Grado en Ingeniería Informática

Trabajo de Final de Grado

Posicionamiento en interiores mediante balizas bluetooth de bajo consumo y sensores inerciales

Autor:
Jaime Higueras Ruiz

Supervisor:
Joaquín Torres Sospedra
Tutor académico:
Pedro García Sevilla

Fecha de lectura: 13 de octubre de 2020
Curso académico 2019/2020

Resumen

En este documento se detalla el proceso de creación de un sistema de posicionamiento en interiores basado en la combinación de un sistema que aprovecha los sensores inerciales de un dispositivo Android y un sistema que requiere un despliegue de balizas bluetooth de bajo consumo.

Este proyecto se ha realizado durante la estancia en prácticas de Ingeniería Informática de la Universitat Jaume I en la empresa UBIK Geospatial Solutions S.L.

El sistema de posicionamiento combinado busca no solo dar una solución eficiente al problema del posicionamiento en interiores, sino mejorar los resultados al combinar dos sistemas cuyos principales problemas pueden resolverse mutuamente.

Palabras clave

Posicionamiento en interiores, sensor inercial, filtro, Kalman, bluetooth

Keywords

Indoor positioning, inertial sensor, filter, Kalman, bluetooth

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto que se presenta en este documento se ha desarrollado durante la estancia en prácticas del grado Ingeniería Informática de la Universitat Jaume I (UJI), en la empresa Ubik Geospatial Solutions S.L. [1]. Se trata de una empresa ubicada en el edificio Espaitec 2, el cual se encuentra en el Parque Científico, Tecnológico y Empresarial de la UJI. Esta empresa se formó a partir del grupo de investigación universitario GEOTEC debido a la necesidad de ofrecer una salida a nivel de mercado a su desarrollo de software.

Ubik es una empresa que se dedica a la creación de aplicaciones para dispositivos móviles y web que requieren una integración de información geoespacial. La empresa utiliza esta información para desarrollar aplicaciones de *SmartCities*, *Web Mapping* y aplicaciones basadas en la localización del usuario como, por ejemplo, *GeoGames*.

Ubik considera que la información geoespacial tiene un gran valor para aquellas empresas y organizaciones que pueden aprovechar la localización de sus usuarios para poder llevar un mejor seguimiento del resultado de sus acciones y estrategias, ya que esto puede afectar de forma positiva a su toma de decisiones.

Además, la información geoespacial también puede ser muy útil para los usuarios, ya que pueden obtener información precisa de su ubicación en tiempo real. Esto es muy útil en entornos amplios, como un almacén o una biblioteca, en los que se pretende buscar algo muy concreto, como un producto o un libro. En este punto entra la localización en interiores, ya que se necesita un nivel de precisión que no podemos obtener mediante el sistema GPS.

Así pues, en este proyecto se ha desarrollado un programa para calcular y gestionar el posicionamiento en interiores de forma precisa mediante el uso de distintos algoritmos y filtros. Este programa está pensado para funcionar sobre dispositivos Android. Aunque el objetivo de este sistema era facilitar a los usuarios de la biblioteca de la UJI un posicionamiento preciso dentro de la misma a la hora de buscar libros mediante la aplicación 'Biblioteca Universitat

Jaume I' [2] proporcionada por la empresa, debido a la situación de emergencia sanitaria ha sido imposible realizar las pruebas, por lo que se ha tenido que probar en un entorno alternativo.

La motivación del desarrollo de este sistema reside en su gran utilidad, no únicamente a los usuarios de la biblioteca, sino a cualquier usuario que pueda necesitar un alto nivel de precisión ya sea por motivos de ocio, trabajo o salud. En el caso de la salud, hay estudios que afirman que ciertos patrones en el movimiento de las personas, sobre todo personas mayores, puede ayudar a detectar casos de demencia o alzheimer [3].

Otra motivación es conseguir una reducción del coste computacional de los algoritmos y filtros que se van a utilizar para el desarrollo del sistema de posicionamiento. Puesto que este sistema está pensado para funcionar sobre un dispositivo Android, es importante conseguir un coste computacional bajo, ya que para estos dispositivos, a pesar de su constante evolución a nivel de capacidad de cómputo, sigue teniendo gran importancia la reducción de este coste con el objetivo de maximizar su eficiencia.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto consiste en el desarrollo de un sistema de posicionamiento en interiores multi-sensor para su implementación en una aplicación Android. El objetivo principal se puede desglosar en los siguientes subobjetivos:

- Desarrollo de un algoritmo de posicionamiento basado en los sensores inerciales de un dispositivo Android con restricciones de cómputo.
- Desarrollo de un algoritmo centroide ponderado (ver Sección 5.2.2) para la estimación de posiciones mediante las señales de balizas bluetooth de bajo consumo (BLE) recibidas en dispositivos móviles.
- Desarrollo de un algoritmo de navegación basado en la integración de sistemas de posicionamiento en interiores disponibles mediante la técnica del filtro de partículas o filtro de Kalman.
- Validación del sistema en 2 escenarios con diferentes configuraciones de balizas BLE.

1.3. Alcance del proyecto

En esta sección se detalla el alcance del proyecto, es decir, se explica con exactitud tanto aquello que se debe desarrollar para realizar el proyecto, como aquello que no entra dentro de los planes de desarrollo o que no forma parte del proyecto en si.

Con lo que respecta al alcance del proyecto, este comprende la creación y optimización de un sistema de posicionamiento basado en la combinación de sensores inerciales y la tecnología bluetooth de bajo consumo (BLE). Esto no implica la creación de una aplicación donde se

ejecute este sistema, puesto que está pensado para funcionar a modo de librería para que otras aplicaciones hagan uso de este sistema.

Por lo tanto, el proceso de desarrollo del proyecto comprende lo siguiente:

- El análisis y estudio de las tecnologías a utilizar.
- El desarrollo de dos sistemas de posicionamiento (con sensores inerciales y con BLE).
- La combinación y optimización de ambos sistemas de posicionamiento.

Por otra parte, no entra en el proceso de desarrollo del proyecto, y por tanto no forma parte del alcance del proyecto:

- La creación de una aplicación Android que implemente el sistema de posicionamiento.
- La creación de un algoritmo o filtro innovador para optimizar el sistema.

Con respecto al alcance funcional, el sistema debe ser capaz de:

- Ubicar al usuario utilizando únicamente el sistema basado en sensores inerciales.
- Ubicar al usuario utilizando únicamente el sistema basado en balizas BLE.
- Ubicar al usuario utilizando ambos sistemas.

1.4. Estructura de la memoria

En esta sección se detalla la estructura de la memoria a partir de este punto. En el Capítulo 2 se muestra la metodología, planificación, la estimación de recursos y costes y el seguimiento que se ha realizado del proyecto. En el Capítulo 3, se realiza una introducción teórica a las principales tecnologías que se emplean en el proyecto, las cuales son los sensores inerciales, las balizas BLE y el filtro de Kalman. En el Capítulo 4 se desarrolla el análisis y diseño del sistema, incluyendo descripción de requisitos y distintos diagramas. En el Capítulo 5, se explica la implementación del código necesario para el desarrollo del proyecto, indicando los patrones de diseño, problemas encontrados y sus soluciones y explicando las decisiones tomadas. Finalmente, en el Capítulo 6 se encuentra una breve conclusión y una propuesta de continuación del proyecto.

Capítulo 2

Planificación y seguimiento del proyecto

2.1. Metodología

Ubik dio total libertad a la hora de seleccionar la metodología de trabajo que mejor funcionase para el desarrollo del proyecto. Puesto que se tenía una idea clara del resultado final del proyecto, se optó por seguir una metodología tradicional o predictiva. En concreto se ha empleado el modelo de desarrollo en cascada.

2.2. Planificación

A continuación, se enumeran las tareas identificadas para la planificación del proyecto junto con su duración estimada:

- Planificación del proyecto (Total: 24h)
  - Desarrollo de la propuesta técnica (10h)
  - Revisión de la propuesta técnica (14h)
  - Entrega de la propuesta técnica (0h)
- Investigación (Total: 10h)
  - Búsqueda de información de conceptos clave (4h)
  - Búsqueda de información sobre las tecnologías a utilizar (6h)
- Aprendizaje de las tecnologías (Total: 26h)
  - Aprendizaje de recepción de datos de balizas BLE (5h)
  - Recogida de datos de sensores inerciales (6h)

    - Aprendizaje de filtro de Kalman (15h)
- Desarrollo del proyecto (Total: 240h)
    - Creación de tests
        - Creación de tests unitarios (15h)
        - Creación de tests de integración (15h)
    - Implementación de algoritmos
        - Implementación de algoritmo de posicionamiento basado en sensores inerciales (50h)
        - Optimización del coste computacional del algoritmo de posicionamiento inercial (12h)
        - Implementación de algoritmo de posicionamiento basado en balizas BLE (30h)
        - Algoritmo de navegación con filtro de Kalman (33h)
        - Mejora del algoritmo del centroide ponderado (24h)
    - Validación
        - Validación del algoritmo en el entorno: Espaitec 2 (15h)
        - Validación del algoritmo en el entorno: Biblioteca UJI (6h)
    - Integración
        - Creación de la librería con el sistema de posicionamiento (30h)
        - Integración de la librería en la aplicación *Biblioteca UJI* (10h)
- Fin del proyecto (0h)
- Desarrollo de la memoria del TFG (Total: 150h)
    - Redacción de informes quincenales (10h)
    - Redacción de la memoria del TFG (90 horas)
    - Revisión de la memoria del TFG (30 horas)
    - Preparación de la presentación y demo (20 horas)

En la Figura 2.1 se presenta la organización de las tareas junto con su estructura temporal mediante un diagrama de Gantt. A pesar de existir tareas que se ejecutan en paralelo, y que por tanto, pueden ser ejecutadas a la vez en caso de disponer de un equipo con varios miembros, puesto que no es el caso, se han ejecutado en el mismo periodo de tiempo de forma que la duración en horas de cada tarea simplemente se suma, obteniendo como resultado la misma duración que si se hubiesen ejecutado en serie. Para diferenciar estas acciones en paralelo, se han asignado distintos roles para poder distinguir, en caso de disponer de un equipo, que miembro realizaría cada tarea.

2.3. Estimación de recursos y costes del proyecto

En esta sección se realiza una estimación de los costes de los distintos recursos que serán necesarios para la realización del proyecto. Estos recursos se dividen en recursos humanos, recursos de software y recursos de hardware, además de los costes indirectos.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Posicionamiento en interiores con balizas BLE**	**300 horas**	**lun 17/02/20**	**vie 29/05/20**
2	**Planificación**	**24 horas**	**lun 17/02/20**	**jue 27/02/20**
3	Desarrollo de la propuesta técnica	10 horas	lun 17/02/20	jue 20/02/20
4	Revisión de la propuesta técnica	14 horas	vie 21/02/20	jue 27/02/20
5	Entrega de la propuesta técnica	0 horas	jue 27/02/20	jue 27/02/20
6	**Investigación**	**10 horas**	**lun 17/02/20**	**vie 21/02/20**
7	Búsqueda de información de conceptos clave	4 horas	lun 17/02/20	mar 18/02/20
8	Búsqueda de información de las tecnologías a utilizar	6 horas	mié 19/02/20	vie 21/02/20
9	**Aprendizaje de las tecnologías**	**26 horas**	**lun 24/02/20**	**jue 05/03/20**
10	Aprendizaje de recepción de datos de balizas BLE	5 horas	lun 24/02/20	mié 26/02/20
11	Recogida de datos de sensores inerciales	6 horas	mié 26/02/20	lun 02/03/20
12	Aprendizaje de filtro de Kalman	15 horas	lun 02/03/20	jue 05/03/20
13	**Desarrollo del proyecto**	**240 horas**	**vie 06/03/20**	**vie 29/05/20**
14	**Creación de tests**	**30 horas**	**vie 06/03/20**	**vie 13/03/20**
15	Creación de tests unitarios	15 horas	vie 06/03/20	mié 11/03/20
16	Creación de tests de integración	15 horas	mié 11/03/20	vie 13/03/20
17	**Implementación de algoritmos**	**149 horas**	**mar 17/03/20**	**lun 18/05/20**
18	Algoritmo de posicionamiento basado en sensores inerciales	50 horas	mar 17/03/20	jue 02/04/20
19	Optimizar coste computacional del algoritmo	12 horas	jue 02/04/20	mar 07/04/20
20	Implementación de algoritmo de posicionamiento basado en balizas BLE	30 horas	mié 08/04/20	lun 27/04/20
21	Algoritmo de navegación con filtro de Kalman	33 horas	mar 28/04/20	lun 11/05/20
22	Mejora del algoritmo Weighted Centroid	24 horas	mar 12/05/20	lun 18/05/20
23	**Validación**	**21 horas**	**jue 02/04/20**	**lun 11/05/20**
24	Validación del algoritmo en el entorno ESPAITEC	15 horas	jue 02/04/20	jue 30/04/20
25	Validación del algoritmo en el entorno de la Biblioteca UJI	6 horas	lun 04/05/20	lun 11/05/20
26	**Integración**	**40 horas**	**mar 19/05/20**	**vie 29/05/20**
27	Creación de la libreria con el sistema de posicionamiento	30 horas	mar 19/05/20	mié 27/05/20
28	Integración de la libreria en la app "Biblioteca UJI"	10 horas	mié 27/05/20	vie 29/05/20
29	Fin de estancia en prácticas	0 horas	vie 29/05/20	vie 29/05/20

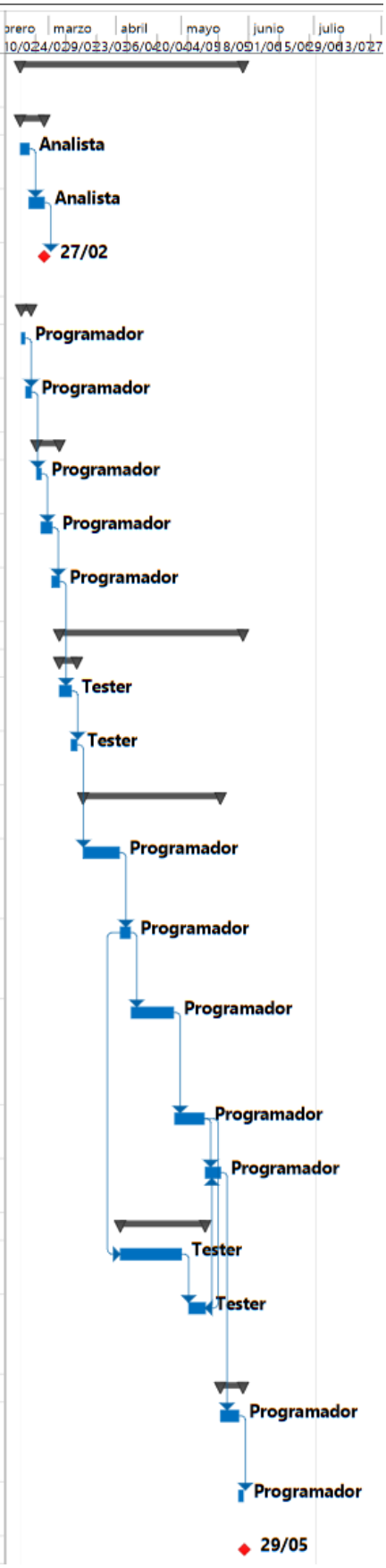

Figura 2.1: Diagrama de Gantt con la lista de tareas y su planificación temporal

Empezando por los recursos humanos, tal y como se puede apreciar en el diagrama de Gantt anterior (Figura 2.1) existen tres roles para el desarrollo del proyecto. Estos roles son el analista, el programador y el tester. Para estimar el coste de cada rol, se ha tenido en cuenta las horas de trabajo previstas de cada uno y el salario medio en España de cada rol [4].

Recurso	**Horas**	**Sueldo**	**Total**
Analista	24	14.4€/h	345€
Programador	225	9.6€/h	2160€
Tester	51	14€/h	714€
Total			**3219€**

Tabla 2.1: Coste de recursos humanos

Tal y como se aprecia en la Tabla 2.1, el coste que supondría el salario de los trabajadores ascendería a 3219€ netos. A esta cantidad se le debe añadir los impuestos y costes de contratación, los cuales varían entre un 20 % y un 30 %, por lo que el coste real sería de **4023.75€** suponiendo un 25 % de impuestos.

Con respecto a los recursos de software, se utilizan tanto programas de pago como gratuitos. Los de pago son IntelliJ [5] y MagicDraw [6]. En el caso del primero se puede optar por una versión gratuita con menor funcionalidad, no obstante, se ha utilizado la versión de pago. Con respecto a MagicDraw, consta de un pago único, por lo que supondremos que se puede utilizar en 20 proyectos. En la Tabla 2.2 se puede observar el coste de los recursos de software.

Recurso	**Uso**	**Coste**	**Total**
IntelliJ	4 meses	149€/año	49.7€
MagicDraw	20 proyectos	800€	40€
Total			**189€**

Tabla 2.2: Coste de recursos de software

Con respecto a los recursos de hardware, se tendrá en cuenta un teléfono inteligente Android, el ordenador portátil desde el cual se ha realizado el proyecto y las balizas bluetooth desplegadas en la biblioteca de la Universitat Jaume I. Para calcular el coste del teléfono y el ordenador portátil, se supondrá que el primero tiene una vida útil de 3 años y el segundo de 4 y se calculará su coste prorrateado. Con respecto a la balizas bluetooth, inicialmente se iba a utilizar un despliegue de 50 balizas ubicadas en la biblioteca de la UJI, y se asumirá que estas balizas se pueden utilizar para otros 5 proyectos de investigación. En la Tabla 2.3 se encuentra el coste total de los recursos de hardware.

Recurso	**Uso**	**Coste**	**Total**
Xiaomi Mi 9T	4 meses	280€	31.11€
Asus TUF Gaming FX504_GE	4 meses	900€	75€
Accent System iBKS 105	50 balizas / 5 proyectos	12.3€/baliza	123€
Total			**229.11€**

Tabla 2.3: Coste de recursos de hardware

Finalmente, falta por calcular los costes indirectos del desarrollo del proyecto. Estos costes incluyen los gastos de luz, agua, conexión a internet, servicio de limpieza, alquiler de oficinas... Como norma general, estos costes se calcula que ascienden a un 20 % del coste de los recursos humanos. La Tabla 2.4 muestra el resumen de los costes para el desarrollo del proyecto.

Origen	Coste
Recursos humanos	4023.75€
Recursos de software	189€
Recursos de hardware	229.11€
Costes indirectos	804.75€
Total	**5246.61€**

Tabla 2.4: Costes totales estimados del proyecto

2.4. Seguimiento del proyecto

En primer lugar, cabe destacar que la planificación inicial difiere ligeramente con las tareas realizadas finalmente debido a la situación de emergencia sanitaria ocasionada por la CoVid-19. Esta situación excepcional ha impedido que se realice todo el trabajo de forma presencial, tal y como se iba a realizar en un principio.

El proyecto estaba inicialmente pensado para funcionar en un entorno real, como puede ser el edificio Espaitec 2 y sobretodo la Biblioteca de la Universitat Jaume I. El cambio de la forma de trabajo de presencial a no presencial y el confinamiento imposibilita realizar la validación del algoritmo en estos entornos. Para solventar este problema, se propuso realizar esta validación en una habitación de mi hogar.

El cambio de entorno ha supuesto varios cambios en el desarrollo del proyecto que difiere con la planificación inicial, ya que en espacios como la biblioteca de la Universitat Jaume 1 o el edificio Espaitec 2 ya disponían del despliegue de balizas bluetooth de bajo consumo para realizar la validación en un entorno real. Por esta razón se han añadido tareas relacionadas con el estudio del nuevo entorno y el despliegue de las balizas. Esto ha provocado un aumento de las horas de trabajo necesarias para finalizar el proyecto que no estaba contemplado en la planificación inicial, y esta es una de las principales razones por las que el proyecto no se ha podido finalizar.

En la Tabla 2.5 se muestra la diferencia que existe entre la planificación inicial y las horas reales invertidas en cada tarea. En la primera columna se encuentra el id que hace referencia al id de la tarea en el diagrama de Gantt de la Figura 2.1. En caso de que el id sea 'N', significa que esta tarea es nueva y no aparece en el diagrama de Gantt de la planificación inicial. En la segunda columna se encuentra el nombre de la tarea a realizar. En la tercera columna se encuentran las horas previstas para la realización de la tarea, mientras que en la cuarta columna se encuentran las horas que se han invertido realmente en dicha tarea. Finalmente, en la quinta columna se encuentran las observaciones que dan una breve justificación con respecto al tiempo invertido en la tarea. Además, al final de la tabla se encuentra un resumen de las horas realizadas y las horas extra que serían necesarias invertir para finalizar el proyecto.

Comparación entre planificación y horas reales				
id	**Tarea**	**Planif**	**Real**	**Observaciones**
3	Desarrollo de la propuesta técnica	10h	10h	-
4	Revisión de la propuesta técnica	14h	14h	-

Continúa en la siguiente página

id	Tarea	Planif	Real	Observaciones
7	Búsqueda de información de conceptos clave	4h	4h	-
8	Búsqueda de información de las tecnologías a utilizar	6h	16h	Se ha hecho una investigación sobre su funcionamiento
10	Aprendizaje de recepción de datos de balizas BLE	5h	5h	-
11	Recogida de datos de sensores inerciales	6h	10h	-
12	Aprendizaje de filtro de Kalman	15h	36h	El estudio del filtro de Kalman ha llevado más tiempo del esperado.
15	Creación de test unitarios	15h	15h	Se tiene en cuenta implementación y ejecución de pruebas.
15	Creación de test de integración	15h	15h	Se tiene en cuenta implementación y ejecución de pruebas.
N	Estudio del entorno de pruebas	0h	8h	Tarea añadida por el confinamiento.
N	Despliegue de balizas en el entorno de pruebas	0h	2h	Tarea añadida por el confinamiento.
18	Algoritmo de posicionamiento basado en sensores inerciales	50h	86h	Errores en el funcionamiento del algoritmo. Investigación y estudio de alternativas.
19	Optimizar el coste computacional del algoritmo	12h	6h	Implementación sin finalizar y por tanto queda código por optimizar.
20	Algoritmo de posicionamiento basado en BLE	30h	30h	-
21	Algoritmo de navegación con filtro de Kalman	33h	12h	No se ha finalizado su implementación.
22	Mejora del algoritmo del centroide ponderado	24h	4h	Se ha estudiado pero no se ha mejorado.
N	Combinación de los sistemas de posicionamiento	0h	6h	Por un error no se le asignaron horas a la planificación inicial. Tampoco se ha finalizado debido a los retrasos.
24	Validación del algoritmo en el entorno ESPAITEC	15h	0h	Imposible de realizar debido a la situación de emergencia sanitaria del país.

Continúa en la siguiente página

id	Tarea	Planif	Real	Observaciones
25	Validación del algoritmo en el entorno de la Biblioteca UJI	6h	0h	Imposible de realizar debido a la situación de emergencia sanitaria del país.
N	Validación del algoritmo en nuevo entorno de pruebas	0h	21h	Reemplaza las horas de las tareas anteriores
27	Creación de la librería con el sistema de posicionamiento	30h	0h	Tarea opcional (No ha dado tiempo)
28	Integración de la librería en la app 'Biblioteca UJI'	10h	0h	Tarea opcional (No ha dado tiempo)

Horas realizadas:	300h
Horas restantes para finalizar el proyecto (sin tareas opcionales):	47h + posibles retrasos
Horas restantes para finalizar el proyecto (con tareas opcionales):	87h + posibles retrasos

Tabla 2.5: Comparación entre horas planificadas y horas reales del proyecto.

Tal y como se observa al final de la Tabla 2.5, la cantidad de horas necesaria para realizar el proyecto son 347 horas más lo necesario para conseguir arreglar el funcionamiento del sistema de posicionamiento basado en sensores inerciales que se comenta más adelante en el Capítulo 6. El exceso de horas se debe principalmente a 3 factores:

- **Problemas con el posicionamiento basado en sensores inerciales:** Es el principal factor de retraso del proyecto, ya que ha supuesto 36 horas más de las planificadas inicialmente y no se ha conseguido finalizar.
- **Nuevo entorno de pruebas:** Tal y como se ha mencionado anteriormente en esta sección, la situación de crisis sanitaria ha ocasionado que la estancia se realizase de forma telemática, y el confinamiento ha impedido que el entorno de pruebas que estaba preparado por la empresa se haya podido utilizar y se ha tenido que crear uno nuevo. Esto ha supuesto 10 horas extra de trabajo.
- **Tareas no planificadas:** Por un error inicial no se tuvo en cuenta las horas de trabajo para llevar a cabo la combinación de los sistemas de posicionamiento, en el cual se han invertido 6 horas de trabajo y no se ha finalizado por problemas relacionados con el primer punto.

Capítulo 3

Conceptos teóricos de las tecnologías

3.1. Introducción a las tecnologías

Para el desarrollo del proyecto se van a utilizar principalmente 2 herramientas para ubicar al usuario. Estas dos herramientas son los sensores inerciales y las balizas bluetooth de bajo consumo.

Los sensores inerciales actualmente se encuentran en todos los móviles inteligentes del mercado, por lo que la disponibilidad de estos sensores es factible para cualquier usuario.

Por otra parte, las balizas bluetooth son externas a estos dispositivos móviles, por lo que se deben comprar por separado y ubicar en el entorno. Los dispositivos móviles son capaces de detectar estas balizas, las cuales emiten señales de forma constante con información relevante para llevar a cabo el proyecto.

Además, se utilizará el filtro de Kalman para mejorar la precisión de los datos recopilados por los sensores.

Una de las tareas más relevantes del proyecto consiste en la investigación y estudio de todas estas tecnologías para así lograr mejorar los algoritmos ya existentes. Es por esto que a continuación se explican más detenidamente el funcionamiento de estas herramientas y del filtro de Kalman. De esta forma, también se dará pie a entender mejor la implementación de los algoritmos, los errores de precisión y las dificultades que han podido ocasionar para desarrollar el proyecto, así como la ventaja que supondría la optimización del sistema de posicionamiento.

3.2. Sensores inerciales

Los sensores inerciales son aquellos que se utilizan para capturar y analizar el movimiento. Hay distintos tipos de sensores para capturar distinta información. Estos sensores son el ace-

lerómetro, para medir la aceleración lineal, el giroscopio, para medir la velocidad angular, y el magnetómetro, para obtener la ubicación del norte magnético.

A continuación, se explicará más detalladamente para qué se utilizan los distintos sensores inerciales y una breve explicación técnica de cómo funcionan.

3.2.1. Acelerómetro

El acelerómetro es un sensor que se encarga de medir la variación de velocidad de un objeto dentro de su propio marco de reposo instantáneo. Esto quiere decir que el sensor detectará una aceleración nula si el objeto en cuestión se encuentra en caída libre, es decir, los datos recibidos por el acelerómetro incluyen la aceleración debida al campo gravitatorio de la tierra, que es aproximadamente de 9,81m/s^2.

Los dispositivos Android incluyen un acelerómetro de 3 ejes para poder medir la variación de velocidad en cualquier dirección. En la Figura 3.1 se puede observar la distribución de estos ejes en un dispositivo móvil.

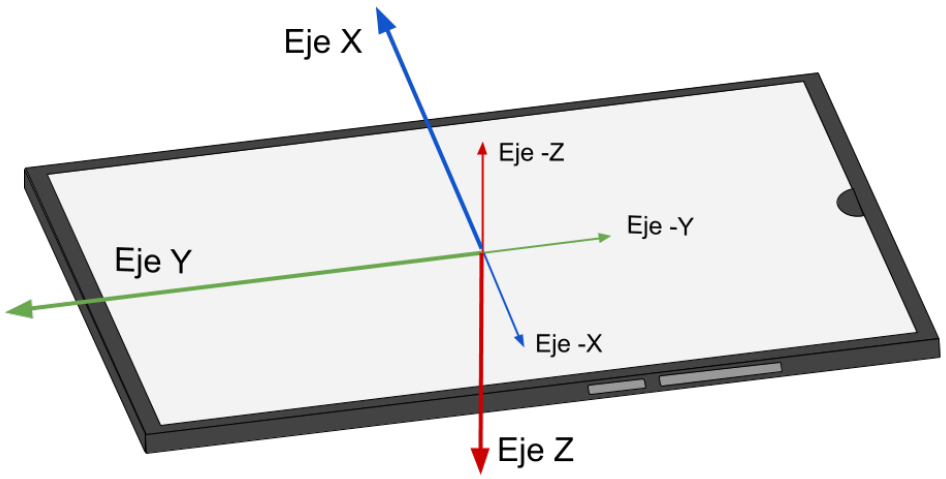

Figura 3.1: Distribución de los 3 ejes del acelerómetro incluido en los teléfonos inteligentes, diferenciando el sentido en el cual la aceleración es positiva o negativa en cada eje.

Existen varios tipos de acelerómetros como los piezoeléctricos, los mecánicos y los de efecto Hall, pero vamos a centrarnos en el funcionamiento del acelerómetro capacitivo, ya que es el tipo más común en los teléfonos inteligentes.

El acelerómetro capacitivo consiste en en dos placas metálicas, una fija y otra móvil, ubicadas una enfrente de la otra formando un condensador con un material dieléctrico entre ambas, de forma que cuando se produce una aceleración la distancia entre las placas se ve incrementada o reducida y la capacidad del condensador cambia. Esta información es recogida por el material dieléctrico, de forma que es capaz de calcular la aceleración producida [7].

3.2.2. Giroscopio

El giroscopio se encarga de medir la velocidad angular en los 3 ejes de rotación, es decir, es capaz de calcular la velocidad de rotación medida en rad/s de cada uno de estos ejes. En la Figura 3.2 se puede observar la distribución de los distintos ejes y el sentido de la rotación en la que se considera velocidad positiva en el caso de un teléfono inteligente.

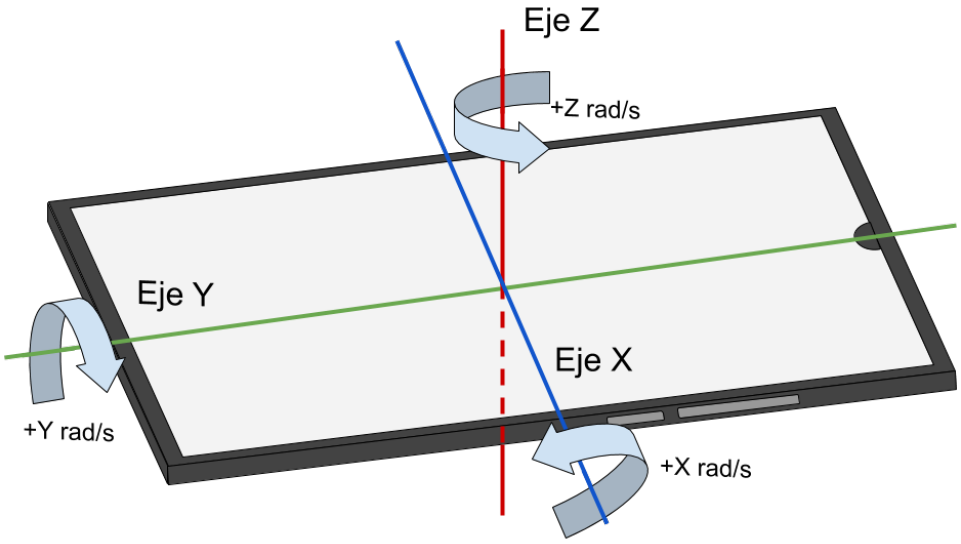

Figura 3.2: Distribución de los 3 ejes del giroscopio incluido en los teléfonos inteligentes, indicando en que sentido de la rotación se considera rotación positiva.

El funcionamiento del giroscopio que se encuentra en los teléfonos inteligentes se basa en un brazo de accionamiento sobre el cual actúa una fuerza llamada fuerza de Coriolis al rotar sobre un componente fijo llamado estátor. Cuando la fuerza de Coriolis actúa, el estátor se dobla, lo cual produce un movimiento en el brazo de accionamiento. Finalmente, este movimiento es traducido a una señal eléctrica con la cual se obtiene la velocidad angular del eje en el que se encuentra [7].

3.2.3. Magnetómetro

El magnetómetro es un sensor que detecta la fuerza o la dirección de la señal magnética de una fuerza. Estos sensores se utilizan para una gran variedad de tareas, aunque el uso más habitual, que recibe el nombre de geomagnetómetro, es el de medir el campo magnético terrestre para ubicar el polo norte magnético, de forma que funciona como una brújula. Gracias a este sensor, podemos ubicar la dirección en la que el dispositivo está orientado con respecto a la tierra [8].

3.2.4. Sistema AHRS

El sistema AHRS (del inglés *Attitude and Heading Reference Systems* o Sistemas de Referencia de Inclinación y Rumbo en español) se trata de un sistema que está formado por giroscopios, acelerómetros y magnetómetros.

En los dispositivos android, se puede obtener los datos AHRS del dispositivo, ya que se calcula internamente con los datos recibidos de los otros sensores. La información que proporciona este sistema es de orientación del dispositivo, es decir, devuelve la inclinación angular en grados de cada eje.

En la Figura 3.3 se puede observar el caso en el cual el sistema AHRS nos devolvería $+90^{o}$ en el eje Y, 0^{o} en el eje X, y 0^{o} en el eje Z. La rotación positiva en el eje Y es en el mismo sentido que la aceleración positiva del giroscopio, al igual que sucede en el eje X y en el Z.

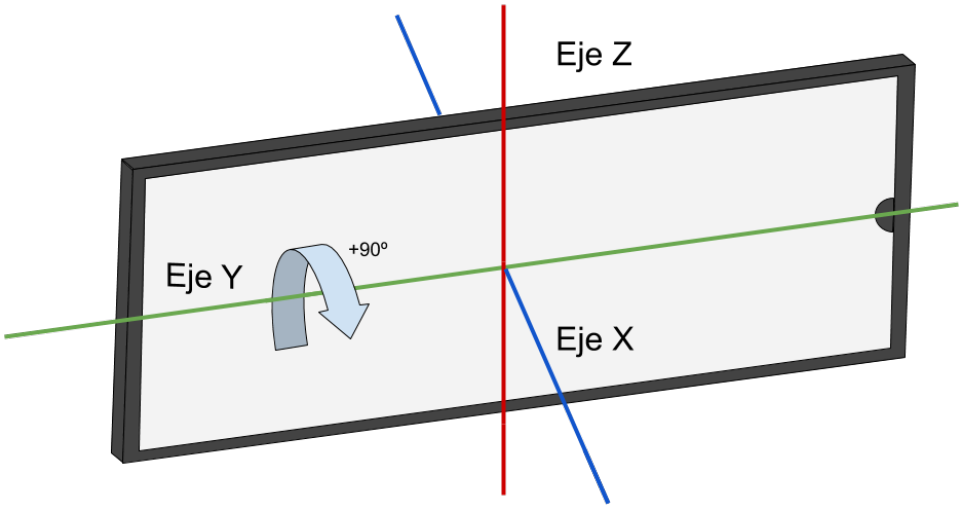

Figura 3.3: Teléfono inteligente rotado $+90^{o}$ en el eje Y, 0^{o} en el eje X y 0^{o} en el eje Z

Este sistema tiene una particularidad, y es que a la hora de representar los ángulos de cada eje, no siempre muestra los datos que uno puede esperar, ya que dependiendo del eje de rotación, una posición concreta del teléfono se podría interpretar de distintas formas. Es por esto que los ejes no representan el mismo rango de ángulos.

En la Figura 3.4 se puede observar un caso en el cual el teléfono es rotado 45^{o} y 135^{o} en el eje X, pero ambas las considera como 45^{o} en este eje, mientras que los otros dos ejes pasan de ser de 0^{o} a 180^{o}. Esto se debe a que también se puede llegar a ese punto realizando dichas rotaciones, es decir, si primero rotamos el móvil 180^{o} sobre el eje Y, a continuación lo rotamos 180^{o} en el eje Z, de forma que apunte al sur, y por ultimo lo rotamos 45^{o} en el eje X, obtenemos el mismo resultado que rotando el dispositivo 135^{o} sobre el eje X.

Con el fin de evitar estas redundancias, el sistema AHRS define su propio rango de ángulos para cada eje, midiendo desde -180^{o} hasta 180^{o} tanto en el eje Y como en el Z, mientras que en el eje X mide desde -90^{o} hasta 90^{o}.

El eje X, se considera entre 0º y 90º si la inclinación se encuentra entre el primer y el segundo cuadrante, y entre 0º y -90º si la inclinación se encuentra entre el tercer y el cuarto cuadrante. En otras palabras, se considera inclinación positiva si la parte superior del dispositivo se encuentra por encima de la parte inferior, y se considera negativa en caso contrario. El dispositivo tendrá una inclinación de 90º cuando se encuentre en vertical con la parte superior por encima de la inferior, y -90º cuando se encuentra en vertical siendo la parte inferior la que se encuentra por encima de la parte superior.

Con respecto al eje Y, se considera 0º cuando el dispositivo se encuentra de forma horizontal con la pantalla en la parte superior. Se consideran 90º o -90º cuando el dispositivo se encuentra apoyado sobre los laterales. Se considera 180º o -180º cuando la pantalla se encuentra mirando hacia abajo.

Por último, el eje Z se considera 0º cuando se apunta al norte, y a partir de este punto, se considera orientación positiva si el dispositivo se orienta hacia el oeste hasta un máximo de 180º cuando apunta al sur, y se considera orientación negativa si el dispositivo se orienta hacia el este hasta un máximo de -180º.

Puesto que este sistema está incluido en los teléfonos inteligentes y nos devuelve directamente los datos relacionados con la orientación del mismo, es decir, su inclinación en cada eje y la dirección relativa al norte magnético, no será necesario consultar los datos del giroscopio ni del magnetómetro para obtener esta información.

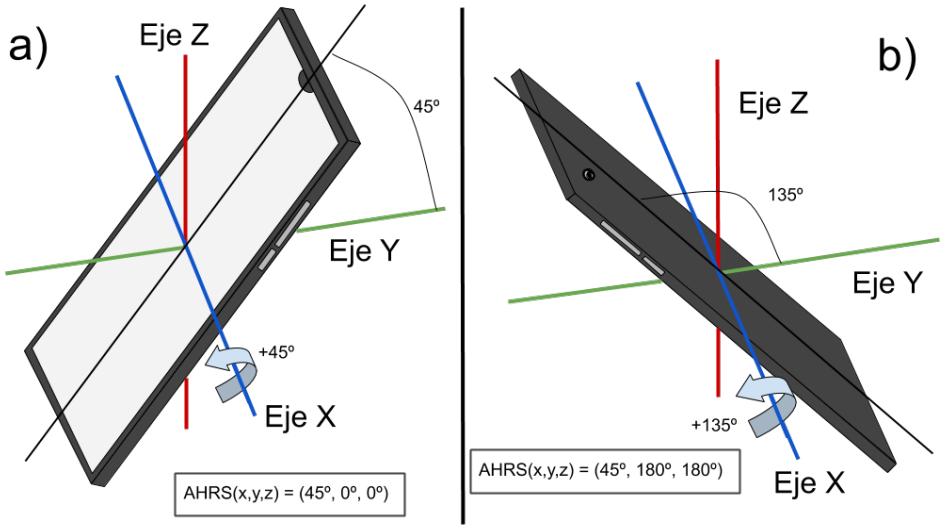

Figura 3.4: El sistema AHRS devuelve 45º tanto en el caso a) como en el caso b). En el caso a) se considera que el dispositivo está simplemente rotado 45º en el eje X, mientras que en el caso b), al rotarlo 135º, el sistema AHRS considera que el dispositivo ha sido rotado 180º en los ejes Y y Z, y 45º en el eje X

3.3. Bluetooth de bajo consumo (BLE)

BLE (*Bluetooth Low Energy*) es una tecnología de red de área personal inalámbrica basada en la tecnología bluetooth, con la particularidad de que el consumo es mucho más reducido a la vez que mantiene un rango de alcance de comunicación muy similar.

Esta tecnología surgió en 2006, y no fue hasta 2011 que se creo el esquema Bluetooth Smart para aclarar la compatibilidad de los dispositivos con este sistema, indicando que si un dispositivo es compatible con BLE se le denomina Bluetooth Smart Ready. Los dispositivos Android son compatibles con este sistema a partir de la versión 4.3 de este sistema operativo, permitiendo detectar señales BLE.

Las balizas envían señales por difusión de forma que cualquier dispositivo pueda recibir estos datos. Existen varios protocolos de difusión de datos para las balizas BLE. Los más importantes son *iBeacon*, desarrollado por *Apple*, y *Eddystone*, desarrollado por Google, aunque también existen otras opciones como AltBeacon, una opción de código abierto desarrollada por Radius Networks, y URIBeacon, que tiene la particularidad de difundir una url en lugar de un identificador [9].

Para la realización de este proyecto, el modelo de baliza utilizado ha sido el Smart Beacon SB16-2 de la empresa kontakt [10]. En la página del modelo se pueden observar las características de este modelo. La duración de la batería de este modelo es de aproximadamente 2 años, ya que contiene 2000 mAh y, como ya se ha comentado antes, la tecnología BLE consume muy poca batería. Esta baliza es compatible con iBeacon y Eddystone. Debido a que la empresa trabaja principalmente con iBeacon, esta ha sido el protocolo elegido para realizar el proyecto.

3.3.1. Protocolo iBeacon

El protocolo iBeacon fue desarrollado por Apple en 2013 [11]. Se trata de un protocolo pensado para tecnología bluetooth de bajo consumo que transmite un identificador universal único (UUID) junto con algunos bytes que ofrecen información específica de la baliza electrónica. Con esta información se puede localizar la ubicación del dispositivo o realizar una acción específica basada en la localización como por ejemplo activar una notificación de tipo *push* en un teléfono inteligente. Además, gracias a esta información también ofrece una alternativa como sistema de posicionamiento en interiores.

Por lo general, un UUID se utiliza para diferenciar dispositivos, pero en el caso de las balizas electrónicas, el UUID se utiliza para diferenciar el fabricante, el propietario o la aplicación [12]. Para entenderlo mejor, pondremos el caso de una persona que decide comprar unas balizas y utilizarlas en su comercio. En este caso se utilizaría el mismo UUID en todas las balizas, ya que existen otros datos para diferenciar las balizas entre si. Por otra parte, pongamos el caso de que este usuario tiene éxito y decide abrir un segundo local para su comercio, haciendo un segundo despliegue de balizas en el nuevo local. En este caso, todas las balizas dentro de un mismo local utilizarían el mismo UUID, pero el UUID sería distinto en cada local.

Como se ha comentado antes, existen otros datos para diferenciar las balizas de un mismo usuario o una misma aplicación que utilizan el mismo UUID. Estos datos son el *major* y el *minor* [12]. Estos valores se utilizan por lo general para distinguir entre conjunto o entorno y algo específico. Continuando con el ejemplo anterior, supongamos que el local del comerciante está dividido en varias secciones que contienen varios productos específicos. Para identificar la sección en la que se encuentra un usuario se utilizaría el *major*, mientras que para identificar delante de qué producto de la sección se encuentra se utilizaría el *minor*.

La estructura [13] de un paquete de datos que utiliza el protocolo iBeacon se compone de 30 bytes. En la Tabla 3.1 muestra la información que contiene cada byte. Los primeros 3 bytes son los valores estándar de una señal BLE, es decir, siempre son los mismos valores para cualquier señal emitida por esta tecnología. Los siguientes 6 bytes contienen información especificada por el fabricante que contiene el protocolo que se está utilizando. A continuación, hay 16 bytes que contienen el UUID (*Universally Unique Identifier*) y 4 bytes definiendo el *major* y el *minor*, siendo los primeros dos bytes los del *major* y los ultimos 2 los del *minor*. Por último, se encuentra el byte que indica la intensidad de la señal que se recibe cuando el receptor se encuentra a 1 metro de la baliza.

3.4. Filtro de Kalman

El filtro de Kalman [14], o estimación lineal cuadrática, es un algoritmo iterativo que ayuda a identificar el estado no medible de un sistema dinámico lineal. Este filtro es capaz de detectar, y por tanto reducir o eliminar, el ruido blanco de una señal. El ruido blanco es una señal aleatoria que no guarda correlación estadística en distintos tiempos y que por tanto, altera el valor real de la señal principal.

La eliminación del ruido blanco es muy importante para este proyecto, ya que la falta de precisión de los sensores inerciales es considerada ruido blanco que hace que los resultados no sean exactos y que con el tiempo se incremente exponencialmente el error a la hora de calcular la velocidad de desplazamiento y por tanto la posición del usuario.

El funcionamiento del filtro de Kalman se puede dividir en dos pasos generales:

- **La predicción:** a partir del estado anterior del sistema y mediante las ecuaciones que marcan la evolución del mismo, el algoritmo predice cual será el valor de los datos actuales.
- **La corrección:** a partir de los datos recogidos por los sensores se corrige la predicción para obtener el estado actual

Además, como se ha comentado anteriormente, el filtro de Kalman es iterativo. Esto quiere decir que una vez se ha realizado la corrección, se vuelve a realizar la predicción del siguiente estado, lo que permite que este algoritmo este funcionando en tiempo real utilizando únicamente los datos actuales de entrada, el estado calculado previamente y su matriz de incertidumbre.

Byte	**Valor por defecto**	**Descripción**	**Propiedad**
0	0x02	Longitud del indicador - 2 bytes	Constante
1	0x01	Tipo de dato - Indicador	Constante
2	0x06	Indicador LE y BR/EDR	Constante
3	0x1a	Longitud de datos - 26 bytes	Constante
4	0xff	Tipo - dato específico del fabricante	Constante
5	0x4c	Identificación del fabricante	Constante
6	0x00	Identificación del fabricante	Constante
7	0x02	Subtipo (iBeacon = 0x02)	Constante
8	0x15	Longitud de subtipo (iBeacon = 0x15)	Constante
9	0xf7	1r byte del UUID	UUID del propietario
10	0x82	2o byte del UUID	UUID del propietario
11	0x6d	3r byte del UUID	UUID del propietario
12	0xa6	4o byte del UUID	UUID del propietario
13	0x4f	5o byte del UUID	UUID del propietario
14	0xa2	6o byte del UUID	UUID del propietario
15	0x4e	7o byte del UUID	UUID del propietario
16	0x98	8o byte del UUID	UUID del propietario
17	0x80	9o byte del UUID	UUID del propietario
18	0x24	10o byte del UUID	UUID del propietario
19	0xbc	11r byte del UUID	UUID del propietario
20	0x5b	12o byte del UUID	UUID del propietario
21	0x71	13r byte del UUID	UUID del propietario
22	0xe0	14o byte del UUID	UUID del propietario
23	0x89	15o byte del UUID	UUID del propietario
24	0x3e	16o byte del UUID	UUID del propietario
25	xx*	1r byte del Major	Valor Major
26	xx*	2o byte del Major	Valor Major
27	xx*	1r byte del Minor	Valor Minor
28	xx*	2o byte del Minor	Valor Minor
29	0xb3	Potencia de señal (calibrado RSSI@1m)	Potencia de señal a 1 metro

Tabla 3.1: Estructura del paquete de datos de una señal BLE mediante el protocolo iBeacon. El valor por defecto en los bytes de los datos del *Major* y el *Minor* son aleatorios

3.5. Combinación de sistemas

Para el desarrollo del proyecto se pretende utilizar tanto el sistema de posicionamiento mediante sensores inerciales como el sistema de posicionamiento basado en balizas BLE. Para llevarlo a cabo, se han desarrollado los dos sistemas de posicionamiento por separado, para una vez estén los dos acabados y funcionales, se puedan combinar y obtener un resultado mucho más preciso.

En ambos sistemas de posicionamiento se utilizan ventanas temporales para mejorar su precisión. Esto quiere decir que una vez recibidos los datos, en lugar de calcular inmediatamente la posición, se almacenan y se hace la media de los datos recibidos dentro de la ventana temporal. A pesar de que se implementan ventanas temporales en ambos sistemas de posicionamiento, no tienen la misma duración en los dos. Para explicar el porqué, deben explicarse primero los problemas que tiene cada uno de estos sistemas.

El problema principal de bluetooth, es que dependiendo de como estén configuradas las balizas, el tiempo entre la emisión de datos puede variar entre 100 milisegundos y 1 segundo. Al emitir datos con mayor frecuencia, es decir, menor tiempo entre un mensaje y el siguiente, la duración de la batería de la baliza se ve drásticamente reducida. Es por esto que se suele configurar a menor frecuencia, y por tanto, la ventana temporal debe incrementarse a 1 o 2 segundos dependiendo de la configuración. Esto provoca que, a cambio de tener una ubicación precisa, perdemos uno de los factores más importantes del proyecto como es la ubicación en tiempo real, ya que la posición que nos devolvería sería la de hace uno o dos segundos.

Por otra parte, los sensores inerciales tienen muchos problemas de precisión, pero la frecuencia en la que el dispositivo nos proporciona los datos es más elevada. En los sensores inerciales hay mucho ruido, el cual provoca que el error a la hora de calcular la posición se vea incrementado exponencialmente con el paso del tiempo. Una solución sería implementar un filtro como el filtro de Kalman, pero este es un algoritmo con demasiado coste computacional como para calcularlo a la misma frecuencia en la que se reciben los datos. Para poder implementar el filtro de Kalman sin un sobrecoste computacional, se implementan ventanas temporales en el sistema de posicionamiento basado en sensores inerciales. La diferencia con respecto a las ventanas temporales en BLE, es que en este caso son de una duración mucho más reducida gracias a la frecuencia elevada del envío de datos de los sensores, lo que permite calcular la posición hasta 5 veces por segundo.

Explicados los problemas y ventajas de cada sistema, es fácil comprender la motivación de combinar estos sistemas. Con el sistema BLE podemos obtener datos muy fiables, pero tarda más en darnos un resultado. Por otra parte, los sensores inerciales nos proporcionan un resultado con mucha rapidez, pero conforme pasa el tiempo, pierde completamente la precisión. Es por esto que la fuente fiable de la posición siempre será la calculada a partir de las balizas BLE, pero los sensores inerciales proporcionaran a corto plazo información sobre el desplazamiento entre que se recibe el siguiente dato válido para calcular la posición por medio de las balizas, reiniciando en ese momento los datos acumulados por los sensores para evitar el incremento del error.

También cabe destacar que no existe un solapamiento entre ventanas temporales. Normalmente, cuando se trabaja con ventanas temporales existe un solapamiento entre los datos, es

decir, un dato aparece en varias ventanas temporales.. Explicado de otra forma, si se contemplan 12 instantes en los que se ha recibido datos y una ventana temporal contempla 6 instantes, en este proyecto se obtendrían 2 ventanas temporales, una de los instantes 1 al 6 y otra de los instantes 7 al 12, mientras que si existe solapamiento se podrían obtener 3 o más ventanas temporales, contemplando, por ejemplo, los instantes del 1 al 6, del 4 al 9 y del 7 al 12. En este proyecto se ha optado por que no exista solapamiento para conseguir un menor coste computacional, no obstante, no se cierra la posibilidad de adaptar el código en un futuro para que si exista solapamiento.

Capítulo 4

Análisis y diseño del sistema

4.1. Análisis del sistema

En esta sección se muestra los requisitos y el diagrama de casos de uso correspondientes al análisis del sistema de posicionamiento, junto con sus respectivas documentaciones.

4.1.1. Requisitos del sistema

Este sistema cuenta con varios requisitos:

- El sistema debe poder estimar la posición del usuario a partir de los datos de los sensores inerciales de un dispositivo Android.
- El sistema debe poder estimar la posición del usuario a partir de los datos recibidos de las balizas BLE.
- El sistema de posicionamiento por BLE debe utilizar el algoritmo del centroide ponderado (ver Sección 5.2.2).
- El sistema debe combinar los dos métodos de posicionamiento para ubicar al usuario.
- El sistema debe poder estimar la posición del usuario varias veces por segundo de forma óptima.

El primer requisito implica la implementación de código necesaria para tratar los datos recibidos de los sensores inerciales. Esto incluye la corrección de la fuerza gravitatoria del acelerómetro, la creación de ventanas temporales y el cálculo necesario para estimar la posición del usuario.

El segundo requisito implica la implementación de código necesaria para tratar los datos recibidos de las balizas BLE. Esto incluye la creación de ventanas temporales y el cálculo necesario para estimar la posición del usuario.

El tercer requisito es un requisito impuesto por la empresa. A pesar de poder implementar distintos algoritmos para ubicar el usuario, la empresa decidió que se utilizase el algoritmo del centroide ponderado ya que es el que utilizan habitualmente.

El cuarto requisito implica la implementación de código necesaria para gestionar los datos generados por los dos métodos de posicionamiento y obtener un resultado preciso.

El quinto requisito es un requisito de calidad, orientado a la optimización del algoritmo teniendo en cuenta su posterior extensión con el filtro de Kalman y a cumplir con la expectativa de ubicar de forma fluida al usuario.

4.1.2. Diagrama de casos de uso

En la Figura 4.1 se muestra el diagrama de casos de uso del sistema de posicionamiento combinado. Como se puede observar solo hay un único actor:

- Usuario: Cualquier persona o aplicación externa que haga uso del sistema de posicionamiento.

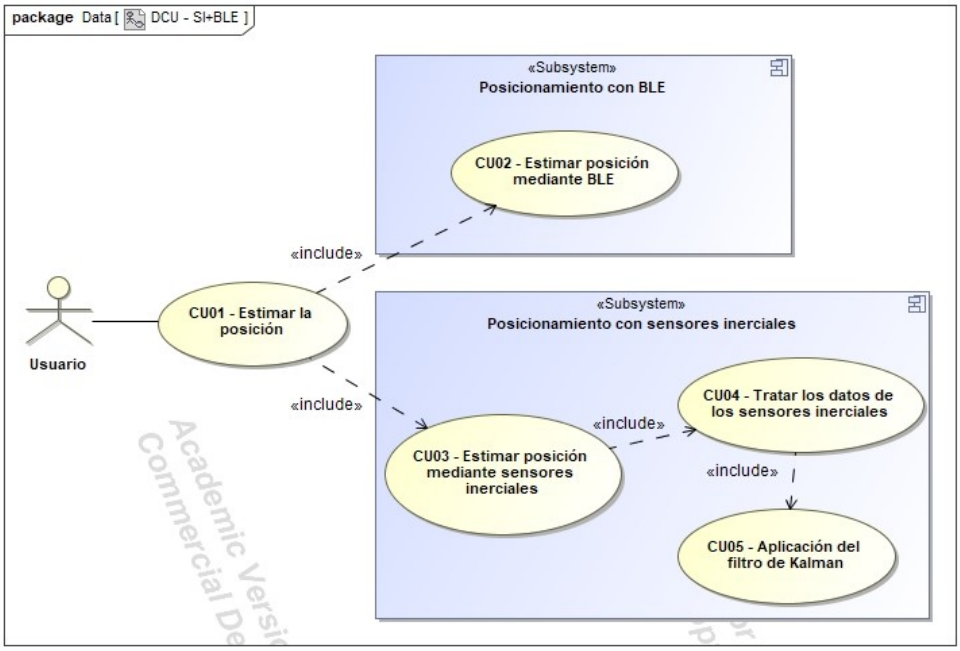

Figura 4.1: Diagrama de casos de uso del sistema de posicionamiento combinado

A pesar de que se podría haber resumido todo en un solo caso de uso, se ha separado en dos subsistemas. Se ha hecho de esta forma ya que cada método para posicionar al usuario se ha implementado por separado y, por tanto, ambos deben ser funcionales por separado. Además, en el caso de los sensores inerciales se ha separado el tratamiento de los datos y la aplicación del filtro de Kalman debido a su compleja funcionalidad.

En las Tablas 4.1, 4.2, 4.3, 4.4 y 4.5 se encuentra la documentación de cada uno de los casos de uso que incluye el sistema.

ID: CU01 **Nombre:** Estimar la posición **Versión:** 1.0 **Fuente:** Jaime Higueras Ruiz
Descripción: Este caso de uso representa la funcionalidad que permite ubicar al usuario
Flujo de eventos: 1. El usuario solicita la estimación de su posición. 2. El sistema recoge los datos del dispositivo móvil. 3. El sistema calcula la posición a partir de los datos de los sensores inerciales (CU03, Tabla 4.3). 4. El sistema calcula la posición a partir de los datos BLE (CU02, Tabla 4.2). 5. El sistema combina los cálculos de ambos sistemas. 6. El sistema devuelve al usuario la estimación de su posición. **Precondiciones:** El usuario debe ubicarse en la zona donde se encuentra el despliegue de balizas BLE

Tabla 4.1: Documentación del caso de uso CU01

ID: CU02 **Nombre:** Estimar posición mediante BLE **Versión:** 1.0 **Fuente:** Jaime Higueras Ruiz
Descripción: Este caso de uso representa la funcionalidad que permite ubicar al usuario mediante el uso de las balizas bluetooth de bajo consumo (BLE)
Flujo de eventos: 1. El sistema recibe los datos de las balizas BLE. 2. El sistema descarta los datos de las balizas lejanas. 3. El sistema asigna los datos a la ventana temporal **Flujo de eventos alternativo:** En caso de ser el último dato de la ventana temporal (tras el punto 4): 1. El sistema inicia una nueva ventana temporal. 2. El sistema realiza una media de los datos de la ventana temporal finalizada. 3. El sistema estima la posición del usuario mediante el algoritmo del centroide ponderado. 4. El sistema devuelve la estimación de la posición. **Precondiciones:** El usuario se debe encontrar en la zona donde se encuentran desplegadas las balizas BLE

Tabla 4.2: Documentación del caso de uso CU02

ID: CU03
Nombre: Estimar posición mediante sensores inerciales
Versión: 1.0
Fuente: Jaime Higueras Ruiz

Descripción: Este caso de uso representa la funcionalidad que permite ubicar al usuario mediante el uso de los sensores inerciales de un dispositivo Android

Flujo de eventos:

1. El sistema solicita estimar la posición mediante sensores inerciales.
2. El sistema recibe los datos de los sensores inerciales.
3. El sistema realiza un tratamiento a los datos de los sensores inerciales (CU04, Tabla 4.4).
4. El sistema devuelve el resultado de la ubicación calculada.

Precondiciones: Ninguna

Tabla 4.3: Documentación del caso de uso CU03

ID: CU04
Nombre: Tratar los datos de los sensores inerciales
Versión: 1.0
Fuente: Jaime Higueras Ruiz

Descripción: Este caso de uso representa la funcionalidad que se encarga de pulir los datos de los sensores inerciales para poder realizar una estimación correcta del usuario

Flujo de eventos:

1. El sistema analiza los datos recibidos.
2. El sistema elimina la influencia de la fuerza gravitatoria en los datos del acelerómetro.
3. El sistema asigna los datos a la ventana temporal.

Flujo de eventos alternativo:
En caso de ser el último dato de la ventana temporal (tras el punto 3):

1. El sistema realiza una media de los datos de la ventana temporal.
2. El sistema aplica el filtro de Kalman (CU05, Tabla 4.5).
3. El sistema inicia una nueva ventana temporal.
4. El sistema devuelve la estimación de la posición.

Precondiciones: El sistema debe estar recibiendo datos de los sensores inerciales

Tabla 4.4: Documentación del caso de uso CU04

ID: CU05 **Nombre:** Aplicación del filtro de Kalman **Versión:** 1.0 **Fuente:** Jaime Higueras Ruiz
Descripción: Este caso de uso representa la funcionalidad que filtra los datos de las ventanas temporales de los sensores inerciales aplicando el filtro de Kalman
Flujo de eventos: 1. El algoritmo recibe los datos de la ventana temporal 2. El algoritmo compara los datos recibidos con la predicción realizada anteriormente. 3. El algoritmo estima la posición más probable del usuario. 4. El algoritmo realiza una predicción de la próxima ubicación. **Flujo de eventos alternativo:** En caso de ser la primera vez en ejecutarse: 1. El algoritmo realiza una primera predicción partiendo del caso base. 2. El sistema continúa con el flujo de eventos normal. **Precondiciones:** La ventana temporal actual acaba de finalizar

Tabla 4.5: Documentación del caso de uso CU05

4.2. Diseño del sistema

En esta sección se muestran los diagramas de clases de diseño para el desarrollo del sistema de posicionamiento combinado.

4.2.1. Diagrama de clases

En la Figura 4.2 se puede observar el diagrama de clases del sistema de posicionamiento. En este diagrama se aprecian las siguientes clases:

- **Dato:** Clase generalizada que contiene la marca de tiempo del dato recibido.
- **Acelerómetro:** Clase que desciende de *Dato* y contiene los valores aportados por el acelerómetro en cada eje.
- **AHRS:** Clase que desciende de *Dato* y contiene los valores aportados por el sistema AHRS que aportan la orientación del dispositivo en cada eje.
- **SeñalBLE:** Clase que desciende de *Dato* y contiene los datos de la BalizaBLE que ha emitido la señal y su RSSI (indicador de fuerza de la señal recibida).
- **BalizaBLE:** Clase que contiene los datos de una baliza desplegada en el entorno.
- **VentanaTemporal:** Clase que contiene las marcas de tiempo del inicio y el final de una ventana temopral.

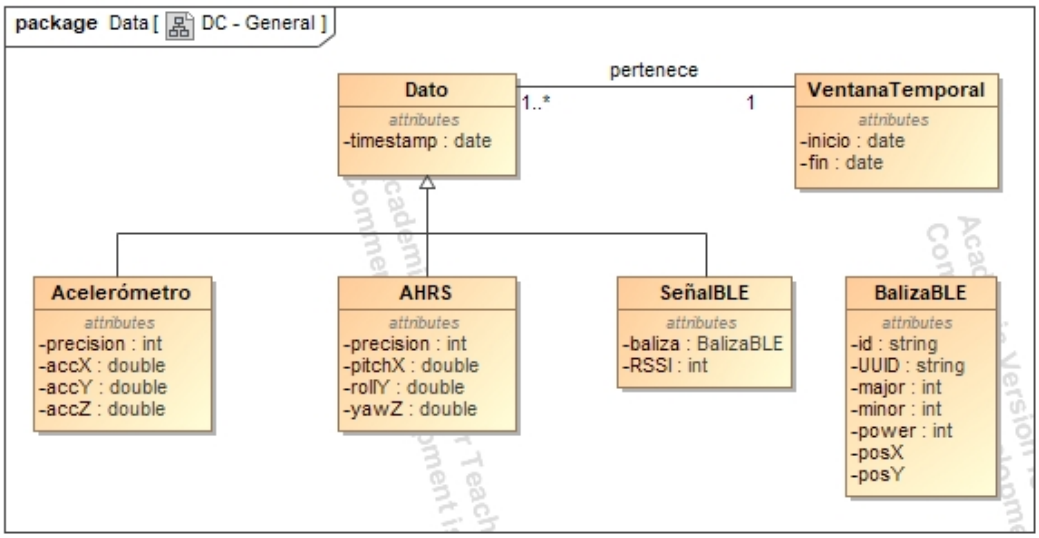

Figura 4.2: Diagrama de clases de diseño general

4.2.2. Gestión de datos

En esta sección se explica como se ha diseñado la gestión de datos del sistema de posicionamiento.

Para la persistencia de los datos, no se ha utilizado ninguna base de datos, ya que esto añadiría complejidad al sistema a la hora de realizar consultas y no es esencial para cumplir con el objetivo del proyecto. Esto se debe a que el sistema está pensado para funcionar como una librería a la que se le pasarán los datos directamente, y esta forma de almacenar estos datos se utilizará para realizar algunas pruebas.

En su lugar, se ha utilizado 2 tipos de ficheros. En el primer tipo de fichero se almacena la información del despliegue de balizas bluetooth, y el segundo tipo de fichero almacena información extraída del dispositivo, ya sea de los sensores inerciales o de la tecnología BLE.

En las Tablas 4.6 y 4.7 se puede observar la disposición de los datos en los ficheros de texto, los cuales van separados por ';' para poder separarlos a la hora de extraer los datos.

Id	**MAC**	**Major**	**Minor**	**PosX**	**PosY**
XXXX (string)	XX:XX:XX:XX:XX:XX	xx (int)	xx (int)	xx (int)	xx (int)
... (string)	...	... (int)	... (int)	... (int)	... (int)

Tabla 4.6: Disposición de los datos de las balizas desplegadas en un fichero de texto

Tipo	**AppTimestamp**	**Datos**
BLE4	x.xxx (seg)	Protocolo, MAC, RSSI(dBm), Power, MajorID, MinorID, UUID
AHRS	x.xxx (seg)	SensorTimestamp(s), Acc_X(m/s^2), Acc_Y(m/s^2), Acc_Z(m/s^2), Precisión
ACCE	x.xxx (seg)	SensorTimestamp(s), PitchX(deg), RollY(deg), YawZ(deg), Quat(2), Quat(3), Quat(4), Precisión

Tabla 4.7: Disposición de los datos de los sensores en un fichero de texto

Capítulo 5

Implementación y pruebas

5.1. Herramientas utilizadas para la implementación

En esta sección se detallan las herramientas que han sido necesarias para realizar la implementación del proyecto. Estas herramientas son:

- **IntelliJ IDEA:** Entorno de desarrollo integrado para desarrollar programas informáticos creado por JetBrains. Este IDE ha sido el que se ha utilizado para llevar a cabo toda la programación del sistema de posicionamiento en interiores.
- **Android Studio:** Entorno de desarrollo integrado oficial para desarrollar programas informáticos para la plataforma Android creado por Google. Este IDE se ha utilizada para investigar códigos ya existentes proporcionados por la empresa, e incluso adaptar ligeramente alguno para utilizarlo de apoyo en el desarrollo del proyecto.
- **Trello:** Programa para la organización de proyectos desarrollado por Altassian. Este programa se ha utilizado para llevar a cabo la gestión de las distintas tareas y sobretodo para anotar enlaces y redactar explicaciones en las tareas dedicadas a la investigación.
- **JUnit:** Conjunto de bibliotecas para la realización de pruebas unitarias en aplicaciones Java. Se ha utilizado para llevar a cabo pruebas unitarias sobre los algoritmos y fórmulas utilizadas a la hora de desarrollar el proyecto.
- **Github:** Sistema de control de versiones Git para alojar proyectos. Se ha utilizado para alojar el proyecto.
- **MagicDraw:** Herramienta para el desarrollo de diagramas UML. Se ha utilizado para desarrollar diagramas de casos de uso y diagramas de clases.

Además de estas herramientas, también se ha accedido a distintos códigos que han sido tanto complementarios al proyecto como de apoyo para el desarrollo del mismo. Estos códigos son:

- **Biblioteca Universitat Jaume I [2]:** Aplicación desarrollada por UBIK para el posicionamiento en el interior de la biblioteca de la Universitat Jaume I y para la búsqueda de libros en esta. Esta aplicación cuenta con distintas librerías privadas, por lo que no se pueden dar detalles. De este código se ha consultado principalmente una librería para el posicionamiento BLE que recibe los datos BLE, los interpreta, aplica filtros y calcula la posición.

- **GetSensorData_Android [15]:** Aplicación Android que recoge los datos de todos los sensores de los dispositivos. Esta aplicación se ha utilizado para recoger los datos de los sensores. Además, incluye una funcionalidad que permite crear un fichero con los datos recogidos en el intervalo de tiempo deseado. Esta parte del código ha sido modificada para que almacene únicamente los datos de sensores inerciales y BLE y poder utilizar los ficheros para la implementación de pruebas de verificación y validación.

5.2. Implementación

En esta sección se detalla la implementación de cada sistema de posicionamiento, incluyendo patrones utilizados, algoritmos empleados, problemas encontrados y sus soluciones, así como qué es lo que no se ha llegado a implementar y el motivo.

5.2.1. Posicionamiento basado en sensores inerciales

Lo primero que se implementó fue el sistema de posicionamiento basado en sensores inerciales. Este sistema consiste en emplear los datos aportados por los sensores inerciales, es decir, el acelerómetro, el giroscopio y el magnetómetro, el funcionamiento de los cuales ha sido explicado en el Capítulo 3, para aproximar la posición del usuario.

Para desarrollar este sistema, primero se implementó la gestión de los datos de los sensores inerciales a partir de un fichero de texto. La primera clase que se creó para ello fue la clase Main, la cual no forma parte del sistema de posicionamiento en si, sino que funciona de apoyo para la realización de pruebas de validación. Esta clase se encarga de pedir por consola el fichero de texto que contiene datos de los sensores inerciales y extraer todos los atributos de cada lectura para enviarlos al gestor de datos de forma que simula una lectura en tiempo real. En resumen, esta clase funciona como si se tratase de la aplicación externa que llama a la librería con el sistema de posicionamiento.

El gestor de datos se encarga de crear las ventanas temporales e introducir en estas los datos correspondientes de las lecturas recibidas. Una vez se finaliza el tiempo que enmarca la ventana temporal, el gestor de datos realiza la media aritmética de los datos de la ventana temporal, de forma que se obtiene la media de aceleración y orientación en cada eje en dicho marco temporal. Este tratamiento de los datos no es del todo correcto, ya que con la media de aceleración obtenemos una distancia de desplazamiento y con la media de orientación obtenemos la dirección. Poniendo un ejemplo, si el usuario se ha desplazado realizando un arco en su desplazamiento, la media de orientación sería la dirección entre el punto inicial y el final, mientras

que el desplazamiento que se calculará con la media de aceleración será la del recorrido en arco, cuya distancia es mayor a la línea recta entre el punto inicial y el punto final.

No obstante, se realiza este tratamiento de los datos, pese a no ser del todo correcto, debido a que los datos de la ventana temporal tienen muy poca separación temporal entre si y la cantidad de datos que almacena es pequeña. Con este tratamiento se consigue reducir el coste computacional, mientras que el error producido no es grave, incluso imperceptible en la mayoría de los casos, ya que el pequeño marco de tiempo que ocupa la ventana temporal no permite realizar movimientos complejos.

Esto es muy positivo para la eficiencia del sistema de posicionamiento combinado, sobre todo si tenemos en cuenta que el posicionamiento basado en sensores inerciales está pensado para dar resultados rápidos a corto plazo con una precisión relativamente baja que serán corregidos y descartados posteriormente.

Una vez obtenida la media de los datos de la ventana temporal, se realiza la corrección de gravedad en cada eje de la media de datos del acelerómetro. Gracias a disponer de la media de la aceleración y orientación del dispositivo en un breve espacio de tiempo, se puede realizar correctamente la corrección de gravedad. Para corregir la gravedad es necesario saber la inclinación del dispositivo en cada eje, y puesto que los datos llegan cuando están disponibles, podría darse el caso de recibir los datos de la aceleración pero no de su inclinación. Es por esto que la mejor opción, tanto por sencillez como por coste computacional, es realizar medias en espacios cortos de tiempo.

Puesto que la gravedad afecta de forma distinta en cada eje dependiendo del ángulo de inclinación de este, es necesario calcular la imagen de la fuerza gravitatoria sobre cada eje. En la Figura 5.1 se puede apreciar cómo afecta la fuerza de la gravedad en sus ejes Y y Z cuando el dispositivo se inclina sobre su eje X.

La eliminación de la gravedad es muy importante, ya que es necesario eliminar la influencia que tiene esta sobre los datos reales de aceleración. En la Figura 5.2 se puede observar la aceleración real y el efecto de la gravedad y cómo estos datos afectan a los ejes del dispositivo. En el caso del eje Z del dispositivo, este devolvería como dato la suma de la imagen de la aceleración real sobre el eje Z que ocasiona la aceleración real del usuario más la imagen de la aceleración de la gravedad sobre el eje Z. En el caso del eje Y, en lugar de sumarse las aceleraciones, se restarían, ya que el sentido de las imágenes de las dos aceleraciones es opuesto.

Por tanto, basta con aplicar las Fórmulas (1), (2) y (3) para obtener la influencia de la gravedad en los ejes X, Y y Z respectivamente. En estas ecuaciones, la constante $\boldsymbol{G}$ representa la fuerza de la gravedad (aproximadamente $9.81 m/s^2$), mientras que ***pitch*** y ***roll*** representan la inclinación del dispositivo en los ejes X e Y respectivamente. Finalmente las aceleraciones reales ($A_r(x)$, $A_r(y)$ y $A_r(z)$) se obtienen restando (o sumando en el caso del eje X) la aceleración medida del eje con la aceleración causada por la gravedad en dicho eje. Tal y como se ha explicado anteriormente, el sistema AHRS (ver Sección 3.2.4) contempla un rango concreto de ángulos para evitar redundancias, lo que simplifica estas fórmulas.

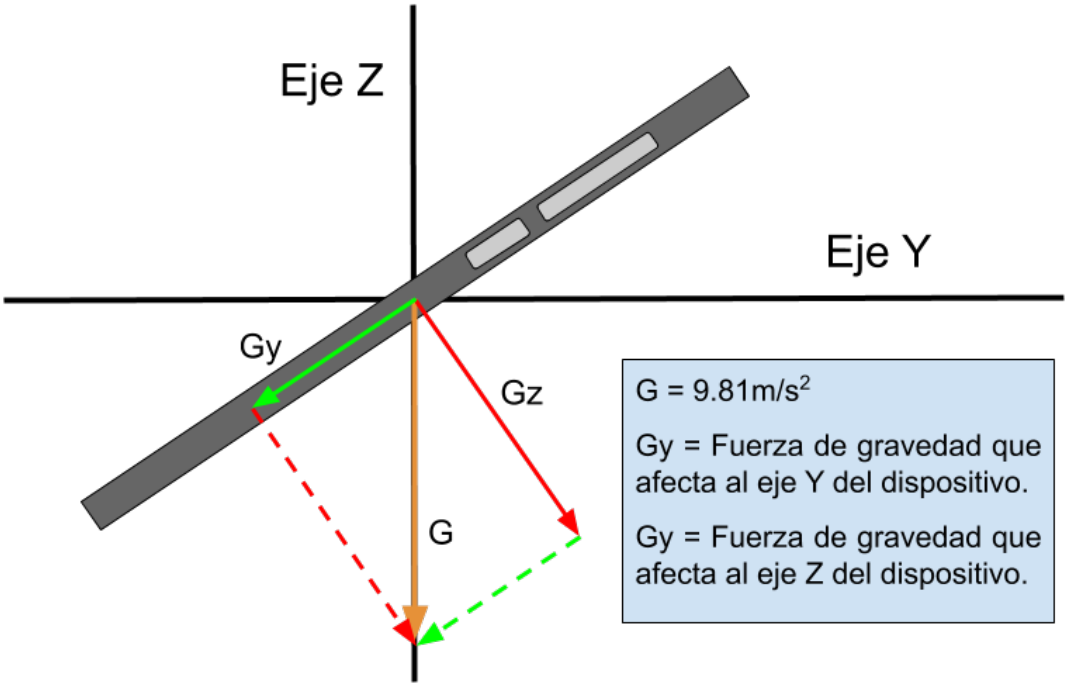

Figura 5.1: Fuerza de la gravedad en los ejes de un dispositivo inclinado sobre el eje X.

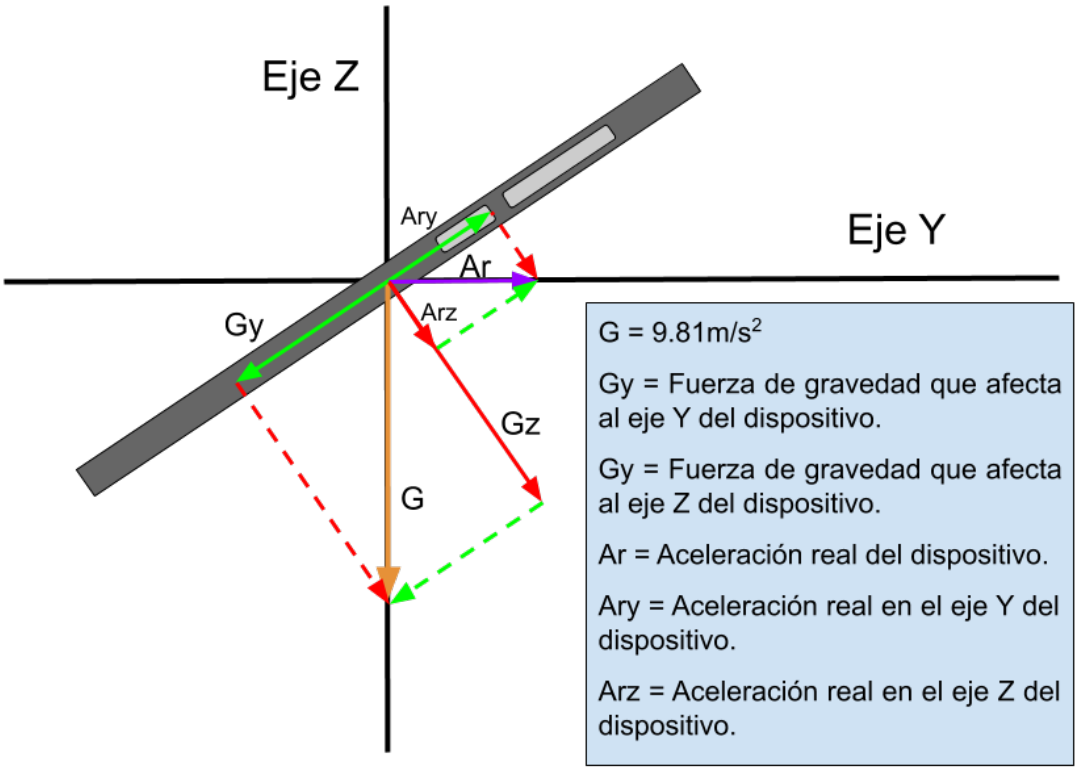

Figura 5.2: Conjunto de fuerzas que afectan al dispositivo.

$$G(x) = G \times \sin(roll\_Y) \times \cos(pitch\_X); \qquad A_r(x) = A(x) + G(x) \tag{1}$$
$$G(y) = G \times \sin(pitch\_X); \qquad A_r(y) = A(y) - G(y) \tag{2}$$
$$G(z) = G \times \sin(pitch\_X - 90) \times \sin(roll\_Y - 90); \qquad A_r(z) = A(z) - G(z) \tag{3}$$

Una vez eliminada la gravedad en los ejes, únicamente nos quedan las imágenes de la aceleración real en cada eje del dispositivo. Para calcular la aceleración final sobre el plano XY del entorno, es decir, el plano horizontal, primero hay que calcular qué aceleración sobre el eje Z se produce al caminar (lo que provoca que el movil ascienda y descienda) y cual pertenece al movimiento final sobre los ejes X e Y. Para ello se aplican las Fórmulas 4 y 5.

$$A_f(x) = A_r(x) + A_r(z) \times \cos(roll\_Y - 90) \tag{4}$$
$$A_f(y) = A_r(y) - A_r(z) \times \cos(pitch\_X - 90) \tag{5}$$

Tras calcular las aceleraciones finales sobre el plano horizontal del entorno, se calcula el vector de desplazamiento producido. Para ello, se aplican las fórmulas de movimiento rectilíneo uniformemente acelerado, ya que al tratarse de ventanas temporales muy pequeñas, podemos suponer una aceleración media constante en el pequeño margen de tiempo de la ventana temporal y partiendo de una velocidad inicial igual a cero o a la velocidad de la ventana temporal anterior, y un desplazamiento inicial igual a cero. Por tanto, se han utilizado las Fórmulas 6 y 7 para calcular el desplazamiento en cada eje y las Fórmulas 8 y 9 para calcular la velocidad media de desplazamiento en esa ventana temporal.

$$e_k(x) = V_{k-1}(x) \times t + \frac{A_f(x) \times t^2}{2} \tag{6}$$
$$e_k(y) = V_{k-1}(y) \times t + \frac{A_f(y) \times t^2}{2} \tag{7}$$
$$V_k(x) = V_{k-1}(x) + A_f(x) \times t \tag{8}$$
$$V_k(y) = V_{k-1}(y) + A_f(y) \times t \tag{9}$$

El siguiente paso tras la obtención el vector de desplazamiento, consiste en aplicar a este una rotación para ubicar correctamente el desplazamiento en el entorno, ya que el desplazamiento en los ejes X e Y son referentes al dispositivo. Para ello, se tiene en cuenta la orientación del eje Z, ya que este nos indica la dirección del norte magnético. Para realizar la rotación de un vector bidimensional se utilizan las Fórmulas 10 y 11, siendo $W = (a, b)$ el vecor de desplazamiento real y α el ángulo de rotación con respecto al norte magnético.

$$a = e_k(x) \times \cos(\alpha) - e_k(y) \times \sin(\alpha) \tag{10}$$
$$b = e_k(x) \times \sin(\alpha) + e_k(y) \times \cos(\alpha) \tag{11}$$

Finalmente, se debe obtener el punto en el que se encuentra el usuario. Por tanto, solo queda aplicar la Fórmula 12, siendo $P_k = (p1_k, p2_k)$ el punto donde se encuentra el usuario actualmente, que consiste en sumar el vector de desplazamiento actual al punto en el que se encontraba previamente el usuario.

$$P_k = (p1_{k-1} + a, p2_{k-1} + b) \tag{12}$$

En la Figura 5.3 se puede observar el diagrama de clases simplificado de la implementación del sistema de posicionamiento basado en sensores inerciales. En este diagrama se puede observar la relación que existe entre cada clase junto con sus multiplicidades y dependencias.

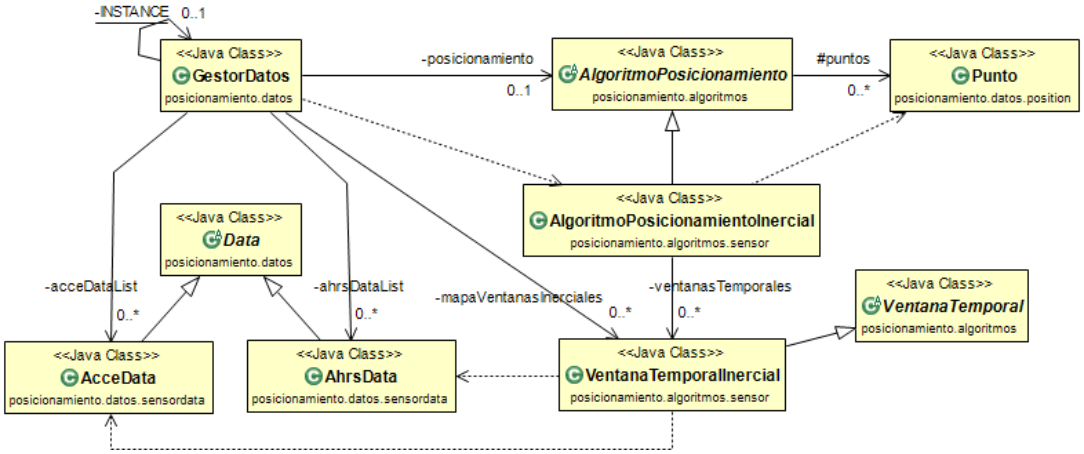

Figura 5.3: Diagrama de clases simplificado de la implementación del posicionamiento basado en sensores inerciales

Tal y como se puede observar, el gestor de datos utiliza el patrón *Singleton*, ya que se trata de una instancia única que se encarga de gestionar el sistema de posicionamiento combinado.

Además, en la clase *AlgoritmoPosicionamientoInercial* se utiliza un patrón *Facade*, es decir, existe un método que se encarga de llamar en orden a todos los métodos que implementan las operaciones mencionadas en esta sección.

También se puede apreciar que se han utilizado varias clases abstractas en lugar de utilizar interfaces. Se ha hecho de esta forma para encapsular métodos y atributos comunes con el sistema de posicionamiento basado en bluetooth de bajo consumo.

Durante la implementación de este sistema de posicionamiento han surgido varios errores que han ocasionado retrasos con respecto a la planificación inicial. Estos errores fueron principalmente errores en las fórmulas, ya que muchas de las fórmulas se consultaron inicialmente en internet y no funcionaban correctamente, tanto por ser incorrectas como por no estar bien explicadas, por lo que no se entienden bien las variables utilizadas. Fue por esto que muchas de las fórmulas se decidió calcularlas manualmente partiendo de otras fórmulas más básicas.

Además, se planteó otra forma más sencilla de implementar este sistema. Este otro método (véase [16]) consiste en analizar la variación de aceleración en los distintos ejes para detectar

los pasos del usuario. Se trata de un método que está pensado principalmente para ubicar los sensores en el pie, aunque se puede adaptar a los dispositivos inteligentes.

Este método fue descartado principalmente por un motivo, y ese motivo es la base de su funcionamiento. Puesto que este proyecto se desarrolla para funcionar en una biblioteca universitaria, es fundamental que sea funcional para la totalidad de los usuarios. Este sistema detecta los pasos de las personas, por lo que no funcionaria para aquellos usuarios cuya movilidad se encuentra restringida por el uso de una silla de ruedas por ejemplo. Es por esto que se optó por continuar utilizando el método de posicionamiento elegido inicialmente e intentar solucionar los problemas que surgían.

No obstante, el resultado final no funciona como se esperaba. A partir de aquí se han realizado tres suposiciones:

- **Fórmulas incorrectas o errores de programación:** Es posible que alguna de las fórmulas no sea correcta, falte algún paso o no se haya programado correctamente. Esto se ha revisado varias veces y parece ser la menos probable.
- **Sensores defectuosos:** El código se ha probado únicamente con los datos de un dispositivo, por lo que existe la posibilidad de que los sensores de este dispositivo no detecten correctamente las aceleraciones. No obstante, si que se detecta correctamente la gravedad en cada eje y la anulación de esta funciona correctamente, por lo que tampoco es demasiado probable.
- **Error de precisión:** Es el problema más probable. Puede que los sensores del dispositivo utilizado no sean lo suficientemente precisos o que simplemente los datos de los sensores son menos precisos de lo esperado.

Vistos los posibles orígenes del error, se planteó una solución al error más probable. Esta solución consiste en la implementación del filtro de Kalman, ya que el error de precisión puede que sea generado en su mayoría por ruido que puede ser reducido o eliminado mediante el filtro de Kalman. Puesto que esta parte del proyecto ya había supuesto retrasos, se aplazó la implementación del filtro de Kalman para llevarla a cabo antes de realizar la combinación de los sistemas de posicionamiento, aunque finalmente no se llegó a implementar.

5.2.2. Posicionamiento basado en BLE

En esta sección se detalla la implementación del sistema de posicionamiento basado en balizas bluetooth de bajo consumo (BLE). Tal y como se ha explicado en el Capítulo 3, este sistema ubica al usuario basándose en las intensidades de las señales emitidas por balizas BLE desplegadas en el entorno.

Para la implementación de este sistema se ha aprovechado parte del código utilizado en el posicionamiento basado en sensores inerciales. La clase *Main* se ha ampliado para que se encargue de enviar al gestor de datos la información del despliegue de balizas en el entorno. Además, al igual que con el sistema de posicionamiento basado en sensores inerciales, esta clase se sigue encargando de leer ficheros y enviar los datos al gestor de datos.

El gestor de datos ahora almacena la información del despliegue de las balizas mediante un mapa de *Beacons*, cuya clave es la dirección MAC de la baliza y el valor es un objeto de la clase *Beacon* que contiene toda la información relativa a este, incluyendo su posición en el despliegue. Además, el gestor de datos se encarga de gestionar las ventanas temporales relativas este sistema de posicionamiento y de llamar a la clase encargada de calcular la posición cuando la ventana temporal se cierre.

Las ventanas temporales contienen un mapa de lecturas de las balizas, cuya clave es la dirección MAC de la baliza y el valor es una lista de los valores RSSI (del inglés *Received Signal Strength Indicator*) de cada lectura de dicha baliza. Además, esta clase tiene un método que se encarga de realizar la media de los valores RSSI de cada baliza y devuelve una lista de las balizas con el valor medio de su RSSI.

Una vez se cierra la ventana temporal, el gestor de datos se encarga de llamar a la clase *AlgoritmoPosicionamientoBLE* que se encarga de estimar la posición del usuario mediante el algoritmo del centroide ponderado.

En geometría, el centroide o baricentro es la intersección que se produce al dividir un objeto en dos partes de igual volumen [17]. Cuando se obtienen los datos correspondientes a las balizas bluetooth, estas forman una figura geométrica conforme a su despliegue. Para calcular el centroide ponderado se aplican las Fórmulas (13) y (14), siendo Bx_i la componente X de la Baliza i, By_i la componente Y de la Baliza i y W_i el peso calculado para dicha baliza, para obtener el punto P(x,y) en el que se encuentra el usuario.

$$x = \frac{\sum Bx_i \times W_i}{\sum W_i} \tag{13}$$

$$y = \frac{\sum By_i \times W_i}{\sum W_i} \tag{14}$$

Para calcular el peso de cada baliza se utiliza el valor RSSI. Este valor nos indica la fuerza con la que se ha recibido la señal en decibelios. Existe una fórmula para calcular la distancia a la que se encuentra la baliza sabiendo el valor RSSI y la potencia que se recibe a 1 metro de distancia. A pesar de que se conocen estos valores y se podría utilizar esta fórmula para obtener la distancia y posicionar por medio de la trilateración, no se utiliza este método ya que es menos eficiente que el centroide ponderado. Para realizar la trilateración se debería calcular la distancia a la que se encuentra cada baliza para después aplicar el algoritmo de trilateración, con lo que el coste computacional es significativamente mas alto que la obtención de pesos y el calculo del centroide ponderado.

Para obtener el peso de una baliza se aplica la Fórmula (15) presentada en [18]. Esta es una fórmula sencilla de implementar y con un coste computacional muy bajo, al igual que las Fórmulas (13) y (14) para calcular el centroide ponderado. Es la eficiencia y el bajo coste de este algoritmo el motivo por el cual es la mejor opción para este sistema de posicionamiento.

$$W_i = 10^{(\frac{RSSI_i}{10})} \tag{15}$$

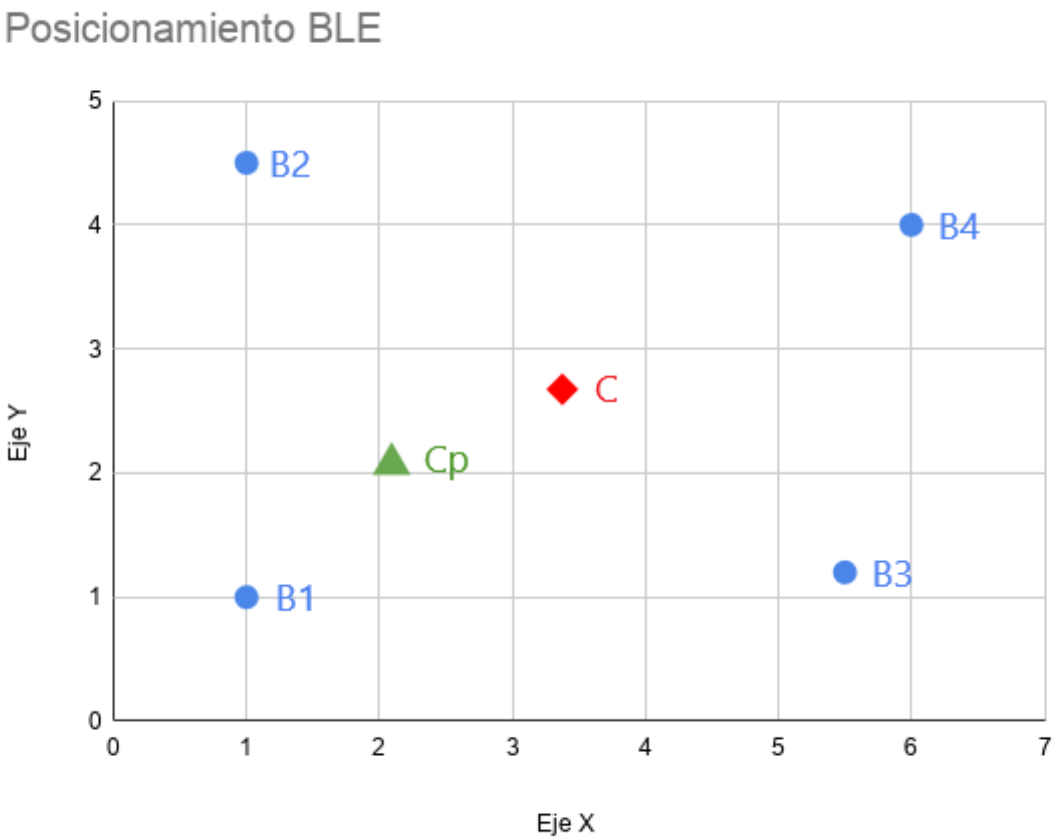

Figura 5.4: Despliegue de 4 balizas (B1, B2, B3 y B4), el centroide de la figura geométrica que forman (C) y la posición real del usuario estimada por el centroide ponderado (Cp)

Id	**X**	**Y**	**Peso**
B1	1	1	80
B2	1	4.5	30
B3	5.5	1.2	18
B4	6	4	15
C	3.375	2.675	-
Cp	2.091	2.074	-

Tabla 5.1: Datos de los puntos de la Figura 5.4

En la Figura 5.4 se muestra un ejemplo del despliegue de 4 balizas bluetooth (B1, B2, B3 y B4) junto con el centroide (C) de la figura geométrica que forman y el centroide ponderado (Cp) donde se estima que se encuentra el usuario. En la Tabla 5.1 se encuentran los valores de todos estos puntos, incluyendo el peso de ejemplo que se ha utilizado en el caso de las balizas BLE.

En la Figura 5.5 se muestra el diagrama de clases simplificado de la implementación del sistema de posicionamiento basado en BLE, donde se puede apreciar la relación entre las clases y sus dependencias.

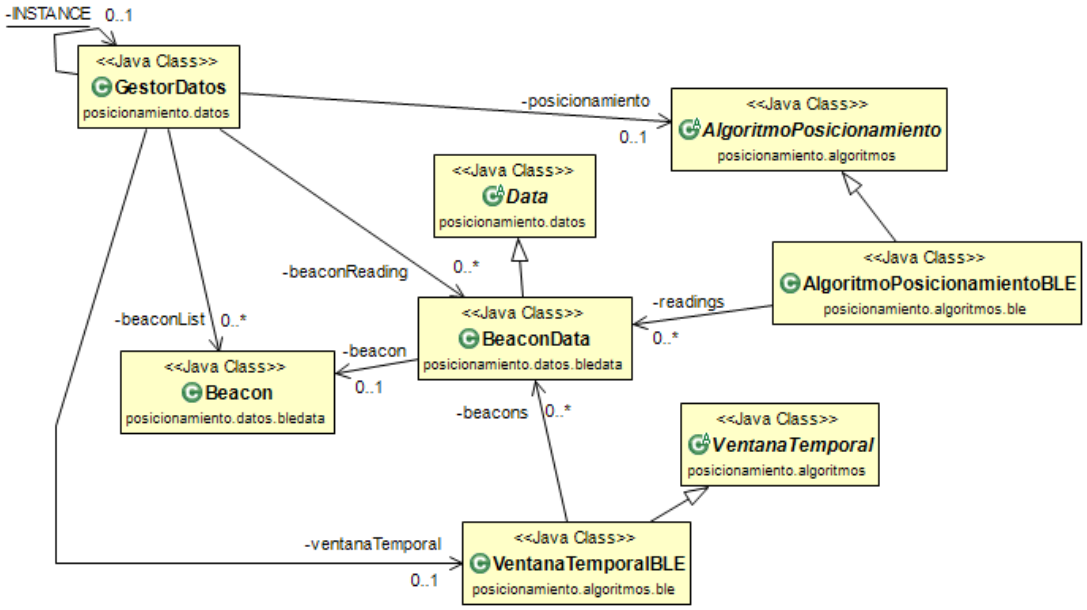

Figura 5.5: Diagrama de clases simplificado de la implementación del posicionamiento basado en BLE

Al igual que con el sistema de posicionamiento basado en sensores inerciales, el gestor de datos sigue utilizando el patrón *Singleton*. No obstante, en la implementación de *AlgoritmoPosicionamientoBLE* no se sigue el patrón *Facade*, ya que no se realiza un trato tan exhaustivo de los datos como en el otro sistema.

La implementación de este sistema de posicionamiento no ha supuesto apenas errores y los resultados obtenidos son bastante precisos.

5.2.3. Filtro de Kalman

El filtro de Kalman no se ha llegado a implementar en el proyecto debido a la falta de tiempo provocada por los retrasos en el sistema de posicionamiento basado en sensores inerciales. No obstante, si que se ha estudiado su funcionamiento y se ha llegado a plantear su implementación.

El filtro de Kalman se implementa en el sistema de posicionamiento basado en sensores inerciales para conseguir una mayor precisión y evitar los errores provocados por el ruido de estos sensores.

En la Figura 5.6 se puede observar el esquema de funcionamiento del filtro de Kalman. Este esquema muestra de una forma más detallada los pasos que sigue el filtro de Kalman explicados en el Capítulo 3.

Se entiende por estado actual aquel que contiene la información del punto en el que se encuentra el usuario X(k) y a su matriz de covarianza P(k). La covarianza indica el grado de variación conjunta de dos variables aleatorias con respecto a sus medias, es decir, nos aporta información para determinar si existe o no una relación entre ambos datos [19]. En la Fórmula

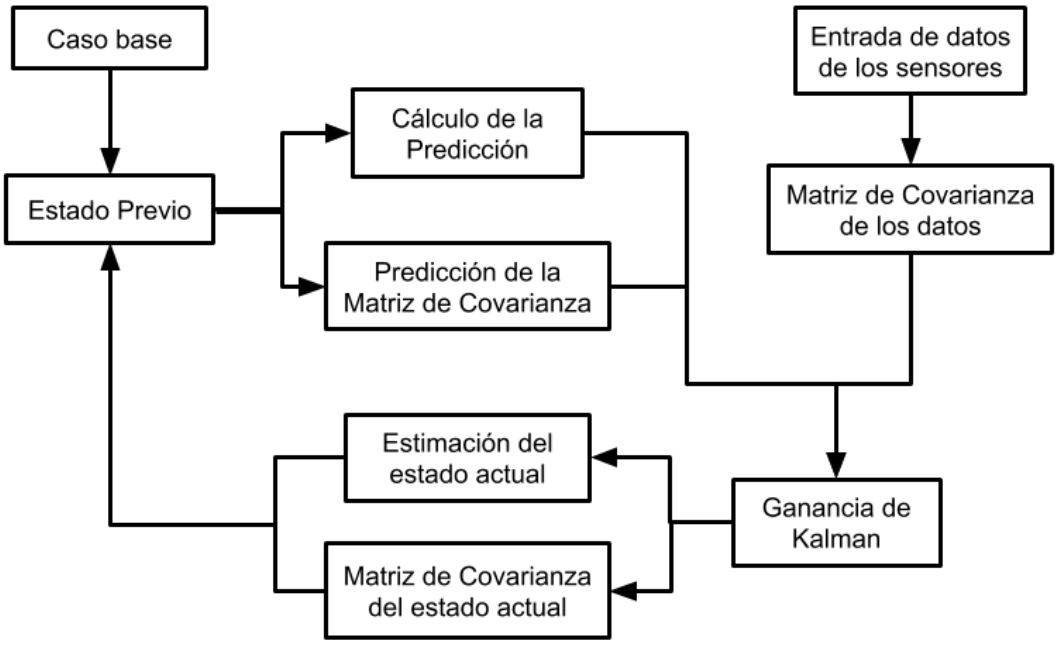

Figura 5.6: Esquema de la secuencia de pasos del filtro de Kalman

(16) se muestra la forma en la que se calcula la covarianza, siendo $\overline{x}$ la media de los datos en x, x_i el dato concreto en x y N la cantidad de datos.

$$\sigma_x \sigma_y = \frac{\sum_{i=1}^{N} (\overline{x} - x_i)(\overline{y} - y_i)}{N} \tag{16}$$

Si el valor de la covarianza es cero, esto significa que no existe relación entre las variables. Si el valor de la covarianza es mayor a cero, esto significa que a grandes valores de x le corresponden grandes valores de y. Por el contrario, si el valor de la covarianza es negativo, esto quiere decir que a grandes valores de x le corresponden pequeños valores de y y viceversa.

Puesto que el sistema está pensado para funcionar sobre tres ejes, se van a representar las operaciones por medio de matrizes. Por tanto, la matriz de covarianza será la Matriz (17).

$$Matriz\ de\ Covarianza = \begin{pmatrix} \sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z \\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2 \end{pmatrix} \tag{17}$$

A continuación, se van a explicar los pasos a la hora de implementar el filtro de Kalman:

1. **Caso base:** Caso desde el que se inicia el funcionamiento del filtro de Kalman. Se utilizarán los datos de la primera posición del usuario calculada por el sistema de posicionamiento basado en BLE.

2. **Predicción:** Se realiza una predicción del estado actual del sistema mediante el caso previo o el caso base

   a) **Cálculo de la predicción:** Mediante la Fórmula (18) se calcula la predicción del estado actual, siendo A la matriz que realiza el cálculo de la posición actual teniendo en cuenta la velocidad enterior en cada eje, G_k la matriz que anula la fuerza de la gravedad en las aceleraciones y W_k la matriz que corrige el ruido causado por elementos no considerados (se puede elegir un valor simbólico o eliminar esta matriz).

   $$X'_k = A \times X_{k-1} + G_k + W_k \tag{18}$$

   b) **Predicción de la matriz de covarianza:** Se realiza la predicción de la matriz de covarianza mediante la Fórmula (19), siendo Q la matriz de ruido de covarianza del proceso, que se encarga de evitar que la predicción de la covarianza sea cero en caso de que la covarianza del estado previo sea cero.

   $$P'_k = A \times P_{k-1} + Q \tag{19}$$

3. **Entrada de datos:** Se obtiene la matriz de los datos recibidos por los sensores inerciales (Y_k).

4. **Matriz de covarianza de los datos:** La matriz R se obtiene calculando la covarianza de los datos actuales comparados con los previos.

5. **Ganancia de Kalman:** Mediante las matrizes de covarianza P'_k y R, pondera para decidir cual de las dos es más fiable, la predicción o la medida. La ganancia de Kalman (K) se calcula utilizando la Fórmula (20)

   $$K = \frac{P'_k}{P'_k + R} \tag{20}$$

6. **Corrección:** Se estima el estado actual y su matriz de covarianza.

   a) **Estimación del estado actual:** Solución tras el filtrado de los datos. Contiene los datos de la posición actual del usuario. Para obtener esta estimación se utiliza la Fórmula (21)

   $$X_k = X'_k + K \times (Y_k - X'_k) \tag{21}$$

   b) **Matriz de covarianza del estado actual:** Muestra la dispersión entre los valores estimados con respecto a la media, por lo que se puede utilizar para saber si la estimación es buena o no. Para hallar esta matriz se utiliza la Fórmula (22).

   $$X_k = X'_k + K \times (Y_k - X'_k) \tag{22}$$

7. **Estado previo:** El estado actual pasa a ser el estado previo para la próxima predicción.

5.2.4. Combinación de los sistemas de posicionamiento

No se ha llegado a realizar la combinación de los sistemas debido a la falta de tiempo. No obstante, si que se han adaptado ligeramente algunas clases para facilitar esta combinación en un futuro.

A pesar de no contar con la implementación, se va a aprovechar esta sección para explicar como se debería realizar esta combinación de sistemas.

El primer paso sería adaptar el gestor de datos para descartar los datos de los sensores inerciales hasta que consiga cerrar la primera ventana temporal de datos BLE. Una vez obtenida la primera posición del usuario, se utilizarían estos datos como caso base del filtro de Kalman para el sistema de posicionamiento basado en sensores inerciales.

A partir de este punto, el sistema combinado se encargaría de recibir los datos y realizar las ventanas temporales con normalidad, a excepción de corregir o reiniciar los datos de los sensores inerciales cada vez que se cierre una ventana temporal de datos BLE. Esto se haría de esta forma para evitar que el error que ocasionan los sensores inerciales se viese incrementado exponencialmente con el paso del tiempo, ya que con los datos BLE podemos obtener una ubicación precisa cada uno o dos segundos.

5.3. Verificación y validación

En esta sección se detallarán las pruebas de verificación y validación que se han llevado a cabo para el desarrollo del proyecto. En primer lugar, se hablará de la preparación del entorno de pruebas, el cual afectará principalmente al sistema de posicionamiento basado en bluetooth de bajo consumo (BLE).

5.3.1. Preparación del entorno de pruebas

Tal y como se ha explicado en el seguimiento del proyecto en el Capítulo 2, las pruebas se iban a realizar en la Biblioteca de la UJI y en el edificio Espaitec 2, pero debido a la situación de emergencia sanitaria esto ha sido imposible.

Para solventar este problema, se decidió añadir la tarea de estudiar un nuevo entorno de pruebas y realizar el despliegue de balizas BLE. Esto ha sido posible gracias a que el supervisor de la empresa me pudo hacer llegar hasta 8 de estas balizas BLE para realizar el despliegue en mi casa.

El entorno de pruebas que se ha utilizado ha sido el de mi habitación. Para ello, se ha adaptado un entorno relativamente abierto en una zona de la habitación y se ha realizado a mano un mapa a escala de la misma marcando puntos de referencia tanto en el mapa como en el suelo de la propia habitación. Una vez realizado el despliegue, también se ha marcado en el mapa la ubicación de cada baliza.

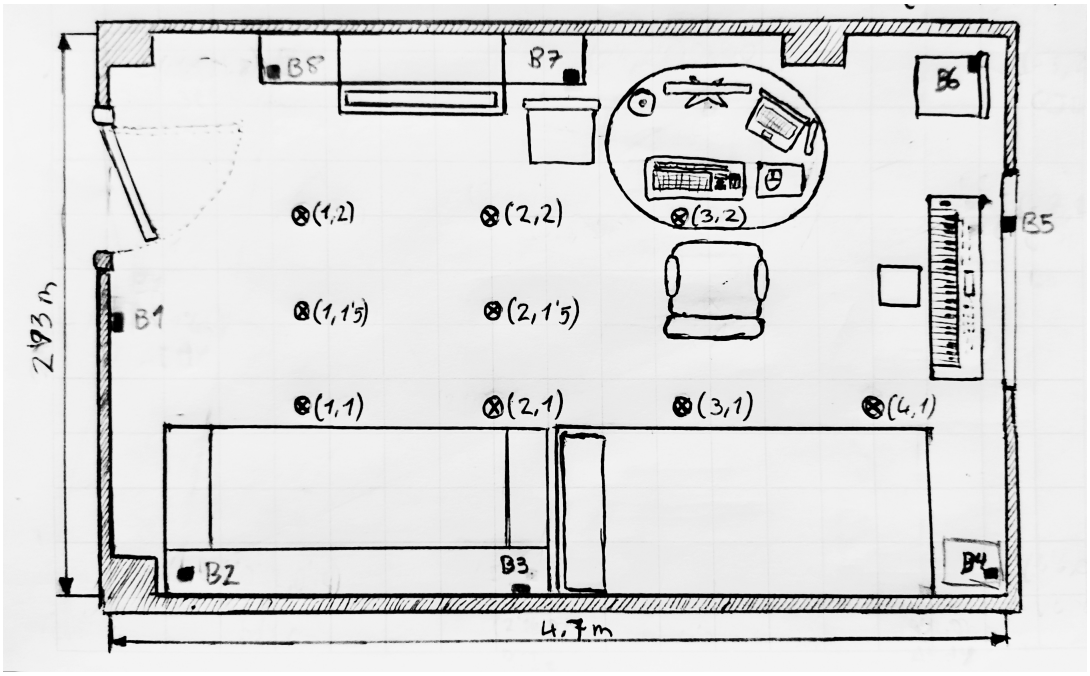

Figura 5.7: Mapa del nuevo entorno de pruebas con el despliegue de balizas.

Id	X	Y
B1	0.10	1.46
B2	0.42	0.14
B3	2.14	0.14
B4	4.60	0.10
B5	4.70	1.93
B6	4.50	2.75
B7	2.40	2.60
B8	0.85	2.68

Tabla 5.2: Despliegue de balizas BLE de la Figura 5.7

La Figura 5.7 contiene el mapa mencionado previamente con toda la información del despliegue. Además, en la Tabla 5.2 se indica la posición de cada baliza desplegada. En este mapa se han marcado las balizas mediante cuadrados junto con su Id, y se han indicado puntos de referencia como por ejemplo el punto (1,1) mediante circulos con una x en su interior.

5.3.2. Sensores inerciales

En esta sección se detallan las pruebas de verificación y validación que se han llevado a cabo en el sistema de posicionamiento basado en sensores inerciales. Se han realizado pruebas unitarias para comprobar el funcionamiento de dos métodos concretos de este sistema de posicionamiento. Puesto que lo que se pretende comprobar es que las fórmulas son correctas, se han introducido manualmente valores calculados previamente en lugar de datos reales de los sensores.

El primer método para el que se realizaron test fue el método encargado de eliminar la gravedad que afecta al acelerómetro. Para ello, se prepararon 3 escenarios para comprobar la eliminación de la gravedad cuando afecta la totalidad de esta a un único eje y 1 escenario para comprobar la eliminación de la gravedad cuando el dispositivo se encuentra inclinado de forma que la gravedad afecta a dos ejes.

Para realizar estas pruebas se ha creado una ventana temporal con los datos medios de la orientación y de las aceleraciones calculados previamente. Posteriormente, se llama al método encargado de corregir la gravedad, realiza los cálculos programados y se comprueba que el resultado sea cero en las aceleraciones de todos los ejes. En la Figura 5.8 se puede observar como el código pasa las pruebas y además se muestra el resultado devuelto por el escenario que prueba la corrección de la gravedad cuando esta afecta a dos ejes.

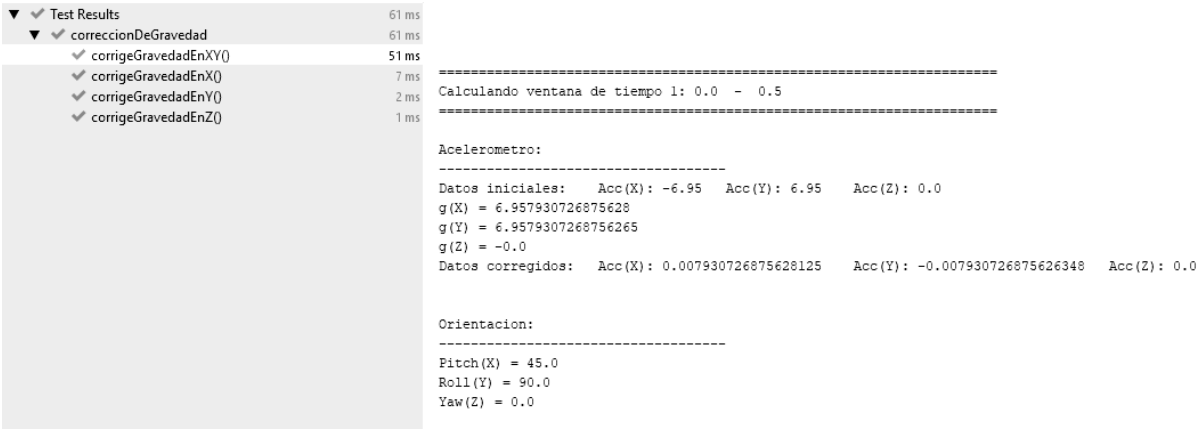

Figura 5.8: Resultado de la ejecución de test para comprobar la corrección de la gravedad.

El segundo método para el que se realizaron test fue el método encargado de calcular el desplazamiento del usuario. Para ello se prepararon 2 escenarios. El primer escenario se había calculado de forma que el usuario había avanzado 4 metros en el eje X, por lo que se crearon 6 ventanas temporales para cumplir esta condición y comprobar si finalizada la última ventana temporal, el usuario se ubicaba en el punto (4,0). El segundo escenario se preparó de la misma forma, pero se calculó de forma que el usuario retrocedía 1 metro en el eje Y, y después volvía al punto (0,0). Ambos escenarios pasan las pruebas y en la Figura 5.9 se muestran los puntos que ha calculado este método en cada escenario.

Además, como ya se ha mencionado anteriormente, la clase *Main* se utiliza para simular las llamadas de una aplicación externa al sistema de posicionamiento. Para hacer esto, se han generado varios ficheros con datos reales recogidos mediante la aplicación *GetSensorsData* mencionada en la primera sección de este capítulo.

Cuando se ejecuta la clase *Main* esta muestra varias opciones a elegir entre los distintos ficheros. Todos estos ficheros tienen comentarios que especifican los movimientos que se han realizado, incluyendo dirección y tiempo de desplazamiento entre puntos clave. En la Figura 5.10 se muestran las opciones disponibles, junto con una breve explicación del movimiento realizado. Una vez se elige el fichero, se ejecuta el código correspondiente, mostrando por consola varios datos e información para realizar un seguimiento del proceso de estimación y, finalmente, muestra todos los puntos que ha dibujado (uno por cada ventana temporal).

Desplazamiento en X

```
----------------------------------------
Puntos dibujados:
----------------------------------------

Punto 1: (-0.25, 0.0)       Tiempo: 0.5
Punto 2: (-1.0, 0.0)        Tiempo: 1.0
Punto 3: (-2.0, 0.0)        Tiempo: 1.5
Punto 4: (-3.0, 0.0)        Tiempo: 2.0
Punto 5: (-4.0, 0.0)        Tiempo: 2.5
```

Desplazamiento en Y

```
----------------------------------------
Puntos dibujados:
----------------------------------------

Punto 1: (0.0, -0.25)       Tiempo: 0.5
Punto 2: (0.0, -0.75)       Tiempo: 1.0
Punto 3: (0.0, -1.0)        Tiempo: 1.5
Punto 4: (0.0, -0.75)       Tiempo: 2.0
Punto 5: (0.0, -0.25)       Tiempo: 2.5
Punto 6: (0.0, 0.0)     Tiempo: 3.0
```

Figura 5.9: Resultado de la ejecución de test para comprobar el cálculo del desplazamiento.

```
Elige el fichero que quieres porcesar (opcion 1 por defecto, 0 para salir):

Sensores inerciales:
----------------------------
1 - Recogida de datos (movil quieto sobre la mesa)
2 - Gravedad en Y (movil en vertical)
3 - Gravedad en X (movil en horizontal)
4 - Gravedad en X e Y (45 grados en pitch y roll)
5 - Gravedad lenta (rotaciones lentas en movil)
6 - Gravedad rapida (rotaciones rapidas en movil)
7 - Desplazamiento de 3 metros en eje Y (movil inclinado en posicion natural)
```

Figura 5.10: Opciones de ficheros para realizar pruebas del sistema de posicionamiento basado en sensores inerciales.

5.3.3. BLE

En esta sección se detallan las pruebas de verificación y validación que se han llevado a cabo en el sistema de posicionamiento basado en bluetooth de bajo consumo (BLE).

Puesto que este sistema no tiene a penas tratamiento de datos ni filtros, se decidió no realizar pruebas unitarias, ya que no había ningún método ni parte del código que fuese necesaria la comprobación de su funcionamiento.

Es por esto que directamente se han hecho pruebas a partir de ficheros con datos reales recogidos mediante la aplicación *GetSensorsData*. Esto funciona de la misma forma que con el sistema de posicionamiento basado en sensores inerciales. En la Figura 5.11 se muestran los ficheros que se pueden elegir para realizar estas pruebas, y en la Figura 5.12 se muestran los puntos al ejecutar la prueba 14. Como se puede observar, los resultados estiman de forma bastante precisa la posición del usuario.

```
Elige el fichero que quieres porcesar (opcion 1 por defecto, 0 para salir):

Sensores inerciales:
----------------------------
1 - Recogida de datos (movil quieto sobre la mesa)
2 - Gravedad en Y (movil en vertical)
3 - Gravedad en X (movil en horizontal)
4 - Gravedad en X e Y (45 grados en pitch y roll)
5 - Gravedad lenta (rotaciones lentas en movil)
6 - Gravedad rapida (rotaciones rapidas en movil)
7 - Desplazamiento de 3 metros en eje Y (movil inclinado en posicion natural)


Bluetooth Low Energy (BLE):
----------------------------
10 - Recogida de datos (movil quieto sobre la mesa)
11 - BLE movil en punto (3,2)
12 - BLE movil en punto (3,2) (8 Balizas)
13 - BLE movil en punto (1,1) (8 Balizas)
14 - BLE movil en movimiento (4,1)-->(1,1)-->(1,2)-->(2,2) (16 seg)
```

Figura 5.11: Opciones de ficheros para realizar pruebas de los sistemas de posicionamiento.

```
----------------------------------------
Puntos dibujados:
----------------------------------------

Punto 1: (4.0871, 1.1675)       Tiempo: 0.22
Punto 2: (4.0871, 1.1675)       Tiempo: 1.16
Punto 3: (4.0871, 1.1675)       Tiempo: 2.1
Punto 4: (4.0871, 1.1675)       Tiempo: 3.2
Punto 5: (3.7543, 0.9347)       Tiempo: 4.05
Punto 6: (2.1943, 0.3586)       Tiempo: 5.12
Punto 7: (1.4794, 0.3213)       Tiempo: 6.03
Punto 8: (1.0134, 0.2716)       Tiempo: 7.17
Punto 9: (0.7673, 0.2927)       Tiempo: 8.07
Punto 10: (0.8653, 0.4312)      Tiempo: 9.17
Punto 11: (1.0386, 0.5446)      Tiempo: 10.09
Punto 12: (0.425, 0.9151)       Tiempo: 11.18
Punto 13: (0.5278, 1.42)        Tiempo: 12.09
Punto 14: (0.6896, 2.2043)      Tiempo: 13.02
Punto 15: (1.0938, 2.2568)      Tiempo: 14.18
Punto 16: (1.5294, 2.1181)      Tiempo: 15.07
```

Figura 5.12: Puntos dibujados al ejecutar la prueba 14 de la Figura 5.11.

Capítulo 6

Conclusiones

En este proyecto se ha intentado desarrollar un sistema de posicionamiento combinando dos sistemas ya existentes, los cuales son el posicionamiento basado en sensores inerciales y el posicionamiento basado en el despliegue de balizas bluetooth de bajo consumo. Este sistema ha sido pensado para funcionar en la Biblioteca de la Universitat Jaume I de Castellón mediante la aplicación Biblioteca UJI desarrollada por Ubik G.S. con el objetivo de mejorar la experiencia de los usuarios de la biblioteca a la hora de buscar cabinas o libros.

No obstante, el sistema combinado no ha sido finalizado debido principalmente a los problemas ocasionados por el sistema de posicionamiento basado en sensores inerciales. Esto quiere decir que, desde un punto de vista técnico, no se ha conseguido lograr todos los objetivos planteados al inicio del proyecto.

Con respecto a la gestión del proyecto, a pesar de no ser del todo satisfactoria, si que se ha realizado una gestión aceptable de este. Las tareas que se propusieron en un inicio se plantearon para tener una duración total de 300 horas. No obstante, la situación de emergencia sanitaria ha complicado significativamente la estancia en práctias y se han tenido que añadir nuevas tareas que no estaban previstas en un principio. Pese a todo esto y a que la implementación no está completa, cabe destacar que sí que se ha realizado un estudio exhaustivo de todas las tecnologías necesarias para el desarrollo del proyecto.

Por último, a nivel personal considero que este proyecto me ha aportado bastantes conocimientos tanto técnicos como transversales, así como las herramientas para enfrentarme a problemas y desafíos que surgen a la hora de desarrollar un proyecto como este. Por otra parte, se ha echado en falta trabajar en equipo y la integración en un entorno de trabajo, ya que la mayor parte de la estancia en prácticas se ha realizado a distancia a causa de la situación de emergencia sanitaria causada por la CoVid-19.

Bibliografía

[1] Ubik Geospatial Solutions S.L. http://www.ubikgs.com/. [Consulta: 13 de Junio de 2020].

[2] Biblioteca Universitat Jaume I - Aplicación de Google Play. [Consulta: 11 de Junio de 2020 - No disponible actualmente].

[3] Lyons BE, Austin D, Seelye A, Petersen J, Yeargers J, Riley T, Sharma N, Mattek N, Wild K, Dodge H and Kaye JA. Pervasive computing technologies to continuously assess Alzheimer's disease progression and intervention efficacy. https://www.frontiersin.org/articles/10.3389/fnagi.2015.00102/full. [Consulta: 11 de Junio de 2020].

[4] Buscar y comparar sueldos - indeed. https://es.indeed.com/salaries. [Consulta: 24 de Julio de 2020].

[5] JetBrains IntelliJ IDEA. https://www.jetbrains.com/idea/. [Consulta: 24 de Julio de 2020].

[6] MagicDraw - No Magic. https://www.nomagic.com/products/magicdraw. [Consulta: 24 de Julio de 2020].

[7] Antonio Sabán. Cómo funcionan el acelerómetro y el giroscopio de los móviles. https://hipertextual.com/2016/08/acelerometro-giroscopio. [Consulta: 13 de Julio de 2020].

[8] Wikipedia contributors. Magnetometer — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Magnetometer&oldid=967541754, 2020. [Consulta: 20 de Julio de 2020].

[9] Blackstone, Austin. Understanding the different types of BLE Beacons. https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/. [Consulta: 20 de Julio de 2020].

[10] Smart Beacon SB16-2. https://store.kontakt.io/our-products/30-smart-beacon-sb16-2.html. [Consulta: 20 de Julio de 2020].

[11] Wikipedia contributors. Ibeacon — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=IBeacon&oldid=949727972, 2020. [Consulta: 20 de Julio de 2020].

[12] Toulson, Simon. iBeacon parameters: UUID, Major and Minor. https://support.kontakt.io/hc/en-gb/articles/201620741-iBeacon-Parameters-UUID-Major-and-Minor. [Consulta: 20 de Julio de 2020].

[13] Ziobro, Adrian. iBeacon advertising packet structure. https://support.kontakt.io/hc/en-gb/articles/201492492-iBeacon-advertising-packet-structure. [Consulta: 20 de Julio de 2020].

[14] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. https://doi.org/10.1115/1.3662552, Marzo 1960.

[15] Gutiérrez Gallardo, Juan Diego. GetSensorData_Android - Código en GitLab. https://gitlab.com/getsensordatasuite/getsensordata_android. [Consulta: 23 de Julio de 2020].

[16] Jiménez Ruiz, Antonio Ramón. Pedestrian Dead-Reckoning(PDR) Tutorial. https://lopsi.weebly.com/uploads/2/5/6/4/25642375/pedestrian_dead-reckoning__pdr__tutorial_ipin2017.pdf. [Consulta: 25 de Julio de 2020].

[17] Wikipedia. Centroide — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Centroide&oldid=121142697, 2019. [Consulta: 28 de Julio de 2020].

[18] Elena Simona Lohan, Jukka Talvitie, Pedro Silva, Henri Nurminen, Simo Ali-Löytty, and Robert Piché. Received signal strength models for WLAN and BLE-based indoor positioning in multi-floor buildings. https://www.researchgate.net/publication/308834363_Received_signal_strength_models_for_WLAN_and_BLE-based_indoor_positioning_in_multi-floor_buildings, 06 2015. [Consulta: 28 de Julio de 2020].

[19] Wikipedia. Covarianza — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Covarianza&oldid=126121025, 2020. [Consulta: 29 de Julio de 2020].