ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Máster

MEJORAS BASADAS EN EL ANÁLISIS DEL PATH-LOSS SOBRE UN SISTEMA BLE PARA LA LOCALIZACIÓN EN INTERIORES (IMPROVEMENTS BASED ON PATH-LOSS ANALYSIS ON A BLE SYSTEM FOR INDOOR LOCATION)

Para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Carlos Calzada Cabano

10 - 2020

Índice

Índice de Figuras

Índice de Fórmulas

Índice de Tablas

Resumen

El objetivo principal de este trabajo es el de mejorar la solución al problema de localización en interiores que se propuso en el TFG realizado por mí mismo Carlos Calzada [1]. En este trabajo de fin de grado, se abordaba el problema de posicionamiento en interiores por medio de un sistema de balizamiento, usando módulos Bluetooth. Estos módulos transmiten señales o "beacons" los cuales son captados y almacenados por un terminal móvil, el cual será el objetivo para localizar. Estos datos serán procesados posteriormente aplicando un algoritmo, que lo primero que hace, es transforma las potencias recibidas de cada baliza en distancia, a través de una fórmula que tiene en cuenta la variable de perdidas (path loss) obtenida en un estudio del entorno en el que se vayan a colocar el despliegue de balizas, previamente a la realización de este mismo despliegue. Posteriormente se obtendrá la posición estimada usando la trilateración. Los resultados de este sistema obtenían resultados relativamente buenos o malos dependiendo de en qué zona del despliegue nos situemos, siendo buenos cuando el usuario está en una zona rodeado y relativamente cerca de 3 de las balizas (3 es el número necesario para realizar la trilateración), y malos cuando este no se sitúa a distancias cercanas de al menos 3 balizas. La solución que se propuso a este problema es el de poblar aun más de balizas el entorno del experimento, lo cual encarecía mucho más esta solución.

A la vista de los resultados se ha diseñado un nuevo sistema que viene a mejorar el método anterior. Se ha observado que la obtención de la variable de pérdidas de camino que se realizaba en el anterior trabajo era ineficiente, debido a que esta medida se tomaba en un estudio previo al experimento, y cuando se iban a capturar los datos de los "beacons" para estimar la posición, el path loss ya no tenía el mismo valor. Por este motivo, en este trabajo se ha decidió introducir dos calibradores los cuales obtendrán medidas de los "beacons" a la vez que lo hará el usuario, para así obtener el valor de la variable de perdidas en el mismo instante que se obtienen los datos para estimar la posición. Tras realizar las pruebas y comprobar los resultados, se ha obtenido una gran mejora respecto al sistema anterior, sobre todo en las zonas más problemáticas comentadas anteriormente.

Abstract

The main objective of this work is to improve the solution to the indoor location problem that was proposed in the TFG carried out by myself Carlos Calzada. In this final degree project, the problem of positioning indoors was approached by means of a beacon system, using Bluetooth modules. These modules transmit signals or "beacons" which are captured and stored by a mobile terminal, which will be the objective to locate. These data will be processed later by applying an algorithm, which the first thing to do is transform the powers received from each beacon into distance, through a formula that takes into account the variable of path loss obtained in a study of the environment in the one that we are going to place the deployment of beacons, prior to carrying out this same deployment. Later the estimated position will be obtained using trilateration. The results of this system obtained relatively good or bad results depending on which area of the deployment we are located in, being good when we are in an area surrounded and relatively close to 3 of the beacons (3 is the number necessary to carry out the trilateration), and bad when we were not at close distances of at least 3 beacons. The proposed solution to this problem is to populate the experiment environment with even more beacons, which made this solution much more expensive.

In view of the results, a new system has been designed that improves on the previous method. It has been observed that obtaining the path loss variable that was carried out in the previous work was inefficient, because this measure was taken in a study prior to the experiment, and when the data from the beacons were to be captured to estimate the position, the path loss no longer had the same value. For this reason, in this work it has been decided to introduce two calibrators which will obtain measurements of the beacons at the same time as the user, in order to obtain the value of the loss variable at the same moment that we obtain the data for estimate the position. After running the tests and checking the results, we have obtained a great improvement over the previous system, especially in the most problematic areas discussed above.

1 INTRODUCCION Y OBJETIVOS

En este primer capítulo se realizará una breve introducción al problema de localización en interiores, se explicará cuáles son los objetivos de este proyecto respecto a este problema, y por último se definirá la estructura que sigue este documento.

1.1 Introducción

Dentro del ámbito de la computación móvil, uno de los campos más prometedores son los sistemas de posicionamiento. Los sistemas de geolocalización basados en satélites como el (GPS o Galileo) o en las redes de telefonía móvil, son las tecnologías más comunes y asentadas para solucionar el problema del posicionamiento en exteriores donde tenemos visión directa entre el móvil del sujeto a localizar y los satélites o estaciones base. Sin embargo, estas tecnologías son ineficaces a la hora de localizar objetos o personas dentro de los edificios, ya que la señal recibida de estos sistemas se enfrenta al problema de la atenuación y multirayecto debido a los rebotes de la señal [2].

Por este motivo en la actualidad se está buscando cual podría ser la tecnología y método ideal para abordar el problema de la localización en interiores. Una de las soluciones que consigue obtener un alto nivel de precisión son las balizas de ultrasonidos, de Ultra Wide Band (UWB) [3], de infrarrojos y usando otros tipos de sensores dedicados. Estos sistemas ya se han implementado en diversos almacenes automatizados en los cuales se ha obtenido un adecuado funcionamiento, el problema es que esta tecnología obliga a sensorizar todas las estancias y edificios en los que se pretende conseguir posicionamiento. Además, el problema sería que el usuario que quisiera hacer uso de estos sistemas necesitaría un dispositivo diferente para cada edificio. Por este motivo, podemos decir que es verdad que ya existen tecnologías que abordan este problema con una precisión relativamente aceptable, pero sin embargo estos sistemas en su mayoría son muy caros y no son escalables al gran público.

En la actualidad este problema es objeto de investigación por instituciones como el MIT y grandes compañías como Samsung o Google, las cuales, están apostando fuerte por encontrar una tecnología que permita al ciudadano de a pie, poder navegar por dentro de los túneles, edificios y cualquier estancia o lugar cerrado, al igual que lo hacen en las calles, carreteras y demás lugares exteriores.

Para lograr esta hazaña, actualmente se están investigando un gran número de técnicas. Estas técnicas se pueden diferenciar entre las que hacen uso de la infraestructura presente en el edificio o estancia, y las que necesitan la instalación de infraestructura dedicada. Las tecnologías más importantes son [3]:

- **WiFi fingerprinting**: es un sistema de posicionamiento en interiores que utiliza las características de los puntos de acceso Wi-Fi cercanos y otros puntos de acceso inalámbrico para descubrir la ubicación de un dispositivo. Este sistema

se beneficia por el rápido crecimiento a principios del siglo XXI de los puntos de acceso inalámbricos en áreas urbanas. Esta tecnología se basa en medir la intensidad de la señal recibida (indicación de intensidad de la señal recibida o RSSI) y el método de "huella digital". Los parámetros típicos útiles para geolocalizar el punto de acceso inalámbrico incluyen su SSID y dirección MAC. La precisión de este método depende del número de puntos de acceso cercanos cuyas posiciones se hayan ingresado en la base de datos.

- **Fingerprinting**: una evolución de la técnica anterior sería el Fingerprinting, pero usando cualquier tipo de señal que pueda ser captado por el móvil: WiFi, campo magnético, GPS (en el interior de los edificios la intensidad no es suficiente para posicionar, pero se puede recibir alguna señal), etc.
- **Campo magnético**: el campo magnético también se puede usar para hacer un tipo de fingerprinting, pero en este caso no es tanto de puntos, como de líneas. Sería por tanto una técnica más útil para la navegación.

Estas tecnologías son relativamente eficaces cuando se requiere una precisión baja, pero sin embargo siguen sufriendo una gran cantidad de problemas a la hora de localizar: no todos los móviles portan los mismos sensores de recepción de señal, por lo que los móviles con sensores diferentes recibirán la señal de distinta forma; otro problema a la hora de recibir la señal es que las personas absorbemos radiación, por lo que el número de personas presente en una misma estancia afectara a los datos; hacer un mapa de fingerprinting lleva mucho tiempo, aunque para probar algoritmos hay bases de datos de prueba y mecanismos capaces de aumentar la nube de puntos WiFi, etc.

Por último, otro gran problema es el mapeado de las estancias, pasillos y túneles, con la creación de mapas que nos permitan navegar, como lo hacemos ya por las calles y carreteras, al estilo de los mapas del navegador que todos utilizamos. En este sentido hay iniciativas como indoorGML que están estudiando el desarrollo de un estándar para la creación de estos mapas.

1.2 Motivación y Objetivos

El objetivo de este trabajo es el de dar una solución efectiva, barata, y escalable al usuario corriente sobre el problema de localización en interiores. El objetivo final es dar al usuario una aplicación móvil capaz de hallar su posición en un área cerrada como los edificios o túneles, similar a la aplicación de Google llamada Google Maps que es capaz de localizar a un sujeto en cualquier zona abierta como carreteras, calles, senderos, etc.

Al igual que en el Trabajo de fin de Grado se hara uso de la tecnología Bluetooth, ya que esta está altamente desarrollada y todos los teléfonos móviles actuales la integran en su sistema. De esta forma cualquier usuario podría acceder a esta aplicación con su propio Smartphone. Para ello se han planteado los siguientes objetivos:

- Estudiar el sistema de localización del TFG anterior, analizar sus resultados, los posibles fallos y explorar posibles correcciones.

- Partiendo del algoritmo anterior implementar las mejoras estudiadas de tal forma que suplan las carencias de este, logrando obtener la localización del dispositivo con mucha más precisión.
- Comprobar la viabilidad del método sugerido, mediante pruebas reales del sistema comparando los resultados entre el sistema anterior y el sistema mejorado.

Todos estos objetivos, al igual que en el TFG anterior, se intentarán conseguir con el menor coste posible y la mayor sencillez utilizando elementos fáciles de obtener e implementar en un escenario real, con el fin de que la implementación de este método sea posible de llevar a cabo en un futuro próximo.

1.3 Organización del documento

En cuanto a la organización de este documento, en el segundo capítulo se explicarán los **Conceptos Teóricos** en los que se basa el sistema de posicionamiento en interiores que se ha desarrollado. Tales conceptos serian: especificaciones, protocolos, formulas y métodos matemáticos.

A continuación, en el tercer capítulo de este trabajo, se definirá los elementos que componen el **Entorno de Desarrollo,** diferenciando entre los elementos físicos y el software utilizado, resaltando las funcionalidades y aspectos de estos que sean más interesantes para el desarrollo de este trabajo.

Posteriormente en el capítulo cuarto, **Algoritmo de Localización: Descripción y Análisis** se explica el funcionamiento del algoritmo de posicionamiento desarrollado en el TFG, y se han expuesto sus carencias y limitaciones, viendo así cuales son las áreas donde se debe mejorar.

En el quinto capítulo, **Mejora de Algoritmo de Localización en Interiores,** una vez sentadas las bases, habiendo explicado los conceptos, elementos utilizados y explicado el algoritmo en el que se basa este nuevo método de localización en interiores, se pasara a explicar dicho sistema mejorado en detalle**,** exponiendo los escenarios y la distribución de las balizas elegidas, el desarrollo del algoritmo, su implementación, y un análisis de los resultaos obtenidos comparando estos con los resultados que se obtendrían con el algoritmo antiguo del TFG.

Por último, en el sexto capitulo se hablar sobre las conclusiones que se han obtenido de este Trabajo de Fin de Grado, así como de las posibles vías de trabajo de cara al futuro.

2 CONCEPTOS TEORICOS

En este segundo capítulo se expondrán los principales conceptos teóricos y modelos matemáticos de necesaria comprensión para el desarrollo de la solución propuesta en este trabajo al problema de posicionamiento en interiores.

2.1 Localización en exteriores vs interiores

Se puede apreciar una notable similitud entre la evolución que están teniendo los sistemas de posicionamiento global en los últimos tiempos y el fenómeno que surgió al aparecer la telefonía móvil. Años atrás apenas existían, sin embargo, hoy en día no podemos vivir sin ellos. Ha llegado a tal punto que podemos decir que las tecnologías de geolocalización han motivado a desarrollar infinidad de aplicaciones y en muchos sectores el uso de estas llega a ser imprescindible.

Las tecnologías de posicionamiento global más populares y establecidas en nuestra sociedad son las basadas en satélites [4]. Su funcionamiento trata de recibir la señal de tres o más satélites, obteniendo el tiempo en que tarda la señal se puede saber la distancia entre el receptor y los satélites, por último, se aplica el método de trilateración para obtener la posición. Existen diferentes sistemas lanzados por diferentes países. La antigua Unión Soviética lanzo los primeros satélites de su sistema de posicionamiento global llamado GLONASS en 1982. En el año 2000 la República Popular China ponía en marcha su sistema de navegación BeiDou con el lanzamiento de su primer satélite. La Unión Europea por su parte ponía en marcha el sistema Galileo en el año 2003. Sin embargo, el sistema de navegación más popular y establecido en la actualidad es el Navstar-GPS (conocido simplemente como GPS) desarrollado por Estados Unidos en el año 1973.

Como podemos ver, los sistemas de posicionamiento global basados en satélites están ampliamente extendidos e implementados tanto a nivel militar y comercial (operaciones de rescate, estudios medio ambientales, topografía, agricultura de precisión, transporte de mercancías y personas) como a nivel usuario de a pie (cualquier ciudadano puede hacer uso de esta tecnología a través de su dispositivo móvil).

Estos sistemas son altamente eficaces para localizar personas u objetos que se encuentran en el exterior (carreteras, calles peatonales, senderos, etc). Sin embargo, si el dispositivo GPS se encuentra en interiores (en el interior de un edificio, puente, o cualquier sitio en el que entre este y los satélites se interponga algún obstáculo) estos sistemas no podrán obtener la posición del objeto correctamente. Esto se debe a que la señal recibida de estos sistemas se enfrenta al problema de la atenuación y multirayecto debido a los rebotes de la señal. No obstante, la sociedad está demandando cada vez más la necesidad de conocer la posición y estar localizados también en interiores (como en aeropuertos, centros comerciales, museos, edificios administrativos, hospitales y centros de la tercera edad).

Por esta razón, hoy en día se están desarrollando diferentes soluciones de posicionamiento en interiores (IPS) que permiten localizarnos dentro de entornos cerrados. Al contrario de los sistemas de posicionamiento en exteriores, existen muchas tecnologías diferentes para el posicionamiento indoor que además no suelen ser compatibles entre sí, lo que dificulta su difusión y adopción por parte del gran público. Actualmente, existen sistemas muy precisos y fiables en entornos empresariales, pero estos son muy caros, específicos y difícilmente escalables. Por este motivo en este trabajo se intenta llevar a cabo una solución barata, precisa y de fácil adaptación para la persona de a pie, la cual solo necesitara de su dispositivo móvil, al igual que con el GPS, para localizarse en interiores.

A continuación, se listarán las soluciones implementadas actualmente, que hacen uso de dispositivos móviles al igual que la solución propuesta en este trabajo. Si se quiere conocer mas en detalle la funcionalidad de estas propuestas, en el TFG anterior se puede consultar esta información [1].

- Códigos QR / NFC / NFT
- Mapeado de redes WiFi
- Mapeado de redes móviles
- Navegación inercial
- Campo magnético local
- Beacons

La propuesta de la cual se basa este Trabajo, la que fue desarrollada en el TFG, se trata de la última solución listada que hace uso de los Beacons. Se trata de desplegar varias balizas bluetooth en un escenario, como podría ser una habitación, un túnel, un hangar etc. Las balizas, compuestas por módulos BT, envían beacons los cuales son capturados por un dispositivo móvil. Posteriormente estos datos son mandados a un ordenador con el programa MATLAB el cual ejecutara el algoritmo diseñado para esta solución. Este algoritmo leerá los valores RSSI (Received Signal Strength Indicator) de los beacons para determinar la posición. Basándonos en la lectura de los beacons utilizada en esta solución, se mejorará dicho algoritmo para obtener mayor precisión en los resultados.

2.2 Tecnología Bluetooth

Esta tecnología es la base de este trabajo. Las balizas y dispositivos móviles utilizados en esta solución se comunicarán entre ellos usando bluetooth. A continuación, se expondrán los fundamentos de esta tecnología.

2.2.1 Introducción de la tecnología Bluetooth.

La Tecnología Bluetooth fue desarrollada por Jaap Haartsen y Mattisson Sven en 1994, con la idea de remplazar las conexiones de cable, mientras trabajan en la empresa Ericsson. BT fue basada en la tecnología de saltos de frecuencia de amplio espectro. En los comienzos de esta tecnología, Bluetooth llegaba a alcanzar una capacidad de 720 kbs, sin duda era una gran hazaña para aquella época, sin embargo, esta tecnología ha

ido evolucionando con el paso del tiempo llegando a alcanzar velocidades de más de 50 Mbs. Otro gran progreso que ha conseguido Bluetooh con el paso de los años ha sido la de ampliar su rango de conexión, pasando de rangos menores a un metro a rangos de hasta 100 metros.

El Bluetooth Special Interest Group (SIG) [5] es una asociación sin ánimo de lucro cuya sede se sitúa en Bellevue, Washington y que velan por el desarrollo y adopción de BT. El SIG publico oficialmente las prestaciones de esta tecnología el día 20 de mayo de 1998. Fue fundado inicialmente por Ericsson, Intel Corporation, Nokia, Toshiba e IBM, posteriormente se unieron muchas otras compañías a este consorcio. Hoy en día cuenta con más de 20.000 empresas asociadas en todo el mundo. Todas las versiones de los estándares de Bluetooth están diseñadas para la retro compatibilidad, lo que permite que la última versión cubra todos los estándares anteriores, pudiendo así conectarse un dispositivo de versión 4 con uno de versión 3, 2 o 1.

2.2.2 Versiones

En este apartado se hablará de las mejoras que ha implementado el estándar bluetooth a lo largo de los años, desde la versión 1 hasta la versión 5 [6].

- **Versión 1:** la versión 1.0 de Bluetooth tuvo muchos problemas en sus inicios. Los fabricantes tuvieron grandes dificultades para hacer que sus productos fueran interoperables entre sí. Además, esta versión obligaba a que la dirección del dispositivo BT (BD_ADDR) se incluyera en el hardware, lo que hacía imposible el anonimato a nivel de protocolo. En las versiones 1.1 y 1.2 se solucionaron muchos de estos problemas y se consiguieron varias innovaciones, entre otras: se consiguió compatibilidad con USB 1.1, mayor velocidad de transmisión llegando hasta 721 kbit/s, y se introdujo el indicador de señal recibida o RSSI, muy importante para el desarrollo de este trabajo, el cual se expondrá en los siguientes apartados.

- **Versión 2:** Esta versión fue lanzada el año 2004 y es compatible con versiones anteriores. Bluetooth v2.0 introdujo la tecnología EDR (Enhanced Data Rate), se trata de una tasa de datos mejorada la cual conseguía acelerar la transferencia de datos llegando a velocidades de 3 Mbits/s. Esta nueva tecnología hacia uso de dos tipos de modulaciones: modulación por desplazamiento de frecuencia gaussiana o GFSK, y modulación por desplazamiento de fase o PSK. Además, se consiguió bajar el consumo de energía debido a funcionar con un ciclo de trabajo reducido. La llegada de la versión 2.1 introdujo la tecnología Secure Simple Pairing (SSP), obteniendo de esta forma una mejor experiencia a la hora de emparejar dispositivos BT a la vez que se lograba una mayor seguridad.

- **Versión 3:** Este nuevo estándar fue aprobado el 21 de abril del 2009 por el Bluetooth SIG. La versión 3.0 trajo consigo la nueva tecnología llamada HS (High Speed) la cual incrementaba notoriamente la velocidad de transmisión de datos llegando hasta 24 Mbit/s. Su principal innovación fue la incorporación de AMP

(alternativas MAC y PHY) para el transporte de datos de perfil Bluetooth. De esta forma, la radio Bluetooth se usa tanto para la detección de dispositivos, como la conexión inicial y configuración del perfil, sin embargo, para el envió de grandes cantidades de datos, se hace uso de PHY MAC 802.11 (asociados con Wi-Fi) para la transferencia de datos. Por lo tanto, Bluetooth utiliza su modo de baja energía de conexión cuando el sistema está inactivo, y la radio 802.11 cuando se necesitan enviar grandes cantidades de datos.

- **Versión 4:** La versión 4.0 de Bluetooth fue lanzada el 30 de junio de 2010, incluyendo al BT clásico, el BT de alta velocidad y los protocolos de BT de bajo consumo. El BT de alta velocidad fue basado en la tecnología Wi-Fi, mientras que el Bluetooth clásico consta de protocolos BT preexistentes. Este nuevo estándar introdujo la tecnología BLE (Bluetooth Low Energy) la cual incorporaba una nueva pila de protocolo para el rápido desarrollo de enlaces sencillos. Esta nueva tecnología está dirigida a aplicaciones de muy baja potencia y consumo, alimentados con una pila de botón.

  En el desarrollo de la solución al problema de posicionamiento de este trabajo se hará uso de balizas BT con esta versión, ya que son las más baratas, tiene bajo consumo, y fue también la versión elegida en el trabajo del TFG el cual se comparará con los resultados que se obtengan en la nueva solución propuesta.

- **Versión 5:** EL SIG público el estándar Bluetooth 5.0 el año 2016. Esta versión prometía dar el doble de velocidad, mayor fiabilidad y ampliaba el rango de coberturas. No obstante, el estándar que promete dar el salto más grande en cuanto a obtener una solución para el problema de localización en interiores, es el Bluetooth 5.1 anunciado en 2019. Esta versión incorpora una tecnología que permite saber el Angulo de entrada de la señal recibida, y de esta forma se puede saber la dirección del dispositivo a localizar.

  Sin embargo, como se ha dicho antes en este proyecto se ha utilizado la versión 4.0 ya que el estándar 5.1 acaba de publicarse y apenas hay dispositivos que implementen esta tecnología y los que la implementan son sustancialmente más caros. Pero sin duda, bluetooth 5.1 es una vía prometedora para investigar en el futuro para solucionar este problema.

2.2.3 Topología Bluetooth.

Bluetooth trabaja utilizando como base, el modelo jerárquico de maestro esclavo:

- **Maestro**: El maestro es el que establece la secuencia de frecuencias a utilizar, el que controlar la conexión y el intercambio de información.

- **Esclavo**: La funcionalidad de los esclavos es la de sincronizarse en tiempo y frecuencia con el maestro siguiendo la secuencia de salto establecida por éste, y transmitiendo información cuando el maestro lo ordene.

A continuación, se expondrán las dos posibles topologías que tiene Bluetooth [7]:

- **Piconet**: Esta topología pude ser punto a punto, o punto a multipunto. Las piconets pueden alcanzar un máximo de 7 conexiones. Estas conexiones se producen cuando los dispositivos se encuentran dentro de sus radios de cobertura unos de otros. Los dispositivos que están dentro de la piconet pueden intercambiar los roles maestro-esclavo. No obstante, sólo puede haber un maestro al mismo tiempo dentro de una misma piconet.

- **Scatternet**: La Scatternet se forma cuando dos o más piconets están parcial o completamente solapadas. Para crear una scatternet debe de haber un dispositivo que pertenezca a dos piconets distintas al mismo tiempo. Para esto suceda pueden darse los siguientes casos: que el dispositivo sea esclavo de las dos piconets o que sea esclavo de una piconet y maestro de la otra.

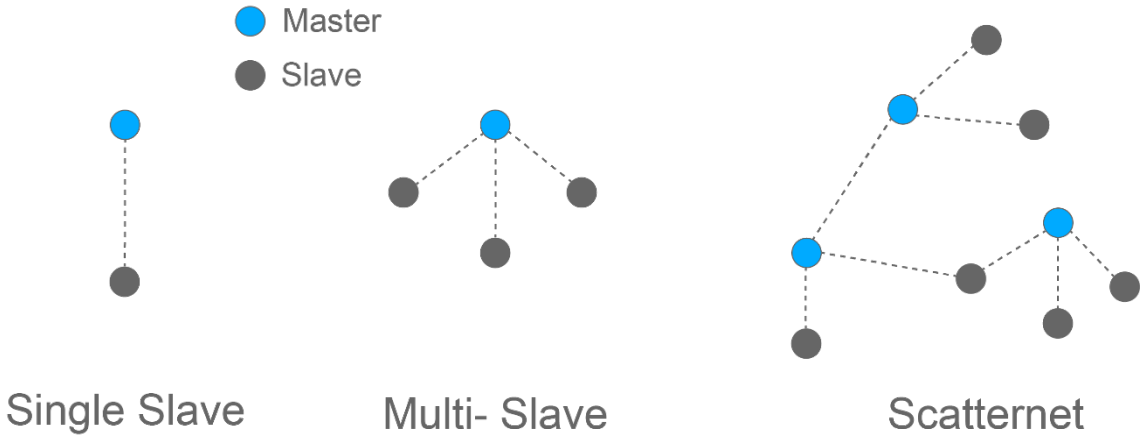


Figura 1 - *Estructura de Piconet y Scatternet.*

No obstante, en este como en el anterior trabajo del TFG no es necesaria la utilidad de estas topologías, ya que el sistema de balizamiento utilizado no necesita establecer conexiones entre dispositivos. Lo único que se transmite y recibe son los beacons enviados por los módulos BT. A pesar de ello, es posible que en el futuro sea interesante emplear este tipo de topologías para, por ejemplo, extraer información específica almacenada en las balizas u otros módulos BT, o emplear este módulo como pasarela de acceso a otro tipo de información de interés.

2.2.4 Conexiones Bluetooth.

A continuación, se expondrá los diferentes estados que atraviesan los dispositivos bluetooth a la hora de establecer una conexión [8]:

- **Standby:** En este modo los dispositivos de una "piconet" no han establecido una conexión aún. Estos se encuentran escuchando cada 1,28 segundos, sobre 32 saltos de frecuencias por si hay algún mensaje entrante.

- **Page/Inquiry:** Si un dispositivo quiere realizar una conexión con otro, éste le envíara un mensaje de tipo page, si conoce la dirección del otro dispositivo; si no la conoce enviara una petición de conexión a través de un mensaje de tipo Inquiry. El dispositivo con el rol maestro envía 16 mensajes tipo page idénticos, en 16 saltos de frecuencias, a la unidad con rol esclavo. Si no el maestro no recibe respuesta vuelve a retransmitir en los siguientes 16 saltos de frecuencia. Los mensajes de tipo inquiry requiere una respuesta extra por parte del dispositivo esclavo, desde la dirección MAC, desconocida por el maestro.

- **Active:** En este modo es cuando se lleva a cabo la transmisión de los datos.

- **Hold:** Cuando se requiera, o cualquiera de los dos dispositivos quiera, se puede establecer este modo en el cual no se transmite información, de esta forma se ahora energía.

- **Sniff:** Este modo solo puede llevarse a cabo por el dispositivo esclavo. Su finalidad como el del modo Hold es la de ahorrar energía. Durante este modo el esclavo toma un rol pasivo en el cual se encontrará en un modo escucha a un nivel de potencia reducido.

- **Park:** Este modo está a un nivel aún más reducido, que el modo hold. En este modo el esclavo se sincroniza a la "piconet", por eso no requiere una reactivación completa, pero no es parte del tráfico. En modo Park, los dispositivos no tienen direcciones MAC y solo escuchan para mantener su sincronización con el dispositivo maestro y revisar los mensajes broadcast de este.

En la imagen siguiente se muestra un croquis de los distintos modos en los que pueden encontrarse los dispositivos BT y sus posibles cambios a diferentes estados dependiendo del estado en que se encuentren.

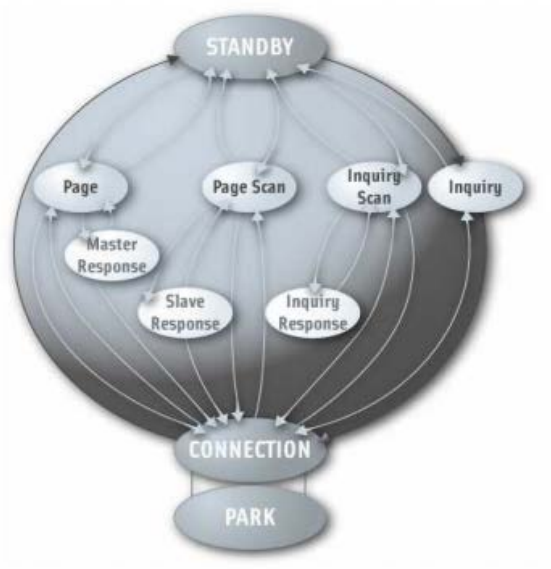


Figura 2 - *Diagrama de estados del controlador de enlace Bluetooth.*

2.3 Distancia a partir del RSSI

En el apartado anterior se expuso que el estándar Bluetooth en su versión 1.1 incorporaba un indicador de potencia de salida o RSSI por sus siglas en inglés (Received Signal Strength Indicator). El RSSI es una escala de referencia (con relación a 1 mW) con el cual se mide el nivel de potencia de las señales recibidas por un dispositivo inalámbrico (WIFI, Bluetooth o Zigbee). Este parámetro es un pilar fundamental para el desarrollo de este trabajo. Los beacons que transmitirán las balizas BT utilizadas en esta solución incluirán el indicador RSSI. De esta forma el usuario recibirá los RSSI de las balizas sabiendo la potencia recibida de cada una de ellas y posteriormente en Matlab se aplicarán las fórmulas que se expondrán a continuación para estimar la distancia entre las balizas y el usuario a partir de este parámetro.

En la formula siguiente podemos ver la relación que tiene el RSSI de una baliza que transmite beacons con la distancia entre esta y el dispositivo receptor del beacon [1]:

$$RSSI_i[dBm] = RSSI_0[dBm] - 10n\log_{10}(\frac{d_i}{d_0}) \quad (2.1)$$

- $RSSI_i$: Indicador de potencia recibida estimado en *dBm*.
- $RSSI_0$: RSSI tomado a una distancia de referencia d_0 y expresado en *dBm*.
- n: Path loss o constante de pérdida de ruta, su valor típico esta entre 2 y 4.
- d_i: distancia entre los módulos.
- d_0: distancia de referencia, típicamente 1 metro.

n hace referencia a la constante de pérdidas de camino, dependiente del medio y el entorno en el que se encuentran los dispositivos (si hay más o menos obstáculos, por ejemplo). El path loss en espacio libre es *n=2*, no obstante, en la localización en interiores a menudo nos encontramos rodeado de obstáculos y paredes por lo que la propagación de la señal se ve afectada por fenómenos de reflexión, difracción y dispersión, por lo que su valor será diferente en un entorno u otro.

Para simplificar la formula toma el $RSSI_0$ de referencia a un metro por lo que d_0 será igual a *1*, despejando n de la siguiente manera:

$$n = \frac{RSSI_0 - RSSI_i}{10\log_{10} d_i} \quad (2.2)$$

Para obtener la distancia a través de un valor $RSSI_i$ se debe despejar la variable d_i de la formula anterior, obteniendo:

$$d_i = 10^{\frac{RSSI_0 - RSSI_i}{10n}} \quad (2.3)$$

Si el $RSSI_0$ de referencia se encuentra a una distancia distinta a un metro la formula seria la siguiente:

$$n = \frac{RSSI_0 - RSSI_i}{10\log_{10}(\frac{d_i}{d_0})} \quad (2.4)$$

La fórmula para la obtención de la distancia d_i a partir de la formula 2.4 seria la siguiente:

$$d_i = 10^{\frac{RSSI_0 - RSSI_i}{10n} + \log_{10} d_0} \quad (2.5)$$

La distancia entre el dispositivo móvil y las balizas es imprescindible para poder estimar la posición del usuario en el escenario. Una vez halladas las distancias se precisa hacer uso del método de trilateración, expuesto a continuación, para obtener la estimación final del usuario.

2.4 Cálculo de la posición: Trilateración

La trilateración es un método matemático utilizado para hallar las posiciones relativas de objetos usando la geometría de esferas, círculos o triángulos. Por otra parte, la triangulación tiene el mismo objetivo que la trilateración, pero hace uso de los ángulos para estimar la posición. En esta solución, como se ha visto en el apartado anterior, utiliza el RSSI para obtener la distancia entre el usuario y las balizas. El Bluetooth v4.0 no permite conocer los ángulos de entrada de la señal recibida por lo que, disponiendo solo la distancia, el único meto que se puede usar para hallar la posición es la trilateración. Sin embargo, como veremos más adelante, la versión 5.1 incorpora una nueva tecnología que permite conocer los ángulos de la señal entrante, por lo que en el futuro seria interesante añadir el método de triangulación para obtener la posición de forma más precisa.

A la hora de obtener la posición con el método de trilateración, en un espacio bidimensional es preciso conocer las distancias relativas entre el objetivo y, por lo menos, tres nodos con posiciones conocidas, tal como podemos ver en la siguiente imagen [9]. Por otro lado, en un espacio de tres dimensiones se necesita saber las distancias relativas entre el objetivo a localizar y al menos, 4 nodos. Como podemos ver en las figuras 5 y 6, para un espacio bidimensional se usarán círculos cuya intersección será la posición del objetivo, y en geometría tridimensional se utilizan esferas.

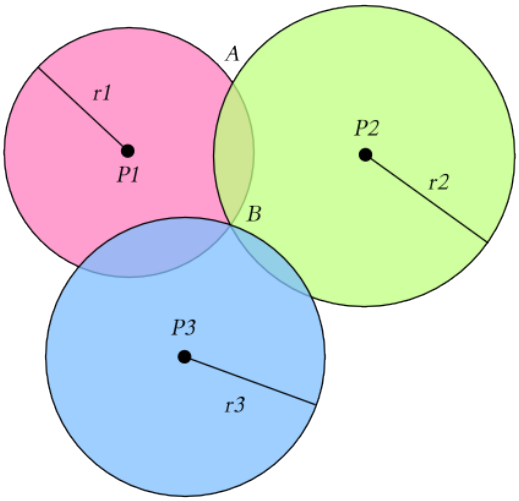


Figura 3 - *Trilateración 2d.*

Por ejemplo, podemos ver el uso del método de trilateración en los Sistemas de GPS. Como podemos ver en la figura siguiente, el método de trilateración tridimensional del que hacen uso estos sistemas precisa de 4 esferas, las cuales podrían ser formadas por 4 satélites y la distancia entre ellos y el usuario o por 3 esferas formadas por 3 satélites igualmente y la cuarta sería el globo terráqueo [9]. La opción más utilizada por estos sistemas es la de usar el planeta tierra como cuarta esfera. Para hallar la distancia entre un dispositivo GPS y los satélites el dispositivo envía señales cronometradas con precisión a los satélites, de esta forma se obtiene la distancia dependiendo del tiempo que tarde el mensaje en llegar. Una vez obtenidas las distancias con tres satélites con posiciones conocidas se formarán tres esferas que tendrán 2 puntos de corte comunes entre ellas. Si a estas 3 esferas se le suma la esfera que forma el globo terráqueo se obtiene un solo punto de corte común a estas 4. Este punto es la posición del dispositivo estimada por el sistema.

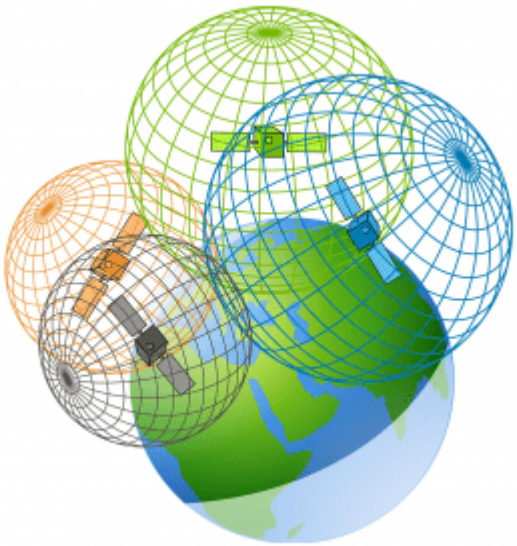


Figura 4 – *Trilateración 3d usada en sistemas de geolocalización GPS.*

Sin embargo, el posicionamiento en interiores tiene el inconveniente de que las señales utilizadas por GPS sufren un elevado desvanecimiento en escenarios sin vista directa con los satélites al tener que atravesar paredes u otros obstáculos, por lo que se usaran balizas BT en vez de satélites en el sistema de localizamiento de este trabajo. Estas balizas estarán a poca distancia del dispositivo a localizar, lo que produce que las esferas definidas por estas balizas sean de un tamaño muy inferior a la esfera descrita por nuestro planeta, por lo que se ha descartado usar esta última esfera para el cálculo de la posición. Por este motivo y debido a que las esferas descritas por las balizas se encuentran en un único plano, se decide usar la trilateración bidimensional en este trabajo.

Como se expuso previamente, el método de trilateración bidimensional utiliza al menos 3 circunferencias (En los cálculos de este trabajo se utilizan justamente 3 al igual que en el TFG [1]) descritas tres nodos cuyas posiciones son conocidas y las distancias relativas entre estos y el objeto a localizar. En la figura siguiente podemos ver cual seria el caso ideal en el que las tres circunferencias tienen un único punto de corte común entre ellas. Este punto corresponde con la estima de la posición del objeto.

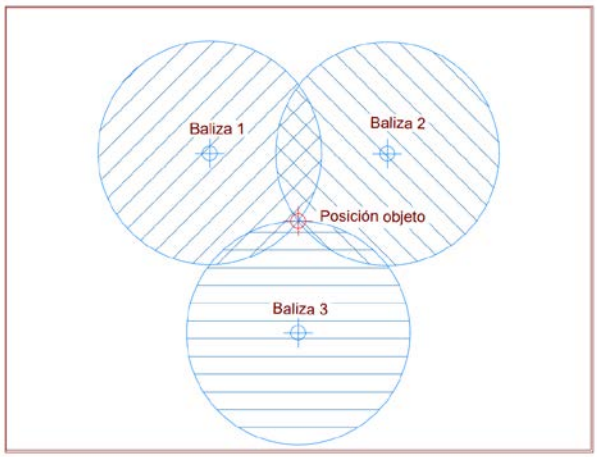


Figura 5 – *Trilateración 2d situación ideal.*

No obstante, después de realizar varias pruebas con el despliegue de balizas se ha observado que el caso anterior es el menos frecuente, siendo los casos típicos los mostrados a continuación:

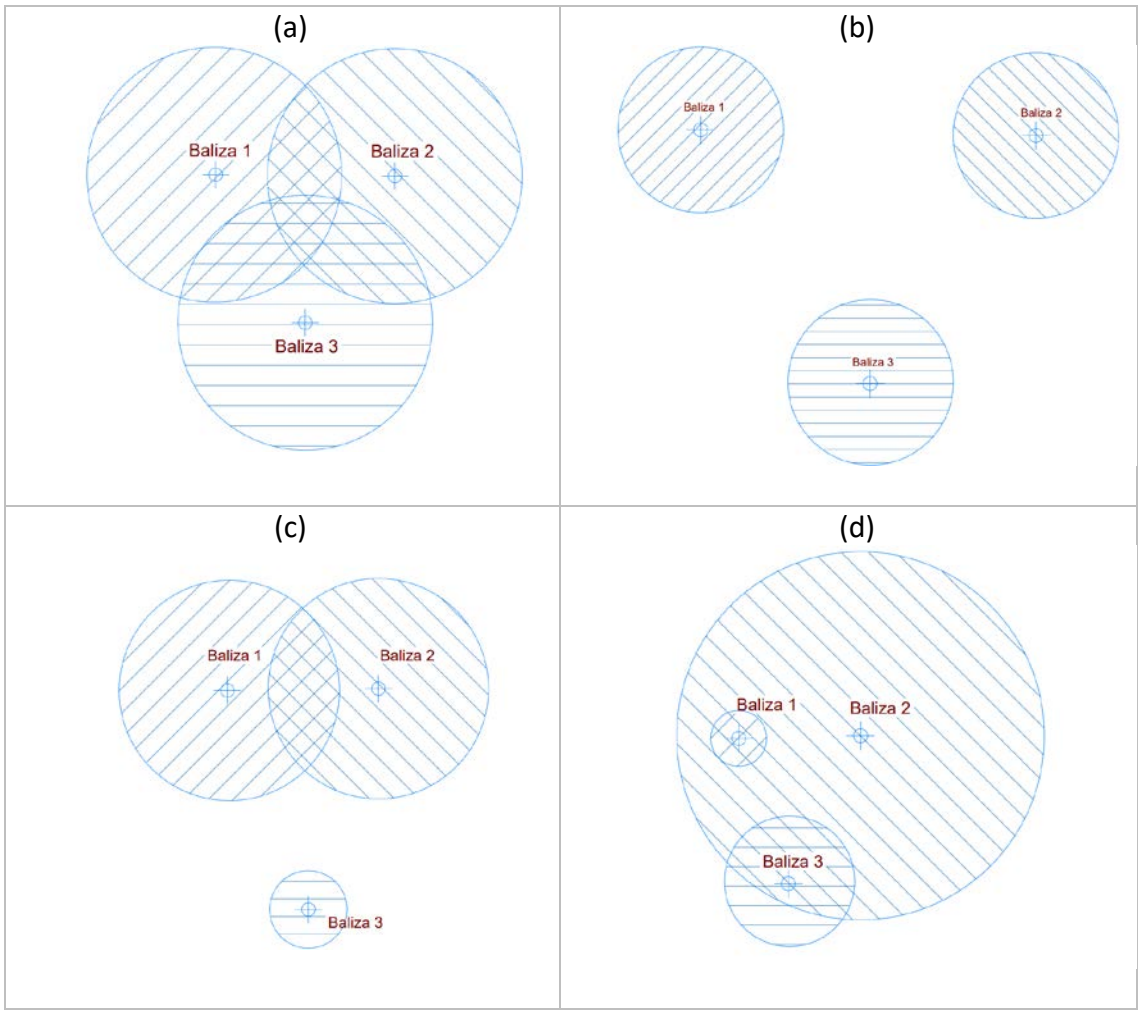


Figura 6 – *Trilateración 2d, circunferencias.*

Si el cálculo de las distancias a partir del RSSI da una estimación de distancia mayor que la longitud real, las circunferencias utilizadas en la trilateración darían 6 puntos de corte como vemos en la figura 8 caso (a). Por otro lado, si la distancia estimada es menor que la real, se obtendría un resultado como el mostrado en el caso (b) donde no hay ningún punto de corte entre las esferas. A parte, de estos extremos se pueden dar situaciones intermedias. En el caso (c) podemos ver como dos de las circunferencias tienen puntos de corte en común y a la vez ninguno en común con la tercera, y en el caso (d) uno de los radios es incluso mayor que la distancia entre los nodos, formando una circunferencia que puede llegar a contener a una o incluso a ambas circunferencias. En el primer caso, como vemos en la imagen, solamente se cortarían dos circunferencias, siendo la solución similar a la del caso(c), mientras que en el segundo caso no existirían puntos de corte entre ellas, lo cual se resolvería de forma similar al caso (b).

Una vez obtenidas las circunferencias se analizan estas de dos en dos, dependiendo de cada caso, y se obtiene 3 puntos [1]. Estos 3 puntos forman un triangulo en cuyo interior se encuentra la localización relativa del objeto. Para hallar estos puntos se ha de seguir los siguientes pasos:

- Si dos circunferencias se cortan en un único punto: Se tomará ese punto.

- Si dos circunferencias se cortan en dos puntos: Se tomará, de estos dos, el más cercano a la tercera circunferencia.
- Si dos circunferencias no se cortan: Se tomará el punto más cercano y equidistante entre ellas.

Si se aplican estos pasos a las circunferencias de los casos a, b, c, d de la figura 8 se obtienen los puntos que se han dibujado en la siguiente figura.

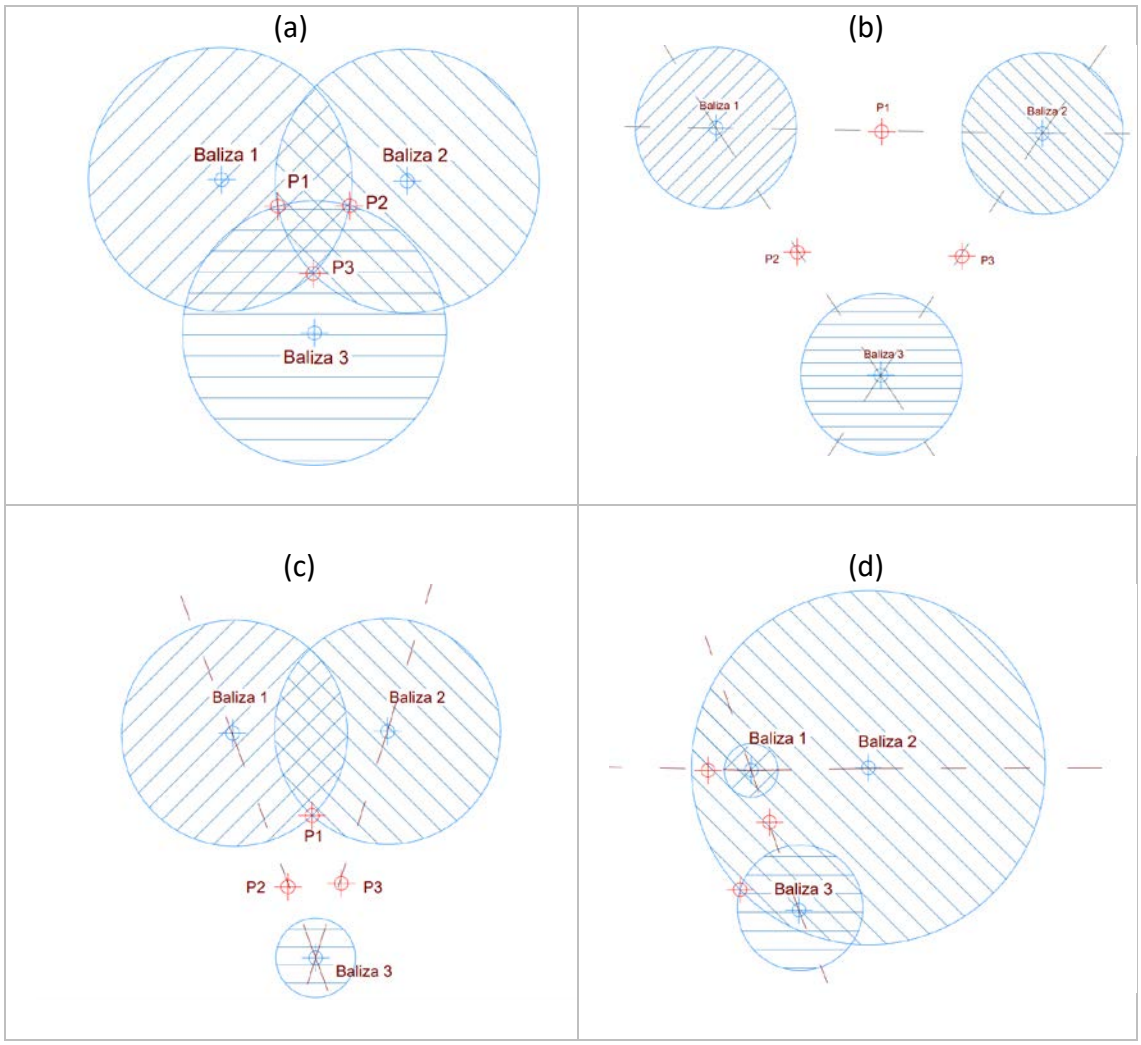


Figura 7 – *Trilateración 2d, puntos del triángulo.*

El último paso para hallar la posición relativa del objeto es el de obtener el centro del triangulo descrito por los tres puntos calculados anteriormente. No obstante, un triángulo tiene hasta cuatro posibles centros: incentro, baricentro, circuncentro y ortocentro [10].

- **INCENTRO:** El incentro es el centro de la circunferencia inscrita al triángulo y tangente a sus lados, por lo que la distancia entre el incentro y cada uno de estos lados es la misma, siendo esta igual al radio de la circunferencia. Otra definición, es que el incentro es el punto de intersección de las bisectrices de cada uno de los ángulos del triángulo (la bisectriz es la recta que divide a un

ángulo en dos ángulos iguales). Para su representación gráfica se requiere dibujar las tres bisectrices y el de corte entre estas será el incentro, tal como podemos ver en la Figura 13.

- **BARICENTRO:** El baricentro o centroide, es el punto de intersección de las medianas de un triángulo (una mediana es el segmento que une un vértice con el punto medio del lado opuesto). De esta forma, para representar gráficamente el baricentro primero se dibujan las tres medianas, y el punto de intersección entre ellas será el baricentro.

- **CIRCUNCENTRO:** El circuncentro es el centro de la circunferencia circunscrita al triángulo, de esta forma la distancia entre el circuncentro y cada uno de los vértices del triángulo es la misma e igual al radio de la circunferencia. También se puede definir como el punto de intersección de las mediatrices del triángulo (una mediatriz es una recta que pasa por el punto medio de un segmento y además es perpendicular a este). De esta forma, para representar gráficamente el circuncentro se dibujan las tres mediatrices del triángulo y el punto de intersección de estas será el circuncentro.

- **ORTOCENTRO:** El ortocentro queda definido como el punto de intersección de las tres alturas del triángulo (una altura es el segmento que parte de un vértice y acaba en el lado opuesto, o la prolongación de este, siendo perpendicular a este). De esta forma, para representar gráficamente el ortocentro de un triángulo se dibujan las tres alturas y se obtiene el punto en el que se intersecan.

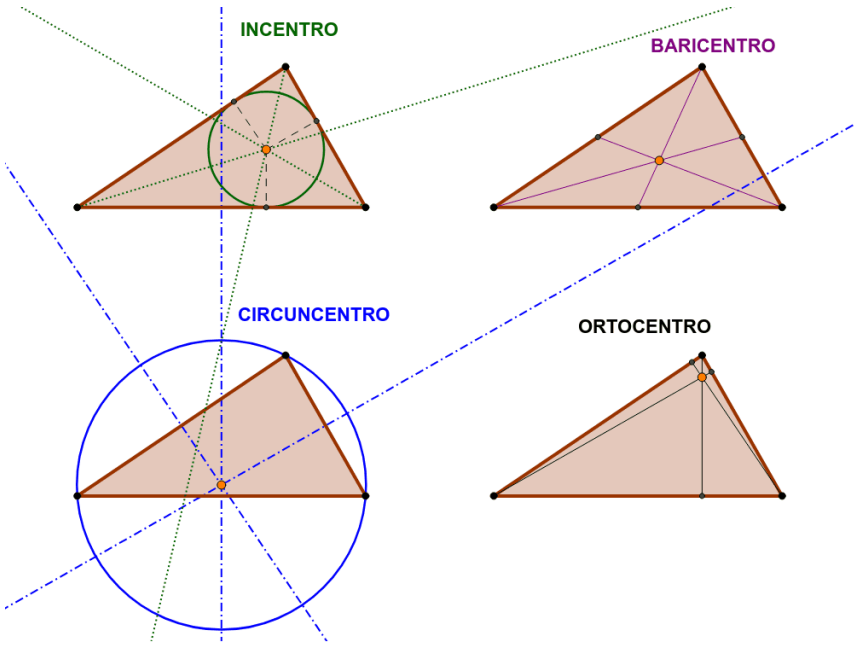


Figura 8 – *Centros de un triángulo.*

Tanto en el cálculo de la posición del trabajo del TFG como en este, se decidió utilizar el Baricentro como centro del triangulo debido a su sencillez de a la hora de programar y computacional con respecto al incentro y a que el baricentro asegura que el centro del triángulo siempre estará dentro de este (al contrario que en el caso del ortocentro y el circuncentro).

2.5 Bluetooth 5.1

He querido hacer un apartado específico para hablar de la nueva tecnología que implementará bluetooth en su versión 5.1 ya que esta será a mi parecer y el de muchos expertos, el futuro de los sistemas de geoposicionamiento en interiores. En este trabajo no se ha hecho uso de ella, de echo nuestras balizas portan la versión 4.0. Ni siquiera se ha utilizado la versión 5.0 ya que esta no aportaba nada nuevo en cuanto al problema de localización en interiores. Sin embargo, las mejoras que han implantado en la versión 5.1, en su gran mayoría van dirigidas justamente a este problema. El problema es que es aún una tecnología nueva, en la cual se acaba de definir el estándar y apenas se pueden encontrar dispositivos que porten esta versión y mucho menos a un costo bajo. Por eso en este trabajo no se ha contado con esta tecnología, pero también se ha llegado a la conclusión de que este nuevo estándar será el futuro y una interesante vía a tener en cuenta para solucionar con más eficacia y precisión el problema de posicionamiento en interiores.

2.5.1 Técnicas de radiogoniometría.

Como se ha visto anteriormente se puede llegar a conocer la distancia entre una baliza con bluetooth y un dispositivo móvil a través del indicador de fuerza de la señal recibida (RSSI) y las fórmulas explicadas en el apartado 2.3. En la versión 5.1 de bluetooth, además de esto, han implementado tecnología Radiogoniometría o RDF (acrónimo del inglés "'Radio Direction Finding") que permite obtener la dirección de la señal recibida. Estos métodos de Radiogoniometría se llaman: Ángulo de llegada (AoA), ángulo de salida (AoD) [11].

- **Ángulo de llegada (AoA):** En AoA, un dispositivo, o etiqueta, transmitirá un paquete de radiogoniometría específico, utilizando solo una antena. El dispositivo receptor, o localizador, tendrá múltiples antenas. La señal entrante de la etiqueta llegará a estas antenas con cambios de tiempo muy leves entre sí. Los cambios en la fase de la señal que se ven en estas antenas se muestrean como componentes IQ de la señal. Luego, con el uso de algoritmos asociados en el dispositivo receptor o localizador, se podrá obtener el Angulo de la señal recibida.

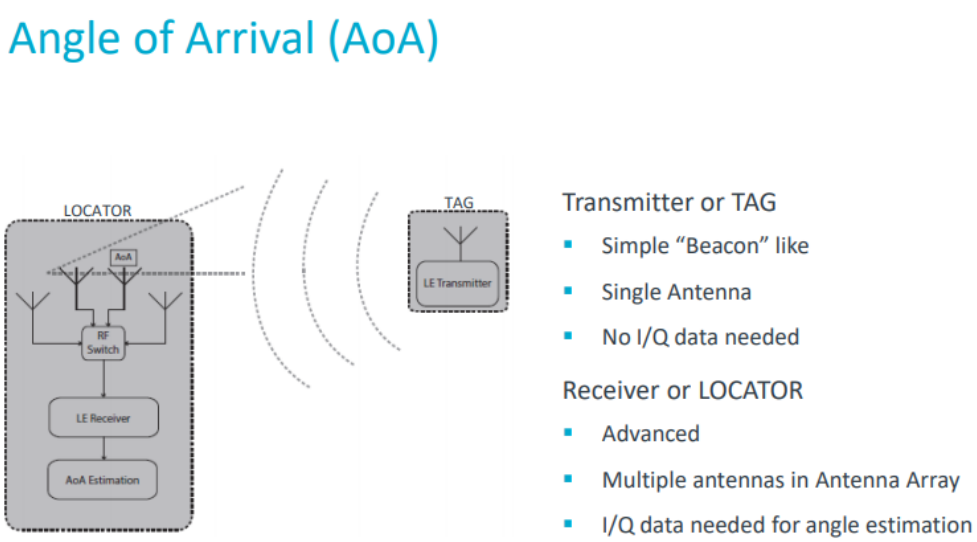


Figura 9 – *Angulo de llegada (AoA).*

- **Ángulo de salida (AoD):** El enfoque de AoD invierte este escenario, y el dispositivo que es clave para determinar la dirección. En este caso será el dispositivo emisor el que tenga múltiples antenas. Esta baliza de localización transmite un paquete a través de del array de antenas. Lo más probable es que el dispositivo receptor en este escenario sea un teléfono móvil, el teléfono realizará un muestreo de IQ en su antena y, desde allí, realizará el cálculo del ángulo. Este segundo método es el que más interesante para este trabajo de los dos, debido a que el AoD permite fijar a los dispositivos emisores en una posición (en el caso de este trabajo serán las balizas) y dejar en libre movimiento al dispositivo receptor (sujeto a localizar con smartphone) que, como se ha dicho antes, será el encargado de procesar los datos.

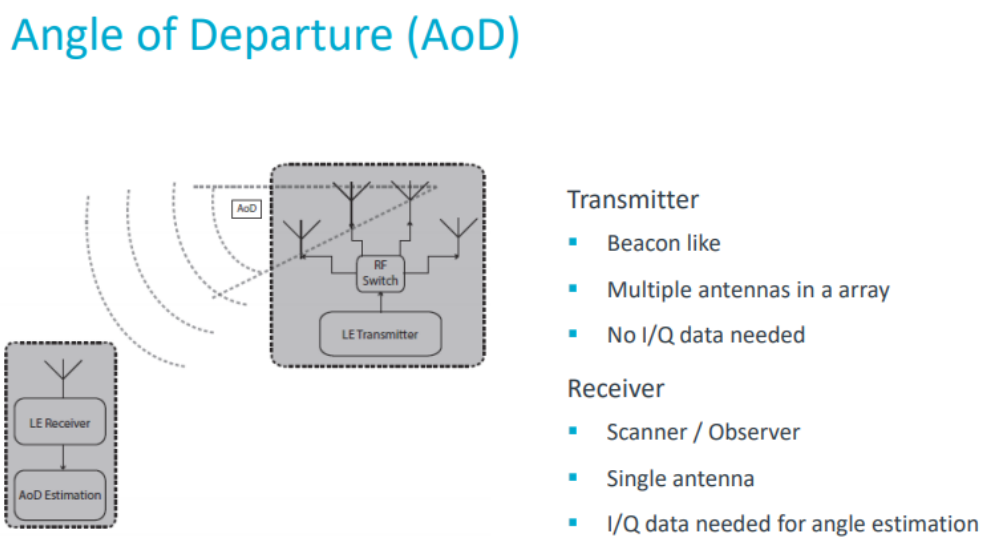


Figura 10 – *Angulo de salida (AoD).*

2.5.2 Actualizaciones de Bluetooth 5.1

Bluetooth 5.1 exige cambios en el protocolo de software de RF (o "pila") y, según el fabricante del chip, algunas mejoras de hardware (radio). Primero, el protocolo revisado agrega una extensión de tono continuo (CTE) a cualquier paquete Bluetooth utilizado para la radiogoniometría [12].

CTE es un tono puro, no modulado, enviado a la frecuencia de la portadora de Bluetooth, más 250 kHz, o incluso cuando se usa el modo de mayor rendimiento, llega a sumar 500 kHz más, y con un periodo entre 16 y 160 µs. El tono consiste en una secuencia “no blanqueada” de '1s' transmitidos el tiempo suficiente para que el receptor extraiga los datos de IQ sin los efectos perturbadores de la modulación. Debido a que la señal CTE se transmite en último lugar, la verificación de redundancia cíclica (CRC) del paquete no se ve afectada.

La segunda adición significativa a la especificación hace que sea mucho más sencillo para el desarrollador configurar el protocolo para realizar el muestreo de IQ. Esta configuración incluye el establecimiento de la sincronización de la muestra y la conmutación de antena, que son fundamentales para estima de la posición se realice con precisión.

Aunque se pueden emplear varias configuraciones de temporización de muestreo de IQ, normalmente se registra una muestra de IQ cada 1 o 2 µs dentro del período de referencia para cada antena. Los resultados se registrarán en la memoria de acceso aleatorio (RAM) del BLE SoC. En la siguiente imagen podemos ver cómo varía la fase de la señal recibida a medida que es muestreada por diferentes antenas de la matriz.

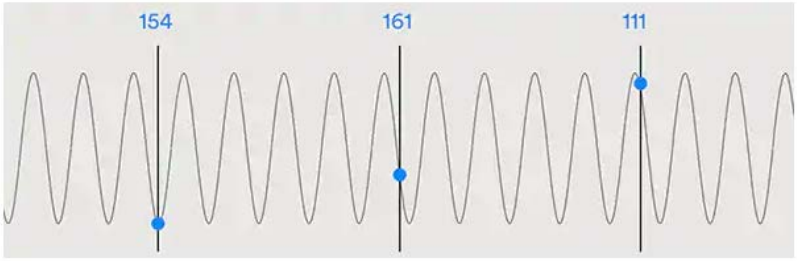


Figura 11 – *Señal recibida por dispositivo BT.*

La grabación de las muestras de IQ es el primer paso para poder logra obtener la posición de un dispositivo con BLE. Los próximos pasos para resolver el problema de localización en interiores serian el de desarrollar una adecuada configuración de redes de antenas para los localizadores y balizas utilizados, y la creación e implementación de un algoritmo que sea capaz de transformar los datos IQ en coordenadas.

2.5.3 Calcular la dirección de la señal

Las matrices de antenas para radiogoniometría se dividen normalmente en tres tipos de matrices: matriz lineal uniforme (ULA), matriz rectangular uniforme (URA) y matriz

circular uniforme (UCA). La primera configuración de antenas forma una matriz lineal unidimensional, mientras que las matrices rectangular y circular son bidimensionales. De esta forma, ULA es la configuración más fácil de diseñar e implementar, pero a su vez es más limitada, ya que al ser unidimensional solo puede calcular el ángulo azimutal. De esta forma, solo puede hallar el ángulo del dispositivo a localizar en un plano horizontal. Las matrices URA y UCA además de poder medir los ángulos de acimutales, como ULA, también pueden medir el ángulo de elevación de elevación. Por esto mismo, estas dos configuraciones de antenas en matrices bidimensionales permiten localizar a otro dispositivo en un entorno de tres dimensiones. En la siguiente imagen podemos ver 3 ejemplos de las configuraciones comentadas anteriormente [12].

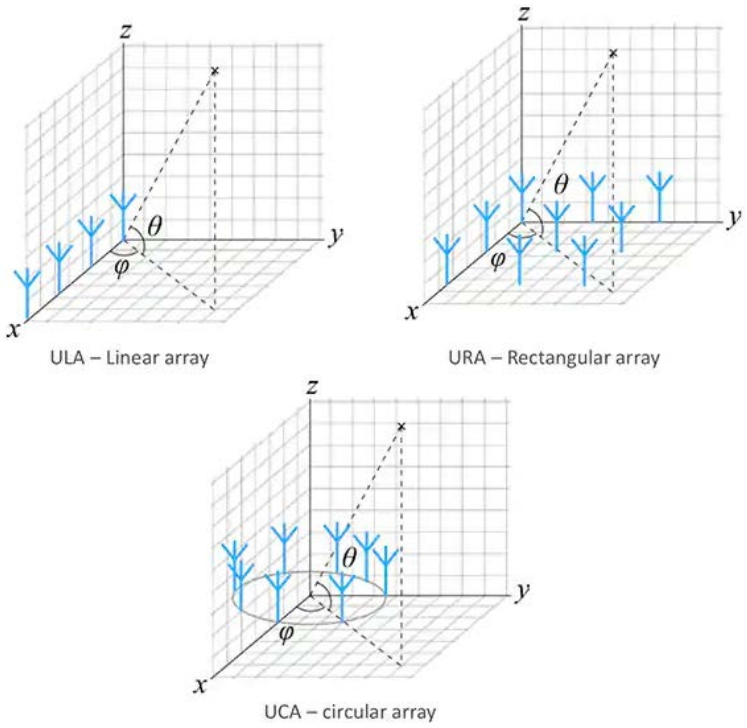


Figura 12 – *Configuración de múltiples antenas en matrices.*

2.5.4 Obtención de la posición

Ya se ha visto que gracias a la colocación de múltiples antenas en una baliza y usando el enfoque AoD se puede obtener el ángulo de recepción de señal con un dispositivo móvil. Ahora, para obtener la posición bastaría con tener dos o más balizas fijas en una posición determinada [13].

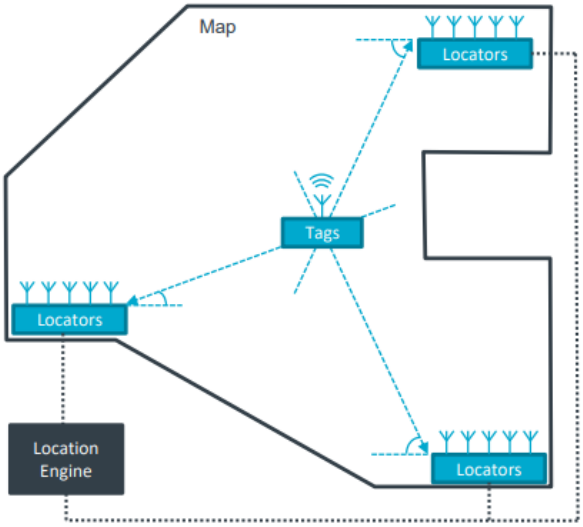


Figura 13 –*Triangulación de balizas BT v5.1.*

En la imagen siguiente podemos ver como con tres balizas y aplicando el método de triangulación, se obtiene la posición del sujeto. Esto, sumado al método que se utiliza en este trabajo, en el cual se calcula la distancia por medio del RSSI, la calibración del path loss, y el cálculo de la trilateración, permiten obtener una estima de la posición mucho más precisa.

2.6 Algoritmo de localización en interiores: Punto de partida

En este punto se va a explicar brevemente el escenario de partida desarrollado en el trabajo anterior [1]. Se describe la solución planteada entonces, ya que el método utilizado en ese caso es la base desde la que se plantea este trabajo.

El objetivo de este trabajo era el de obtener la posición de un usuario en un escenario bidimensional. Se realizaron dos tipos de pruebas: la primera de ellas simulaba la obtención de la posición de un sujeto fijo en cualquier parte del escenario, y la segunda prueba consistía en estimar el camino que recorría un sujeto moviéndose libremente por la estancia.

Las dos pruebas se realizaron en el mismo escenario y con los mismos componentes. Se desplegaron varias balizas por la estancia formando una malla rectangular en el centro de esta. Las balizas estaban compuestas por módulos Bluetooth conectadas a pequeñas baterías de LiPo creando balizas independientes, versátiles y fáciles de desplegar. Las balizas BT están continuamente emitiendo beacons, que son recibidos por un dispositivo móvil y posteriormente son analizados en el programa MATLAB. Aunque en el capítulo 4 se explicará dicho algoritmo con mucho más detalle, de una forma simplificada el algoritmo funciona de la siguiente manera: Primero lee los RSSI de cada baliza capturados por el dispositivo móvil, gracias a las fórmulas de apartado 2.3

se puede obtener la distancia entre el dispositivo y las balizas, y por último se realizara la trilateración con las tres balizas más optimas obteniendo así la posición.

3 ENTORNO DE DESARROLLO

En este capítulo se hará una breve descripción de los elementos y programas usados en el desarrollo e implementación de la mejora en el algoritmo de posicionamiento en interiores propuesto en este TFM.

3.1 Hardware

En el siguiente apartado se describen los principales elementos físicos utilizados en el desarrollo de las pruebas.

3.1.1 Placa Arduino

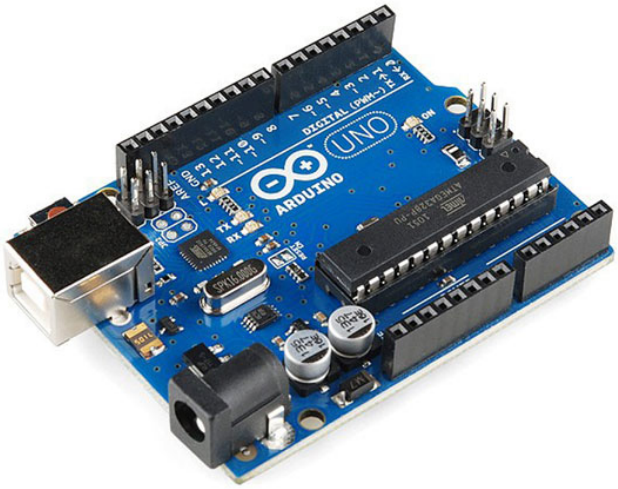


Figura 14 – *Placa controladora Arduino UNO.*

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo presionando un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea. Se puede manejar el controlador Arduino mandándole un conjunto de instrucciones al microcontrolador situado en la placa. Para conseguirlo, se utiliza el lenguaje de programación Arduino (basado en Wiring) y el Software Arduino (IDE), basado en Processing [14].

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto. Sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos.

Se ha elegido esta placa sobre otras para el desarrollo de este trabajo por las siguientes razones:

- **Barato:** Las placas Arduino son relativamente económicas en comparación con otras placas de microcontroladores del mercado. La versión menos costosa del módulo Arduino se puede ensamblar a mano, e incluso los módulos Arduino preensamblados cuestan menos de 50 dólares. La placa utilizada en este trabajo tiene un precio de 20 euros.
- **Multiplataforma:** El software de Arduino (IDE) es compatible con Windows, Macintosh OS X, y Linux. Sin embargo, la gran mayoría de los sistemas de microcontroladores están limitados a Windows.
- **Entorno de programación simple y claro:** el software Arduino (IDE) es fácil de usar para principiantes, pero lo suficientemente flexible para que los usuarios avanzados también lo aprovechen.

Las capacidades de la placa utilizada en este trabajo, denominada Arduino UNO Revisión 3, son las siguientes [14]:

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de entrada/salida (6 salidas PWM12).
- 6 entradas analógicas.
- conexión USB.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

Otra de las características de las placas Arduino es la posibilidad de conectar en las entradas de estos diferentes módulos que aumentan sus capacidades.

La funcionalidad que tiene la placa Arduino en este trabajo es la de configurar adecuadamente los módulos BT, que se usaran como balizas, mediante comandos AT.

3.1.2 MLT-BT05 BLE module

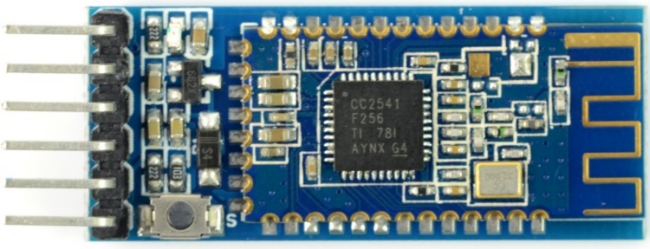


Figura 15 - *Modulo MLT-BT05.*

El éxito arrollador que ha tenido Arduino con sus placas microcontroladores sobre otras compañías ha producido que los desarrolladores hayan decidido crear infinidad de módulos compatibles con este, lo cual aporta a Arduino aumentar su número de funcionalidades. Uno de estos dispositivos de los que estamos hablando es el Módulo MLT-BT05 utilizado en este y en el anterior trabajo como baliza BT. Se ha escogido este módulo por su compatibilidad con Arduino, su sencilla programación y configuración, su bajo coste (apenas 3 euros la unidad) y por su diseño compacto. Todas estas características facilitan un despliegue de balizas barato y sencillo de montar.

El módulo MLT-BT05 es un clon del famoso HM-10. Es un módulo Bluetooth Low Energy (BLE, Bluetooth 4.0 o Bluetooth Smart) basado en los chips de Texas Instruments CC2540 o CC2541. Este módulo permite simular diferentes clases de dispositivos bluetooth gracias a sus amplias posibilidades de configuración. Hace uso de comandos AT para configurar los diferentes parámetros tales como: la potencia de transmisión, el intervalo de envío de los Beacons, el silenciado de los mismos entre otros. Para introducir estos comandos y configurar el dispositivo se debe conectar este a la placa Arduino y está a un ordenador donde tengamos instalado el Software del mismo Arduino. Una vez configurado el módulo necesita de alimentación para poder funcionar. Se pueden conectar a la placa Arduino para suministrarles energía, pero al igual que se hizo en el TFG en este trabajo se optó, debido al bajo consumo de los módulos, por conectarlos a pequeñas baterías de LiPo dotando así a las balizas de independencia. Esto permite montar despliegues versátiles y dinámicos, adaptándose así a las situaciones cambiantes del entorno.

Los comandos AT mas relevantes a la hora de configurar los módulos para su actuación en este trabajo como balizas BT son los siguientes [15]:

- AT: Sirve para comprobar la conectividad del módulo. Si es asi devolverá un OK por pantalla.
- AT+RESET: Este comando reinicia el módulo.
- AT+ADDR?: Devuelve la dirección MAC del módulo MLT-BT05.
- AT+POWE?: Este comando devuelve un numero entre 0 y 3, los cuales representan diferentes niveles de potencia: −23 *dBm* para 0, −6 *dBm* para 1, 0 *dBm* para 2 y por último +6 *dBm* para 3.
- AT+ADVI?: Esta instrucción permite conocer el intervalo de anuncio del dispositivo. En este caso la respuesta está codificada desde 0 hasta *F*, donde el valor mínimo es de 100 *ms* y el más alto de 7 *segundos*.
- AT+ADTY?: Esta instrucción permite averiguar si el dispositivo Bluetooth transmite Beacons o no.

Además de leer los valores actuales del módulo, estos también pueden ser modificados. Bastaría con cambiar la interrogación final de las instrucciones por el valor deseado.

3.1.3 Baterías LiPo

Las baterías LiPo (o batería de polímero de iones de litio) son baterías recargables compuesta en ocasiones de múltiples celdas idénticas montadas en paralelo y usadas en aplicaciones que requieren corrientes superiores a 1A con bajo peso y tamaño reducido, por ejemplo, sistemas de radio control, como aviones, helicópteros, drones, cámaras, celulares, linternas, etc.

Figura 16 - *Batería LiPo de 750mAh.*

Las principales características de las baterías de LiPo son las siguientes [16]:

- **Vida útil:** 2 a 3 años o unas 500 cargas completas.
- **Formas:** son fabricadas en diversas formas y tamaños.
- **Eficiencia:** mejor relación tamaño eficiencia que otras tecnologías.
- **Tasa de descarga:** Alta tasa de descarga desde 1A hasta más de 25A.
- **Voltaje de Celda:** Cada celda tiene 3.7V, y se puede encontrar baterías de 1 a 6 celdas.

En el desarrollo de este trabajo se ha decidido utilizar las mismas baterías que se utilizaron en el TFG anterior para alimentar a los módulos MLT-BT05. Estas serían las baterías de liPo de 750 mAh, como la que se muestra en la imagen anterior. Estas baterías son perfectamente compatibles con los módulos MLT-BT0 y tienen una durabilidad de 5 días transmitiendo beacons continuamente. Aunque en el trabajo anterior ya se dijo que existían baterías en el mercado con mayor amperaje y por tanto mayor durabilidad a bajo coste, no han sido necesario su uso para la realización de estas pruebas. Ciertamente para un despliegue real y definitivo seri muy interesante estudiar la posibilidad de implementar baterías mucho más duraderas, pero en este trabajo se ha centrado en la mejora de la precisión del algoritmo y no en la durabilidad del despliegue.

3.2 Software

Una vez descritos los elementos físicos que se utilizaran en las diferentes pruebas, se exponen los principales softwares utilizados a lo largo de estas.

3.2.1 Software Arduino

La empresa Arduino proporciona un software el cual permite controlar y programar cualquiera de sus placas. El software Arduino (IDE) de código abierto, con su sencillez, facilita la escritura de código y su carga en la placa. Antiguamente Arduino te proporcionaba un archivo ZIP descargable para la instalación de su programa. No obstante, en la actualidad, aparte de la primera opción, Arduino a lanzado su editor web desde el cual se podrá codificar en línea y guardar los proyectos en la nube [14].

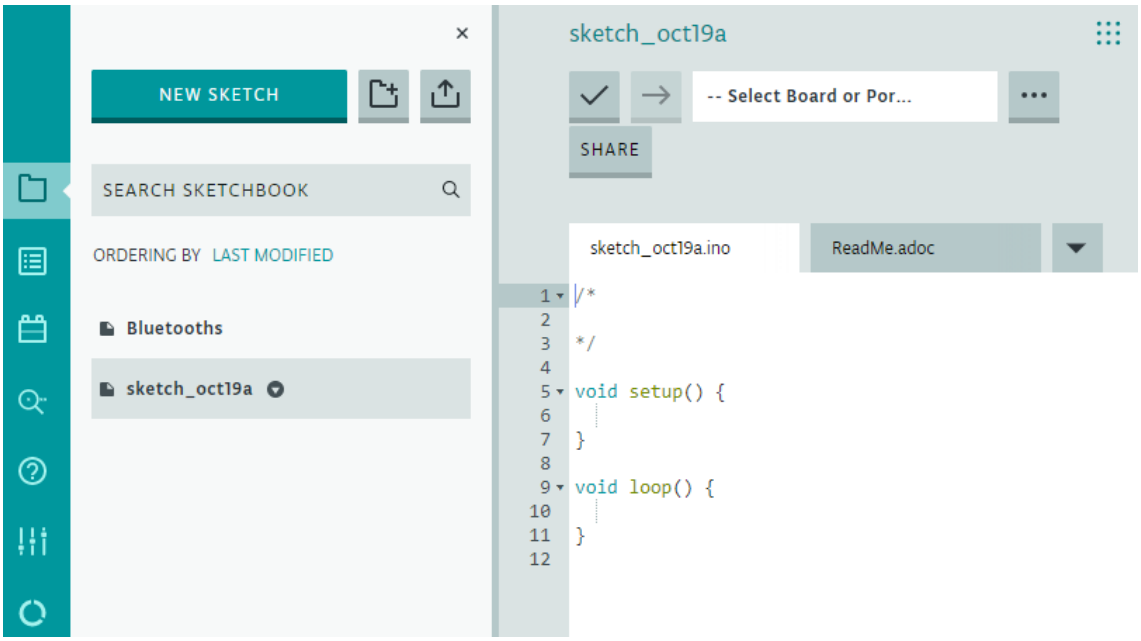


Figura 17 - *Pantalla de inicio Arduino web editor.*

Se programarán los correspondiente comandos AT desde Arduino web editor para configurar los módulos MLT-BT05. Se escribirá un nuevo sketch específico con la configuración pertinente, se conectará el MLT-BT05 a la placa Arduino, y ésta al ordenador por un puesto USB. Finalmente, se cargará el programa en el módulo.

3.2.2 Interfaz móvil

La interfaz que se muestra en la Figura 18, es la aplicación que utiliza el dispositivo móvil para ser localizado por las balizas en los escenarios planteados. Esta interfaz fue utilizada para la toma de medidas en el TFG anterior [1]. De la misma forma, se usará para tomar las medidas que se analizaran con los dos algoritmos, el de este trabajo y el del anterior. De esta manera usando los mismos instrumentos que en las pruebas del

TFG se obtendrá una comparación más eficaz. Esta, es una interfaz muy sencilla, que se encarga de almacenar la potencia que recibe de cada uno de los dispositivos Bluetooths.

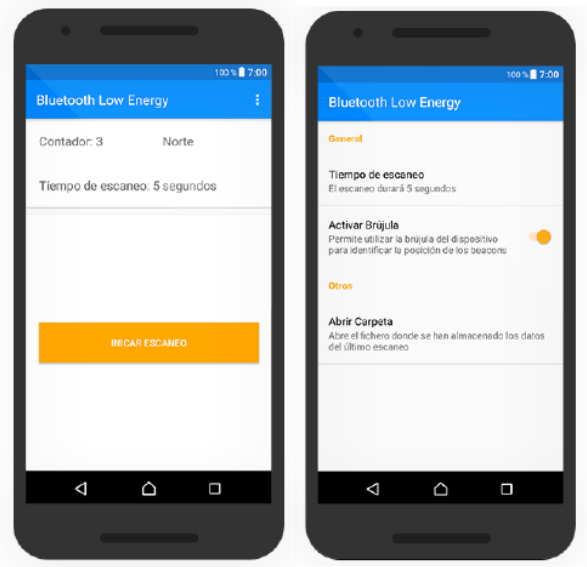


Figura 18 - *Interfaz de la aplicación de medida y posicionado.*

Al pulsar el botón “Iniciar Escaneo” el dispositivo empieza a recibir todos beacons e las balizas del despliegue. De estos beacons extrae los valores de ponencia ordenados cronológicamente y acompañados de la dirección MAC de cada módulo bluetooth, guardando todos estos datos en un fichero *CSV.* Desde la misma aplicación se puede configurar el tiempo de escaneo.

El formato de ficheros CSV es el ideal para procesar datos con programas matemáticos como Matlab, ya que permiten una fácil lectura con simples funciones.

3.2.3 MATLAB

MATLAB (es la abreviatura de laboratorio de matrices en ingles ***MAT****rix* ***LAB****oratory*) es un lenguaje de programación propietario multi-paradigma y un entorno de computación numérica desarrollado por MathWorks. MATLAB permite manipular matrices, trazado de funciones y datos, implementación de algoritmos, creación de interfaces de usuario e interconexión con programas escritos en otros lenguajes. MATLAB es compatible con la mayoría de los sistemas operativos y plataformas como Unix, Windows, Mac OS X y GNU/Linux.

En 2020 MATLAB supero la cifra de 4 millones de usuario en todo el mundo, procedentes de diversos ámbitos de la ingeniería, la ciencia y la economía.

Las tareas más típicas por las cuales se utiliza MATLAB son las siguientes [17]:

- Cálculos numéricos.

- Desarrollo de algoritmos.
- Modelado, simulación y prueba de prototipos.
- Análisis de datos, exploración y visualización.
- Graficación de datos con fines científicos o de ingeniería.
- Desarrollo de aplicaciones que requieran de una interfaz gráfica de usuario (GUI, Graphical User Interface).

Se usa MALTAB en este trabajo ya que permite la lectura de ficheros CSV y su posterior procesado de los datos incluidos en este. De esta manera, al igual que don el trabajo anterior, con MATLAB se procesan los ficheros proporcionados por la interfaz móvil del sujeto y de los dos calibradores, con los datos de las balizas bluetooth. Una vez obtenidos los datos, el algoritmo creado en MATLAB se encargará de realizan los cálculos para determinar la posición del móvil, y representara los resultados gráficamente.

4 ALGORITMO DE LOCALIZACION: Descripción y Análisis inicial

En este Punto número 4 se describe detalladamente la propuesta, que se desarrolló en el TFG anterior, al problema de localización en interiores. A continuación, se exponen sus resultados obtenidos de las pruebas, recalcando de esta forma sus fallos y debilidades.

4.1 Descripción

La propuesta del TFG [1] trata de localizarnos en un espacio de dos dimensiones (ancho y largo) dentro de una estancia. Para ello hace uso de balizas BT distribuidas en forma de malla por el aula. El sujeto a localizar tomara datos de los beacons de las balizas con su móvil. Estos datos son leídos por el algoritmo desde Matlab, el cual dará una posición estimada de la posición. Para ver la precisión y funcionalidad del algoritmo se decide realizar dos tipos de pruebas: Prueba en estático, donde el sujeto a localizar se situará en un punto fijo del aula y se tomaran las medidas durante varios segundos, y la prueba en dinámico, la cual se trata de con una sola medida de los datos de las balizas, mientras el sujeto se mueve por el aula, ser capaces de estimar el camino recorrido por este.

4.1.1 Escenario.

Las dos pruebas, prueba estática y dinámica, se llevaron a cabo en la "Sala de multiusos" del edificio de I+D+i de la Universidad de Cantabria, situada en la planta -2 y con unas dimensiones de unos 135 m2 (9 metros de ancho por 15 de largo) [1].

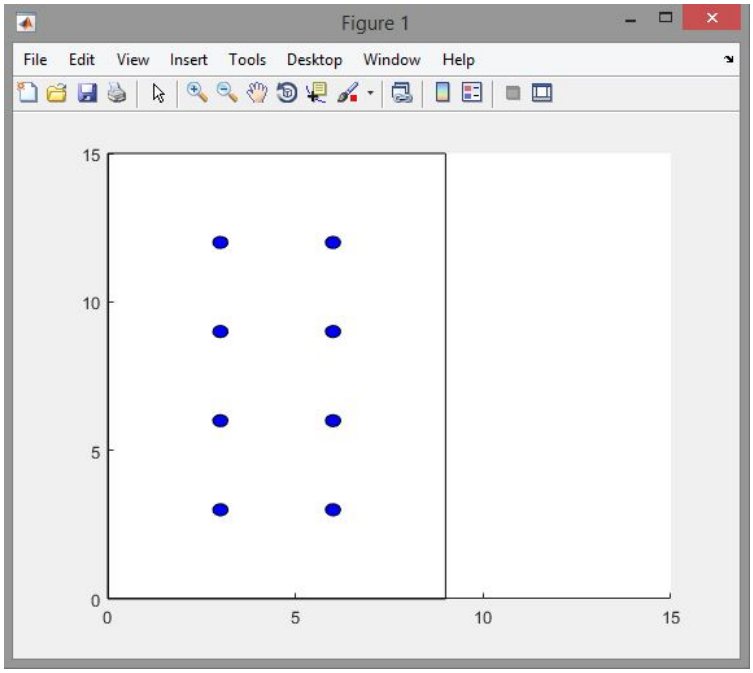


Figura 19 – *Escenario de pruebas del TFG.*

En la imagen anterior podemos ver una representación en Matlab de la distribución de las balizas BT dentro de la sala. Los puntos azules representan a cada una de las 8 balizas colocadas. Estas se encuentran a una distancia de 3 metros entre ellas y otros 3 metros de separación con las paredes.

4.1.2 Calibración previa.

Antes de realizar las pruebas en la sala es necesario configurar las balizas BT adecuadamente y hacer un análisis del path loss que habrá en el aula. Para hallar las pérdidas de camino del aula "n" se colocará un móvil en un punto central de este y se situaran varias balizas a distintas distancias conocidas respecto a este. Se tomarán medidas y se obtendrán los RSSI de las balizas. De esta forma sustituyendo estos valores de RSSI en la fórmula 2.2, descrita en el segundo capítulo de este trabajo, se obtendrá la "n".

Este es uno de los puntos más diferenciadores entre esta solución y la nueva que se propone en este TFM. En el TFG se hace una calibración previa a la realización de la prueba, por lo que solo se obtiene "n" una única vez. Por otro lado, en el TFM se realizan calibraciones en tiempo real, a la vez que se toma cada medida para así saber la "n" de cada instante y obtener así una estimación de la posición más precisa. Este proceso se explica con más detalle en el capítulo quinto.

4.1.3 Algoritmo.

A continuación, se explicará el funcionamiento del algoritmo de posicionamiento del TFG que sirve como base para el nuevo algoritmo desarrollado en este trabajo.

El primer paso que realiza el algoritmo de MATLAB es el de leer y ordenar los datos de los ficheros CSV, recopilados por le móvil, donde se encuentran los datos de RSSI de las balizas. Los datos RSSI van acompañados de la dirección MAC de cada modulo BT. Esto permite al algoritmo distinguir y agrupar los RSSI de las balizas del despliegue. El algoritmo realiza la media de cada grupo de RSSI y los mete en una matriz cuyos índices corresponden con la posición de las balizas en el aula.

El segundo paso es el de seleccionar cuales son las tres balizas que van a realizar el calculo de trilateración. La primera baliza corresponde con la que tenga la señal mas potente, que es lo mismo que decir que el RSSI mas alto. La segunda baliza tiene que estar o en el mismo eje "x" o "y" que la primera, e inmediatamente seguidas sin que haya otra entre medias. El algoritmo elegirá la siguiente mas potente teniendo en cuenta estos requisitos. Por ultimo la tercera baliza tiene que cumplir los mismos requisitos que la segunda, pero añadiendo la restricción de que no puede encontrarse en el mismo eje de "x" ni de "y" que la segunda baliza. Esto asegura que las tres balizas no se encuentren en el mismo eje lo que dificultaría los cálculos de la trilateración.

En el tercer paso el algoritmo calculara la distancia de las tres balizas elegidas al dispositivo móvil. El algoritmo obtiene estas distancias a través de la formula 2.3 de este trabajo que relaciona el RSSI con la distancia.

En el cuarto y ultimo paso el algoritmo aplica la trilateración para obtener la posición. A estas alturas ya ha obtenido las tres balizas mas óptimas para este cálculo, conociendo la posición de estas en el escenario y la distancia entre estas y el usuario. Por lo tanto, ya dispone de 3 es esferas para realizar la trilateración bidimensional. Con esas 3 esferas obtiene el punto de corte común a las 3 obteniendo así la posición. Sin embargo, lo mas probable es que no haya un punto de corte común a las 3, siendo los casos más típicos los expuestos en el apartado 2.4. En este mismo apartado se explica como solucionar estos casos. Se obtiene 3 puntos de corte estudiando las circunferencias dos a dos y se obtiene el baricentro del triangulo formado por dichos puntos obteniendo así la posición.

Esto seria suficiente para tomar medidas en estático, pero si por el contrario se quiere tomar medidas en movimiento por el escenario recorriendo un camino, el algoritmo trocea la información tomada en la medida en movimiento y halla los puntos de cada partición. Posteriormente une estos puntos consecutivamente y en orden cronológico para dibujar el camino estimado.

4.2 Fallos y limitaciones

En la imagen siguiente [1] podemos ver los resultados obtenidos de las pruebas en estático realizadas en el aula. Como se ha dicho antes los puntos azules oscuros representan la posición de las balizas. Ahora en la representación de los resultados en Matlab podemos ver también un punto rojo que representa la posición real del sujeto y un punto azul cian que representa el punto estimado para el algoritmo de posicionamiento. De esta forma la distancia entre estos dos puntos dará el error de precisión.

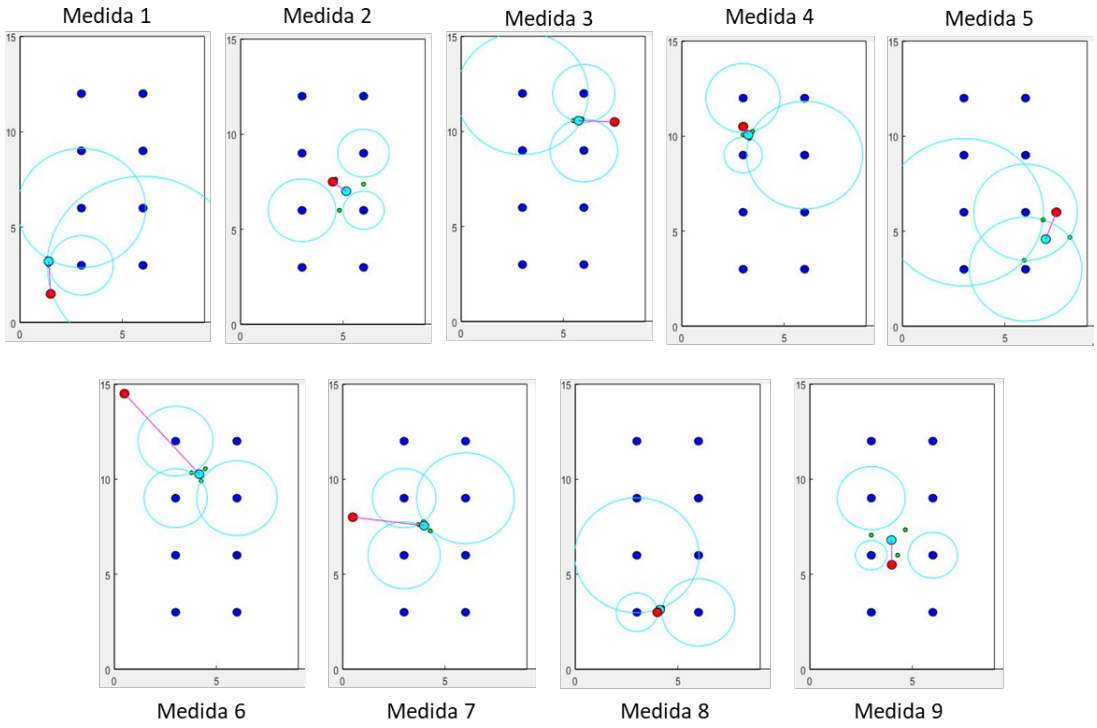


Figura 20 – *Resultados medidas en estático TFG.*

A la vista de los resultados se puede separar el aula en dos zonas, como se aprecia en la imagen siguiente. La zona verde es el espacio que queda en el interior de la malla descrita por las balizas y la zona roja será la zona exterior a esta malla.

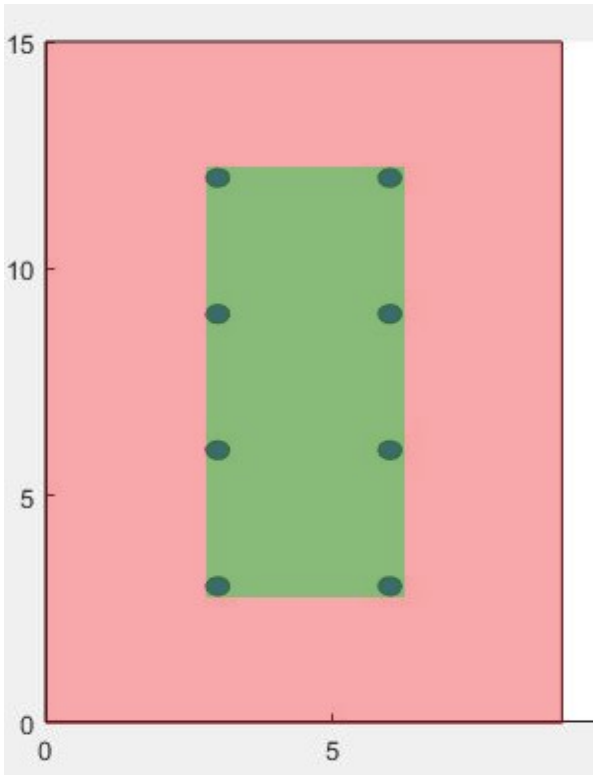


Figura 21 – *División de áreas de localización optimas TFG.*

Como podemos ver en la tabla siguiente el error cometido es considerablemente menor al obtenido en la zona roja, estando el error de la zona verde comprendido entre 0.18 metros (medición 9) y 1.30 metros (medición 8), mientras que el error de la zona roja se sitúa entre 1.51 metros (medición 5) y 5.60 metros (medición 6).

Medición	***Error***	***Zona***
1	1,71m	Rojo
2	0,82m	Verde
3	1,76m	Rojo
4	0,51m	Verde
5	1,51m	Rojo
6	5,60m	Rojo
7	3,51m	Rojo
8	1,30m	Verde
9	0,18m	Verde

Tabla 1 - *Resultados de las medias en estático TFG.*

En la siguiente imagen vemos la representación de los resultados de la prueba 2 (prueba dinámica) [1]. En esta prueba se recorre un camino tomando una única medida de 60 segundos. Posteriormente se mandan esta medida al algoritmo de Matlab, el cual troceara esta medida de 60 segundos en 20 medidas de 3 segundos. El algoritmo ira obteniendo la posición de estas 20 medidas e irá uniendo los puntos consecutivamente de forma ordenada obteniendo así la estela de la estimación del camino recorrido.

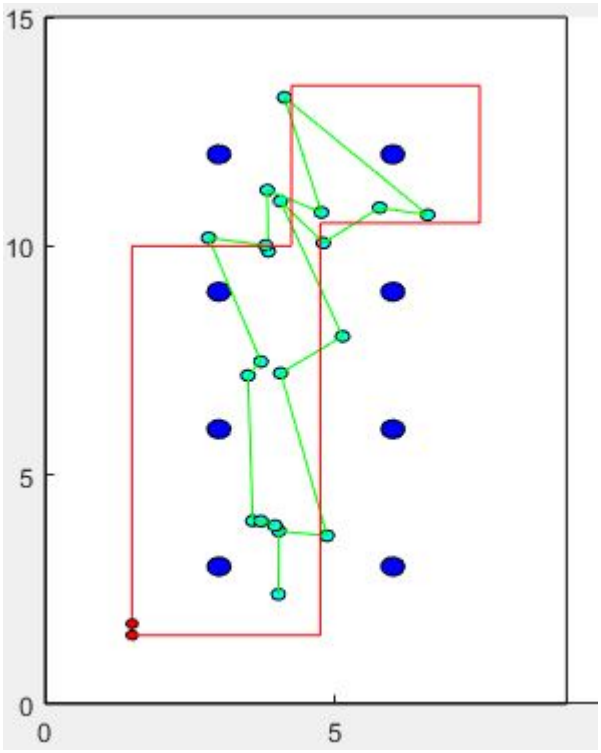


Figura 22 – *Resultados de las medidas en dinámico TFG.*

Las líneas rojas de la imagen representan el camino real, mientras que los puntos y líneas verdes representan la estimación del camino estimado por el algoritmo.

Como se puede observar los resultados obtenidos llevan a la misma conclusión que los resultados de las pruebas en estático. Diciéndonos así que en las dos pruebas la efectividad del algoritmo es mucho mayor cuando el sujeto a localizar se encuentra dentro del are verde respecto a cuando se encuentra en el área roja.

5 MEJORA DE ALGORITMO DE LOCALIZACION EN INTERIORES

En el capítulo anterior se ha explicado resumidamente el funcionamiento del algoritmo expuesto en el TFG anterior, además se han analizado los resultados encontrando sus puntos débiles. En este capítulo, se explicará el diseño del nuevo algoritmo que se ha diseñado partiendo del algoritmo antiguo. Primero se hará una pequeña prueba con una sola baliza para probar la calibración. Posteriormente, se montará un despliegue nuevo con cuatro balizas, para simular una situación más real y se tomaran medidas que posteriormente se analizaran en Matlab con los diferentes algoritmos (El del TFG y el nuevo algoritmo desarrollado en este trabajo de fin de máster). De esta forma se podrá compara la eficacia del algoritmo nuevo respecto a viejo.

5.1 Escenario

Para la realización de una simulación real se ha decidido realizar el despliegue en un taller de carpintería que se encuentra en una nave situada en Astillero, Cantabria. Dentro de esa nave se ha escogido una zona delimitada por dos paredes y varias estanterías, formando de esta forma un área cuadrada de 9x9 metros.

5.1.1 Despliegue TFG

En la imagen siguiente podemos ver el despliegue que se ha llevado a cabo para la simulación usando el algoritmo del TFG. Para esta simulación se realizará un estudio de la constante de perdidas previo al despliegue y no será necesario ningún tipo de calibración durante la prueba. Por este motivo para el despliegue solo se necesitará desplegar cuatro balizas BT. Estas balizas estarán a una distancia de 3 metros entre ellas y entre ellas y las paredes o estanterías, formando así un cuadrado como el que vemos en la imagen.

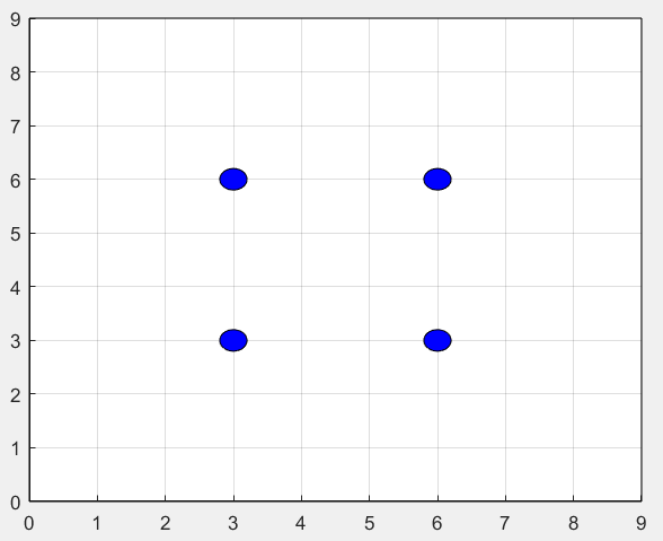


Figura 23 – *Escenario de pruebas para el algoritmo del TFG.*

5.1.2 Despliegue TFM

Ahora lo que se muestra en la siguiente imagen es el despliegue llevado a cabo para la simulación del algoritmo desarrollado en este trabajo. Al contrario que en la simulación del TFG, no se necesita de un estudio previo de la variable de pérdidas de camino. Sin embargo, si que se hará uso de dos calibradores que irán tomando medidas de las balizas a la vez que el usuario recoge sus datos para estimar la posición. Por este motivo vemos en la imagen siguiente dos puntos morados que representan a dos móviles que usaran como calibradores. Estos dos calibradores se situarán: uno en el mismo eje x que el de las balizas de la izquierda y entre las dos, y el otro calibrador en el mismo eje x que las balizas de la derecha y entre ellas dos.

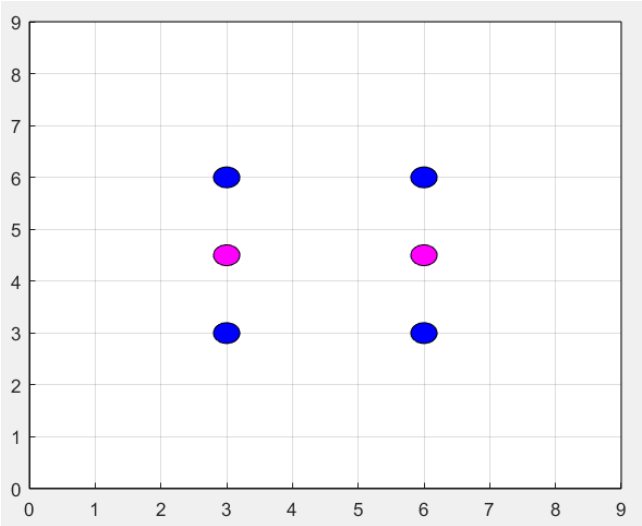


Figura 24 – *Escenario de pruebas con calibradores.*

5.2 Preparación y calibración previa

Ya con la elección de los escenarios, y antes de montar el despliegue de balizas, se necesitará configurar estas adecuada y posteriormente calcular el valor de la constante "n", también llamada Path loss o constante de perdida de ruta. Este valor, calculado previamente al despliegue, dependerá de cada escenario y será utilizado únicamente por el algoritmo de localización en interiores del TFG.

Para configurar las balizas el primer paso será el de conectar los módulos MLT-BT05 a la placa Arduino. En la siguiente figura se muestra el esquema de conexiones que se ha seguido. Hay que recordar que el módulo MLT-BT05 es un clon del HM-10, por lo que las conexiones son idénticas para los dos módulos [15]:

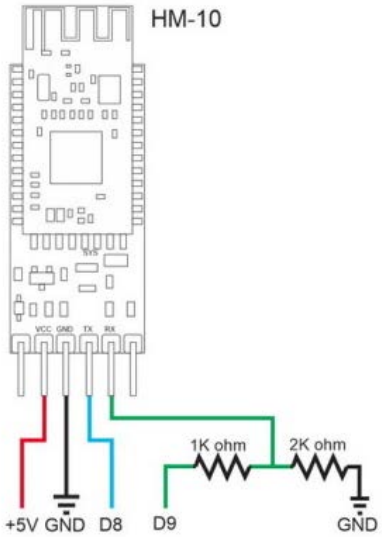


Figura 25 - *Diagrama de conexión entre el módulo MLT-BT05 y la placa Arduino.*

Los cables rojo y negro se conectan al Arduino simplemente para suministrar corriente eléctrica al módulo para que este se encienda. Por otro lado, el cable azul conecta el pin de transmisión de datos del MLT-BT05 con el de recepción del Arduino, mientras que el verde, al revés, conecta el pin de recepción de datos del MLT-BT05 con el de transmisión del Arduino.

El siguiente paso para seguir es conectar la placa Arduino al ordenador mediante un cable USB, abrir el programa Arduino y cargar en este el programa con los comandos "AT", que permiten configurar el módulo MLT-BT05 con los valores deseados.

Los comandos "AT" utilizados son los siguientes [15]:

- **AT+ADDR?**: Devuelve la dirección MAC del módulo MLT-BT05, necesaria para identificar cada baliza y así poder referenciarla adecuadamente en el algoritmo.
- **AT+POWE3**: Permite modificar el valor de la potencia emitida, siendo en este caso la máxima, a +6 dBm.
- **AT+ADVI0**: Este comando permite que el módulo MLT-BT05 emita beacons a un régimen de 1 cada 100 ms.

El último paso antes de colocar las balizas en el escenario es el de obtener el valor de la constante de perdida de ruta, cuya expresión es la siguiente:

$$n = \frac{RSSI_0 - RSSI_i}{10 \log_{10} d_i} \tag{5.2}$$

Para realizar el cálculo basta con utilizar la aplicación de medida deseada, y obtener todos los beacons de un dispositivo Bluetooth, a un metro de distancia del mismo. De esta forma es posible conocer cuál es la RSSI del MLT-BT05 a un metro en el espacio seleccionado, y se corresponde con la variable $RSSI_0$. A continuación, haciendo uso de la fórmula anterior, se van obteniendo los diferentes valores de "n" correspondientes a diferentes distintas (d_i) y, a poder ser, en diferentes localizaciones del aula. Como

resultado, los valores de "n" así obtenidos permiten hacer una estimación del valor real de "n" en el aula en cuestión.

Con esto finaliza el proceso de calibración previa utilizada para el algoritmo del TFG, que solamente depende del espacio físico, y no tanto de la distribución de las balizas. El siguiente paso es simplemente conectar cada módulo BT a una batería de LiPo, y colocar estas balizas según la configuración elegida.

5.3 Prueba de calibración en una sola de las balizas

En esta prueba lo que se pretende es compara el error obtenido al medir la distancia con una sola baliza, realizando un cálculo previo de la constante "n" frente a la obtención de la "n" a la vez que se toman las medidas. O lo que es lo mismo, el método del TFG contra el método de calibración instantánea de este trabajo de fin de máster.

5.3.1 Toma de medidas

En la siguiente imagen podemos ver un esquema de la toma de medidas de esta prueba. Como elemento principal tenemos la baliza BT la que estará enviando beacons constantemente. Luego los puntos rojos simbolizan la posición del usuario con el teléfono móvil que ira recogiendo datos de los beacons de la baliza. Se tomarán medidas a 4 distancias diferentes de 1, 2, 3 y 4 metros de distancia a la baliza, en tiempos diferentes y modificando la disposición de ciertos objetos y muebles del escenario. De esta forma se obtendrá una simulación que se ajuste más a la realidad, ya que, con el paso del tiempo, en un escenario real es posible que la disposición de los objetos de la estancia cambie. Y por último para el algoritmo de este TFM serán necesarios dos móviles que harán la función de calibradores. Estos dos móviles se colocarán, uno a distancia una distancia cualquiera (en este caso 3 metros) y el otro a una distancia de 1 metro para no complicarnos con los cálculos.

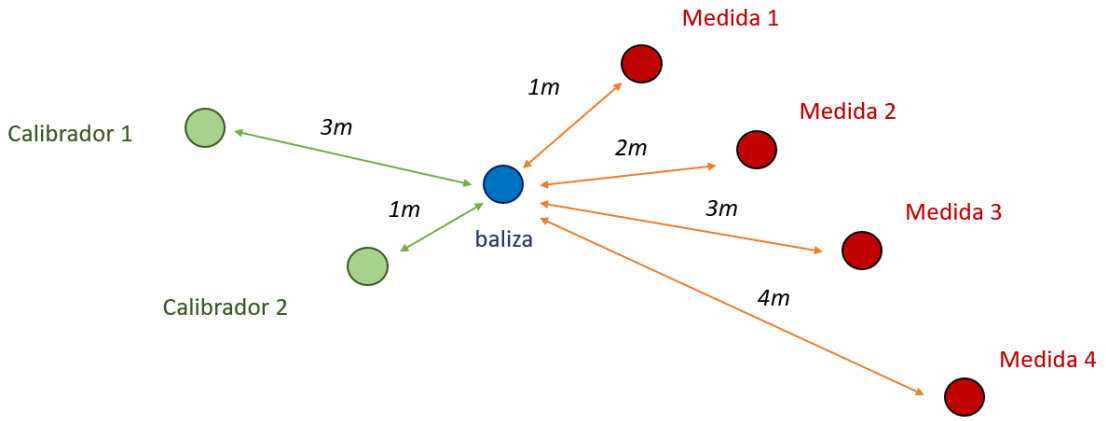


Figura 26 – *Escenario de pruebas para una sola baliza.*

5.3.2 Estimación de la posición

Tras tomar las 4 medidas ya solo queda calcular la estima de la distancia usando los dos métodos, el método antiguo de TFG y el actual. Para los dos métodos se partira de la siguiente fórmula para calcular la distancia:

$$d_i = 10^{-\frac{RSSI_0 - RSSI_i}{10n}} \tag{5.2}$$

Para los dos métodos $RSSI_0$ representara el RSSI tomado a 1 metro y $RSSI_i$ hace referencia al RSSI que obtiene el usuario al obtener los datos de los beacons de la baliza. El valor de $RSSI_i$ será el mismo para los dos métodos, pero el valor de $RSSI_0$ será diferente. En el método del TFG se obtiene el valor de $RSSI_0$ midiéndolo en la calibración previa, lo que quiere decir que el valor $RSSI_0$ será el mismo para las 4 medidas. Por otro lado, el valor $RSSI_0$ que se usara en este nuevo método es el valor que se obtiene por medio del Calibrador 2, que se encuentra a distancia de 1 metro de la baliza. En este caso el valor de $RSSI_0$ será diferente para cada una de las 4 medidas, ya que este valor se obtiene a la vez que cada una de las medidas, y podrá variar con el tiempo y el cambio del entorno.

La última diferencia entre los dos métodos es la obtención de la "n". En el primer caso, como se ha dicho en el punto 5.2, se hará un estudio previo de la "n" la cual será la misma para todas las medidas. Sin embargo, en el algoritmo desarrollado en este trabajo se ha decidido hallar la "n", usando dos calibradores, en el mismo instante que se toman las medidas. Cada vez que se toman medidas con el móvil del usuario a localizar también se tomaran con los calibradores. Luego se sustituyen los RSSI obtenidos por estos calibradores en la fórmula 5.1 y se calculara la "n" para esa medida en concreto. A este proceso se le ha llamado calibración instantánea.

Una vez definidas las variables para los dos métodos y calculadas las distancias, solo nos queda compara los resultados.

5.3.3 Análisis de resultados

En la siguiente tabla podemos ver una comparación de la eficacia de los dos métodos. La primera columna muestra las 4 medidas cuyos índices son iguales a las distancias reales en metros. La segunda y tercera columna hace referencia a las "n" calculadas para el método del TFG y el de este TFM respectivamente. Como podemos observar para el método del TFG la "n" es la misma para todas las medidas, y para el nuevo método es distinta. La cuarta y quinta columna representan las distancias estimadas en metros por los dos métodos. Y por último la sesta y séptima columna el error obtenido entre la distancia real y la estimada en los dos casos.

Nº de Medida (Igual a Distancia en metros)	N (Path loss) previa	N (Path loss) instantánea	Distancia estimada sin calibración	Distancia estimada con calibración	Error sin calibrar (m)	Error tras calibrar (m)
1	3,24	3.1891	0,9720	0.9857	*0,03*	*0,01*
2	3,24	3.2223	1,4888	1.7337	*0,51*	*0,27*
3	3,24	5.2819	4,3232	2.7978	*1,32*	*0,20*
4	3,24	2.7904	4,6416	4.3801	*0,64*	*0,38*
Media error	**x**	**x**	**x**	**x**	***0,63***	***0,22***

Tabla 2 – *Resultados de la prueba con una sola baliza.*

Como podemos ver el error que se obtiene del método uno va de 0,03 metros (medida 1) a los 1,32 metros (medida 3) y con un error promedio de 0,63 metros. Por otro lado, el error obtenido del método de este trabajo va de 0,01 metros (medida 1) a los 0,38 metros (medida 4) y con un error promedio de 0,22 metros. Con lo cual se puede avanzar diciendo que la calibración instantánea promete una mejora respecto al algoritmo del TFG.

Llegados a esta conclusión se decide mejorar el algoritmo del TFG implementando en él la calibración instantánea para obtener así una medida más precisa de la posición estimada.

5.4 Algoritmo de posicionamiento con calibración

Sea un escenario en el que se despliegan en el plano *K* balizas de forma equidistante, de forma que se puede representar cada posición de las balizas sobre una matriz de dimensión *mxn*. Así, por ejemplo, se puede definir una matriz *MAC*, como la matriz de identificadores del conjunto de balizas *K*:

$$MAC = \begin{pmatrix} mac_{11} & \cdots & mac_{1n} \\ \vdots & \ddots & \vdots \\ mac_{m1} & \cdots & mac_{mn} \end{pmatrix} = (MAC)_{mxn} \leftrightarrow m\ x\ n = K \qquad (5.3)$$

De esta forma, el elemento mac_{xy} representa la dirección física de la baliza situada en la posición *(x, y)* del escenario seleccionado.

De acuerdo con lo anterior, se pueden definir tres nuevas matriz *RSSI* ($RSSI\_usu_{(x,y)}$, $RSSI\_cali1_{(x,y)}$ y $RSSI\_cali2_{(x,y)}$), en la que se almacena el valor medio de potencia recibida, desde cada una de las balizas que componen el escenario, por un dispositivo móvil situado en una posición *(x,y)* desconocida pero dentro del

espacio *mxn* y por dos calibradores situados en dos posiciones *(x,y)* conocidas y dentro del espacio *mxn* :

$$RSSI\_usu_{(x,y)} = \begin{pmatrix} rssi_{11} & \cdots & rssi_{1n} \\ \vdots & \ddots & \vdots \\ rssi_{m1} & \cdots & rssi_{mn} \end{pmatrix} = (RSSI)_{mxn} \leftrightarrow m\ x\ n = K \quad (5.4)$$

$$RSSI\_cali1_{(x,y)} = \begin{pmatrix} rssi_{11} & \cdots & rssi_{1n} \\ \vdots & \ddots & \vdots \\ rssi_{m1} & \cdots & rssi_{mn} \end{pmatrix} = (RSSI)_{mxn} \leftrightarrow m\ x\ n = K \quad (5.5)$$

$$RSSI\_cali2_{(x,y)} = \begin{pmatrix} rssi_{11} & \cdots & rssi_{1n} \\ \vdots & \ddots & \vdots \\ rssi_{m1} & \cdots & rssi_{mn} \end{pmatrix} = (RSSI)_{mxn} \leftrightarrow m\ x\ n = K \quad (5.6)$$

El siguiente paso que realiza el algoritmo es el cálculo de las "$n_{(x,y)}$" (path loss) de cada una de las balizas, por medio de las matrices $RSSI\_cali1_{(x,y)}$ *y* $RSSI\_cali2_{(x,y)}$ obtenidas anteriormente y de las distancias $d\_cali1_{(x,y)}$ y $d\_cali2_{(x,y)}$ que son las distancias entre el calibrador 1 y las balizas y el calibrador 2 y estas respectivamente.

$$n_{(x,y)} = \frac{RSSI\_cali1_{(x,y)} - RSSI_{cali2\,(x,y)}}{10\log_{10}\left(\frac{d\_cali2_{(x,y)}}{d\_cali1_{(x,y)}}\right)} \quad (5.7)$$

Como son bien conocidas las posiciones de (x,y) de las balizas y de los calibradores también se conocen las distancia entre estos.

Una vez calculadas las "$n_{(x,y)}$" se procede al calculo de la distancia estimada de cada baliza con el usuario a localizar. Estas distancias se llamaran "$r_{(x,y)}$" y serán calculadas con las matrices $RSSI\_usu_{(x,y)}$, $RSSI\_cali1_{(x,y)}$, $RSSI\_cali2_{(x,y)}$ y con "$n_{(x,y)}$" con cualquiera de las dos siguientes formulas.

$$r_{(x,y)} = 10^{\frac{RSSI\_cali1_{(x,y)} - RSSI\_usu_{(x,y)}}{10n_{(x,y)}} + \log_{10} d\_cali1_{(x,y)}} \quad (5.8)$$

$$r_{(x,y)} = 10^{\frac{RSSI\_cali2_{(x,y)} - RSSI\_usu_{(x,y)}}{10n_{(x,y)}} + \log_{10} d\_cali2_{(x,y)}} \quad (5.9)$$

Llegados a este punto el algoritmo debe de escoger 3 balizas para la realización del método de trilateración, P_1, P_2, P_3. Las tres balizas escogidas han de ser las que se encuentren más cerca del sujeto a localizar. Por este motivo se escogen las tres que

tengan la menor "$r_{(x,y)}$" calculado anteriormente. De esta forma, el primer punto de referencia coincidirá con la posición de la baliza con menor valor relativo de "$r_{(x,y)}$":

$$P_1 = (x_1, y_1) \leftrightarrow \min\left(r_{(x,y)}\right) = r_{x_1,y_1} \tag{5.10}$$

Posteriormente es necesario hallar dos posiciones más, la primera ha de ser mayor que "r_{x_1,y_1}" pero menor que todas las demás y la siguiente mayor que las dos anteriores pero menor al resto de balizas.

$$P_2 = (x_2, y_2) \leftrightarrow \min\left(r_{(x,y)} > r_{x_1,y_1}\right) = r_{x_2y_2} \tag{5.11}$$

$$P_3 = (x_3, y_3) \leftrightarrow \min\left(r_{(x,y)} > r_{x_2,y_2}\right) = r_{x_3y_3} \tag{5.12}$$

A continuación, se definen las circunferencias asociadas con cada punto, P_1, P_2, P_3, y con radios menores calculados anteriormente r_{x_1,y_1}, r_{x_2,y_2}, r_{x_3,y_3}.

La trilateración devuelve los puntos de corte de las tres circunferencias, mediante la resolución del correspondiente sistema de ecuaciones:

$$\begin{cases} (x-x_1)^2 + (y-y_1)^2 - r_{x_1,y_1}^2 = 0 \\ (x-x_2)^2 + (y-y_2)^2 - r_{x_2,y_2}^2 = 0 \\ (x-x_3)^2 + (y-y_3)^2 - r_{x_3,y_3}^2 = 0 \end{cases} \tag{5.13}$$

Sin embargo, al igual que en el algoritmo del TFG, la solución de dicho sistema solamente devuelve la posición buscada en el caso ideal, en el que la distancia de este a los centros de cada circunferencia coincide con el radio de esta. Por este motivo se realizará el mismo estudio que se hace en el algoritmo anterior, particularizado al conjunto de puntos de corte entre cada par de circunferencias, tal como se explica en el apartado 2.4 de este trabajo.

Una vez obtenidos dichos puntos, se establece el triángulo de referencia sobre el que se ha de calcular su baricentro, que coincide con la posición buscada. En el siguiente apartado se ha explicado cómo se ha llevado a cabo la implementación de este algoritmo en el entorno de programación utilizado para las simulaciones, Matlab.

Para la comprobación de los resultados del procedimiento descrito, una vez colocadas las balizas en el escenario, el proceso de medida en estático se realiza de la siguiente manera: se selecciona una localización concreta a la que se desplaza el dispositivo móvil y, mediante la aplicación de medida, se recogen datos durante 10 segundos, los cuales se guardan en un fichero CSV. Este proceso se repite a lo largo de diferentes posiciones dentro del aula. Además, como se ha hecho en la prueba de calibración de una sola baliza, entre medida y medida se hacen pequeños cambios en la disposición de los objetos y muebles del escenario de pruebas, para así simular una situación más realista.

Para la comprobación de las medidas en dinámico el único cambio es que se manda al algoritmo una medida de 27 segundos que es troceada en 9 medidas de 3 segundos. El funcionamiento del algoritmo es igual para el caso de la prueba en dinámico que el de la prueba en estático. Se obtiene la posición de las 9 medidas y se unen los puntos de forma ordenada para obtener el camino estimado.

5.4.1 Implementación

Tal como se ha indicado anteriormente, lo primero que hace el algoritmo es leer los datos correspondientes a las medidas tomadas por el sujeto a localizar, en forma de tabla, en la que se encuentran desordenados los valores RSSI de todos los dispositivos Bluetooth del escenario, los cuales a su vez están asociados a su correspondiente MAC.

```
% Se lee el fichero con los datos
medida = readtable('../../1 - Datos/CSV/Resultados9.csv','Delimiter',',');
medida = sortrows(medida,{'Var3','Var1','Var4'});
```

Posteriormente el algoritmo hace la media de los RSSI de cada dispositivo BT, operación que ha sido implementada en una función denominada "*hallar_balizas_nuevo*":

```
[rssi_usu]=hallar_balizas_nuevo(elements);
```

Esta función devolverá la media de los RSSI de forma ordenada en la matriz "*rssi_usu*".

En el siguiente ciclo "*for*" lo que se está haciendo es obtener la media de todos los RSSI que se haya captado con el dispositivo móvil, independientemente de si es de una de las balizas del despliegue o de otro elemento que porte la tecnología bluetooth.

```
for j=1:length(uniqueVal)
    elements = group(strcmp(group.Var1,uniqueVal(j,1)),:).Var2;
    mean(j,1) = sum(elements)/length(elements);
end
```

Tras obtener la media, se seleccionan solo los valores de las balizas que están referenciadas en el escenario, desechando el resto. Dichos valores deben ser guardados de forma matricial, de acuerdo con la configuración espacial escogida para ese escenario. Por ejemplo, en el escenario que en el cual se han realizado las simulaciones, los valores son guardados en una matriz de 2 filas y 2 columnas, ya que este despliegue es de 4 balizas formando un cuadrado:

```
for j=1:length(uniqueVal)
    uniqueVal{j,1} ;
    valores(strcmp(valores.Var1,uniqueVal{j,1}),:).Var2;
    A = zeros(2,2);
    if(strcmp(uniqueVal{j,1},'C8:FD:19:11:9C:93'))
        A(2,1) = valores(strcmp(valores.Var1,uniqueVal{j,1}),:).Var2;
    elseif(strcmp(uniqueVal{j,1},'C8:FD:19:4B:04:D3'))
        A(1,1) = valores(strcmp(valores.Var1,uniqueVal{j,1}),:).Var2;
    elseif(strcmp(uniqueVal{j,1},'C8:FD:19:13:67:F2'))
        A(2,2) = valores(strcmp(valores.Var1,uniqueVal{j,1}),:).Var2;
    elseif(strcmp(uniqueVal{j,1},'C8:FD:19:11:A8:20'))
        A(1,2) = valores(strcmp(valores.Var1,uniqueVal{j,1}),:).Var2;
    end
   beacons = beacons+A;
end
```

Una vez obtenida la matriz de RSSI medidos por el usuario a localizar (a la que se ha llamado $rssi\_usu_{(x,y)}$ se procederá a calibrar las balizas, hallar las distancias estimadas entre las balizas y el usuario y elegir las tres balizas más óptimas para emplear la trilateración.

```
[rssi_der,rssi_izq,rssi_usu,h1,k1,h2,k2,h3,k3,r1,r2,r3]=calibracion(rssi_usu);
```

Todo esto lo se hara dentro de la función "*calibracion*" en la cual se itroducira la matriz hallada anteriormente "rssi_usu" obtenido así las coordenadas de las 3 balizas más optimas en las variables "*h1, k1, h2, k2, h3, k3*" donde las "*h*" representan la coordenada "*x*", las "*k*" la coordenada "*y*" y su índice corresponden con la numeración de las 3 balizas ordenadas de menor a mayor distancia con el usuario. Esta función también devuelve la distancia o radio de estas 3 balizas con el usuario en las variables "*r1, r2, r2*".

Lo primero que se hace dentro de la función "*calibracion*" es leer los datos de RSSI obtenidos por cada uno de los calibradores (calibrador derecho y calibrador izquierdo), luego se ordenarán y se obtendrá su media formando las matrices "$rssi\_der_{(x,y)}$" y "$rssi\_izq_{(x,y)}$". El código para formar estas matrices es similar al código utilizado para hallar la matriz "$rssi\_usu_{(x,y)}$" descrito anteriormente.

```
%% Se lee el fichero con los datos calibracion_izq
medida_izq = readtable('../../1 - Datos/CSV/m_m_estatico/Calibracion_izq_9.csv','Delimiter',',');
medida_izq = sortrows(medida_izq,{'Var3','Var1','Var4'});

%% Se lee el fichero con los datos calibracion_der
medida_der = readtable('../../1 - Datos/CSV/m_m_estatico/Calibracion_der_9.csv','Delimiter',',');
medida_der = sortrows(medida_der,{'Var3','Var1','Var4'});
```

Una vez halladas ya las matrices "$rssi\_der_{(x,y)}$", "$rssi\_izq_{(x,y)}$" y "$rssi\_usu_{(x,y)}$" queda saber cual es la distancia entre los calibradores y las balizas para poder calibrar.

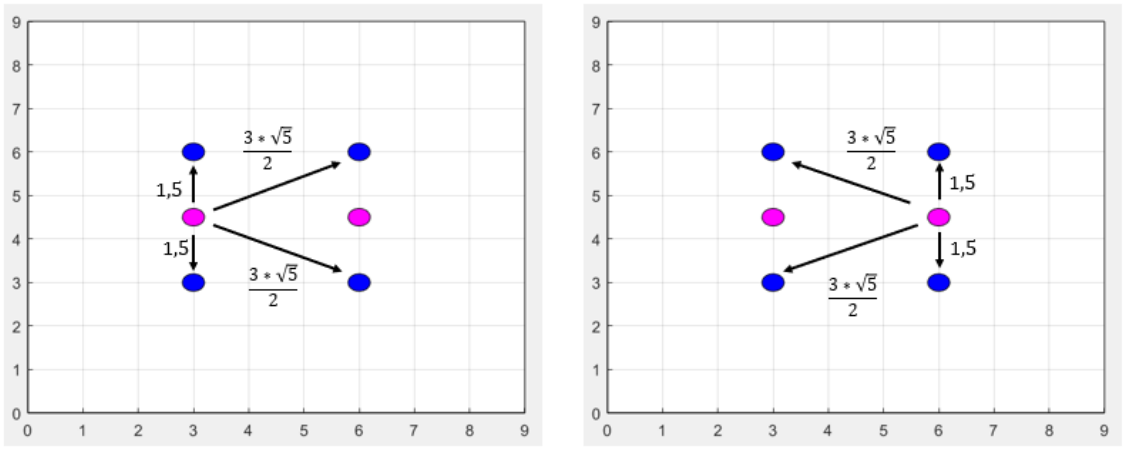


Figura 27 – *Distancias entre los calibradores y las balizas.*

La colocación que se ha propuesto para los calibradores permite manejar únicamente dos valores de distancias.

```
%% Calibración de path loss (n)
    n = zeros(2,2);
    d1 = 1.5;
    d2 = 3*sqrt(5)/2;
    % Distancias de calibracion [d1 = 1.5], [d2 = 3*sqrt(5)/2]
    n(1,1) = (rssi_der(1,1) - rssi_izq(1,1))/(10*log10(d1/d2));
    n(1,2) = (rssi_der(1,2) - rssi_izq(1,2))/(10*log10(d2/d1));
    n(2,1) = (rssi_der(2,1) - rssi_izq(2,1))/(10*log10(d1/d2));
    n(2,2) = (rssi_der(2,2) - rssi_izq(2,2))/(10*log10(d2/d1));
```

En el código anterior lo que hace es calcular las pérdidas de camino "$n_{(x,y)}$" que tiene cada baliza independientemente de las otras. Esto se obtendra por medio de los RSSI medidos por los calibradores y con las distancias conocidas entre ellos y las balizas "*d1, d2*", utilizando la fórmula 2.4.

Una vez obtenidas las "$n_{(x,y)}$" de cada baliza se procede a calcular las distancias entre las balizas y el usuario a localizar tras la calibración. Estas distancias se guardarán en la variable "$cr_{(x,y)}$", la cual será calculada usando las variables "$rssi\_der_{(x,y)}$", "$rssi\_usu_{(x,y)}$", "*d1, d2*" y la fórmula 2.5.

```
%% Calculo de los 4 radios de las balizas al usuario
    cr = zeros(2,2);
    cr(1,1)=10^((rssi_der(1,1) - rssi_usu(1,1))/(n(1,1)*10)+log10(d2));
    cr(1,2)=10^((rssi_der(1,2) - rssi_usu(1,2))/(n(1,2)*10)+log10(d1));
    cr(2,1)=10^((rssi_der(2,1) - rssi_usu(2,1))/(n(2,1)*10)+log10(d2));
    cr(2,2)=10^((rssi_der(2,2) - rssi_usu(2,2))/(n(2,2)*10)+log10(d1));
```

Lo último que queda por hacer en la función "*calibracion*" es averiguar cuales con las tres balizas más próximas al sujeto a localizar. Para ello se utilizará el siguiente código.

```
%% Calculo de los 3 radios menores
    x = zeros(2,2);
    x(1,1)=cr(2,1);x(1,2)=cr(2,2);
    x(2,1)=cr(1,1);x(2,2)=cr(1,2);
    min=100;
    for i=1:2
        for j=1:2
            if(x(i,j) < min)
                min=x(i,j);
                h1=i;
                k1=j;
                r1=min;
            end
        end
    end
   min=100;
   for i=1:2
       for j=1:2
           if(h1 ~= i) || (k1 ~= j)
               if(x(i,j) < min)
                   min=x(i,j);
                   h2=i;
                   k2=j;
                   r2=min;
               end
           end
       end
   end
   min=100;
   for i=1:2
       for j=1:2
           if(h1 ~= i) || (k1 ~= j)
           if(h2 ~= i) || (k2 ~= j)
               if(x(i,j) < min)
                   min=x(i,j);
                   h3=i;
                   k3=j;
                   r3=min;
               end
           end
           end
       end
   end
```

```
hh1=h1*3;kk1=k1*3;
hh2=h2*3;kk2=k2*3;
hh3=h3*3;kk3=k3*3;
k1=hh1;h1=kk1;
k2=hh2;h2=kk2;
k3=hh3;h3=kk3;
```

Una vez obtenidas las tres balizas más optimas y sus distancias o radios con el usuario, se puede empezar con el método de trilateración que se encuentra en la función con su mismo nombre "$trilateracion$", la cual devuelve la posición buscada, siendo "h" la variable que representa la coordenada x en el aula, y "k" la coordenada y.

```
rr1=r1;rr2=r2;rr3=r3;rh1=h1;rk1=k1;rh2=h2;rk2=k2;rh3=h3;rk3=k3;
[h,k]=trilateracion(rr1,rr2,rr3,rh1,rh2,rh3,rk1,rk2,rk3);
```

Las particularidades de dicha función hacen que sea necesario una explicación más detallada del proceso implementado. Hasta este momento, el cálculo se encuentra en un estado como puede representar la figura 31. Se han hallado las tres balizas más aptas para llevar acabo la trilateración, así como la distancia relativa de éstas al móvil:

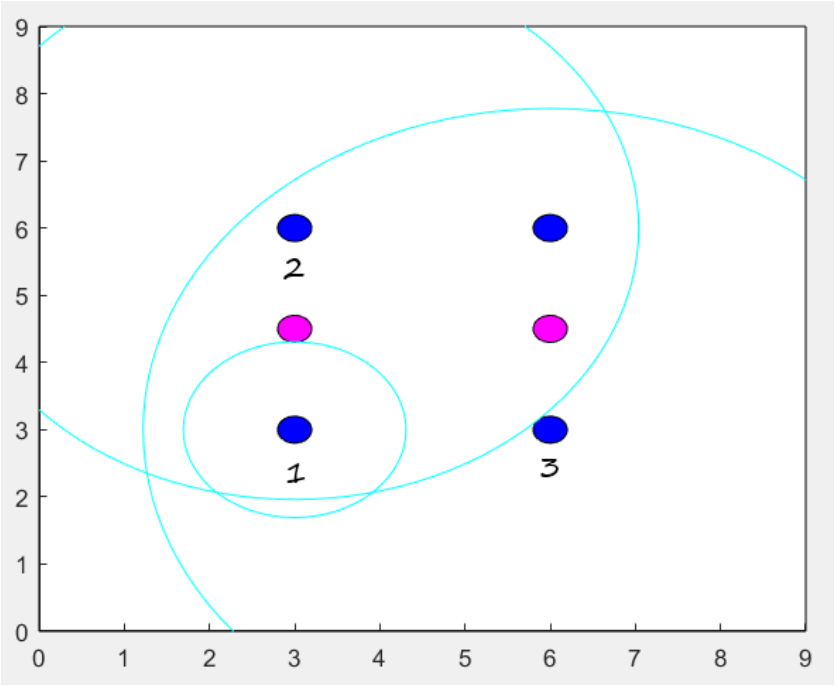


Figura 28 - *Representación en MATLAB de los radios de las balizas.*

Como se explicó en el apartado 2.3 el siguiente paso sería analizar las circunferencias de dos en dos, hallando 3 puntos que formarán un triángulo cuyo centro será la posición relativa del móvil. Para esto se define la función "$puntos\_triangulo$", que es llamada tres veces dentro de la función "$trilateracion$", de forma que se halla, cada vez, uno de estos puntos.

```
[hh1,kk1]=puntos_triangulo(r1,r2,r3,h1,k1,h2,k2,h3,k3);
```

Para ello se contemplan los dos casos posibles, definidos cuando se analizan las circunferencias de dos en dos:

- El primero es que estas dos circunferencias se corten. Si se cortan en un punto, éste será el punto buscado, pero si se cortan en dos, se elige el que esté más cerca de la tercera circunferencia. El siguiente código lleva a cabo este proceso:

```
    dp=sqrt((h1-h2)^2+(k1-k2)^2);
if r2 + r1 >= dp && r1 + dp >= r2
    %%%%%%%%%%%%%%%% Corte circunferencias %%%%%%%%%%%%%%%%%%

    syms h k x1 x2 y1 y2 rr1 rr2
    equ1 = (h - x1)^2 + (k - y1)^2 - (rr1)^2;
    equ2 = (h - x2)^2 + (k - y2)^2 - (rr2)^2;
    %
    equ1=subs(equ1,[x1,x2,y1,y2,rr1,rr2],[h1,h2,k1,k2,r1,r2]);
    equ2=subs(equ2,[x1,x2,y1,y2,rr1,rr2],[h1,h2,k1,k2,r1,r2]);
    %
    [h,k]=solve(equ1 == 0,equ2 == 0);
    clear x1 x2 y1 y2 rr1 rr2
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

La variable "dp" es la distancia entre los centros de las circunferencias y en las variables "h" y "k" se guardan las coordenadas de los puntos de corte entre ellas.

```
    if r2 + r1 == dp || r1 + dp == r2
        d=r3-sqrt((h-h3)^2+(k-k3)^2);
        hh=h;
        kk=k;
    else
        d1=r3-sqrt((h(1,1)-h3)^2+(k(1,1)-k3)^2);
        da1=abs(d1);
        d2=r3-sqrt((h(2,1)-h3)^2+(k(2,1)-k3)^2);
        da2=abs(d2);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %
        if da1 < da2
            d=d1;
            hh=h(1,1);
            kk=k(1,1);
        else
            d=d2;
            hh=h(2,1);
            kk=k(2,1);
        end
    end
```

Para finalizar, se comprueba si tiene uno o dos puntos de corte. Si tiene uno, las coordenadas serán el único valor que hay en "h" y "k". Si tiene dos, serán el valor de "h" y "k" más cercano a la circunferencia de la tercera baliza. Estos puntos son guardados en las variables "hh" y "kk".

- El segundo caso se da cuando las circunferencias no se cortan. En este caso el punto de interés será el punto equidistante y más cercano a estas dos circunferencias. Para ello, se define el siguiente código:

```
elseif r1 + r2 < dp || r1 + dp < r2
    %%%%%%%%% Punto medio dentre circunferencia 1 y 2 %%%%%

    if r1 + r2 < dp
        rf = r1 + (dp - (r1 + r2))/2;
    elseif r1 + dp < r2
        rf = r1 + (r2 - dp - r1)/2;
    end
    %
    syms hs ks h k x1 x2 y1 y2 rr1 rr2
    equ1 = (h - x1)^2 + (k - y1)^2 - (rr1)^2;
    equ2 = (h - x2)^2 + (k - y2)^2 - (rr2)^2;
    if h2 == h1
        equ3 = x1 - h;
    else
        equ3 = y1 - k + ((y2 - y1)/(x2 - x1))*(h - x1);
    end
    %
    equ1=subs(equ1,[x1,y1,rr1],[h1,k1,rf]);
    equ2=subs(equ2,[x2,y2,rr2],[h2,k2,r2]);
    equ3=subs(equ3,[x1,x2,y1,y2],[h1,h2,k1,k2]);
    %
    [hs,ks]=solve(equ2 == 0,equ3 == 0);
    [h,k]=solve(equ1 == 0,equ3 == 0);
    clear x1 x2 y1 y2 rr1 rr2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

La variable "rf" es el radio desde el centro de la primera circunferencia, en el cual se encuentra el punto de interés. Así, "$equ1$" es la ecuación de la circunferencia con dicho radio, "$equ2$" es la ecuación de la segunda circunferencia y "$equ3$" es la ecuación de la recta que pasa por los centros de la primera y segunda circunferencia. En "hs" y "ks" se hallan las coordenadas de los puntos de corte entre "$equ2$" y "$equ3$", y en "h" y "k" se encuentran las coordenadas de los puntos de corte entre "$equ1$" y "$equ3$".

```
    d1=sqrt((h(1,1)-hs(1,1))^2+(k(1,1)-ks(1,1))^2);
    d2=sqrt((h(2,1)-hs(1,1))^2+(k(2,1)-ks(1,1))^2);
    d3=sqrt((h(1,1)-hs(2,1))^2+(k(1,1)-ks(2,1))^2);
    d4=sqrt((h(2,1)-hs(2,1))^2+(k(2,1)-ks(2,1))^2);
    %
    if d1 < d2 && d1 < d3 && d1 < d4
        hh=h(1,1);
        kk=k(1,1);
    elseif d2 < d1 && d2 < d3 && d2 < d4
        hh=h(2,1);
        kk=k(2,1);
    elseif d3 < d1 && d3 < d2 && d3 < d4
        hh=h(1,1);
        kk=k(1,1);
    elseif d4 < d1 && d4 < d2 && d4 < d3
        hh=h(2,1);
        kk=k(2,1);
    end
end
```

Ahora lo único que queda es ver cuál es el punto de corte de la circunferencia de radio "rf" con la recta que se encuentre más cerca de la segunda circunferencia. Las coordenadas de dicho punto son las variables "hh" y "kk". En la figura 32 se puede ver dónde estarían los puntos que forman el triángulo en cuyo centro se haya el móvil.

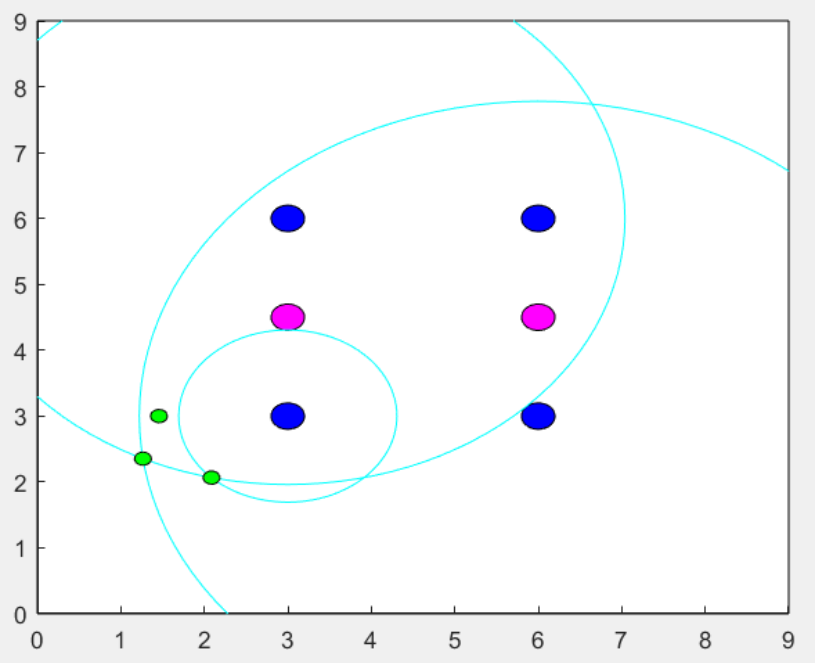


Figura 29 - *Representación en MATLAB de los 3 puntos del triángulo donde se halla el móvil.*

Tras ejecutar tres veces la función "$puntos\_triangulo$" dentro de la función "$trilateracion$", se obtienen los tres puntos del triángulo, del que solo queda hallar su centro. De acuerdo con lo explicado en el apartado 2.3, el triángulo tiene hasta cuatro

centros diferentes, de los cuales el elegido es el baricentro, el cual es calculado en la función "*centro_triangulo*":

```
function [h,k]=centro_triangulo(hh1,kk1,hh2,kk2,hh3,kk3)
    hhh1=(hh1+hh2)/2;
    kkk1=(kk1+kk2)/2;
    hhh2=(hh1+hh3)/2;
    kkk2=(kk1+kk3)/2;
    syms h k x1 x2 x3 x4 y1 y2 y3 y4
    equ1 = k - y1 ==((y2-y1)/(x2-x1))*(h-x1);
    equ2 = k - y3 ==((y4-y3)/(x4-x3))*(h-x3);
    %
    equ1=subs(equ1,[x1,y1,x2,y2],[hhh1,kkk1,hh3,kk3]);
    equ2=subs(equ2,[x3,y3,x4,y4],[hhh2,kkk2,hh2,kk2]);
    %
    [h,k]=solve(equ1,equ2,h,k);
    clear x1 x2 x3 x4 y1 y2 y3 y4
end
```

La función obtiene el baricentro hallando el punto de intersección entre dos medianas del triángulo (siendo una mediana el segmento que une un vértice con el punto medio del lado opuesto).

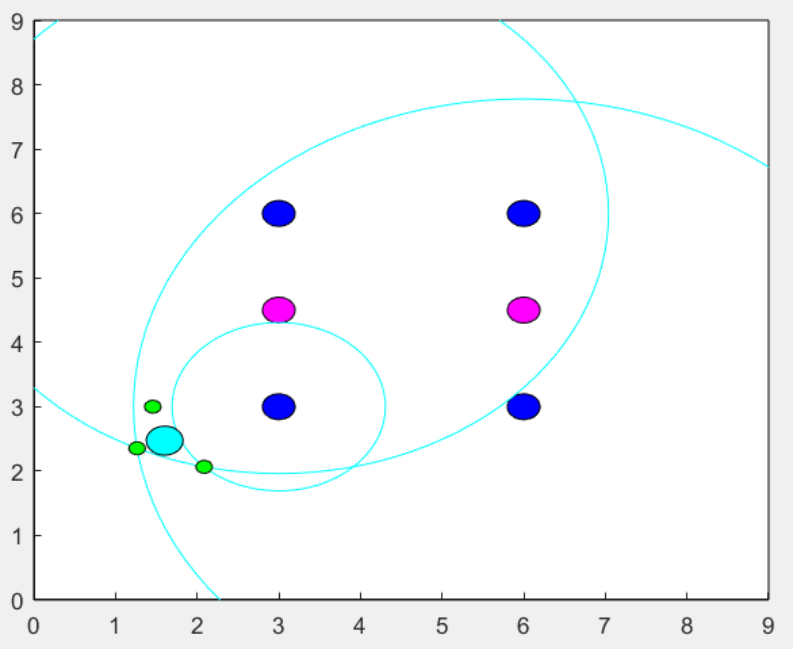


Figura 30 - *Representación en MATLAB del baricentro del triángulo móvil.*

El baricentro representa la posición relativa en la que se encuentra el dispositivo móvil, y que queda representada en el punto azul claro de la figura 33, de acuerdo con el ejemplo utilizado.

Posteriormente, con una pequeña modificación del algoritmo, las funciones definidas pueden ser utilizadas iterativamente para, por ejemplo, trazar la ruta de un sujeto

andando por el escenario. En este caso, el dispositivo móvil recoge los datos de los beacons de las balizas durante 27 segundos, durante el cual se obtienen los puntos de posición cada 3 segundos, obteniendo así 9 puntos. De esta forma es posible unir los puntos consecutivos con rectas para representar la supuesta ruta seguida por el móvil.

```
se=0;
for s=0:MEDIDAS

    group = medida(medida.Var3==0,:);
    % Se eliminan las potencias por debajo del umbral
    group = group(group.Var2>-100,:);
    uniqueVal = unique(group.Var1);
    elements = table;
    se = s * 3000;
    for j=1:length(uniqueVal)
        a = group(strcmp(group.Var1,uniqueVal(j,1)),:);

        %Se le da el periodo de inicio a fin para trocear la informacion
        ini = a.Var4(1)+se;
        fin = a.Var4(1)+(se+3000); % 3 seg
        a = a(a.Var4>ini,:);
        a = a(a.Var4<fin,:);
        elements = vertcat(elements,a);
    end
```

En el código anterior se observa la modificación, respecto del algoritmo en estático, en la cual se ejecuta un bucle que trocea los 27 segundos en 9 trozos de 3 segundos.

5.5 Análisis de los resultados

Se han probado los dos algoritmos (El desarrollando en el TFG anterior y el desarrollado en este TFM) para el escenario expuesto en el principio de este capítulo y bajo dos supuestos, primero con un dispositivo móvil en estático durante un tiempo de 10 segundos por medición y, posteriormente, con el dispositivo en movimiento durante 27 segundos. A continuación, se comparan los resultados más destacables obtenidos por los dos algoritmos. Para más información, en el Anexo de este documento se puede consultar todos los resultados más detalladamente.

5.5.1 Análisis de los resultados en las medidas tomadas en estático.

Los primeros resultados obtenidos mediante Matlab han sido las estimaciones de la posición con relación a las medidas tomadas en 9 puntos concretos de la habitación, en la que el dispositivo ha permanecido inmóvil, durante un tiempo de 10 segundos, durante el cual se han recogido datos de las balizas BT.

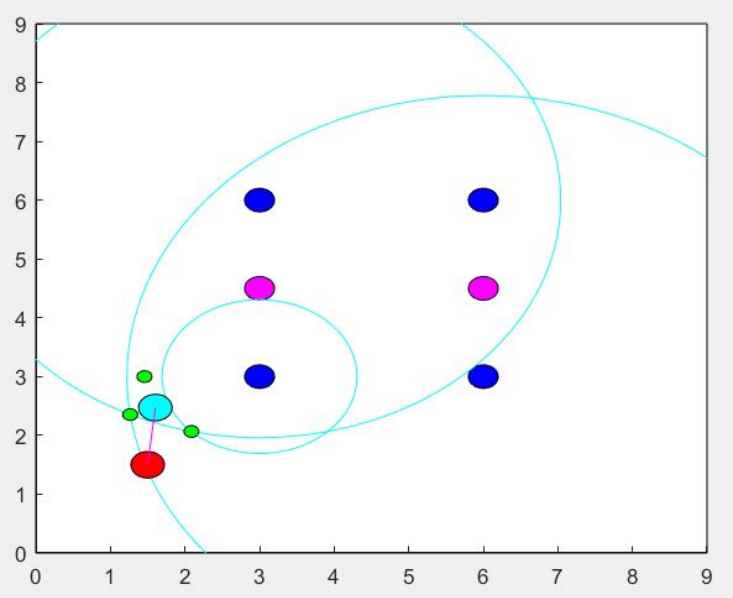


Figura 31 - *Resultado 1 de medición en estático.*

En la Figura 34 podemos ver un ejemplo de representación de resultados de medidas en estático, tal como se pueden encontrar en el anexo. Este ejemplo en concreto corresponde con la estimación de la posición de la medición 1 en estático. Como se puede observar, el escenario está definido en una sala cuadrada de 9x9 metros. Los círculos azules oscuros representan la localización del despliegue de las 4 balizas BT, de forma que la baliza inferior izquierda se encuentra en la coordenada (3,3), la de su derecha en la coordenada (6,3), la de arriba de la primera en (3,6) y la baliza de la esquina superior derecha en la coordenada (6,6). De esta forma las balizas forman un cuadrado de 3x3 metros, siendo ellas las esquinas de este cuadrado. En la figura se muestran 3 circunferencias cuyos centros son las tres balizas más optimas, elegidas por el algoritmo, para realizar la trilateración. Sus radios han sido calculados de acuerdo con la estimación, que hace el algoritmo, de la distancia desde la posición de medida hasta la baliza. El punto azul cian es la posición estimada, obtenida tras realizar la trilateración, mientras que el punto rojo representa la posición real, desde donde se han tomado las medidas. Ambos puntos están unidos por una línea, que representa la distancia entre la posición real y la estimada, lo que permite estimar el error cometido. Para este caso, el error que se comete en la estimación de la posición es de 0.98 metros.

5.5.1.1Resultados del algoritmo sin calibración

En esta prueba se han tomado 9 medidas, que aparecen representadas en los esquemas de la Figura 35, mediante los correspondientes puntos rojos.

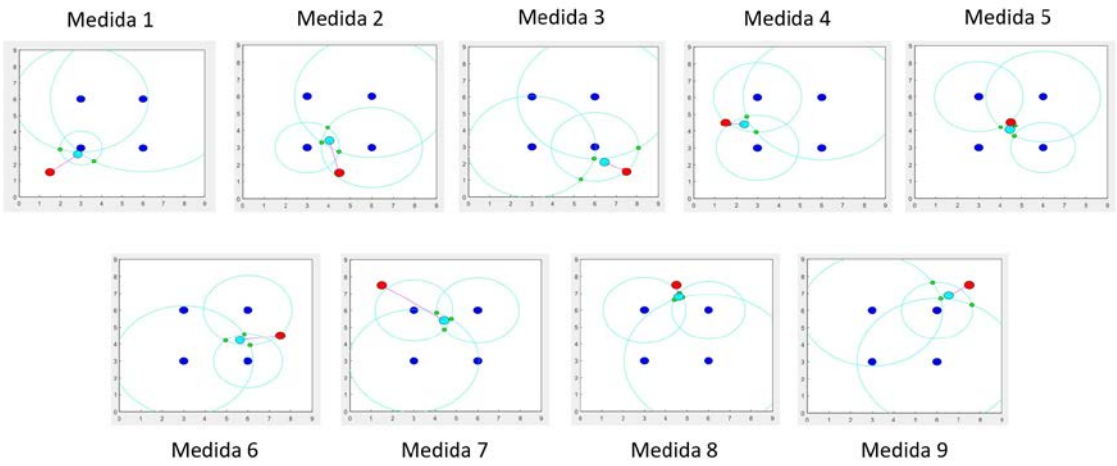


Figura 32 - *Conjunto de medidas obtenidas por el algoritmo del TFG.*

En la figura anterior también se observan los errores cometido tras procesar las medidas con el algoritmo del TFG, cuyos valores reales se muestran en la Tabla 1.

Medición	Error de estimación
1	*1,75m*
2	*1,96m*
3	*1,20m*
4	*0,87m*
5	*0,44m*
6	*1,88m*
7	*3,59m*
8	*0,71m*
9	*1,15m*

Tabla 3 - *Resultados de mediciones en estático por el algoritmo del TFG.*

En esta tabla, podemos ver que el error cometido entre la posición real y estimada se encuentra entre 0,44 metros (medida 5) y 3,59 metros (medida 7).

5.5.1.2Resultados del algoritmo con calibración

Ahora, sin tomar nuevas mediciones, y teniendo en cuenta esta vez los calibradores (representados en la figura siguiente con círculos morados), se procesarán los mismos datos mediante el nuevo algoritmo de posicionamiento desarrollado en este trabajo.

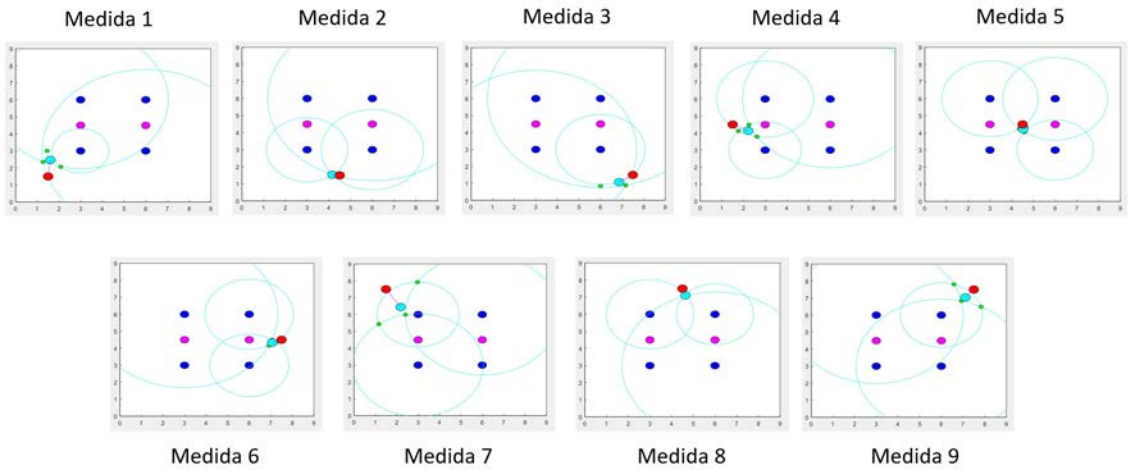


Figura 33 - *Conjunto de medidas obtenidas por el algoritmo con calibración.*

En la tabla siguiente podemos observar el error cometido en metros por el algoritmo de este TFM.

Medición	Error de estimación
1	*0,98m*
2	*0,34m*
3	*0,77m*
4	*0,80m*
5	*0,29m*
6	*0,46m*
7	*1,26m*
8	*0,42m*
9	*0,60m*

Tabla 4 - *Resultados de mediciones en estático con calibración.*

De esta forma, podemos ver que el error cometido entre la posición real y estimada se encuentra entre 0,29 metros (medida 5) y 1,26 metros (medida 7).

5.5.1.3Comparación de resultados

En la siguiente tabla se comparan los resultados obtenidos por el algoritmo del TFG contra los obtenidos en el TFM. Además, la última columna muestra la mejora de estimación en metros que consigue el nuevo algoritmo frente al antiguo. Este valor se ha obtenido restando el "*Error TFG*" menos el "*Error TFM*".

Medición	Error TFG	Error TFM	Mejora
1	*1,75m*	*0,98m*	*0,77m*
2	*1,96m*	*0,34m*	*1,62m*
3	*1,20m*	*0,77m*	*0.43m*
4	*0,87m*	*0,80m*	*0,07m*
5	*0,44m*	*0,29m*	*0,15m*
6	*1,88m*	*0,46m*	*1,42m*
7	*3,59m*	*1,26m*	*2,33m*
8	*0,71m*	*0,42m*	*0,29m*
9	*1,15m*	*0,60m*	*0.55m*
Media	***1,51m***	***0,66m***	***0,85m***

Tabla 5 – *Comparación del error cometido por los dos algoritmos en estático.*

Como podemos ver en la tabla el algoritmo de este trabajo tiene un error menor, respecto al algoritmo del TFG, en todas las medias en estático realizadas en el escenario propuesto. La media de error del algoritmo sin calibración es de 1,51 metros, mientras que la del algoritmo aplicando la calibración es de 0,66 metros. Esto quiere decir que la mejora en metros es de 0,85 metros de media, y que porcentualmente el algoritmo nuevo mejora al antiguo en un 56,29 %.

5.5.2 Análisis de los resultados en las medidas tomadas en movimiento.

Tras comparar los resultados obtenidos en situaciones ideales, con tiempos de medida relativamente altos por cada medida realizada, se continuo con la comparación de los algoritmos de posicionamiento, con una prueba tomando medidas dinámicas. Para ello, ahora se tomarán medidas durante 27 segundos, durante los cuales el dispositivo móvil se desplaza libremente por el escenario, a una velocidad baja, aunque constante. De esta forma la medida simula el desplazamiento de un individuo por el escenario de pruebas. Los algoritmos permiten mostrar el camino recorrido por el sujeto durante el tiempo de medición. Para ello, se ha definido previamente cual es el camino que va a recorrer el individuo por el escenario de pruebas.

5.5.2.1Resultados del algoritmo sin calibración

En la Figura 38 se muestran los resultados obtenidos por el algoritmo del TFG, de las medidas en movimiento obtenidas en el escenario de pruebas. Las líneas rojas corresponden con el trayecto real, trazado por el individuo que toma las medidas con el dispositivo móvil, empezando su ruta en el punto rojo inferior y acabando en el punto rojo superior. Los puntos azul cian corresponden con las 9 medidas tomadas, que se corresponden con 3 segundos de los 27 totales. Las líneas verdes unen medidas consecutivas describiendo así el trayecto estimado.

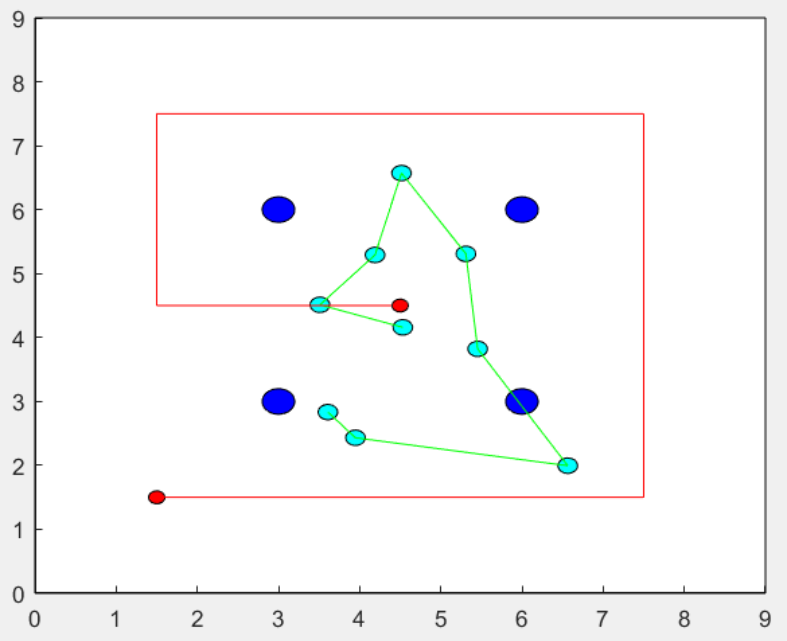


Figura 34 - *Resultado de la medición en dinámico por el algoritmo del TFG.*

Como podemos ver en la figura anterior el camino rojo, que es el que hace referencia al camino real, está en su mayor parte rodeando a las balizas, por fuera del cuadrado de 3x3 metros cullas esquinas serían las balizas. Este diseño del camino trazado, se a realizado de esta forma debido a que las conclusiones del TFG decían que el algoritmo era bueno estimando posiciones internas a este cuadrado, pero sin embargo era ineficiente para estimar las posiciones externas a este. Con el nuevo algoritmo lo que se pretende es mejorar es la estima de la posición en esa área externa. Si nos fijamos ahora en el camino estimado por el algoritmo antiguo en el escenario, se puede ver que efectivamente la mayoría de las medidas tomadas en el exterior del cuadrado son erróneas, pintando este los puntos dentro de dicho cuadrado o muy próximos a él.

5.5.2.2Resultados del algoritmo con calibración

En la siguiente figura podemos ver los resultados obtenidos por el nuevo algoritmo. No se ha vuelto a realizar el camino tomando nuevas medidas, lo que se ha hecho es utilizar la misma medida de 27 segundos que se ha procesado con el algoritmo del TFG, pero esta vez teniendo en cuenta las medidas de los calibradores. Tanto la medida del sujeto como la de los calibradores han sido de 27 segundos tomadas a la vez, las cuales posteriormente el algoritmo troceara en 9 medidas de 3 segundos.

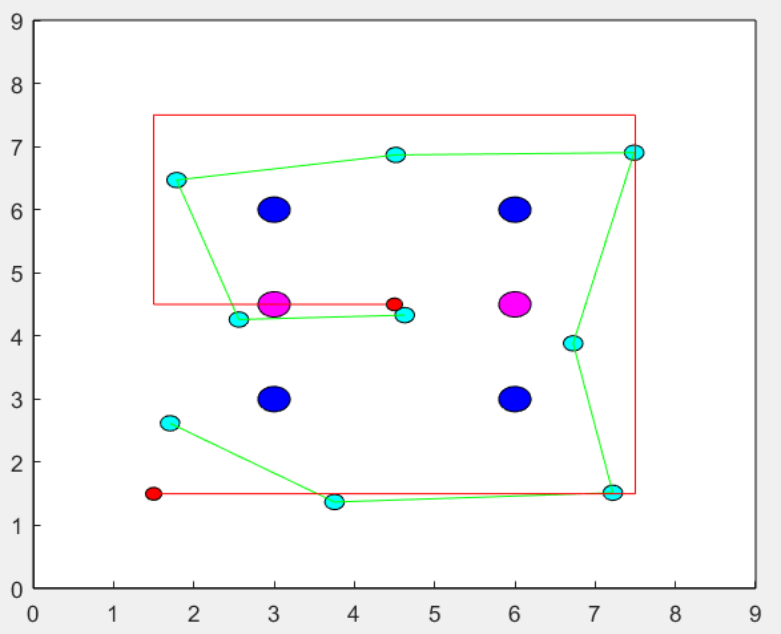


Figura 35 - *Resultado de la medición en dinámico con los calibradores.*

En la imagen siguiente podemos ver la estela estimada por el algoritmo desarrollado en este trabajo. Comparando los resultados de los dos algoritmos se puede apreciar una significante mejora en la estimación de este algoritmo respecto al del TFG anterior. Se puede observar que, aun no siendo aun perfecto, los puntos estimados cuando el trayecto esta tanto dentro como fuera del cuadrado formado por las balizas, tiene menos error respecto al algoritmo sin calibración.

5.5.2.3Comparación de resultados

Para hacer una comparación mas precisa, a parte de apreciar visualmente la notoria mejora, se han marcado los 9 puntos del camino real que deberían obtener los algoritmos. Estos puntos aparecen ordenadas cronológicamente en la siguiente figura. De esta forma se puede obtener el error cometido en cada punto estimado por sendos algoritmos.

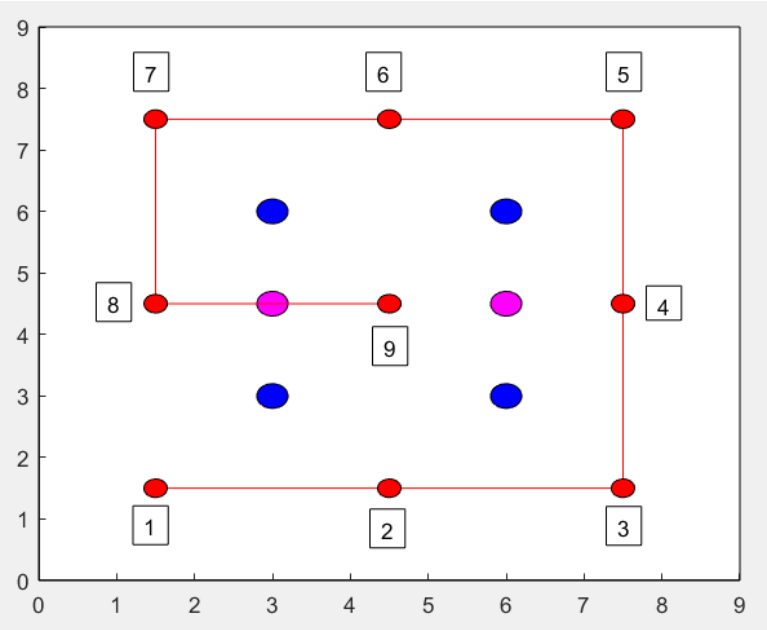


Figura 36 – *9 puntos del camino real.*

En la siguiente tabla se comparan los resultados obtenidos por el algoritmo del TFG contra los obtenidos en el TFM en las medidas en movimiento. Al igual que en la tabla anterior, la última columna muestra la mejora de estimación en metros que consigue el nuevo algoritmo frente al antiguo.

Medición	Error TFG	Error TFM	Mejora
1	*2,44m*	*1,13m*	*1,31m*
2	*0,67m*	*0,71m*	*-0,04m*
3	*0,94m*	*0,25m*	*0.69m*
4	*2,11m*	*0,96m*	*1,15m*
5	*3,02m*	*0,41m*	*2,61m*
6	*0,62m*	*0,47m*	*0,15m*
7	*3,24m*	*0,97m*	*2,27m*
8	*1,98m*	*1,03m*	*0,95m*
9	*0,21m*	*0,14m*	*0,07m*
Media	***1,69m***	***0,67m***	***1,02m***

Tabla 6 – *Comparación del error cometido por los dos algoritmos en dinámico.*

Los resultados de esta tabla confirman la mejora, que ya se podía apreciar visualmente, del algoritmo de este trabajo frente al algoritmo del TFG. El error del algoritmo de este TFM es menor en todos los puntos menos en el punto 2 teniendo un empeoramiento de 4 cm. La media de error del TFG es de 1.69 metros mientras que la media del TFM es de 0,67 metros, obteniendo así una mejora de 1,02 metros, o expresada porcentualmente una mejora del 60.36 %.

CONCLUSIONES Y LINEAS FUTURAS

En este sexto y último capítulo, es el momento de reflexionar sobre los resultados obtenidos y extraer las correspondientes conclusiones, así como las posibles modificaciones y mejoras que podrían dar continuidad a este trabajo.

5.6 Conclusiones

La finalidad de este trabajo era la de mejorar la precisión del algoritmo de posicionamiento en interiores realizado en mi TFG anterior. En el TFG se llegó a la conclusión de que el algoritmo era eficaz para estimar las medidas que se tomaban en una posición interna al rectángulo descrito por las balizas, pero que sin embargo resultaba ser ineficaz para la estimación que se hacía de las medidas tomadas fuera de este. Por este motivo en este trabajo se ha decidido diseñar un escenario, y tomar medidas en posiciones en las que este algoritmo sea ineficaz.

Se creía que el algoritmo fallaba en estas zonas debido a que fuera de este rectángulo la distancia entre las 3 balizas usadas para la trilateración era relativamente alta, y por eso se llegó a la conclusión que la solución a este problema era el de poblar aún más al escenario de balizas, cubriendo a este por completo.

Sin embargo, en este trabajo se ha planteado una nueva idea. No se descarta que esa sea una solución valida. Pero además del problema de la distancia, se cree que el método de hallar la constante de pérdidas de camino en la estancia, en la solución del TFG es ineficaz. En el TFG se realiza un estudio previo a la realización de la prueba, tomando varias medidas y haciendo una media de la constante "n" utilizando esta para la realización de todas las posteriores medidas y para el calculo de la distancia con todas las balizas por igual. Calcular de esta forma la "n" es erróneo, ya que no simula un entorno real donde las condiciones del aula varían con el tiempo, lo que provoca que la "n" no sea constante.

Por este motivo en este trabajo lo que se ha planteado es modificar esta parte del algoritmo, cambiando el análisis previo de la constante de pérdidas de camino, por un análisis continuo. Para ello se han empleado dos calibradores los cuales toman medidas de los beacons de las balizas en los mismos instantes en los cuales el individuo a localizar toma sus medidas. De esta forma se puede obtener el valor que "n" tendrá en el momento de cada medida, y no solo eso, sino que además se obtendrá un valor especifico y particularizado para cada una de las balizas. De esta forma se pensaba obtener una considerable mejora respecto al algoritmo anterior.

Para comprobar que este nuevo método que usa una calibración instantánea era mejor al método utilizado en el TFG, se han realizado 3 pruebas, cada una más compleja a la anterior pero cada vez más simulando una situación más realista.

La primera prueba se realizó con una sola baliza y consistió en estimar la distancia de esta al sujeto en estático por los dos métodos (Estudio previo de la "n" contra la

calibración instantánea). El error medio obtenido en esta prueba sin usar la calibración fue de 0,63 metros, mientras que usando la calibración se cometió un error de 0,22 metros. Esto quiere decir que usando la calibración se obtuvo una mejora del 65,08%.
Al ver la mejora que se consiguió empleando los calibradores para la estima de la distancia con una baliza, se decide implementar la calibración en el algoritmo y probarlo en las siguientes pruebas, comparando los resultados con los que se obtendrían con el método antiguo.

Las siguientes pruebas fueron realizadas en el mismo escenario y con el mismo despliegue de balizas. El escenario era una estancia de 9x9 metros y las balizas se colocaron en las esquinas del cuadrado central de la estancia de 3x3 metros.

La primera de estas pruebas donde se ha probado ya el algoritmo con la calibración consistía en tomar 9 medidas durante 10 segundos don de la mayoría de ellas las se tomaban fuera del cuadrado de 3x3 metros que forman las balizas. Se eligieron estas medidas externas porque es donde el algoritmo del TFG es menos eficaz y lo que se pretende con el nuevo algoritmo es obtener una mejora precisamente en esta área. Además, entre medida y medida se ha ido cambiando la disposición de los objetos de la sala, simulando de esta forma el posible cambio con el tiempo del entorno. Después de analizar las 9 medidas con los dos algoritmos, se puede concluir que, con el nuevo algoritmo, usando la calibración, se ha obtenido una mejora del 56,29% respecto al antiguo. Por otro lado, también se ha llegado a la conclusión que el método de la calibración instantánea se adapta mucho mejor a cambios en el entorno. Esto es debido gracias al estudio individualizado por medida y por baliza de la variable de pérdidas de camino o path loss por medio de los calibradores.

La última prueba que se realizó para la comparación de los dos algoritmos se realizó en la misma estancia y con el mismo despliegue. Sin embargo, en esta ocasión el sujeto a localizar se ira moviendo por el aula por un recorrido prefijado en el que como anteriormente la mayoría del tiempo se encontrara en la parte exterior al cuadrado formado por las balizas, tomando una única medida de 27 segundos. En este caso los dos algoritmos trocearan esta medida de 27 segundos en 9 medidas de 3 segundos, las cuales calcularan y unirán consecutivamente para hallar la estela del camino recorrido. Esta última prueba es la que simula una situación más realista. Fijándonos en los resultados obtenidos por cada uno de los algoritmos en esta prueba de toma de medidas en movimiento, se puede apreciar visualmente la mejora obtenida al introducir la calibración instantánea en el algoritmo. Para confirmar esta apreciación a simple vista, se han marcado 9 puntos en el camino real y se han comparado con los 9 puntos obtenidos por sendos algoritmos. Tras analizar los errores cometidos se obtiene una mejora del 60,36 % con el nuevo algoritmo.

En conclusión, este trabajo presenta una significante mejora de un algoritmo de posicionamiento en interiores, obteniendo de esta forma resultados prometedores y que aporta las bases para el desarrollo de nuevas aplicaciones que puedan hacer uso de la localización en interiores de una forma más precisa.

5.7 Líneas futuras

En cuanto a las líneas futuras se llega a la misma conclusión que en el trabajo del TFG. Estas tres vías de mejora que se deberían investigar son las siguientes:

- **Mejoras de los materiales**: En cuanto a la mejora de los elementos que componen el sistema, los módulos Bluetooth utilizados (MLT-BT05) vienen con bluetooth en su versión 4.0, y como se comentó en el apartado 2.5 sería muy interesante probar balizas con la nueva tecnología BT, con su versión 5.1, la cual trae muchas mejoras en cuanto a la solución para el problema de posicionamiento en interiores. En cuanto al uso de baterías, las tecnologías de almacenamiento de energía también evolucionan rápidamente, lo que hace posible no solo aumentar la autonomía de las balizas, sino que también se podrá ir reduciendo el tamaño de estas progresivamente, lo que permitirá hacer despliegues de balizas todavía más flexibles y duraderos.

- **Mejoras del error de estimación**: Los errores aún existentes en el mecanismo propuesto, son motivados, en su mayoría, por patrones de radiación muy variantes entre módulos idénticos, lo que se traduce en ruido adicional, al falsear los valores de referencia basados en la potencia recibida. Aunque en la propuesta de este trabajo se han conseguido reducir estos efectos respecto al algoritmo anterior; la implementación de filtros, estimadores y la mejora en los calibradores, junto con un correcto conocimiento de los patrones de radiación, permitirían estabilizar y mejorar las estimaciones.

- **Mejoras del servicio**: todos los cálculos realizados durante este trabajo se llevan a cabo en diferido. Esto significa que las medidas de señal de las balizas, tanto las medidas por el usuario como la de los calibradores, se obtienen con un móvil, y posteriormente se analizan en un equipo con el programa MATLAB. Sin embargo, el objetivo de tener una aplicación de posicionamiento en interiores debería ser el de obtener en tiempo real nuestra posición**.** Por este motivo los cálculos los debería realizar el dispositivo móvil al instante de tomar las mediciones o bien a través de un servidor web. Se deberían estudiar sendas opciones comprobando cuál de ellas seria la más eficiente y plausible.

6 BIBLIOGRAFIA

1. Calzada Cabano, Carlos. Análisis y mejora de un algoritmo de posicionamiento en interiores. 25-10-2018. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/15537

2. Singular team. Aplicaciones y soluciones para la geolocalización en interiores. https://blog.sngular.com/aplicaciones-y-soluciones-para-la-geolocalizacion-en-interiores

3. Toni Pérez Navarro. Posicionamiento en interiores (indoor positioning). 21-04-2016. Disponible en: http://informatica.blogs.uoc.edu/2016/04/21/posicionamiento-en-interiores-indoor-positioning/

4. José M. Llamas. Sistemas de geolocalización en interiores. 29-09-2016. Disponible en: https://blog.cartif.com/sistemas-de-geolocalizacion-en-interiores/

5. Marín Guerrero, Jorge. Sistemas de guiado para invidentes (S.G.I) basado en Bluetooth, J2ME y dispositivos telefónicos móviles. Capitulo 2. Disponible en: http://bibing.us.es/proyectos/abreproy/11972/fichero/Cap%C3%ADtulo+2+-+Bluetooth.pdf

6. Wikipedia. Bluetooth. Disponible en: https://es.wikipedia.org/wiki/Bluetooth

7. Argenox. Introduction to Bluetooth clasic. Disponible en: https://www.argenox.com/library/bluetooth-classic/introduction-to-bluetooth-classic/

8. Gutiérrez Reina, Daniel. Sistema pasarela Bluetooth para una red de sensores Zigbee. Disponible en: http://bibing.us.es/proyectos/abreproy/40048/fichero/VOLUMEN+1.+MEMORIA%252F4.+Tecnolog%C3%ADa+Bluetooth.pdf

9. Wikipedia. Trilateración. Disponible en: https://es.wikipedia.org/wiki/Trilateraci%C3%B3n#:~:text=La%20trilateraci%C3%B3n%20es%20un%20m%C3%A9todo,forma%20an%C3%A1loga%20a%20la%20triangulaci%C3%B3n.

10. Fernando Pareja. Líneas notables de un triángulo. Disponible en: https://fernandopareja.jimdofree.com/viejas-clases/05may/teoria/

11. John Leonard. Bluetooth 5.1 Puts Bluetooth In Its Place. 30-01-2019. Disponible en: https://blog.nordicsemi.com/getconnected/bluetooth-5.1-puts-bluetooth-in-its-place

12. Digi-Key's North American Editors. Use Bluetooth 5.1-Enabled Platforms for Precise Asset Tracking and Indoor Positioning - Part 1. 25-07-2019. Disponible en: https://www.digikey.com/en/articles/use-bluetooth-5-1-enabled-platforms-part-1

13. Ken Lam. Bluetooth 5.1 Direction Finding. 05-2019. Disponible en: https://www.bluetooth.com/wp-content/uploads/2019/05/BTAsia/1145-NORDIC-Bluetooth-Asia-2019Bluetooth-5.1-Direction-Finding-Theory-and-Practice-v0.pdf

14. Arduino. Guia Arduino. Disponible en: https://www.arduino.cc/

15. Martyn Currey. HM-10 Bluetooth 4 BLE Modules. 05-01-2017. Disponible en: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/

16. Fabio Leon. Baterías LiPo, características y cuidados!. Disponible en: https://www.dynamoelectronics.com/baterias-lipo-caracteristicas-y-cuidados/

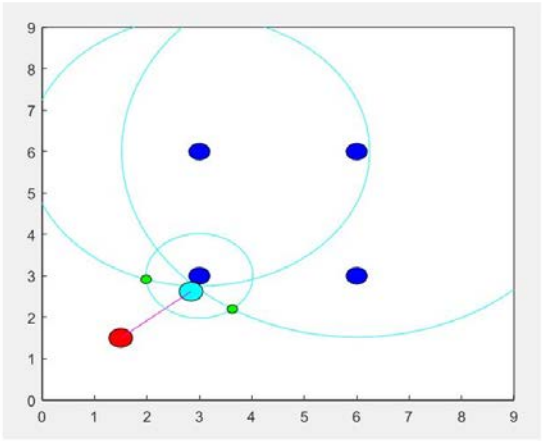
17. MathWorks. Foro sobre problemas MATLAB. Disponible en: https://es.mathworks.com

7 ANEXO

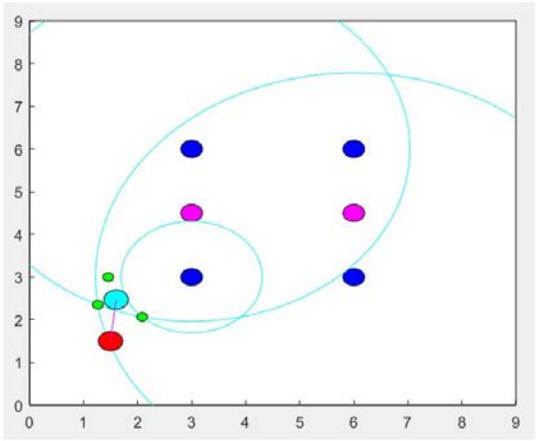
En este anexo se muestran los resultados de las pruebas en estático y en dinámico. En la columna izquierda se pueden ver los resultados obtenidos por el algoritmo del TFG y en la derecha los del algoritmo desarrollado en este trabajo usando el método de calibración instantánea. Encima de cada imagen pondrá en número de prueba y el error en distancia cometido por cada algoritmo.

7.1 Resultados en estático.

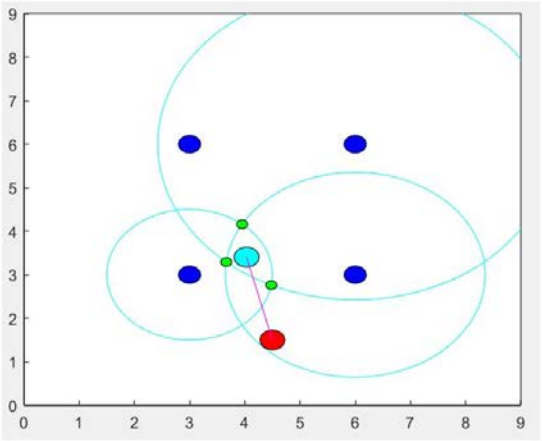
Medición 1, distancia 1,75m



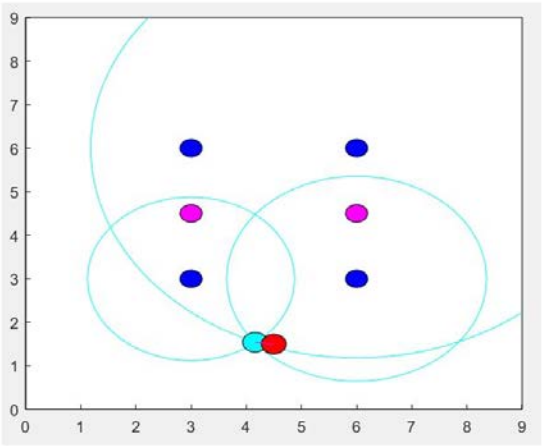
Medición 1, distancia 0,98m



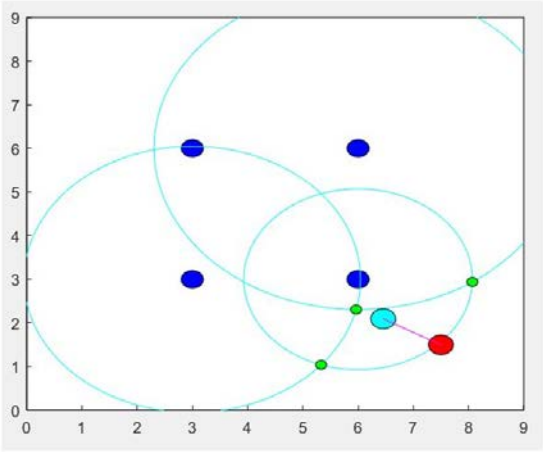
Medición 2, distancia 1,96m



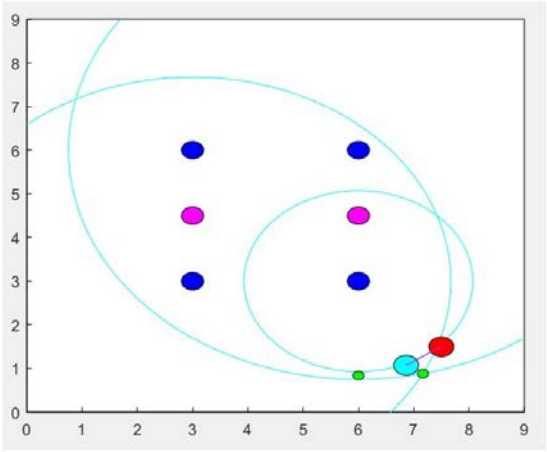
Medición 2, distancia 0,34m



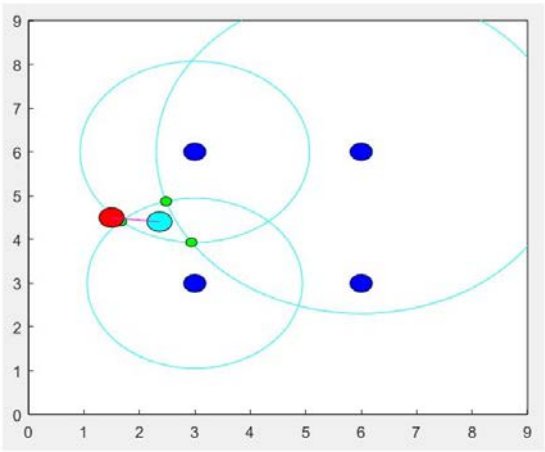
Medición 3, distancia 1,20m



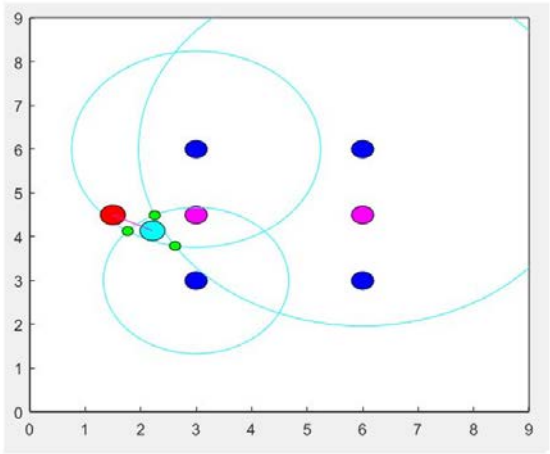
Medición 3, distancia 0,77m



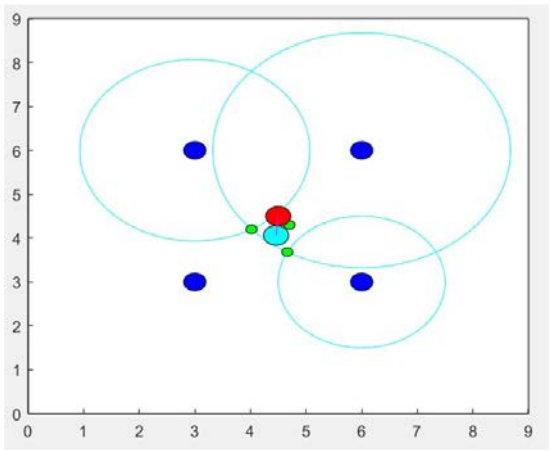
Medición 4, distancia 0,87m



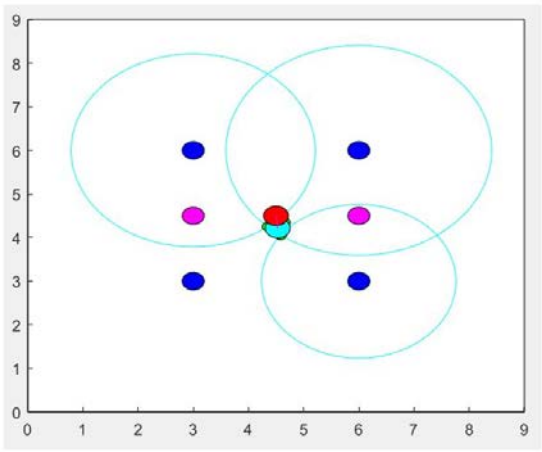
Medición 4, distancia 0,80m



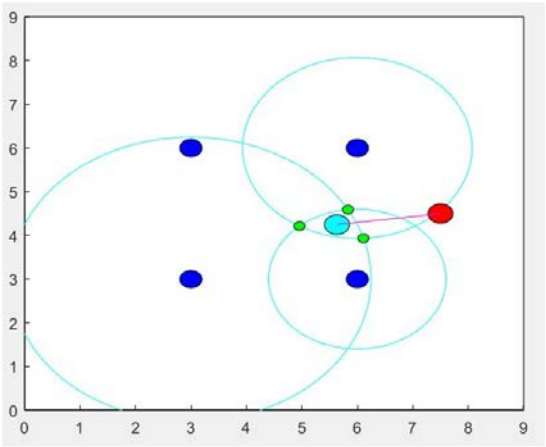
Medición 5, distancia 0,44m



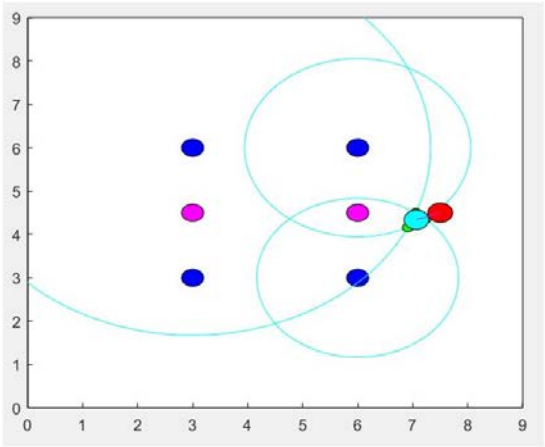
Medición 2, distancia 0,29m



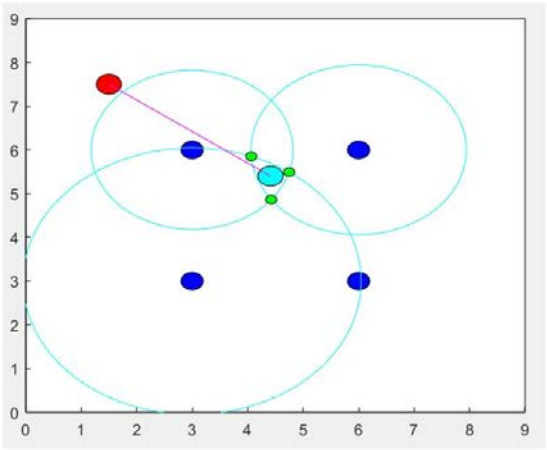
Medición 6, distancia 1,88m



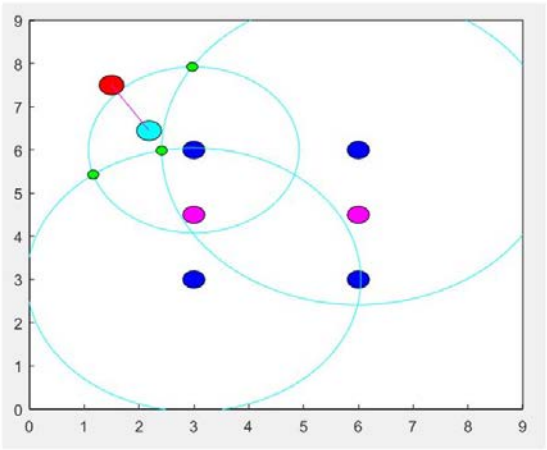
Medición 6, distancia 0,46m



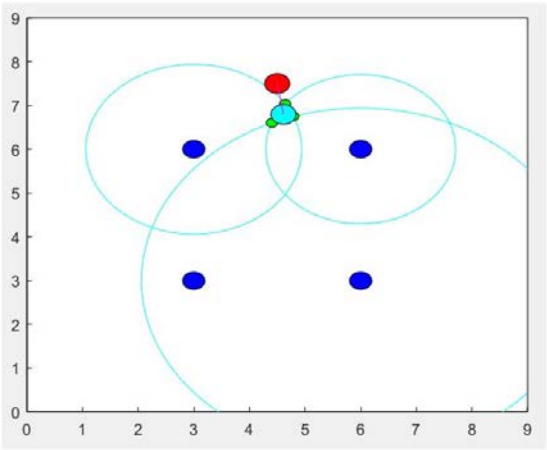
Medición 7, distancia 3,59m



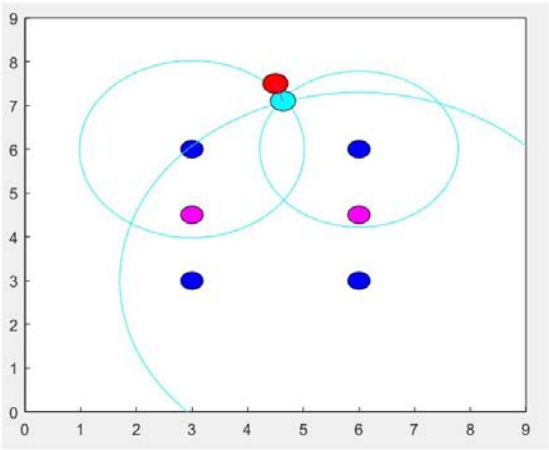
Medición 7, distancia 1,26m



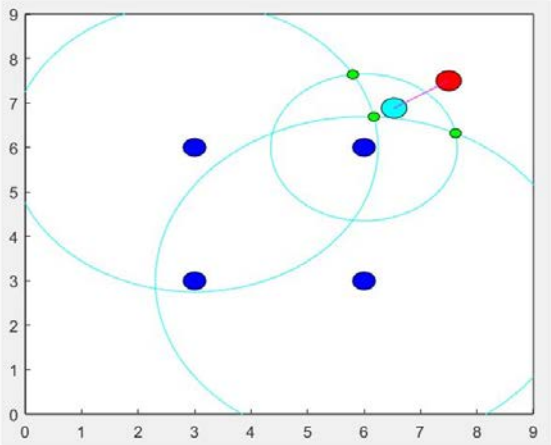
Medición 8, distancia 0,,71m



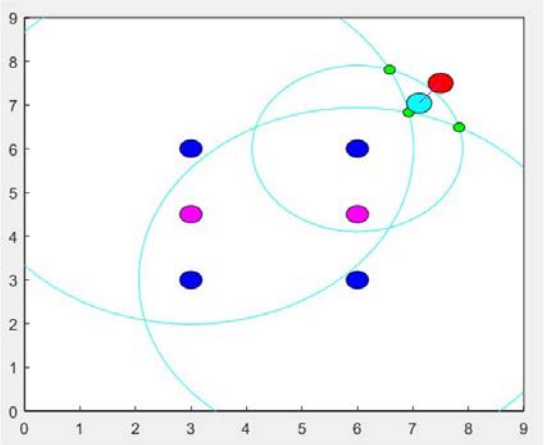
Medición 8, distancia 0,42m



Medición 9, distancia 1,15m



Medición 9, distancia 0,60m



7.2 Resultados en movimiento.

Ahora veremos los resultados que han estimado cada algoritmo cuando se toman medidas en movimiento. Las líneas rojas representan el camino real trazado por el usuario y las líneas verdes el camino estimado por sendos algoritmos. A la izquierda tenemos los resultados obtenidos por el algoritmo de TFG y a la derecha los resultados obtenidos con la mejora incorporada en este trabajo.