



# **APLICACIÓN ANDROID DE ASISTENCIA AL CAMINANTE**

*(WALKER ASSISTENT ANDROID APPLICATION)*

**Desarrollo de aplicaciones de software libre**

Autor: **Enrique R. Delgado Garrido**

Tutor: **Gregorio Robles Martínez**

*9 de junio 2011*



## Licencia



### Creative Commons License Deed

#### Atribución-CompartirIgual 3.0 Unported (CC BY-SA 3.0)

Esto es un resumen fácilmente legible del Texto Legal ( <http://creativecommons.org/licenses/by-sa/3.0/legalcode> ).

Este resumen (Commons Deed) no es una licencia. Es simplemente una referencia práctica para entender el Texto Legal (la licencia completa) — es una redacción legible por humanos de algunos de los términos clave de la licencia. Tómelo como una interfaz amigable del Texto Legal que se encuentra más abajo. Este resumen por sí mismo no tiene valor legal, y su contenido no aparece en la auténtica licencia.

Creative Commons no es un bufete de abogados y no ofrece servicios legales. La distribución, la muestra, o el enlace de este Resumen no crea ningún vínculo abogado-cliente.

#### Usted es libre de:

- Compartir - copiar, distribuir, ejecutar y comunicar públicamente la obra
- hacer obras derivadas
- hacer un uso comercial de esta obra

#### Bajo las condiciones siguientes:

- **Atribución** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).
- **Compartir bajo la Misma Licencia** — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

#### Entendiendo que:

- **Renuncia** — Alguna de estas condiciones puede [no aplicarse](#) si se obtiene el permiso del titular de los derechos de autor
- **Dominio Público** — Cuando la obra o alguno de sus elementos se halle en el [dominio público](#) según la ley vigente aplicable, esta situación no quedará afectada por la licencia.
- **Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:
  - Los derechos derivados de [usos legítimos](#) u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
  - Los derechos [morales](#) del autor;
  - Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo [derechos de imagen](#) o de privacidad.



### agradecimientos.

- A mi tutor **Gregorio Robles Martínez**: comprensión, buenos consejos y ayuda, con un trato personal exquisito.
- A **Miguel Fernández del Olmo**, por haberme propuesto este proyecto, aguantado mis incursiones en su trabajo y en general a todo el personal del **Motor Control Group** de INEF Galicia en la Universidade Da Coruña (<http://www.motorcontrolgroup.com/>) que no han tenido inconveniente en ayudarme en todo lo que les he solicitado.
- A **Jorge Fernández González** de libresoft, que me ha ayudado a quitar los palos en las ruedas de mis primeros viajes por Android y por sus aclaraciones teóricas.
- A **IrisLibre** por permitirme utilizar su repositorio
- A los innumerables **internautas** que con sus aportaciones me han ayudado.
- A todos los **profesores** y **compañeros** de este Máster,

*es un placer.*

- A mi **Familia**, por soportarme en mi 'ausencia'.



## **Resumen del Proyecto.**

### **DESCRIPCIÓN:**

Desarrollo de una aplicación bajo software Android de apoyo a la marcha en pacientes con la enfermedad de Parkinson.

### **OPCIONES de la aplicación.**

Tres son las opciones que podremos elegir es esta aplicación y que compondrán su Menú Principal, respondiendo cada una de ellas a un determinado problema de movilidad. La opciones no son excluyentes pudiéndose activar una, varias o todas a la vez.

#### **a) Opción *marcha*:**

Proporciona de manera continua un estímulo sonoro rítmico a una frecuencia previamente establecida y que puede ser modificada.

- Justificación. Se ha comprobado en numerosos estudios que el uso de estimulación sonora rítmica mejora la amplitud del paso en pacientes con EP. Además, permite que la marcha se vuelva más estable (la mayor inestabilidad de la marcha se asocia a un mayor riesgo de caídas) y disminuye el número de episodios de bloqueo.

#### **b) Opción *anti-bloqueo*:**

La aplicación emite un sonido de gran intensidad una vez detecte ausencia de movimiento bajo ciertas condiciones.

- Justificación. Cuando escuchamos un sonido de gran intensidad (ej. una explosión) el sistema nervioso desencadena lo que se conoce como reflejo de sobresalto. Este reflejo de sobresalto provoca una contracción involuntaria de nuestra musculatura. Un reciente estudio ha comprobado que cuando queremos iniciar la marcha si se provoca el reflejo de sobresalto somos capaces de iniciarla más rápidamente. Aunque hasta el momento no se ha comprobado estos resultados en pacientes con EP si se sabe que estos presentan un comportamiento más o menos normal ante este reflejo.

#### **c) Opción *anti-caída*:**

La aplicación emite un sonido de gran volumen cuando los sensores del teléfono detectan variaciones, previamente identificadas y establecidas, que nos indican que se está produciendo una caída.

- Justificación. Dado que el reflejo de sobresalto mejora el tiempo de reacción, la lógica de la opción anti-caída es que el sonido acelerará la respuestas de reequilibración. Es decir, permitirá al paciente reaccionar más rápidamente ante una posible caída lo cual puede ser de crucial importancia para evitarla y en el peor de los casos minimizar las lesiones asociadas a las mismas. Esta hipótesis aún no ha sido investigada, por lo que hay que considerarla meramente especulativa, pudiendo esta aplicación ayudar a corroborarla.



## Tabla de contenidos

<b>Licencia</b> .....	<b>2</b>
Usted es libre de:.....	2
Bajo las condiciones siguientes:.....	2
Entendiendo que:.....	2
<b>Motivación</b> .....	<b>8</b>
¿Porqué este proyecto?.....	8
¿Porqué software libre?.....	8
¿Porqué Android?.....	8
<b>JUSTIFICACIÓN</b> .....	<b>9</b>
<b>1.- Android. Teoría general</b> .....	<b>10</b>
1.1.- Android:.....	10
1.2.- Arquitectura de Android.....	10
1.2.1.- Kernel de Linux.....	11
1.2.2.- Librerías.....	11
1.2.3.- Runtime de Android.....	11
1.2.4.- Armazón (framework) de Aplicaciones.....	11
1.2.5.- Aplicaciones.....	12
1.3.- Dalvik, la maquina virtual para Android.....	12
1.4.- Las APLICACIONES en Android.....	13
1.4.1.- Componentes fundamentales.....	14
Activity.....	14
Intent.....	14
Broadcast Intent Receiver.....	15
Service.....	15



## Aplicación Android de Asistencia al Caminante

Content Provider.....	15
1.4.2.- Procesos y sus estados.....	15
Primer plano.....	15
Visibles.....	15
Servicio.....	16
Segundo Plano.....	16
Vacíos.....	16
1.4.3.- Activity – Ciclo de Vida.....	16
1.4.4.- Service – Ciclo de Vida.....	18
<b>2.- Android. Desarrollo.....</b>	<b>19</b>
2.1.- Eclipse y SDK de Android.....	19
2.2.- Estructura de Android - Eclipse - Programa Caminante_DATOS.....	20
src.....	20
gen.....	20
Android.....	21
assets.....	21
res.....	21
- AndroidManifest.xml.....	22
<b>3.- Programa Caminante_DATOS.....</b>	<b>24</b>
3.1.- Primer análisis.....	24
Qué necesito y qué propongo.....	24
3.2.- Análisis definitivo.....	25
3.3.- Desmenuzando el Código.....	26
3.3.1.- layout - main.....	27
3.3.2.- Activity - PantallaCAMINANTE.....	28
3.3.3.- AndroidManifest.xml.....	34



## Aplicación Android de Asistencia al Caminante

3.4.- Datos.....	35
3.4.1.- Datos acelerómetro.....	35
3.4.2.- Datos orientación.....	38
<b>4.- Programa Caminante.....</b>	<b>41</b>
4.1.- Análisis.....	41
4.1.1. Análisis general.....	41
4.1.2.- Análisis de Bloqueo.....	42
4.1.3.- Análisis de Caída.....	43
4.2.- Desarrollo.....	43
4.2.1.- Diagrama de clases.....	44
4.2.2.- Diagrama de la clase MenuCaminante.....	45
4.2.3.- Diagrama de Clase de SensorsService.....	45
4.2.4.- Diagrama Clase FuncionBloqueoCaída.....	45
4.3.- Código.....	45
<b>5.- Resultados.....</b>	<b>54</b>
<b>6.- Conclusiones.....</b>	<b>55</b>
6.1.- Conclusiones subjetivas.....	55
6.2.- Dificultades.....	56
6.3.- Ampliaciones y mejoras.....	56
6.4.- Icono.....	56
<b>7.- Instalación de los programas.....</b>	<b>57</b>
<b>8. Bibliografía.....</b>	<b>57</b>



## **Motivación.**

### **¿Porqué este proyecto?**

Cuando se diagnóstica una enfermedad no común y/o grave, las reacciones en los pacientes y su entorno pueden ser de muy distinta índole. Pero hay algo que en la mayoría de los casos se produce y consiste en saber más sobre ella. Para esto último Internet es una gran fuente de información aunque también de peligro puesto que no siempre la información esta contrastado y puede ser causa de falsas expectativas e incluso de estafas por tanto en temas sensibles como este cualquier precaución es poca. Además de Internet la prensa tradicional sigue aportando conocimiento y en el caso que nos ocupa ha sido esencial puesto que la noticia aparecida en la prensa

[http://www.lavozdegalicia.es/sociedad/2010/03/22/0003\\_8371279.htm?](http://www.lavozdegalicia.es/sociedad/2010/03/22/0003_8371279.htm?utm_source=buscavoz&utm_medium=buscavoz)

[utm\\_source=buscavoz&utm\\_medium=buscavoz](http://www.lavozdegalicia.es/sociedad/2010/03/22/0003_8371279.htm?utm_source=buscavoz&utm_medium=buscavoz) hizo que me pusiera en contacto con Miguel Fernandez del Olmo al cual, de manera informal, le pregunté si necesitaba hacer alguna aplicación informática toda vez que tenía pendiente la propuesta del trabajo fin de máster. La idea fue acogida con ánimo por Miguel Fernandez que fijó los fundamentos de la aplicación que expuestos al tutor del presente trabajo Gregorio Robles consideró aceptables para desarrollar como proyecto fin de máster.

Todos de acuerdo, hemos acometidos la realización de la APLICACIÓN ANDROID DE ASISTENCIA AL CAMINANTE.

### **¿Porqué software libre?**

Este trabajo forma parte de un máster en software libre, por tanto es evidente que deba utilizar software de este tipo, pero además consideramos que cualquier aportación a la mejoría en la movilidad de un paciente en este caso con enfermedad de Parkinson merece ya no que sea gratuito, puesto que como es sabido o debería serlo libre no significa gratuito pudiéndose cobrar por las aplicaciones, sino que el código de la aplicación pueda ser compartido y modificado, con las únicas restricciones que podría permitir la licencia GPL v3 que es la que hemos asumido para este trabajo.

### **¿Porqué Android?**

Para lo que pretendemos, que es ayudar a los pacientes con enfermedad de Parkinson de forma autónoma, necesitamos un dispositivo potente pero a la vez pequeño y que disponga de sensores de movimiento y los smarphones cumplen esos requisitos. En el mercado existen una gran variedad de modelos con distintos sistemas operativos: Wndows Mobile, Symbian, iOS (iPhone), Android, siendo este último el único libre con una, cada vez más, importante implantación en el mercado, una gran cantidad de desarrolladores y numerosisimas aplicaciones y por tanto con mucho código disponible.





## Aplicación Android de Asistencia al Caminante

### JUSTIFICACIÓN.

La enfermedad de Parkinson (EP) es una enfermedad neurodegenerativa, en estos momentos incurable que, entre otros, presenta unos síntomas asociados a numerosos trastornos del movimiento. La acción de los medicamentos consigue, por regla general, unos buenos resultados sintomáticos pero sus efectos beneficiosos van desapareciendo paulatinamente haciéndose más patentes los efectos secundarios, por este motivo el conseguir pautas terapéuticas distintas de las puramente farmacológicas se presenta como una opción más que importante, necesaria.

Uno de las principales afectaciones de estos pacientes y que más repercuten en su calidad de vida es la dificultad de realizar un patrón correcto de marcha. A medida que progresa la enfermedad los pacientes con EP tienden a disminuir su amplitud de paso junto con un ritmo de marcha inestable, a esto hay que añadirle los episodios de bloqueo, durante los cuales el paciente es incapaz de iniciar la marcha, y las caídas. Todo esto ocasiona que el paciente opte por andar lo menos posible lo cual acarrea un empeoramiento de estos síntomas creando así un círculo vicioso que afecta dramáticamente a la movilidad y por tanto a su calidad de vida y a la de su entorno.

En los últimos años se ha producido importantes avances en el ámbito de la investigación en la rehabilitación del Parkinson. Actualmente existen una diversidad de estrategias que permiten mejorar o en su medida minimizar el deterioro motor de los pacientes con EP, y más concretamente la capacidad de marcha de estos.

La tecnología en su continuo avance nos proporciona aparatos cada vez más potentes y pequeños destacando desde el punto de vista del usuario normal los llamados comercialmente smartphones que además de dar servicio de telefonía móvil, que sería su primer cometido, se convierten en auténticos ordenadores de bolsillo.

En base a la literatura especializada en la rehabilitación en la enfermedad de Parkinson y a los conocimientos e ideas que sobre el tema posee y expone Miguel Fernandez del Olmo, Doctor en Educación Física y profesor titular de la asignatura Aprendizaje y Control Motor en la Facultad de Ciencias del Deporte y la Educación Física de la Universidad de A Coruña y director del equipo de Aprendizaje y Control del Movimiento Humano (<http://www.motorcontrolgroup.com>) pretendemos que móviles bajo el sistema operativo abierto Android, se constituyan en una herramienta de apoyo para la marcha en pacientes que sufren esta enfermedad.

Las tres opciones que mostramos en el resumen conforman los elementos sobre los que actuaremos para conseguir una autentica ayuda a la movilidad y a la seguridad.



# 1.- Android. Teoría general.

## 1.1.- Android:

De manera concisa diremos de Android:

- Es un Sistema Operativo para dispositivos móviles (smartphones, tablets, etc)
- Es un Proyecto de software libre con licencia [Apache 2.0](#).
- Está Desarrollado inicialmente por Android Inc comprada por Google que comanda la [Open Handset Alliance](#). Que actualmente lo desarrolla junto con innumerables colaboradores.
- Tiene el Núcleo monolítico de Linux.

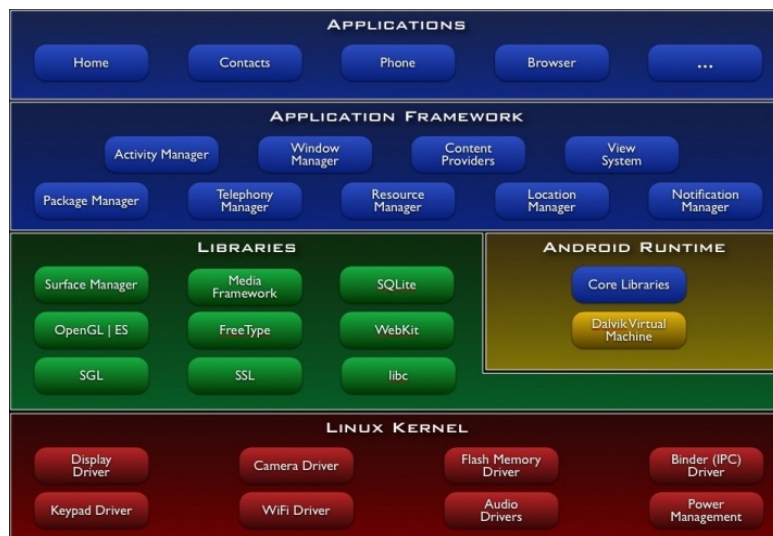
pero por Android se conoce no solamente un sistema operativo, sino que también es:

- un lenguaje de programación. Como lenguaje de programación utiliza Java, pero hay que dejar claro que no es Java, de echo no implementa todas las bibliotecas de [Java ME](#) (Java Micro Edition). Tampoco utiliza la maquina virtual de Java ([JVM](#)) que esta basada en una pila, sino una propia llamada [Dalvik](#) que esta basada en registros.
- un framework para desarrollar aplicaciones.

es decir se trata de una plataforma para dispositivos móviles

## 1.2.- Arquitectura de Android.

La siguiente imagen ((c) Google) recoge la pila de software que conforma Android en la cual se incluyen el sistema operativo, un [middleware](#) y las aplicaciones básicas para el usuario. Las distintas capas que componen la pila utilizan los servicios ofrecidos por las anteriores y ofrece los suyos a las capas superiores



*Arquitectura de Android*



## Aplicación Android de Asistencia al Caminante

**1.2.1.- Kernel de Linux:** los servicios base del sistema, seguridad, memoria, procesos, red, etc. son aportados por el núcleo de Linux, que actúa también como capa de abstracción entre las superiores de software y el hardware específico de cada máquina

**1.2.2.- Librerías:** En esta capa están las librerías usadas por Android. Estas escritas en C y C++ y dotan a Android de la mayor parte de sus capacidades. Junto con el Núcleo constituye en corazón de Android. Entre las más importantes están:

- **libc:** cabeceras y funciones según el estándar de C.
- **Surface Manager** (administrador de superficies):  
compone los distintos elementos de navegación de pantalla. Gestiona las ventanas de las distintas aplicaciones abiertas en cada momento.
- **OpenGL/SL y SGL:**  
son las librerías gráficas y sustentan la capacidad gráfica de Android. OpenGL/SL nos proporciona y maneja gráficos 3D, (si el hardware lo permite), mientras que de los 2D se encarga SGL.
- **Media Libraries:**  
de ella obtenemos los codesc necesarios para los contenidos multimedia.
- **FreeType:** para trabajar de forma rápida y sencilla con variedad de fuentes.
- **SSL:** permite la utilización de ese protocolo para comunicaciones seguras.
- **SQLite:** para creación y gestión de bases de datos relacionales.
- **WebKit:** motor para aplicaciones tipo navegador.

**1.2.3.- Runtime de Android** (tiempo de ejecución): se encuentra en el mismo nivel de capa que las librerías citadas anteriormente. Este entorno de ejecución esta formado por las Core Libraries que contiene multitud de clases Java y la máquina virtual Dalvik.

**1.2.4.- Armazón (framework) de Aplicaciones:** cualquier aplicación desarrollada para Android utiliza fundamentalmente este conjunto de herramientas de desarrollo, es decir usan el mismo conjunto de APIs y el mismo framework representado por este capa.

Las API más importantes situadas aquí son:

- **Activity Manager:**  
API que gestionan el ciclo de vida de las aplicaciones.
- **Window Manager:**  
gestiona las ventanas utilizando la librería Surface Manager
- **Telephone Manager:**  
las API vinculadas a las funcionalidades del teléfono se encuentran aquí.
- **Content Provider:**  
permite que los datos de una aplicación puedan ser compartidos con las otras aplicaciones en Android.
- **View System:**  
proporciona los elementos, en gran número, que nos van a permitir construir interfaces de usuario (GUI).



### Aplicación Android de Asistencia al Caminante

- **Location Manager:**  
nos permite obtener información de posicionamiento y localización.
- **Notification Manager:**  
gracias a estas, las aplicaciones permiten comunicar eventos que suceden durante una ejecución y ajenos a esta como puede ser una llamada entrante, etc. Si estos eventos requieren una acción (Intent) esta se activa mediante un clic.
- **XMPP Service:**  
API que permiten el uso de este protocolo de intercambio de mensajes basado en XML

**1.2.5.- Aplicaciones:** este nivel lo conforman las aplicaciones propiamente dichas entre las que se encuentran las que trae Android por defecto como todas aquellas que se quieran cargar. Para su funcionamiento tienen a su disposición el software de todas las capas citadas anteriormente.

### 1.3.- Dalvik, la máquina virtual para Android.

Dalvik como máquina virtual que es nos permite que las aplicaciones escritas para Android se ejecuten en distintas máquinas reales independizando el software de aplicación de cada hardware específico.

Ha sido diseñada y creada por “Dan Borstein” y otros ingenieros de Google.

Como ya hemos dicho está basada en registros y actúa como intérprete corriendo solamente los archivos ejecutables que tienen formato .dex (Dalvik Executable), por tanto el código ejecutable no se basa en el “bytecode” de Java.

Para la creación de los archivos dex, el SDK (Software Development Kit) proporciona la herramienta dx que transforma las clase compiladas de Java (.class) en .dex.

Este formato (dex) esta optimizado para dispositivos pequeños, como los smartphones, obteniendo un eficiente almacenamiento en memoria lo que consigue pasando algunas tareas al núcleo (kernel) como la gestión de: hilos de ejecución, uso de memoria y procesos. Dalvik permite de manera optimizada la ejecución de varias instancias simultáneamente.

Además los archivos de las clases Java son combinados al crear los dex evitando duplicidades y reduciendo significativamente el tamaño en comparación con su equivalente en Java.

La distribución e instalación de componentes y aplicaciones en Android se lleva a cabo por medio de unos ficheros comprimidos de extensión .apk (Android Package) que vendrían siendo los .jar de Java.

### 1.4.- Las APLICACIONES en Android.



## Aplicación Android de Asistencia al Caminante

Si bien en estos apartados tratamos cuestiones de índole teórica en este momento en que hablaremos de las aplicaciones en Android no lo haremos disertando sobre la ingeniería del software sino simplemente citando ciertos puntos, que entre lo leído para este trabajo y la experiencia personal y la adquirida durante el análisis del proyecto y su desarrollo, consideramos importantes sobre todo a la hora de enfrentarse a nuevos proyectos.

Las explicaciones que siguen se completan y/o amplían al tratar la parte práctica del presente trabajo centrándonos allí en los elementos que realmente hemos utilizado.

Característica fundamental de las aplicaciones Android es que, **se ejecuta dentro de su propio proceso Linux**, lo que hace que su ciclo de vida y tiempo no los controla la propia aplicación sino el sistema, para lo cual tiene en cuenta la combinación de distintas situaciones como pueden ser las aplicaciones en funcionamiento, la prioridad de estas para el usuario, la memoria disponible, etc.

Toda aplicación Android posee un fichero XML de nombre **AndroidManifest** (<http://developer.android.com/guide/topics/manifest/manifest-intro.html>) en el cual se deben de declarar los distintos elementos que componen la aplicación cómo:

- Actividades
- Servicios
- Puntos de entrada
- Capas
- Permisos
- etc

su estructura es la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
  <uses-permission />
  <permission />
  <permission-tree />
  <permission-group />
  <instrumentation />
  <uses-sdk />
  <uses-configuration />
  <uses-feature />
  <supports-screens />
  <compatible-screens />
  <supports-gl-texture />
  <application>
    <activity>
      <intent-filter>
        <action />
        <category />
        <data />
      </intent-filter>
      <meta-data />
    </activity>
    <activity-alias>
```



## Aplicación Android de Asistencia al Caminante

```

        <intent-filter> . . . </intent-filter>
        <meta-data />
    </activity-alias>
    <service>
        <intent-filter> . . . </intent-filter>
        <meta-data/>
    </service>
    <receiver>
        <intent-filter> . . . </intent-filter>
        <meta-data />
    </receiver>
    <provider>
        <grant-uri-permission />
        <meta-data />
    </provider>
    <uses-library />
</application>
</manifest>

```

No solamente es necesario reflejar en este fichero los componentes que se exigen y son utilizados por la aplicación sino que además es importante que se utilicen de forma apropiada si no queremos tener sorpresas durante el ciclo de vida de nuestra aplicación en ejecución.

La explicación de algunos de los elementos de este fichero se hacen en los siguientes apartados

### 1.4.1.- Componentes fundamentales:

- Una aplicación como la que nos ocupa, sobre todo para los aparatos que estamos utilizando como los smartphones, agradecen un entorno gráfico para su manejo, el siguiente componente nos lo proporciona:

**Activity.-** Es el componente más común y podríamos decir que más importante de la aplicaciones Android (en adelante AA) puesto que es el que lleva asociado de forma primordial la interfaz gráfica de usuario (GUI). Se implementa a partir de una clase que tiene su mismo nombre Activity. En un párrafo posterior veremos el importante **ciclo de vida** de este componente.

- Ahora que tenemos la interfaz de usuario queremos que desde esta se realicen acciones y para esto tenemos:

**Intent.-** Básicamente consiste en al voluntad de realizar una acción, pero esta no tiene porque estar en la misma aplicación pudiendo el Intent delegar el trabajo a realizar en otra de entre las instaladas, encargándose el sistema de esta labor buscando entre estas los Intents con la información que más se ajuste a la del Intent que hace la solicitud. Un ejemplo claro de esto es la petición de una URL, no indicamos el navegador, pero el sistema nos busca uno que este instalado.

- Si en vez de lo anterior queremos que se lance alguna acción cuando se produzca un evento aunque este se tenga lugar en otra aplicación debemos utilizar:



## Aplicación Android de Asistencia al Caminante

**Broadcast Intent Receiver.**- Como hemos dicho, permite lanzar un proceso dentro de la aplicación pero que proceda de otra. Pese a no tener interfaz de usuario puede, por medio del API Notification Manager, a través de la barra de estado del móvil avisar al usuario de lo ocurrido. Se implementa por medio de la clase BroadcastReceiver. No requiere que la aplicación esté activa cuando se produce el evento, el sistema se encargará de lanzarla si es necesario. Un ejemplo típico del uso de este componente es la recepción de llamadas, correos o SMS.

– Hasta ahora hemos visto como tenemos un componente que utiliza la interfaz de usuario, otra que lanza procesos incluso de aplicaciones distintas y otra que recibe notificaciones de eventos incluso de fuera, pero algunas veces no necesitamos tener delante una GUI pero si que se este ejecutando un proceso. Para esta necesidad Android nos propone el componente:

**Service.**- Este componente nos permite ejecutar aplicaciones que no necesitan interfaz de usuario y que por tanto se ejecutan comúnmente en segundo plano. Se implementa con una clase del mismo nombre. Podremos decir que su ejemplo más característico lo constituye un reproductor de música que generalmente esta constituido por una Activity con su GUI que lanza la canción que escucharemos (la reproducción seria el Service) mientras la GUI puede permanecer en pantalla

– Todos los componentes anteriores están muy bien, pero ¿y si queremos guardar información?, Android no nos puede dejar sin esto y nos ofrece:

**Content Provider.**- Este componente le permite a toda aplicación que lo implemente guarda información en formato base de datos SQLite, en un fichero y en multitud de otros formatos, existiendo en el paquete android.provider distintas clases que admiten distintos tipos de gestión de datos, debiendo pertenecer a este paquete cualquier formato que deseemos implementar. La clase que implemente este componente deberá contener los métodos que permitan almacenar, recuperar, actualizar y compartir los datos. Más adelante veremos que la información la podemos guardar en la memoria interna o en tarjetas tipo sdcar.

### 1.4.2.- Procesos y sus estados.

Ya se indicó que cada aplicación Android corre en su propio proceso, este proceso lo crea la aplicación cuando se ejecuta pero finaliza no solamente cuando esta deja de ejecutarse sino también por requerimientos del sistema como puede ser la necesidad de memoria. Como no todos los procesos son iguales el sistema no puede actuar sobre ellos de cualquier manera y de hecho crea una jerarquía de procesos basada en **estados**, que citamos a continuación

- **Primer plano (Active process):** aloja una Activity en la pantalla con la que actúa el usuario llamando a su método onResume o esta ejecutándose un IntentReceiver: Son los procesos eliminados como último recurso si el sistema tiene necesidades de memoria.

- **Visibles (Visible process):** también aloja una Activity pero esta no esta en primer plano porque se ha llamado al método onPause. Esta situación se da cuando cuando la aplicación interactúa con el usuario a través de un cuadro de diálogo. Aún en el caso de necesidades de memoria para mantener corriendo las aplicaciones en primer plano esté proceso no será



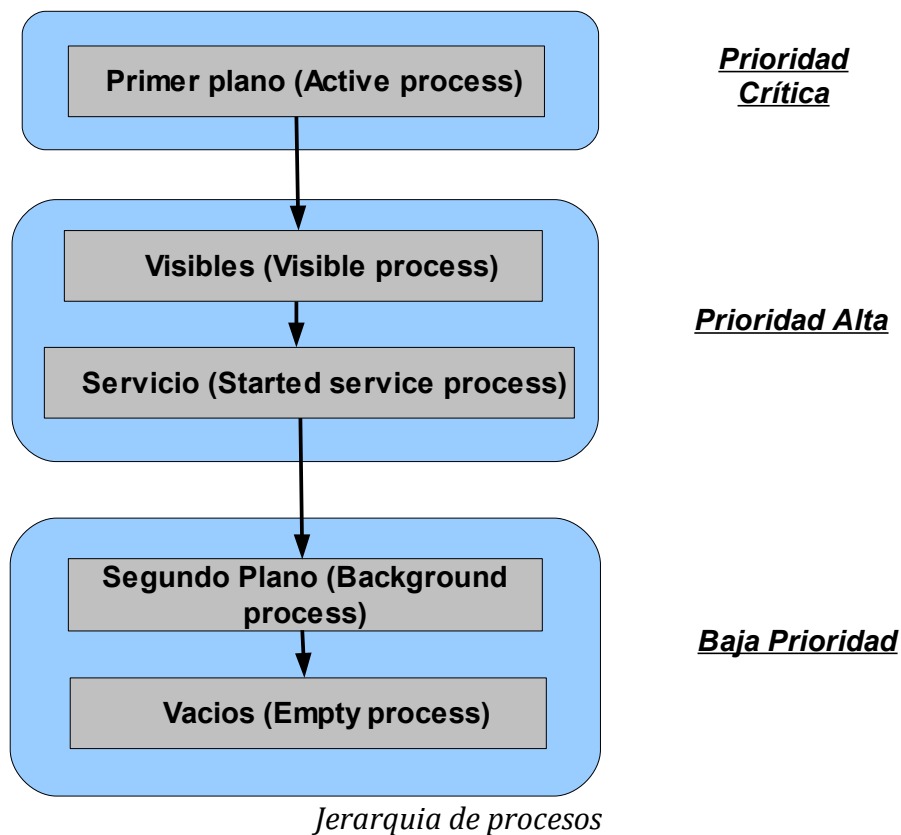


### Aplicación Android de Asistencia al Caminante

eliminado.

- **Servicio (Started service process):** aloja un Service iniciado con el método startService. Son procesos no visibles y por lo general importantes para el usuario.
- **Segundo Plano (Background process):** aloja una Activity que no esta visible para es usuario en ese momento por haberse activado su método onStop. Por lo general suelen existir varios procesos en esta situación y su eliminación generalmente no supone un problema, de cualquier manera el sistema elimina en último lugar el último visto por el usuario.
- **Vacios (Empty process):** No alojan ningún componente y existen para tener caché disponible para su próxima activación. Se eliminan con frecuencia por necesidades de memoria.

El esquema siguiente muestra la jerarquía anterior:



Se recomienda visitar la página

<http://developer.android.com/guide/topics/fundamentals.html>

#### 1.4.3.- Activity - Ciclo de Vida

Con anterioridad hemos dicho que en Android una parte importante del control de los procesos es llevado a cabo por el propio sistema operativo, así lanzar y parar procesos, gestionar la ejecución de estos, asignar los recursos en función de las peticiones de la aplicación y la



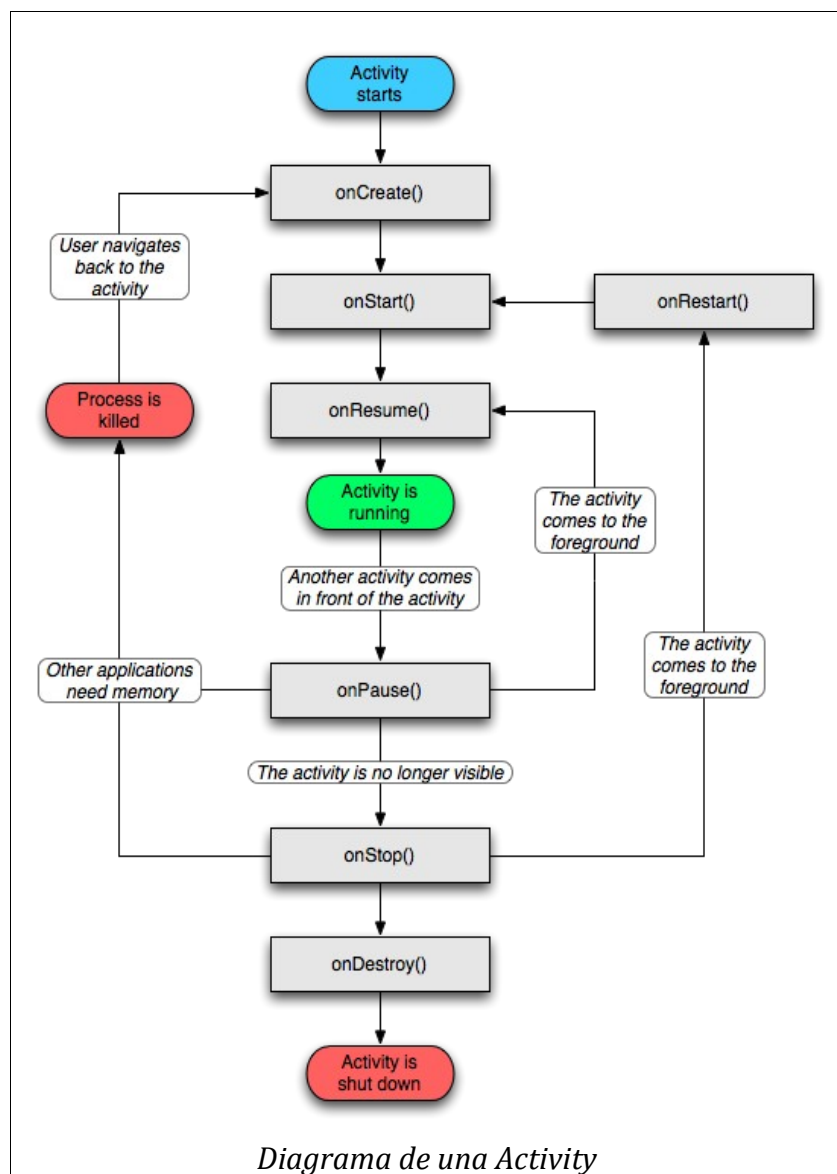


Aplicación Android de Asistencia al Caminante

disponibilidad, son tareas desconocidas por el usuario. Se puede decir que una característica poco común y fundamental de Android es que el tiempo de vida de los procesos de las aplicaciones **no** están controlados directamente por la propia aplicación.

De lo anterior se puede pensar que poco le queda al programador y no es cierto porque cada uno de los componentes de Android tiene un ciclo de vida perfectamente definido y que se debe seguir para que el desarrollador tenga controlado lo que en cada momento está realizando un determinado componente y poder generar de esta manera el código que realice en nuestra aplicación la acción que deseamos. No utilizar correctamente los componentes puede dar lugar a que el sistema mate procesos de la aplicación que los sustenta mientras está realizando labores importantes.

En la siguiente imagen, del SDK de Android, mostramos el ciclo de vida básico de una Activity.





### Aplicación Android de Asistencia al Caminante

De este diagrama podemos deducir:

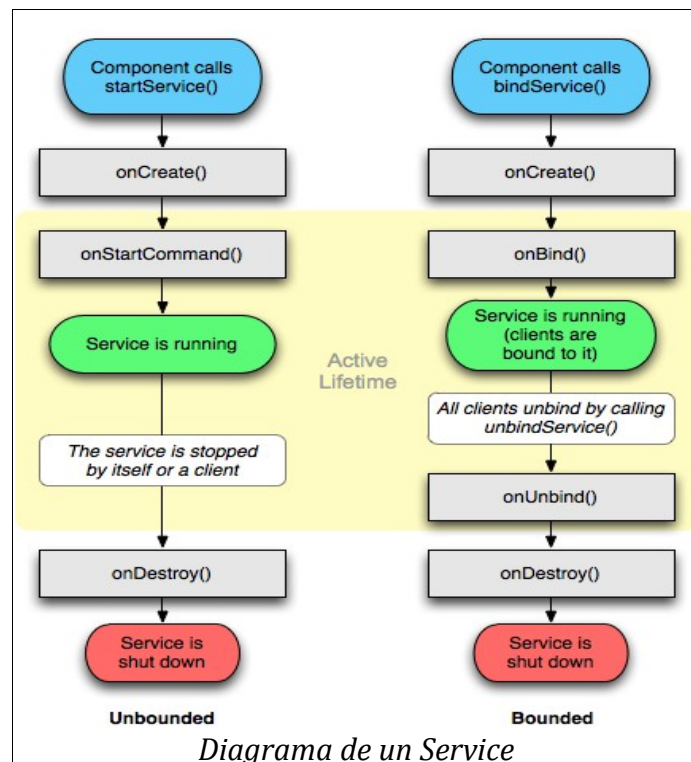
- El ciclo de vida esta comprendido entre los métodos onCreate() - principio y onDestroy - fin de la actividad.
- La parte visible del ciclo de vida esta comprendida entre los métodos onStart() y onStop. Pueden ser llamados más de una vez. Es posible que no tenga el foco de la acción.
- La parte útil del ciclo de vida está delimitada por los métodos onResume() y onPause(). La actividad no solo es visible sino que además tiene el foco de la acción pudiendo el usuario interactuar sobre ella.

También es importante constatar que el proceso de esta actividad puede ser eliminado cuando la aplicación no tiene el foco, circunstancia que se da cuando el ciclo está en onPause() o onStop(). Nunca Android elimina procesos que están interactuando con el usuario.

Cuando un proceso no esta activo el usuario desconoce su situación, de hecho puede estar en segundo plano y si se quisiera volver a el lo restauraría un onStart(), pero también puede estar dormido y Android la despertaría si deseáramos utilizarlo de nuevo, ahora bien, si en vez de estar dormido por necesidad el sistema lo hubiera eliminado se restauraría su situación anterior gracias a que Android guarda el estado del proceso.

#### 1.4.4.- Service - Ciclo de Vida

Si bien no vamos a explicar el ciclo de vida de un Service, dado que ha sido utilizado en el programa Caminante, lo mostramos aquí. De los dos esquemas utilizamos el ilimitado (unbounded)





## 2.- Android. Desarrollo.

Hemos hecho una pequeña introducción teórica sobre lo que es Android y no nos quedaremos ahí en esa vertiente, pero ahora comenzamos con la parte más práctica que consistirá en un leve paso sobre el entorno de desarrollo empleado en este trabajo para posteriormente meternos ya en la realización de la aplicación propiamente dicha. De aquí al final mostraremos:

- **Sistema operativo empleado para el desarrollo:** nos hemos decidido por Windows porque es el sistema disponible en prácticamente todos los ordenadores que puedo emplear aparte de mi ya viejo portátil, en donde si tengo también Linux, pero que no soporta las emulaciones de Android por parte de Eclipse y al principio de este trabajo no tenía a mi disposición ningún móvil con Android que me permitiera hacer las pruebas sobre él.
- **Entorno de desarrollo empleado:** utilizamos Eclipse con el SDK de Android y los plugins correspondientes. Toda vez que Eclipse funciona tanto en Linux como en Windows, a la elección indicada en el punto anterior no le falta una dosis de comodidad.
- **Estructura de un proyecto Android:** nos apoyaremos en la perspectiva visual de eclipse
- **Programa Caminante\_DATOS:** conforma la primera parte del proyecto aunque es un programa que no forma parte de la aplicación final ha sido muy importante puesto que ha supuesto, desde el punto de vista del código, el primer contacto y por tanto también los primeros errores y aciertos. Visto desde la aplicación consiste en la captura de datos de los sensores para su almacenamiento y análisis en la procura de pautas que permitan la detección de los bloqueos y la prevención de las caídas. Se aprovechará para continuar con explicaciones teóricas.
- **Programa Caminante:** es el fin perseguido por este proyecto y como ya se explicó consta de tres opciones marcha, bloqueo y caída.

### 2.1.- Eclipse y SDK de Android.

**Eclipse** es un entorno de desarrollo integrado (IDE en inglés) multiplataforma, escrito mayoritariamente en Java, de código abierto (Licencia Pública Eclipse) y que tiene versión en castellano. Es el IDE típico para desarrollo de Java (JDK), pero por medio del uso de plugins se utiliza para otros lenguajes como C/C++, Python, también para entornos de base de datos, admite control de versiones como CVS, SVN e incluso Git, etc., pero lo que nos interesa a nosotros es como entorno para Android. <http://www.eclipse.org/>

**SDK de Android:** Android tiene su propio kit de desarrollo de software que se puede integrar en eclipse instalando el correspondiente plugins que para nuestro caso lleva el nombre de ADT (Android Development Tools). No explicaremos como se realiza la instalación porque existen en la red suficientes manuales, además las nuevas versiones hacen que algunas veces las explicaciones queden obsoletas. A continuación daremos una relación de direcciones que consideramos convenientes para disponer del entorno eclipse para Android.



## Aplicación Android de Asistencia al Caminante

<http://developer.android.com/sdk/index.html> forma parte de la web oficial pero está en inglés y no existe versión en español, a partir de ella se puede obtener, además del software, información para la instalación,

Otras páginas en español, que pueden ayudar:

<http://sites.google.com/site/swcuc3m/home>

<http://www.android-spa.com/infoAndroid.php?tag=instalarSDKyPluginEclipse>

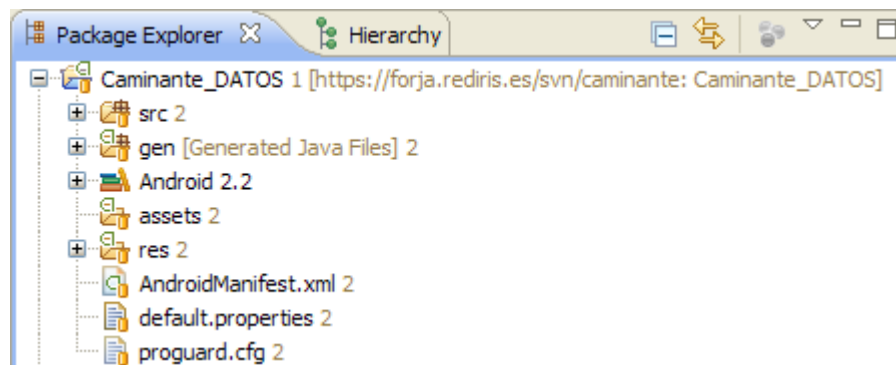
<http://androide.hijodeblog.com/2009/10/27/>

<http://www.epidataconsulting.com/tikiwiki/tiki-index.php?page=Android>

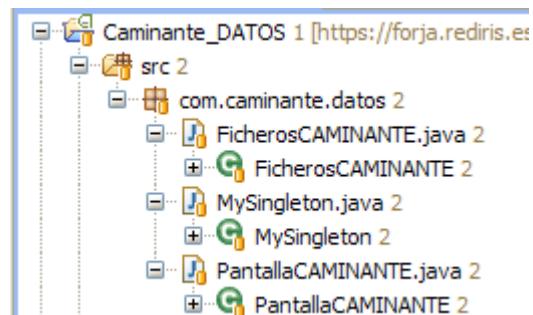
## 2.2.- Estructura de Android - Eclipse - Programa Caminante\_DATOS

Aprovecharemos el programa Caminante\_DATOS para ver la estructura de una aplicación en Android con la estética de eclipse.

Vemos que de la carpeta del proyecto cuelgan otras carpetas y un par de ficheros:



- **src** En esta carpeta se guardan el código fuente de la aplicación agrupados dentro del paquete que hemos definido.



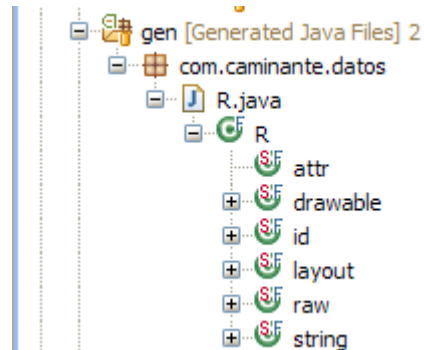
Vemos que en este caso tenemos tres ficheros y cada uno con una clase. Si desplegamos las clases podemos ver los métodos y variables que utiliza.

--**gen** Al compilar un proyecto Android de forma automática se generan unos ficheros de



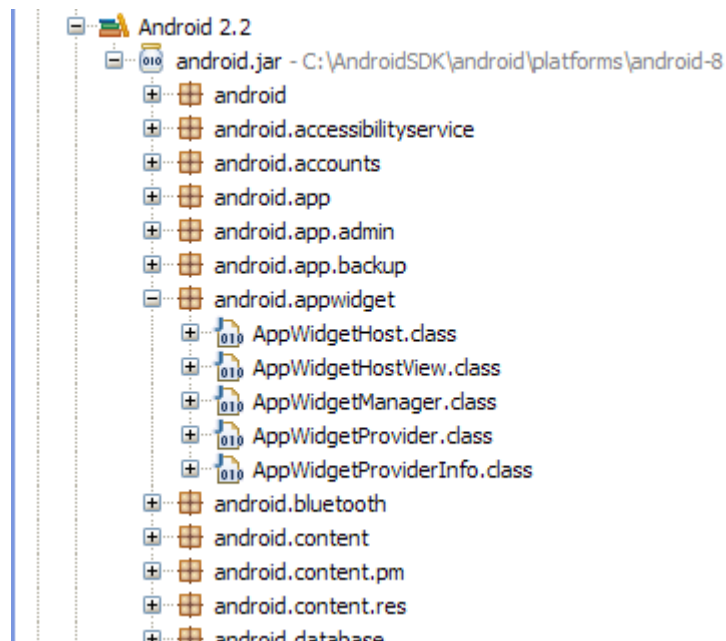
### Aplicación Android de Asistencia al Caminante

código en java que se guardan en esta carpeta y que están destinados al control de los recursos de la aplicación.



El más importante es R (el único que aparece en nuestro caso) en el que esta definido la clase R la cual contiene una constantes con los ID de los recursos de la aplicación definidos en la carpeta /res/ de la cual hablaremos más adelante. Estas constantes nos permiten acceder a los recursos que representan desde el código de nuestras aplicaciones

- **Android** Referencia al fichero jar que contiene los paquetes de Android para la versión que utilizamos para el desarrollo, en nuestro caso la 2.2. La figura de abajo es una pequeña muestra de estos paquetes y su contenido.



- **assetts** En nuestro ejemplo esta vacía esta carpeta, pero su misión es contener ficheros auxiliares de la aplicación pero que a diferencia de los que se verán a continuación de la carpeta /res/ no generan un ID en R y para acceder a ellos deberemos hacerlo a través de su ruta.

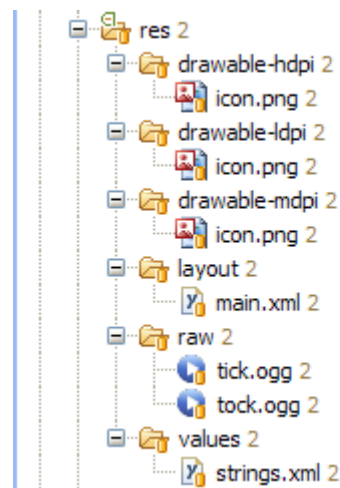


-**.res** En los apartados anteriores algo hemos dicho de esta carpeta como que contiene



### Aplicación Android de Asistencia al Caminante

ficheros de recursos necesarios para el proyecto y que a diferencia de los que contiene la carpeta /assets/ estos pueden ser reverenciados a través de los IDs que se generan en la clase R cuando se compila el proyecto. Como los recursos que utiliza una aplicación son variados, esta carpeta se divide en subcarpetas para recoger más ordenadamente esta variedad. En la figura las vemos:



- **drawable** se utiliza para contener la imágenes de la aplicación. En realidad aparecen tres para poder tener en cuenta la resolución del dispositivo que vamos a utilizar.
- **layout** contiene los ficheros que conforman la definición de la interfaz gráfica. En este caso tenemos una sola carpeta pero puede existir la layout-land si queremos definir diferentes interfaces en función de la orientación del dispositivo.
- **raw** contiene recursos adicionales que no se incluyen en el resto de las carpetas, generalmente en formato distinto de xml. En nuestro caso cobija dos ficheros de sonido en formato ogg que se utilizan para ser escuchados uno el activar el botón de iniciar y el otro con el botón terminar.
- **values** se utiliza para otros recursos de la aplicación, generalmente para cadenas de texto, como en nuestro caso, con el fichero strings.xml, colores con (colors.xml), estilos con (styles.xml), etc.

No aparecen en el ejemplo pero también pueden existir las carpetas:

- **anim** donde se pondrían los ficheros que definen animaciones.
- **menu** para la definición de los menús de la aplicación.
- **xml** con ficheros xml que utiliza la aplicación y que no se asignan a las otras carpetas.

- **AndroidManifest.xml**. (<http://developer.android.com/guide/topics/manifest/manifest-intro.html>)

Ya hemos hablado de este fichero, en este momento remarcamos un atributo importante con



### Aplicación Android de Asistencia al Caminante

respecto a la seguridad y que consiste en su utilización para otorgar permisos a la aplicación. A través de atributos del elemento `<uses-permission>` se definen y se matiza su alcance. En la clase `android.Manifest.Permission` se muestran todos los permisos que se pueden dar a ua aplicación.

(<http://developer.android.com/reference/android/Manifest.permission.html>)

- **default.properties**, -. **proguard.cfg**, Archivos generados automáticamente, el segundo de os cuales optimiza el código.





### 3.- Programa Caminante\_DATOS.

**Finalidad:** Captura y almacenamiento de datos con las coordenadas de los sensores necesarios, en un principio acelerómetro y orientación, para su análisis y estudio intentando conseguir unas pautas que nos lleven a detectar bloqueos para su eliminación, o prever caídas para evitarlas.

Constituye el paso previo a la realización del proyecto definitivo con el valor añadido de que mucho de su código y el conocimiento que aporta puede ser aprovechado en el trabajo final.

Este programa tiene el mérito de haber sido la primera incursión en Android por eso creo que puede resultar interesante hacer una recapitulación de las distintas etapas que ha tenido su desarrollo hasta llegar a la resultado final.

A continuación hablaremos de análisis, no se tome esto en el sentido concienzudo del término porque la complejidad de este programa viene más del código en si que del diseño.

#### 3.1.- Primer análisis.

¿Qué sabíamos de Android cuando comencé la realización de este proyecto?, pues poco más de que era un sistema operativo libre para móviles que permitía la ejecución de numerosas aplicaciones algunas de ellas tan curiosas para esa herramienta como una brújula, un detector de metales, etc. Cuando Miguel Fernández del Olmo me propuso hacer esta aplicación no conocía la existencia en estos aparatos de un sensor acelerómetro, por tanto la primera aproximación a Android debía hacerla con la vista puesta en el uso de sensores en especial el acelerómetro y el de orientación en principio, necesarios para conseguir los fines propuestos.

Una vez obtenida y leída variada documentación teórica sobre Android, de la que se ha hecho un resumen en apartados anteriores, y alguna más práctica sobre el uso de sensores acometí la escritura de las primeras líneas de código a partir del esquema simple, que expongo a continuación, fruto de un alto componente teórico que con la práctica ha sido modificado como se comentará

#### Qué necesito y qué propongo

– **(necesito)** Una interfaz de usuario que me permita inicializar y finalizar el programa, **(propongo)** en términos de Android, una Activity

– **(necesito)** Un proceso en segundo plano en que los sensores estén escuchando las modificaciones para lanzar los eventos, **(propongo)** por tanto pensamos en un Service. Nota: en un principio se pensó en obtener la lectura de los sensores en periodos regulares de tiempo (ver código siguiente), y con esa visión se hicieron las primeras pruebas en Android aunque sin el uso de sensores, sin embargo con posterioridad llegue al conocimiento de que los sensores trabajan por eventos enviando información cuando se produce un cambio en sus coordenadas, aunque internamente utilice barridos por tiempo, de echo se pueden cambiar esta velocidad de barrido,.





### Aplicación Android de Asistencia al Caminante

```
private void inicializarServicioTiempo() {
    //Registra los escuchadores
    try {
        this.timer = new Timer();
        this.timer.scheduleAtFixedRate (
            new TimerTask() {
                public void run() {
                    ejecutarTareaTiempo();
                }
            },
            0, 4000 );
    }
    catch (Exception e) {
        Log.i(getClass().getSimpleName(), e.getMessage());
    }
}

private void ejecutarTareaTiempo() {;
    // código para almacenar los datos u otra queramos
}
```

– **(necesito)** Un lugar en el que guardar los datos (**propongo**) pensando en una base de datos y que mejor que aprovechar SQLite que tiene su librería específica Android. Lo relativo a esto se pensó en ponerlo en otra Activity, evidentemente de forma errónea, puesto que no necesitábamos una interfaz de usuario que es una característica de las Activity, de hecho el programa funcionaba pero mostrando una pantalla en negro cuando pasaba el programa por esta actividad.

En resumen tenemos: **Activity – Service - SQLite**

Con estas premisas se creo un código con el que se lograba el fin de obtener y guardar datos, pero que tenía se punto débil en la finalización dado que al solicitarla lo hacía pero no de forma correcta sino por medio del indeseable cuadro de diálogo Aplicación No Responde (ANR - Application Not Responding). Esto posiblemente motivado por la no utilización de BroadcastReceiver para la comunicación entre la Activity y el Service

### 3.2.- Análisis definitivo.

En los apartados anteriores mostramos un análisis simple fruto de los conocimientos recién adquiridos y de la inexperiencia con Android que aunque dio lugar a un programa ejecutable podía y creo que ha sido mejorado.

Estructura Final: **Activity – Clase MySingleton – Clase FicherosCAMINANTE.**

Como se ve hay cambios significativos:

- desaparece el service. Pero ¿no necesitamos un proceso de escucha funcionando en segundo plano? Si, pero de lo que no teníamos conocimiento es de que el funcionamiento de los sensores viene siendo un trabajo en segundo plano, se activan y quedan en escucha esperando modificaciones en su valores para lanzar un evento, además este programa no tiene como finalidad funcionar durante mucho tiempo, sino el hacerlo por periodos limitados con una



### Aplicación Android de Asistencia al Caminante

necesidad relativa de proceso en segundo plano.

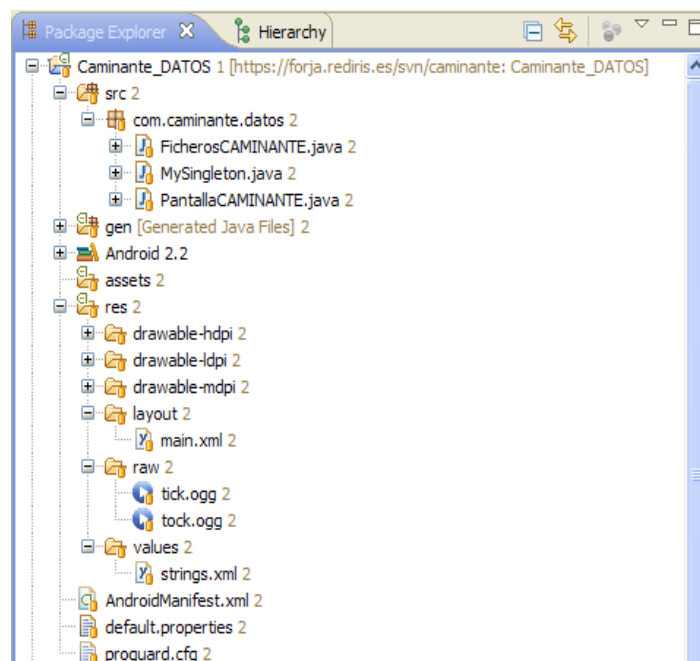
- se cambió el guardar los datos en una base de datos SQLite por hacerlo en un fichero de texto plano con una línea de cabecera que contiene los nombres de cada columna. Tanto las líneas de datos como la de cabecera llevan los campos separados por tabuladores. ¿Porqué este cambio? La decisión vino motivada por considerar este formato más práctico sobre todo a la hora de 'ver' los datos a través de una hoja de cálculo. Hay que resaltar un echo y es que los datos en SQLite en principio eran guardados en la memoria del sistema y estos hemos decidido guardarlos en una tarjeta de memoria (sdcar). Los datos obtenidos adjuntan y envían por correo electrónico a una cuenta de Gmail (waltercaminante@gmal.com) creada ex-proceso, pero teniendo en cuenta que no se predetermina el cliente de correo que se utiliza por tanto hay una visualización previa en cuyo momento puede cambiarse la cuenta de destino. Se siguiere el envío de datos adicionales como si existieron caídas, paradas etc. Ver el **apéndice 1 Manual de uso de Caminante\_DATOS** en el que se ha puesto el manual de uso de esta aplicación.

- se crea la clase MySingleton que nos permite tener en todo la aplicación una única instancia de la clase para la que se define en nuestro caso FicherosCAMINANTE. Quizás no sea necesaria pero lo que ha quedado de manifiesto es que no ha perjudicado en absoluto el funcionamiento del programa.

### 3.3.- Desmenuzando el Código.

Con anterioridad hemos hablado de la estructura de un proyecto Android tomando como ejemplo esta aplicación Caminante\_DATOS ahora llega el momento de desmenuzar el código que encierra esta estructura aprovechándolo para completar la parte teórica de Android.

Estructura:





## Aplicación Android de Asistencia al Caminante

### 3.3.1.- layout - main

Comenzamos por aquí porque es en donde se definen las pantallas que constituyen la interfaz de usuario. Para esto no se utilizó ninguna herramienta de ayuda para el diseño, se ha hecho todo de forma 'manual' puesto que opino que es una buena práctica, cuando se está comenzando con algo nuevo, procurar hacer algunas cosas de este modo.

código

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10px">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/introduccion"
        android:id="@+id/TextViewIntroduccion">
    </TextView>

    <EditText
        android:text="@string/nombre"
        android:id="@+id/editTextName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5px">
    </EditText>

    <RadioGroup
        android:id="@+id/gruporopciones"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
        <Button
            android:text="@string/iniciar"
            android:id="@+id/buttonStart"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight = "1"
            android:textSize="25dp"
            android:textStyle="bold"
            android:clickable="true"
            android:layout_gravity="left"
            android:layout_marginTop="5px">
        </Button>
        <Button
            android:text="@string/terminar"
            android:id="@+id/buttonFinish"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight = "1"
            android:textSize="25dp"
            </Button>
    </RadioGroup>
</LinearLayout>
```



### Aplicación Android de Asistencia al Caminante

```

        android:textStyle="bold"
        android:clickable="true"
        android:layout_gravity="right"
        android:layout_marginTop="5px">
    </Button>
</RadioGroup>
</LinearLayout>

```

Este código da como resultado la pantalla:



Tenemos por lo tanto, aparte de los cuadros de texto indicadores:

- Un cuadro de edición de texto para introducir un nombre que nos permita identificar quien originó los datos.
- Dos botones de radio para iniciar y finalizar el programa y por tanto la captura de datos.

### 3.3.2.- Activity - PantallaCAMINANTE

A continuación mostramos el código completo de esta Activity, lo haremos solo en este caso como ejemplo, el resto del código se enviará en ficheros adjuntos.

Este código que viene a continuación tiene párrafos auto-explicativos haremos incisos teóricos más completos utilizando apéndices.

```

/**
 * Nombre del Programa: Caminante_DATOS
 * Proyecto: Aplicación Android de Asistencia al Caminante
 * Uso: captura coordenadas de sensores \(acelerometro, orientación\)
   pasandolas a ficheros de texto para su estudio.
 **/
/**
  Copyright (C) 2011 Enrique Ramón Delgado Garrido

  This program is free software: you can redistribute it and/or modify

```



## Aplicación Android de Asistencia al Caminante

it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

```

**/
/**
 *
 **/
package com.caminante.datos;

import java.util.List;

import com.caminante.datos.R;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

/**
 * Esta clase Activity PantallaCAMINANTE constiruye el elemento inicial y principal
de este programa
 * muestra la pantalla y
 * captura los dato de entrada - nombre del caminante
 * gestion los botones y su estado - Inicio - Terminar
 * usa las clases
 * FicherosCAMINANTE
 * gestiona los ficheros de texto que guardan los datos
 * crea y envía el correo adjuntando los ficheros
 * MySingleton
 * gestion la utilización de los sensores
 *
 * @author Enrique Delgado
 */

    public class PantallaCAMINANTE extends Activity implements OnClickListener {

/** variables de los elementos de la pantalla */
        private Button startButton = null;
        private Button finishButton = null;

```



### Aplicación Android de Asistencia al Caminante

```

        private EditText      textName      =      null;

/** variables para el almacenamiento del Nombre del caminante
 * en -preferences- */
        private static final String      DEFAULT_WALKER= "Nombre";
        private String      WALKER      = DEFAULT_WALKER;
        private static final String      KEY_WALKER      = "NOMBRE_CAMINANTE";
        private static final String      PREFS      = "caminante.prefs";
    
```

#### Ver *apéndice 2 Almacenamiento de Datos en Android*

```

/** variable objeto de la clase FicherosCAMINANTE
 * para lo referente a los ficheros de almacenamiento */
        private FicherosCAMINANTE      dataFiles;

/** variables para los sensores */
        private SensorManager      sensorManager;
        private Sensor      accelerometer;
        private Sensor      orientation;
        //public float[]      coordenadasAceleracion = new float[3];
        //public float[]      coordenadasOrientacion = new float[3];
        static float[]      coordinatesZero      = new float[3];
    
```

#### Ver *apéndice 3 Sensores*

```

/** variable para el control del proceso */
        private boolean      sensorrunning = false;

/** variables para los sonidos */
        private MediaPlayer      mPStart;
        private MediaPlayer      mPStop;

/**
 * Módulo de inicio de la Activity
 * @param savedInstanceState
 * @see #onCreate(Bundle savedInstanceState)
 */
        // @Override
        public void onCreate(Bundle savedInstanceState) {
            //
            super.onCreate(savedInstanceState);
            // capturar y visualizar la pantalla
            setContentView(R.layout.main);
            // mantener el nombre del caminante entre sesiones
            SharedPreferences settings = getSharedPreferences(PREFS, 0);
            WALKER = settings.getString(KEY_WALKER, DEFAULT_WALKER);
            // módulo para inicializar lo concerniente a la pantalla
            setScreen();
            // módulo para inicializar lo concerniente a los sensores
            setSensors();
            // uso de singleton para tener una instancia de la clase en
            // toda la aplicación en este caso lo referente a ficheros
        }
    
```



### Aplicación Android de Asistencia al Caminante

```

MySingleton mySingleton = MySingleton.getInstance();
dataFiles = mySingleton.getDataFile();
}
/**
 */
protected void setScreen() {

    this.startButton=(Button) findViewById(R.id.buttonStart);

    if(this.startButton!=null) {
        this.startButton.setOnClickListener(this);
        this.startButton.setEnabled(true);
    }

    this.finishButton=(Button) findViewById(R.id.buttonFinish);

    if(this.finishButton!=null) {
        this.finishButton.setOnClickListener(this);
        this.finishButton.setEnabled(false);
    }

    // Área de texto
    CharSequence cS_Walker = wALKER;
    wALKER = cS_Walker.toString();
    textName = (EditText) findViewById(R.id.editTextName);
    textName.setText(cS_Walker);
    //((EditText) findViewById(R.id.editTextName)).setText(cS_Walker);
}
/**
 *
 */
protected void setSensors() {

    sensorManager =
(SensorManager) getSystemService(Context.SENSOR_SERVICE);

    List<Sensor> list;

    //Obtención del Sensor Acelerómetro
    list=sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
    if (list.size()>0) accelerometer=list.get(0);

    //Obtención del Sensor Orientación
    list=sensorManager.getSensorList(Sensor.TYPE_ORIENTATION);
    if (list.size()>0) orientation=list.get(0);

    for (int i=0;i<3;i++) {
        //coordenadasAceleracion[i]=0;
        //coordenadasOrientacion[i]=0;
        coordinatesZero[i]=0;
    }
}
/**
 * Gestor de los eventos Click
 */
@Override

```



### Aplicación Android de Asistencia al Caminante

```

public void onClick(View vista) {
    // captura del nombre del caminante
    WALKER =
((EditText) findViewById(R.id.editTextName)).getText().toString();
    // almacenamiento del nombre con preferences
    SharedPreferences settings = getSharedPreferences(PREFS, 0);
    SharedPreferences.Editor editor = settings.edit();
    editor.putString(KEY_WALKER, WALKER);
    editor.commit();
    //
    // Filtro de los controles
    // Botón de inicio de la captura de datos
    if(vista.getId()==R.id.buttonStart) {
        sensorrunning = true;
        //paso del nombre del caminante para su uso en la clase
        // FicherosCAMINANTE - uso de un constructor -ficticio-
        dataFiles = new FicherosCAMINANTE(WALKER, 0);
        // Llamada al constructor -real- de FicherosCAMINANTE
        dataFiles = new FicherosCAMINANTE(this);
        // uso de Singleton
        MySingleton mySingleton = MySingleton.getInstance();
        mySingleton.setDataFile(dataFiles);
        // Lanzar el proceso de escucha de los sensores
        onResumeSensors();
        // tratar elementos de pantalla
        this.startButton.setEnabled(false); //DESACTIVADO
        this.finishButton.setEnabled(true); //ACTIVADO
        this.textName.setEnabled(false);
        // emitir sonido de inicio
        MyPlayerStart(getContext());
        mPStart.start();
    }
    // Botón de terminación el proceso de captura de datos
    if(vista.getId()==R.id.buttonFinish) {
        sensorrunning = false;
        stopProcess();
        // emite el sonido de cierre
        MyPlayerStop(getContext());
        mPStop.start();
        // termina la aplicación
        finish();
    }
}
}
/** módulo para el comienzo real del uso de los sensores por parte del programa
 *
 */
@Override
protected void onResumeSensors() {

    super.onResume();

    //Registro de los escuchadores
    if (accelerometer!=null) {
        sensorManager.registerListener(sensorEventListener,
            //acelerometro, SensorManager.SENSOR_DELAY_NORMAL);
            accelerometer, SensorManager.SENSOR_DELAY_FASTEST);
    }
}

```





### Aplicación Android de Asistencia al Caminante

```

        if (orientation!=null) {
            sensorManager.registerListener(sensorEventListener,
                //orientacion,SensorManager.SENSOR_DELAY_NORMAL);
                orientation,SensorManager.SENSOR_DELAY_FASTEST);
        }
    }
}
/** módulo para la recepción de los valores de los sensores cuando se detecta
 * un evento de cambio en dichos valores
 */
private SensorEventListener sensorEventListener = new
SensorEventListener() {
    // variables de coordenadas
    public float[] coordinatesAccelerometer = new float[3];
    public float[] coordinatesOrientation = new float[3];
    // modulo para capturar los eventos de cambio de precisión
    // no se usa
    //@Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO Auto-generated method stub
    }
    // modulo para capturar los eventos de cambio de valor
    //@Override
    public void onSensorChanged(SensorEvent event) {

        //implantación de los escuchadores que sincronizamos
        synchronized (this) {
            //damos formato de número float a los valores de los eventos
            for (int i=0;i<3;i++) {
                int w=(int) (10*event.values[i]);
                event.values[i]=(float) (w/10.0f);
            }
            //actualización de los valores dependiendo del tipo de sensor
            if (event.sensor==accelerometer) {
                this.setAccelerometer(event.values);
                //escritura los dato en el fichero del acelerómetro
                // por medio del objeto de la clase FicherosCAMINANTE
                dataFiles.addDataACEL(coordinatesAccelerometer[0],
                    coordinatesAccelerometer[1],
                    coordinatesAccelerometer[2]);
            } else {
                this.setAccelerometer(coordinatesZero);
            }
            if (event.sensor==orientation) {
                this.setOrientacion(event.values);
                //escritura los dato en el fichero de orientación
                // por medio del objeto de la clase FicherosCAMINANTE
                dataFiles.addDataORIE(coordinatesOrientation[0],
                    coordinatesOrientation[1],
                    coordinatesOrientation[2]);
            } else {
                this.setOrientacion(coordinatesZero);
            }
        }
    }
    public void setAccelerometer (float[] coordinates) {
        this.coordinatesAccelerometer=coordinates;
    }
}

```



### Aplicación Android de Asistencia al Caminante

```

    }

    public void setOrientacion(float[] coordinates) {
        this.coordinatesOrientation=coordinates;
    }
};

protected void stopProcess() {
    // elimina de escucha los sensores
    sensorManager.unregisterListener(sensorEventListener);
    // uso del Singleton
    MySingleton mySingleton = MySingleton.getInstance();
    dataFiles = mySingleton.getDataFile();
    // cierra los ficheros (y lanza el correo)
    dataFiles.fileClose(this);
}

/**
 *
 * @param mPctx
 */
public void MyPlayerStart(Context mPctx) {
    mPStart = new MediaPlayer();
    mPStart = MediaPlayer.create(mPctx, R.raw.tick);
}

/**
 *
 * @param mPctx
 */
public void MyPlayerStop(Context mPctx) {
    mPStop = new MediaPlayer();
    mPStop = MediaPlayer.create(mPctx, R.raw.tock);
}

/** módulo para las paradas fuera del botón terminar
 *
 */
protected void onDestroy() {
    super.onDestroy();
    if (sensorrunning) {
        stopProcess();
        //sensorManager.unregisterListener(sensorEventListener);
    }
}
}

```

### 3.3.3.- AndroidManifest.xml

Como elemento importante en las aplicaciones Android mostraremos aquí el código del AndroidManifest.xml,

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0" package="com.caminante.datos">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name="com.caminante.datos.PantallaCAMINANTE"
            android:label="@string/app_name"

```



## Aplicación Android de Asistencia al Caminante

```

        android:screenOrientation = "portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

</application>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-sdk android:minSdkVersion="8" />
</manifest>

```

### 3.4.- Datos.

Nos adentramos ya en la parte que nos va a llevar a sacar algunas conclusiones para intentar deducir a través de los datos de los sensores si nos encontramos con un bloqueo (una parada) o una caída.

Hemos obtenido y analizado datos de personas sanas y enfermas, así como del móvil en la mano cambiando de posición, incluso hemos tomado datos de una patinadora y también nos han simulado caídas aunque estas con las necesarias precauciones.

Los datos han sido importados a unas hojas de cálculo de OpenOffice.org Calc y se han creado unos gráficos para su análisis., mostrando a continuación un ejemplo de esto.

El análisis de los datos y las conclusiones los dejamos para los apartados en que tratamos el programa final (Caminante) en donde asumiremos también el codificar las conclusiones.

#### 3.4.1.- Datos acelerómetro.

En la figura siguiente se muestran los datos obtenidos simplemente paseando por el pasillo de mi vivienda que si bien son sencillos han sido de gran importancia para obtener conclusiones.

Las tres primera columnas (A,B,C) esta claro lo que son simplemente aclarar que:

- coordenada `coorAcel_0` corresponde a la aceleración en  $m/s^2$  en el eje X (ver apéndice 3 Sensores).

- coordenada `coorAcel_1` corresponde a la aceleración en  $m/s^2$  en el eje Y.

- coordenada `coorAcel_2` corresponde a la aceleración en  $m/s^2$  en el eje Z.

y que los valores tienen en cuenta la aceleración de la gravedad.

La columna D es el timestamp del sistema en milisegundos.

La columna E es el tiempo en milisegundos referenciado al primer valor.

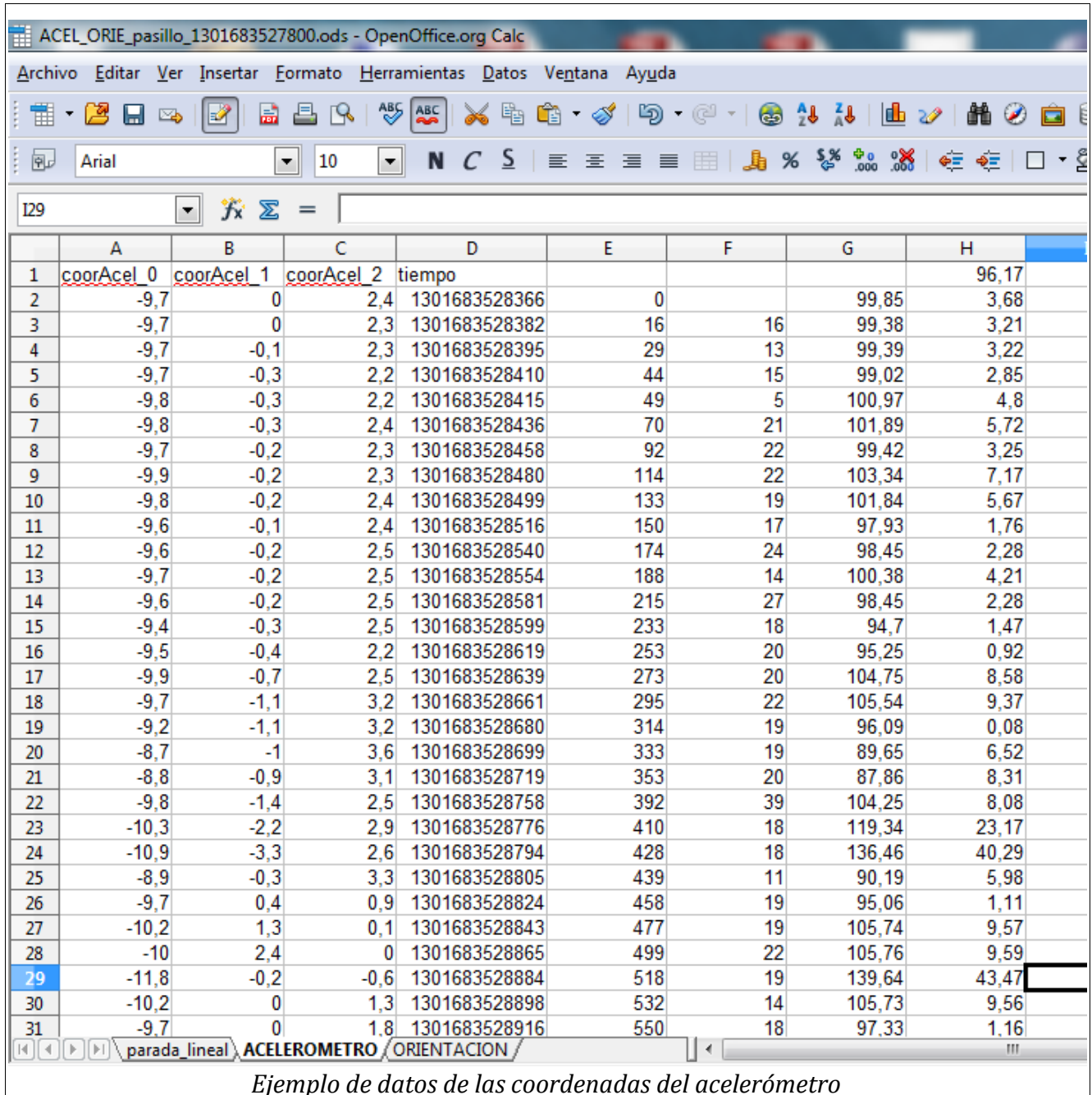
La columna F es el tiempo en milisegundos entre eventos.

La columna G es la módulo de la aceleración al cuadrado que se obtiene de la suma de los cuadrados de las columnas A, B y C.



### Aplicación Android de Asistencia al Caminante

La columna H es la diferencia entre la columna G y el cuadrado del valor de la gravedad tomada esta como  $9,82 \text{ m/s}^2$ .



	A	B	C	D	E	F	G	H
1	coorAcel 0	coorAcel 1	coorAcel 2	tiempo				96,17
2	-9,7	0	2,4	1301683528366	0		99,85	3,68
3	-9,7	0	2,3	1301683528382	16	16	99,38	3,21
4	-9,7	-0,1	2,3	1301683528395	29	13	99,39	3,22
5	-9,7	-0,3	2,2	1301683528410	44	15	99,02	2,85
6	-9,8	-0,3	2,2	1301683528415	49	5	100,97	4,8
7	-9,8	-0,3	2,4	1301683528436	70	21	101,89	5,72
8	-9,7	-0,2	2,3	1301683528458	92	22	99,42	3,25
9	-9,9	-0,2	2,3	1301683528480	114	22	103,34	7,17
10	-9,8	-0,2	2,4	1301683528499	133	19	101,84	5,67
11	-9,6	-0,1	2,4	1301683528516	150	17	97,93	1,76
12	-9,6	-0,2	2,5	1301683528540	174	24	98,45	2,28
13	-9,7	-0,2	2,5	1301683528554	188	14	100,38	4,21
14	-9,6	-0,2	2,5	1301683528581	215	27	98,45	2,28
15	-9,4	-0,3	2,5	1301683528599	233	18	94,7	1,47
16	-9,5	-0,4	2,2	1301683528619	253	20	95,25	0,92
17	-9,9	-0,7	2,5	1301683528639	273	20	104,75	8,58
18	-9,7	-1,1	3,2	1301683528661	295	22	105,54	9,37
19	-9,2	-1,1	3,2	1301683528680	314	19	96,09	0,08
20	-8,7	-1	3,6	1301683528699	333	19	89,65	6,52
21	-8,8	-0,9	3,1	1301683528719	353	20	87,86	8,31
22	-9,8	-1,4	2,5	1301683528758	392	39	104,25	8,08
23	-10,3	-2,2	2,9	1301683528776	410	18	119,34	23,17
24	-10,9	-3,3	2,6	1301683528794	428	18	136,46	40,29
25	-8,9	-0,3	3,3	1301683528805	439	11	90,19	5,98
26	-9,7	0,4	0,9	1301683528824	458	19	95,06	1,11
27	-10,2	1,3	0,1	1301683528843	477	19	105,74	9,57
28	-10	2,4	0	1301683528865	499	22	105,76	9,59
29	-11,8	-0,2	-0,6	1301683528884	518	19	139,64	43,47
30	-10,2	0	1,3	1301683528898	532	14	105,73	9,56
31	-9,7	0	1,8	1301683528916	550	18	97,33	1,16

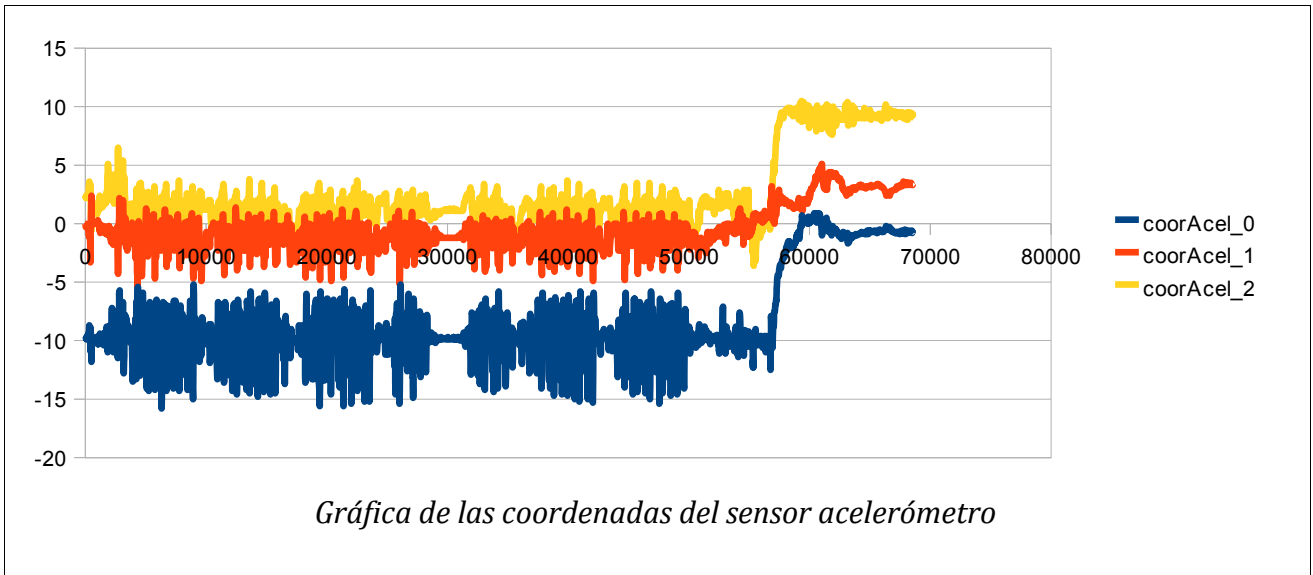
*Ejemplo de datos de las coordenadas del acelerómetro*

La gráficas que corresponden son:

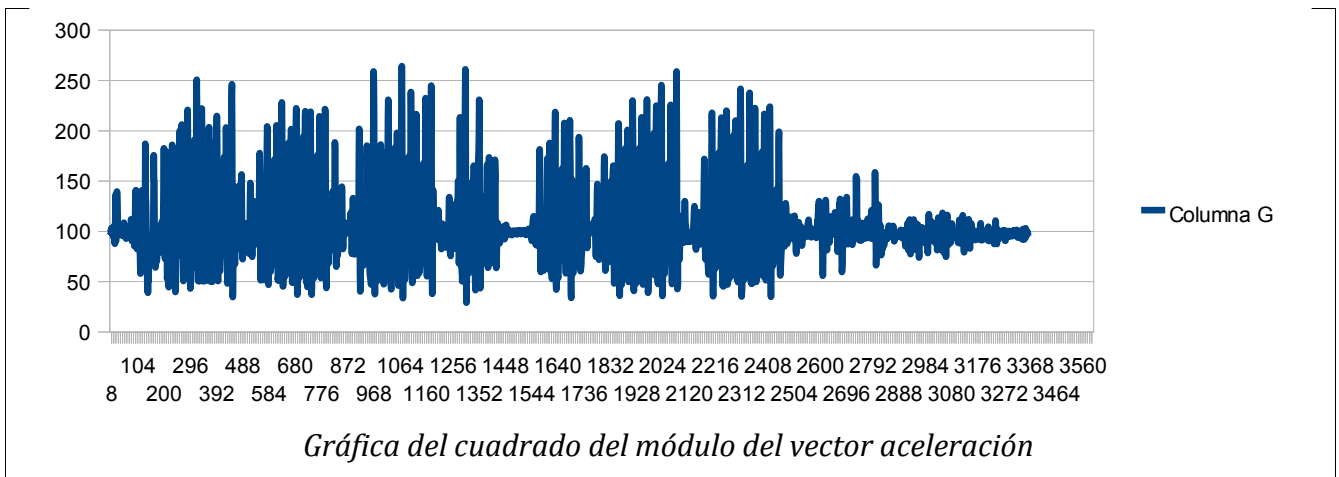
De las columnas A,B y C con respecto a la E.



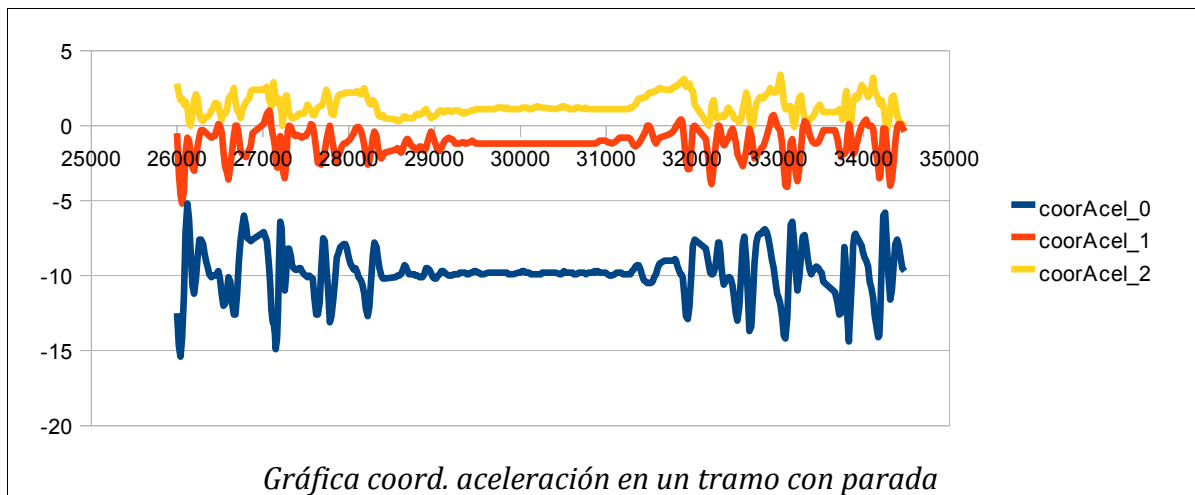
### Aplicación Android de Asistencia al Caminante



De la columna G respecto al número de línea (evento)

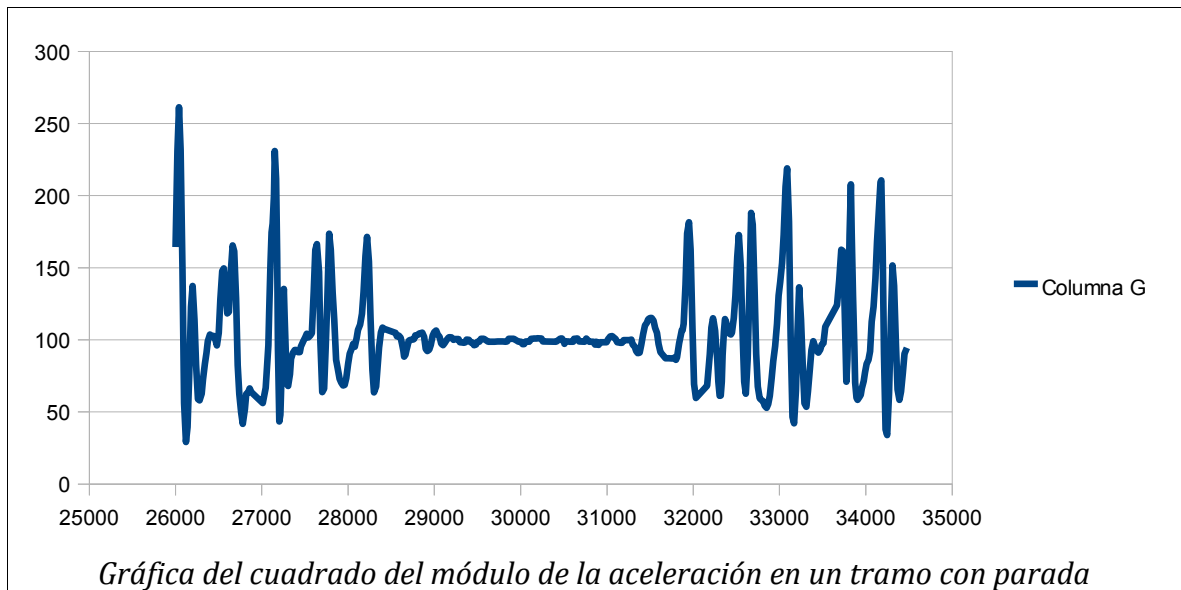


Es interesante ver la gráfica en los intervalos en que ha habido una parada.





### Aplicación Android de Asistencia al Caminante



#### 3.4.2.- Datos orientación.

Se muestra a continuación una pantalla con el ejemplo de los datos obtenidos en el mismo proceso que los anteriores de aceleración.

Las tres primera columnas (A,B,C) son (ver apéndice 3 Sensores):

- coordenada `coorMagn_0` corresponde en grados al giro al rededor del eje Z (0-259). Valor 0 cuando el eje Y apunta al norte.
- coordenada `coorMagn_1` corresponde en grados a la rotación alrededor del eje X. Valor positivo cuando el eje Z se mueve hacia el eje Y. (-180 a 180)
- coordenada `coorMagn_2` corresponde en grados a la rotación alrededor del eje Y. Valor positivo cuando el eje X se mueve hacia el eje Z. (-90 a 90).

La columna D es el timestamp del sistema en milisegundos.

La columna E es el tiempo en milisegundos referenciado al primer valor.

La columna F el el tiempo en milisegundos entre eventos.



Aplicación Android de Asistencia al Caminante

ACEL\_ORIE\_pasillo\_1301683527800.ods - OpenOffice.org Calc

Archivo Editar Ver Insertar Formato Herramientas Datos Ventana Ayuda

Arial 10

F1700 =E1700-E1699

	A	B	C	D	E	F	G
1	coordMagn_0	coordMagn_1	coordMagn_2	tiempo			
2	247	0	-76	1301683528332	0		
3	245	0	-77	1301683528376	44	44	
4	246	1	-76	1301683528387	55	11	
5	245	2	-77	1301683528402	70	15	
6	247	2	-76	1301683528430	98	28	
7	248	2	-76	1301683528450	118	20	
8	247	1	-77	1301683528471	139	21	
9	248	1	-76	1301683528491	159	20	
10	249	1	-76	1301683528510	178	19	
11	248	2	-75	1301683528531	199	21	
12	250	2	-75	1301683528573	241	42	
13	251	2	-75	1301683528593	261	20	
14	250	2	-77	1301683528610	278	17	
15	249	4	-75	1301683528631	299	21	
16	250	6	-70	1301683528653	321	22	
17	248	7	-70	1301683528673	341	20	
18	252	6	-67	1301683528691	359	18	
19	251	6	-70	1301683528711	379	20	
20	245	8	-73	1301683528743	411	32	
21	243	12	-71	1301683528771	439	28	
22	239	17	-69	1301683528782	450	11	
23	253	2	-69	1301683528799	467	17	
24	245	-3	-84	1301683528816	484	17	
25	252	-8	-82	1301683528836	504	20	
26	269	-166	-76	1301683528859	527	23	
27	222	179	-87	1301683528876	544	17	
28	243	0	-83	1301683528893	561	17	
29	246	0	-79	1301683528910	578	17	
30	245	-1	-79	1301683528932	600	22	
31	244	-1	-80	1301683528952	620	20	

parada\_lineal / ACELEROMETRO / ORIENTACION

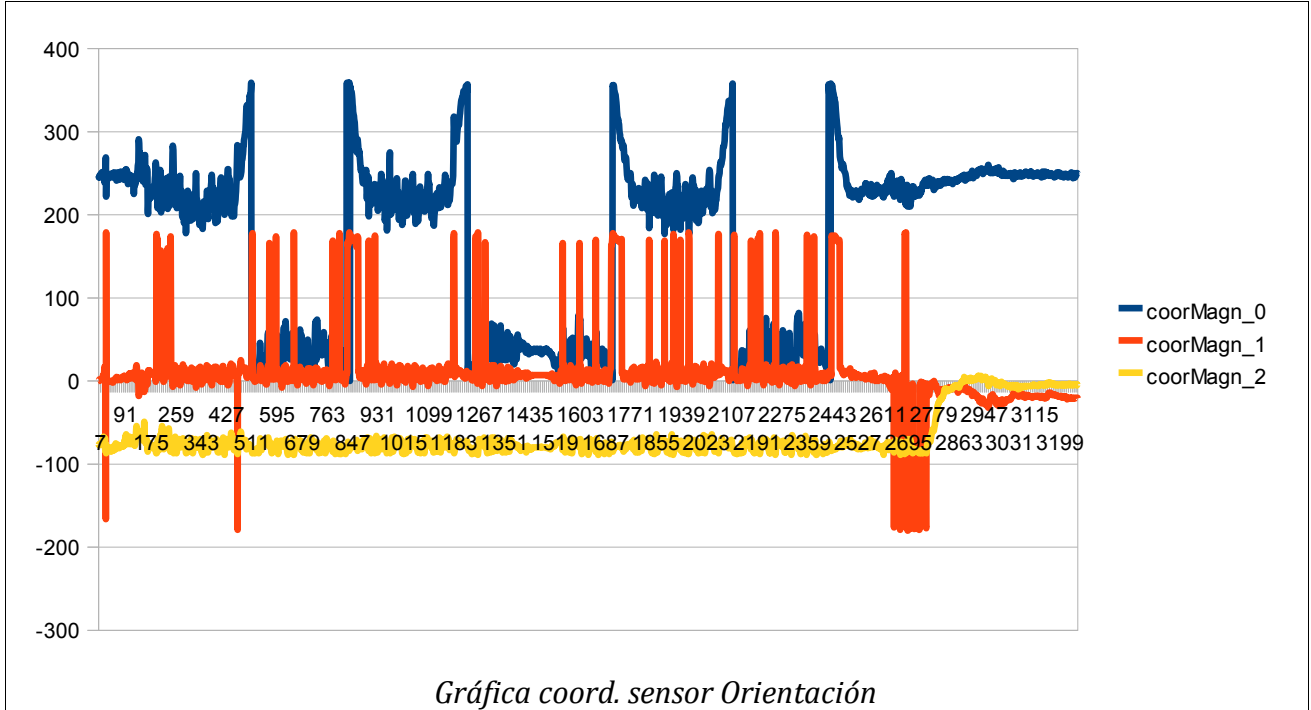
*Ejemplo de datos del sensor orientación*

La gráfica que corresponde es:

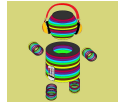


### Aplicación Android de Asistencia al Caminante

De las columnas A,B y C con respecto al número de línea (evento).







## 4.- Programa Caminante.

Este programa constituye el fin a lograr en este proyecto.

La pantalla que engloba las tres opciones posibles, Marcha, Anti Bloqueo y Anti Caída, así como sus botones de configuración es la siguiente:



En el *apéndice 4 Manual\_de\_uso\_Caminante* se explica cómo se utiliza.

### 4.1.- Análisis.

#### 4.1.1. Análisis general.

En principio no parece muy complicado el análisis de la aplicación, si nos abstraemos de la forma de detectar un bloqueo y una caída fruto del estudio de los datos obtenidos de Caminante\_DATOS. Desde el punto de vista del uso tenemos solamente tres opciones que nos tienen que dar como resultado tres salidas sonoras, pero desde la perspectiva de la programación según íbamos pensando se iba, más que complicando, que también, ampliando el análisis:

- Sobre la opción marcha. Esta ha sido la más sencilla, aprovechándome de las ventajas de utilizar software libre, he cogido una aplicación existente de un metrónomo y me he limitado a adaptarla y a cambiar el número de parámetros a configurar a dos y la pantalla de configuración convertirla en un cuadro de diálogo. En un principio no hubo mayor complicación. (el programa utilizado ha sido **Metronome de Akshat Aranya** bajo licencia GPL v3)

- Sobre la opción Anti Bloqueo. A esta opción como elemento independiente tampoco se le percibía mayor dificultad aunque aquí ya entraban los sensores y se planteó el dilema de



### Aplicación Android de Asistencia al Caminante

utilizar la opción definitiva del programa Caminante\_DATOS o usar la primera en la cual la parte de los sensores se le dejaba a un servicio, al final he optado por crear este.

– Pero ahora ya tenemos dos partes de programa que tienen que ir cada una por su lado, además de permitir funcionar a las aplicaciones del móvil, ¿qué hacer entonces?, me decido por crear hilos distintos (thread), además aun falta la tercera opción..

– Ya estamos en la tercera opción la Anti Caída. Primer análisis rápido, igual que el caso del bloqueo, su hilo, su servicio de sensores; ¡primer problema!, las opciones pueden y deben funcionar a la vez, ¿tendremos dos servicios escuchando eventos de sensores y dos hilos?, parece ser que una respuesta afirmativa no es lo más conveniente, optamos entonces por lo siguiente: cada vez que se lanza una de estas dos opciones si existe la otra funcionando se mata esta y se lanza con las dos un solo hilo que lanza un solo servicio que escucha una sola vez a los sensores cuyos valores se analizan en cada momento para ver si se produce un bloqueo o una caída. Dado que el detectar caídas es preferente con respecto al bloqueo, también lo es a la hora de lanzar los procesos de análisis de los eventos.

– Ahora ya tenemos las tres opciones listas para trabajar, pero ¿para parar?. La aplicación en su conjunto se cierra con el botón atrás del móvil y cada una de ellas en su botón respectivo de activar-parar, pero cuando se detecta un bloqueo o una caída ¿qué ocurre?.

- Caída, esto puede constituir un acontecimiento grave que se pretende evitar o mitigar con la emisión de un sonido por tanto esta opción se para una vez emitida la alarma. (se comentará más adelante la conveniencia de ampliar esta aplicación con el envío de una señal con las coordenada si ocurre este suceso)

- Bloqueo, en un principio se emitía un sonido cada cierto tiempo (configurable), los cinco primeros con volumen creciente y a partir del quinto al máximo y estos dejaban de emitirse en el momento que se detectaba de nuevo movimiento o se pulsaba el botón, actualmente por recomendación de Miguel Fernandez del Olmo hemos dejado dos opciones, la primera ya citada y que llamamos progresiva y la segunda que actúa como el caso de la parada y que llamamos brusca, estando estas opciones disponibles en el cuadro de diálogo de la configuración de esta opción.

#### 4.1.2.- Análisis de Bloqueo.

Cuando acometí este proyecto uno de las cuestiones que mas me preocupaba era la situación del móvil, el programa debía funcionar igual independientemente de su posición en el cuerpo y con respecto al suelo, para no exigir demasiados requisitos y me pasaba por la cabeza lo estudiado sobre curvas, superficies, planos osculadores, giros, etc. Durante bastante tiempo era una obsesión este tema, de echo en el programa Caminante\_DATOS se recomienda enviar la situación del teléfono.

Haciendo uso de una frase, cuyo autor en este momento no recuerdo, que dice: “Comprender no puede ser sustituido por una actividad intensa”, me decidí a revisar lo que tenía hasta ese momento y una cosa se centró en el pensamiento, la gravedad terrestre, el sensor aceleración



### Aplicación Android de Asistencia al Caminante

tiene en cuenta al vector gravedad, es decir tenemos un dato fijo en dirección y sentido y que su magnitud varia aunque poco de un lugar a otro de la tierra, por tanto cuando estamos parados la aceleración que nos mide el sensor es la gravedad, pero por desgracia tiene dirección y sentido (fijos respecto a la tierra), volvemos a la posición del móvil y de nuevo la frase anterior me aclaró el tema: me da igual como esté el móvil y cual sea el valor en sus ejes X, Y, Z, porque la raíz cuadrada de la suma de los cuadrados de los valores de las tres coordenada del vector aceleración, su módulo, cuando estamos parados tiene que ser el módulo de la gravedad que es un dato que sabemos. Por tanto **consideraremos que estamos parados** cuando durante una serie de eventos contiguos o durante los eventos que acontecen en un determinado tiempo los valores de los módulos de la aceleración del sensor y del de la gravedad se mueven dentro de un umbral fijado y que permitiremos sea configurable.

La anterior conclusión se vio corroborada por las gráficas de las hojas de cálculo como la que se mostró anteriormente, y por un programa en Java que leía los ficheros de datos y que se realizó para probar la implementación en código del análisis anterior.

#### 4.1.3.- Análisis de Caída.

Ya se habló de la dificultad de detectar una caída y no digamos ya de predecirla, dificultad que viene no solo por el echo de la variedad de situaciones que pueden provocarlas sino también por lo difícil que resulta tomar datos de caídas reales sean estas de personas sanas o enfermas e incluso la de simular las caídas.

Aunque bastante recientemente gracias al personal del laboratorio del Motor Control Group de Inef Galicia en la Universidade Da Coruña he conseguido datos de caídas simuladas hechas con arnés y de cuyo análisis he concluido que como primera aproximación utilizarnos también el módulo de la gravedad (en realidad el cuadrado) puesto que en la zona de caídas se ven en la gráfica unos picos pronunciados. Por tanto **consideraremos que tenemos una caída** cuando la pendiente de la curva, incremento del valor del cuadrado del módulo del vector aceleración dividido por el incremento del tiempo (a grosso modo, la derivada de la función), es mayor que un valor dado (configurable) durante un número dado de eventos contiguos (también configurable).

Al igual que en el caso del bloqueo un programa Java hecho al efecto comprobó una cierta bondad de la conclusión anterior en lo que respecta a la detección no a la previsión

Como se puede comprender este tema queda abierto y se antoja como elementos a estudiar para conseguir acercarnos a la realidad de una caída el uso del sensor orientación, que al final no hemos utilizado, o mejor los valores de la matiz de rotación y nuevos sensores como el giroscopio, pero lo que es más importante es la obtención de datos reales.

#### 4.2.- Desarrollo.

El desarrollo del programa ha sido progresivo lo que de alguna manera he intentado hacer ver con la exposición hecha en el análisis.

A diferencia de lo hecho con el programa Caminante\_DATOS en el cual nos centramos en mostrar el código, ahora para este nos centraremos en diagramas. Estos no son muy ortodoxos



### Aplicación Android de Asistencia al Caminante

fundamentalmente por el echo de la falta de experiencia en programación orientada a objetos y por tanto en la forma de plasmar en él lo que hace el programa.

Hemos considerado la posibilidad de utilizar herramientas de ayuda para la generación de la documentación, pero sumando los problemas que he tenido a la hora de la instalación de estas y la necesidad de aprendizaje he optado por la realización manual de esta que aporta la ventaja de que se plantean muchas dudas que ayudan a mejorar el conocimiento del sistema que hemos empleado. Dejamos el uso de estos programas para futuros desarrollos.

Se han omitido los diagramas de las clases Dialog debido a su sencillez, la simple visión del código es suficientemente aclaradora aunque sea en el configuración del bloqueo que es el más amplio en el número de botones.

Tampoco se muestran los diagramas de las clases referidas al sonido como la MediaPlayerPool y la TickPlayer por no ser hechas por mi y la SobresaltoMediaPlayer por su sencillez.

#### 4.2.1.- Diagrama de clases.

Más que un diagrama de clases habría que hablar de diagrama de las clases puesto que es una muestra de las clases que existen, que extienden e implementan y las relaciones entre ellas..

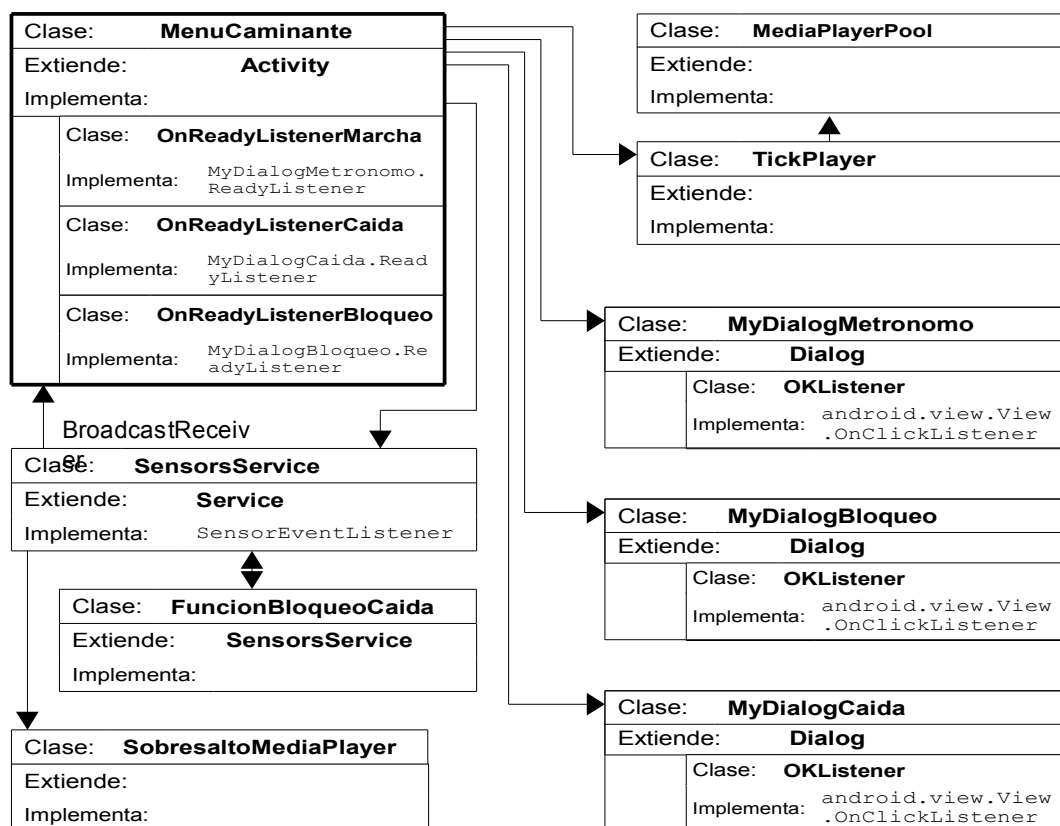


Diagrama de Clases



## Aplicación Android de Asistencia al Caminante

### 4.2.2.- Diagrama de la clase MenuCaminante.

Este es posiblemente el diagrama menos ortodoxo, se muestran los módulos más importantes y las relaciones con mayor significado desde el punto de vista de la ejecución, que van de color rojo. Como peculiaridad hacer constar el hecho de que los dibujos de los rombos que representan comparaciones (if) ejecutan las instrucciones que los acompañan si la respuesta es verdadera.

(se muestra en las páginas siguientes)

### 4.2.3.- Diagrama de Clase de SensorsService.

Este diagrama ya empieza a tener elementos más concisos y los rombos de comparación ya se acompañan de la afirmación y de la negación.

(se muestra en las páginas siguientes)

### 4.2.4.- Diagrama Clase FuncionBloqueoCaida.

De todo el desarrollo de esta aplicación lo que viene a continuación ha constituido, para mí, lo más bonito e interesante, quizás porque lleve elementos de programación más clásica o porque consiste en plasmar lo más personal.

Comentar aquí que se ha mantenido en algunas partes la referencia al sensor orientación pese a no trabajar con los datos de este, el motivo es por si se desea implementar su uso en el apartado de caída resulte más llevadero.

Del esquema de clase se deduce que es SensorsService quien se relaciona con la clase SobresaltoMediaPlayer cuando en principio lo hacía desde aquí, he mantenido en el código como comentarios por si se quiere volver hacia atrás.

(se muestra en las páginas siguientes)

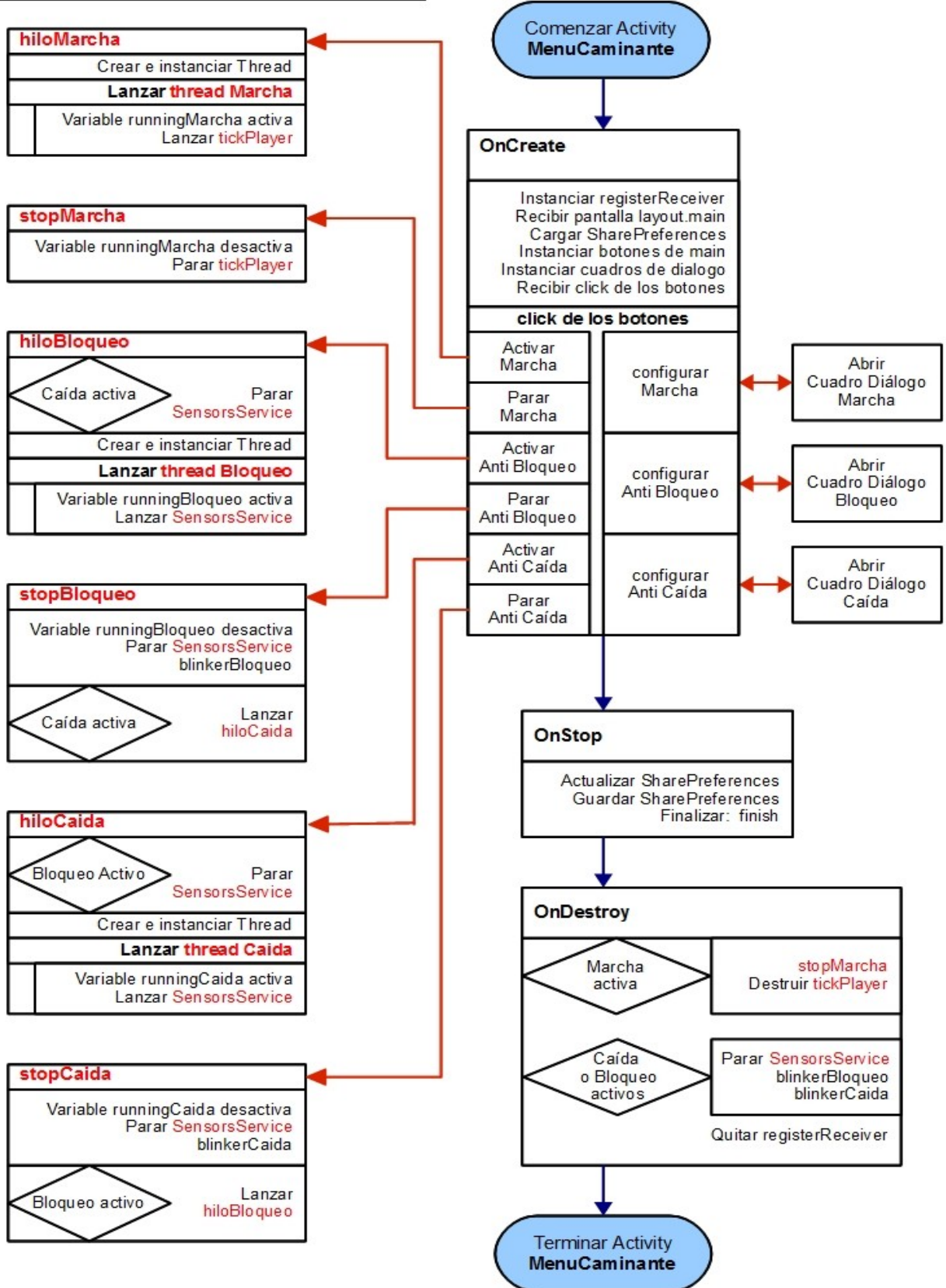
## 4.3.- Código.

El código correspondiente se encuentra en los ficheros adjuntos, en concreto en el proyecto Caminante de eclipse.



Aplicación Android de Asistencia al Caminante

Clase Activity MenuCaminante









Aplicación Android de Asistencia al Caminante

**Clase MenuCaminante**

(Continuación)

<p><b>Clase OnReadyListenerMarcha</b> implementa MyDialogMetronomo.ReadyListener</p>
<b>ready</b>
Variables: Periodo Tiempo

<p><b>Clase OnReadyListenerBloqueo</b> implementa MyDialogBloqueo.ReadyListener</p>
<b>ready</b>
Variables: Gravedad Modulo Evento Bloqueo Tiempo Acción Bloqueo

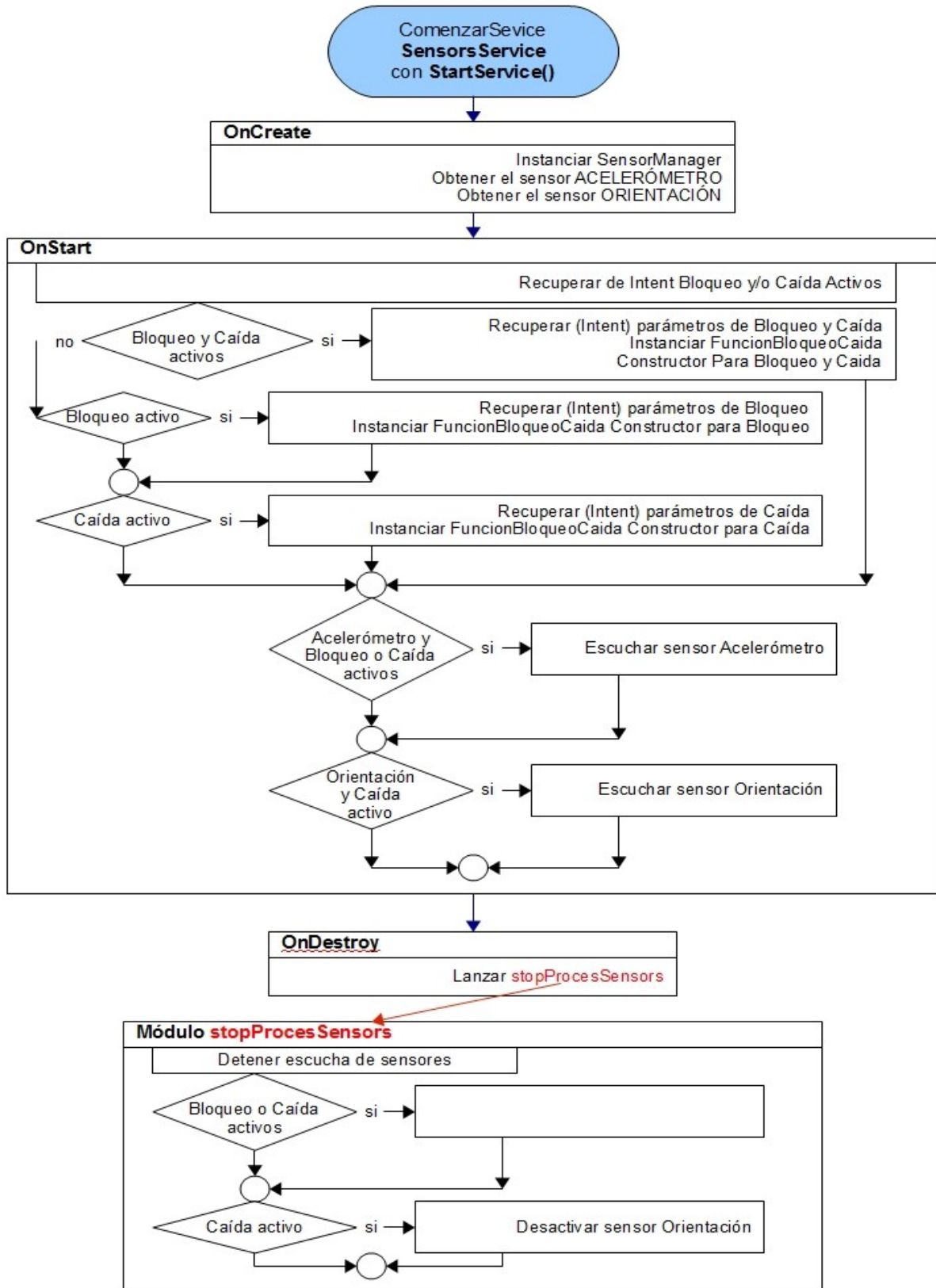
<b>BroadcastReceiver</b>		
<b>onReceiver</b>		
	si	
<table border="1"> <tr> <td style="padding: 5px;">Desactivar botón Caída Lanzar <b>stopCaída</b></td> </tr> </table>		Desactivar botón Caída Lanzar <b>stopCaída</b>
Desactivar botón Caída Lanzar <b>stopCaída</b>		
	si	
<table border="1"> <tr> <td style="padding: 5px;">Desactivar botón Bloqueo Lanzar <b>stopBloqueo</b></td> </tr> </table>		Desactivar botón Bloqueo Lanzar <b>stopBloqueo</b>
Desactivar botón Bloqueo Lanzar <b>stopBloqueo</b>		

<p><b>Clase OnReadyListenerCaída</b> implementa MyDialogCaída.ReadyListener</p>
<b>ready</b>
Variables: Eventos Pendiente



Aplicación Android de Asistencia al Caminante

Clase Service **SensorsService** implementa **SensorEventListener**



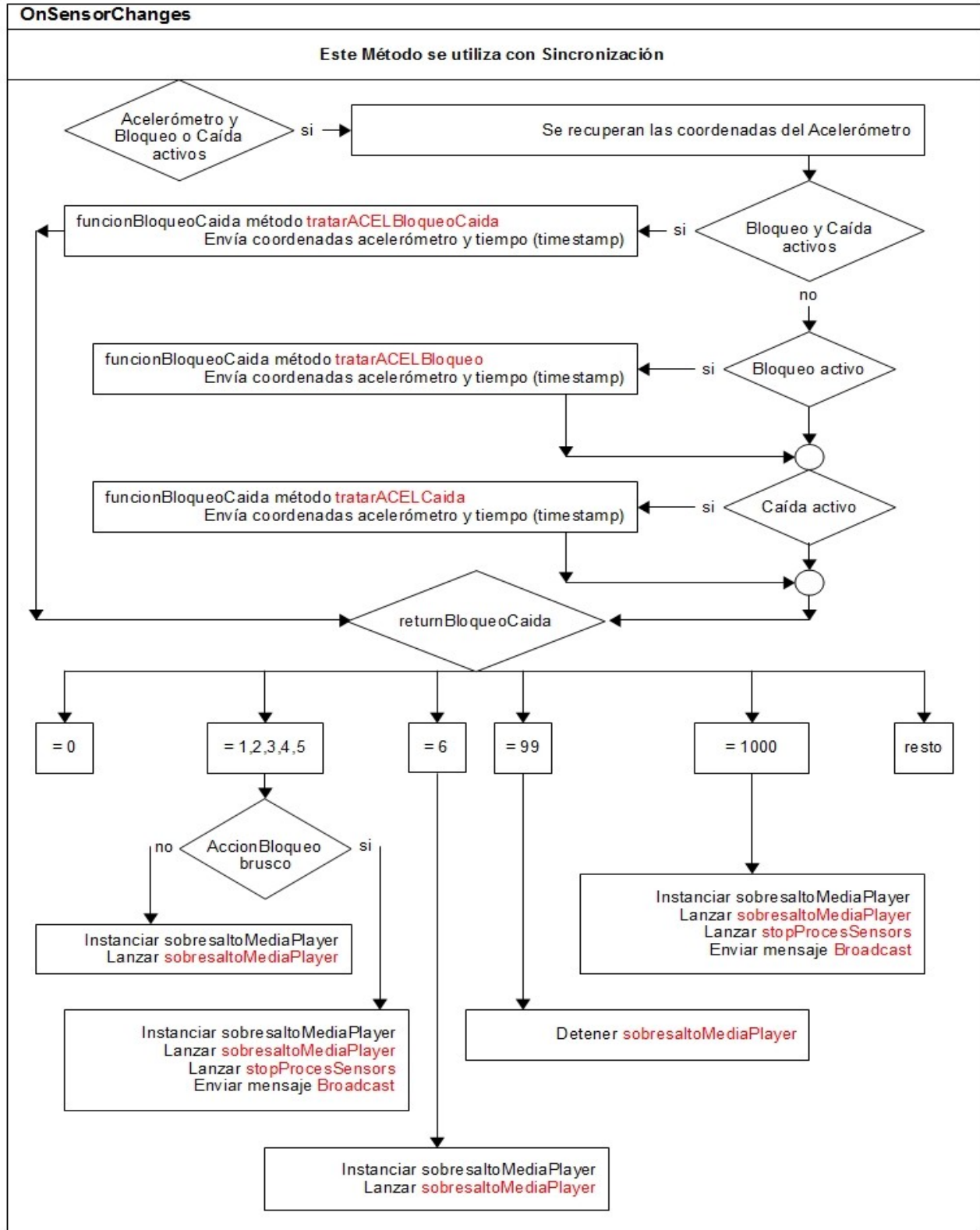




Aplicación Android de Asistencia al Caminante

Clase Service **SensorsService**

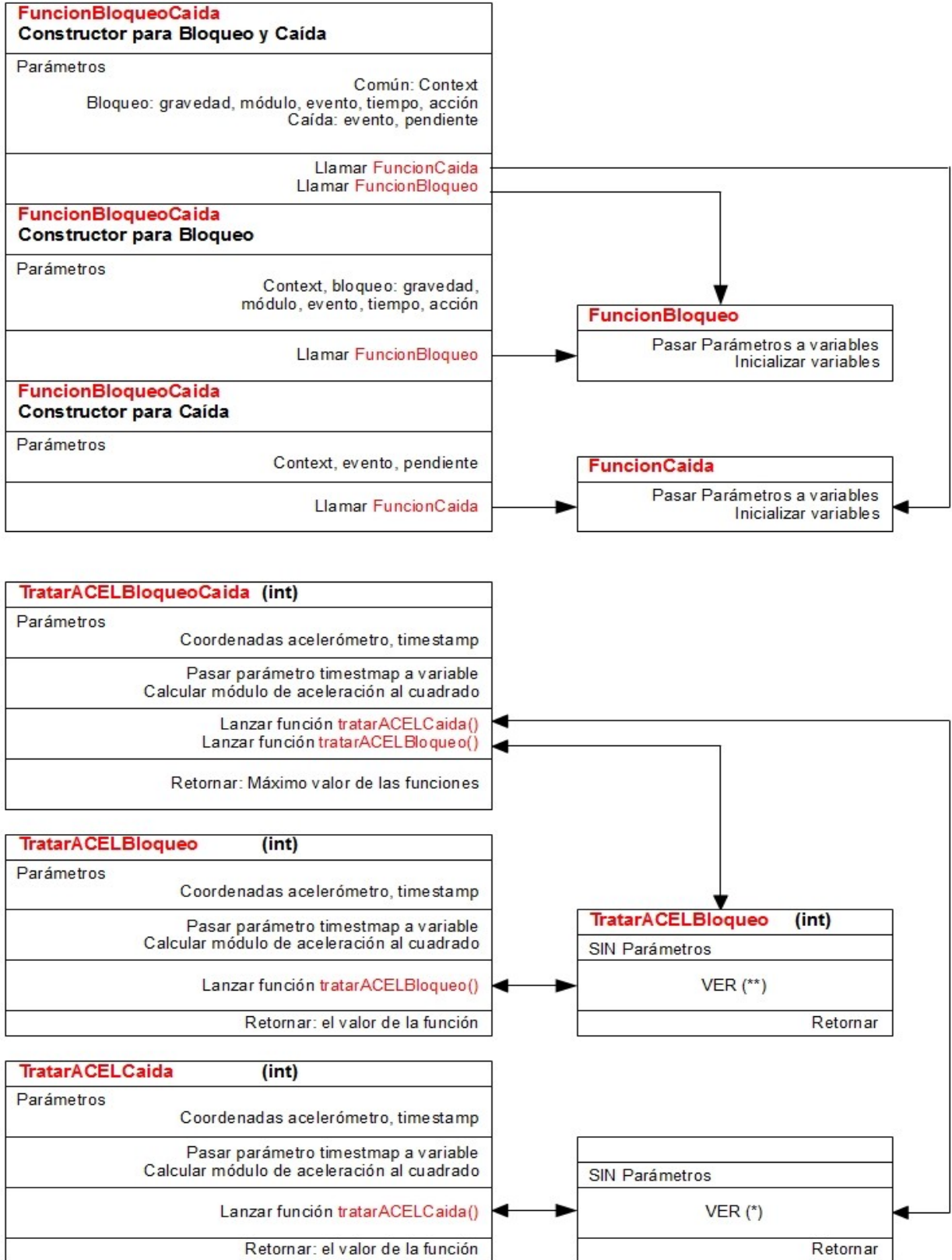
(Continuación)





Aplicación Android de Asistencia al Caminante

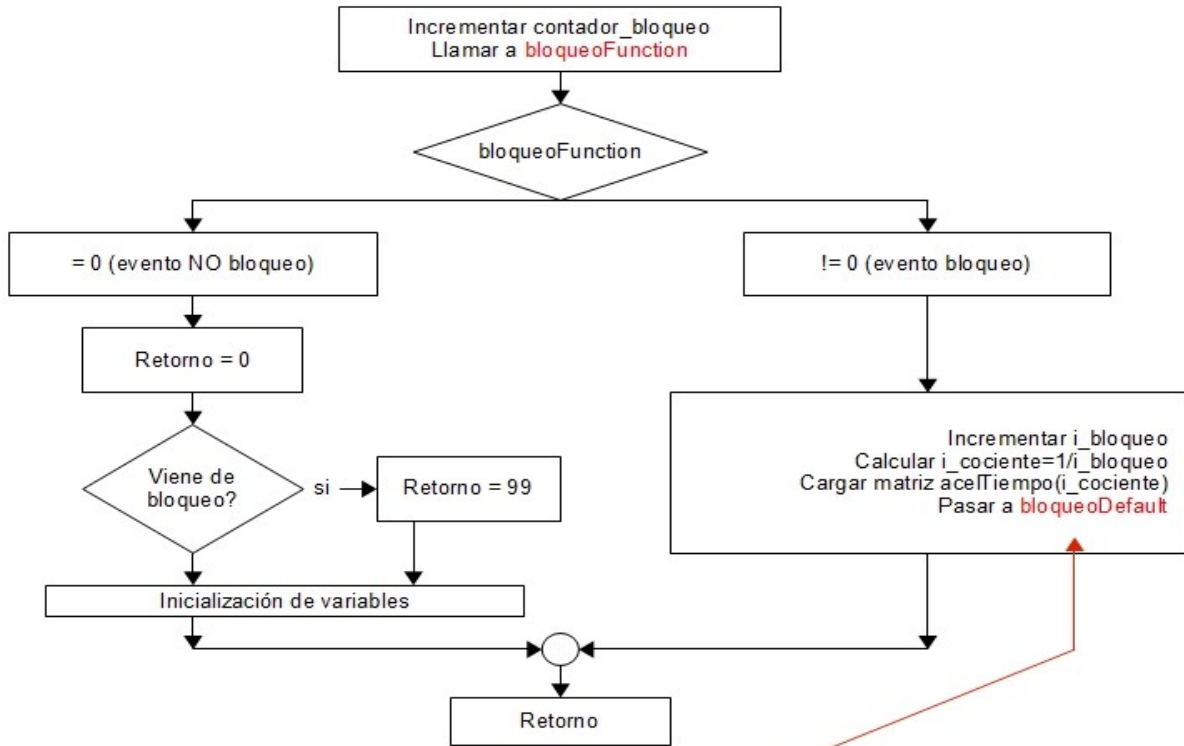
Clase **FuncionBloqueoCaída** extiende **SensorsService**



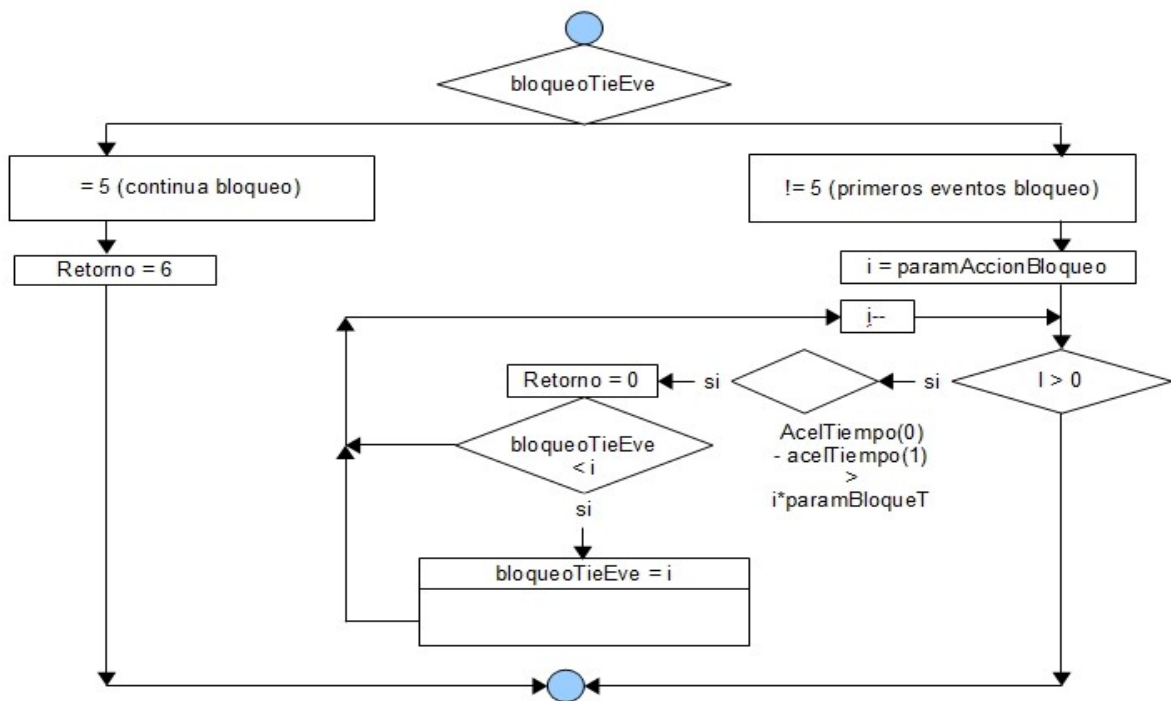


Aplicación Android de Asistencia al Caminante

(\*) Módulo **tratarACELBloqueo()**



Módulo **bloqueoDefault**

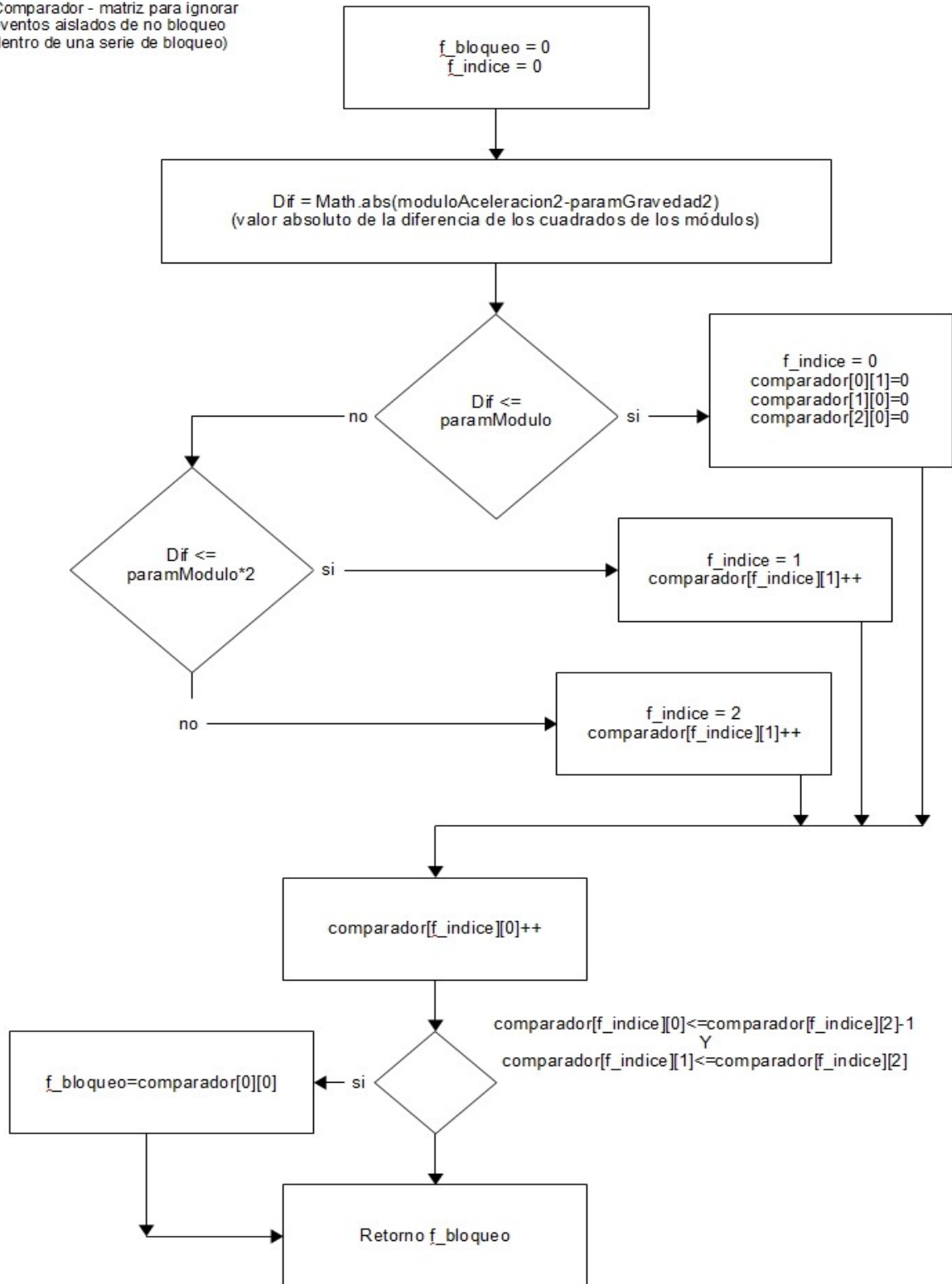




Aplicación Android de Asistencia al Caminante

Módulo **bloqueoFunction**

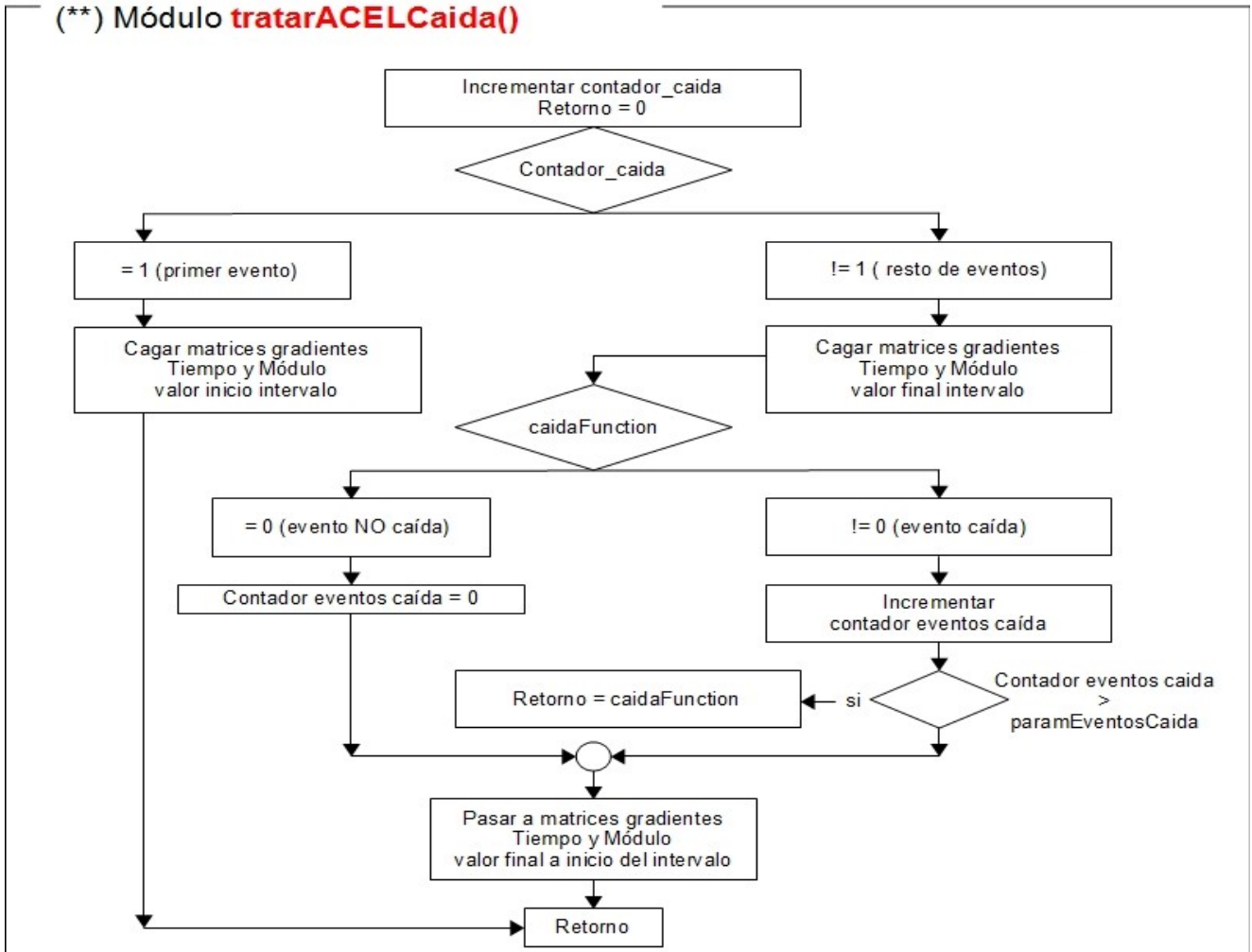
(Comparador - matriz para ignorar eventos aislados de no bloqueo dentro de una serie de bloqueo)



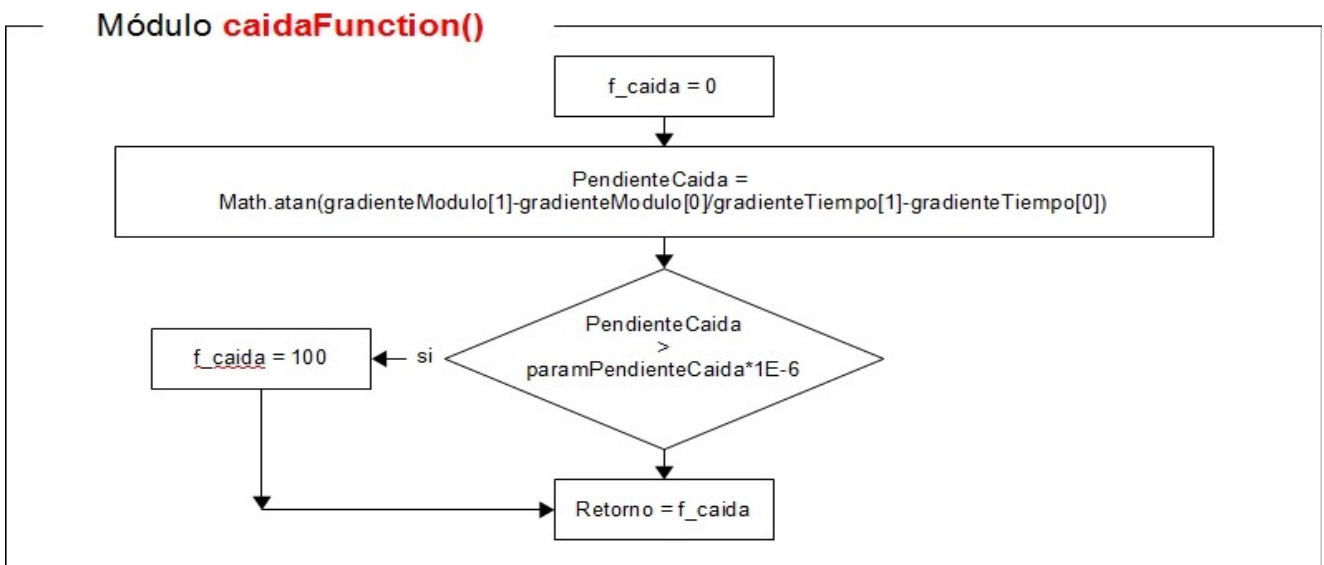


Aplicación Android de Asistencia al Caminante

(\*\*) Módulo **tratarACELCaida()**



Módulo **caidaFunction()**





## 5.- Resultados.

Es imposible presentar en este documento los resultados porque son sonoros pero puedo decir que:

- la opción de marcha funciona sin problemas, como ya se señaló se dejaron solamente dos campos de configuración, periodo y tiempo, quedando el manejo de volumen para los controles del móvil. Por recomendación de un miembro del Motor Control Group cambié los sonidos originales por otros mas cortos que parece ser son más efectivos a la hora de marcar el ritmo de un caminante.

- en lo que respecta al bloqueo o parada, en las pruebas que se hicieron parece ser que funciona de manera aceptable, pese a la sencillez conceptual de su detección. En este caso tenemos además la ventaja de que se puede jugar con varios parámetros de configuración para personalizar y buscar el resultado optimo. Como se comenta en el manual de uso el solicitar el parámetro brusco detiene la opción después de emitir el sonido, no así la opción progresivo que emite los sonidos a periodos regulares volviendo el silencio de manera automática al detectarse movimiento. Aquí el controlar el volumen no supone ningún beneficio porque lo que se busca es el sobresalto por tanto el máximo se controla pero internamente.

- Sobre el tema de las caídas ya hemos hablado y el resultado esta por ver, funciona al mover bruscamente el móvil en sentido vertical pero habría que jugar con varias simulaciones para ajustar los parámetros lo mejor posible. Emitido el sonido el resultado es similar al caso de bloqueo brusco. Aquí las posibilidades de mejora son notables, quizás de manera paralela a la dificultad de conseguir solamente detectar la caída, no digamos ya si intentamos tratar la prevención.

Como resultado no final tenemos que citar el programa Caminante\_DATOS que si bien no formaba parte del proyecto en si ha constituido un hito importante por varios motivos, algunos ya comentados, como ser el primer contacto con el mundo Android en general y con los sensores en particular y otros como la obtención de los datos que me han permitido detectar las pautas que dieron lugar a lo implementado para los bloqueos y las caídas, además queda ahí para poder obtener más valores de los sensores que utiliza y con un código, creo que, fácilmente modificable y ampliable para incluir otros sensores.





## 6.- Conclusiones.

A la pregunta de si el programa que alberga este proyecto alcanzó al cien por cien lo previsto evidentemente hay que contestar que no, pero de alguna manera sobre todo en la parte más que comentada de las caídas era de esperar. No se trata de un proyecto habitual de gestión en el cual las posibilidades están acotadas, ni tan siquiera algunos para móviles en los cuales se juega con datos reales en comparación con circunstancias estáticas, en el caso que nos ocupa tenemos la dificultad de obtener datos reales y lo que le da una complejidad quizás mayor, que entra en juego el factor humano en su vertiente personal, cada individuo es diferente y el entorno en que se mueve también.

A la pregunta de si este proyecto sirve para algo, aquí tengo que decir rotundamente que si, visto desde una perspectiva objetiva como subjetiva. Quizás el reflejo de sobresalto no se consiga siempre y para todos los casos, pero estamos en un proyecto de software libre y el código queda ahí, en concreto quedará en el repositorio de IrisLibre de Red Iris, para que cualquiera lo pueda ejecutar, copiar o modificar. Esta memoria también creo y espero que sirva para que otras personas que se quieran meter con Android puedan aclarar temas que a mi me costó bastante asimilar y que a mi manera e intentado aclarar.

### 6.1.- Conclusiones subjetivas.

Desde el punto de vista personal el proyecto ha sido sumamente positivo por los motivos que expondré a continuación:

- Incursión en el mundo de los móviles. Hasta que no acometí este trabajo, los móviles o incluso los smartphones eran aparatos que servían para llamar por teléfono, bueno estos últimos ya permitían muchas más cosas, pero era un mundo ajeno y aquí tuve las primera dificultades al querer utilizar el HTC Magic de mi mujer. Cuando me planteé su uso, en parte motivado porque el simulador de eclipse no iba en mi antiguo portátil y el de mi hijo lo tenía de forma esporádica, lo primero que me propuse fue hacer una salva de seguridad, lo conseguí después de un par de sustos con la pantalla roja, pero me sirvió para quitar esa especie de miedo que uno siente cuando se enfrenta a algo nuevo agravado en este caso al utilizar una herramienta que no era mía.

- Nueva experiencia con la programación orientada a objetos. No es mi fuerte en materia de programación pero creo que al final hemos avanzado en este campo, de echo las últimas modificaciones en el código han sido mucho más eficaces.

- Dar los primeros pasos en el uso de repositorios y control de versiones. En esto no se puede decir que me haya adentrado demasiado, en realidad hasta ahora mi única experiencia ha sido subir a IrisLibre el programa CaminanteDATOS, y no he utilizado el control de versiones. El motivo de este no uso es la falta de experiencia y el miedo a perder demasiado tiempo toda vez que dar esos primeros pasos han sido bastante dificultosos para mí, el caso positivo es que la semilla ya está echada y una vez subida el total de la aplicación y ya sin ese temor a perder el



## Aplicación Android de Asistencia al Caminante

tiempo espero hacer uso de esas herramientas si como tengo previsto continuo con el desarrollo del programa.

- Conocer personas que se dedican a la investigación (Motor Control Group) buscando elementos que sirvan de ayuda a pacientes con enfermedades como el Parkinson que además de hacer bien su trabajo lo hacen de forma solidaria y agradable.
- Conocer gente como Jorge Fernandez de libsoft que de manera totalmente desinteresado ha atendido mis consultas sobre Android pese a su abundante trabajo.

### 6.2.- Dificultades.

Hemos encontrado varias dificultades también algunas objetivas y otras subjetivas: dentro de las objetivas tenemos a Android que todo parece indicar que tiene una curva de aprendizaje con bastante pendiente al principio, a lo que hay que sumar la poca, aunque cada vez mayor, existencia de ejemplos de código de los propios desarrolladores de Android. Por otro lado la ya reiterada dificultad de obtener datos de caídas reales.

Dentro de las subjetivas, aparte de la ya citada falta de experiencia en campos concretos hay una que para este tipo de actividades es bastante crítica y es la falta de dominio del inglés. Actualmente existen números traductores automáticos pero los resultados son francamente malos.

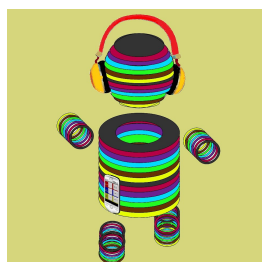
### 6.3.- Ampliaciones y mejoras.

Dentro de las ampliaciones que se podrían hacer a este programa sería el emitir un mensaje o una llamada de advertencia si se ha detectado una caída enviando las coordenadas de posición conseguidas a través del GPS o incluso la calle y el número.

Dentro de las mejoras, quitando las consabidas caídas, el tratamiento del sonido posiblemente sea el campo a explorar para conseguir eficazmente el reflejo de sobresalto.

### 6.4.- Icono.

Para que aparezca en el móvil de forma diferenciada, hemos creado un icono de un muñeco que porta cascos y un HTC Magic. Los aros de colores tienen su simbología representando el temblor que constituye uno de los síntomas habituales y más visibles de los pacientes con Parkinson. Como distintivo he optado por ponerlo en la cabecera de cada página.







## 7.- Instalación de los programas.

Los ejecutables de Android llevan al extensión apk y si se ha utilizado eclipse para generar la aplicación este fichero se encuentra en el subdirectorio bin del directorio de la aplicación, en nuestro caso. Workspace-> Caminante → bin ->Caminante.apk.

Si este programa no se encuentra en el Market, como en estos momentos es el caso, la manera más sencilla de cargarlo es mandar un correo al móvil en el que se desea ejecutar la aplicación con el apk adjunto, el móvil te pregunta si deseas instalar el programa pero hay que tener activado la opción orígenes desconocidos si, como es nuestro caso, el programa no esta firmado: Ajustes > Aplicaciones > Orígenes desconocidos.

Si se tiene eclipse con el plugging de Android y el proyecto, simplemente conectando el móvil al pc y solicitando en eclipse la ejecución de este, ya se carga en el móvil.

También se puede pasar a la tarjeta de memoria el apk y con un navegador buscarlo y activarlo.

## 8. Bibliografía.

Como no podía ser de otra manera actualmente la mayor bibliografía es Internet en su conjunto, particularmente a las ya citadas páginas de android-spa y la oficial de los desarrolladores de Android añadiré la bitácora de Javier Cancela sobre todo al comienzo (<http://javiercancela.com/>).

En cuanto a código:

<https://svn.forge.morfeo-project.org/libregeosocial/> de Jorge Fernández

Metronome de Akshat Aranya

Libros y publicaciones:

Java Vademecum de José A. Mañas.

Localización de dispositivos móviles para redes sociales dinámicas de Rúl Román López