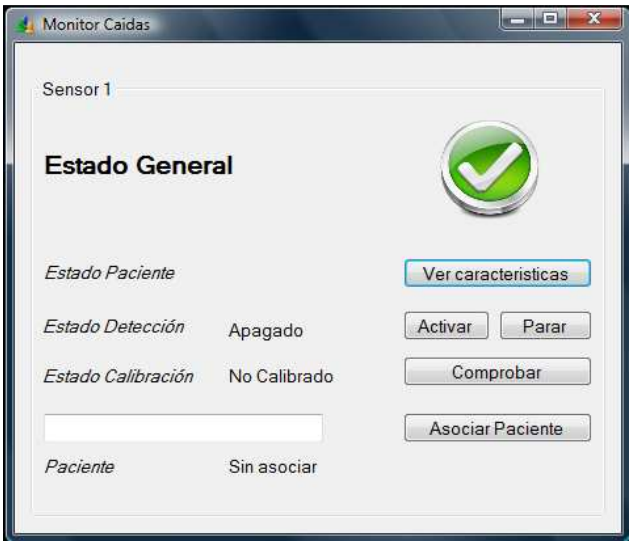


Figura 91 Aplicación para la detección de caídas con un wiimote conectado.

Mientras que si tenemos conectados 2 wiimotes aparecerá la misma ventana pero con el siguiente aspecto.

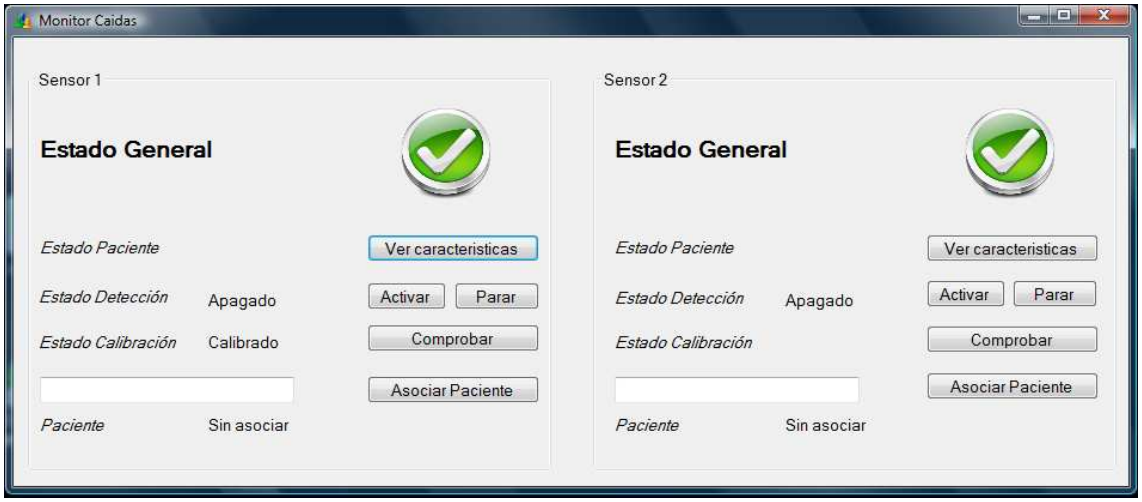


Figura 92 Aplicación para la detección de caídas con 2 wiimote conectados.

El estado inicial en el que se encuentra la aplicación es con la detección de caídas apagada y el mando no se encuentra asociado a ningún paciente. En este estado es cuando debemos colocar en el tronco del paciente el mando, debe quedar de la siguiente forma colocado:

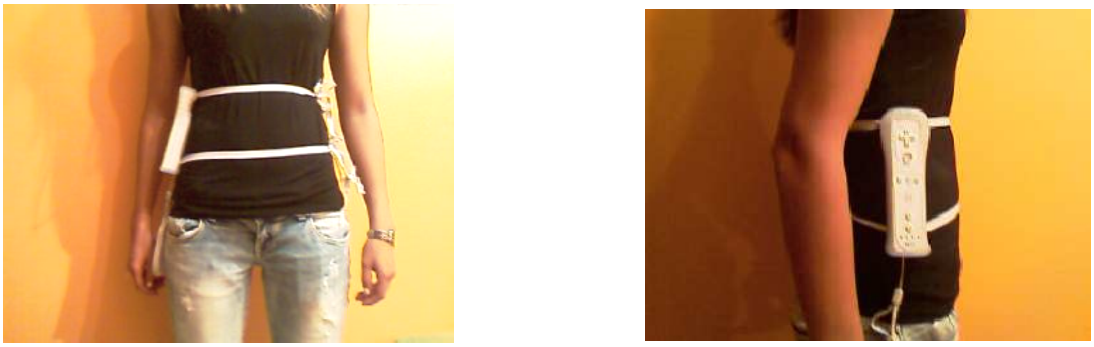


Figura 93. Forma correcta de colocar el mando wiimote

El mando wiimote debe estar colocado en la parte derecha de la cintura como muestran las fotografías, es importante que sea el lado derecho para el correcto funcionamiento de la aplicación.

4. Asignar un paciente al wiimote.

Una vez colocado el/los mandos a los pacientes vamos a darles un nombre para poder diferenciar a que paciente se le ha colocado cada mando, para ello introducimos un nombre en el cuadro de texto del wiimote que queramos asociar y pulsamos el botón asociar paciente.

La forma de diferenciar el sensor 1 del sensor 2 será que en el wiimote 1 tendrá encendido el led número 1, mientras que el wiimote 2 tendrá encendido el led número 2.

Aplicación para la detección de caídas.

Figura 94. Imagen ventana principal aplicación detección de caídas con wiimote.

5. Calibración del wiimote.

Una vez asociado el paciente podemos iniciar la detección pulsamos el botón Iniciar detección para el wiimote que queramos.

Al pulsar este botón la aplicación buscará en una base de datos los valores para la calibración del mando. Si este mando ha sido utilizado antes, este se calibrará solo, si no es así veremos las siguientes ventanas y tendremos que seguir los pasos que se indican en cada una de ellas:

Figura 95. Imagen primera ventana calibración wiimote

Figura 96. Imagen segunda ventana calibración wiimote

Figura 97. Imagen primera ventana calibración wiimote

Una vez asociado el nombre del paciente y calibrado el wiimote la pantalla principal tendrá el siguiente aspecto:

Aplicación para la detección de caídas.

Figura 98.

Como podemos ver las ahora la detección está encendida, el mando Calibrado y el nombre del paciente es María.

6. Ver la información detallada de los sensores.

Si pulsamos ahora el botón Ver características nos encontramos con la siguiente información:

Aplicación para la detección de caídas.

Figura 99. Imagen ventana controlador wiimote 1 con la detección de caídas encendida.

Esta es la información avanzada del estado del wiimote, a través de esta ventana podemos conocer la orientación actual del wiimote además de:

- El paciente al que está asociado.
- El valor de las aceleraciones leídas.
- El estado de la batería en porcentaje.
- Si se está aplicando la detección de caídas.
- Si se ha producido alguna caída o alarma de emergencia.

7. Realizar una llamada de emergencia

Si el paciente desea realizar una llamada de emergencia debe pulsar el botón con forma de cruz del wiimote:

Aplicación para la detección de caídas.

Figura 100, vista perspectivas de un mando wiimote.

Si se pulsa este botón en la aplicación tendremos los siguientes cambios:

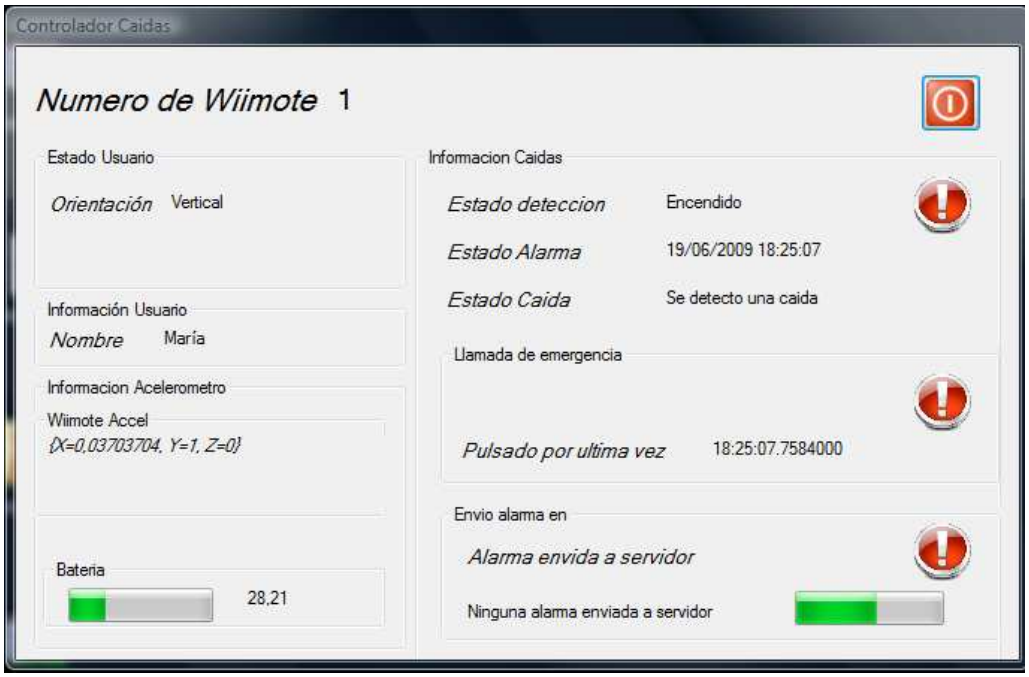


Figura 101. Imagen controlador caídas cuando se ha pulsado el valor enviar alarma servidor.

8. Desactivar una alarma.

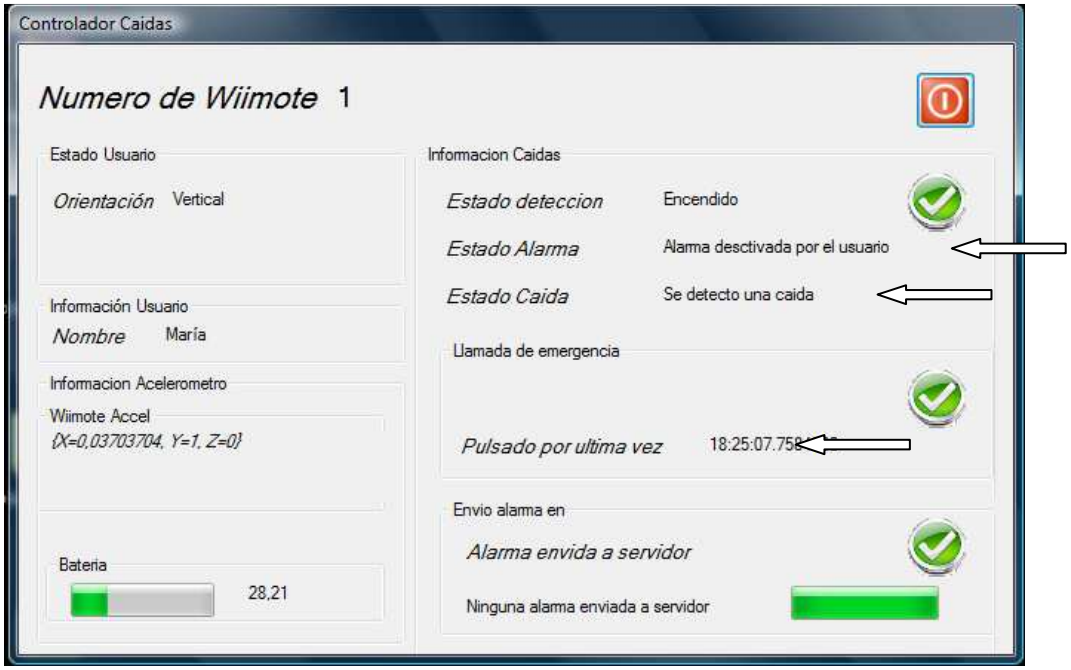
Aplicación para la detección de caídas.

Una vez que se produce una alarma ya sea una caída o una llamada de emergencia se disponen de 10 segundos para desactivarla antes de notificarla, para desactivar una falsa alarma se debe pulsar el botón A de wiimote.

Figura 102. Imagen mando wiimote.

Cuando se activa una alarma el wiimote vibra durante 2 segundos y sus 4 leds empiezan a parpadear durante los 10 segundos que tiene el paciente para desactivarla.

La aplicación nos queda de esta manera si se desactiva una alarma



Aplicación para la detección de caídas.

Figura 103. Imagen aplicación cuando el paciente ha desactivado una alarma.

9. Notificación de una alarma

Cuando la aplicación detecta una alarma ya sea una caída o una llamada de emergencia y han pasado los 10 segundos de los cual dispone el paciente para desactivarla, se producen los siguientes cambios en nuestra aplicación

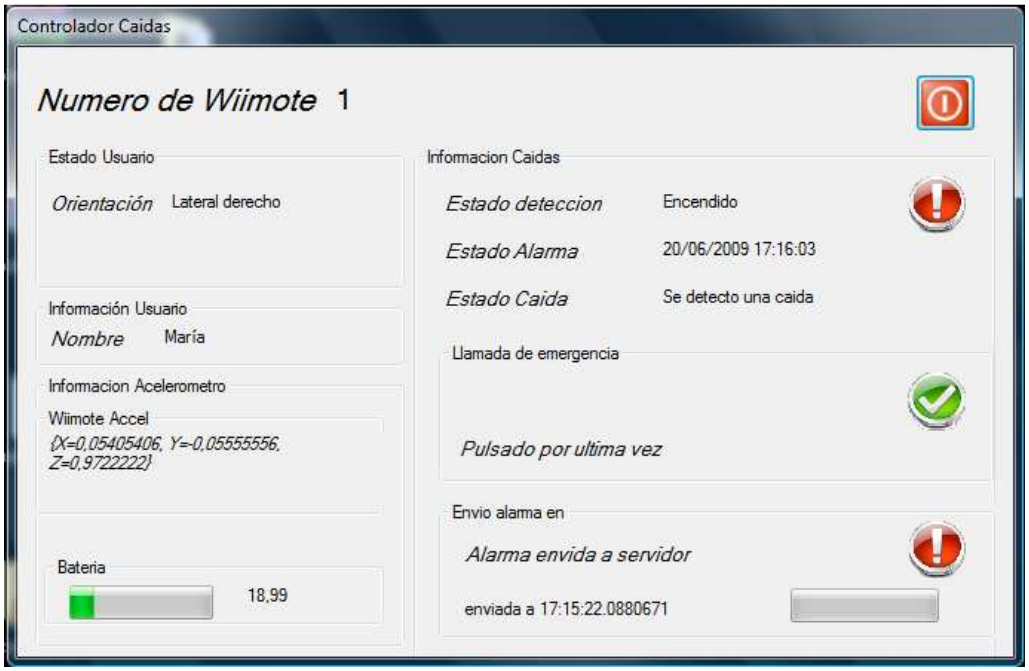


Figura 104. Imagen ventana controlador caídas para el wiimote 1.

Aplicación para la detección de caídas.

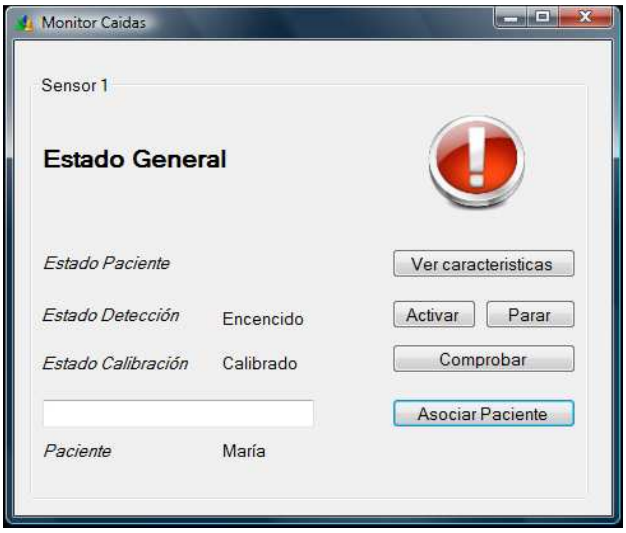


Figura 105. Imagen ventana principal del programa después de que una alarma haya sido notificada.

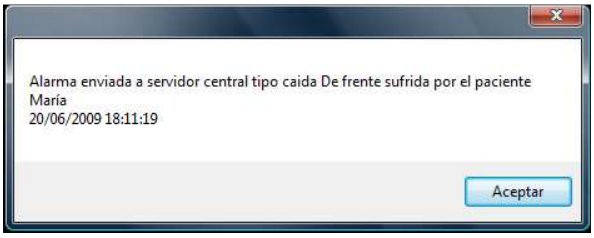


Figura 106, mensaje de advertencia de caída en la aplicación.

Aplicación para la detección de caídas.

Conclusiones

1. Objetivos alcanzados

Con la finalización de este proyecto hemos cumplido los objetivos que nos planteábamos al inicio de esta memoria, en particular:

- Desarrollar un algoritmo de detección de caídas lo más fiable y robusto posible que minimice los falsos positivos y sea capaz de detectar una amplia gama de caídas.
- Definir las características técnicas del sensor elegido para poder implementar el algoritmo desarrollado.
- Implementar dicho algoritmo en una aplicación informática y comprobar su funcionamiento a través de caídas simuladas y movimientos bruscos.
- Implementar el algoritmo un modulo basado en un dispositivo móvil con Windows Mobile para integrarlo en la línea de productos de monitorización de pacientes desarrollada por el grupo de investigación GIRO.

Los resultados obtenidos han sido los siguientes:

1. Hemos desarrollado y documentado un método de detección de caídas utilizando un acelerómetro triaxial. Partiendo de métodos y estudios publicados y mejorando los resultados de forma muy satisfactoria.

2. Las características técnicas que debe tener el dispositivo sobre el que se puede aplicar este algoritmo de forma práctica en situaciones reales son las siguientes:
   - Acelerómetro triaxial con una sensibilidad de al menos 3G.
   - Debe ser inalámbrico
   - Poco pesado y de pequeño tamaño.

3. Hemos programado una aplicación sobre PC utilizando como acelerómetro un dispositivo comercial de bajo coste (un wiimote de Nintendo) con el que hemos podido comprobar el funcionamiento del algoritmo.

4. De cara a la integración en la línea de productos genérica de monitorización móvil, se ha programado este algoritmo utilizando los acelerómetros presentes en dos tipos de dispositivos móviles con SO Windows Mobile. Con esto hemos comprobado que podemos implementar este algoritmo con acelerómetros diferentes al wiimote. Sin embargo los resultados no son tan fiables debido al menor rango de aceleraciones medibles. [El apéndice B presenta algunos detalles de estas experiencias]. En cuanto a los objetivos secundarios, nos queda por dar un paso más, probando el dispositivo en pacientes reales.

2. Conocimientos Adquiridos

Con el desarrollo de este proyecto se han adquirido gran cantidad de conocimientos de todo tipo y ha sido posible aplicar multitud de técnicas, estudiadas a lo largo de la carrera, al desarrollo de un sistema real.

En cuanto a conocimientos técnicos, el más importante es el aprendizaje del *framework* .NET y de su entorno de desarrollo. No menos importante ha sido el trabajo de familiarizarse con el funcionamiento del mando wiimote y comprender todo el partido que se le puede sacar a un dispositivo tan revolucionario.

Completamente diferentes pero muy interesantes han sido los conocimientos que se han adquirido gracias a las pruebas y a los estudios de las aceleraciones presentes en el cuerpo humano. Además , se ha comprobado lo útil que puede llegar a ser la Informática para facilitar la autonomía de personas dependientes, lo que nos lleva a animar a otros estudiantes a seguir con este tipo de proyectos que, aunque más complicados de sacar adelante, una vez terminados conllevan una amplia satisfacción personal y profesional.

3. Líneas de trabajo futuras.

Las líneas futuras de trabajo de este proyecto son muy amplias, pero centrándonos solo en aquellas que podrían desarrollarse de forma más inmediata, proponemos:

- Probar el dispositivo en un entorno real
- Eliminar todos los componentes que no hagan falta en el mando wiimote para poder conseguir un dispositivo más pequeño que pueda ser integrado en la ropa. Con ello conseguiríamos eliminar la molestia que causa llevar puesto el wiimote con su tamaño y su forma actuales.
- Añadir la funcionalidad de cuenta pasos o cuenta kilómetros ya que con la tecnología que dispone el wiimote sería bastante fácil de implementar.
- Implementar el algoritmo de detección de caídas dispositivos móviles con sistema operativo Symbian, ya que este sistema operativo es capaz de conectarse directamente con el wiimote sin tener que utilizar otro tipo de acelerómetro. Además, permitiría enviar fácilmente un SMS o llamada perdida a un móvil cuando se detectara una caída.

Bibliografìa.

Referencias citadas en el proyecto.

[1] Bourke A.K., O'Brien J.V., Lyons G.M. Biomedical Electronics Laboratory, Department of Electronic, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm".

[2]Centro para el Desarrollo de las Telecomunicaciones de Castilla y León http://www.cedetel.es/

[3] Chen Jay, Kwong Karric, Chang, Jerry Dennis Luk, Bajcsy Ruzena, Department of Electrical Engineeringand Computer Sciences, University of California, Berkeley, CA 94720

[4] Degen T., Jaeckel H., Rufer M., and. Wyss S,"SPEEDY: a fall detector in a wrist watch," Proc. Seventh IEEE International Symposium on Wearable Computing, 2003, pp. 184-187, "Wearable Sensors for Reliable Fall Detection".

[5] Doughty K., Lewis R., and McIntosh A., "The design of a practical and reliable fall detector for community and institutional telecare," J. Telemed. Telecare 2000, 6 S150-4.

[6]Lord SR, Ward JA, Williams P, Anstey KJ. An epidemiological study of falls in older community-dwelling women: the Randwick falls and fractures study. Aust J Public Health 1993;17(3):240–5.

[7]O'Neill TW, Varlow J, Silman AJ, Reeve J, Reid DM, Todd C, et al. Age and sex influences on fall characteristics. Ann Rheum Dis 1994;53(11):773–5.

[8] Peek Brian, programación con wiimote. WiimoteLib http://www.codeplex.com/WiimoteLib

[9] Proyectos con wiimote de Johnny Lee http://johnnylee.net/projects/wii/

Otras fuentes bibliográficas consultadas:

Benoit Marchal, "XML con Ejemplos", Prentice Hall, 2001

Grady Booch, Ivar Jacobson, James Rumbaugh, "El Lenguaje Unificado de Mode Addison Wesley, 1999

Jacobson Ivar, Booch Grady, Rumbaugh James, "El Proceso Unificado de Desarrollo de Software", Addison Wesley, 1999

Larman Craig, "UML y patrones", Prentice Hall, 1999

Nevitt MC, Cummings SR. Type of fall and risk of hip and wrist fractures: the study of osteoporotic fractures. The study of osteoporotic fractures research Group. J Am Geriatr Soc 1993;41(11):1226–34.

Schmuller Joseph, "Aprendiendo UML en 24 Horas", Prentice Hall, 2001

Sitios web consultados

Programación para dispositivos con .NET Compact Framework
http://msdn.microsoft.com/es-es/library/btyhs18b(VS.80).aspx

Sitio oficial Nintendo Wii http://www.nintendowii.es/

Visual estudio y Lenguaje C# http://msdn.microsoft.com/es-es/vcsharp/default.aspx

.NET Compact Framework y Smart Device Extensions
http://www.elguille.info/NET/netCF/netCF.htm

Apéndice A: Calibración del Wiimote.

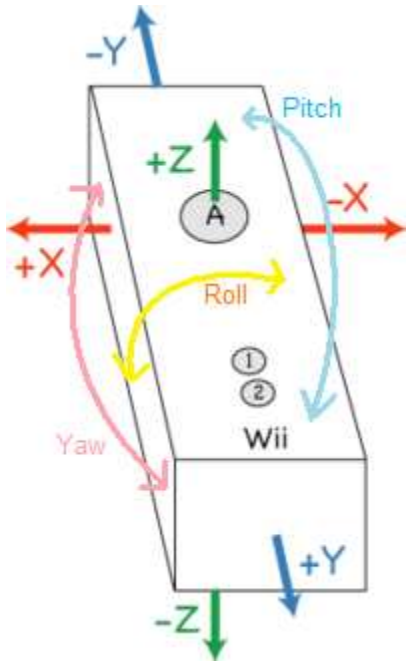


Figura A.1 Mando wiimote con ejes cartesianos.

El mando Wiimote contiene los controladores de acelerómetros de 3 ejes (ADXL330) que informa, en unidades arbitrarias, la aceleración instantánea en el controlador impartido por el jugador que posea (o cualquier superficie que lo apoyan).En reposo, esta aceleración es g, pero en la dirección ascendente. En caída libre, el controlador informa aproximadamente aceleración cero.

Sin embargo, cuenta con 6 grados de libertad: 3 lineal traducción direcciones (X, Y, Z) y 3 ángulos de rotación (pitch, roll, yaw), como se muestra en la figura.

Una vez accedidos a las aceleraciones arbitrarias que leemos del wiimote y asumiendo que la respuesta de los acelerómetros es aproximadamente lineal podemos tomar las posiciones más sencillas para calibrar el mando, estas tres posiciones han sido:

- (x1, y1, z1): En posición horizontal sobre una superficie plana, con el gatillo B en contacto con dicha superficie y la cara con la cruceta digital y el resto de botones hacia arriba.
- (x2, y2, z2): En posición vertical sobre una superficie plana, con el sensor de infrarrojos en contacto con dicha superficie y el puerto de expansión a la vista
- (x3, y3, z3) En posición horizontal sobre una superficie plana con el lado derecho del mando en contacto con dicha superficie.

Debido a que los acelerómetros registran la fuerza de la gravedad, los datos que estaremos recibiendo del mando en esas tres posiciones deben corresponderse con tres vectores de aceleración ortogonales en R3. Lo deseable desde un punto de vista matemático es contar con una base ortonormal con respecto a g para trabajar, de forma que aprovecharemos que en

cada una de las tres posiciones arriba indicadas dos de las tres componentes de cada vector deberían ser nulas en la base ortonormal a la que queremos llegar. Esto nos permite encontrar la correspondencia entre dicho valor nulo y los datos tomados anteriormente:

$x0 = (x1 + x2)/2$

$y0 = (y1 + y3)/2$

$z0 = (z2 + z3)/2$

Comenzando con cualquier dato leído de forma arbitraria (RawValueX, RawValueY, RawValueZ) la forma de convertir estos valores a vector ortogonal con respecto a g es la siguiente.

$X= RawValueX –x0 / x3-x0$

$Y= RawValueY -y0 / y2-y0$

$Z = RawValueZ – z0 / z1-z0$

Con esta transformación de los datos brutos del wiimote conseguimos que en la posición 1 descrita anteriormente se obtengan los valores (1,0,0) en la segunda posición (0,1,0) y en la tercera (0,0,1).

Aplicación para la detección de caídas.

Apéndice B: Conversión del proyecto de detección de caídas en una línea de producto.

1. Introducción.

Este capítulo está dedicado a la creación de un modulo específico de detección de caídas integrado con otro proyecto de monitorización de pacientes que tenemos disponible.

La línea de producto de la que partimos incluye la localización remota de pacientes por medio de GPS, y la monitorización del pulso y saturación de oxigeno en sangre por parte de un sensor pulsi-oxímetro.

2. Finalidad de la línea de producto monitorización de pacientes.

Con la elaboración de esta línea de productos queremos facilitar la creación de nuevos proyectos de monitorización de pacientes sobre PDAs. Si alguna vez se necesita una nueva funcionalidad y se dispone del sensor correspondiente poder añadir nuevos paquetes a este proyecto y que este continúe creciendo.

Además de las posibles nuevas incorporaciones de mas sensores, la línea de producto facilita la capacidad de elegir qué tipo de monitorización queremos para cada paciente, por ejemplo puede que existan pacientes para los que sea necesario la detección de caídas y no sea necesario conocer el nivel de saturación de oxigeno en sangre, o que se disponga de un dispositivo móvil sin GPS y no se pueda añadir la funcionalidad de localización.

De este modo dejamos abierto un abanico de posibilidades para la futura incorporación tanto de nuevos dispositivos móviles como de otros posibles sensores.

3. Características del proyecto anterior.

En Enero de 2009 se presento un proyecto en la Universidad de Valladolid para la detección remota de pacientes utilizando PDAs. Este proyecto estaba diseñado para la monitorización simultánea de un paciente con información procedente de varios sensores diferentes, esta idea viene bien esquematizada en la siguiente figura.

Aplicación para la detección de caídas.

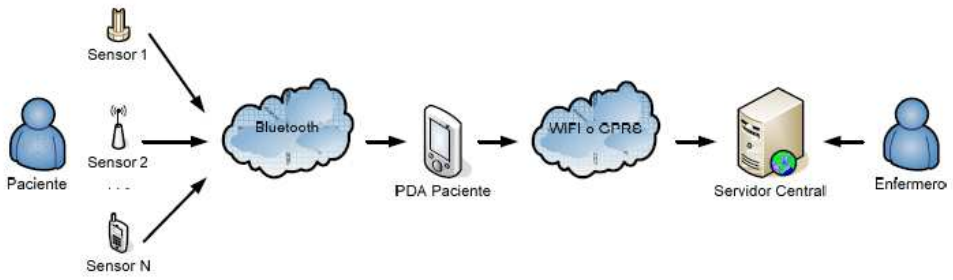


Figura B.1.

En la aplicación original se disponía de un solo sensor, este sensor que era capaz de medir el pulso y la saturación de oxigeno en sangre a través de un pulsioxìmetro bluetooth situado en un dedo del paciente, esta información era procesada en una PDA y enviada a un servidor Central el cual informa al personal médico del estado de los pacientes.

Como se puede observar este proyecto es muy similar a la detección de caídas mediante wiimote con una diferencia esencial, la detección mediante wiimote es procesada en PC mientras que esta línea de producto es procesada en PDA.

3.1. Tecnología disponible en la línea de producto monitorización de pacientes.

Para integrar la detección de caídas en la línea de producto debemos adaptarnos a la tecnología con la que fue creado el primer proyecto, esta tecnología es la siguiente.

Para el proyecto se utilizo una PDA con sistema operativo Windows Mobile 6.0, esta PDA dispone de tecnología GPS, Bluetooth y Wifi.

El proyecto fue programado con Visual Studio .net y Compack framework 2.0.

3.2. Problemas con el bluetooth del mando wiimote y PDA con Windows Mobile.

A la hora de integrar el proyecto el camino más fácil hubiese sido tratar al wiimote como un sensor bluetooth igual que se trataba a el pulsioximetro, el cual se conectaría a la PDA y esta realizaría el algoritmo de detección de caídas con los datos leídos, para posteriormente enviar las posible alarmas al servidor central.

Todo esto hubiese sido posible si la comunicación entre PDA y Wiimote mediante bluetooth hubiese sido posible, pero después de varias semanas intentando con varias pilas bluetooth del mercado no conseguimos leer los datos del wiimote desde una PDA.

Aplicación para la detección de caídas.

El otro gran problema que presentaba esta forma de ver el proyecto era la duración de la batería, si tenemos cualquier móvil durante varias horas leyendo y procesando datos de un dispositivo bluetooth no tardará en acabársenos la batería.

3.3. Solución propuesta para la integración de caídas en la línea de productos

Después de desistir en la conexión de PDA-Wiimote se pensó otra solución factible: existen numerosos móviles y PDA en el mercado las cuales tienen integrada un acelerómetro triaxial, ¿porque no utilizar el propio acelerómetro del móvil para aplicar el algoritmo de detección de caídas?.

El siguiente paso era buscar un dispositivo móvil que además de las características necesarias para el proyecto anterior tuviese también un acelerómetro triaxial.

Los dispositivos móviles con Windows mobile, bluetooth , wifii y acelerómetro triaxial que hemos probado para este proyecto son una Samsumg Onmia, y una HTC diamond. Se puede observar en la siguiente imagen.

Figura B.2. Samsug onmia i900

Figura B.3. HTC diamond

4. Descripción de la línea de producto

Una vez descrita la tecnología disponible vamos a explicar en qué paquetes se encuentra distribuida la línea de productos monitorización de pacientes:

4.1. Diagrama de características

Aplicación para la detección de caídas.

La forma más fácil y esquemática de explicar los módulos procedentes en una línea de productos es observando el diagrama de características, este diagrama es el siguiente:

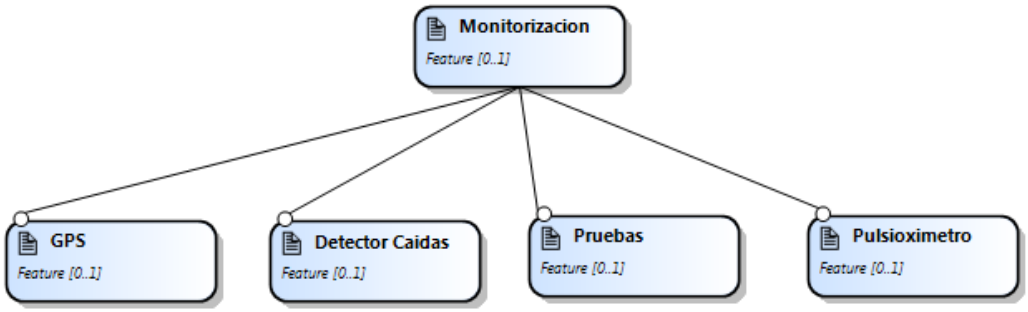


Figura B.4.

Este diagrama de características se traduce en un diagrama de paquetes, en el que cada paquete representa una característica de la línea de producto, como se puede observar en la siguiente figura:

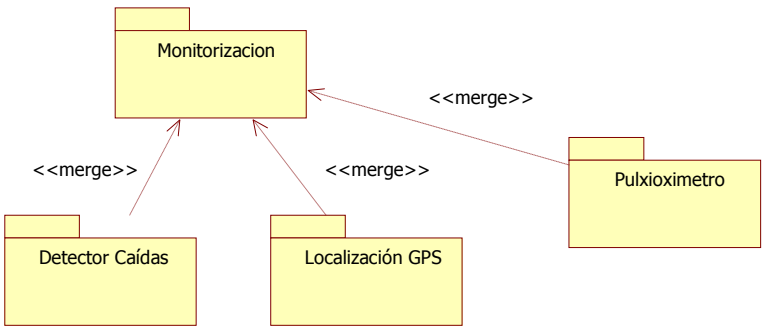


Figura B.5.

4.2. Características de los paquetes

Vamos a dedicar unas líneas a las características de cada paquete por separado

24.2.1 Paquete monitorización.

Aplicación para la detección de caídas.

El paquete monitorización debe estar presente en todas los proyectos de la línea de producto, en el se encuentra la funcionalidad mínima por lo que siempre va a estar presente.

24.2.2 Paquete localización GPS

Este paquete contiene la funcionalidad de conocer la posición que ocupa la PDA en todo momento, para que cuando se produzca una alarma poder conocer donde se encuentra el paciente. Si tuviéramos una PDA sin GPS integrado eliminaríamos este paquete de la línea de producto ya que solo está disponible para móviles con GPS integrado.

24.2.3 Paquete pulsioximetro.

Este paquete está asociado al sensor pulsioximetro del proyecto anterior de monitorización de pacientes. Si tenemos conectado el sensor pulsioximetro incluiremos este paquete en la línea de producto y si no es así le eliminaremos.

24.2.4 Paquete detector de caídas.

Es el paquete que realiza toda la funcionalidad de la detección de caídas, implementa los requisitos funcionales y no funcionales del proyecto detector de caídas con wiimote, aunque en este caso el wiimote no está presente y los datos de las aceleraciones se obtienen directamente del acelerómetro del móvil.

5. Conclusiones paquete detector de caídas en dispositivo móvil.

Para la integración de la detección de caídas en la línea de productos obtuvimos las aceleraciones como se ha explicado antes a partir de los acelerómetros integrados en los móviles. Estos acelerómetros presentan un grave problema ya que no son tan sensibles como lo es el mando wiimote basta con decir que el mando wiimote tiene una sensibilidad de [-3,3] g o en escala positiva de [0-7]g, mientras que la sensibilidad tanto del móvil Sansung Omnia, de la HTC Diamond así como otros móviles estudiados es de solo [-1,1]g o de [0-3]g en escala positiva.

Esto se traduce en que aunque la integración de la detección de caídas en una PDA ha sido sencilla una vez conocido el algoritmo no es tan útil, ya que el acelerómetro presente con tan poca precisión no se le puede aplicar el umbral de aceleración 3,56g que obtuvimos en el apartado anterior.

La solución que se ha propuesto es cambiar el umbral de 3,56g por el valor máximo de la aceleración que puede detectar este acelerómetro que es 3g.

El problema que conlleva esta modificación es que la aplicación móvil tendrá más falsos positivos que la aplicación con wiimote ya que el umbral de caída se superará con mayor frecuencia.

Por tanto queda por investigar y buscar un acelerómetro triaxial que pueda ser acoplado a una PDA y realizar de este modo el algoritmo de detección de caídas de forma fiable, como sucede con el wiimote.

Apéndice C. Microsoft .Net y Net Compact Framework

1. Introducción.

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft desarrolla una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado. .NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems. Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y bibliotecas de clases adicionales); siendo algunas de ellas software libre, distribuibles ciertas bajo la licencia GPL.

Con esta plataforma Microsoft incursiona de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas.

.NET intenta ofrecer una manera rápida y económica pero a la vez segura y robusta de desarrollar aplicaciones - o como la misma plataforma las denomina, soluciones - permitiendo a su vez una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

Aplicación para la detección de caídas.

Figura C.1. Resumen Arquitectura .Net

2. .NET Framework

El framework o marco de trabajo, constituye la base de la plataforma .NET y denota la

infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que

simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido. Aplicación para la monitorización remota de pacientes.

Bajo el nombre .NET Framework se encuentran reunidas una serie de normas impulsadas por varias compañías además de Microsoft (como Hewlett-Packard , Intel, IBM, Fujitsu Software, Plum Hall, la Universidad de Monash e ISE), entre las cuales se encuentran:

La norma que define las reglas que debe seguir un lenguaje de programación para ser considerado compatible con el marco de trabajo .NET (ECMA-335, ISO/IEC 23271). Por medio de esta norma se garantiza que todos los lenguajes desarrollados para la plataforma ofrezcan al programador un conjunto mínimo de funcionalidad, y compatibilidad con todos los demás lenguajes de la plataforma.

La norma que define el lenguaje C# (ECMA-334, ISO/IEC 23270). Este es el lenguaje insignia del marco de trabajo .NET, y pretende reunir las ventajas de lenguajes como C/C++ y Visual Basic en un solo lenguaje.

La norma que define el conjunto de funciones que debe implementar la librería de clases base

(BCL por sus siglas en inglés) (incluido en ECMA-335, ISO/IEC 23271). Tal vez el más importante de los componentes de la plataforma, esta norma define un conjunto funcional mínimo

que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo.

Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente

o futuro, permitiendo que las aplicaciones corran sobre la plataforma independientemente del sistema operativo para el cual haya sido implementada. El Proyecto Mono emprendido por Ximian pretende realizar la implementación de la norma para varios sistemas operativos adicionales bajo el marco del software libre o código abierto.

Aplicación para la detección de caídas.

Los principales componentes de .NET Framework son:

- El conjunto de lenguajes de programación
- La Biblioteca de Clases Base o BCL
- El Entorno Común de Ejecución para Lenguajes o CLR

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje. Algunos de los lenguajes desarrollados para .NET Framework son: C#, Visual Basic, Delphi (Object Pascal), C++, J#, Perl, Python, Fortran y Cobol.NET.

3. Objetivos del .NET Framework

El .NET Framework fue diseñado para cumplir varios objetivos:

Interoperabilidad.

Un objetivo muy importante es proporcionar la posibilidad de interacción entre nuevas y viejas aplicaciones. Para ello, el .NET Framework proporciona mecanismos que permiten el acceso a funcionalidad que está implementada en programas que se ejecutan fuera del entorno de .NET.

Por ejemplo, el acceso a componentes COM mediante el EnterpriseServices.

- Motor común de ejecución.

Los programas escritos bajo .NET se compilan a un lenguaje intermedio conocido como Common Intermediate Lenguage o CIL. Microsoft proporciona su propia implementación de CIL, el conocido como Microsoft Intermediate Language, o MSIL. En la implementación de Microsoft este lenguaje intermedio no es interpretado, ya que se usa una técnica de compilación en tiempo de ejecución (JIT, just-in-time compilation) que traduce el programa a código nativo. La combinación de todos estos conceptos se denomina Common Language Infraestructure (CLI). La implementación de Microsoft del CLI se conoce como el Common Language Runtime (CLR).

- Independencia de lenguaje.

El .NET Framework emplea el Common Type System, o CTS.

La especificación del CTS define todos los posibles tipos de datos y estructuras de programación soportados por el CLR y cómo pueden o no pueden interactuar con cada

uno. Esta característica permite que el .NET Framework soporte el desarrollo de varios lenguajes de programación distintos.

- Base Class Library.

La Base Class Library (BCL) es una biblioteca de tipos disponibles para todos los lenguajes que usan el .NET Framework. La BLC proporciona clases que encapsulan varias funciones comunes, como funciones de lectura y escritura de ficheros, funciones gráficas, funciones de interacción con bases de datos y funciones de manipulación de documentos XML.

- Ayuda a la instalación.

La instalación de software debe ser cuidadosamente gestionado para asegurarse que no interfiere con otro software previamente instalado. El .NET Framework incluye características y herramientas que ayudan a lograr mantener la estabilidad.

- Seguridad.

.NET permite al código ser ejecutado en diferentes niveles de seguridad mediante el uso de “cajas de arena”.

Hay que remarcar que, a pesar que el objetivo principal de .NET Framework sea la independencia de plataforma, Microsoft únicamente ha implementado versiones completas del .NET Framework en sus propios sistemas operativos. Actualmente se está trabajando en implementaciones de .NET

Framework en otros sistemas operativos.

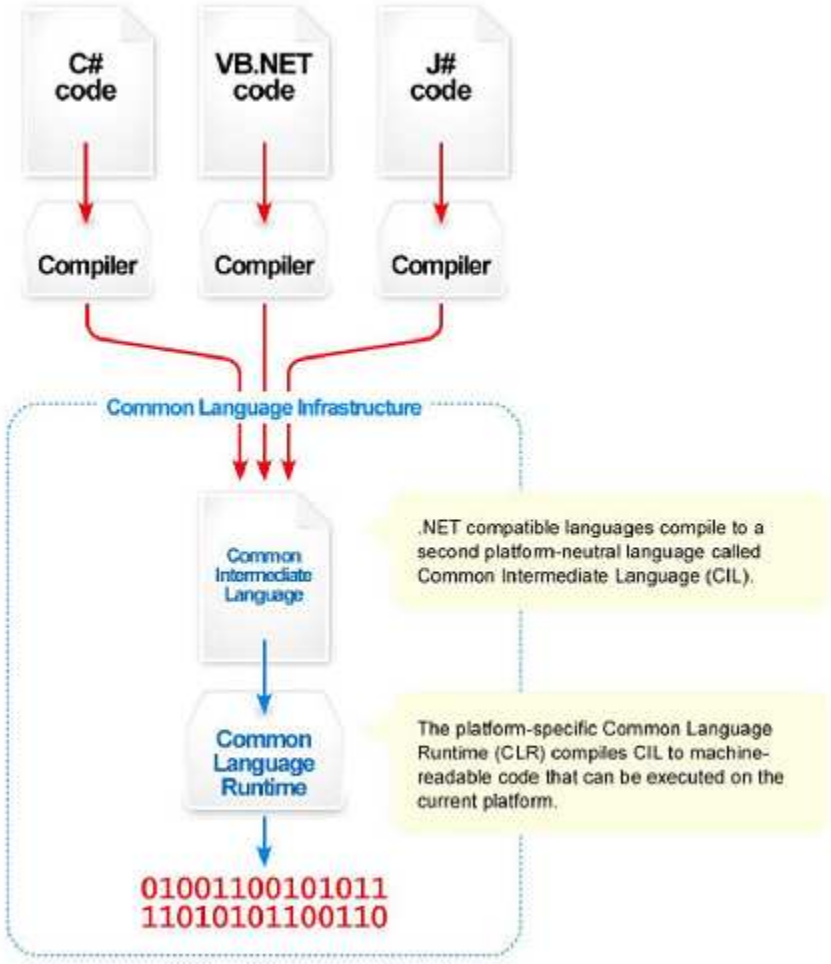


Figura C.2

4. Common Language Runtime (CLR)

El CLR es la implementación de Microsoft del CLI (Common Lenguage Infraestructure). El CLR es el verdadero núcleo del Framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo (W2k y W2003).

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio (MSIL, Microsoft Intermediate Lenguaje), similar al BYTECODE de Java. Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear ese código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso, un compilador JIT (Just-In-Time) es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .NET independencia de la plataforma hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos, el código ejecutable obtenido, se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

5. Base Class Library (BCL).

La Librería de Clase Base (BCL) es una librería incluida en el .NET Framework formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Además, a partir de estas clases prefabricadas el programador puede crear nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL. Por ejemplo, implementando ciertos interfazs podemos crear nuevos tipos de colecciones que serán tratadas exactamente igual que cualquiera de las colecciones incluidas en la BCL.

Esta librería está escrita en MSIL, por lo que puede usarse desde cualquier lenguaje cuyo compilador genere MSIL. A través de las clases suministradas en ella es posible desarrollar cualquier tipo de aplicación, desde las tradicionales aplicaciones de ventanas, consola o servicio de Windows NT hasta los novedosos servicios Web y páginas ASP.NET. Es tal la riqueza de servicios que ofrece que puede crearse lenguajes que carezcan de librería de clases propia y sólo usen la BCL, como C#.

Dado la amplitud de la BCL, ha sido necesario organizar las clases en ella incluida en espacios de nombres que agrupen clases con funcionalidades similares. Por ejemplo, los espacios de nombres más usados son:

Espacio de nombres	Utilidad de los tipos de datos que contiene
System	Tipos muy frecuentemente usados, como los tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.
System.Collections	Colecciones de datos de uso común como pilas, colas, listas, diccionarios, etc.
System. Data	Manipulación de bases de datos. Forman la denominada arquitectura ADO.NET.
System. IO	Manipulación de ficheros y otros flujos de datos.
System. Net	Realización de comunicaciones en red.
System. Reflection	Acceso a los metadatos que acompañan a los módulos de código.
System. Runtime.Remoting	Acceso a objetos remotos.
System.Security	Acceso a la política de seguridad en que se basa el CLR.
System.Threading	Manipulación de hilos.
System.Web.UI.WebControls	Creación de interfaces de usuario basadas en ventanas para aplicaciones Web.
System.Winforms	Creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.
System.XML	Acceso a datos en formato XML.

6. .NET Compact Framework

El .NET Compact Framework es una versión "reducida" del .NET Framework y se utiliza en los

Windows Mobile, o en los equipos que utilicen el Windows CE o el Windows CE .NET. Para utilizarlo, es necesario el Visual Studio .NET y el SDE (Smart Device Extensions), aunque a partir de Visual Studio 2005, el SDE se encuentra integrado en el ID; ya que el SDE es el que permite crear en VS .NET proyectos para los Windows Mobile, tanto para VB .NET como para C#.

.NET Compact Framework es un entorno independiente del hardware para ejecutar programas en dispositivos como asistentes digitales personales (PDA), teléfonos móviles y descodificadores. Se ejecuta en el sistema operativo Microsoft Windows CE y se basa en una reconstrucción de Common Language Runtime (CLR) diseñada para funcionar eficazmente cuando se ejecutan programas en dispositivos con limitaciones de recursos. .NET Compact Framework lleva el código administrado y los servicios Web XML a los dispositivos y proporciona ventajas como la seguridad de tipos, la recolección de elementos no utilizados, el control de excepciones y la seguridad. .NET Compact Framework es un

subconjunto de la biblioteca de clases .NET Framework y también contiene clases diseñadas expresamente para dispositivos con limitaciones de recursos. Hereda la arquitectura .NET Framework completa de Common Language Runtime y la ejecución de código administrado.

.NET Compact Framework hereda la arquitectura .NET Framework completa de Common Language Runtime para ejecutar código administrado. Proporciona interoperabilidad con el sistema operativo Windows CE de un dispositivo para tener acceso a funciones nativas e integrar los componentes nativos favoritos en una aplicación. Puede ejecutar aplicaciones nativas y administradas de manera simultánea. El host del dominio de aplicación, que también es una aplicación nativa, inicia una instancia del Common Language Runtime para ejecutar el código administrado.

.NET Compact Framework utiliza el sistema operativo Windows CE para la funcionalidad central y para diversas características específicas de dispositivos. Varios tipos y ensamblados, como los de los formularios Windows Forms, gráficos, dibujos y servicios Web, se han recompilado para que se ejecuten eficazmente en los dispositivos, en lugar de copiarse de .NET Framework completo.

En la ilustración siguiente se resume la arquitectura de la plataforma .NET Compact Framework.

Figura C.3.

Con respecto a los servicios web, hay que destacar que el .NET Compact Framework permite a las aplicaciones invocar métodos de servicios web, pero no crearlos ni alojarlos en el dispositivo portátil.

7. Desarrollo bajo .NET: Visual Studio.

Visual Studio 2005 es el entorno de programación con el que se ha desarrollado todo el proyecto. Visual Studio .NET es un IDE desarrollado por Microsoft a partir de 2002. Es para el sistema operativo Microsoft Windows y la última versión en funcionamiento del IDE, Visual Studio .NET soporta los nuevos lenguajes .NET: C#, Visual Basic .NET y Managed C++, además de C++. Visual Studio .NET puede utilizarse para construir aplicaciones dirigidas a Windows (utilizando Windows Forms), Web (usando ASP.NET y Servicios Web) y dispositivos portátiles(utilizando .NET Compact Framework).

La característica más notable del IDE es su soporte de los nuevos lenguajes .NET. Los programas desarrollados en esos lenguajes no se compilan a código máquina ejecutable (como por ejemplo hace C++) sino que son compilados a algo llamado CIL. Cuando los programas ejecutan la aplicación CIL, ésta es compilada en ese momento al código de máquina apropiado para la plataforma en la que se está ejecutando. Mediante este método, Microsoft espera poder soportar varias implementaciones de sus sistemas operativos Windows (como Windows CE). Los programas compilados a CIL pueden ejecutarse sólo en plataformas que tengan una implementación de .NET framework. Es posible ejecutar programas CIL en Linux o en Mac OS X utilizando algunas implementaciones .NET que no pertenecen a Microsoft, como Mono y DotGNU.

8. C#.

C# es el lenguaje de programación empleado para elaborar el cliente PDA del proyecto, se trata de un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (más notablemente de Delphi y Java). C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic.

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones; mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Aunque aún no existen, es posible implementar compiladores que no generen programas para dicha plataforma, sino para una plataforma diferente como Win32 o UNIX.

9. ASP.NET

ASP.NET es el lenguaje de programación empleado para elaborar el servidor web del proyecto. ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones web comercializado por Microsoft. Es usado por programadores para construir sitios web domésticos, aplicaciones web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la Tecnología Active Server Pages (ASP).

Aplicación para la detección de caídas.

Apéndice D contenido del CD-ROM

Aplicación para la detección de caídas.

1. Contenido CD-ROM

Los contenidos del CD-ROM que acompaña esta documentación son:

- Directorio de Instalación:
    - Instalación aplicación con wiimote

  Contiene el instalador y los requisitos previos para la instalación de la aplicación de detección de caídas.

    - Instalación versión móvil

  Contiene el instalador y los requisitos previos para la instalación de la aplicación de detección de caídas en la versión para móvil.

- Subdirectorio Códigos Fuentes

  Contiene el código fuente de la aplicación.

- Directorio Memoria: Contiene la memoria en formato PDF.
- Directorio Manual de Usuario: Contiene el manual de usuario en formato PDF.