Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Implementación Android de algoritmos de fusión sensorial para navegación de personas en espacios interiores extensos mediante el teléfono móvil

Autor: Sergio Pérez Bachiller

Tutor: Mª del Carmen Pérez Rubio

Cotutor: David Gualda Gómez

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Implementación Android de algoritmos de fusión sensorial para navegación de personas en espacios interiores extensos mediante el teléfono móvil

Autor: Sergio Pérez Bachiller

Tutor/es: M.ª del Carmen Pérez Rubio

David Gualda Gómez

TRIBUNAL:

Presidente: **D. J. Antonio Jiménez Calvo**

Vocal 1º: **D. Luciano Boquete Vázquez**

Vocal 2º: **Dª. M.ª del Carmen Pérez Rubio**

CALIFICACIÓN: ……

FECHA: ……

A mis padres, a mi hermano y a Elena…

AGRADECIMIENTOS

Para empezar, agradecer a mi tutora Mª Carmen, y a mi cotutor David, por el apoyo y la ayuda que me han proporcionado a lo largo del desarrollo de este trabajo. Sin la forma de guiarme y sus consejos este trabajo no habría sido posible.

En segundo lugar, quería agradecer al grupo de investigación GEINTRA, especialmente al grupo US&RF, por brindarme una oportunidad única de realizar este trabajo unido a un grupo de investigación, donde he aprendido y me he formado continuamente. Quería agradecer especialmente el trato recibido por parte tanto de cada uno de los profesores que integran este grupo, como de los compañeros que he tenido en el mismo.

Quería agradecer también a todos aquellos que empezaron como compañeros de carrera, y que a día de hoy son grandes amigos, los cuales han hecho más ameno cada uno de los cursos realizados. También quería agradecer especialmente a todo el resto de mis amigos que como durante la mayor parte de mi vida, han estado siempre dispuestos a ofrecerme una parte de su tiempo para desconectar, descansar y vivir momentos inolvidables cuando más lo necesitaba.

Por último y posiblemente el agradecimiento más especial, me gustaría dárselo a mi familia, especialmente a mis padres y a mi hermano, los cuales han estado a mi lado siempre, ofreciéndome todo el cariño y el apoyo necesario en cada una de las decisiones que he ido tomando a lo largo de mi vida hasta llegar a este punto, gracias a ellos el recorrido y superar los momentos más difíciles ha sido mucho más sencillo. Como no, junto a mis padres y mi hermano el agradecimiento especial también es para ella, Elena, quien ha confiado ciegamente en mí, en cada uno de los momentos vividos a lo largo de este camino, estando a mi lado y apoyándome incondicionalmente en los momentos más difíciles y en alguna de las decisiones más importantes de mi vida, gracias.

RESUMEN

Este Trabajo Fin de Grado (TFG) presenta una aplicación Android que permite de forma robusta y flexible integrar información sensorial de las señales emitidas por un conjunto de Sistemas de Posicionamiento Locales Ultrasónicos (U-LPS), con la de sensores inerciales de una unidad de medición inercial (IMU) portada por el usuario, estimando la posición de personas a través de sus dispositivos portables en zonas interiores extensas.

Los U-LPSs se ubican en las zonas que demandan mayor precisión (como entradas y salidas), mientras que la navegación de un U-LPS a otro se realiza por medio de los sensores inerciales, corrigiendo la trayectoria y el error acumulativo de los mismos al entrar de nuevo en el área de cobertura U-LPS. Se obtiene así una aplicación flexible, capaz de ajustarse a las demandas de un entorno extenso. Los U-LPSs constan de cinco transductores cuya emisión, a una frecuencia de 41.67kHz, se multiplexa en el tiempo y codifica para conseguir autentificación y reducir las interferencias por acceso múltiple.

En la recepción, un número no limitado de usuarios con sus dispositivos móviles podría calcular su posición de forma autónoma. La captura de la señal ultrasónica se realiza mediante un módulo externo que digitaliza la señal y la envía al dispositivo portable para su procesamiento. Paralelamente, se capturan los datos de la IMU y, mediante un Filtro de Kalman Extendido, se fusionan ambas medidas y se corrigen los posibles errores acumulativos de estos últimos.

Por último, la posición estimada del usuario se representa en tiempo real en el dispositivo portable a través de los mapas de interiores proporcionados por Google Maps.

Palabras Claves: Fusión sensorial, Android, posicionamiento en interiores, U-LPS, señales ultrasónicas, sensores inerciales.

ABSTRACT

This work presents an Android application that allows estimating the position of the device in a robust and flexible way combining ultrasound signals emitted by a set of Ultrasonic Local Positioning Systems (U-LPSs), with inertial information obtained by an Inertial Measurement Unit (IMU) carried by the user and communicated via Bluetooth with the portable device.

The U-LPSs are located in areas that require a high accuracy (such as entrances and waypoints) and the trajectory between U-LPSs where there is not ultrasound (US) coverage is estimated by the inertial sensors. When the user is inside an U-LPS area again, the trajectory and cumulative error of the inertial sensor are corrected. Therefore, this proposal provides a flexible and robust application, which is capable to be adjusted to the demands of a large environment, reducing the number of U-LPSs to place. Each U-LPS consists of five transducers or beacons whose emissions, at a frequency of 41.67kHz, are multiplexed in time and encoded to achieve authentication and reduce interferences by multiple access. At reception, an unlimited number of users with their mobile devices could calculate their position autonomously.

The ultrasonic signals are captured by an external module that digitizes the signals and sends them to the portable device for processing. At the same time, IMU data is captured and, using an Extended Kalman Filter (EKF), both measures are fused, and possible errors accumulated by the IMU are corrected.

Then, the user`s position is represented on the screen of the portable device by using indoor Google Maps.

Keywords: Sensor fusion, Android, indoor positioning, U-LPS, ultrasonic signals, inertial sensors.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE ACRÓNIMOS Y SIMBOLOS

APP	Aplicación.
LBS	Servicios Basados en Localización.
GNSS	Sistemas de Navegación Global.
GPS	Sistema de Posicionamiento Global.
RF	Radiofrecuencia.
U-LPS	Sistema de Posicionamiento Local Ultrasónico.
US	Señales Ultrasónicas.
TDOA	*Time Difference of Arrival.*
IMU	Unidad de Medición Inercial.
KF	Filtro de Kalman.
EKF	Filtro de Kalman Extendido.
API	Interfaz de Programación de Aplicación.
AC	Auto Correlación.
CC	Correlación Cruzada.
DAC	Conversor Digital Analógico.
TDMA	Acceso Múltiple por División de Tiempo.
MAI	Interferencias por Acceso Múltiple.
CDMA	Acceso Múltiple por División de Código.
FFT	*Fast Fourier Transform.*
PDR	*Pedestrial Dead Reckoning.*
ECG	Electrocardiograma.
EMG	Electromiografía.
ANR	Aplicación No Responde.
MBS	*Most Significant Byte.*
LBS	Low Significant Byte.
SL	*Step Lenght.*
CDF	*Cumulative Distribution Function.*

θ	Orientación.
$\Delta\theta$	Variación de la orientación.
$\Delta\odot$	Amplitud del *Pitch.*
$\odot$	Ángulo *Pitch.*
ϕ	Ángulo *Roll.*
Ψ	Ángulo *Yaw.*
$\widehat{\odot}$	Estimación Ángulo *Pitch.*
$\widehat{\phi}$	Estimación Ángulo *Roll.*
$\widehat{\Psi}$	Estimación Ángulo *Yaw.*
α	Acelerómetros IMU.
ω	Giróscopos IMU.
σ	Desviación típica.

CAPÍTULO 1: INTRODUCCIÓN

1.1 Contexto del Trabajo Fin de Grado.

En la actualidad existe un gran potencial de desarrollo de servicios basados en localización (LBS) sobre dispositivos portables, como teléfonos móviles o tabletas en espacios interiores. Esto ha supuesto un gran impulso para la investigación en sistemas de posicionamiento y navegación personal. Sin embargo, en espacios interiores no se dispone de tecnología tan desarrollada como existe en espacios exteriores mediante los Sistemas de Navegación Global (GNSS), como el sistema de posicionamiento global (GPS), los cuales proporcionan datos de posición con precisiones del orden de pocos metros. Estos sistemas no son adecuados para aplicaciones interiores en las que se demanden precisiones del orden de centímetros, no existiendo una tecnología alternativa desarrollada al mismo nivel para resolver el posicionamiento [Lopes, 2014].

Hoy en día las personas pasan entre el 80%-90% de su tiempo en espacios interiores [Wang, 2016]. Al mismo tiempo, el aumento de las tecnologías de comunicación ha facilitado que la mayoría de las personas de los países desarrollados posean un teléfono inteligente o una tableta: el 71% de los españoles posee un teléfono inteligente, porcentaje que aumenta al 91% si sólo se considera a las personas entre 18 y 29 años, números que se pueden obtener similares para varios países europeos, y superiores en EEUU. Resultan, por tanto, comprensibles los esfuerzos en el desarrollo de tecnologías de localización precisa (centimétrica) de dispositivos portables. En este sentido, el empleo de estos u otros dispositivos portables expande las posibilidades de los LBS gracias a la oportunidad de implementar el sistema en plataformas de destino (típicamente Android o iOS). Ejemplos de aplicaciones pueden ser la navegación guiada en interiores de edificios (aeropuertos, hospitales, fábricas…), aplicaciones de realidad aumentada para turismo cultural, juegos, estudio de puntos de interés y patrones de movimiento con objeto de mejorar los servicios o insertar publicidad [Filonenko, 2013].

Para estimar la posición de dispositivos portables se utilizan principalmente métodos basados en radiofrecuencia (RF) como por ejemplo el *fringerprinting* [Ferris, 2006], y más recientemente, en señales acústicas [Filonenko, 2013; Lazik,

2012; Lopes, 2014]. Los sistemas basados en RF proporcionan precisiones del orden del metro en el mejor de los casos, mientras que en los sistemas acústicos se alcanzan niveles centimétricos e incluso sub-centimétricos de precisión [Prieto, 2009].

Este TFG, se enmarca dentro del proyecto LOCATE-US, "Sistema de localización basado en señales ultrasónicas codificadas para posicionamiento de dispositivos móviles en interiores" ref. CCG2016/EXP-078. Este proyecto propone un sistema de posicionamiento local de bajo coste, que permite la localización de dispositivos portables en entornos interiores extensos, haciendo uso de Sistemas de Posicionamiento Locales Ultrasónicos (U-LPSs), junto a sensores inerciales. Este sistema permite la localización de un número no limitado de usuarios de manera que de forma autónoma son capaces de obtener su posición.

La primera parte del proyecto, que, ya se encuentra desarrollada, propone la localización centimétrica de dispositivos portables en entornos interiores mediante el procesamiento de señales ultrasónicas (US) emitidas por un conjunto de sistemas U-LPSs ubicados en el entorno [Pérez, 2016]. En [Pérez, 2016] los U-LPS emiten a 41kHz, y dado que muchos teléfonos incluyen filtros paso bajo que no permiten trabajar por encima de los 22 kHz, se hace uso de una tarjeta de adaptación diseñada en un proyecto previo (US-PHONE, CCG2014/EXP-077) donde se captura la señal ultrasónica emitida, se digitaliza y se envía al dispositivo móvil vía USB para su procesamiento.

Se opta por la utilización de señales ultrasónicas frente a trabajos basados en radio frecuencia (RF) debido a que esta última tecnología consigue precisiones menores a las deseadas (orden de metros). Además, existen otros trabajos relacionados, como el sistema BeepBeep [Peng, 2007] que mide la distancia relativa entre dos dispositivos móviles a partir de la medida de señales acústicas audibles (con la consiguiente molestia para los usuarios). Otro sistema es el LOK8 [Filonenko, 2013] que aprovecha la capacidad del altavoz de los teléfonos móviles inteligentes para producir sonidos a frecuencias inaudibles de hasta 22kHz. Se trata de un sistema centralizado, en el que el teléfono emite y un conjunto de 4 micrófonos detectan la señal recibida y envían los datos de posición a un sistema central. Sin embargo, la

tendencia de este tipo de aplicaciones es el diseño de sistemas orientados a la privacidad de modo que sea el propio dispositivo el que controle su posición y no una unidad central, lo que podría causar conflictos en cuanto al uso de esa información. Un trabajo orientado a privacidad es el de [Aguilera, 2013], que también utiliza señales ultrasónicas, pero en la banda de 18-22kHz, lo que implica molestias debidas a artefactos audibles y una calidad de las señales adquiridas por el móvil muy pobre.

Dado que el rango de medida de los U-LPS es limitado, para que un usuario pueda moverse libremente por un espacio interior extenso se deberían instalar varios U-LPS para cubrir toda la zona. Como la localización de personas no exige precisiones centimétricas en todo el recorrido y con el objetivo principal de reducir el despliegue y el mantenimiento de la infraestructura de las balizas, se propone espaciar los U-LPS de modo que se encuentren en los sitios clave (accesos y salidas, por ejemplo), y utilizar la información de sensores inerciales en zonas donde no sea necesario un posicionamiento tan preciso. El error acumulativo de los sistemas inerciales se puede corregir al llegar al área de influencia de un U-LPS.

En esta idea va dirigida la segunda línea de trabajo del proyecto LOCATE-US, abordándose en este Trabajo Fin de Grado (TFG). Concretamente este TFG propone la implementación de algoritmos de fusión sensorial de la información proveniente de los U-LPS con la de los sensores inerciales, en una app que pueda estar incluida en cualquier dispositivo Android, para posicionamiento en espacios interiores extensos.

En la Figura 1 se puede observar el esquema de localización propuesto en el TFG.

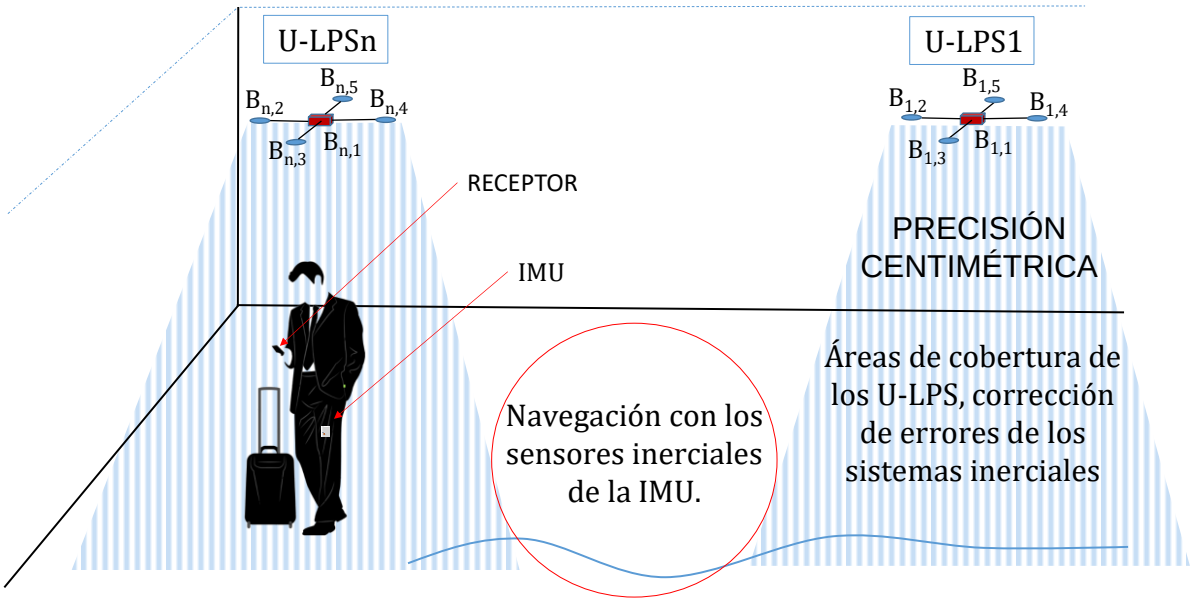


Figura 1: Esquema de localización a partir de U-LPSs y sensores inerciales.

El diagrama de bloques de la Figura 2 esquematiza el procesamiento integrado en la aplicación de presente TFG, en línea con el proyecto LOCATE-US. Se observa un primer bloque que se encarga de la adquisición y el procesamiento de las señales ultrasónicas. Este primer bloque consta de una primera parte hardware, que se compone por las balizas U-LPS junto una tarjeta de adquisición US diseñada en un proyecto previo (US-PHONE, CCG2014/EXP-077). La información adquirida por la tarjeta es enviada vía USB al dispositivo portable, en donde un software desarrollado en [Díaz, 2017], se encarga de realizar las correlaciones de los códigos recibidos, y obtener diferencias de distancias a partir de las diferencias de tiempo de vuelo (TDOA, *Time Difference Of Arrival*) de cada señal emitida por cada baliza del U-LPS. A lo largo de este trabajo cuando se habla de TDOA, se considera directamente las diferencias de distancias obtenidas a partir de las diferencias de tiempos de llegada, ya que las diferencias de distancias son directamente las diferencias de tiempos de llegada multiplicadas por un factor que relaciona la velocidad del sonido en el aire y la frecuencia de muestreo de las señales recibidas.

Por otro lado, en paralelo se realiza el procesamiento de las señales obtenidas a través de los sensores inerciales. En esta parte, se dispone de una IMU (*Inertial Measurement Unit*) como dispositivo hardware, la cual envía las medidas de los sensores inerciales a través del *Bluetooth* al dispositivo portable, donde una parte

software realiza el procesamiento gracias a un Filtro de Kalman Extendido (EKF), y un algoritmo de detección de pasos, de estimación de la anchura del paso y de posición relativa, desarrollado en un TFG previo [Cervigón, 2017].

Una vez realizado estos dos procesos en paralelo, la posición se obtendrá, en función de tres situaciones diferentes. En la situación en la que no se tenga cobertura U-LPS, se tomará como posición la relativa proporcionada por los sensores inerciales directamente. En el caso que sea la primera vez que se ha obtenido cobertura U-LPS, la posición se obtendrá mediante la aplicación del algoritmo Gauss-Newton. Si se dispone de cobertura, pero no es la primera vez que se ha obtenido, se aplicarán algoritmos de fusión sensorial a la información proporcionada por los US, es decir se considerarán las TDOA junto a la posición relativa obtenida a partir de las medidas de los sensores inerciales, aprovechando esta situación para corregir los errores acumulados por los sensores inerciales. En este último caso se centrará este TFG, además de la etapa de visualización en pantalla.

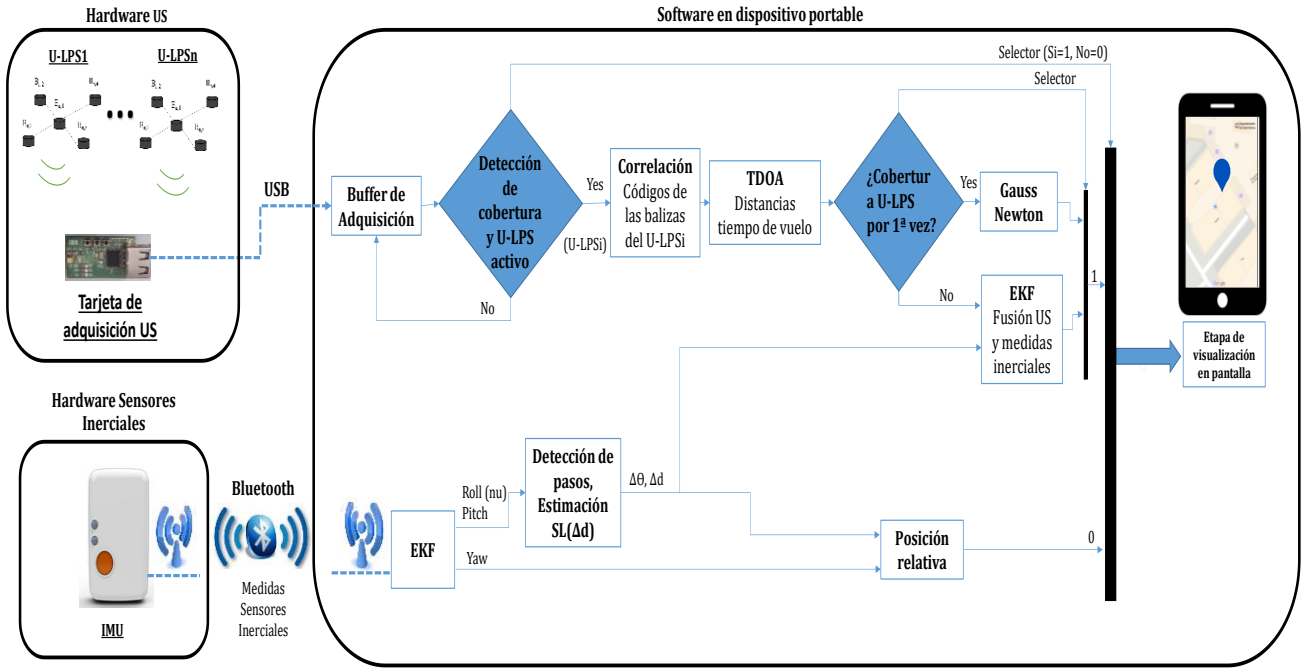


Figura 2: Diagrama de bloques del proyecto LOCATE-US.

La fusión de la información sensorial de la IMU junto a la de las US se realizará a través de un Filtro de Kalman Extendido (EKF). Concretamente se hará uso del EKF implementado en Android en [Cervigón, 2017], el cual se modificará para poder ajustar los algoritmos de fusión sensorial que se precisen, y para poder posicionar de forma dinámica.

También se hará uso de los algoritmos de Gauss-Newton desarrollados en [Díaz, 2017], los cuales se optimizarán para mejorar el posicionamiento y poder realizarlo de forma dinámica.

Para la etapa de visualización en pantalla se hará uso de los mapas proporcionados por la Google Maps Android API (interfaz de programación de aplicaciones), donde se podrá visualizar la trayectoria realizada por el usuario en tiempo real. En todo momento se pretenderá reducir la carga computacional asociada a los algoritmos para mejorar el procesamiento en tiempo real.

1.2 Objetivos del trabajo.

Este TFG tiene como objetivo principal la implementación de algoritmos de fusión sensorial para la navegación, junto a la integración de todo el sistema de localización propuesto por el proyecto LOCATE-US en una misma aplicación Android que permita estimar el posicionamiento de personas en tiempo real dentro de espacios interiores extensos.

Concretamente los objetivos a abordar en cada parte son:

1.- Integración en una misma aplicación Android de los algoritmos de procesamiento de la señal ultrasónica desarrollados en [Díaz, 2017] y los de posicionamiento relativo a partir de las medidas inerciales [Cervigón, 2017].

2.- Mejora del procesamiento ultrasónico mediante la aplicación de técnicas de optimización para los procesos de correlación y la aplicación y la modificación de algoritmos de forma dinámica. Se permitirá trabajar ante diferente número de balizas en caso de

pérdida de la señal emitida por alguna baliza del U-LPS, consiguiendo así un sistema más robusto.

3.- Implementación en Android de algoritmos de fusión sensorial de las señales ultrasónicas con las medidas inerciales, en función de la cobertura de U-LPS.

4.- Implementación Android de la etapa de visualización en pantalla en tiempo real a través del mapa proporcionado por la *Google Maps API*.

1.3 Estructura del documento.

Este documento consta de 11 capítulos, en los cuales se pretende abordar cada uno de los aspectos que se han desarrollado en este TFG de manera secuencial hasta llegar al objetivo final.

El capítulo 1 aquí presente, expone una breve introducción del marco en el que esta englobado el TFG. El capítulo 2 realiza una explicación de toda la base teórica necesaria para el desarrollo del TFG, en este capítulo se abordan conceptos relacionados con la codificación de las señales ultrasónicas, el algoritmo de posicionamiento Gauss-Newton, el Filtro de Kalman Extendido, y los algoritmos de fusión sensorial. En el capítulo 3 se describe el material utilizado para desarrollar el proyecto, así como todas sus características. El proceso de desarrollo de la aplicación en Android paso a paso se desarrolla a lo largo del capítulo 4. Los capítulos 5 y 6 recogen tanto los resultados como las conclusiones obtenidas además de las posibles líneas de trabajo futuras que se podrían abordar.

Posteriormente, se encuentran una serie de capítulos que se encargan de recoger los costes de la aplicación (capítulo 7), las condiciones de uso de la aplicación (capítulo 8), además de un manual de la app orientado al usuario (capítulo9). El capítulo 10 recoge toda la bibliografía usada a lo largo del desarrollo de este TFG. Y por último en el capítulo 11 se encuentran parte de los códigos realizados por este TFG.

CAPÍTULO 2: BASE TEÓRICA:

2.1 Estructura del sistema

Observando la Figura 2, se puede dividir el sistema en cinco apartados principales, la codificación y el procesamiento de las señales ultrasónicas; el algoritmo de posicionamiento de Gauss-Newton; el filtro de Kalman Extendido (del cual se hace uso en el posicionamiento inercial y la fusión sensorial); el sistema de posición inercial; y los algoritmos de fusión sensorial. En este capítulo se abordarán todos los fundamentos teóricos que sustentan cada uno de estos cinco apartados.

2.2 Codificación y procesamiento de las señales ultrasónicas

El Sistema de Posicionamiento Local Ultrasónico (U-LPS) aquí presente está basado en la obtención de las diferencias de distancias a partir de las diferencias de tiempo de vuelo (TDOA, Time Difference Of Arrival) de las señales emitidas por cada uno de los cinco emisores que lo componen. Esta técnica se basa en medir las diferencias de tiempo existentes entre las señales recibidas tomando una de ellas como referencia, ya que no existe sincronismo entre los emisores y el receptor.

Debido a que la precisión de las TDOA es directamente proporcional a la calidad con las que se reciben las señales una vez propagadas por el aire, es de suma importancia la utilización de unas señales con un esquema de codificación con características particulares.

LOCATE-US opta por el uso de esquemas de codificación mediante secuencias Kasami debido a sus buenas propiedades de autocorrelación y correlación cruzada. Concretamente se ha hecho uso de códigos Kasami [Kasami, 1969] con una longitud de 255 bits. Las señales codificadas han sido moduladas en BPSK (*Binary Phase-Shift Keying*) para ajustarlas al ancho de banda de los transductores empleados con dos períodos de una portadora sinusoidal a 41.67 kHz (fe), que se ha muestreado a 500 kHz (fs). Por lo tanto, se consideran 24 muestras (12 en cada período de portadora) para cada bit del código Kasami.

Para comprender un poco mejor el esquema de codificación empleado por este trabajo, en la Figura 3 se observa un ejemplo de la modulación de un código Kasami [1, -1, 1] con dos ciclos de portadora por periodo a una frecuencia f=41,67 kHz.

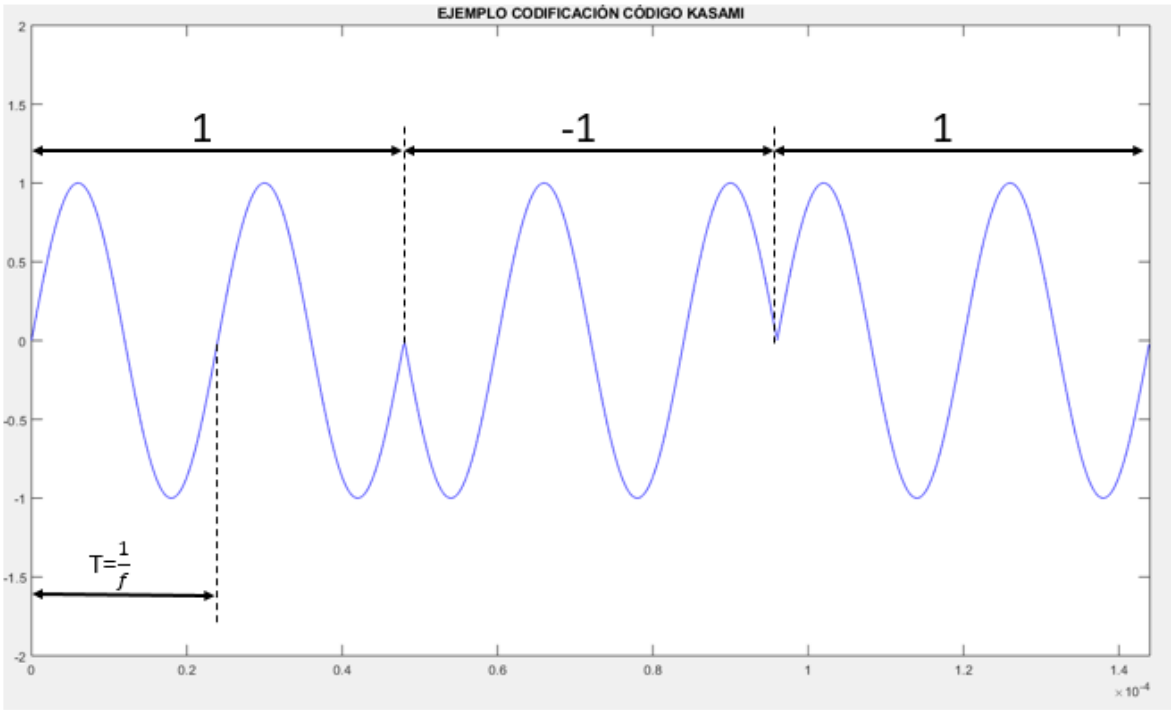


Figura 3: Código Kasami [1, -1, 1] modulado en *BPSK* con 2 ciclos de portadora a 41.67 KHz.

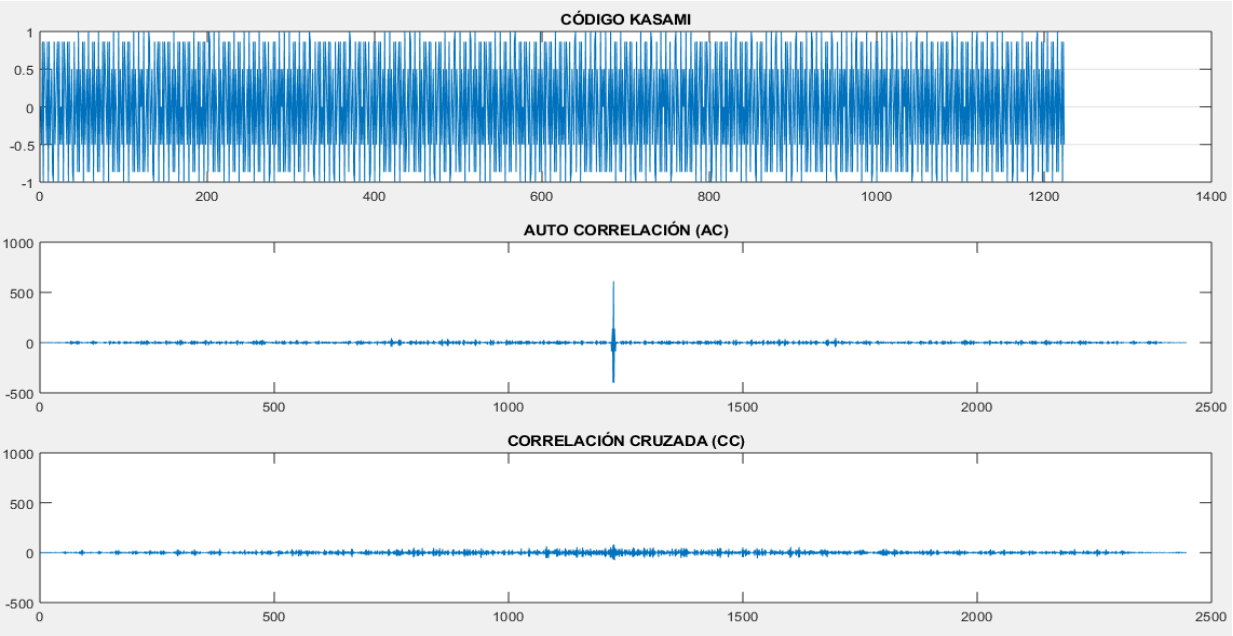


Figura 4: Correlación y autocorrelación del código Kasami modulado en *BPSK*Figura 3.

Además, en la Figura 4 se puede observar la buena respuesta de los códigos Kasami ante la auto correlación (AC) y la correlación cruzada (CC). Esta es una de las características por las que se han elegido estos códigos, además de haber observado un gran comportamiento de estos de manera experimental.

Debido a restricciones del hardware empleado en la recepción (sólo se dispone de un convertidor digital analógico (DAC) en el microcontrolador), se ha implementado una técnica de acceso múltiple por división de tiempo (TDMA) lo que contribuye a reducir el efecto cerca-lejos y las interferencias por acceso múltiple (MAI). En cualquier caso, el sistema es fácilmente adaptable a otros esquemas de emisión como CDMA (*Code Division Múltiple Access*).

Concretamente se ha configurado según el esquema de codificación de la Figura 5, donde se transmiten los cinco códigos emitidos por cada uno de los transductores de manera consecutiva sin dejar espacio entre ellos, exceptuando entre el código n+4 y el código n, momento en el que se vuelve a repetir la secuencia y se deja un tiempo de guarda. La duración de cada código asociado a cada baliza dura $\frac{1}{fe} * 255 * 2 = 12.24\ ms$, y el tiempo de guarda que se ha configurado tiene una duración de 3.8 ms.

Teniendo en cuenta este modelo de emisión, y que la frecuencia de recepción usada es de 100 kHz, el tamaño de cada código en la recepción es $12.24ms * 100\ kHz = 1224\ muestras$**,** y el tamaño del tiempo de guarda es de 380 muestras. Como no se hace uso de una señal de sincronismo en la recepción, se debe garantizar que se almacenan los cinco códigos emitidos completos. Para ello se ha configurado un buffer de recepción sobredimensionado de 8192 muestras, para garantizar que en la recepción se guardan 6 códigos completos y con ello los 5 deseados. En el peor de los casos el tamaño del buffer necesario sería (1224*6) + 380 = 7724 muestras. Debido a que la identificación de cada código en la recepción se realiza a través de una correlación cruzada mediante una FFT en la plataforma Android, por especificaciones del sistema se exige que el tamaño del buffer de correlación sea potencia de dos. En ese sentido, el número más cercano a 7724 potencia de dos es 8192.

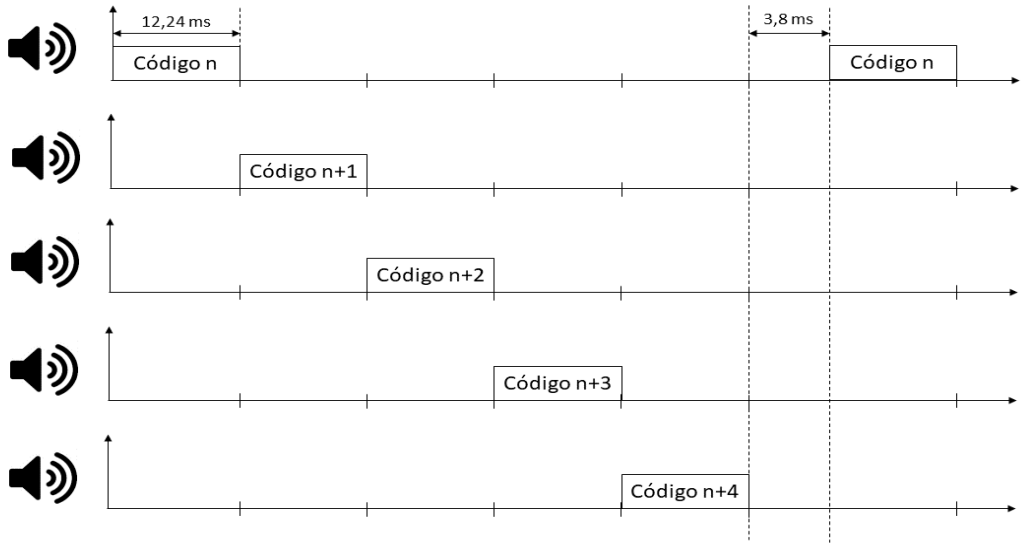


Figura 5: Esquema de emisión de los códigos.

Como se ha comentado anteriormente, la identificación de los códigos emitidos se realiza a través de una correlación cruzada de la señal adquirida con los patrones de emisión. El pico de correlación indica el instante de tiempo de llegada de cada señal asociada a cada emisor puesto que cada emisor está codificado con un código diferente.

Una vez que se tienen los instantes de tiempo de llegada de cada señal ya se pueden calcular los TDOA, de manera que uno de esos instantes de llegada se considera de referencia, y el resto se diferencian respecto al de referencia.

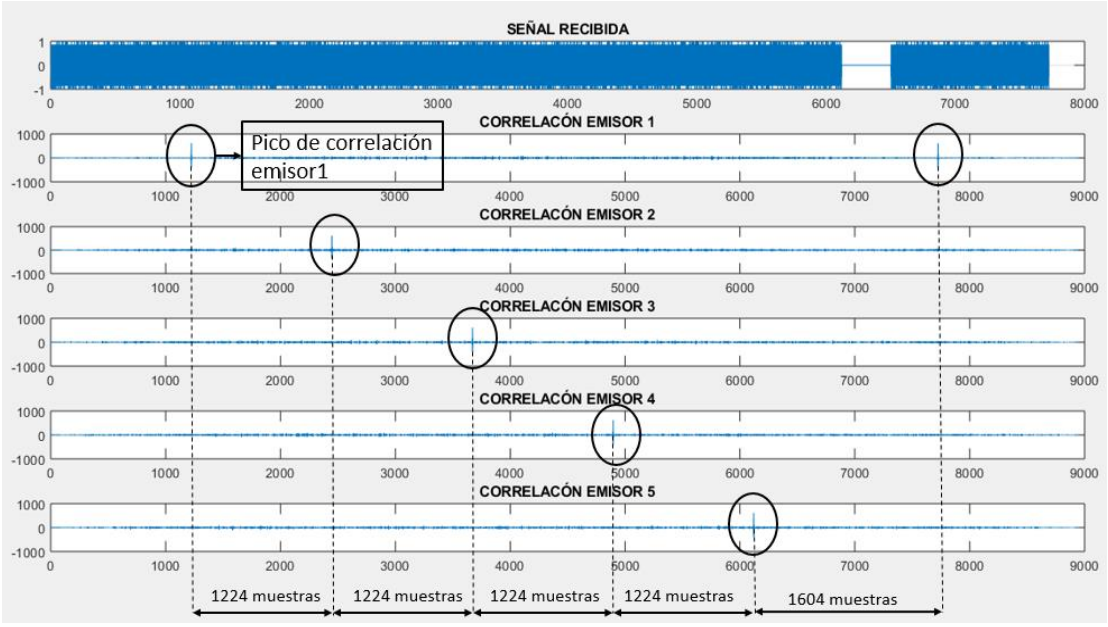


Figura 6: Correlación señal recibida.

De esta manera se puede observar en la Figura 6 como quedaría la señal obtenida en el buffer de recepción (SEÑAL RECIBIDA), y las diferentes correlaciones con cada uno de los códigos emitidos por cada emisor, si el receptor estuviese colocado en un punto equidistante a todos los transductores (diferencias de distancias igual a cero). Se observa como la distancia entre los picos de correlación es 1224 muestras (tamaño de cada código) exceptuando la distancia entre el emisor 5 y el emisor 1 que la distancia es 1604 muestras (1224 + 308).

2.3 Algoritmo de posicionamiento GAUSS NEWTON

Una vez que se tienen las diferencias de distancias (TDOA), para la determinación de la posición relativa se ha aplicado el método matemático de trilateración hiperbólica. Este método, a partir del conocimiento de tres hipérbolas en posición 2D (Figura 7) estima la posición del receptor. De esta manera el sistema necesita cuatro balizas, ya que cada una de las hipérbolas se generan a partir de la diferencia de distancias entre una baliza y la de referencia, con lo que para tener tres hipérbolas se necesitan 3 balizas más la baliza de referencia, es decir 4 balizas. El sistema hace uso de cinco balizas con el fin de que la quinta baliza actúe como sistema redundante para asegurar el posicionamiento ante posibles fallos de una de las otras balizas.

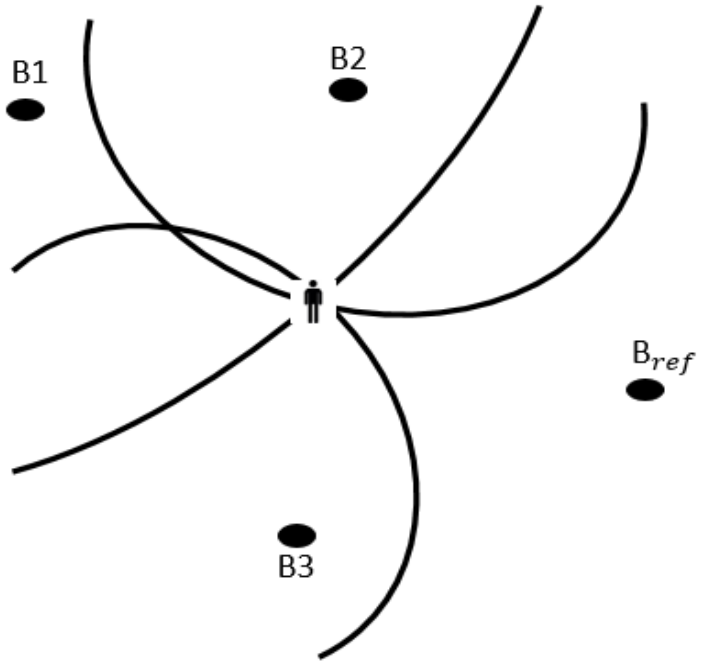


Figura 7: Ejemplo de intersección hiperbólica.

Para la resolución del sistema de ecuaciones no lineales que propone la trilateración hiperbólica se particulariza el algoritmo de minimización Gauss-Newton [Díaz, 2017]. Debido a que Gauss-Newton es un algoritmo iterativo, se ha de inicializar para que converja a una solución adecuada.

La particularización del algoritmo de Gauss-Newton para el caso hiperbólico se observa en la Figura 8. Primero como se ha comentado anteriormente, se introduce una inicialización de los valores que se quiere estimar. Después se realiza un cálculo del valor absoluto de la tolerancia, que posteriormente se comprueba si es mayor que un valor previamente definido por el usuario. Si es mayor, se recalculan las posiciones que se desean estimar a través del valor de esta tolerancia calculada. Si el valor absoluto de la tolerancia es suficientemente pequeño, es decir menor que el valor establecido, se considera que la estimación de los valores tiene suficiente precisión para validarlos, con lo que se devuelven como salida.

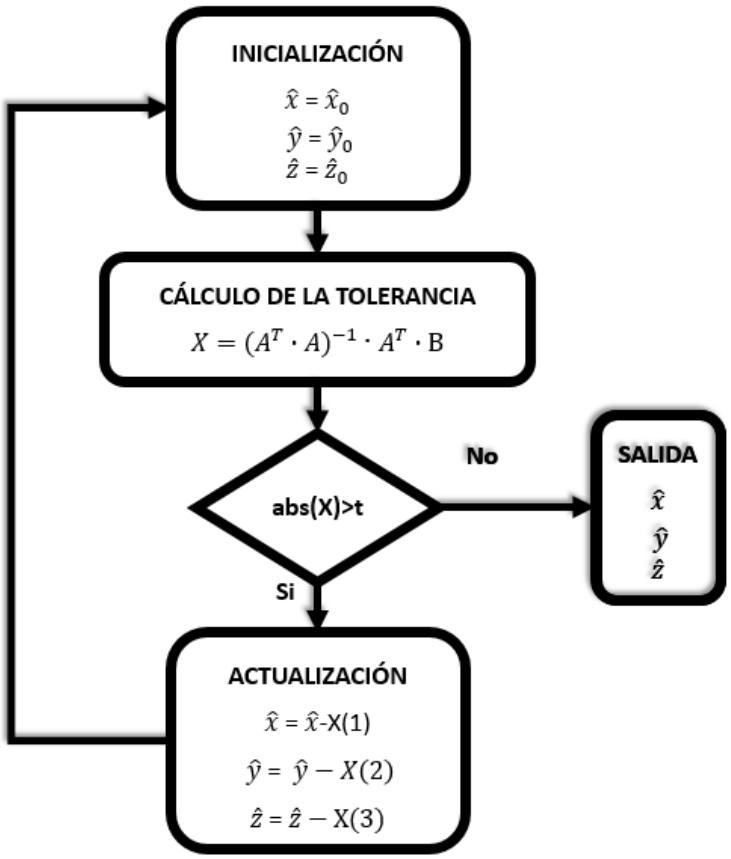


Figura 8: Algoritmo de Gauss-Newton.

El cálculo de la tolerancia X (2.3.1) depende de la matriz **B**, la cual está compuesta por la diferencia entre las diferencias de distancia entre el emisor y el receptor,

estimada y la medida; y la matriz **A**, la cual se define como la derivada de la diferencia de distancias estimadas con respecto a cada una de las variables de la posición.

$$\boldsymbol{X} = (\boldsymbol{A^T} \cdot \boldsymbol{A})^{-1} \cdot \boldsymbol{A^T} \cdot \boldsymbol{B} \tag{2.3.1}$$

La estimación de la posición también se podría obtener a través del método matemático de trilateración esférica y con ello particularizar el algoritmo de Gauus-Newton para este caso, pero para ello se debería tener acceso a las distancias absolutas entre la recepción y la emisión. Para la obtención de estas medidas sería necesario conocer el instante de emisión de las balizas, por lo que debería existir algún método de sincronismo entre el receptor y las balizas. Esto aumentaría la complejidad de la infraestructura, con lo que este trabajo ha optado por trabajar con diferencia de distancias y con ello trilateración hiperbólica.

2.4 Descripción Filtro de Kalman y Filtro de Kalman Extendido

El Filtro de Kalman (KF) es uno de los descubrimientos más grandes en la historia de la teoría de la estimación y fue descrito por Rudolf Emil Kalman en [Hervás, 2014] y [Kalman, 1961]. Rudolf Emil Kalman propuso un estimador estocástico dinámico recursivo como solución al problema del filtrado lineal de datos discretos.

Tanto la predicción de señales aleatorias como la separación de éstas de un ruido aleatorio son un gran problema en la teoría de estimación. Estos problemas se deben a las perturbaciones aleatorias que aparecen en un sistema y están presentes en la mayoría de los sistemas reales. Además, a causa de estas perturbaciones las salidas de los sistemas difieren del comportamiento ideal. Debido a esto, resulta de suma importancia obtener medidas satisfactorias a la salida de estos sistemas incluso cuando estas difieren del comportamiento ideal.

El filtro de Kalman es un proceso matemático iterativo que como se observa en la Figura 9, describe de forma recursiva la predicción y la corrección del estado actual. Más concretamente predice la estimación del estado actual en el tiempo a partir del estado anterior y las ecuaciones dinámicas, y posteriormente ajusta o corrige la

predicción mediante la observación del estado actual. Estos pasos los realiza cíclicamente.

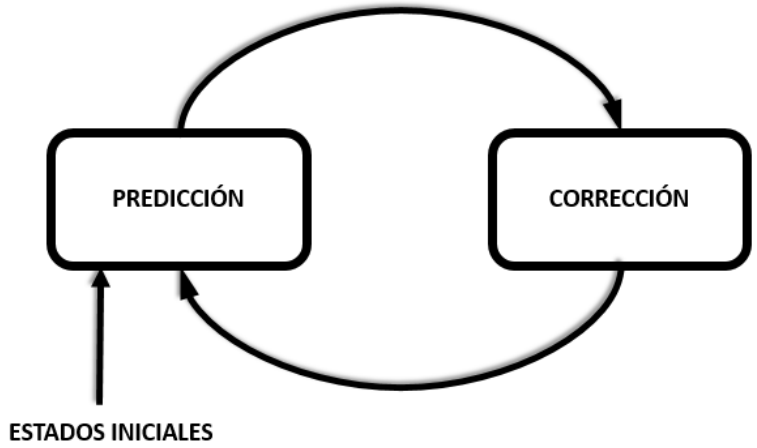


Figura 9: Funcionamiento del filtro de Kalman.

Una de las mayores virtudes del KF es que ofrece un correcto funcionamiento en sistemas dinámicos lineales con presencia de ruido blanco o aleatorio. Permite estimar el estado de la variable desconocida $\boldsymbol{x_k}$, mediante las señales conocidas $\boldsymbol{u_k}$(entrada) y $\boldsymbol{z_k}$(observable), las cuales se encuentran linealmente relacionadas con dicho estado, pero están corrompidas por un ruido blanco. Dicho ruido genera una incertidumbre en las medidas, pero el filtro de Kalman consigue la estimación óptima y fiable minimizando la matriz de covarianza de estimación del error. Se debe resaltar que el filtro de Kalman es capaz de estimar tanto el estado presente como pasados y futuros.

Para realizar un filtrado se debe tener conocimiento de dos ecuaciones. La ecuación de la observación, la cual describe cómo han sido tomadas las medidas por los sensores, y la ecuación del modelado, que describe como se espera que el sistema evolucione con el tiempo. El filtro de Kalman asume que la ecuación del modelado es lineal, pero en la práctica existen muchos sistemas que no son lineales como el caso de la navegación, y con ello la ecuación con que se modelan tampoco. Para solventar esta carencia en [Cox, 1964] se introdujo una modificación del filtro de Kalman para problemas no lineales. A esta modificación del filtro se le denomina Filtro de Kalman Extendido (EKF) y se basa en la linealización de un sistema no lineal respecto a la estimación actual aplicando la aproximación de Taylor. Debido a

que el EKF linealiza el modelo del sistema en cada estimación del estado, el modelo del sistema cambia constantemente en cada estimación.

Salvando la linealización del modelo del sistema el EKF realiza los mismos pasos recursivos que el filtro de Kalman lineal, la predicción y la corrección. Aunque se debe tener en cuenta que, debido a esta linealización, la estimación no es tan buena en el EKF como en el KF, pero dependiendo de la aplicación, en muchas ocasiones esta estimación presenta suficiente precisión como en el caso de la navegación.

A lo largo de este trabajo se hará uso del Filtro de Kalman Extendido, con lo que se procede a realizar una presentación de las ecuaciones en las que se basa.

Para empezar, se debe tener en cuenta que en este trabajo se hace uso del EKF porque el sistema al que se aplica es no lineal. Un sistema no lineal se puede representar de la siguiente forma:

$$\boldsymbol{X_k} = \boldsymbol{f}\ (\boldsymbol{X_{k-1}}) + \boldsymbol{w_k} \tag{2.4.1}$$

$$\boldsymbol{Z_k} = \boldsymbol{h}\ (\boldsymbol{X_k}) + \boldsymbol{v_k} \tag{2.4.2}$$

Donde en (2.4.1) $\boldsymbol{X_k}$es el vector de estados; $\boldsymbol{f}\ (\boldsymbol{X_{k-1}})$ es una función que representa la relación entre el estado anterior y el actual; y $\boldsymbol{w_k}$representa las perturbaciones del proceso. En (2.4.2) $\boldsymbol{Z_k}$ es un vector que representa la información de la medición (observación); $\boldsymbol{h}\ (\boldsymbol{X_k})$ es una función que representa la relación entre el vector de estados y las mediciones; y $\boldsymbol{v_k}$representa las perturbaciones de la medida. Tanto $\boldsymbol{f}\ (\boldsymbol{X_{k-1}})$ como $\boldsymbol{h}\ (\boldsymbol{X_k})$ son funciones a particularizar para cada uso concreto del EKF. Y tanto $\boldsymbol{w_k}$ como $\boldsymbol{v_k}$son ruidos blancos que siguen una distribución gaussiana con media cero, incorrelados y con matriz de covarianza finita, **Q** y **R** respectivamente.

Una vez que se conoce el sistema no lineal con el que se trabaja, se procede a describir las etapas que realiza el EKF:

Etapa de Predicción:

La etapa de predicción es el primer paso a realizar por el EKF, en esta etapa se calcula la estimación a priori del estado y la matriz de covarianza de erro a priori.

$$\widehat{\mathbf{X}}_{\boldsymbol{k}}^{\ -} = \boldsymbol{f}(\widehat{\mathbf{X}}_{\boldsymbol{k-1}}) \tag{2.4.3}$$

$$\mathbf{P}_{\boldsymbol{k}}^{\ -} = \mathbf{A}_{\boldsymbol{k}} \cdot \mathbf{P}_{\boldsymbol{k-1}} \cdot \mathbf{A}^{\boldsymbol{T}}{}_{\boldsymbol{k}} + \mathbf{Q}_{\boldsymbol{k}} \tag{2.4.4}$$

Donde en (2.4.3), $\widehat{\mathbf{X}}_{\boldsymbol{k}}^{\ -}$ es la estimación a priori del vector de estados, basada en la estimación del estado anterior. En (2.4.4) $\mathbf{P}_{\boldsymbol{k}}^{\ -}$ es la matriz de covarianza del error a priori, la cual depende de la matriz de covarianza del error del estado anterior $\mathbf{P}_{\boldsymbol{k-1}}$; la varianza del ruido del proceso **Q**; y la matriz **A**, que es la derivada con respecto a cada componente del vector de estados que relaciona el estado previo con el estado actual, es decir es el jacobiano de la ecuación de transición del sistema, y tiene como finalidad la linealización del sistema.

$$\mathbf{A} = \frac{\boldsymbol{d}\widehat{\mathbf{X}}_{\boldsymbol{k}}^{\ -}}{\boldsymbol{d}\widehat{\mathbf{X}}_{\boldsymbol{k}}} \tag{2.4.5}$$

<u>Etapa de Corrección:</u>

Una vez realizada la etapa de predicción, se procede a la etapa de corrección, la cual tiene como finalidad corregir la estimación prevista en la etapa anterior mediante la observación del estado actual. Las ecuaciones que se abordan en esta etapa son:

$$\mathbf{S} = \mathbf{H} \cdot \mathbf{P}_{\boldsymbol{k}}^{\ -} \cdot \mathbf{H}^{\boldsymbol{T}} + \mathbf{R} \tag{2.4.6}$$

$$\mathbf{K} = \mathbf{P}_{\boldsymbol{k}}^{\ -} \cdot \mathbf{H}^{\boldsymbol{T}} \cdot \mathbf{S}^{\boldsymbol{-1}} \tag{2.4.7}$$

$$\mathbf{V} = \boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{h}(\widehat{\mathbf{X}}_{\boldsymbol{k}}^{-}) \tag{2.4.8}$$

$$\widehat{\mathbf{X}}_{\boldsymbol{k}} = \widehat{\mathbf{X}}_{\boldsymbol{k}}^{\ -} + \mathbf{K} \cdot \mathbf{V} \tag{2.4.9}$$

$$\mathbf{P}_{\boldsymbol{k}} = \mathbf{P}_{\boldsymbol{k}}^{\ -} - \mathbf{K} \cdot \mathbf{S} \cdot \mathbf{K}^{\boldsymbol{T}} \tag{2.4.10}$$

Donde en (2.4.6) **S** se define como la matriz intermedia y es dependiente de **H** (2.4.11)que es el jacobiano de la ecuación de la observación (2.4.9); de $\boldsymbol{P_k}^-$ que es la matriz de covarianza del error a priori; y de R que representa la matriz de covarianza del ruido de la medida.

$$\boldsymbol{H = \frac{dh(X_k^-)}{dX_k^-}} \tag{2.4.11}$$

K en (2.4.7)representa la ganancia de Kalman, y depende de $\boldsymbol{P_k}^-$; **H**; y la matriz intermedia **S**. En (2.4.8) **V** representa la diferencia entre las observaciones y la estimación de éstas. $\widehat{\mathbf{X}}_{\boldsymbol{k}}$ en (2.4.9) es la estimación del estado a posteriori, y depende de la estimación del estado a priori $\widehat{\mathbf{X}}_{\boldsymbol{k}}^-$; de la ganancia de Kalman **K**, y del vector **V**. Por último (2.4.10) define la matriz de covarianza del error a posterior $\boldsymbol{P_k}$ y depende de la matriz de covarianza del error a priori $\boldsymbol{P_k}^-$; de la ganancia de Kalman **K**; y de la matriz intermedia **S**.

Una vez finalizada la etapa de corrección, esta devuelve la estimación del estado a posteriori $\widehat{\mathbf{X}}_{\boldsymbol{k}}$, junto a la estimación de la matriz de covarianza del error $\boldsymbol{P_k}$. Estos estados devueltos por la etapa de corrección se introducen en la etapa de predicción, y se vuelve a realizar el ciclo sucesivamente.

Particularizando la Figura 9 para el EKF, y añadiendo las ecuaciones que acarrea, en la Figura 10 se puede observar el funcionamiento general del EKF.

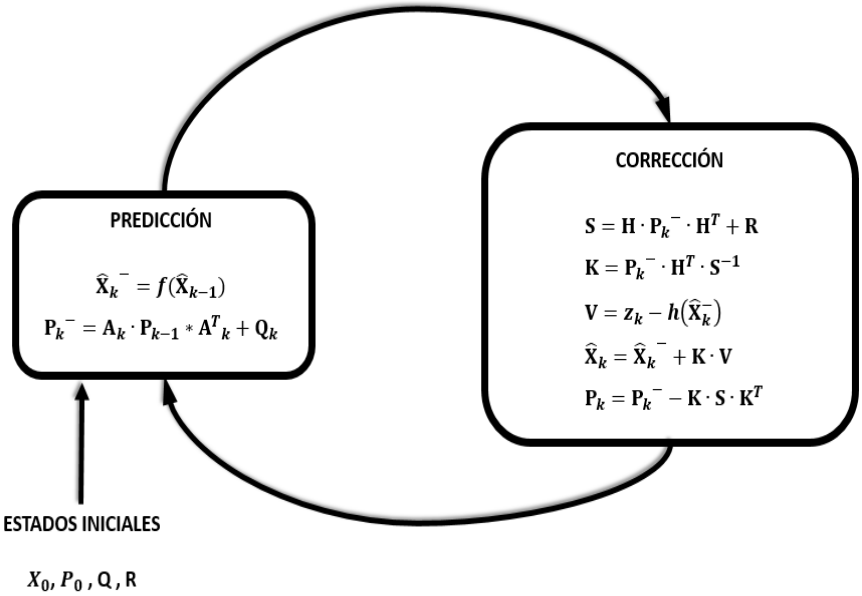


Figura 10: Funcionamiento del filtro de Kalman Extendido.

2.5 Sistema de posicionamiento inercial:

Este trabajo para resolver esta parte del sistema ha hecho uso del algoritmo de posicionamiento relativo (PDR) descrito en [Cervigón, 2017]. En zonas que no requieran gran precisión, se pueden usar únicamente las medidas de los sensores inerciales.

En [Cervigón, 2017] la IMU envía las medidas de los sensores inerciales de ésta al dispositivo portable vía Bluetooth. Estas medidas que están compuestas por la información de los acelerómetros (α en unidades de m/s2) y los giróscopos (ω en unidades de º/s) de los tres ejes, son fusionadas a través de un Filtro de Kalman Extendido (EKF) (Figura 10), el cual devuelve los ángulos Euler de orientación del sensor, *Roll* (ϕ), *Pitch* (⊙) y *Yaw* (Ψ). En la Figura 11 se puede observar el funcionamiento de esta primera etapa.

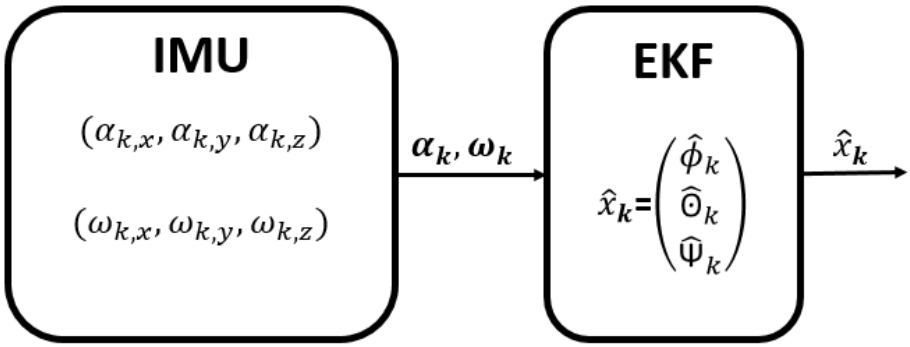


Figura 11: Diagrama de bloques de la estimación del *Roll, Pitch* y *Yaw.*

Donde k el número de iteración del ciclo y $\boldsymbol{x}$ el vector de estados que devuelve el EKF, que será un vector columna de dimensiones 3x1 compuesto por los ángulos de Euler.

Una vez que se tiene los ángulos de Euler, estos son introducidos en un algoritmo de posicionamiento. Este recibe la señal *Pitch* y *Yaw* obtenidas a partir del EKF y devuelve la posición relativa del individuo. Este algoritmo de posicionamiento está compuesto de un detector de pasos, un estimador de la anchura del paso (SL) y un algoritmo de posición relativa como se expone en la Figura 12.

Donde x e y son las coordenadas sobre el plano, y θ la orientación.

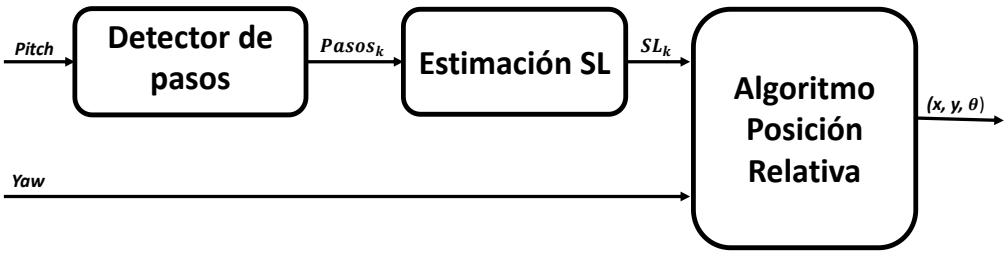


Figura 12: Diagrama de bloques del algoritmo de posicionamiento.

El primer bloque que se encuentra en el algoritmo de posicionamiento es un detector de pasos. Este bloque a partir del valor del ángulo Pitch ($\boldsymbol{\Theta_k}$) recibido, estima la detección de un paso ($\boldsymbol{Pasos_k}$) basándose en la detección de máximos y mínimos absolutos de la señal Pitch.

Una vez que se detecta el $\boldsymbol{Paso_k}$, se estima la anchura de paso $\boldsymbol{SL_k}$ en el siguiente bloque de la Figura 12. Para ello se hace uso del paso detectado junto con su máximo y su mínimo absoluto asociado al mismo ($\boldsymbol{\Delta\Theta_k}$), y gracias a las ecuaciones (2.5.1) y (2.5.2) se puede obtener la anchura de paso k-ésimo ($\boldsymbol{SL_k}$).

$$\boldsymbol{\Delta\Theta_k = \Theta_{máx} - \Theta_{mín}} \tag{2.5.1}$$

$$\boldsymbol{SL_k = a_h * \Delta\Theta_k + b_h} \tag{2.5.2}$$

Donde a_h y b_h pueden ser 0.0294 y 0.232 respectivamente si se calculan a través de la regresión lineal universal según [Cervigón, 2017], o pueden ser calculados de manera experimental como se ha realizado en este trabajo obteniendo 0.294 y 0.513 respectivamente. En esta parte también se obtiene la amplitud del ángulo (ΔΘ) que junto a la amplitud del paso son usados para el cálculo de la posición relativa además de ser necesarios para realizar la fusión sensorial.

Como último bloque del algoritmo de posicionamiento aparece la estimación de la posición relativa, el cual estima la posición relativa sobre el plano del usuario ($\boldsymbol{\hat{Y}_k}$) que se compone de la estimación del eje x ($\boldsymbol{\hat{x}_k}$), del eje y ($\hat{y}_k$) y de la orientación ($\boldsymbol{\hat{\theta}_k}$) gracias a la ecuación (2.5.3):

$$\widehat{\mathbf{Y}}_{\boldsymbol{k}} = \begin{bmatrix} \widehat{\boldsymbol{x}}_{\boldsymbol{k}} \\ \hat{y}_{\boldsymbol{k}} \\ \widehat{\boldsymbol{\theta}}_{\boldsymbol{k}} \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \boldsymbol{SL_k} \cdot \boldsymbol{cos}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\Psi}) \\ \widehat{\boldsymbol{y}}_{\boldsymbol{k-1}} + \boldsymbol{SL_k} \cdot \boldsymbol{sin}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\Psi}) \\ \widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\Psi} \end{bmatrix} \quad (2.5.3)$$

2.6 Fusión Sensorial:

Como se ha comentado anteriormente, el EKF ofrece un correcto funcionamiento ante la presencia de ruido blanco, y por ello es muy usado en los sistemas de la actualidad. Al igual que en [Cervigón, 2017] hace uso de un EKF para la estimación de la posición relativa a través de la información de los sensores inerciales, este trabajo también hace uso de un Filtro de Kalman Extendido para fusionar la información proporcionada por las US y la IMU con el objetivo de obtener una posición más fiable y aplicar algoritmos de corrección para los errores acumulados que generan los sensores inerciales.

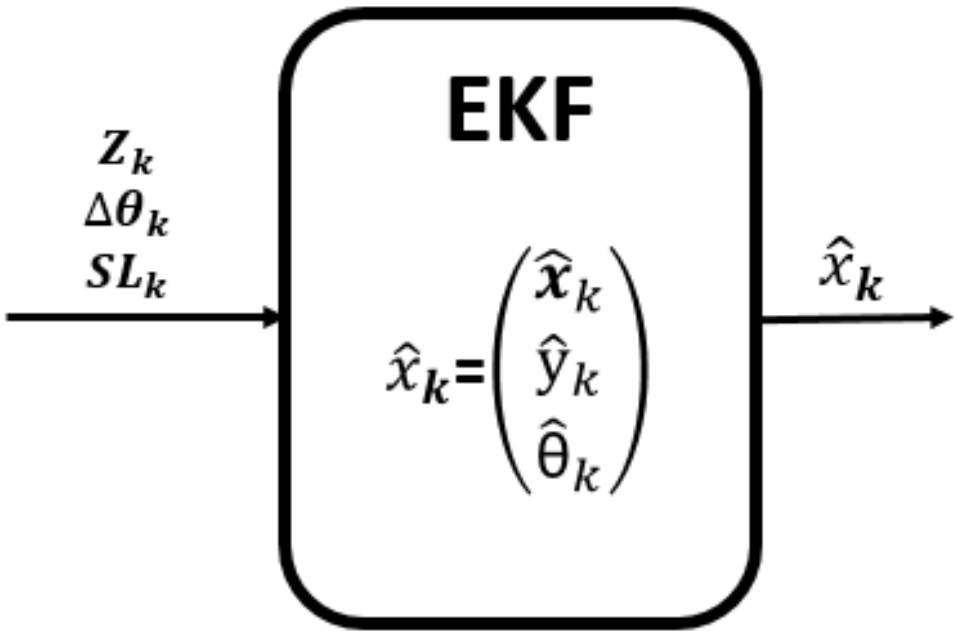


Figura 13: Filtro de Kalman Extendido aplicado a la Fusión Sensorial.

Concretamente, como se observa en la Figura 13, la información proveniente de las US que se le introduce al EKF incluye las diferencias de distancia calculadas a partir de las diferencias de tiempo de vuelo (TDOA) en un vector $\boldsymbol{Z_k}$ y como información

proveniente de la IMU se introduce la diferencia del ángulo $\Delta\boldsymbol{\theta_k}$ junto a la anchura de paso obtenida $\boldsymbol{SL_k}$.

Una vez realizada una breve introducción del EKF aplicado a la fusión sensorial, se procede a realizar una explicación de la particularización de las ecuaciones del EKF realizada por este trabajo, el cual se ha basado en [Gualda, 2014].

La actualización del vector de estado a priori se realiza a través de la información proporcionada por la IMU, $\Delta\boldsymbol{\theta_k}$ y $\boldsymbol{SL_k}$, gracias a la ecuación (2.6.1).

$$\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-} = \begin{bmatrix} \widehat{\boldsymbol{x}}_{\boldsymbol{k}} \\ \hat{y}_k \\ \widehat{\boldsymbol{\theta}}_{\boldsymbol{k}} \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \boldsymbol{SL_k} \cdot \boldsymbol{cos}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\theta_k}) \\ \widehat{\boldsymbol{y}}_{\boldsymbol{k-1}} + \boldsymbol{SL_k} \cdot \boldsymbol{sin}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\theta_k}) \\ \widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\theta_k} \end{bmatrix} \tag{2.6.1}$$

La matriz global de transición **A** al ser la derivada con respecto a cada componente del vector de estados que relaciona el estado previo con el estado actual queda según la ecuación (2.6.2).

$$\boldsymbol{A_k} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} & -\boldsymbol{SL_k} \cdot \boldsymbol{sin}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\theta_k}) \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{SL_k} \cdot \boldsymbol{cos}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}} + \Delta\boldsymbol{\theta_k}) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{2.6.2}$$

Otra de las matrices a particularizar del EKF, es $\boldsymbol{h}$ $(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$, una matriz que estima las TDOAs a priori como se observa en la ecuación (2.6.3).

$$\mathbf{h}\,(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}) = \begin{bmatrix} \boldsymbol{h_1}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}) \\ \vdots \\ \boldsymbol{h_M}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B}_2) - \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B_r}) \\ \vdots \\ \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B}_m) - \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B_r}) \\ \vdots \\ \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B}_M) - \boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B_r}) \end{bmatrix} \tag{2.6.3}$$

Siendo $\boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B_i})$ la distancia euclídea entre la posición estimada a priori, y la posición de una de las balizas del sistema ULPS ($\boldsymbol{B_i} = (\boldsymbol{x_B}, \boldsymbol{y_B}, \boldsymbol{z_B})$). $\boldsymbol{d}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-}, \boldsymbol{B_r})$ representa la distancia euclídea entre la posición a priori estimada y la posición de una de las balizas de nuestro sistema ULPS la cual se ha tomado como referencia en el cálculo, ya que se recuerda que se está trabajando en el caso hiperbólico, es decir se usa diferencia de distancias.

La siguiente matriz a actualizar es $\mathbf{H}_k$, que se define como la derivada de $\boldsymbol{h}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ con respecto a la estimación del vector de estados a priori (2.6.4).

$$\mathbf{H}\,(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})=\begin{bmatrix}\frac{\partial \boldsymbol{h}_1(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})}{\partial(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})}\\ \vdots \\ \frac{\partial \boldsymbol{h}_M(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})}{\partial(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})}\end{bmatrix} \tag{2.6.4}$$

Las matrices de covarianza, del error a priori y el ruido del proceso, $\mathbf{P}_k$ y $\mathbf{Q}$ respectivamente, se actualizan según (2.6.5) y (2.6.6).

$$\mathbf{P}_k=\begin{bmatrix}\boldsymbol{\sigma}_x^2 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\sigma}_y^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_z^2\end{bmatrix} \tag{2.6.5}$$

$$\mathbf{P}_k=\boldsymbol{Q} \tag{2.6.6}$$

Siendo $\boldsymbol{\sigma_x}=\boldsymbol{\sigma_y}=\boldsymbol{\sigma_z}=\mathbf{0.002}$.

Otra de las matrices a actualizar es la matriz de covarianza del ruido de la medida $\mathbf{R}_k$ (2.6.7).

$$\boldsymbol{R_k}=\begin{bmatrix}\boldsymbol{\sigma}_w^2 & \mathbf{0.5}*\boldsymbol{\sigma}_w^2 & \mathbf{0.5}*\boldsymbol{\sigma}_w^2\\ \mathbf{0.5}*\boldsymbol{\sigma}_w^2 & \boldsymbol{\sigma}_w^2 & \mathbf{0.5}*\boldsymbol{\sigma}_w^2\\ \mathbf{0.5}*\boldsymbol{\sigma}_w^2 & \mathbf{0.5}*\boldsymbol{\sigma}_w^2 & \boldsymbol{\sigma}_w^2\end{bmatrix} \tag{2.6.7}$$

Donde $\boldsymbol{\sigma_w}=0.005$.

Por último, se actualizará el vector de observaciones $\mathbf{Z}_k$ (2.6.8), que en esta particularización coincidirá con las diferencias de distancia calculadas a partir de las diferencias de tiempo de vuelo (TDOA). Es decir, con la diferencia de las distancias euclídeas de nuestra posición y las balizas del ULPS, estas diferencias de distancia son las obtenidas a través de los US.

$$\boldsymbol{Z_k}=\begin{bmatrix}\boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B}_2)-\boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B_r})\\ \vdots \\ \boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B}_m)-\boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B_r})\\ \vdots \\ \boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B}_M)-\boldsymbol{d}(\boldsymbol{X_k},\boldsymbol{B_r})\end{bmatrix} \tag{2.6.8}$$

Una vez actualizada las ecuaciones del EKF, éste por cada iteración realizada, nos devolverá una matriz con la estimación del vector de estados actualizado $\widehat{\mathbf{X}}_{\boldsymbol{k}}$ (2.24) y la actualización de la matriz de covarianza del error a priori $\mathbf{P}_{\boldsymbol{k}}$.

$$\widehat{\boldsymbol{X}}_{\boldsymbol{k}}=\begin{pmatrix}\widehat{\boldsymbol{x}}_{\boldsymbol{k}}\\ \hat{\mathrm{y}}_{\boldsymbol{k}}\\ \hat{\theta}_{\boldsymbol{k}}\end{pmatrix} \tag{2.6.9}$$

CAPÍTULO 3: DESCRIPCIÓN DEL MATERIAL UTILIZADO

En este capítulo se procede a realizar una descripción de cada uno de los materiales que han sido necesarios para poder desarrollar este trabajo.

3.1 IMU

La información de los sensores inerciales se obtiene a través de una unidad de medición inercial o IMU (*inertial measurement unit*). Una IMU es un dispositivo electrónico que mide e informa acerca de la velocidad, la orientación y las fuerzas gravitacionales de un dispositivo, usando una combinación de acelerómetros (sensores que miden la aceleración) y giróscopos (sensores que miden los cambios de orientación del dispositivo en los tres ejes de coordenadas).

La IMU es usada en un gran número de sistemas de navegación inercial (aviones, naves espaciales, etc). Los datos ofrecidos por la IMU son procesados por un computador, y permiten estimar la posición objeto a partir de la detección de los cambios en los ángulos de Euler *Roll, Pitch* y *Yaw*. Concretamente, en sistemas de navegación inercial como los aviones o las naves espaciales, a estos ángulos se le denominan ángulos de alabeo (*Roll*), cabeceo (*Pitch*) y guiñada (*Yaw*).

Particularmente, este trabajo ha optado por usar como IMU el Shimmer3 (Figura 14) [Shimmer, 2018], sensor diseñado para usos portátiles y de detección remota, además de ser altamente flexible y adaptable; de ahí que se utilice con frecuencia en diversas actividades como la vigilancia de la actividad, la ciencia del deporte, y las aplicaciones de construcción inteligente. Las características y especificaciones principales de esta IMU son las siguientes:

- CPU MSP430 de 24MHz.
- Detección inercial integrada mediante acelerómetros, giróscopos y magnetómetros (de los tres ejes), cada uno con rango seleccionable por el usuario en un software que proporciona el fabricante Consensys.
- Dos tipos de acelerómetros, que dan la opción entre ruido ultrabajo (*accel Low Noise*) o la gama amplia (*accel Wide Range*).
- Consumo de energía muy bajo y peso ligero.
- Interruptor deslizante para encender/apagar.
- Opción de conexión fácil vía Bluetooth para el envío de los datos a otro dispositivo o almacenamiento local de los datos en una tarjeta microSD.

- Conectores internos y externos para la expansión entre otras, de señales ECG (electrocardiograma) y EMG (electromiografía)).
- 5 LEDs de color en 2 ubicaciones para indicar el estado del dispositivo y el modo de funcionamiento, así como la funcionalidad de transmisión Bluetooth.
- Dimensiones del dispositivo: 51mm x 34mm x 14mm.

Figura 14: IMU Shimmer 3.

3.2 U-LPS

Los U-LPS son sistemas de posicionamiento local ultrasónicos de bajo coste. Los U-LPS usados en el TFG consisten en cinco transductores ultrasónicos Prowave 328St160 [Pro-Wave, 2014], distribuidos alrededor de una estructura cuadrada 70.7x70.7 cm (Figura 15). Cubre una superficie aproximada de 30m2 instalados en el techo a una altura de 3,5m. El microcontrolador para todos los U-LPS proporciona un enlace inalámbrico a un PC externo, para configurar fácilmente la frecuencia de muestreo, las secuencias de codificación y el esquema de modulación. Estos parámetros pueden modificarse en tiempo de ejecución, aunque en la práctica cada U-LPS sólo se configura una vez en una fase de configuración. En [Ureña, 2016] pueden ampliarse la información relativa al U-LPS utilizado.

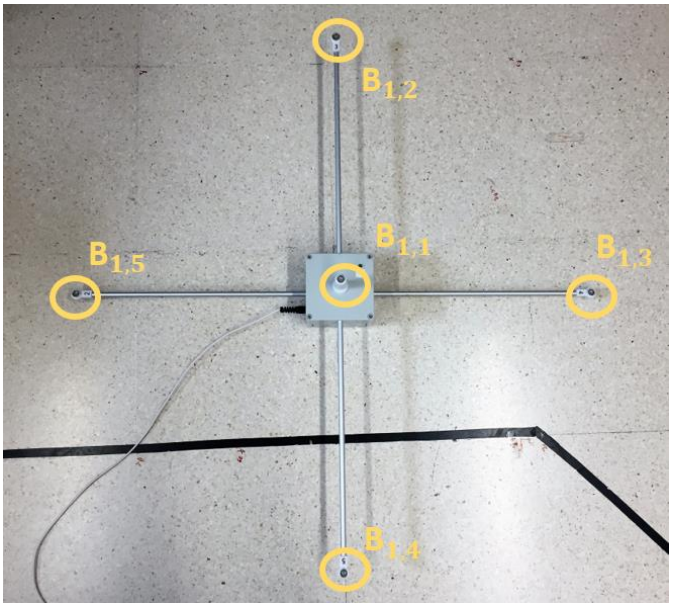


Figura 15: U-LPS.

3.3 Tablet

Para la ejecución del programa desarrollado en Android, se ha hecho uso de una Tablet, concretamente de la Samsung Galaxy Tab S2 9.7 WiFi (Figura 16), modelo SM-T813. Esta Tablet tiene un procesador de 64 bits y 8 núcleos ARM Cortex-A72 a 1.9 GHz. Además, dispone de 3GB de RAM y 32 GB de ROM. La versión Android que integra es la 7.0 (Nougat), con un nivel de API 24.

Figura 16: Samsung Galaxy Tab S2.

3.4 Receptor de ultrasonidos

Para la adquisición de los US, se ha hecho uso de un módulo de adquisición de US Figura 17 que incluye un micrófono MEMS SPU0414HR5H-SB [Knowles, 2018], digitaliza las señales ultrasónicas entrantes, las almacena en el buffer del receptor y envía mediante protocolo USB al dispositivo portable. Las dimensiones del dispositivo son 48mm x 18mm. Véase [Lindo, 2014] para más detalles sobre el módulo de adquisición.

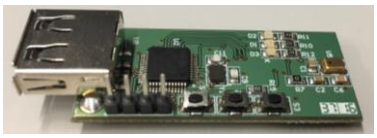


Figura 17: Receptor de ultrasonidos.

3.5 Matlab

Para el desarrollo de los algoritmos, y las pruebas iniciales, este trabajo ha hecho uso de la herramienta software matemática Matlab, concretamente la versión 2017b. Este software ofrece un entorno de desarrollo con un lenguaje de programación propio, lenguaje M.

3.6 Android Studio

Android Studio es un entorno de desarrollo integrado oficial para la plataforma Android. Se ha hecho uso de él para el desarrollo de la aplicación para dispositivos portables en la plataforma Android. Concretamente se ha hecho uso de la versión Android Studio 2.2.3.

3.7 Ordenador

El ordenador que se ha utilizado es un Cooler Master con procesador Intel Core i7-7700K a 4.20 GHz, con 8GB de memoria RAM y un procesador de 64 bits.

CAPÍTULO 4: DESARROLLO DE LA APLICACIÓN

4.1 Introducción y estructura del sistema

El objetivo final de este TFG ha sido el desarrollo de una aplicación Android que permite de forma robusta y flexible integrar información sensorial de las señales emitidas por un conjunto de Sistemas de Posicionamiento Locales Ultrasónicos (U-LPS), con la de sensores inerciales de una unidad de medición inercial (IMU) portada por el usuario, permitiendo el posicionamiento de personas a través de sus dispositivos portables en zonas interiores extensas.

Una vez que se tiene toda la base teórica (capítulo 2), se deben implementar todas estas partes en una misma aplicación junto a la sincronización de cada uno de los dispositivos involucrados con la misma. Para ello se han diferenciado las siguientes partes en esta sección:

- Programación y configuración del receptor ultrasónico (como se ha implementado la conexión vía USB entre el receptor US y el dispositivo portable).
- Programación y configuración de la IMU (donde se observará la configuración de la IMU necesaria para la app y su conexión Bluetooth al dispositivo).
- Programación del EKF para la fusión sensorial (particularización y programación de los algoritmos de fusión sensorial).
- Programación de las mejoras a implementar en el sistema (mejoras y modificaciones introducidas en el sistema para mejorarlo).
- Programación de la visualización de la posición (programación de la visualización de la posición en tiempo real).
- Diagrama de flujo de la aplicación (desglose de la aplicación en un diagrama de flujo para comprender su funcionamiento de una manera más visual).

Para facilitar la comprensión de los diferentes apartados de esta sección, se procede a realizar una breve introducción a Android y su programación.

En este trabajo la programación Android se ha implementado a través del entorno de desarrollo integrado para Android, Android Studio. Se debe tener en cuenta que

el objetivo final de la aplicación son los dispositivos móviles, y hoy en día existe una gran variedad de dispositivos móviles Android además de una gran variedad de versiones (actualizaciones) Android. Debido a esto lo primero que se debe tener en cuenta desde qué versión mínima va a funcionar la futura aplicación (*minimun SDK*).

A parte de elegir la mínima versión de ejecución, se debe elegir la versión objetivo (*target SDK version*), que equivale a la versión para la cual está prevista que se use la aplicación. Lo más común es usar como *minimun* SDK la mínima versión Android y como *target SDK* version la última, de esta manera se crea una aplicación que puede ser usada por el mayor número de usuarios posible. Como mínima versión para la aplicación aquí presente se ha hecho uso de la API 15: Android 4.0.3 (*IceCreamSandwich*), la cual funciona hoy en día aproximadamente en el 100% de los dispositivos. Como versión objeto se ha escogido la API 25: Android 7.1.1 (*Nougat*), es recordable usar una de las últimas versiones ya que puede depurar posibles fallos en versiones anteriores.

Otra de las partes que se encuentran en un proyecto Android es el descriptor de la aplicación o *manifest*. En el *manifest* encontraremos la solicitud de permisos que la aplicación requiere para su correcto funcionamiento (<*uses-permission*>), la aplicación aquí presente hará uso de permisos *Bluetooth*, Wifi, Localización y SD. También se encontrarán la declaración de <*uses-feature*>, con las que se informarán a una entidad externa sobre el conjunto de funciones hardware y software que la aplicación requiere, en esta aplicación se necesita para hacer uso del USB. En el *manifest* también se configurará el icono y el nombre de la aplicación (LOCATE-US) y las diferentes actividades de las que constará el proyecto además de la selección entre ellas de cual se configurará como actividad de arranque al iniciar la aplicación (*Laucher*).

Otro de los ficheros necesarios para la configuración del proyecto es el *Gradle* Scripts, donde se configurará la *minimun SDK* y el *target SDK* versión. Además, en el *Gradle* Scripts se añadirán las librerías que se necesitarán para el desarrollo de la aplicación. Las librerías necesarias para este proyecto han sido: *android.hardware.usb* para poder hacer uso del puerto USB, *com.felhr.usbserial* para configurar y hacer uso de la transmisión de la información por el puerto USB;

Apache *Commons Math* 3.4.1, para poder realizar operaciones matemáticas de matrices; *Androidplot* Core *Library* 0.9.8, para poder trabajar con gráficas; y la librería *Jcoord* 1.0 que permite convertir entre latitud/longitud (WG S84) y coordenadas UTM, necesaria a la hora de obtener las coordenadas para representar en Google Maps.

Por último, en un proyecto Android aparecen las actividades (*activities*), es la parte más importante del proyecto ya que es donde se desarrolla todo el código de los algoritmos a implementar y el manejo de la aplicación, además de la interface de ésta. Cada actividad está compuesta por dos partes, la parte lógica y la parte gráfica. En la parte lógica es donde reside el código fuente de la aplicación, y extiende de una clase que es un archivo *.java*. En esta parte también se programa la funcionalidad de la aplicación (algoritmos) y de cada uno de los eventos que se lleguen a configurar (pulsadores, cuadros de texto, etc). La parte gráfica de las actividades es un XML y se encarga del diseño de las diferentes pantallas de la aplicación (*layaouts*). En la parte gráfica se pueden encontrar todos los elementos necesarios para posteriormente ver por pantallas, como iconos, cuadros de texto, botones, barras de progreso, etc.

Particularmente, en esta aplicación para poder manejar variables en varias clases sin tener que usar constructores (función dentro de una clase con su mismo nombre, que permite manipular las variables que se introducen como entradas a la clase), se ha hecho uso de una clase llamada *Link_data*, en la cual se almacenan las variables que se desean tener acceso por diferentes clases.

A la hora de programar en Android se debe tener en cuenta que en el hilo principal se gestionan la interfaz de usuario de la aplicación además de las actividades, servicios y broadcast receivers. Debido a esto cuando se programa en Android se debe tener especialmente cuidado en no bloquear este hilo principal con tareas largas ya que bloquearán el resto de los componentes de la aplicación, y por supuesto también la interfaz de usuario. Esto es primordial tenerlo en cuenta ya que, si la aplicación tuviese ese comportamiento, el usuario tendría la sensación de tener una aplicación con una gran lentitud incluso con bloqueo de la misma. De hecho, Android muestra un dialogo ANR (aplicación no responde) automáticamente si la

tarea que se ejecuta en el hilo principal supera los 5s de duración. Este dialogo ANR muestra un mensaje al usuario notificando que la app no responde y da la opción de cerrarla o esperar. Teniendo en cuenta que en la aplicación aquí presente se implementan una serie de algoritmos que conllevan un proceso interno que el usuario no observa, se debe tener especial cuidado donde y como ejecutar todo ese procesamiento para evitar el bloqueo de la app ya que esto podría llevar a que el usuario desinstale la aplicación. La solución implementada en esta aplicación y en la mayoría de ellas es hacer uso de tareas asíncronas (hilos) que se ejecuten en paralelo al hilo principal, ya que estas tareas si pueden extenderse a un tiempo de ejecución superior de 5s y además en ellas no se gestionan otras actividades a parte de las especificadas por el usuario.

A la hora de trabajar con tareas asíncronas (hilos) Android Studio ofrece dos formas. Una mediante *Thread* que consiste en una clase extendida a hilo que en su interior contiene una función run, en la cual se introduce todo el código a ejecutar cuando el hilo se crea, es decir al arranque (*hilo.start()*). Además esta forma ofrece otras opciones de interactuar con el hilo como (*hilo.run (), hilo.join (), hilo.ggetPriority (),* etc.). La otra forma de trabajar con hilos es mediante AsyncTask, que es una solución pura que ofrece Android Studio. *AsyncTask* clase que viene ya implementada. Dentro de la clase *AsyncTask*, encontraremos una serie de funciones de las que se podrá hacer uso: *onPreExexute (), doInBackground (Void…params), onProgressUpdate (Integer…values), onPostExecute (Void aVoid), onCancelled ()*.

Se debe tener en cuenta que los hilos no pueden interactuar con la interface del usuario, sino que con ésta solo puede interactuar el hilo principal. Para visualizar algo por pantalla desde un hilo se deberá utilizar el manejador del hilo (*Handler*) o se tendrá que llamar al método *runOnUiThread ().*

4.2 Programación y Configuración del receptor de US y conexión USB

En este trabajo se hace uso de un receptor externo para adquirir las señales ultrasónicas emitidas por los U-LPS como se ha comentado anteriormente. Para poder hacer uso de las señales recibidas desde el dispositivo Android portable, la

conexión por la que se ha optado entre el dispositivo portable y el receptor ultrasónico es una conexión USB, con lo que los datos son transmitidos de la tarjeta de adquisición al dispositivo portable a través del puerto USB del dispositivo portable.

La página de desarrollo ofrecida por Android [Android, 2018] permite observar que Android admite una gran variedad de periféricos USB y accesorios USB a través de dos modos: accesorio USB o USB host. En el modo accesorio USB, el dispositivo externo es el que actúa como USB host, proporcionando la energía al dispositivo Android ante un intercambio bidireccional de información, por ejemplo, cuando se conecta el dispositivo portable a un PC y se sincronizan su datos y archivos multimedia. Por otro lado, se encuentra el modo USB host, en este caso el dispositivo Android actúa como USB host y es el que alimenta al dispositivo externo. El funcionamiento de ambos modos puede observarse en la Figura 18. Ambos modos son compatibles a partir de la versión Android 3.1 (API 12) en adelante, teniendo en cuenta que la aplicación se ha diseñado para una versión mínima de Android 4.0.3 (API 15) no presentará problemas de compatibilidad en ningún dispositivo en el que sea instalada.

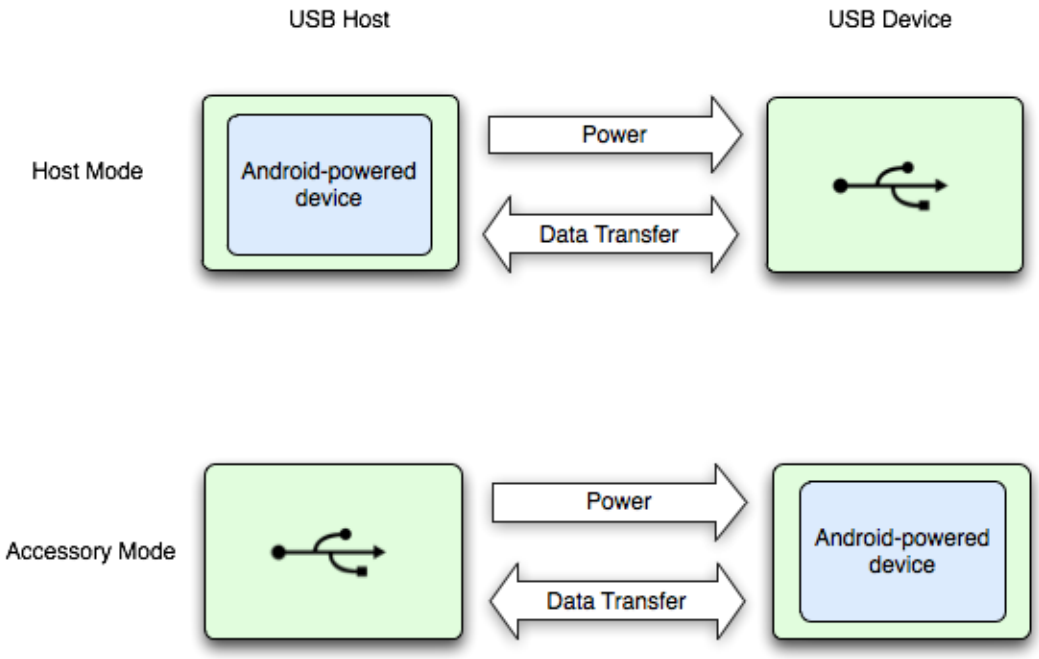


Figura 18: Comunicación USB Android: modos USB host y accesorio USB.

En este trabajo concretamente, se ha decido trabajar en modo USB host, de modo que es el dispositivo Android el que alimenta la tarjeta de adquisición de las señales ultrasónicas.

Para hacer uso de ese modo, como se ha comentado en la introducción de este capítulo, se hace uso de las librerías android*.harware.usb* y *com.felhr.usbserial*, donde se encuentran algunas de las API del puerto USB.

De la variedad de API del puerto del USB este trabajo ha hecho uso de las clases que se encuentran en la Tabla 1.

Clase	Descripción
UsbManager	Permite enumerar y comunicarse con dispositivos USB conectados.
UsbDevice	Representa un dispositivo USB conectado y contiene métodos para acceder a su información de identificación, interfaces y puntos finales.
UsbDeviceConnection	Representa una conexión con el dispositivo, que transfiere datos en los puntos finales. Esta clase permite enviar datos de ida y vuelta de forma sincronizada o asincrónica.
UsbSerialDevice	Permite configurar ya hacer uso de la información que se desea transmitir y recibir por el puerto serie USB.

Tabla 1: API USB host.

El primer paso que se debe realizar es pedir permiso para acceder al puerto USB, que se realiza en el *manifest:*

```
<uses-feature android:name="android.hardware.usb.host" />
```

También en el *manifest* se nos da la posibilidad de declarar los elementos *<intent-filter>* y *<meta-data>* que permitirán notificar si un dispositivo USB ha sido

conectado en el puerto USB de forma automática. Gracias al *<intent-filter>* se hará un intento de conexión a través de *android.hardware.usb.action.USB_DEVICE_ATTACHED*, mientras *<meta-data>* apunta a un archivo de recursos XML externo, donde se declara la información de identificación del dispositivo que se desea detectar. La declaración del código necesario en el *manifest* se muestra a continuación:

```
<intent-filter>
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"/>
</intent-filter>
<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
```

Antes de comunicar con el dispositivo USB, se debe otorgar permiso para que la aplicación envíe y reciba información por el puerto USB además de definir cómo se va a configurar la conectividad USB, ya que se puede realizar en el *manifest* o en el código de la aplicación a través de un *BroadcastReceiver* que se comentará más adelante.

Debido a que se conoce el dispositivo externo USB del que se va hacer uso, se declara una variable que ofrecerá la ruta del permiso.

```
public final String ACTION_USB_PERMISSION =
"com.hariharan.arduinousb.USB_PERMISSION";
```

Una vez declarada la variable, en el código de la aplicación se deberá realizar un filtro de intenciones para activar dicho permiso, además de configurar el receptor para futuros eventos, y declarar la configuración del futuro receptor USB que se conecte a través de un *BroadcastReceiver*.

```
IntentFilter filter = new IntentFilter();
filter.addAction(ACTION_USB_PERMISSION);
filter.addAction(UsbManager.ACTION_USB_DEVICE_ATTACHED);
filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);
registerReceiver(broadcastReceiver, filter);
```

Antes de configurar la conectividad del receptor, se debe enlazar el dispositivo Android con el receptor externo a través de la aplicación Android. Para ello se hace uso de la clase *usb.Manager* mediante la cual:

Se almacenará una lista con los dispositivos detectados:

```
HashMap<String, UsbDevice> usbDevices = usbManager.getDeviceList();
```

Se obtendrá el valor de identificación del dispositivo detectado:

```
device = entry.getValue();
```

Una vez que se tiene el valor de identificación y es el correcto del dispositivo deseado, se hace uso de dicho dispositivo para configurar su conectividad, es decir los parámetros de conexión del puerto serie en el *BroadcastReceiver*. En el *BroadcastReceiver* se comprobará si el usuario ha otorgado permiso para iniciar la conexión con el dispositivo identificado con el valor adquirido. En el caso de que se le haya otorgado, se abrirá el puerto serie (SerialPort) y se configurarán los parámetros en consecuencia, estos últimos son:

```
serialPort.setBaudRate(9600);
serialPort.setDataBits(UsbSerialInterface.DATA_BITS_8);
serialPort.setStopBits(UsbSerialInterface.STOP_BITS_1);
serialPort.setParity(UsbSerialInterface.PARITY_NONE);
serialPort.setFlowControl(UsbSerialInterface.FLOW_CONTROL_OFF);
serialPort.read(mCallback);// comienza a leer
```

Los parámetros seleccionados han sido, 8 bits de datos, 1 bit de parada, ningún bit de paridad y el Flow Control desactivado. Además, se ha configurado una velocidad de transmisión en baudios de 9600.

Además, también se ha configurado como se observa en la última línea del fragmento de código anterior, un método de lectura automático a través de *mCallback*, el método que activará es *UsbSerialInterface.UsbReadCallback*, el cual realizará automáticamente la lectura del puerto USB cuando se detecte cualquier dato a entrante. Concretamente cuando se obtengan en el puerto USB las señales ultrasónicas emitidas por los U-LPS, se activará una rutina automáticamente de lectura de estás.

4.3 Programación y Configuración IMU y Bluetooth

En este apartado se abordará todo lo necesario para poner en marcha la parte relacionada con la adquisición y el procesamiento de las señales obtenidas por los sensores inerciales. Para la realización de esta parte del trabajo se ha hecho uso del trabajo previo desarrollado en [Cervigón, 2017].

La primera parte que se debe abordar en este apartado es la configuración del hardware necesario para la obtención de los sensores inerciales, es decir la

configuración de la IMU, que en este trabajo como se ha comentado anteriormente se ha hecho uso concretamente del Shimmer3 como IMU.

El Shimmer3 se configura a través del software Consensys, este software lo proporciona el fabricante del Shimmer y permite programar el *firmware* al dispositivo, su frecuencia y los sensores de los que se quiere obtener medidas, ya que el Shimmer3 internamente ofrece un gran número de sensores como acelerómetros, giróscopos, señales ECG (electrocardiogramas), EMG (electromiografías), magnetómetro, además información adicional del dispositivo como el estado de su batería y una expansión externa de ADCs.

Para este trabajo como *firmware* se ha hecho uso de LogAndStream_0.7.1_UAH.txt, un *firmware* desarrollado dentro del propio grupo de investigación donde se ha desarrollado este trabajo (GEINTRA). LogAndStream_0.7.1_UAH.txt se desarrolló debido a la detección de fallos en el firmware proporcionado por el fabricante (LogAndStream_Shimmer3_v0.6.0), como la no inicialización del buffer de datos al producirse una desconexión entre el móvil y el Shimmer debido a la distancia entre ambos o la falta de batería de alguno de ellos, situaciones que se podían dar fácilmente al con estos dispositivos.

Como se ha comentado anteriormente, el Shimmer3 proporciona una amplia variedad de sensores, en este trabajo se ha configurado para que envíe únicamente la información de los acelerómetros (*wide range accelerometer*), la información de los giróscopos y del estado de la batería. La frecuencia de envío de las medidas configurada ha sido 128 Hz, con esta configuración se evita la pérdida de paquetes de información a la hora de hacer uso de comunicación *Bluetooth*, situación que sea observado experimentalmente al enviar a través del Shimmer3 medidas de otros sensores como la aceleración LN (*low noise*) y a frecuencias diferentes.

Se ha optado por una frecuencia por una frecuencia de 128 Hz debido a que en la literatura [Hervás, 2014] se encuentra que la frecuencia óptima para las aplicaciones PDR (algoritmos de posicionamiento relativo) es 100 Hz, y como por limitaciones del hardware (Shimmer3), la frecuencia debe ser múltiplo de dos, el múltiplo de dos más cercano es 128 Hz.

Una vez configurado el dispositivo Shimmer3, se debe tener especial cuidado en la lectura de la información en la recepción. Para ello se debe tener en cuenta que el Shimmer3 envía 18 Bytes, y estos se organizan de la siguiente manera:

- Byte [0]: Byte inicial, de valor 255 la primera vez y el resto de veces, se usa ese byte para enviar el byte 18 de la iteración anterior.

- Byte [1]: Byte ACK o byte de control de valor 0 en todas las iteraciones menos en la última (al mandar al Shimmer que pare de enviar información) que tiene valor 255, permitiendo saber si se está o no en la última iteración.

- **Bytes [2:4]:** *Timestamp*, contador interno del Shimmer.

- **Bytes [5:6]:** Batería del Shimmer [5:6].

- **Bytes [7:12]:** Giróscopos de los tres ejes (Giróscopo_x [7:8], Giróscopo_y [9:10] y Giróscopo_z [11:12]).

- **Bytes [13:17]:** Acelerómetro de los tres ejes (Acelerómetro_x [13:14], Acelerómetro_y [15:16] y Acelerómetro_z [17] (el segundo byte correspondiente al Acelerómetro_z es el enviado en la posición cero de la iteración siguiente)).

Tanto la información del *timestamp*, de la batería del Shimmer y de los acelerómetros, es enviada en formato *Little Endian*, es decir se envía primero el byte menos significativo (LBS) y por último el byte más significativo (MSB). En el caso de la información de los giróscopos, se envía en formato *Big Endian*, por cada giróscopo de cada eje, se envía primero el byte de más significativo (MSB) y por último el byte menos significativo (LBS).

Una vez configurado el dispositivo hardware Shimmer3, se debe configurar el protocolo de comunicación para enviar los datos que ofrece el dispositivo Shimmer3 al dispositivo portable. En este trabajo se ha optado por utilizar una comunicación entre dispositivos *Bluetooth* debido a que es la que ofrece el Shimmer3 y nuestro dispositivo portable utilizado también la soporta.

Para la configuración de la comunicación *Bluetooth* la aplicación desarrollada en este trabajo hará uso de una *activity* que permite realizar la conexión *Bluetooth* entre el Shimmer3 del que se hará uso y el dispositivo portable. Para la implementación de dicha *activity*, se ha partido de una *activity* anterior desarrollada en [Díaz, 2017].

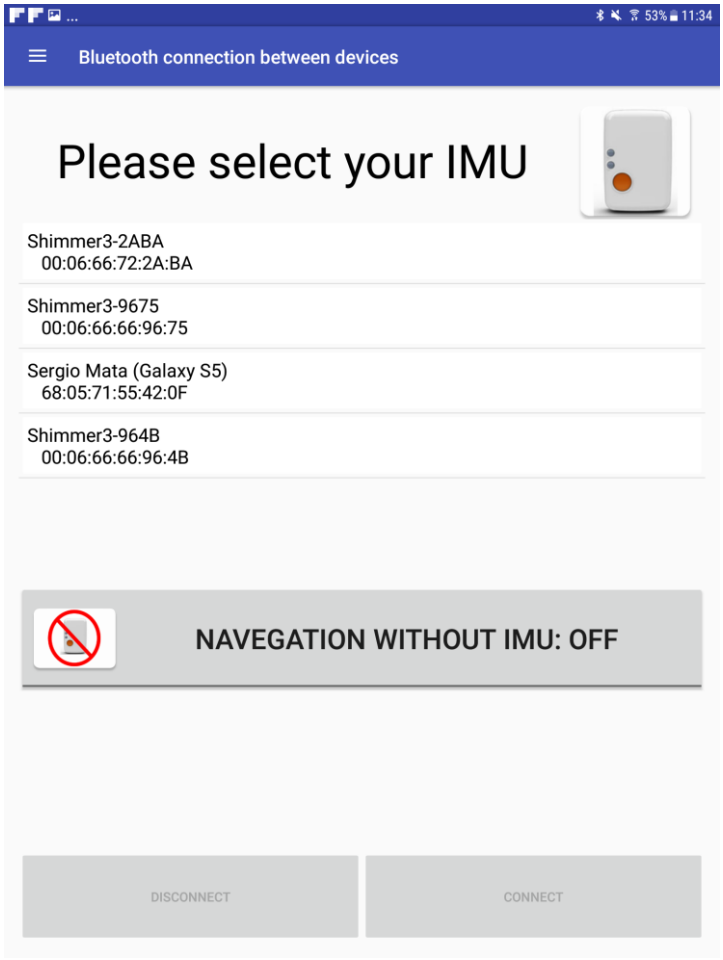


Figura 19:Captura de pantalla del layaout asociado a la actividad del Bluetooth.

En la Figura 19 se puede observar el diseño (*layaout*) asociado a la pantalla de la configuración de la conexión *bluetooth*. Se puede observar como esta pantalla ofrece una lista con los dispositivos visibles a los que se puede enlazar el dispositivo portable vía *Bluetooth*. Cuando se selecciona uno de la lista, se habilita el botón "*CONNECT*" y al presionarlo se enlazan ambos dispositivos, en este caso el dispositvo portable donde se ha cargado la aplicación y el Shimmer3 del que se quiere hacer uso (por ejemplo, el Shimmer3-2ABA).

Si cuando se solicita la conexión entre ambos dispositivos, no estuviese conectado en el dispositivo portable el *Bluetooth*, aparecerá en pantalla un permiso de

activación del *Bluetooth* como el que se muestra en la Figura 20. Esto se realiza gracias al adaptador *Bluetooth* (*BluetoothAdapter*), el cual representa el adaptador local del *Bluetooth* (radio *Bluetooth*), siendo el punto de entrada de toda iteracción vía *Bluetooth*. El *BluetoothAdapter* también permite ver otros dispositivos *Bluetooth*, consultar una lista de dispositivos sincronizados, o crear una conexión con uno de esos dispositivos *Bluetooth* remotos (*BluetoothDevice*) de dirección MAC conocida. Además, en *BluetoothDevice* se almacena información sobre el dispositivo conectado como su nombre, su dirección, su clase o su estado de conexión.

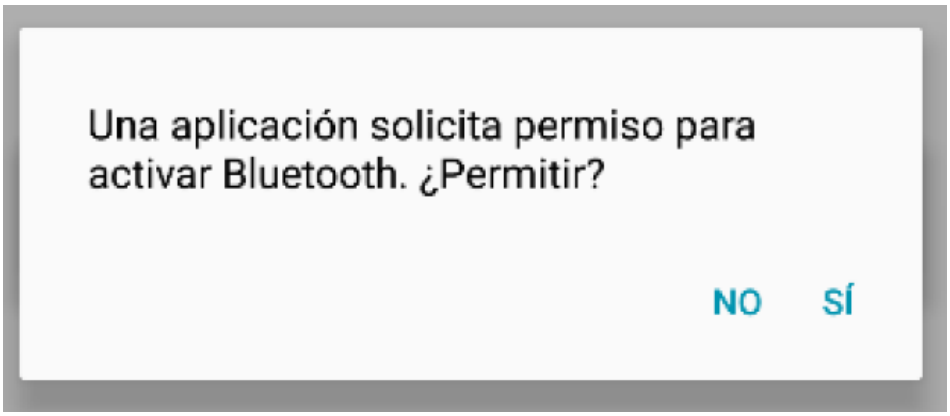


Figura 20: Permiso activación *Bluetooth.*

Una vez realizada la conexión entre ambos dispositivos, en la aplicación Android, se crea el *BluetoothSocket*, que es un punto de conexión entre la aplicación y el dispositivo *Bluetooth*. Gracias al *BluetoothSocket*, se crean los buffers *InputStream* (el cual permite el envío de información del dispositivo *Bluetooth* (Shimmer3) al dispositivo portable donde se está ejecutando la aplicación) y *OutputStream* (que permite el envío de información del dispositivo portable donde se está ejecutando la aplicación al dispositivo *Bluetooth*).

En la Figura 19 se puede observar como la pantalla de conexión *Bluetooth* también ofrece un botón "*DISCONNECT*" el cual se activa una vez realizada la conexión entre dispositivos y permite la desconexión de estos. Por último, podemos observar un botón de conmutación (*togglebotton*) "NAVIGATION WITHOUT IMU: ON/OFF" el cual permite comunicar a la aplicación si se va a desear hacer uso de un dispositivo IMU para la navegación o no.

En este trabajo la conexión *Bluetooth* se establece entre el dispositivo portable y el Shimmer3. Una vez realizada la conexión se necesita decirle al Shimmer3 qué debe hacer. Para ello, se hará uso del buffer *OutputStream*, donde se escribirá un valor determinado a través de la variable *packet_types* dependiendo de lo que se desee que haga el Shimmer:

- packet_types = 7 -> código a enviar al Shimmer para que inicie el envío de datos vía Bluetooth.

- packet_types = 32 -> código a enviar al Shimmer para que finalice el envío de datos vía Bluetooth.

- packet_types = 112 -> código a enviar al Shimmer para que empiece a mandar datos vía Bluetooth y grabe esos valores enviados en la SD del Shimmer.

- packet_types = 151 -> código a enviar al Shimmer para que finalice tanto de mandar datos vía Bluetooth y como de grabar en la SD del Shimmer.

- packet_types = 146 -> código a enviar al Shimmer para que grabe las medidas configuradas en la SD del Shimmer.

- packet_types = 147 -> código a enviar al Shimmer para que finalice de grabar en la SD del Shimmer.

- packet_types = 3 -> código a enviar al Shimmer para mande la frecuencia a la cual se ha configurado el Shimmer.

Hay que tener especial cuidado con la frecuencia del Shimmer3, ya que será necesaria para el correcto funcionamiento del algoritmo del EKF, el cual hará uso de la frecuencia para hallar el periodo de muestreo. Por ello en la aplicación se ha declarado una variable *frequency* en la cual a través del envío al Shimmer3 *packet_types* = 3, el Shimmer devolverá a través del buffer *InputStream* el valor de la frecuencia de éste y se almacenará en dicha variable y en el caso de que el valor de esa frecuencia sea diferente al configurado, se avisará al usuario por pantalla del error y se procederá a la desconexión de los dispositivos.

Una vez configurado el protocolo de comunicación y comprobado el correcto funcionamiento de ambos dispositivos vía *Bluetooth*, se procedería a la lectura de los valores enviados por el Shimmer3. Para ello en la aplicación se obtienen los valores del Shimmer3 a través del buffer *OutputStream* como se ha comentado anteriormente. Para la lectura de ese buffer se debe tener en cuenta que los bytes

recibidos están en decimal, con lo que se debe obtener esa medida de varios bytes e un único valor. Para ello se guarda en una variable el byte de mayor peso (MSB) multiplicado por 256 (1 byte tiene 8 bits, $2^8 = 256$) y se le suma el byte de menor peso (LSB), de la siguiente forma:

$$\mathbf{Variable = -((\sim(short)(msb*256+lsb)) +1)} \quad (4.3.1)$$

De este modo se guarda la información de los acelerómetros y de los giróscopos en sus respectivas variables. Una vez que se dispone de dichas variables, éstas son procesadas por la aplicación Android hasta la obtención de la posición relativa a través del algoritmo desarrollado en [Cervigón, 2017], con lo que no se va a entrar en detalle.

Dicho procesamiento lo primero que realiza es una calibración de las variables obtenidas por el Shimmer3 a través de las fórmulas de calibración ofrecidas por el software Consensys. Una vez calibradas las señales obtenidas se introducen en un EKF, que a partir de los valores de aceleración y giróscopo se obtiene una estimación de la orientación del Shimmer3, de los ángulos de Euler (*Roll*¸ *Pitch* y *Yaw*).

Una vez que se tienen los ángulos de Euler, a través de un algoritmo de posicionamiento se estiman las coordenadas *x* e *y* que contienen la posición relativa del usuario frente a un punto inicial durante toda la trayectoria realizada.

Como resumen de este apartado, y del trabajo realizado en [Cervigón, 2017], a partir de lectura de la información de los acelerómetros y los giróscopos ofrecidos por los sensores inerciales del Shimmer3, la aplicación aquí presente obtiene una posición relativa *x* e *y* de donde se encuentra el usuario respecto a un punto inicial.

4.4 Programación del EKF para la fusión sensorial

Como se ha comentado ya, la fusión sensorial se va a implementar a través de un filtro de Kalman Extendido. Para ello se tienen que implementar además del filtro de Kalman Extendido, las ecuaciones de las que hace uso para su actualización, que son la (2.6.1), (2.6.2), (2.6.3), (2.6.4), (2.6.5), (2.6.6), (2.6.7), (2.6.8) y (2.6.9).

Como paso intermedio, antes de realizar toda la implementación del EKF en la plataforma de Android Studio, se considera realizar un paso previo, donde se implementa el EKF con todas las funciones necesarias para su actualización en Matlab. Una vez que se tiene implementado en Matlab el algoritmo de fusión sensorial y funcionando correctamente, se procederá a su implementación y validación en Android.

Para la validación, tanto en el entorno de programación Matlab como en el entorno de desarrollo Android Studio, se propone generar una ruta manualmente conocida alrededor de un U-LPS conocido, y fusionar a través del EKF, la información que proporcionaría una IMU y las US en la ruta definida.

La ruta que se ha utilizado para realizar las pruebas de validación es una ruta que describe una circunferencia con 1 metro de radio, alrededor de un U-LPS, la circunferencia se ha descrito con 32 pasos. En la Figura 21 se muestra la rutar representada en Matlab, donde las cinco circunferencias negras representan las cinco balizas pertenecientes al U-LPS que se han definido como referencia para realizar los cálculos de los TDOA que ofrecerían los US. Las treinta y dos cruces verdes, definen cada paso de la trayectoria que se ha descrito para realizar el test. Para cada paso representado en la trayectoria, se obtendrá las TDOA desde esa posición gracias a las US. Y además para cada paso realizado se obtendrá la diferencia de estos, es decir la anchura de paso (SL) y la diferencia del ángulo ($\Delta\theta$), dos parámetros que se obtendrían a través de la IMU. Una vez que se tienen las TDOA, la anchura de paso (SL) y la diferencia del ángulo ($\Delta\theta$), se actualizarán todas las ecuaciones del EKF de manera dinámica para cada paso y se obtendrá una estimación de cada posición, que al estar en una situación en la que no interviene ruido y se conoce la ruta, la ruta real y la estimada deberá ser igual.

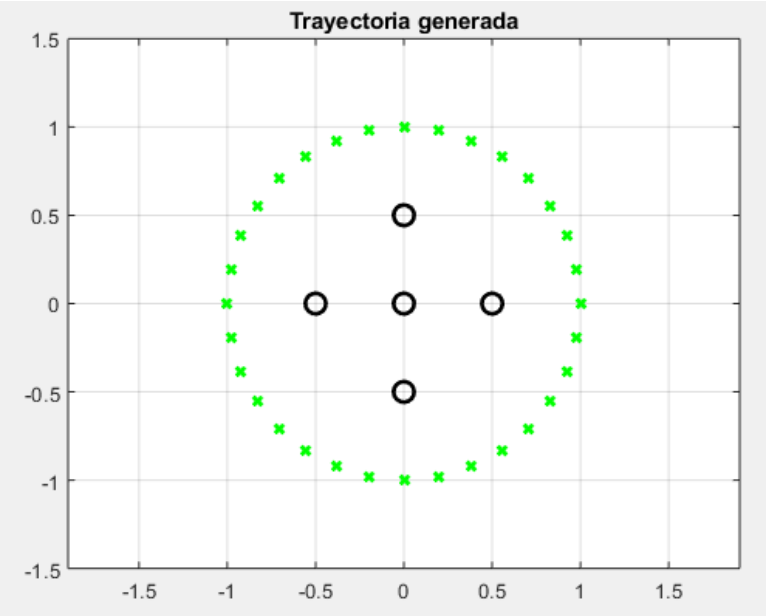


Figura 21: Ruta generada para la prueba.

En la Figura 22, se puede observar la comparativa entre los resultados de la trayectoria real generada (a la izquierda de la figura, con marcadores verdes), junto con la trayectoria estimada a través del EKF mediante la fusión sensorial (a la derecha de la figura, con marcadores azules).

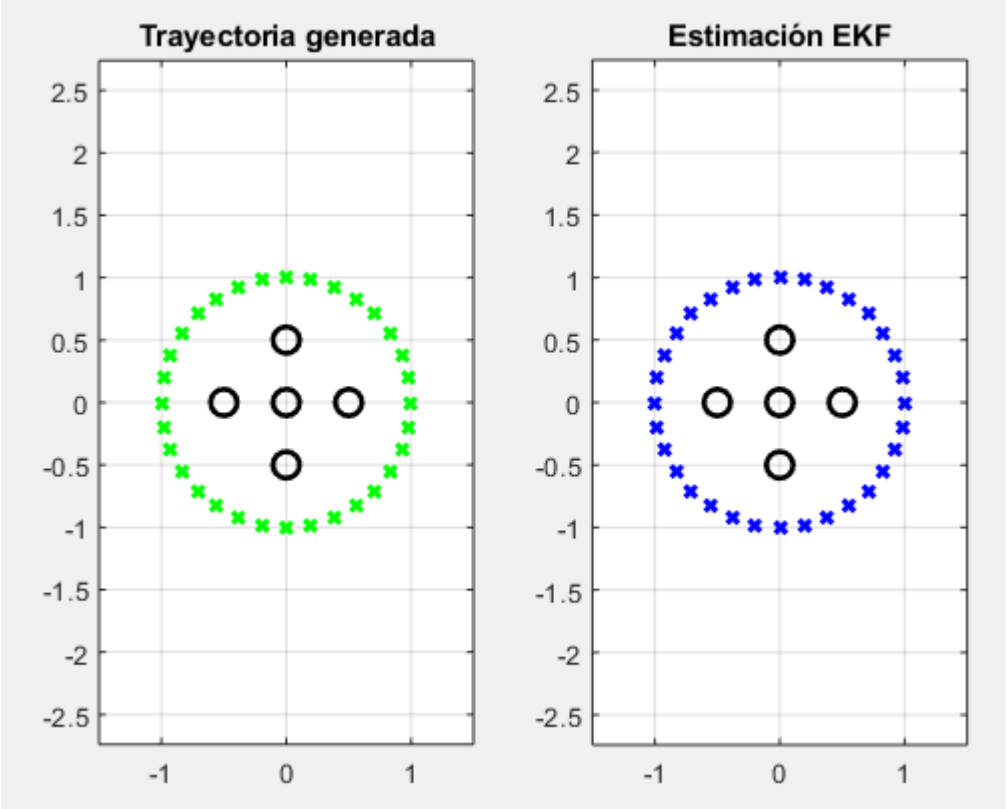


Figura 22: Comparativa ruta real y estimada en Matlab.

Se puede observar visualmente, como la estimación a través del filtro de Kalman es prácticamente igual a la generada, ya que como se ha comentado anteriormente, para este estudio no se han introducido ruidos ni en las medidas de las US ni en las medidas de los sensores inerciales. Realizando un estudio más detallado se obtiene un error medio de la trayectoria estimada a través del EKF con respecto a la trayectoria real de $4.58*10^{-16}$ m, con una desviación estándar de $5.21*10^{-16}$ m, valores que se pueden considerar prácticamente cero.

A través de una representación de una CDF (función de distribución acumulada) en la Figura 23 se puede observar que en ningún caso el error de la estimación de la ruta a través del EKF no ha superado los $1.8*10^{-15}$ m, y que en el 80% de los casos, el error era inferior a los $1.4*10^{-15}$ m.

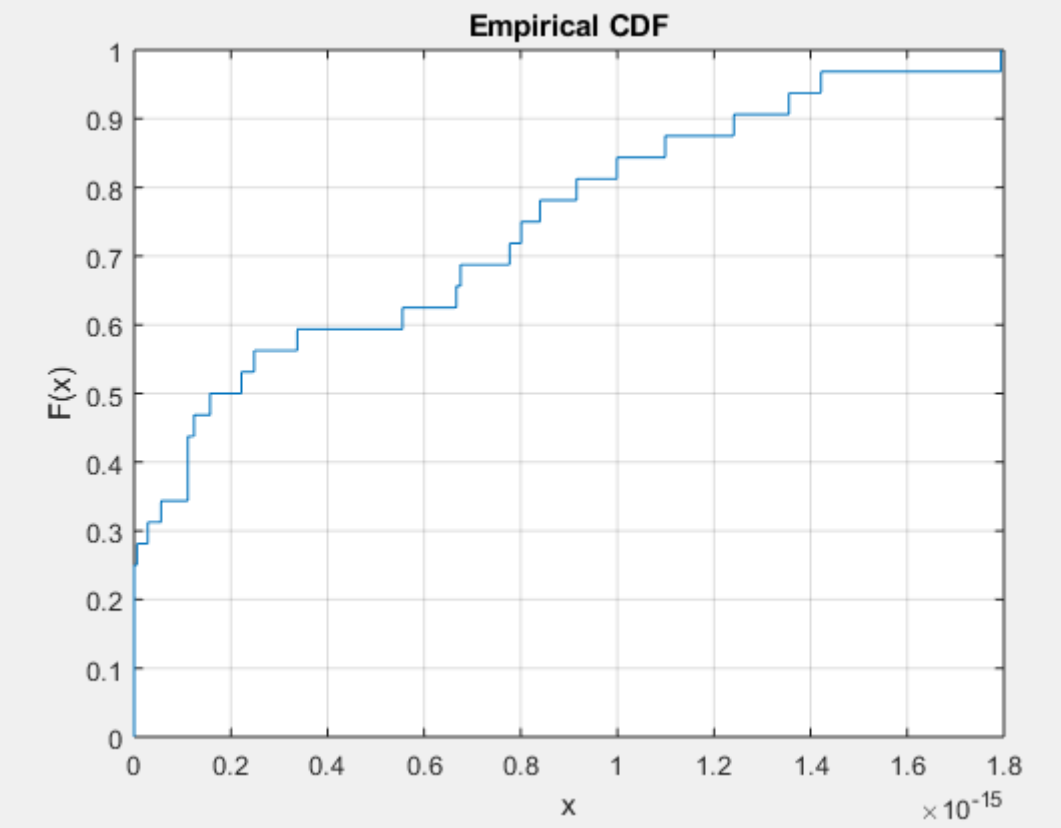


Figura 23: CDF error estimación EKF.

Gracias a toda esta información se puede confirmar el perfecto funcionamiento del algoritmo de fusión sensorial entre la información de las US y de los sensores inerciales, a través de un EKF en la plataforma de Matlab, el cual ofrece errores prácticamente nulos en la estimación de una trayectoria real generada de 32 pasos ante la ausencia de ruido.

Una vez verificado el correcto funcionamiento del algoritmo de fusión sensorial en Matlab, se procede a la implementación de este en Android y su verificación. Para ello se ha introducido la misma ruta en Android, y se realiza la estimación del mismo modo que en caso anterior. En la Figura 24, se puede observar como la ruta generada en Matlab e introducida en Android es igual a la ruta estimada en Android. Además, haciendo un estudio más detallado del error de la estimación de la ruta, se obtiene un error medio de la trayectoria estimada a través del EKF con respecto a la trayectoria real de $1.51*10^{-15}$ m, con una desviación estándar de $7.46*10^{-16}$ m, valores que se encuentran en el mismo orden que los obtenidos en Matlab, y que al igual que en la situación anterior, se pueden considerar cero.

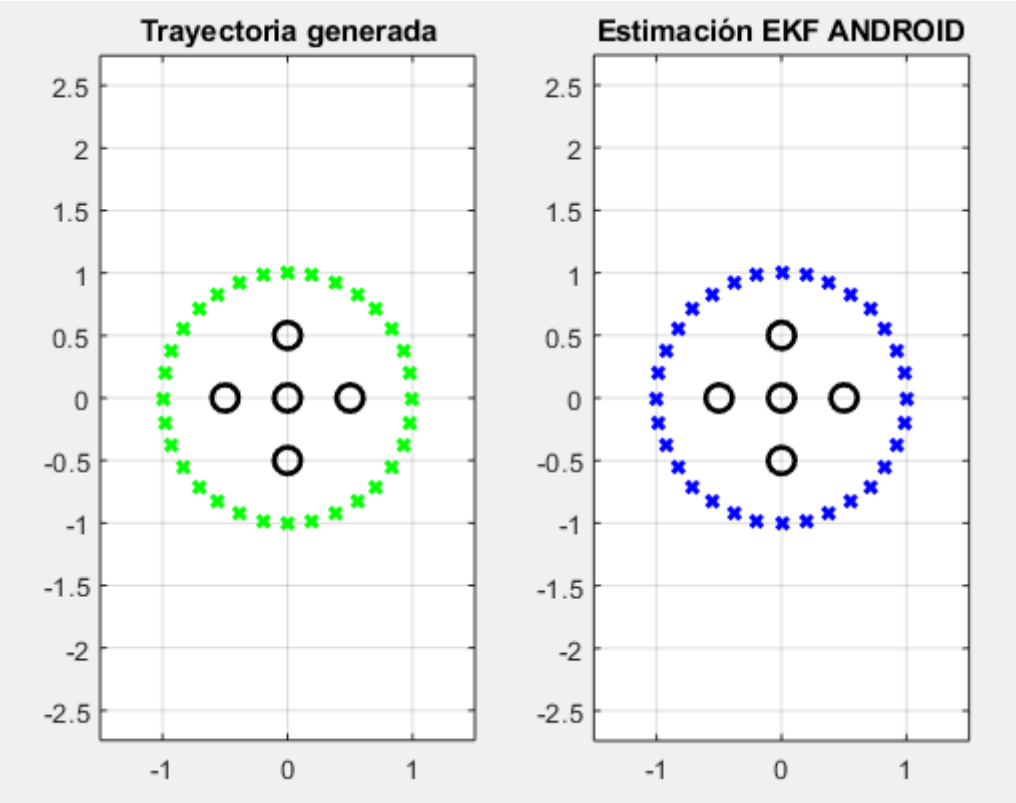


Figura 24: Comparativa ruta real procesada en Matlab y la estimada en Android.

Para continuar con la verificación del algoritmo en Android, al igual que en el caso anterior, se ha realizado una CDF en la Figura 25 y se observa como el máximo error obtenido es de $3*10^{-15}$ m y el 80% de los casos el error se encuentra por debajo de $2.25*10^{-15}$ m.

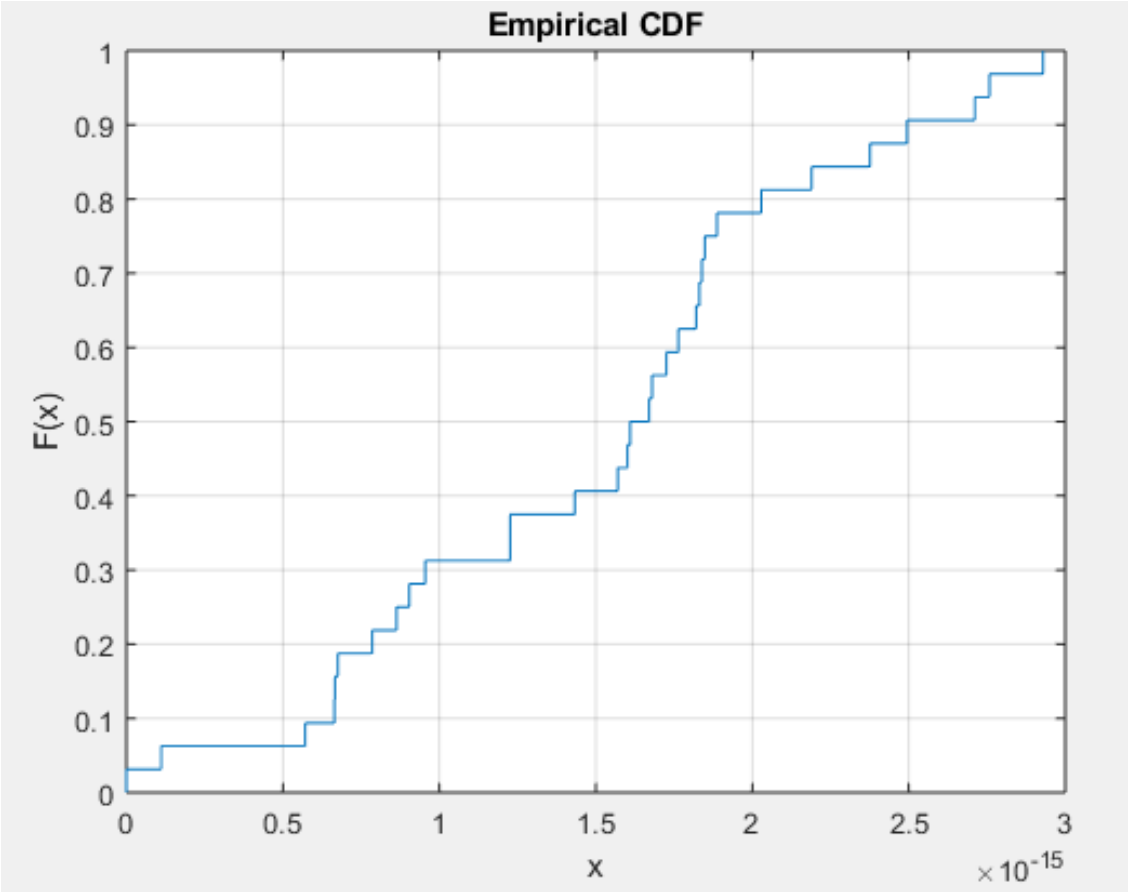


Figura 25: CDF error estimación EKF Android.

Con la realización de este estudio, y la obtención de estos valores, se ha podido verificar el correcto funcionamiento del algoritmo de fusión sensorial a través de un EKF en la plataforma Android, donde se usará posteriormente en la aplicación desarrollada en este trabajo para la navegación de personas en entornos interiores mediante un dispositivo portable.

4.5 Programación de las Mejoras a implementar en nuestro sistema:

En este apartado se van a exponer las diferentes mejoras que se han implementado en el sistema con el fin de robustecer el sistema y mejorar su rendimiento. Para ello se han aprovechado ventajas de la infraestructura y el sistema implementado junto la flexibilidad que ofrecen los diferentes algoritmos de los que se han hecho uso.

4.5.1 Selección del pico de referencia de los US

4.5.1.1 Justificación teórica

Como se ha comentado en el capítulo 2, la obtención del posicionamiento a través de los ultrasonidos se basa en la obtención de las TDOA, es decir la obtención de las diferencias de tiempo de vuelo de las diferentes señales emitidas por cada uno de los emisores de un U-LPS. Ya que una vez que se tiene las TDOA se puede obtener las diferencias de distancias y con ello hacer uso del algoritmo de Gauss-Newton o del EKF para la estimación de la posición. Se debe recordar que se hace uso de diferencias de tiempos de vuelos porque, con el fin de reducir el despliegue de infraestructura, no se ha considerado la sincronización entre el emisor y el receptor, con lo que no se sabe en qué instante se emitieron las señales que se reciben en el receptor.

Concretamente en el apartado 2.2 se explica cómo una vez que se tienen los instantes de llegada de cada señal emitida por cada uno de los emisores del U-LPS, los TDOA se calculan eligiendo uno de los instantes obtenidos como instante de referencia (se hablará de baliza de referencia ya que ese instante se obtiene a partir de la señal emitida por una de las balizas), y posteriormente se calcula la diferencia entre cada uno del resto de instantes recibidos con el elegido como referencia, de esta manera se obtienen cuatro TDOA.

La selección de la baliza de referencia de forma dinámica para la obtención de las diferencias de tiempo de vuelo se puede abordar de diferentes formas. Teniendo en cuenta de que se dispone de un buffer con cinco picos de correlación que indican los infantes de llegada, una de las primeras formas que se podría tener en cuenta es elegir como baliza de referencia el primer pico de correlación recibido. De esta manera, la obtención de las diferencias de tiempo de vuelo sería tan sencilla como ir buscando el siguiente pico obtenido y diferenciar los instantes de llegada y el tamaño de la señal emitida. Esto se puede realizar así, porque se sabe que la emisión de los códigos se realiza de forma consecutiva, así que, al haber elegido la primera como referencia, se garantiza que las demás se van a encontrar de forma ordenada y consecutiva a esta.

El problema de optar por esa forma de selección de baliza de referencia es que, si el primer pico de correlación recibido hubiese sufrido una gran distorsión y con ello se hubiese recibido el instante de llegada con un error, ese error se arrastraría a cada una de las cuatro medidas de TDOA, ya que para el cálculo de éstas se haría siempre uso de la baliza de referencia la que lleva el error. Para ello una de las mejores opciones que se presentan, por la cual ha optado este trabajo, es la elección como baliza de referencia, el pico de correlación recibido con mayor energía, de esta forma al ser el pico con mayor energía se entiende que es el que menos distorsión ha sufrido hasta la recepción y con ello el que menor error lleva para posteriormente hacer el cálculo de los TDOA. A diferencia de la situación anterior, si en el buffer de recepción se tuviese un pico con un error, su energía sería menor, con lo que no sería seleccionado como pico de referencia y ese error solo se vería propagado a una de las TDOA obtenidas, a diferencia de la situación anterior donde se veían afectadas las cuatro TDOA. Es decir, como la precisión de los TDOA es proporcional a la calidad de los códigos recibidos, se opta considerar como código de referencia el código de mayor energía recibido, de este modo conseguimos los TDOA de mayor precisión por cada recepción.

Una de las dificultades que se presentan al hacer uso como código de referencia del pico de mayor energía es que no se sabe ni se pude garantizar si dicho pico de referencia va a ser el primero, el segundo o incluso el quinto de los recibidos. Con lo que, debido a esto, el cálculo de las TDOA no es tan sencillo como ir recorriendo el buffer de manera consecutiva como pasaría en la situación anterior.

En la Figura 26 se puede observar una situación en la que en el buffer de recepción se ha elegido como baliza de referencia para el cálculo de las TDOA el primer pico de correlación obtenido. Como se observa el resto de picos se encuentran a la derecha de éste, con lo que a la hora de realizar la diferencia de cada pico con el de referencia el cálculo es más sencillo.

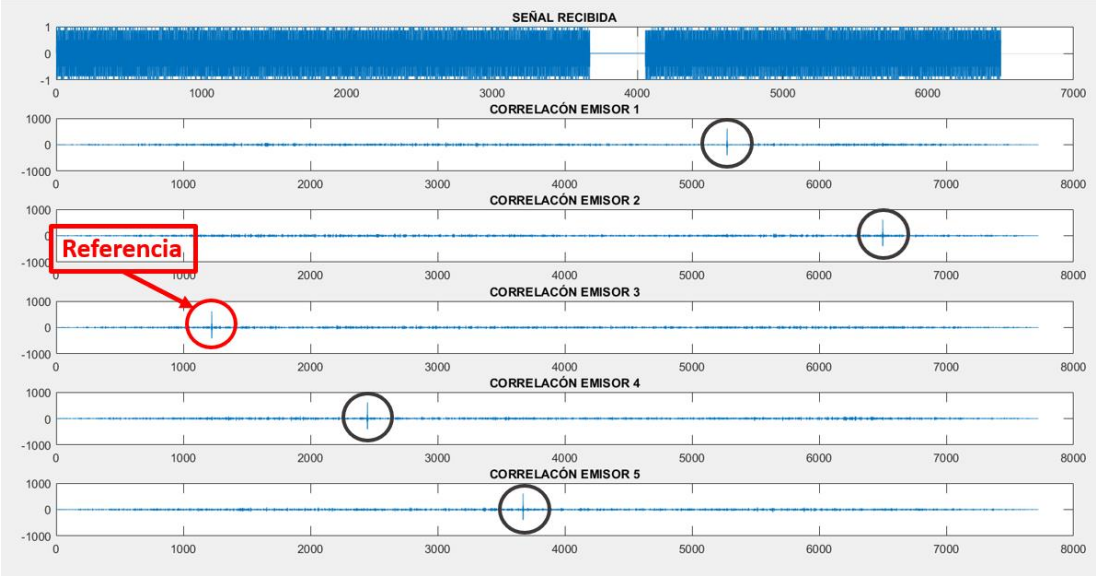


Figura 26: Primer pico, pico de referencia.

En la Figura 27 se observa una situación teniendo en cuenta que el pico de referencia se elige de la forma que ha optado este trabajo, es decir eligiendo el de mayor energía. En este caso el pico de mayor energía es el pico de la señal emitida por el emisor 1. Como se observa en la Figura 27 en esta situación en particular este pico es el tercero recibido, con lo que para obtener las TDOA no será tan sencillo como en la situación anterior.

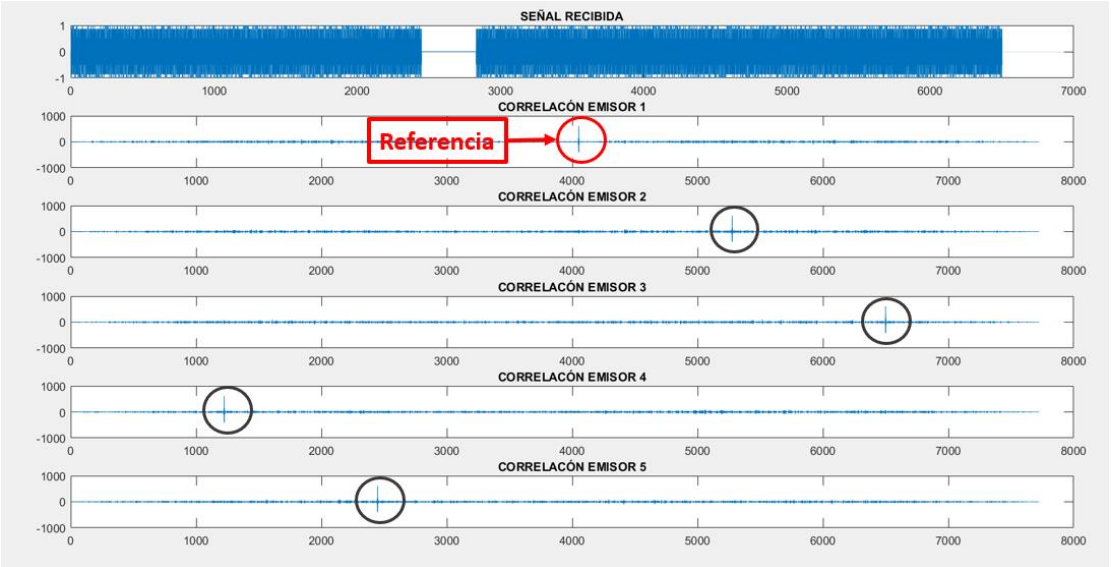


Figura 27: Pico de mayor energía como pico de referencia.

En esta situación el programa debe tener en cuenta si el resto de los picos recibidos se encuentra por delante o por detrás del pico de referencia, y dependiendo de eso el cálculo de cada TDOA cambiará ya que si el pico de la baliza 2 se encuentra a la derecha del pico de la baliza 1 (este caso la de referencia), el TDOA en esa situación será directamente el instante de llegada del pico del emisor 2 menos el instante de llegada del pico del emisor 1 y menos la longitud del código 2. En cambio, si el pico de la baliza 2 se encontrase a la izquierda del pico de la baliza 1, en la diferencia habría que tener en cuenta quitar el tamaño de todos los códigos que se encuentran entre estos, que sería el código 5, 4 y 3, además del *gap* que se ha introducido entre el código del emisor 5 y el emisor 1. Así, se ha tenido que realizar un algoritmo de cálculo de las TDOA dinámico porque no se sabe cuál va a ser el orden de recepción de los códigos en el buffer y además no se sabe cuál va a ser el de mayor energía, es decir el de referencia, con lo que el programa debe adaptarse de forma dinámica a todas las situaciones.

4.5.1.2 Resultados experimentales

Una vez que se ha justificado de forma teórica la selección como pico de referencia del pico de mayor energía, se procede a realizar un estudio, en el cual, se han tomado las señales ultrasónicas emitidas por los U-LPS desde un punto fijo. Una vez que se han adquirido las medidas, se han correlado y se ha obtenido la posición del punto a través del algoritmo de Gauss-Newton. Para la obtención de la posición se ha realizado el algoritmo de Gauss-Newton de dos formas, teniendo en cuenta las TDOA calculadas a través de la selección como pico de referencia, el pico de mayor energía, y también, por otro lado, seleccionado el pico de referencia el primer pico obtenido en el buffer de recepción. De esta manera se han obtenido dos *arrays* de posiciones, dependiendo del pico de referencia que se haya elegido. Una vez que se han obtenido esas posiciones, se ha calculado el error con respecto a la posición real donde se han adquirido las medidas, y se obtenido las media y la desviación típica para observar la diferencia entre las dos situaciones.

En la Figura 28 se observa la prueba realizada, donde las circunferencias negras representan las balizas del U-LPS a través de las cuales se han enviado las señales US. A la izquierda de la imagen se observa el resultado de las posiciones (puntos

rojos) eligiendo como pico de referencia el pico de mayor energía, y a la derecha seleccionado el primer pico recibido (puntos verdes).

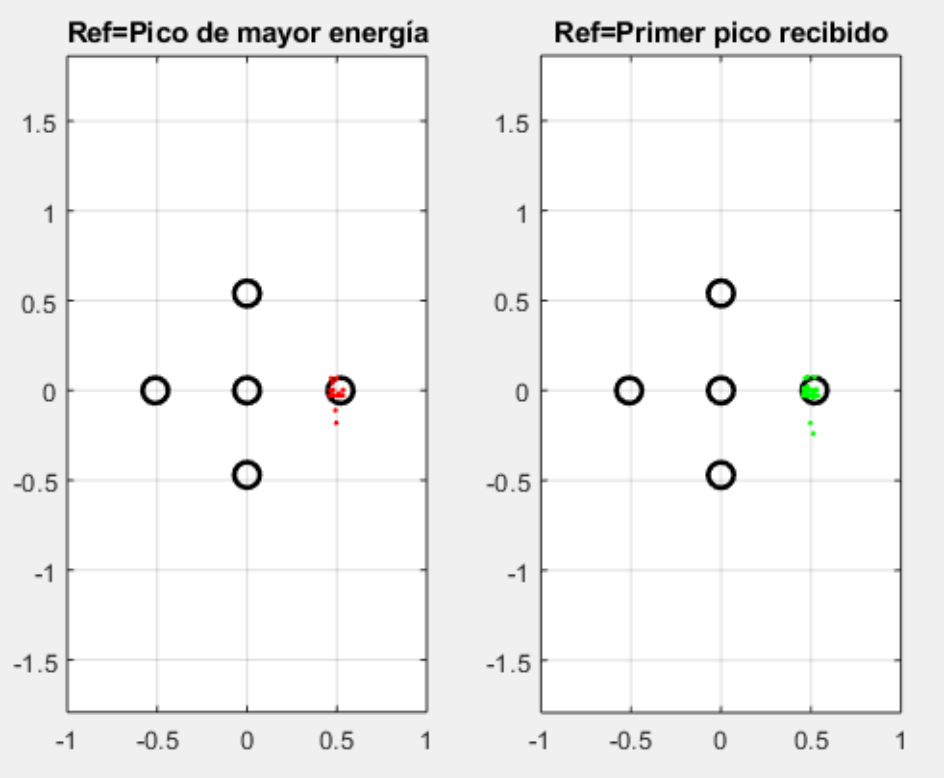


Figura 28: Diferencia selección pico de referencia.

El número de medidas representado ha sido 104, y a simple vista se puede observar como al elegir el pico de mayor energía se observa una menor nube de puntos con respecto al primer pico recibido, esto se debe a que la precisión es mayor y se han obtenido más valores repetidos. Se debe de tener en cuenta que en algunas ocasiones se puede dar que el primer pico recibido es el pico de mayor energía, pero si se selecciona siempre el pico de mayor energía se obtiene una media del error de 0.0452 m, es decir un error de 4,5 cm aproximada mente, y con una desviación estándar de 0.0242 m, es decir de 2.4 cm aproximadamente. Seleccionando como pico de referencia el primer pico recibido, se obtiene una media del error de 0.0465 m, es decir de 4.7 cm aproximadamente con una desviación estándar de 0.0290 m, es decir 3.0 cm aproximadamente.

Observando estos resultados se puede confirmar que seleccionando como pico de referencia el pico de mayor energía recibido, el resultado es que las medidas obtenidas son más exactas (menor media del error), y más precisas (menor desviación típica del error).

4.5.2 Implementación del enventanado

Una vez que se ha identificado el pico de referencia para la obtención de los TDOA, el algoritmo dinámico procede a buscar la posición del instante de llegada del resto de los picos para reordenarlos y obtener la diferencia de tiempo de vuelo.

Con el objetivo de conseguir un sistema más robusto y eficaz, al sistema dinámico del cálculo de TDOA se le ha añadido un filtro que enventana las medidas antes de proceder a realizar el cálculo de las TDOA.

Debido a que los U-LPS presentan una cobertura finita, experimentalmente se ha comprobado que la distancia máxima calculada a partir de la diferencia de tiempo de vuelo entre dos códigos recibidos, a partir de 0.7m deja de ser fiable. Para solventar esta especificación se ha implementado un filtro en el que se han enventanado los picos de recepción de manera que a partir del pico de referencia se descartan el resto de los picos que se encuentran fuera de esa ventana.

En ocasiones, aparecen picos que se encuentran fuera de la ventana, es decir que no se reciben dentro del margen de instantes donde físicamente deberían estar. Esto puede deberse a efectos como el multi-camino causado por el rebote de la señal emitida o ruidos en el ambiente cercanos a la frecuencia de trabajo. Estos picos en algunas situaciones pueden llegar a ser mayores que los picos de correlación de la señal recibida deseados. Estos picos introducen en el sistema grandes errores, por ello se ha configurado una ventana que elimine directamente esos picos con el objetivo de evitar que introduzcan errores en el sistema además de evitar que el sistema invierta tiempo en realizar unos cálculos que posteriormente van a ser erróneos.

La ventana seleccionada es de +/-300 muestras ya que la distancia máxima entre dos picos es de 0.7m, por la geometría del U-LPS, la velocidad del sonido en el aire es de 343m/s y se trabaja en la recepción a una frecuencia de muestreo de 100 kHz, con lo que $\frac{0.7m}{(343\ m/s)} * 100 * 10^3$=204.81muestras. Con el fin de dejar un mayor margen y aumentar el número de medidas válidas, se ha optado por realizar un enventanado de +/-300muestras.

De esta manera en la Figura 29 se puede observar el funcionamiento del filtro del enventanado, donde los picos que se encuentran fuera de la ventana, aun teniendo una mayor energía, son descartados. En la Figura 29 se toma como referencia el pico de la correlación 2 ya que es el pico de mayor energía, y se enventana el resto de los picos obtenidos con las correlaciones, descartando los picos que se encuentran fuera de las ventanas.

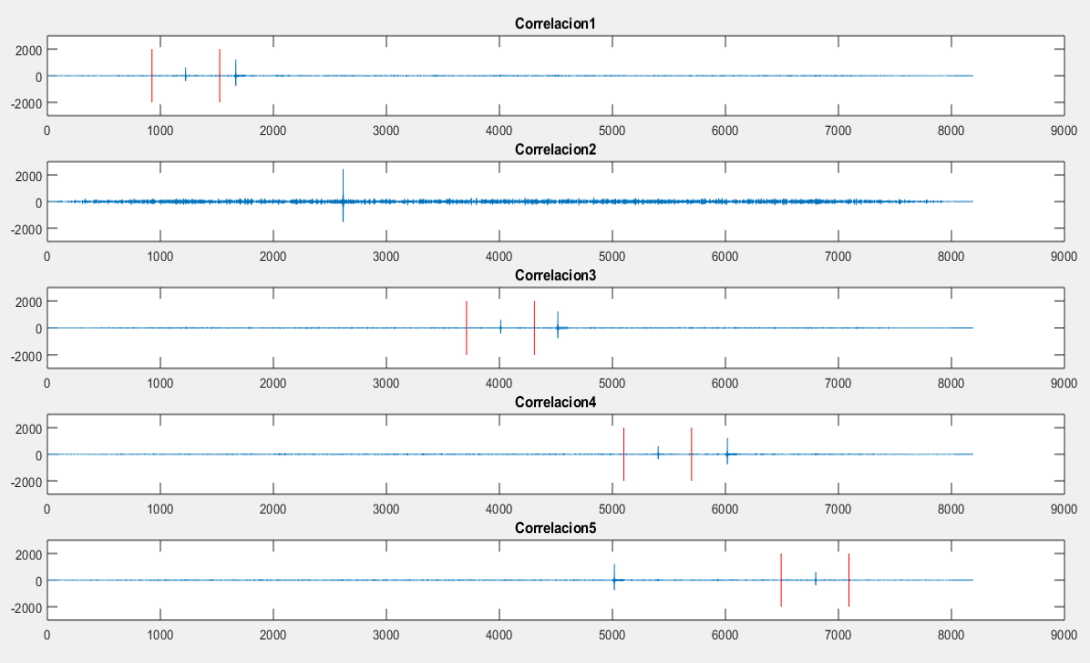


Figura 29: Proceso de enventanado.

Este filtro se ha implementado de manera dinámica ya que el pico de referencia puede variar en función de la situación. Así, cuando una baliza ha sido enventanada, o eliminada directamente porque no presenta suficiente energía (no funcione correctamente la emisión), dicha baliza es descartada ya que no se puede garantizar la fiabilidad de esa medida. Si solo se descarta una baliza, se dan por válidas las otras 4 balizas para obtener tres TDOA, pero en el caso de que se descarte más de una baliza, se elimina el buffer de recepción y se procede a adquirir otro evitando invertir tiempo en realizar el cálculo de los TDOA que ofrecerían una estimación de la posición errónea. Que se den por válidas las medidas en las que solamente una de las balizas se ha eliminado se debe a que como se ha explicado en el capítulo 2, el sistema consta de cinco balizas aun solo necesitando cuatro para la obtención de la posición con el objetivo de tener un sistema redundante en caso de fallo de alguna de las balizas.

4.5.3 Implementación Gauss Newton de manera dinámica para el posicionamiento con 4 balizas

Ya que el sistema es capaz de identificar si alguna de las balizas tiene algún problema, y con ello descartarla del set de medidas válidas, el siguiente paso es modificar el algoritmo de resolución Gauss-Newton para que sea capaz de trabajar de forma indistinta con cuatro TDOA o con tres TDOA. De esta manera en el caso de que se disponga de toda la información proporcionada por las cinco balizas, se aprovechan todas ellas, y en el caso de que alguna de ellas falle, se conseguirá seguir posicionar sin problemas. Se consigue así un sistema robusto y flexible capaz de adaptarse a diferentes situaciones de forma dinámica.

Para ello el primer paso fue modificar el algoritmo de Gauss-Newton en Matlab y posteriormente en Android, comprobando el correcto funcionamiento de ambos.

Para poder hacer dinámico el algoritmo de posicionamiento Gauss-Newton, se debe tener en cuenta que éste se basa como se ha comentado en el apartado 2.3, en el cálculo de la tolerancia a partir de la ecuación (2.3.1), con la que posteriormente se irá modificando la posición inicializada, hasta encontrar la solución de la ecuación generada por la trilateración hiperbólica, la cual será la posición a estimar.

El resultado de la ecuación (2.3.1) en este trabajo será **X**, siendo está una matriz 2x1, debido a que la estimación de la posición se está haciendo para 2D, con lo que se devolverá la posición x e y. Para la obtención de **X**, como se ha comentado en el apartado 2.3, es necesaria la matriz **B**, la cual se define como la diferencia de las diferencias de distancias estimada y medida. Si se hace uso de 5 balizas se obtendrán 4 diferencias de distancias, y con ello 4 diferencias de las diferencias de distancias estimadas y medidas. Es decir, las dimensiones de **B** en el caso de que se obtengan medidas de las 5 balizas será 4x1, pero en el caso que se obtengan medidas únicamente de 4 balizas, se obtendrán 3 diferencias de distancias, y con ello la matriz **B**, tendrá unas dimensiones de 3x1. La matriz **A**, al ser la derivada de las diferencias de distancias estimadas con respecto a cada una de las variables de la posición, tendrá unas dimensiones de 4x2 en el caso de que se tengan 5 balizas, ya que se contará con 4 diferencias de distancias, es decir habrá que realizar 4 derivadas, y como se desea obtener la posición en 2D, como variables respecto a las

que se debe derivar habrá 2, x e y. En el caso de que se cuente con 4 balizas únicamente, las variables seguirán siendo 2, x e y, pero en este caso e contará con 3 diferencias de distancias, con lo que las dimensiones de **A** deberán pasar a ser 3x2.

Es decir, para poder realizar de forma dinámica el algoritmo de Gauss-Newton, se deben declarar de forma dinámica las dimensiones tanto de la matriz **A** como de la matriz **B** en función del número de balizas recibidas.

Para observar el correcto funcionamiento del algoritmo dinámico, primero se va a realizar éste en Matlab, y se comprobará el funcionamiento para una misma posición obteniendo ésta, haciendo uso tanto de 5 balizas como de 4 balizas.

Primero se ha elegido una posición aleatoria de referencia la que posteriormente será estimada a través del algoritmo de Gauss-Newton sin tener en cuenta ruidos para observar únicamente el correcto funcionamiento del algoritmo. Esa posición ha sido x=-0.0869 e y=0.4756. A través del comando '*dist*' de Matlab, se han calculado las distancias euclídeas desde ese punto de referencia a cada una de las cinco balizas que se han situado en un U-LPS con baliza central en (0,0,3). Una vez que se tienen las distancias, se han obtenido las diferencias de distancias con respecto a la baliza central, obteniendo un array de TDOA = [0.0549, -0.0374, 0.0267, 0.1171].

Una vez que se tienen las diferencias de distancias, se han introducido en el algoritmo las mismas, junto con el array de posiciones de las balizas, y la identificación de la baliza de referencia que en este caso era la baliza 1 (baliza central). La posición estimada y devuelta por el algoritmo Gauss-Newton fue: x=-0.0869 e y=0.4756. Como podemos observar es exactamente la misma posición de la que se había partido, esto es así debido a que las diferencias de distancias se han obtenido a partir de esa posición y en ningún momento se ha introducido ruido ni errores al sistema, con lo que al considerar un sistema ideal la respuesta debe ser exactamente la misma.

Seguidamente, se ha introducido al algoritmo de Gauss-Newton el array de diferencias de distancias TDOA = [0.0549, -0.0374, 0.0267], se ha suprimido la diferencia de distancias con la baliza 5, generando una situación en la que esta baliza

no emite información. Se le ha introducido también el array de posiciones de las balizas, sin la baliza 5, ya que debe ser eliminada tanto del array de TDOA como del array de posiciones de las balizas. Una vez enviada esta información al algoritmo, éste al ser dinámico se adapta a esta nueva situación y estima las posiciones: x=-0.869 e y=0.4756. Se puede observar nuevamente como las posiciones estimadas no presentan ningún error debido a que no se ha introducido ruido, además se observa como el sistema es capaz de estimar la posición con 4 balizas únicamente, esto como se ha comentado anteriormente se debe a que el sistema se ha diseñado con 5 balizas con la idea de tener una baliza como sistema redundante.

Una vez que se comprueba el correcto funcionamiento del algoritmo en Matlab, se procede a implementarlo en Android. Para comprobar el correcto funcionamiento del mismo se genera una señal ideal la cual guarda una posición conocida y posteriormente la misma señal, pero eliminando la información de una baliza, así se puede simular la situación en la que falla una baliza. Se ha optado por hacer así la comprobación del funcionamiento del algoritmo en Android debido a que va a funcionar en tiempo real, con lo que debe ser capaz de identificar si una baliza no entrega información e identificar cual es.

Se ha generado como señal de referencia la mostrada en la Figura 30.

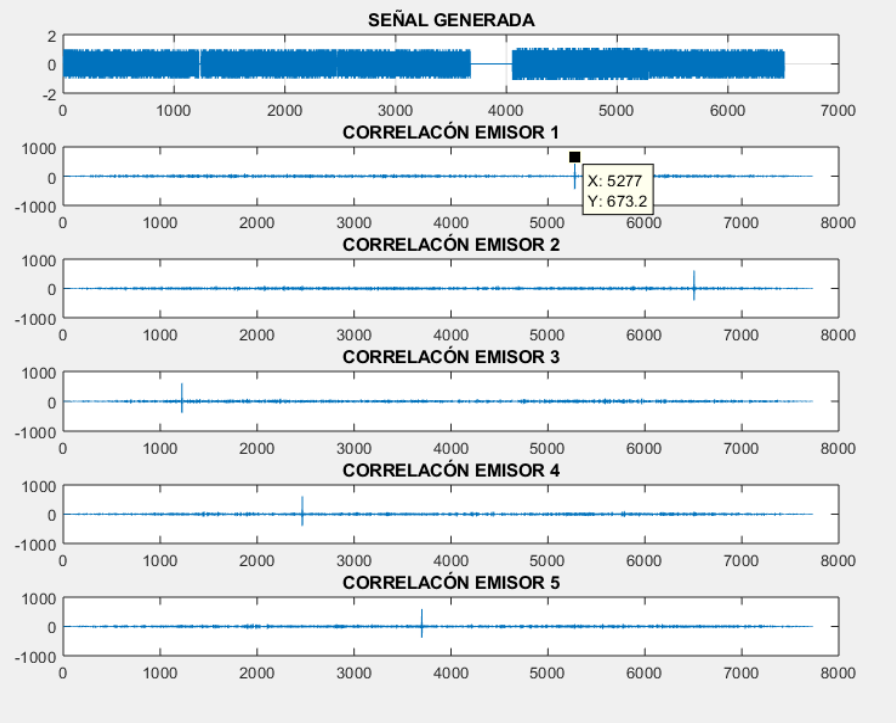


Figura 30: Señal generada para Android.

Como se puede observar en la Figura 30 se ha optado por una señal en la que el buffer se encuentra descolocado con la baliza de referencia (emisor 1) que se encuentra como el 4º pico recibido en el buffer, de esta manera a la vez que se comprueba el correcto funcionamiento del algoritmo dinámico de Gauss-Newton, también se sigue observando como el sistema se adapta a diferentes situaciones de baliza de referencia como se ha compentado en el apartado 4.5.1.

La diferencia de distancias obtenida con esta señal de referencia es TDOA = [0.0240 -0.0054 0.0583 0.0871], y la posición a la que van asociadas a estimar corresponde a x = 0.1048 e y = 0.2827 (obtenida en Matlab), con lo que estos valores serán los que se deben obtener en Android cuando se introduzcan esta señal eliminando una baliza (en este caso se ha optado por la baliza 4), quedando la señal generada según la Figura 31 y los TDOA = [0.0240 -0.0054 0.0871].

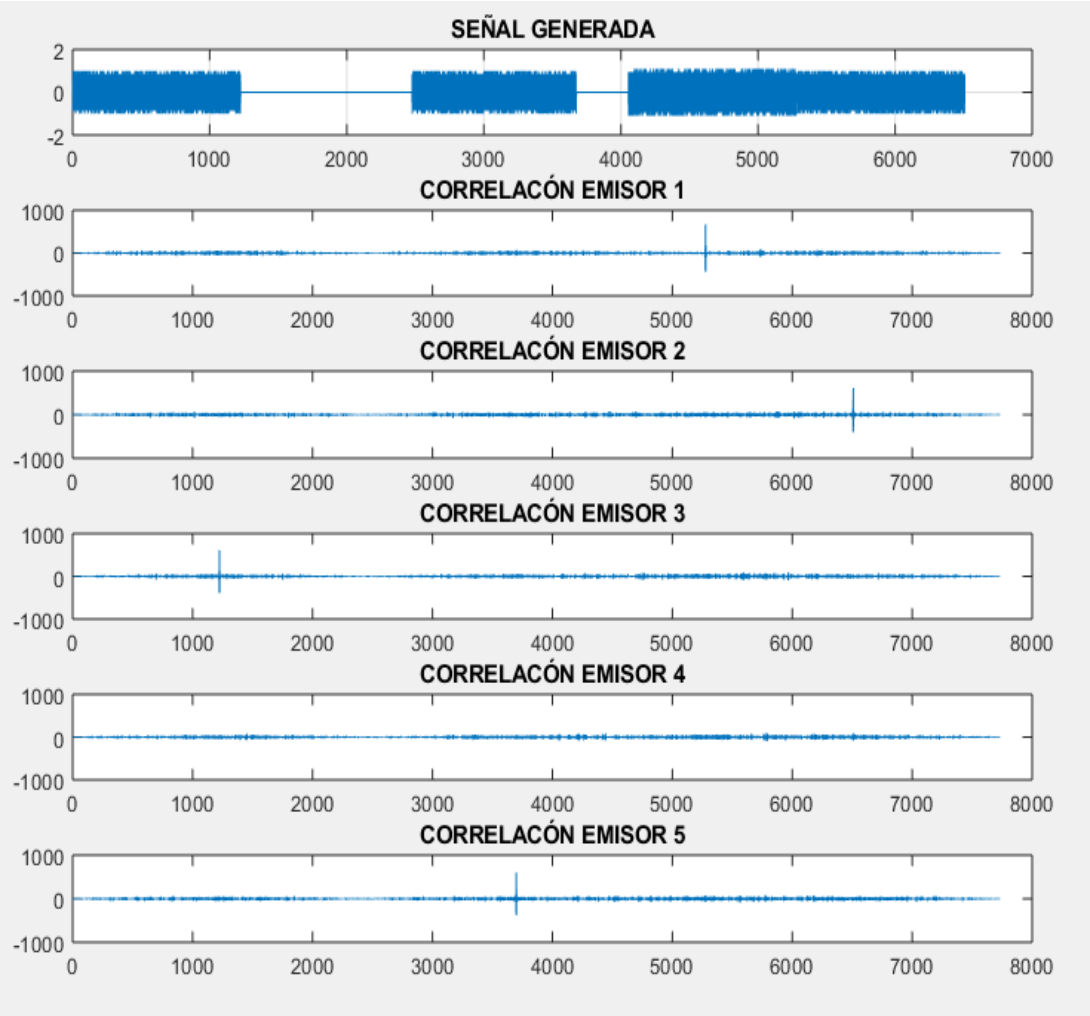


Figura 31: Señal modificada para Android.

Una vez introducida la señal de la Figura 31 en Android, y procesada, en la Figura 32 se puede observar el resultado del procesado de la señal introducida en Android. Se puede opreciar que al realizar las correlaciones con cada una de las balizas, para obtener el pico de mayor energía y su posición, la medida 3 (correspondiente al emisor 4, que ha sido eliminado) se descarta por no superar el umbral de cobertura (fijado a un 20% del pico de mayor energía). Esto se debe a que como se ha comentado anteriormente esa baliza ha sido eliminada de la señal que se ha introducido. También se observa como el sistema identifica cual es la baliza de referencia, (la baliza 0, correspondiente al emisor 1) y como calcula las TDOA (diferencias de distancias). El *array* correspondiente a las TDOA también es usado para identificar la baliza de referencia, que es guardada en la última posición siempre, para así al ejecutar los algoritmos de fusión sensorial o de Gauss-Newton, se sepa cuál ha sido la baliza de mayor energía, y con ello cual se ha tomado como baliza de referencia para obtener los TDOA. También se hace uso del *arrya* de TDOAs para guardar, en el caso de que se haya eliminado alguna de las medidas, en la penúltima posición del *array* la baliza asociada a la medida que se ha eliminado. De esta manera, haciendo uso del *array* TDOA, se tiene información de las TDOA. Además de la baliza de referencia y además de, en el caso de la eliminación de alguna medida, de cual de ella se trata.

```
Máximos y posiciones de las correlaciones antes del enventanado:
Máximo de baliza 0: x = 5275, y = 673.2
Máximo de baliza 1: x = 6506, y = 611.9998
Máximo de baliza 2: x = 1222, y = 611.9998
La medida 3 se descarta porque no supera el umbral de correlación
Máximo de baliza 4: x = 3697, y = 598.9999
La baliza de referencia es: 0, con x=5275,y=673.2
Resultado de las TDOA:
TDOA[0]=0.02401
TDOA[1]=-0.00343
TDOA[2]=0.08918
TDOA[3]=3.0
TDOA[4]=0.0
Baliza Eliminada->3
Matriz ordenada con baliza_eliminada==3
POSICIONANDO...
POSICION CALCULADA = 0.104566835, 0.28272355
```

Figura 32: Resultados Gauss-Newton con 4 balizas en Android.

Una vez que se tiene las TDOA, se accede al algoritmo de Gauss-Newton, donde primeramente se reordenarán la matriz donde se tiene las posiciones de las balizas teniendo en cuenta que baliza se ha considerado de referencia y si alguna de ellas se ha eliminado, y posteriormente se procede a la estimación de la posición. En este caso la posición estimada ha sido x = 0.104566835 e y = 0.28272355, posición a partir de la cual se había generado la señal introducida en Android, con lo que se puede confirmar el correcto funcionamiento del sistema y particularmente el correcto funcionamiento del algoritmo de Gauss-Newton haciendo uso de 4 balizas únicamente.

4.5.4 Implementación EKF de manera dinámica para el posicionamiento con 4 balizas

Al igual que para el algoritmo de Gauss-Newton, los algoritmos de fusión sensorial se deben modificar, para aprovechar la capacidad del sistema de detectar si una de las balizas está fallando, y en caso afirmativo, la capacidad de eliminarla del set de medidas, ya que como se ha comentado anteriormente, ante la ausencia de una baliza, se sigue pudiendo resolver la estimación de la posición en 2D.

En este apartado se van a explicar los pasos que se ha seguido para modificar el algoritmo de fusión sensorial (el EKF aplicado a la fusión sensorial), primero en Matlab, y posteriormente en Android, para que este algoritmo se capaz de trabajar tanto con 5 balizas (es decir 4 TDOAs) como con 4 balizas (es decir 3 TDOAs).

Como se ha explicado en el apartado 2.6 las ecuaciones que se han implementado para particularizar el EKF para la fusión sensorial son (2.6.1), (2.6.2), (2.6.3), (2.6.4), (2.6.5), (2.6.6), (2.6.7), (2.6.8) y (2.6.9). Observando todas estas ecuaciones, se puede determinar que las ecuaciones y con ello las matrices que se ven afectadas al trabajar con 4 balizas, en vez de con 5 balizas son: la matriz $\boldsymbol{h}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ perteneciente a la ecuación (2.6.3), la matriz $\mathbf{H}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ perteneciente a la ecuación (2.6.4) y la matriz $\boldsymbol{Z}_{\boldsymbol{k}}$ perteneciente a la ecuación (2.6.8). Todas estas matrices dependen directamente de las TDOAs, con lo que ante la ausencia de una baliza sus dimensiones se verían modificadas de la siguiente manera:

- $\boldsymbol{h}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ con dimensiones de $\boldsymbol{4x1}$ pasa a tener dimensiones de $\boldsymbol{3x1}$ debido a que en este caso solo debe realizar la estimación a priori de 3 TDOAs en vez de 4 TDOAs.

- $\mathbf{H}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ al ser la derivada de $\boldsymbol{h}(\widehat{\boldsymbol{X}}_{\boldsymbol{k}}^{-})$ con respecto a cada variable del vector de estados, pasa de tener las dimensiones $\boldsymbol{4x3}$ a $\boldsymbol{3x3}$, ya que en esta situación se tiene un TDOA menos, pero el vector de estados sigue teniendo 3 variables (x, y, θ).

- $\boldsymbol{Z}_{\boldsymbol{k}}$ es directamente los TDOA medidos, al pasar de 4 TDOAs a 3 TDOAs, las dimensiones de $\boldsymbol{Z}_{\boldsymbol{k}}$ pasan de $\boldsymbol{4x1}$ a $\boldsymbol{3x1}$.

- Además de estas ecuaciones que se observan a primera vista, hay que tener cuidado en modificar las dimensiones de la matriz **R** de la ecuación (2.6.7), pasan de **4x4** a **3x3** ya que es necesario para que cuadren las dimensiones de la ecuación (2.4.6).

Para la comprobación del correcto funcionamiento del algoritmo, se ha propuesto estimar la misma ruta que en el apartado 4.4, pero en este caso simulando la situación en la que la baliza 5 no funciona. Se observará la estimación de las rutas, y los errores obtenidos.

Como se puede observar en la Figura 33 la ruta estimada es exactamente igual a la generada, y la media del error obtenida ha sido de $7.3537*10^{-16}$ m con una desviación estándar de $6.4841*10^{-16}$ m, valores que se pueden considerar cero, al igual que en el apartado 4.4, con lo que se puede confirmar el correcto funcionamiento del algoritmo usando únicamente 4 balizas.

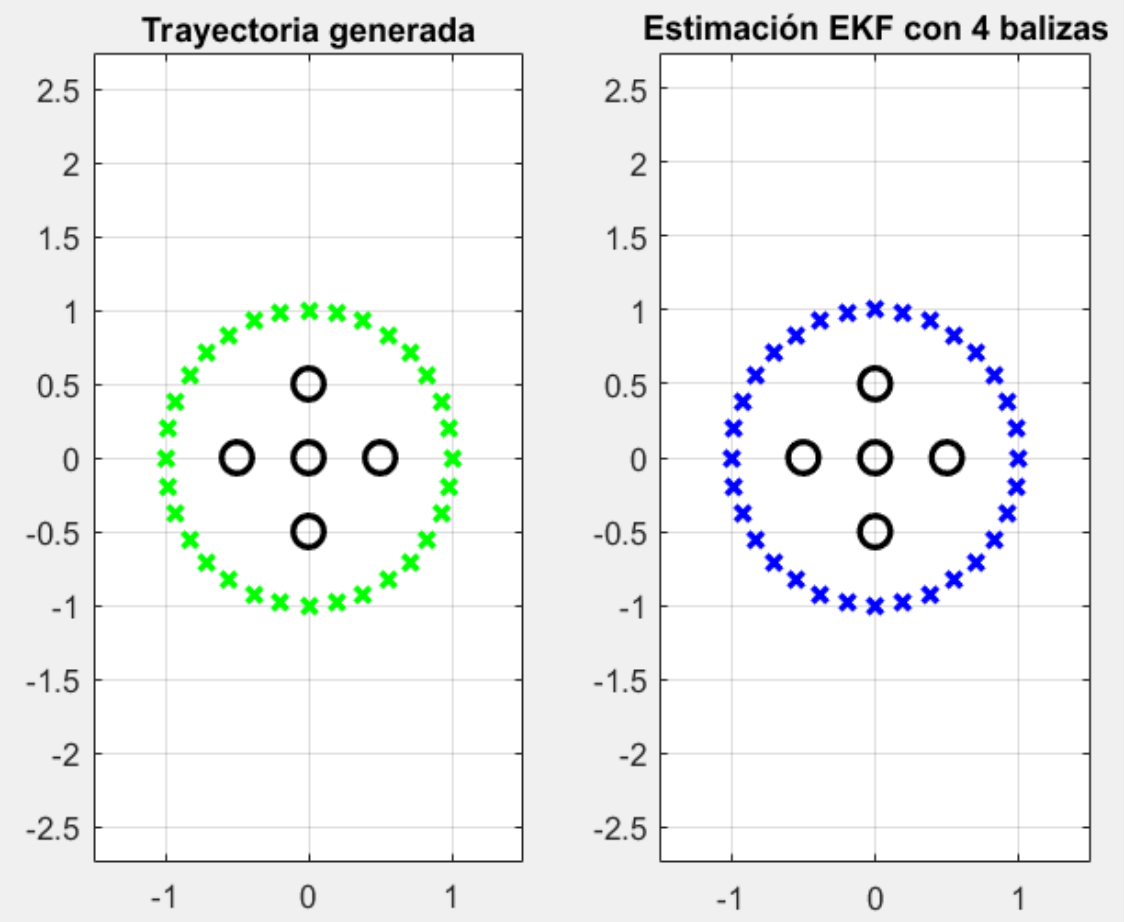


Figura 33: Estimación ruta con EKF en Matlab con 4 balizas.

Una vez comprobado el correcto funcionamiento del algoritmo en Matlab, se procede a realizar la verificación del mismo en Android. Para ello se ha hecho uso de la misma ruta, y se ha simulado la eliminación de la baliza cinco. Los resultados de la trayectoria estimada con el algoritmo de fusión sensorial haciendo uso se muestran en la Figura 34, como se observa el resultado de la trayectoria estimada es el mismo al de la trayectoria generada, lo que se esperaba para verificar el correcto funcionamiento del algoritmo de fusión sensorial dinámico. Además, observando la media de los errores, se obtiene prácticamente cero ($7.35*10^{-16}$) con una desviación del mismo prácticamente cero también ($6.48*10^{-16}$).

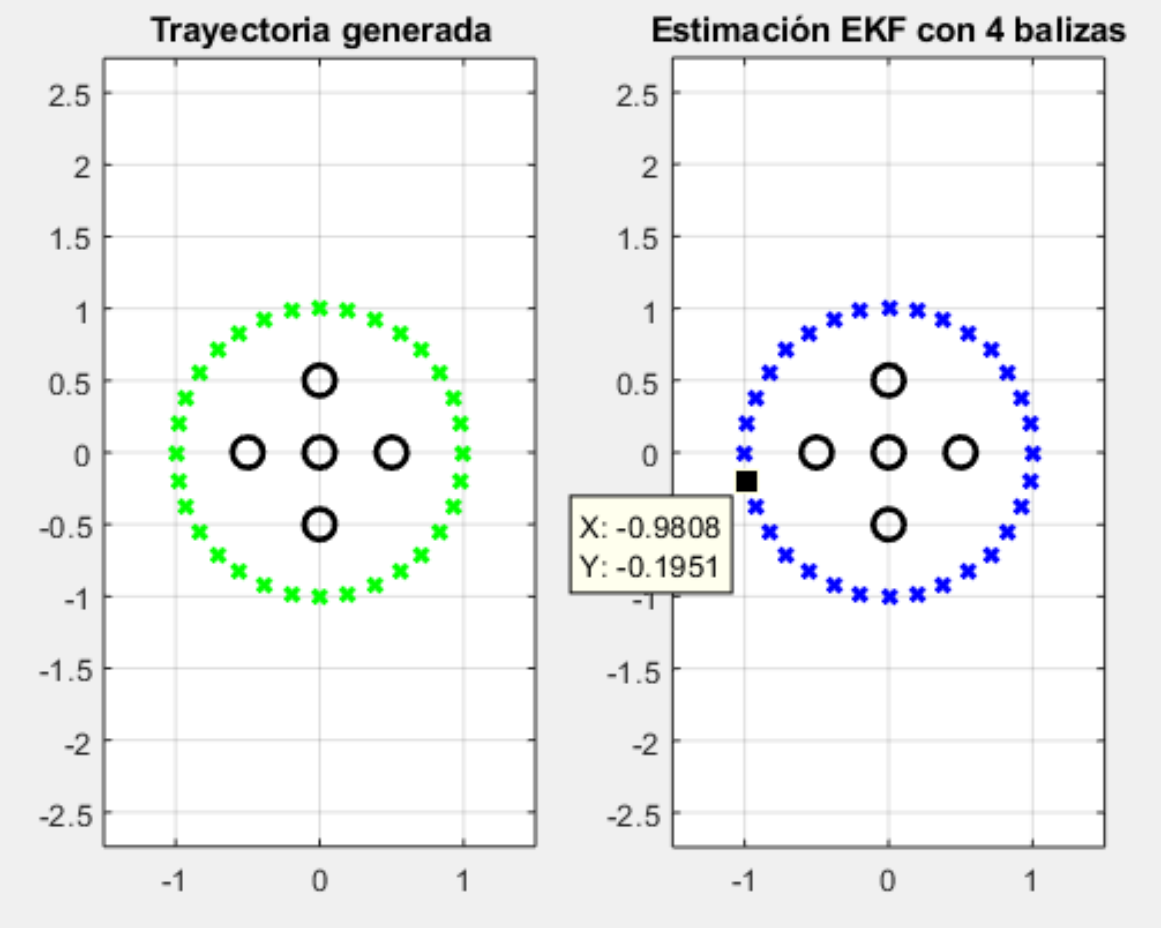


Figura 34: Estimación ruta EKF en Android con 4 balizas.

En la Figura 35 se muestra en el monitor de Android, la estimación de la última posición de la trayectoria estimada, la cual ha sido señalada en la Figura 34. En esta figura se puede observar los valores con los que se han actualizado cada matriz, y las dimensiones, las cuales han variado con respecto a las iniciales como se ha comentado anteriormente en este apartado, y coinciden con las comentadas.

```
X_f:
-0.9807852804032327
-0.19509032201613027
1.865320638068942
Pf:
1.5648274075309937E-4    2.8640869236075042E-5    -1.283575251929069E-4
2.864086923607507E-5    2.8989693567740317E-4    -1.4794808408410747E-4
-1.283575251929069E-4    -1.4794808408410737E-4    0.0024453474484535855
Af:
1.0    0.0    -0.18759311034896117
0.0    1.0    -0.05690574789194407
0.0    0.0    1.0
Hf:
-0.15102385917893268    0.007361757070657254    0.0
0.014634423013525522    -0.1965299402687007    0.0
0.19314860642595713    -0.005580559788796638    0.0
Zf:
0.242485628788369
0.090399046595048
-0.156251984100033
h_X_f:
0.24248562878836877
0.09039904659504838
-0.15625198410003405
Qf:
4.0E-4    0.0    0.0
0.0    4.0E-4    0.0
0.0    0.0    4.0E-4
Rf:
2.5E-5    1.25E-5    1.25E-5
1.25E-5    2.5E-5    1.25E-5
1.25E-5    1.25E-5    2.5E-5
FUSION HECHAAA-> x=-0.9807852804032305 y=-0.1950903220161276 O=1.8653206380689396, Sl= 0.196034280659121
```

Figura 35: Monito Android estimación posición EKF.

4.6 Programación de la Visualización de la posición

La visualización en pantalla de la posición se ha considerado una parte muy importante en este trabajo, ya que es el puente que enlaza toda la algoritmia necesaria para la estimación de la posición de un usuario, con la representación de la mima para que el usuario pueda hacer uso de ella, por ello se ha pretendido realizar una visualización lo más útil para el usuario.

En trabajos anteriores [Pérez, 2016] la representación de la posición se realizaba en un mapa XLM, lo que empeoraba mucho el rendimiento de la aplicación. Posteriormente en los trabajos [Cervigón, 2017] y [Díaz, 2017] el mapa XLM se reemplazó por el mapa de Google Maps, lo cual mejoró el rendimiento. En estas versiones la representación de la posición se realizaba una vez finalizada la prueba.

En el trabajo aquí presente se ha optado por hacer uso de la API de Google maps, ya que frente a trabajos que se hace uso de un mapa XML, el rendimiento y las posibilidades mejoran notablemente. Además, este trabajo también ha conseguido realizar la representación en tiempo real, así el usuario consigue tener un posicionamiento instantáneo y una visualización del mismo en su dispositivo portable en el momento. Esto consigue que la interacción entre el usuario y el dispositivo sea constante, y añade un gran atractivo a la aplicación aquí presenta.

Google Maps a partir del 2013 quiso dar un paso más ofertando la posibilidad de observar los mapas interiores de algunos edificios como universidades, hospitales, aeropuertos, etc. Concretamente en el entorno de trabajo donde se ha desarrollado este proyecto, el Edificio Politécnico de la Universidad de Alcalá, cuenta con su mapa de interiores ofrecido por Google Maps como podemos observar en la Figura 36, en la cual se observa una parte del interior de la planta tercera del Edificio Politécnico de la Universidad de Alcalá. Además, se puede observar en la esquina inferior derecha, que el mapa ofrece las plantas de las que dispone el edificio, y eligiendo cualquiera de ellas, se puede observar.

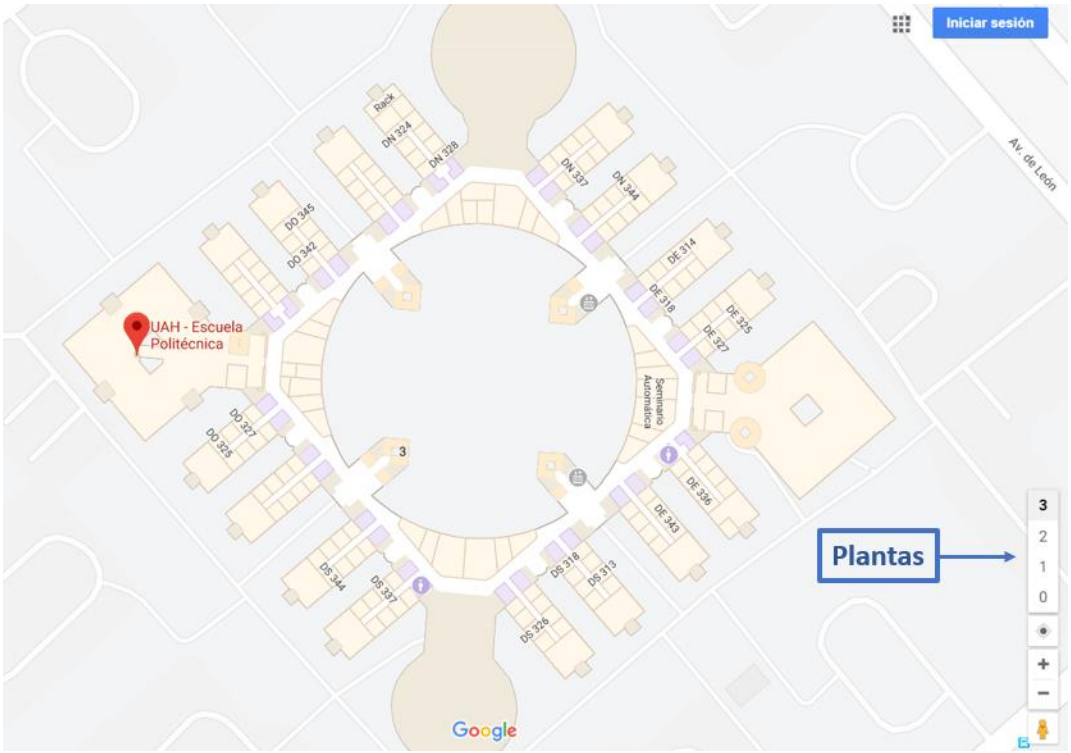


Figura 36: Mapa interior Escuela Politécnica UAH.

Además, para el entorno de desarrollo Android Studio, a la hora de programar una aplicación, se puede hacer uso de la API de Google Maps, lo cual permitirá que la aplicación sea capaz de interactuar con los mapas de Google, incluido los de interiores, ofreciendo una gran variedad de posibilidades.

Para poder insertar mapas de Google Maps en la aplicación lo primero que se debe saber, es que es necesario tener las *Google Play Services* activadas. Para ello en el entorno de programación Android Studio se debe acceder al SDK Manager y comprobar que está instalada la *Google Play Services* además es recomendable tenerla actualizada a la última versión, en la podemos observar la *Google Play Services* instalada y actualizada a la última versión (48).

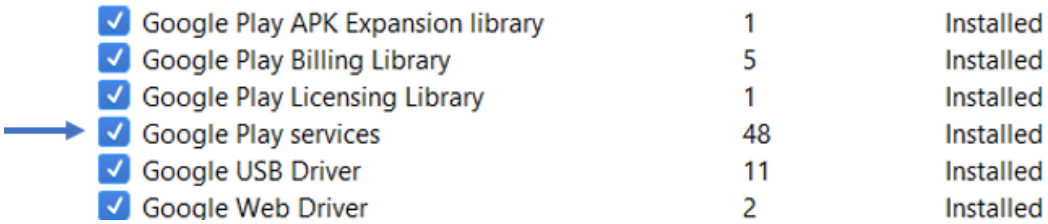


Figura 37: *Google Play Servicies.*

Una vez que se tiene descargadas las *Google Play Services*, se dispone a implementar el mapa de Google en la aplicación. Para ello en la actividad principal se han importado las librerías *com.google.android.gms.maps* las cuales contienen las clase API de Android de Google Maps, y se importan automáticamente al implementar en nuestra actividad principal el método *OnMapReadyCallback* el cual posteriormente se usará.

Además de una de las cosas más importantes es la obtención de la clave de acceso de la API de Google Maps, es decir la API *Key*. Al hacer uso de la API de Google Maps, se creará el archivo *Google_maps_api.xml*:

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.
    To get one, follow this link, follow the directions and press "Create" at
the end:
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backe
nd&keyType=CLIENT_SIDE_ANDROID&r=1F:57:EB:B1:C9:0C:4B:4C:CD:55:6F:14:59:C1:9C:
87:EB:16:0E:0A%3Bcom.us.locate.myapplication
    You can also add your credentials to an existing key, using this line:
1F:57:EB:B1:C9:0C:4B:4C:CD:55:6F:14:59:C1:9C:87:EB:16:0E:0A;com.us.locate.myap
plication
    Alternatively, follow the directions here:
   https://developers.google.com/maps/documentation/android/start#get-key
    Once you have your key (it starts with "AIza"), replace the
"google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">
        YOUR_KEY_HERE
    </string>
</resources>
```

Código 4.6.1: Obtención de la API Key.

Donde a través de la dirección que nos proporciona en las primeras líneas se deberá obtener una API *Key* que deberá copiarse e insertarse donde pone "YOUR_KEY_HERE", gracias a esa clave la aplicación podrá hacer uso de los mapas de Google.

Una vez obtenida la API *Key* en este trabajo se ha implementado un fragmento (*fragment*) en la actividad principal donde se implementará el mapa de Google:

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_weight="0.01"
    android:layout_width="match_parent"
    android:layout_height="388dp"
    tools:context="com.us.locate.myapplication.MapsActivity"/>
```

Código 4.6.2: Implementación fragmento.

El cual tendrá que ser cargado desde la actividad principal:

```
SupportMapFragment smapMapFragment;
//Cargamos el MapFragment
smapMapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
smapMapFragment.getMapAsync(this);//remember getMap() is deprecated!
```

Código 4.6.3: Carga del fragmento.

Gracias al uso de este fragmento en la pantalla principal, se dispone del mapa en la actividad principal, es decir no hace falta cargar una nueva actividad donde se carguen los mapas.

Una vez que se tiene configurado el uso de los mapas de Google Maps, la API de Google Maps dará la opción de hacer uso e interactuar con el mapa a través las bibliotecas que ofrece. Antes de explicar las bibliotecas de las que se hace uso, se debe tener en cuenta que al haber implementado el método *OnMapReadyCallback* la primera vez que se arranque la aplicación y con ello la actividad principal, se accederá automáticamente a la función *public void onMapReady* donde lo primero que se tendrá hacer es guardar en una variable la variable que se recibe al ejecutarse dicha función:

```
//Definición de la variable GoogleMap
private GoogleMap mMap;
//INICIO MAPS
@Override
public void onMapReady(GoogleMap googleMap) {

    mMap = googleMap;
    .........
```

Código 4.6.4: Definición variables *GoogleMap* y método onMapReady.

Posteriormente gracias a esa variable, se podrá acceder a las clases que ofrece la API de Google Maps que serán necesarias para interactuar con el mapa. En *onMapReady* este trabajo también ha aprovechado a configurar algunos identificadores que serán fijos en el mapa, como por ejemplo unos marcadores que representarán la ubicación de los U-LPS en el mapa, además de detectar el número de plantas del edificio que detecta, y seleccionar la planta inicial que se desea visualizar en el arranque de la app.

A continuación, se exponen algunas de las clases que ofrece la API de Google Maps y de las cuales se ha hecho uso en este trabajo.

Se tiene que tener en cuenta que la posición obtenida a través de los diferentes algoritmos de estimación, son unas coordenadas que se encuentran referenciadas a un mapa XML interno del edificio definido por el grupo de investigación a través de los planos AutoCAD de la construcción del edificio. Gracias a un trabajo previo desarrollado en el grupo por un compañero, Sergio Matas, estas coordenadas son transformadas a través de una función a coordenadas UTM (sistema de coordenadas universal transversal de Mercator). Una vez que se dispone de las coordenadas UTM, se convierten al formato latitud, longitud a través de la librería Jcoord gracias a la clase *utm1.toLatLng()*. Una vez que se dispone de las coordenadas en latitud-longitud, estas se pueden representar en el mapa de Google Maps a través de la clase *addMarker*:

```
marker1 = mMap.addMarker(new
MarkerOptions().visible(True).position(posicion).title("US").icon(BitmapDescri
ptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));
```

Código 4.6.5: Sentencia para añadir un marcador en el mapa.

Esta clase además de añadir un marcador en el mapa en la posición indicada, ofrece una serie de opciones como seleccionar si se quiere que el marcador sea visible, la posición donde se desea posicionar el marcador, si se le desea añadir un título, e incluso hacer uso de un marcador y elegir su color, o elegir otro icono generado por el usuario.

Una vez que se ha posicionado el marcador, se debe indicar al mapa se posicione la cámara en el área donde se encuentra el marcador (*moveCamera*), y además con el nivel de zoom mapa deseada (*animateCamera*):

```
mMap.moveCamera(CameraUpdateFactory.newLatLng(posicion));
mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(posicion, 21));
```

Código 4.6.6: Sentencia para interactuar con la posición y el zoom de la cámara del mapa.

Gracias a la clase *IndoorLevel* y *IndoorBuilding*, se permite detectar (en el caso de que exista) un edificio en las coordenadas a las que está enfocando la cámara, y en el caso de que se detecte, se puede acceder al número de plantas que este tiene, y seleccionar cual se quiere que se visualice:

```
//Detección del edificio
IndoorBuilding building = mMap.getFocusedBuilding();
if (building == null) {
    System.out.println("EDIFICIO NULL");
}else {
    //Si se detecta edificio, se accede al número de plantas que tiene y se
activa una
    System.out.println("Planta Seleccionada: "+building.getLevels().get(0));
    IndoorLevel indoorlevel;
    indoorlevel =building.getLevels().get(0);
    indoorlevel.activate();
}
```

Código 4.6.7: Detección de edificio y selección de planta visible.

Por último, se ha hecho uso de las clases *addPolyline* y *addCircle* para realizar la representación de los U-LPS en el mapa, además de hacer uso de *colorToHSV* y *setFillColor* para definir los colores del circulo:

```
mMap.addPolyline(polyline1ULPSF);
mMap.addPolyline(polyline2ULPSF);
circleF = mMap.addCircle(circleOptionsF);
float[] prevHSV = new float[3];
Color.colorToHSV(circleF.getFillColor(), prevHSV);
circleF.setFillColor(Color.HSVToColor(25, prevHSV));
```

Código 4.6.8: Representación de los U-LPS en el mapa.

4.7 Diagrama de flujo de la aplicación

La Figura 38 muestra el flujograma de la aplicación (app). Nada más abrirla lo primero que se carga es la actividad inicial, esta actividad es una pantalla de arranque a la aplicación que sirve como presentación de la app que se va a abrir. En paralelo a esta actividad se arranca el hilo "*Template*" que se encarga de cargar las constantes, las variables globales y las plantillas a utilizar en el procesado, al ejecutarse en paralelo por detrás de la actividad de presentación, la carga de estos archivos no tiene ninguna repercusión visual para el usuario.

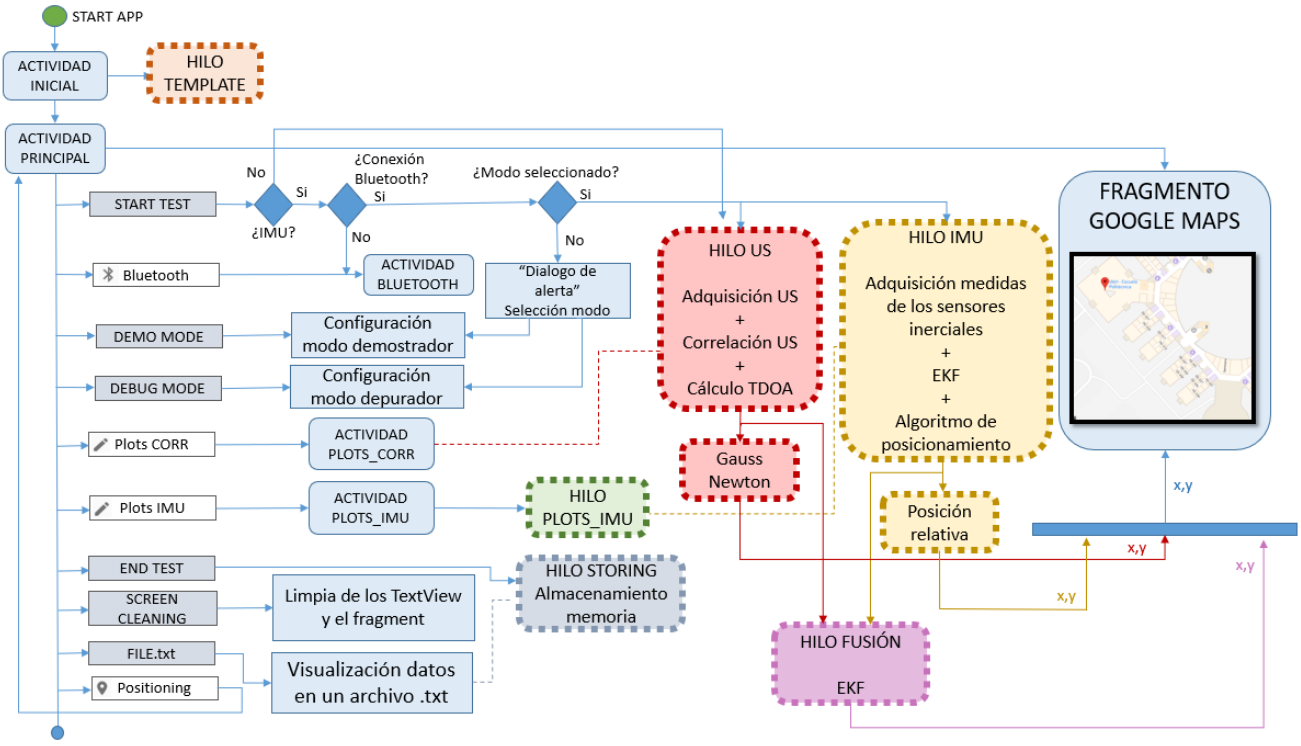


Figura 38: Flujograma de la aplicación.

Una vez ejecutada la actividad inicial y cargados los archivos, arranca la actividad principal. Además de la actividad principal, la aplicación consta de otras actividades, con lo que, para poder transitar entre ellas, se ha implementado un menú deslizante ("Navigation Drawer").

En el menú deslizante se encuentra la opción Bluetooth a partir de la cual se accede a la actividad "BLUETOOTH". En esta actividad se permite al usuario seleccionar si va a hacer uso de una unidad IMU para la navegación o no, y en el caso de que quiera hacer uso se realiza la conexión entre ésta y el dispositivo portable.
A través del menú deslizante también se accede a la actividad "PLOTS_CORR", en esta actividad el usuario puede visualizar las correlaciones realizadas con cada uno de los cinco emisores del U-LPS con cobertura en ese momento. Otra de las actividades a las que se puede acceder es "PLOTS_IMU, mediante la cual y haciendo uso del hilo "Plots_IMU" se realiza una representación de los valores del Roll, Pitch y Yaw generados por la IMU en tiempo real. Por último, en el menú deslizante se encuentra la opción Positioning gracias a la cual se podrá retornar desde cualquier actividad mencionada con anterioridad, a la actividad principal.

El mayor nivel de procesamiento se lleva a cabo en la actividad principal y para gestionarlo se han implementado una serie de opciones y botones. La app tiene la posibilidad de ejecutarse en dos modos: modo depurador, enfocado a observar el comportamiento de los diferentes algoritmos de posicionamiento implementados en el software; y un modo demostrador, orientado al usuario final donde solo tendrá acceso a la posición instantánea, observando cómo se mueve un marcador en la pantalla a la vez que se mueve el mismo. Los diferentes modos de uso se podrán seleccionar en la actividad principal a través de los interruptores ("switch") "DEMO MODE" y "DEBUG MODE".

En la actividad principal también se encuentra el botón "START TEST" con el cual se indicará la posibilidad de iniciar el posicionamiento. Para ello, primero se debe seleccionar si se quiere hacer uso de la IMU para la navegación, o, por el contrario, no se desea hacer uso de ella. Si no se desea hacer uso de la unidad inercial, se activará automáticamente el modo de ejecución demostrador y la estimación de posición se realizará únicamente mediante ultrasonidos, ejecutándose en paralelo el hilo "US" únicamente. Este hilo se encarga del procesamiento de las señales ultrasónicas junto a la ejecución del algoritmo de Gauss Newton. Con lo que en esta configuración solo se obtendrá el posicionamiento en las zonas de cobertura U-LPS en modo demostrador a través del algoritmo Gauss Newton. Por otro lado, en el caso de que se quiera hacer uso de la IMU, y se encuentre conectada (en caso contrario la app hará acceder a la actividad "BLUETOOTH" al usuario automáticamente), se comprueba si se ha seleccionado un modo de ejecución de los mencionamos anteriormente, en caso negativo aparecerá en pantalla un dialogo de alerta ("AlertDialog") a través del cual se permitirá seleccionar el modo de ejecución deseado. Una vez realizadas estas configuraciones, se arrancan dos hilos para que se ejecuten en paralelo y por detrás del hilo principal, el hilo "US" que, ya se ha comentado anteriormente, y el hilo "IMU" que se encarga de realizar el procesamiento de las medidas de los sensores inerciales. Además, en paralelo, cuando se requiera el cálculo de la posición a través de la fusión sensorial se hará uso de una función que ejecutará los algoritmos de fusión sensorial en el hilo principal. El resultado del posicionamiento será representado en un fragmento (Fragment) generado en la actividad principal. En este se hará uso de la interfaz de

programación de aplicaciones (API) Google Maps, con lo que se optimiza el tiempo de representación y se permite realizar la representación de la posición en tiempo real en el mapa el Google.

Por último, en la actividad principal aparecen una serie de botones los cuales permitirán limpiar la interfaz del usuario ("SCREEN CLEANING"), mostrar los datos del proceso en un archivo .txt ("FILE.txt") y finalizar la prueba de posicionamiento ("END TEST"). Al finalizar, se almacenarán los datos de la prueba en memoria en segundo plano a través del hilo "Storing".

Como se puede observar los procesos de mayor peso se realizan en segundo plano gracias a los hilos "US", "IMU" y "Plots_imu", asimismo, los hilos "Template" y "Storing" permiten ejecutar en segundo plano otros procesos que se desea que se realicen al margen de la visualización del usuario. Además, al implementar en la actividad principal la Google Maps API en un fragmento, permite representar en tiempo real la posición del usuario en el mapa de Google dentro de la pantalla principal.

CAPÍTULO 5: PRUEBAS EXPERIMENTALES

En este apartado se van a mostrar los resultados obtenidos de la aplicación desarrollada. Las pruebas se han realizado en la tercera planta de la Escuela Politécnica Superior de la Universidad de Alcalá. En la Figura 39 se puede observar la trayectoria realizada, como se observa se recorren dos pasillos, el primer pasillo desde el punto 1 al punto 2, y el segundo pasillo del punto 2 al punto 3. La intersección de ambos pasillos (punto 2) es de 90º. Se ha decidido que apareciese en la trayectoria un giro de 90º debido a que estos giros suponen puntos críticos en la estimación de trayectoria con sensores inerciales, además gracias a la trayectoria seleccionada, en una misma prueba se puede observar la navegación en trayectorias rectas y giros. La trayectoria abarca 37m y se han utilizado tres U-LPSs, ubicados en el techo a una altura aproximada de 3m. El primero se ha situado en el punto de partida, para así iniciar la trayectoria desde una posición inicial precisa. El segundo se ha ubicado en la intersección de los dos pasillos que forman un giro de 90º ya que como se ha comentado esto supone un punto crítico en la trayectoria. El último se ha ubicado en el punto de finalización de la prueba.

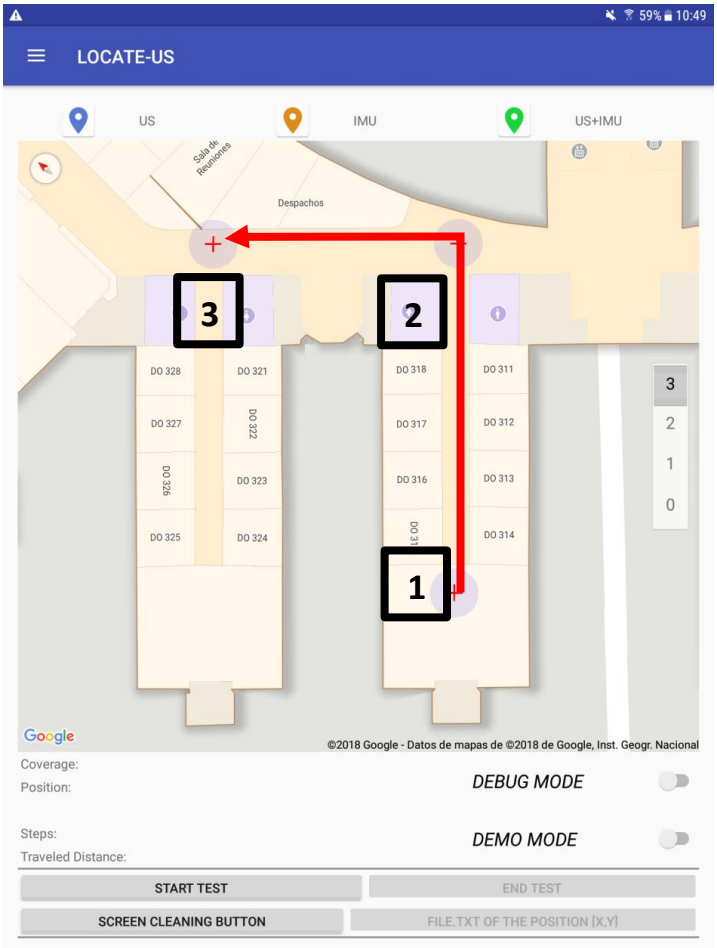


Figura 39: Trayectoria real realizada.

Para observar el entorno donde se han realizado las pruebas, se muestra la Figura 40, donde se observa el punto de intersección de 90º de los dos pasillos y el segundo pasillo. Se pueden observar los dos U-LPSs. El ubicado en la intersección, y el ubicado al final de la trayectoria, y se puede observar al usuario portando la IMU en la pierna derecha, la Tablet sostenida con las manos, como esta última lleva enganchada en el puerto USB la tarjeta de adquisición de US.

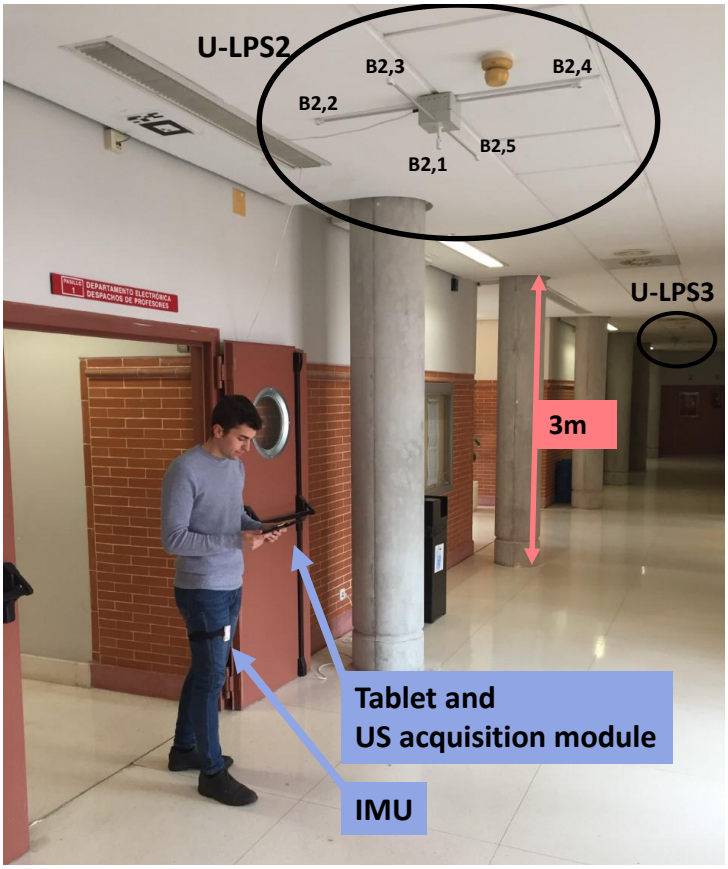


Figura 40: Entorno de pruebas.

Los resultados de la prueba se muestran en la Figura 41. Los marcadores azules representan las posiciones obtenidas a través de las US gracias al algoritmo de Gauss Newton. Los marcadores naranjas representan las posiciones relativas obtenidas a través de la IMU. Por último, los marcadores verdes representan las posiciones obtenidas gracias a la fusión entre la IMU y los US. Dentro de las zonas de cobertura se obtiene la posición gracias a la fusión sensorial implementada a través del EKF, y en las zonas que no existe cobertura, la posición se obtiene gracias a la IMU. Cabe destacar que en las posiciones representadas a través de los marcadores verdes se aplican las correcciones de los errores a través de la fusión sensorial.

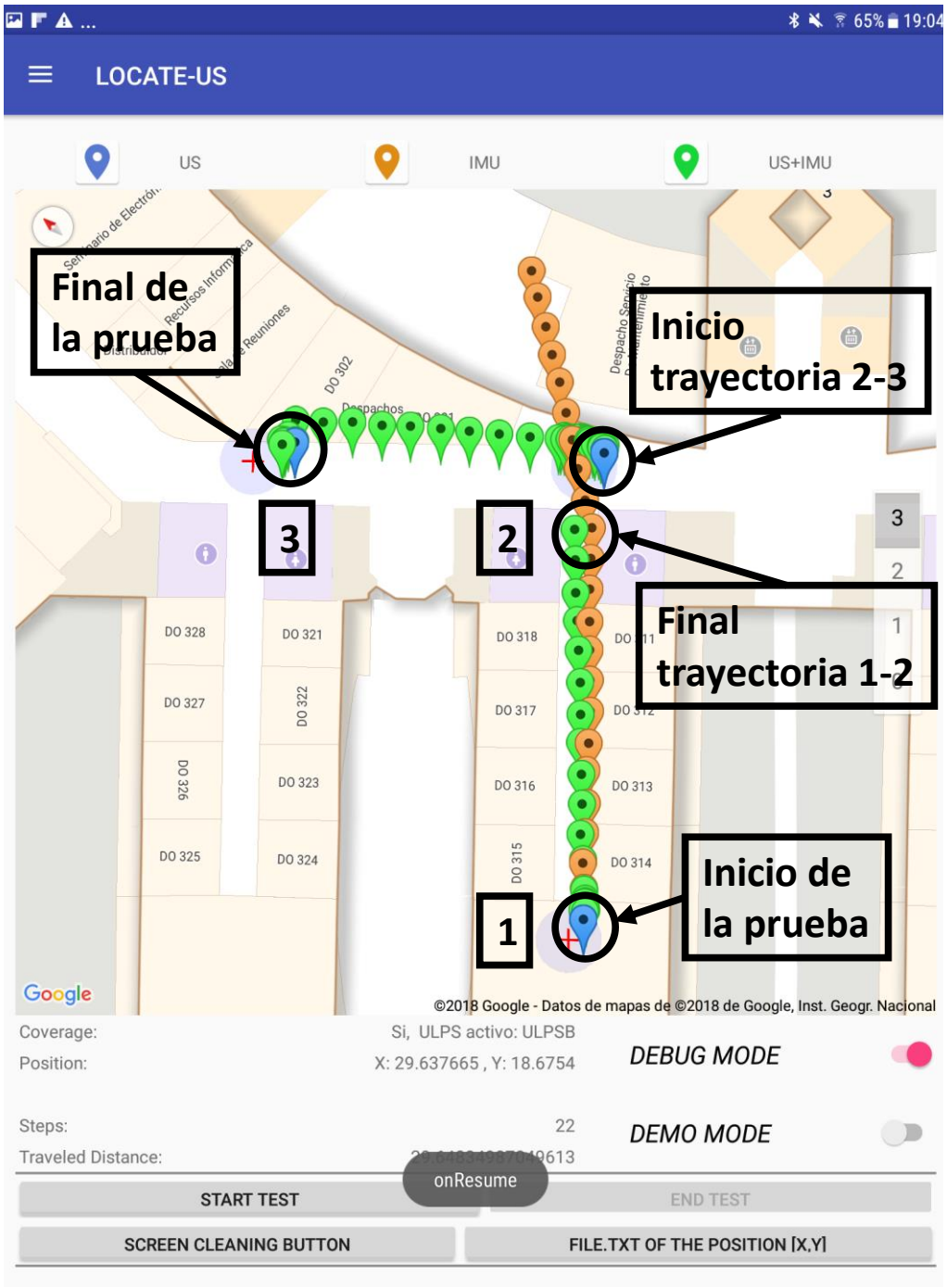


Figura 41: Trayectoria estimada.

La trayectoria se inicia en el punto 1 de la Figura 41, donde la primera posición se obtiene a través de las US y el algoritmo de Gauss Newton (marcador azul). A continuación, mientras se tiene cobertura U-LPS, la posición se obtiene según indican los marcadores verdes a través de la fusión sensorial. Debido a esto se puede observar como en el punto 1 aparece una acumulación de marcadores verdes, ya que las US ofrecen todas esas medidas de posición en un solo paso. La IMU

(marcadores naranjas) se inicia la posición del primer marcador azul. Entre el punto 1 y el punto 2, al no existir cobertura U-LPS tanto los marcadores verdes como los naranjas funcionan únicamente con las medidas de los sensores inerciales, debido a esto se observan cómo progresan casi a la par. Existe una pequeña diferencia debida a que los marcadores verdes antes de iniciar el cálculo únicamente con la IMU han realizado la fusión sensorial con lo que se ha corregido la trayectoria de salida del individuo. Una vez que se alcanza el punto dos, se observa como la posición final de la trayectoria 1-2 obtenida por los sensores inerciales y representada a través de los marcadores verdes y naranjas señalados en la Figura 41, presenta un error con respecto a la posición real al alcanzar el punto 2, obtenida a través de las US y representada por el marcador azul del punto 2 (inicio trayectoria 2-3). Esas dos posiciones deberían ser las mismas pero debido a que los sensores inerciales generan un error acumulativo, al llegar a la posición 2 este error se presenta como una pequeña diferencia entre la posición obtenida por las US y la posición relativa ofrecida por la IMU. Cuando se alcanza el U-LPS ubicado en la posición 2, los marcadores verdes obtienen la posición a través de la fusión sensorial gracias a las US y debido a esto se corrige la posición y el error que almacenaban por culpa de la IMU.

En el punto 2 se debe realizar un giro de 90º, este tipo de giros para los sensores inerciales es una gran dificultad debido a la estimación del ángulo de orientación, y como se puede observar en la Figura 41 los marcadores naranjas pertenecientes a la IMU solo realizan un pequeño giro muy lejano a los 90º que se realizan. A diferencia de esto, a través de la fusión sensorial, este ángulo de orientación se corrige y se obtiene un giro de 90º casi perfecto en los marcadores verdes. Una vez que se sale de la zona de cobertura del punto 2, la trayectoria hasta el punto 3 representada a través de los marcadores verdes se obtiene a través de la IMU, pero corregida gracias a la fusión sensorial. Por último, al alcanzar el punto 3 se observa como el primer marcador azul se encuentra desviado de la trayectoria que traían los marcadores verdes. Esto se debe a que entre el punto 2 y 3 la IMU vuelve a acumular un error que al llegar a una nueva zona de cobertura es corregido nuevamente por la fusión sensorial gracias a las US y se trasladan los siguientes marcadores verdes a la posición correcta.

Con esta prueba se observa claramente como si se hubiese realizado la trayectoria únicamente con la información de la IMU (marcadores naranjas), la trayectoria hubiese sido muy diferente a la real, quedándose más corta y más alejada de la posición final real. En cambio, al usar la información de los U-LPS, se observa como la trayectoria (marcadores verdes) mejora mucho y se asemeja más a la realizada en realidad. Aun así, no se debe olvidar que, si únicamente se hubiese hecho uso de la información de los US, se deberían haber instalado aproximadamente 12 U-LPS para abarcar los 37m de trayectoria recorridos. A diferencia de esto, al haber hecho uso de la información de los sensores inerciales esta infraestructura se ha reducido únicamente a tres U-LPS, lo que reduce el coste de la infraestructura y por consiguiente los costes de mantenimiento.

Para finalizar este análisis se procede a realizar un estudio de los tiempos de ejecución de la aplicación. En la Tabla 2 se observan los diferentes tiempos de ejecución de cada parte de los procesos principales de la aplicación. Se realizó una batería de 590 medidas en movimiento en las que se observaron el tiempo invertido del Hilo US para realizar las correlaciones y el cálculo de Gauss Newton. Se observa que el 97% del tiempo se emplea en las correlaciones. En otras pruebas los tiempos de ejecución del Hilo de la IMU (33 medidas) y el algoritmo de fusión sensorial (101 medidas), son mucho menores que el empleado en el hilo US, con lo que el tiempo de ejecución de la aplicación viene marcado por el hilo US.

Proceso	***Media tempos de ejecución (ms)***	***Desviación estándar del tiempo de ejecución (ms)***
Correlaciones	30	14
Gauss newton	0.71	0.81
Google Maps	5.56	3.37
Hilo IMU	3.06	1.63
Fusión	0.52	0.18

Tabla 2: Análisis de tiempos de ejecución.

Con este estudio se podría confirmar que la media de los tiempos de ejecución de la aplicación viene marcada por el hilo US y la representación en Google Maps y sería 36ms, un tiempo muy aceptable para realizar el posicionamiento en tiempo real de personas, ya que estas suelen caminar con una velocidad media aproximada de 1 paso por segundo. Aun así se debe tener en cuenta que ésta es la media de los tiempos de ejecución de los algoritmos de la aplicación, pero que al tener que recibir 5 buffer de 8192 muestras de los US, aun recibiéndose en paralelo, en esta parte aparece otra limitación en los tiempos, ya que para disponer de la información de los US, muestreando a 100kHz, el buffer de recepción tardará en llenarse aproximadamente 82ms, además de tener que esa información debe ser transmitida por el puerto USB al dispositivo portable. De esta manera se ha observado que la media de los tiempos que se tarda en tener una medida de las US disponible para procesarla de 679 medidas adquiridas es de 100.06ms con una desviación estándar de 8.74ms. Observando estos valores se puede comprobar que los tiempos medios de ejecución con los que trabaja la aplicación son más que suficiente para que se pueda posicionar en paralelo mientras se espera una nueva recepción de las US y no es necesario optimizarlos más debido a que la adquisición de las US limita por hardware el tiempo medio de ejecución de la aplicación como se puede observar en la Figura 42. Cuando se navega únicamente con los sensores inerciales, hay que tener en cuenta que, aunque el tiempo medio de ejecución del hilo de la IMU es 3ms, la estimación de la posición la marcará la velocidad con la que se da cada paso que debe detectar la IMU.

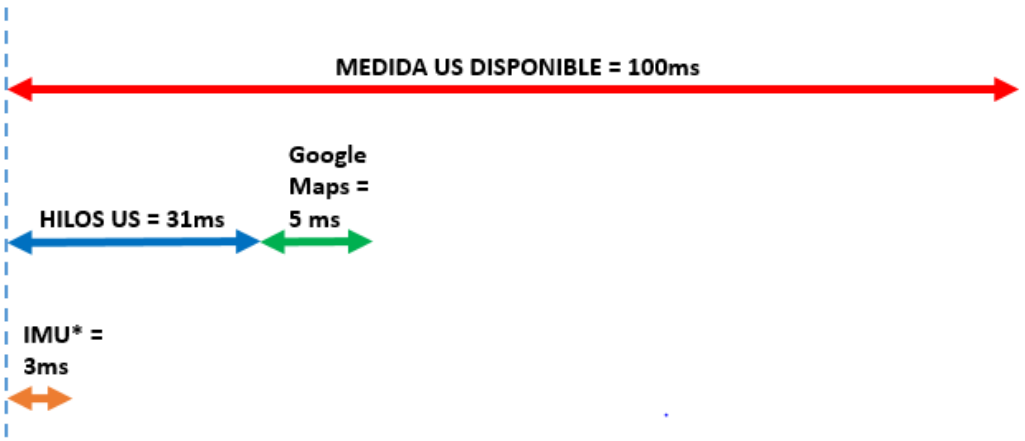


Figura 42: Diagrama de tiempos de ejecución.

CAPÍTULO 6: CONCLUSIONES Y TRABAJOS FUTUROS

El TFG aquí presente ha conseguido abordar cada uno de los objetivos que se han planteado al inicio de éste, en el apartado (1.2). Como objetivo principal se ha desarrollado una aplicación Android capaz de posicionar personas en espacios interiores extensos haciendo uso de señales ultrasónicas y la información de sensores inerciales.

La aplicación desarrollada es capaz de posicionar haciendo tanto uso de las señales ultrasónicas como de los sensores inerciales, además se ha implementado un algoritmo de fusión sensorial basado en un Filtro de Kalman Extendido para fusionar ambas informaciones en el caso de que se disponga de ellas.

Además, este trabajo ha empleado un esfuerzo importante en la mejora del procesamiento de las señales ultrasónicas. Optimizando los tiempos empleados para la correlación de las señales ultrasónicas y modificando los algoritmos para que la aplicación sea capaz de posicionarse en ausencia de alguna de las medidas necesarias.

Se ha desarrollado una interfaz de aplicación atractiva para el usuario, haciendo uso de los mapas de interiores ofrecidos por la API de Google Maps, a través de la cual y gracias a la optimización de los tiempos de ejecución de la aplicación, se ha conseguido la representación en tiempo real de la posición estimada del usuario en los mapas de interiores del edificio. Además, la aplicación ofrece información en segundo plano como una representación de las correlaciones realizadas a las señales ultrasónicas, lo que permite observar si existe algún fallo con estas, además de una representación en tiempo real de las señales ofrecidas por los sensores ultrasónicos (*Roll*, *Pitch* y *Yaw*).

De esta manera se consigue una aplicación capaz de ofrecer al usuario la representación de su posición estimada en ese momento en un entorno interior extenso. La representación de la posición del usuario, gracias a que la aplicación cuenta con dos modos de funcionamiento podrá dejar rastro de las posiciones anteriores en el modo de ejecución “depurador” (*DEBUG*) y en el modo de ejecución “demostrador" (*DEMO*) ofrecerá únicamente la representación de un solo marcador, el cual se mueve a la vez que cambia la posición del usuario dando la misma

sensación que ofrecen las aplicaciones de posicionamiento en exteriores como (*Google Maps*).

Como trabajos futuros se encuentran la implementación de restricciones de mapa una técnica muy utilizada en sistemas de posicionamiento gracias a la cual se mejoraría la navegación en interiores, ya que existen restricciones ofrecidas por el entorno como el no poder atravesar paredes, que podrían ser implementadas y tenidas en cuenta por el sistema.

Otra de las líneas para continuar trabajando sería la implementación del posicionamiento 3D, el cual se puede orientar al posicionamiento de drones en interiores. El posicionamiento de drones es más exigente que el de personas ya que la velocidad a la que se puede mover un dron es mucho mayor, además de las interferencias que puede introducir el ruido generado con los motores a las señales ultrasónicas utilizadas para el posicionamiento.

Por último, otro de los trabajos futuros es la implementación de técnicas para la transición suave entre posicionamiento en exteriores y posicionamiento en interiores, de manera que la aplicación sea capaz de posicionarte tanto cuando te encuentras fuera de un edificio, como cuando entras a un edificio, haciendo uso del sistema presentado en este TFG junto al posicionamiento ofrecido por el GPS en exteriores.

CAPÍTULO 7: PRESUPUESTO

En este apartado se van a exponer todas las partidas presupuestarias necesarias para la realización del proyecto. En él se desglosarán tanto los costes del material utilizado, como los costes directos de la aplicación, los cuales vienen dados por las horas empleadas por el programador, en este caso el propio alumno encargado de realizar este TFG, además de los gastos indirectos de la aplicación.

7.1 Coste del material utilizado

En la Tabla 3 se adjunta el desglose de los costes asociados a cada uno de los elementos físicos necesarios para la realización del proyecto:

Material	Precio Unitario (€)	Unidades	Periodo de amortización (años)	Uso (meses)	Coste de amortización (€)
U-LPS	240	3	5	9	108
Shimmer (IMU)	359	1	3	9	89.75
Tablet Samsung Galaxy Tab S2	425	1	3	9	106.25
Receptor Ultrasónico	20	1	6	9	2.5
PC Cooler Master i7, 4.2GHz	735	1	3	9	183.75
Total	**1779**				**490.25**

Tabla 3: Costes del material utilizado.

7.2 Costes directos de la aplicación

Como coste directo debido al tiempo invertido tanto por el programador como por el escritor para la realización de la aplicación.

A continuación, se observa un desglose de los precios y los tiempos invertidos para la realización del sistema. Aproximadamente se han empleado 600 horas (10

meses), de los cuales aproximadamente 540 horas (9 meses) se han invertido en labores de investigación y programación, mientras que 60 horas (1 mes) se han invertido en la documentación del sistema.

Actividad	Coste (€/Mes)	Tiempo (Meses)	Coste Total (€)
Investigación y desarrollo software	1600	9	14400
Documentación	1000	1	1000
Total			**15400**

Tabla 4: Costes directos de la aplicación.

7.3 Costes indirectos de la aplicación

Los costes indirectos de la aplicación vienen dados por las licencias de los programas necesarios para el desarrollo del proyecto. En la se observa un desglose de estos.

Licencia	Coste (€)	Periodo de amortización (años)	Uso (meses)	Coste de amortización (€)
Matlab r2017b	2000	2	7	583.3
Android Studio 2.3	0	2	7	0
Microsoft Office 2016	450	2	7	131.25
Consensys	0	2	7	0
Total	**2450**			**714.58**

Tabla 5: Costes indirectos de la aplicación.

7.4 Costes del proyecto

Por lo tanto, en la se muestra un resumen de todos los gastos asociados al proyecto, y el coste total de la aplicación.

Concepto	Coste íntegro del proyecto (€)	Coste de amortización (€)
Coste del material utilizado	1779	490.25
Costes directos de la aplicación	15400	15400
Costes indirectos de la aplicación	2450	714.58
Total (sin IVA)	19629	16604.83
Total (IVA incluido 21%)	**23751.09**	**20091.85**

Tabla 6: Coste del proyecto.

Por lo tanto, el importe total estimado del proyecto presente en este trabajo asciende a la cantidad de:

Veinte mil noventa y un euros con ochenta y cinco céntimos.

Guadalajara, a 11 de junio de 2018.

Firmado: Sergio Pérez Bachiller.

Graduado en Ingeniería en Automática y Electrónica Industrial.

CAPÍTULO 8: PLIEGO DE CONDICIONES

En este capítulo se van a exponer todos los requisitos necesarios para poder hacer uso de una forma correcta de la aplicación desarrollada, no asegurando el correcto funcionamiento de la misma si no se cumplen. Las condiciones bajo las cuales se ha desarrollada la aplicación se exponen a continuación:

- Aplicación desarrollada para Smartphone Android con versión 4.0.3 (API 15) o superior.
- IMU que se debe utilizar es la Shimmer3. Esto se debe al protocolo de comunicación bluetooth que se ha programado como se ha visto en la sección (4.3), si se desease hacer uso de otra IMU, se deberá modificar el protocolo de comunicación para la obtención de las señales de los giróscopos y acelerómetros en el dispositivo portable, como por ejemplo hacer uso de un protocolo de comunicación WiFi como incorporan alguno de estos dispositivos. Una vez que se tuviese la información de los giróscopos y los acelerómetros en el dispositivo portable, el procesamiento de esas señales sería el mismo.
- La IMU debe estar sujeta en la parte superior de la pierna, alineada con respecto a la orientación del pie y no se debe mover durante las pruebas. Si se desease posicionar la IMU en otra ubicación como por ejemplo el tobillo, los algoritmos desarrollados en [Cervigón, 2017] deberían de ser modificados.
- Debido a que la comunicación entre el dispositivo portable y la IMU se realiza vía Bluetooth, será necesario dar permiso al Bluetooth del dispositivo portable. En el caso de que no se encuentre activado el Bluetooth, la aplicación solicitará automáticamente este a través de un diálogo como el mostrado en la Figura 43:

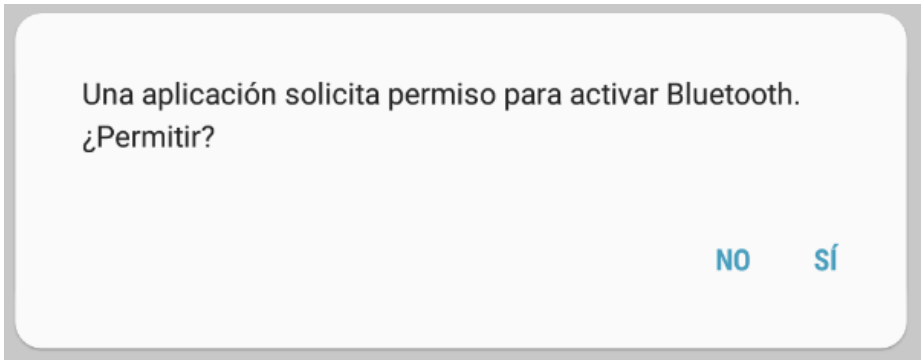


Figura 43: Diálogo para solicitar el permiso Bluetooth.

- Como una vez finalizada la prueba, los datos recopilados en la misma se guardan en la memoria del dispositivo portable por si se quiere hacer uso de ellos posteriormente. Es necesario dar permiso a la aplicación para acceder a los archivos del dispositivo. Al igual que en la situación del Bluetooth, si el permiso no está otorgado, a través de un dialogo como el mostrado en la Figura 44 se ofrecerá al usuario otorgarlo.

Figura 44: Diálogo para permitir el acceso a los archivos del dispositivo portable.

- Al hacer uso de los mapas de Google Maps, para que éstos puedan ser cargados correctamente a través de la API, es necesario que la aplicación tenga acceso a internet, ya que los mapas no se cargarán correctamente en el caso de no tener acceso a internet. Al igual que en las situaciones anteriores, si la aplicación no tiene acceso a internet, debido a que esta desactivado en el teléfono, a través de un diálogo se ofrecerá al usuario la posibilidad de activarlo si ser necesario salir de la aplicación. La aplicación puede activar automáticamente el internet, pero es más correcto pedir permiso al usuario a la hora de realizar esta acción. El diálogo se muestra en la Figura 45:

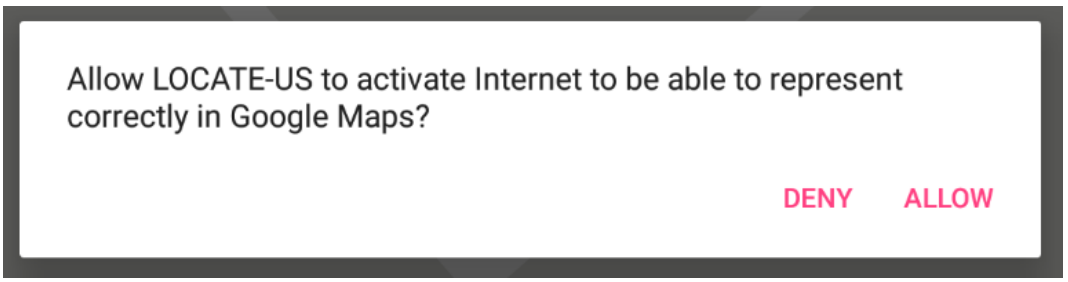


Figura 45: Diálogo para permitir el acceso a internet.

- Los U-LPS que deben ser desplegados por el entorno deben ser los especificados en el apartado (3.2), y deben ser configurados para emitir códigos Kasami, además de saber que códigos han sido asignados a cada U-LPS y concretamente a cada baliza. Además, estos códigos deben cumplir las condiciones comentadas en el apartado (2.2). No obstante, la aplicación puede ser fácilmente adaptada para el procesamiento de otro tipo de códigos.
- El módulo receptor de ultrasonidos debe ser el especificado en el apartado (3.4), y si se hace uso de otro, debe cumplir con especificaciones similares, ya que el dispositivo portable debe recibir las señales emitidas por los U-LPS muestreadas a 100KHz en un buffer mínimo de 8192 muestras.
- Se debe tener información de un mapa vectorial donde se desee desplegar el sistema, es decir se debe tener un mapa XML de cada edificio concretamente donde se conozca la posición 0,0 del mismo y su conversión a formato UTM, la cual se podrá obtener una vez que se conozca la posición 0,0 del edificio a través de la siguiente página web:

  http://boulter.com/gps/ - 40.51293786431562%2C-3.3499702624976635

  Los valores de la posición 0,0 del edificio en formato UTM, serán cargadas en las variables del código X0 e Y0.

CAPÍTULO 9: MANUAL DE USUARIO

9.1 Descripción de la aplicación

LOCATE-US es una aplicación que permite estimar la posición de personas en espacios interiores extensos, permitiendo al usuario visualizar su posición en tiempo real a través de un marcador en los mapas de interiores ofrecidos por Google Maps. Además, ofrece información adicional de la tecnología utilizada: señales ultrasónicas y sensores inerciales, con el fin de facilitar a los investigadores y desarrolladores de estos sistemas el estudio de los mismos.

9.2 Descarga de la aplicación.

La descarga e instalación de la aplicación en el dispositivo portable se realizará a través de la plataforma de desarrollo Android Studio. Para ello se deberá disponer del código de la aplicación desarrollado en este TFG y seguir los siguientes pasos:

1. Abrir el programa "Locate-us" a través de la plataforma Android Studio y seleccionar el boto "*Run 'app'*" como se muestra en la Figura 46:

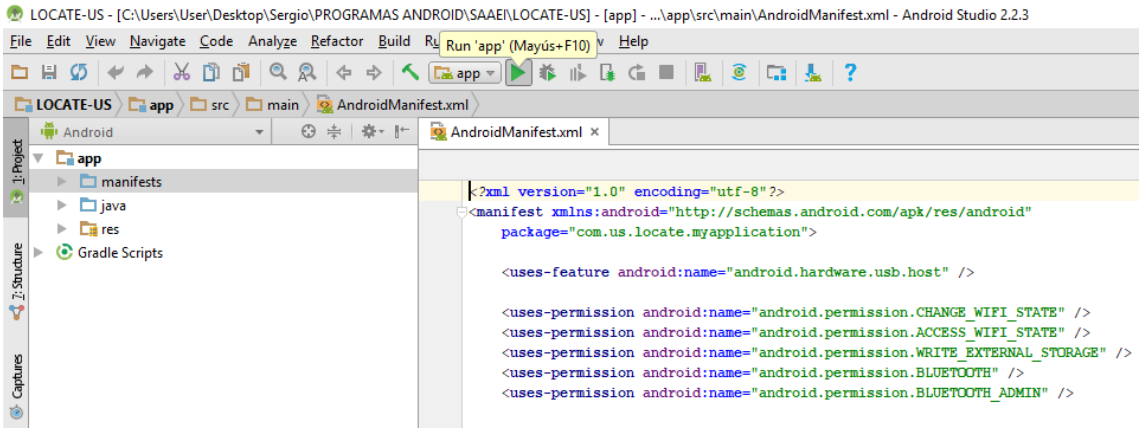


Figura 46: Instalación app paso 1.

2. Seleccionar el dispositivo, el cual debe estar conectado al ordenador y debe tener habilitada la opción de desarrollador (**Ajustes > Información del teléfono > Pulsa varias veces Número de compilación**) donde se desea descargar e instalar la aplicación y pulsar "*OK*" como se observa en la Figura 47. En este trabajo se hace uso de la Tablet Samsung, y como se puede observar en el cuadro de selección viene identificada el nombre de la Tablet de la cual se selecciona para cargar en ella la aplicación.

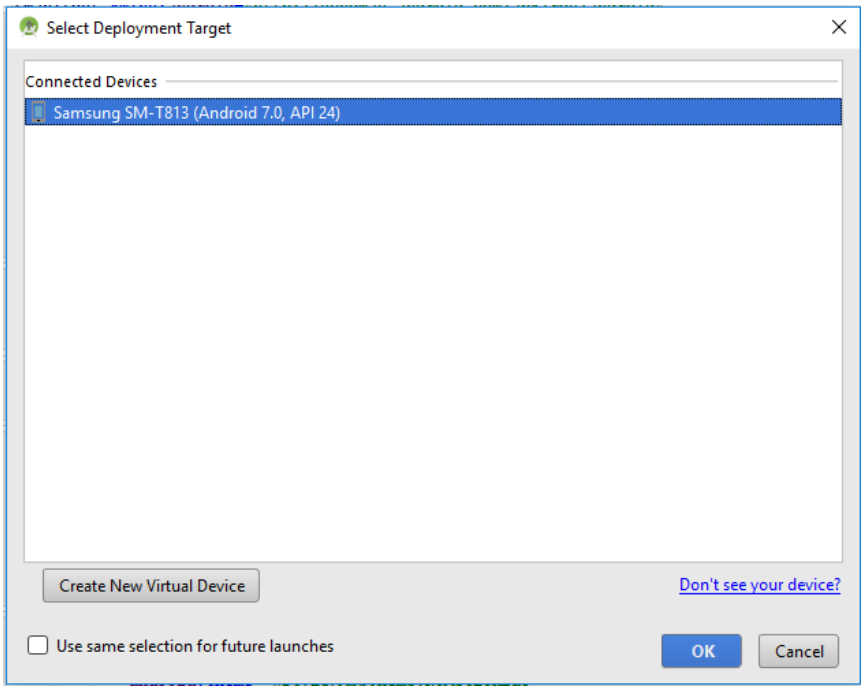


Figura 47: Instalación de la app paso 2.

Una vez se realicen estos dos pasos, la aplicación se descargará e instalará automáticamente en el dispositivo seleccionado, y posteriormente se podrá hacer uso de ella.

9.3 Navegación por la aplicación

En esta sección se va a proceder a realizar una breve explicación del funcionamiento de la aplicación una vez arrancada. Se expondrán tanto las diferentes pantallas y opciones que ofrece la aplicación como la navegación entre pantallas y el uso de las opciones.

9.3.1 Pantalla de presentación

En la Figura 48 se muestra la pantalla de presentación que se ha diseñado para la aplicación, esta pantalla aparecerá nada más arrancar la aplicación. Esta pantalla sirve como presentación para abrir la aplicación, mostrando el logo de ésta y alguna información añadida. Además, por detrás mientras se ejecuta la pantalla se aprovecha a cargar variables y archivos que serán necesarios posteriormente. La duración de esta pantalla es la duración que se tarda en realizar la carga de los archivos, o en el caso de que se ya estén cargados, la pantalla tendrá una duración de 2 segundos. En la parte inferior de la pantalla se ha configurado una barra de progreso circular rosa, que girará mientras la pantalla se está ejecutando para evitar la sensación de bloqueo de la aplicación.

Figura 48: Pantalla de presentación.

9.3.2 Pantalla principal

Una vez ejecutada la pantalla de presentación, se cargará automáticamente la pantalla principal de la aplicación. Esta pantalla Figura 49, es la pantalla de la que se hará uso principalmente cuando se esté ejecutando la aplicación. Está formada principalmente por los mapas de interiores de Google Maps, donde se podrá observar la posición actual en tiempo real del usuario mientras hace uso de la aplicación. En la parte superior aparece una leyenda con los diferentes colores de marcadores que pueden aparecer en la aplicación para representar la posición, lo cuales dependerán como se ha explicado en secciones anteriores, de cómo se haya estimado la posición, si con los US, los sensores inerciales (IMU), o la fusión de ambos (US+IMU), de esta manera se podrá tener una idea de la precisión de la posición ofrecida por la aplicación. En la parte inferior de la aplicación se observan unos cuadros de textos, donde aparecerá información como si se dispone de cobertura U-LPS en ese momento, la posición actual estimada, los pasos dados, y la distancia recorrida. A la derecha de los cuadros de texto se observan dos interruptores, a través de los cuales se podrá seleccionar el modo de ejecución modo depurador ("*DEBUG MODE*") y modo demostrador ("*DEMO MODE*"), explicados en las secciones anteriores, con el que se quiere para realizar la prueba. En la parte más inferior de la pantalla se observan cuatro pulsadores, a través de los cuales se puede seleccionar el comienzo del posicionamiento ("*START TEST*"), la finalización del posicionamiento ("*END TEST*"), la limpieza de los cuadros de texto y los marcadores del mapa ("*SCREEN CLEANNING BUTTON*") y la apertura de un fichero .txt donde se observan las variables con la posición y la orientación una vez finalizada la prueba ("*FILE .txt POSITIOM [X, Y]*"). En la parte superior de la pantalla a la izquierda del nombre de la aplicación "LOCATE-US", aparece un pulsador a través del cual se puede acceder al menú deslizante de navegación (*Navegation drawer*) el cual se explicará a continuación y al cual también se podrá acceder arrastrando el dedo desde el lado izquierdo de la pantalla de la aplicación a la derecha.

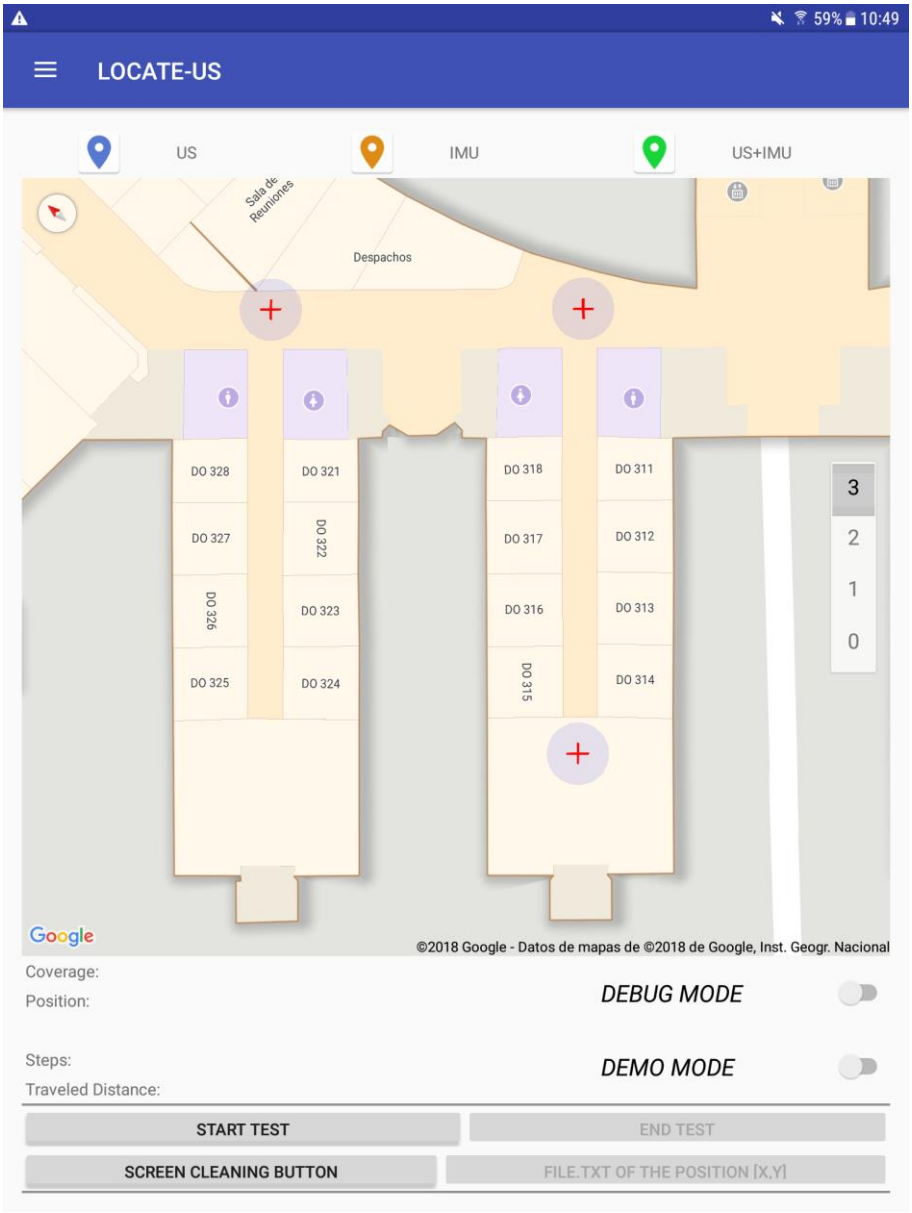


Figura 49: Pantalla principal.

Para el diseño de la pantalla principal se ha tenido en cuenta poder tener en la misma pantalla los mapas del interior del edificio, junto con los pulsadores para interactuar con la aplicación. El objetivo principal de este diseño ha sido que el usuario pueda llevar el dispositivo portable en la mano, observando en tiempo real su posición en el mapa, y a la vez en la misma pantalla tenga al alcance los pulsadores principales necesarios para hacer uso la aplicación.

9.3.3 Menú deslizante de navegación.

Como se ha comentado en el apartado anterior, la aplicación cuenta con un menú deslizante de navegación (*Navegation drawer*). A dicho menú se tendrá acceso desde todas las pantallas de la aplicación del mismo modo, exceptuando la pantalla de presentación, desde la cual no se tiene acceso. La finalidad de este menú es permitir realizar la transición entre las diferentes pantallas de la aplicación. Si el usuario selecciona "*Positioning*", se accederá a la pantalla principal. Con "*Bluetooth*" se accederá a la pantalla de configuración del Bluetooth que se comentará posteriormente. Seleccionado "*Plots IMU*" y "Plots *CORR*", se accederá respectivamente a la pantalla de representación de la información de la IMU y la pantalla de representación de la información de los US, ambas serán explicadas posteriormente.

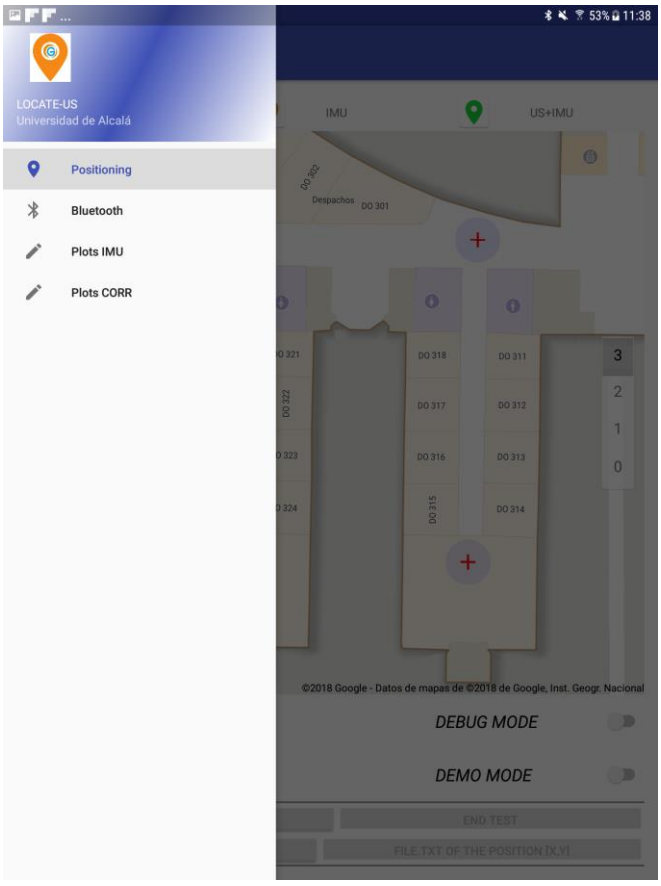


Figura 50: Menú deslizante.

9.3.4 Pantalla configuración Bluetooth

Accediendo a la pantalla de configuración Bluetooth, el usuario es capaz de sincronizar la conexión Bluetooth entre el dispositivo portable y la IMU de la que se quiere hacer uso. En esta pantalla aparecerá una lista con las IMUs de las que se puede hacer uso, seleccionando una de ellas y presionando el botón de conectar, se procederá a enlazar ambos dispositivos. También se ofrece el botón de desconectar, y un pulsador ("NAVEGATION WITHOUT IMU: ON/OFF") que da la posibilidad de no seleccionar ningún dispositivo IMU, y con ello realizar el posicionamiento únicamente con las US como se ha comentado en secciones anteriores.

Figura 51: Pantalla de configuración de la IMU.

9.3.5 Pantalla información de la IMU

En la aplicación ha sido configurada una pantalla que ofrece información sobre los sensores inerciales, concretamente ofrece tres gráficas donde pueden ser visualizados el *Roll,* el *Pitch*, y el *Yaw* en tiempo real. Para observar dichas gráficas es necesario que el usuario pulse el botón "*PLOT IMU*" y si desea que se deje de representar con salir a otra pantalla o pulsar "*STOP PLOT*" la representación dejará de realizarse. Gracias a esta pantalla se puede visualizar la evolución de los valores ofrecidos por la IMU en tiempo real.

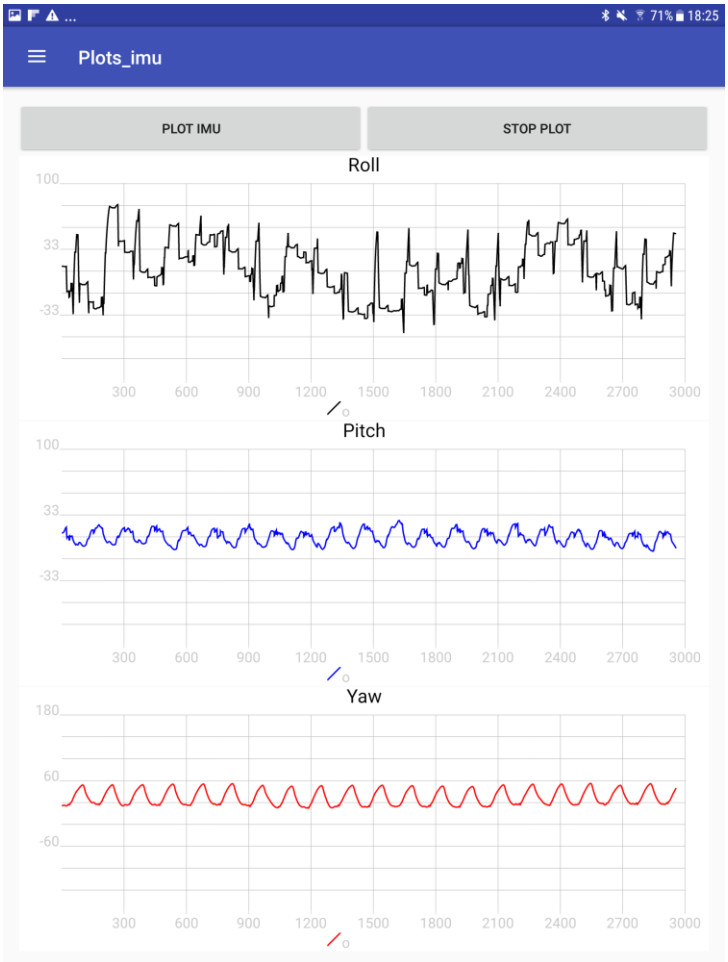
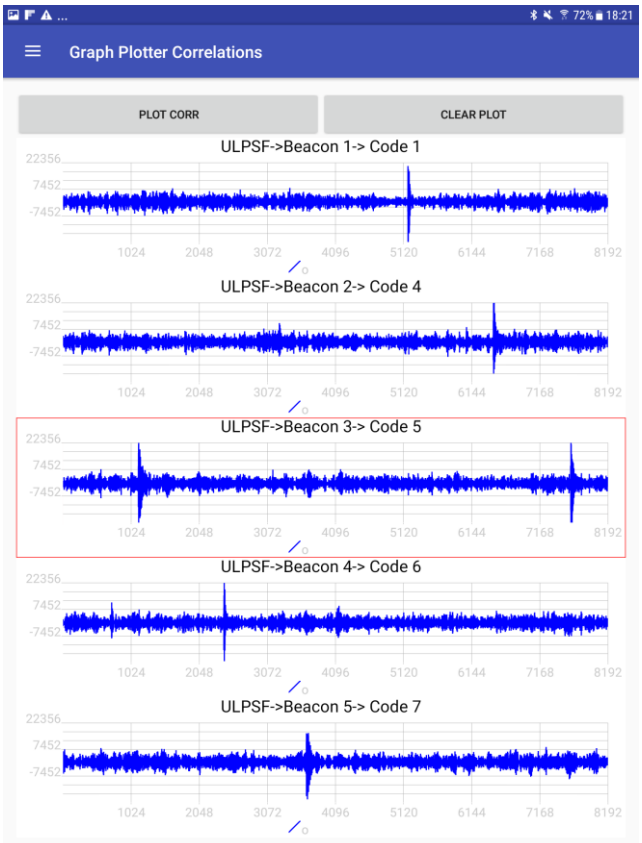


Figura 52: Pantalla información IMU.

9.3.6 Pantalla información de los US

En la aplicación también se encuentra una pantalla que ofrece información de los US. Concretamente permite visualizar las 5 correlaciones realizadas a cada baliza del U-LPS que este ofreciendo cobertura en dicho momento. Para observarlas se debe presionar el botón "*PLOT CORR*" cada vez que se quiera visualizar el resultado de las 5 correlaciones. Además, la app ofrece la posibilidad de borrar las 5 gráficas presionado "*CLEAR PLOT*". Cada representación de cada correlación ofrecerá un título donde se identificará el U-LPS que ofrece cobertura (U-LPSF), la baliza que envía el código que ha sido correlado (*Beacon X; 1<=X<=5*) y el código Kasami US que emite dicha baliza (*Code X; 1<=X<=5*). Gracias a toda esta información se puede observar si todas las balizas funcionan correctamente y se reciben en un orden correcto. Además, la correlación de mayor energía (es decir la baliza que se ha tomado como referencia) aparecerá recuadrada en rojo, y el resto de las gráficas ajustarán su eje y al valor de esta correlación ya que es la de mayor energía, y así conseguir una representación dinámica y escalada al valor de correlación de mayor



energía.

Figura 53: Pantalla información US.

9.3.7 Configuraciones específicas

En esta sección se van a exponer algunas configuraciones que se han programado en la aplicación para facilitar el uso y la navegación en ella del usuario.

1. Los botones de la pantalla principal se activan y se desactivan de forma automática dependiendo de la acción que se pueda realizar en cada momento. Por ejemplo, en la Figura 49 se observa como los únicos botones habilitados son el "START TEST" y el "SCREEN CLEANNING BUTTON" ya que al no haber iniciado aun la prueba, no serviría de nada pulsar el botón de finalizar prueba o el de representar los valores en el '.txt' ya que no se tienen aún valores. Esta configuración aparte de estar orientada para minimizar un incorrecto uso del usuario de la aplicación se ha orientado a que el usuario pueda hacer uso de la aplicación sin haberla visto nunca, ya que la propia aplicación va ofreciendo los pasos que se debe seguir.
2. Si se desea comenzar el posicionamiento, y no se tiene conectada ninguna IMU y/o no se ha especificado que no se vaya a hacer uso de la misma, cuando el usuario pulse el botón "START TEST" de la pantalla principal, la aplicación abrirá automáticamente la pantalla de configuración del Bluetooth para hacer que el usuario seleccione una IMU o especifique que no desea hacer uso de IMU.
3. En el caso se desee comenzar el posicionamiento, pero no se haya conectado el módulo receptor US, aparecerá un diálogo de alerta en la pantalla de la aplicación recordando al usuario que debe verificar la conexión del USB, ya que puede estar mal conectada o puede que se le haya olvidado conectar el módulo de adquisición US.

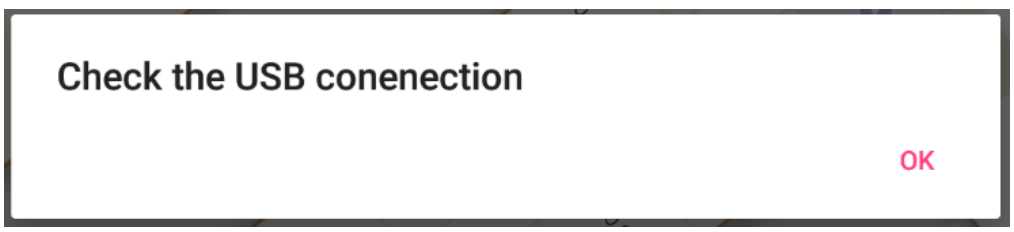


Figura 54: Diálogo de verificación USB.

4. La app requiere que se verifique un modo de ejecución (“DEMO MODE” o “DEBUG MODE”) a no ser que solo se haga uso de US, que se activará automáticamente el modo demostrador automáticamente. Si no es el caso y es necesario seleccionar un modo y el usuario no lo ha hecho, cuando se pulse el botón de comenzar el posicionamiento (“START TEST”), aparecerá un diálogo donde el usuario podrá seleccionar el modo de ejecución que desee, ya que debe de estar especificado.

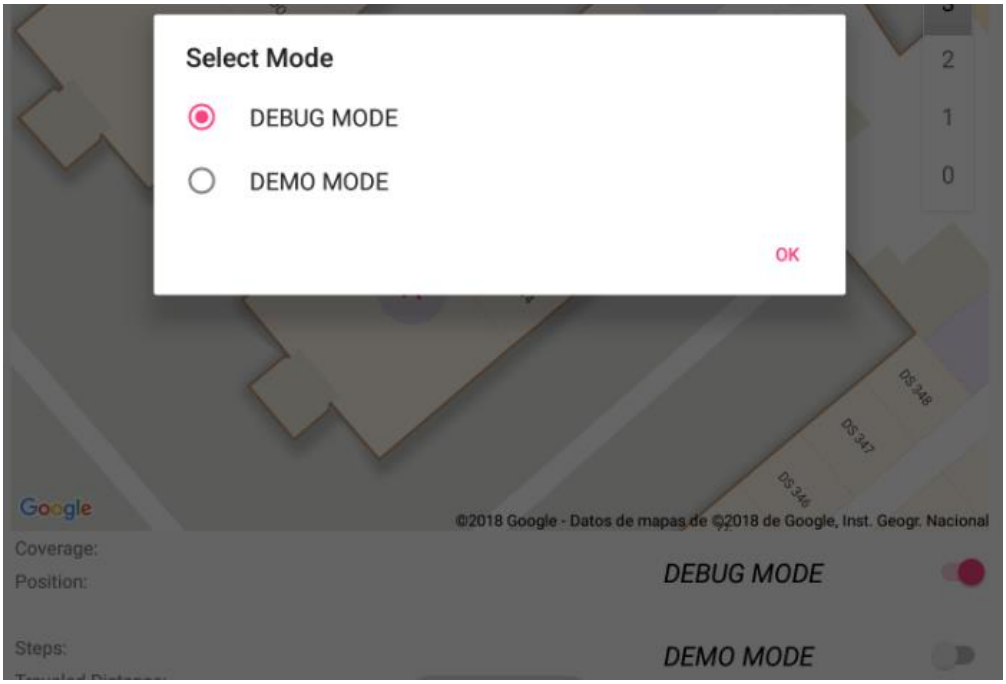


Figura 55: Diálogo de selección de modo.

5. Aunque la aplicación siga funcionando si se deja en segundo plano en el dispositivo, por ejemplo al recibir una llamada, o al abrir otra aplicación, si el usuario pulsa el botón atrás del dispositivo portable estando en la pantalla principal, la aplicación se cerrará y dejará de ejecutarse el posicionamiento, para evitar eso se ha configurado un dialogo de alerta para avisar al usuario si realmente quiere salir de la app, ya que en muchas ocasiones puede que se haya presionado el botón de forma errónea. Si se pulsa dicho botón desde cualquier de las demás pantallas de la aplicación no se cerrará la aplicación, sino que se regresará a la pantalla principal, con lo que este dialogo de alerta no aparecerá.

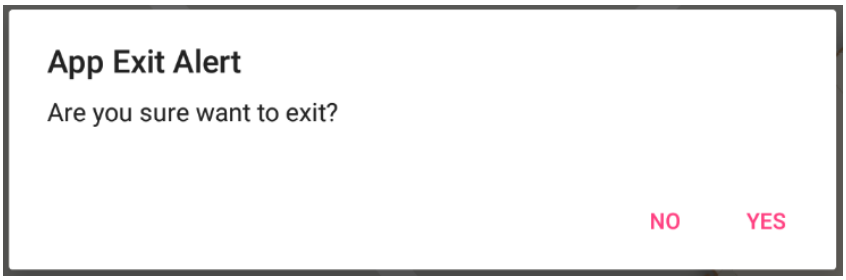


Figura 56: Alerta de salida de la aplicación.

Todas estas configuraciones han sido programas con el fin de que el usuario sea capaz de hacer uso de una aplicación de una forma sencilla y evitando posibles usos incorrectos.

CAPÍTULO 10: BIBLIOGRAFÍA

[Aguilera, 2013] T. Aguilera, J. A. Paredes, F. J. Álvarez, J. I. Suárez, A. Hernández, "Acoustic Local Positioning System Using an iOS device", 2013 International partir de medidas de sensores inerciales", Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad de Alcalá, 2017.Conference on Indoor Positioning and Indoor Navigation, pp. 1-5, Montbeliard, Francia, 2013.

[Android, 2018] https://developer.android.com/index.html, último acceso abril 2018.

[Cervigón, 2017] R. Cervigón. Rey "Implementación de un EKF sobre plataformas Android para posicionamiento de dispositivos portables en espacios interiores a partir de medidas de sensores inerciales, TFG, 2017.

[Cox, 1964] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems", IEEE Transactions on Automatic Control, vol.9, no.1, pp.5-12, 1964.

[Díaz, 2017] E. Díaz, M. C. Pérez, D. Gualda, J. M. Villadangos, J. Ureña, J. J. García "LOCATE-US: sistema de posicionamiento ultrasónico de dispositivos portátiles en espacios interiores" SAAEI, 2017.

[Ferris, 2006] B. Ferris, D. Hahnel, D. Fox, "Gaussian Processes for Signal Strength-Based Location Estimation," Proc. of Robotics Science and Systems, Philadelphia, PA, USA, 2006.

[Filonenko, 2013] V. Filonenko, C. Cullen, J. D. Carswell, "Índoor positioning for smartphones using asynchronous ultrasound trilateration," ÍSPRS Ínternational Journal of Geo-Information, vol. 2, pp. 598-620, 2013.

[Gualda, 2014] D. Gualda, J. Ureña, Juan C. García, Alejandro Lindo, "Locally-Referenced Ultrasonic – LPS for Localization and Navigation" Sensors, 2014.

[Hervás, 2014] R. Hervás, S. Lee, C. Nugent, J. Bravo, "Ubiquitous Computing and Ambient Intelligence: Personalisation and User Adapted Services", 8th International Conference UCAml, Belfast, UK, 2014.

[Kalman, 1960] R.E. Kalman, “A new approach to linear filtering and prediction problems”, J. Basic Eng., vol.82, no.1, pp.35-45, 1960.

[Kalman, 1961] R.E. Kalman, “New Results in Linear Filtering and Prediction Theory”, J. Basic Eng., vol.83, no.1, pp.95-108, 1961.

[Kasami, 1969] T. Kasami, “Weight distribution formula for some class of cyclic codes”, Combinational Mathematics and its Appl., Chapel Hill, N.C.: University of North Carolina Press, 1969.

[Knowles, 2018] Knowles Aoustics Application Note, SiSonic Design Guide, https://media.digikey.com/pdf/Data%20Sheets/Knowles%20Acoustics%20PDFs/SiSonic_Design_Guide.pdf , último acceso marzo 2018.

[Lazik, 2012] P. Lazik, A. Rowe, “Indoor Pseudo-ranging of Mobile Devices using Ultrasonic Chirps,” Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, pp. 99-112, 2012.

[Lindo, 2014] Alejandro Lindo, Mar´ıa del Carmen Perez, Jes ´ us Ure ´ na, David Gualda, Enrique Garc ˜ ´ıa, J. Manuel Villadangos, “Ultrasonic signal acquisition module for smartphone indoor positioning” ETFA, 2014.

[Lopes, 2014] S. Í. Lopes, J. M. N. Vieria, J. Reis, D. Alburqueque, “Accurate Smartphone indoor positioning using WSN infrastructure and non-invasive audio for TDoA estimation,” Pervasive and Mobile Computing, pp. 1-18, 2014.

[Peng, 2007] C. Peng, G. Shen, Y. Zhang, Y. Li, K. Tan, “BeepBeep: A High Accuracy Acoustic Ranging System using COTS Mobile Devices”, Proc. of the 5th International Concference on Embedded Networked Sensor Systems, pp. 1-14, ACM Sensys, Sydney, Australia, 2007.

[Pérez, 2016] M. C. Pérez, D. Gualda, J. M. Villadangos, J. Ureña, P. Pajuelo, E. Díaz, E. García, “Android application for indoor positioning of mobile devices using ultrasonic signals”, Proceedings of the 2016 Ínternational Conference on Índoor Positioning and Indoor Navigation, 2016.

[Pro-Wave, 2014] Pro-Wave Electronics Corporation, Air Ultrasonic Ceramics Transducers 328ST/R160, http://www.prowave.com.tw/pdf/txall.pdf Especificaciones de producto, 2014.

[Prieto, 2009] J. C. Prieto, A. R. Jiménez, J. Guevara, J. L. Ealo, F. Seco, J. O. Roa, F. Ramos, "Performance evaluation of 3D-LOCUS advanced acoustic LPS," IEEE Trans. Instrum. Measurements, vol. 58, no. 8, pp. 2385-2395, 2009.

[Pushter,2016] J. Pushter, "Smarthpnone Ownership and Internet Usage Continues to Climb in Emerging Economies", Pew Research Center, 2016.

[Shimmer, 2018] SHIMMER3 EMG UNIT http://www.shimmersensing.com/products/shimmer3-emg-sensor#download-tab , último acceso marzo 2018.

[Ureña, 2016] J. Ureña, J. M. Villadangos, D. Gualda, M. C. Pérez, A. Hernández, J. J. García, A. Jiménez, J. C. García, J. F. Arango, E. Díaz, "Technical Description of Locate-US: an Ultrasonic Local Positioning System based on Encoded Beacons" IPIN, 2016.

[Wang, 2016] L. K. Wang, M. H.S. Wang, "Handbook of Environmental Engineering", Springer, 2016.

Universidad de Alcalá

Escuela Politécnica Superior