UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

ESTUDIO Y DESARROLLO DE UN SISTEMA DE LOCALIZACIÓN PRECISA DE ACTIVOS Y HUMANOS EN ESPACIOS CERRADOS PARA SEGURIDAD EN ENTORNOS CON INTERACCIÓN HUMANO ROBOT

AUTORA: MARÍA DEL MAR PLAZA CANO

TUTOR: ÁNGEL VALERA FERNÁNDEZ

COTUTOR: JUAN FRANCISCO BLANES NOGUERA

Curso Académico: 2020-21

AGRADECIMIENTOS

"A toda mi familia, por el apoyo y el cariño que he recibido para llegar hasta donde estoy ahora. Gracias por darme alas y confiar en mi en todo momento."

"A mis compañeros y amigos, por hacer de esta etapa un recuerdo inolvidable."

"A mis compañeros de laboratorio, por motivarme y ayudarme en todo momento."

"A Eduard, por la paciencia y las ganas que le pones a todo, gracias por guiarme y aconsejarme en este trabajo."

"A Paco, muchas gracias por darme esta increíble oportunidad. Espero estar a la altura."

"A Ángel, por introducirme en el mundo de la robótica y apoyar mis avances."

RESUMEN

Los sistemas RTL (Real Time Location) han permitido hasta el momento el desarrollo de aplicaciones de localización basadas en señales de radiofrecuencia con precisión suficientemente aceptable para la localización, aunque siempre por encima de las decenas de centímetros. Con la aparición de los sistemas de UWB (Ultra Wide Band), se están alcanzando precisiones teóricas por debajo de los 10 centímetros teóricos.

Este trabajo aborda la evaluación de un sistema UWB y su aplicación a la determinación de la posición de humanos en entornos industriales, para su integración en sistemas de seguridad de los entornos de trabajo. Estos sistemas permiten una mayor interacción humano robot, al tiempo que proporcionan información relevante sobre eficiencia de los sistemas productivos.

Actualmente tanto los robots, como los dispositivos IoT, son parte de la evolución de los sistemas de producción hacia la denominada Industria 4.0. La integración de datos desde todos los puntos de un proceso productivo (máquinas, humanos, productos), permite el control, la supervisión y el análisis, que son necesarios para lograr sistemas eficientes. En el caso de los sistemas IoT, el uso de brokers de comunicaciones con protocolo MQTT es una de las soluciones ampliamente y estandarizada. Por otro lado, el uso de la plataforma ROS (Robot Operating System), como middleware de interacción con robots, es también una solución ampliamente aceptada dentro de las aplicaciones de robótica.

Bajo este paradigma de Industria 4.0 tan demandado actualmente, se propone el presente Trabajo Fin de Máster, en tanto se persigue una aplicación realista y práctica de la propuesta. En esa línea el trabajo abordará las siguientes fases:

-Análisis de las tecnologías de localización mediante radiofrecuencia en interiores.
-Evaluación de un sistema de localización basado en UWB: precisión, alcance, comportamiento temporal y en entornos de producción.
-Diseño de un sistema software de localización de tags.
-Desarrollo de un módulo software para integración en la arquitectura ROS de un robot móvil: esta tiene como objetivo la integración a nivel de información de los datos sobre localización de robots y humanos.
-Desarrollo de controladores basados en información de localización: aplicación la generación de trayectorias seguras en entornos con interacción humano-robot.
-Desarrollo de módulo de evaluación de eficiencia: gestión de los datos de posición, análisis de las rutas y eficiencia en el uso de los recursos.
-Evaluación del sistema en entorno real: el trabajo se desarrollará en un laboratorio cuyo equipamiento permite la prueba y tests en situaciones muy cercanas a las reales que se podrían encontrar en un entorno productivo.

Palabras clave: UWB, RTL, Localización, Radiofrecuencia, Posicionamiento, Industria, Sistema de seguridad, HRI, Sistemas robotizados

RESUM

Els sistemes RTL (Real Time Location) han permés fins al moment el desenvolupament d'aplicacions de localització basades en senyals de radiofreqüència amb precisió prou acceptable per a la localització, encara que sempre per damunt de les desenes de centímetres. Amb l'aparició dels sistemes de UWB (Ultra Wide Band), s'estan aconseguint precisions teòriques per davall dels 10 centímetres teòrics.

Aquest treball aborda l'avaluació d'un sistema UWB i la seua aplicació a la determinació de la posició d'humans en entorns industrials, per a la seua integració en sistemes de seguretat dels entorns de treball. Aquests sistemes permeten una major interacció humà robot, al mateix temps que proporcionen informació rellevant sobre eficiència dels sistemes productius.

Actualment tant els robots, com els dispositius IoT, són part de l'evolució dels sistemes de producció cap a la denominada Indústria 4.0. La integració de dades des de tots els punts d'un procés productiu (màquines, humans, productes), permet el control, la supervisió i l'anàlisi, que són necessaris per a aconseguir sistemes eficients. En el cas dels sistemes IoT, l'ús de brokers de comunicacions amb protocol MQTT és una de les solucions àmpliament i estandarditzada. D'altra banda, l'ús de la plataforma ROS (Robot Operating System), com middleware d'interacció amb robots, és també una solució àmpliament acceptada dins de les aplicacions de robòtica.

Sota aquest paradigma d'Indústria 4.0 tan demandat actualment, es proposa el present Treball Fi de Màster, en tant es persegueix una aplicació realista i pràctica de la proposta. En aqueixa línia el treball abordarà les següents fases:

-Anàlisi de les tecnologies de localització mitjançant radiofreqüència en interiors.
-Avaluació d'un sistema de localització basat en UWB: precisió, abast, comportament temporal i en entorns de producció.
-Disseny d'un sistema programari de localització de tags.
-Desenvolupament d'un mòdul programari per a integració en l'arquitectura ROS d'un robot mòbil: aquesta té com a objectiu la integració a nivell d'informació de les dades sobre localització de robots i humans.
-Desenvolupament de controladors basats en informació de localització: aplicació la generació de trajectòries segures en entorns amb interacció humà-robot.
-Desenvolupament de mòdul d'avaluació d'eficiència: gestió de les dades de posició, anàlisi de les rutes i eficiència en l'ús dels recursos.
-Avaluació del sistema en entorn real: el treball es desenvoluparà en un laboratori l'equipament del qual permet la prova i tests en situacions molt pròximes a les reals que es podrien trobar en un entorn productiu.

Paraules clau: UWB, RTL, Localització, Radiofreqüència, Posicionament, Indústria, Sistema de seguretat, HRI, Sistemes robotitzats

ABSTRACT

RTL (Real Time Location) systems have so far allowed the development of location applications based on radio frequency signals with sufficiently acceptable accuracy for location, but always above tens of centimetres. With the emergence of UWB (Ultra Wide Band) systems, theoretical accuracies below 10 centimeters are being achieved.

This work deals with the evaluation of a UWB system and its application to the determination of the position of humans in industrial environments, for its integration into safety systems of working environments. These systems allow greater human-robot interaction, while providing relevant information on the efficiency of production systems.

Both robots and IoT devices are now part of the evolution of production systems towards Industry 4.0. The integration of data from all points of a production process (machines, humans, products), allows the control, monitoring and analysis that are necessary to achieve efficient systems. In the case of IoT systems, the use of communication brokers with MQTT protocol is one of the widely standardized solutions. On the other hand, the use of the ROS (Robot Operating System) platform as middleware for interaction with robots is also a widely accepted solution within robotics applications.

Under this Industry 4. 0 paradigm, which is so much in demand at the moment, this Master's Thread is proposed, while pursuing a realistic and practical application of the proposal. In this context, the work will address the following phases:

-Analysis of indoor radiofrequency location technologies.
-Evaluation of a localization system based on UWB: accuracy, range, temporal and production environment behaviour.
-Design of a tag localization software system.
-Development of a software module for integration into the ROS architecture of a mobile robot: this aims to integrate data on the location of robots and humans at the information level.
-Development of controllers based on location information: application to generate safe paths in environments with human-robot interaction.
-Development of an efficiency evaluation module: position data management, route analysis and resource efficiency.
-Evaluation of the system in a real environment: the work will be carried out in a laboratory equipped to allow testing and testing in situations very close to the real ones that could be found in a production environment.

Keywords: UWB, RTL, Location, Radiofrequency, Positioning, Industry, Security system, HRI, Robotic systems

ÍNDICE GENERAL

DOCUMENTOS CONTENIDOS EN EL TFM

- **Memoria**
- **Presupuesto**
- **Anexos**

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

ÍNDICE DEL DOCUMENTO ANEXO I

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

TABLAS DE LA MEMORIA

TABLAS DEL PRESUPUESTO

MEMORIA

CAPÍTULO 1. INTRODUCCIÓN

1.1. SITUACIÓN Y CONTEXTO

Con el crecimiento de la industria 4.0 y las tecnologías de la información aumenta el interés por la automatización de procesos para la optimización de los recursos. Humanos y robots deben convivir en las plantas de producción. Sin embargo, para lograr un crecimiento en los índices de rendimiento del personal es necesario asegurar unas condiciones de trabajo confortables y seguras, logrando un equilibrio entre las metas económicas y las motivaciones de los trabajadores.

Desde muchos años atrás existe una corriente de investigación dedicada al ámbito de la seguridad industrial, y concretamente, a los sistemas de posicionamiento en tiempo real (Real Time Location Systems ó RTLS) para lograr este equilibrio en las interacciones humano-robot (HRI) que se dan en la industria.

Existen diversas tecnologías de posicionamiento cuya base de funcionamiento pueden ser ondas de radio, ondas magnéticas, ondas acústicas, infrarrojos o satélites. No obstante, la tendencia actual es la implementación de sistemas de localización por ondas de radio en los espacios cerrados. Estos se dividen en los sistemas de comunicación por ondas de banda estrecha, como pueden ser Bluetooth o WiFi, y los de banda ultra ancha.

Por un lado, la llegada del Bluetooth 5.0 proporciona datos de posicionamiento de activos con mayor precisión que la tecnología WiFi pero con menor alcance. Sin embargo, en ambos casos las medidas obtenidas habitualmente son sobre la fuerza de la señal recibida que, en espacios cerrados con muchos objetos, puede provocar resultados poco precisos debido a las interferencias. En el mejor de los casos, las ondas de banda estrecha permiten obtener una posición con una precisión de algunos metros: de 1 a 5 metros con tecnología Bluetooth y de 5 a 10 metros con la tecnología WiFi [19].

Por el otro lado, la tecnología de banda ultra ancha (Ultra-wide-band) ha tomado una relevancia notable en el campo del posicionamiento en interiores debido a que permite, en la mayoría de los casos, medir la distancia de forma directa con diversos métodos de cálculo y no sólo mediante la fuerza de la señal recibida. Con esta tecnología se pueden lograr datos de posicionamiento con una precisión de centímetros. Además, presta mayor robustez en entornos donde la presencia de obstáculos entre emisor y receptor sea notable, pues las ondas de radio son pulsos de señal muy cortos en una banda de frecuencias ultra ancha que apenas interfieren en las señales de otras tecnologías de posicionamiento y pueden atravesar numerosas estructuras.

Las funcionalidades de los sistemas de posicionamiento en interiores son variadas y pueden dar respuesta a necesidades u oportunidades en un amplio rango aplicaciones. Algunas de las funcionalidades más habituales son las siguientes:

- **Seguimiento de usuarios y/o dispositivos en tiempo real.** Normalmente para el control de las plantas de producción, control de presencia, localización de zonas fuertemente transitadas en las diversas franjas horarias, etc.

- **Implementación de alertas por proximidad.** En ocasiones conviene saber cuándo un operario o un dispositivo está próximo a una zona delimitada para ejecutar una acción. Por ejemplo, en el ámbito de la seguridad industrial, si un trabajador se aproxima a una zona donde hay un robot industrial en movimiento, es deseable parar el robot.

- **Gestión de almacenes.** Una aplicación muy útil es la gestión automática de stocks en los programas de gestión de recursos actuales. Por ejemplo, cuando un dispositivo entre/salga del almacén se actualiza automáticamente su disponibilidad en la base de datos del programa. Otra aplicación puede ser la de localizar el dispositivo/material/objeto más cercano cuando es necesario en alguna operación de producción.

- **Optimización de procesos.** Por ejemplo, cuando un dispositivo entre en una determinada zona de producción, puede comenzar otro proceso a funcionar. Con esto se reducen los tiempos de espera en las estaciones de trabajo.

- **Enrutamiento y navegación.** Este tipo de aplicaciones son similares a las que ofrecen los sistemas de posicionamiento global. Trata de seleccionar la ruta más corta para ir de un punto a otro, pero en el interior de los edificios.

- **Estudio de patrones de comportamiento.** Actualmente se está invirtiendo mucho en investigar este ámbito, especialmente para el área de marketing. El objetivo es determinar el tiempo de permanencia que pasan los usuarios en unas determinadas zonas, ya sea para aumentar las ventas, mejorar las infraestructuras, modificar los procesos, etc.

1.2. MOTIVACIÓN

Este Trabajo de Fin de Máster surge de la necesidad de mejorar la seguridad en los entornos donde conviven humanos y robots en sus labores de producción dentro de la industria empleando sistemas de posicionamiento de interior (Indoor Positioning Systems ó IPS). Estos últimos años se está investigando mucho para la mejora de las tecnologías en este campo, destacando la tecnología UWB frente a Bluetooth o WiFi debido a su precisión en espacios cerrados. Sin embargo, tecnologías como GPS sigue representando un gran porcentaje de implementación en nuestros dispositivos de la vida diaria, a pesar de no ser precisa en interiores. En este proyecto se pretenden exponer las principales diferencias entre los sistemas de localización más comunes actualmente, profundizando en los sistemas de localización por tecnología UWB. Estos sistemas servirán de herramienta para el estudio de diversas funcionalidades aplicables en la industria, siempre con un enfoque de seguridad en el entorno de trabajo y/ó optimización de trayectorias en sistemas robotizados.

Por otro lado, la localización de activos en tiempo real incentiva el uso de interfaces de usuario para la monitorización y manejo de datos de forma sencilla (Human Machine Interfaces ó HMI's), pudiendo aglomerar numerosas funcionalidades en una única pantalla interactiva. Gracias a las numerosas librerías que existen en la actualidad para su diseño, es posible el desarrollo de interfaces hechas a medida de las necesidades que presenta cada aplicación. Uno de los objetivos que tiene este trabajo es crear una interfaz que conecte el sistema de localización con el espacio real de trabajo de forma sencilla. Esta debe proveer funciones prácticas como la obtención de los datos de posicionamiento de activos y humanos en el entorno de trabajo, así como la visualización y procesado de los mismos.

Para lograr grandes avances en el mundo de la industria 4.0, donde los sistemas robotizados y la gestión automatizada están en continuo crecimiento, es necesario invertir en herramientas que permitan prevenir los riesgos propios del trabajo interactivo entre humanos y robots. Un entorno de trabajo seguro propicia la productividad y la motivación de los trabajadores.

1.3. OBJETIVOS

Los objetivos principales de este trabajo son:

- **Análisis de las tecnologías de localización más comunes actualmente.** Analizar sus características, el principio de funcionamiento teórico, el alcance y la capacidad de cada una de ellas para la obtención de la posición de activos en espacios interiores.

- **Evaluación de un sistema de localización basado en la tecnología UWB.** Estudio de la precisión, el alcance y el comportamiento temporal en un entorno de producción real. Así como un breve análisis de sus algoritmos de cálculo del posicionamiento. Muestra de las pautas a seguir para la correcta instalación del sistema de localización en el laboratorio de trabajo, en función del hardware disponible.

- **Diseño de una interfaz de usuario para la localización de activos en tiempo real.** Diseño y programación de una herramienta que permita la interacción entre el usuario y el sistema de localización, permitiendo el procesamiento de los datos de posición y su visualización en tiempo real o a posteriori. Manejo de ficheros para facilitar el análisis de los datos, y comunicación con el robot móvil disponible en el laboratorio.

- **Desarrollo de un módulo software para integrar los datos del sistema de localización en la arquitectura ROS de un robot móvil.** El objetivo es integrar la información que provee el sistema de localización en el software del robot para su uso en distintas aplicaciones con interacción humano-robot.

- **Estudio de propuestas de mejora en seguridad industrial basadas en información de la localización de activos y robots.** Estudio de diversas funcionalidades de generación de trayectorias seguras, con evitación de obstáculos, aplicables en robótica móvil. Así como utilidades que podrían prevenir accidentes derivados de errores humanos. También posibles

técnicas de seguridad aplicables a robots industriales en las áreas de trabajo. Todas las aplicaciones probadas en un entorno de productivo real.

- **Estudio de propuestas de mejora en el trazado de trayectorias y optimización de rutas en robots móviles.** Gestión de los datos de la posición para el análisis de las rutas y la eficiencia en el uso de los recursos. Concretamente, optimización de rutas en entornos que presenten barreras imperceptibles por los sensores convencionales del robot móvil.

1.4. ESTRUCTURA DEL PROYECTO

En este primer capítulo del trabajo se ha tratado de contextualizar al lector en las tecnologías de localización actuales más comunes, y se han expuesto algunas de las razones que motivan el desarrollo de este trabajo. También se han descrito los objetivos principales que se pretenden lograr con este trabajo.

En el segundo capítulo se analizan algunas de las tecnologías de posicionamiento actuales, en base al alcance, precisión y capacidad de los sistemas, así como a sus especificaciones técnicas de funcionamiento. Se realiza además una comparación entre los sistemas descritos para poder valorar cuál es el que más se aproxima a las necesidades propias de este trabajo.

En el tercer capítulo se describe el sistema de localización empleado en este proyecto, estableciendo las pautas que son necesarias para su correcta instalación y funcionamiento. También se muestra el entorno donde se desarrolla el trabajo y las herramientas de las que se dispone para la utilización del sistema de localización en aplicaciones industriales.

En el cuarto capítulo se explica la interfaz gráfica de usuario que ha sido diseñada y programada en Python especialmente para la utilización del sistema de localización en el entorno de trabajo descrito. Se desarrollan las funcionalidades implementadas y se esboza brevemente la estructura del código que se tiene en el Anexo I del documento.

En el quinto capítulo se realizan diversos experimentos para medir la precisión que alcanza el sistema de localización implementado en el entorno de trabajo. Este capítulo sirve de base para tener una visión realista de lo que se puede lograr con la utilización de este sistema de localización en aplicaciones de la industria 4.0.

En el sexto capítulo de la memoria se desarrollan algunas de las aplicaciones que podrían mejorar la seguridad en el entorno de trabajo y/o la optimización de trayectorias para la reducción de tiempos en las cadenas de producción en aquellos entornos que dispongan de robots móviles en su área logística. Se explica cómo se pueden incluir los datos obtenidos por el sistema de localización en los robots, y cómo deben manejarse estos datos para incrementar los beneficios que aportan los robots en la industria.

En el séptimo capítulo se realizan unas conclusiones en base a los objetivos alcanzados del proyecto y se establecen unas líneas de trabajo futuras que podrían continuar el estudio realizado hasta el momento.

En el último capítulo se muestran las referencias bibliográficas que han sido necesarias para profundizar en algunos conceptos de este trabajo.

CAPÍTULO 2. SISTEMAS DE LOCALIZACIÓN

2.1. LOCALIZACIÓN MEDIANTE BLUETOOTH

El término Bluetooth describe una tecnología de red desarrollada por el grupo de trabajo IEEE 802.15.1 como estándar industrial para conexiones inalámbricas de área personal (WPAN)**.** La frecuencia dedicada a Bluetooth es una banda ISM sin licencia entre los 2,402 GHz y los 2,480 GHz. Esta tecnología sirve para transmitir datos en distancias cortas (hasta los 10 metros aproximadamente), así como el establecimiento de conexiones sencillas y de bajo consumo. No obstante, presenta velocidades bajas de transferencia de datos. Este sistema de comunicaciones presenta una estructura de maestro-esclavo, formando una red conocida como "piconet". Esta red puede integrarse por ocho dispositivos Bluetooth activos como máximo.

En distancias cortas, se puede emplear el Bluetooth para estimar la distancia entre dos dispositivos midiendo la intensidad de señal que les llega, con precisiones de 3 a 5 metros. Sin embargo, con la llegada del Bluetooth 5.1 se puede conocer, además, la dirección de la que provienen las señales de cada dispositivo. Esto se consigue midiendo el ángulo con el que llega la señal, y varía el método de cálculo en función del número de antenas Bluetooth disponibles en emisor y receptor.

- **Ángulo de llegada**: en este método, un dispositivo receptor compuesto por una matriz de antenas rastrea los ángulos de llegada de objetos individuales midiendo la diferencia de fase de señal que se da con la diferencia de distancias.
- **Ángulo de salida**: en este método, el dispositivo receptor calcula su propia posición en el espacio usando ángulos de múltiples radiofaros conociendo sus posiciones.

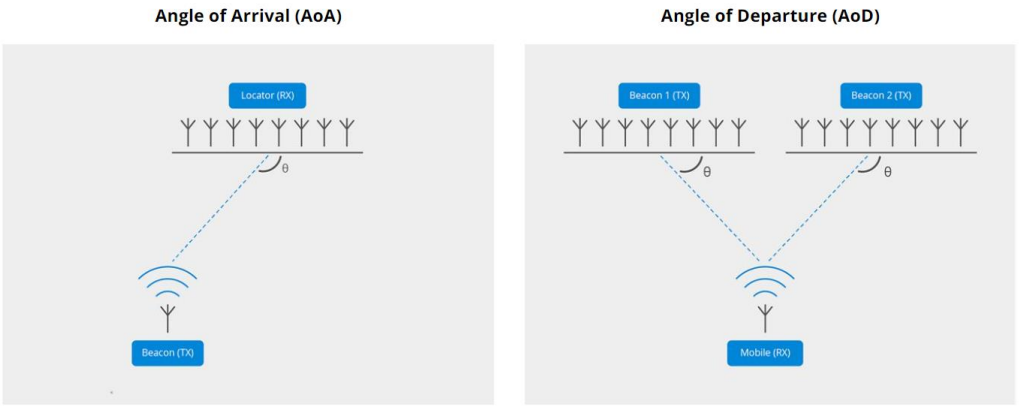


Ilustración 1 Métodos de cálculo de la distancia a partir de la medida de la intensidad de señal [1]

Esta versión permite optimizar las soluciones de localización en la industria con una precisión de centímetros. No obstante, es una tecnología muy reciente y apenas comercializada actualmente.

2.2. LOCALIZACIÓN MEDIANTE WIFI

La tecnología de red WiFi describe una red inalámbrica local con el estándar IEEE 802.1 que usa ondas de radio en una banda de frecuencias de 2.4 GHz a 5 GHz para comunicar datos entre dispositivos o desde Internet. Con la tecnología WiFi se pueden transferir grandes cantidades de información con velocidades de transferencia de hasta varios gigabits por segundo para las versiones más actuales (estándar IEEE 802.11ac) [2]. Además, tiene un alcance de hasta 100 metros.

El empleo de esta tecnología en la geolocalización en interiores requiere como mínimo tres puntos de acceso WiFi para leer la posición en el plano o cuatro si se desea conocer el posicionamiento a nivel tridimensional. Estos puntos de acceso se comunican entre sí, permitiendo que el dispositivo a localizar pueda calcular su ubicación midiendo la velocidad de las ondas de radio que le llegan.

Ilustración 2 Sistema de localización mediante WiFi [3]

Las señales WiFi viajan a través de aire a la velocidad de la luz, por tanto, la distancia se calcula como el tiempo que tarda la transmisión de la señal desde un punto de acceso hasta llegar al dispositivo móvil por la velocidad de la luz. Para determinar la posición exacta, es fundamental sincronizar el reloj de ambos dispositivos a nivel de nanosegundos, esto se hace mediante una medición de ida y vuelta [3]. Cuando el punto de acceso envía una señal marca su hora de salida T1. Esta señal es recibida por el cliente y marcada con la hora de llegada T2, entonces el cliente la envía de vuelta indicando el nuevo tiempo de envío T3 hasta que es recibido por el punto de acceso inicial en el tiempo T4 (Ver ilustración 3).

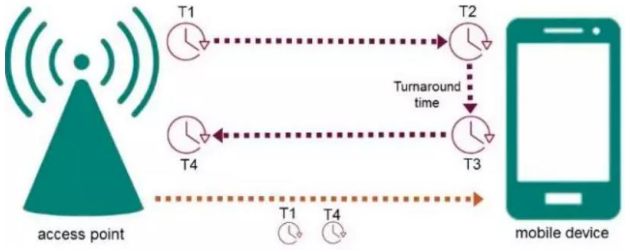


Ilustración 3 Sistema de localización mediante WiFi (Tiempos) [3]

Una de las principales desventajas de WiFi es su precisión. Si la red Wi-Fi empleada para la localización tiene mucho tráfico de datos, los paquetes no tendrán la máxima velocidad de transmisión. Otro aspecto muy importante es el problema del Multipath, cuando se tiene un entorno con muchos obstáculos y/o materiales reflectantes se pueden producir interferencias, la señal recibida puede no provenir de forma directa del emisor provocando errores en las mediciones de tiempos de transmisión.

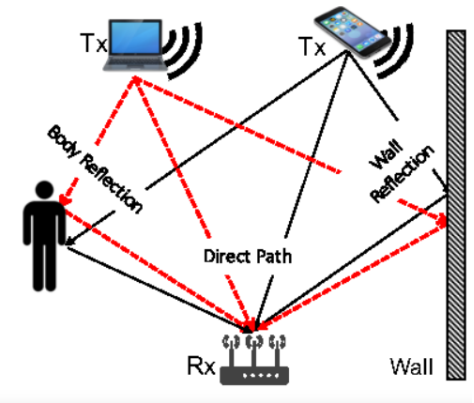


Ilustración 4 Interferencias por Multipath en los dispositivos WiFi [4]

La tecnología WiFi es una buena opción para la geolocalización en interiores que puede alcanzar precisiones de hasta 1 metro. Sin embargo, requiere una gran inversión en la infraestructura de red para lograr dicha precisión, pues aumenta con la cantidad de puntos de acceso disponibles.

2.3. LOCALIZACIÓN MEDIANTE GPS

El GPS o Sistema de posicionamiento global es un sistema que permite obtener la posición de un dispositivo mediante el uso de satélites. Este sistema está formado por una constelación de 24 satélites dispuestos en seis órbitas, lo que se conoce como NAVSTAR (Navigation satellite timing and ranging). Cada satélite está dotado de cuatro relojes atómicos de gran exactitud, pues el tiempo es muy importante para establecer la posición.

Para el cálculo de la localización se debe tener cobertura de al menos tres señales, siendo necesaria una cuarta para determinar la altura. Los satélites envían señales constantemente hacia la Tierra, donde son recibidas por antenas de GPS. Estos equipos miden el tiempo transcurrido entre la emisión y la recepción, luego es fundamental que ambos estén sincronizados. Ese tiempo, multiplicado por la velocidad de la luz, permite calcular la distancia hasta cada uno de los satélites. De esta forma se conoce dónde está cada emisor y la distancia hasta el usuario.

Con estos datos se puede establecer la posición como el lugar geométrico donde confluyen las tres esferas, con centro en cada uno de los satélites, y como radio la distancia calculada. [5]

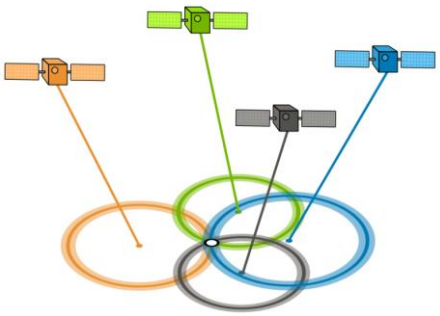


Ilustración 5 Sistema de localización por GPS [5]

No obstante, esta posición puede verse afectada por la posible desincronización entre los relojes de emisor y receptor, así como por las perturbaciones atmosféricas [5]. El Sistema de Posicionamiento Global Diferencial, DGPS, se ocupa de corregir los posibles errores en las órbitas satelitales y afinar la posición aportada por el GPS.

Este sistema está compuesto por una serie de estaciones terrestres fijas, para las que se conoce su ubicación exacta. De esta forma se puede comparar la posición real con la que proporciona el GPS y establecer el error de posición en cada momento. El error calculado se emite a las antenas de usuario que lo utilizan para corregir ese error en el posicionamiento con un margen máximo de 10 metros. [6]

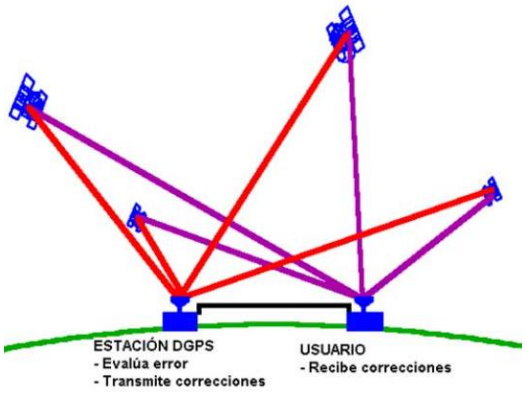


Ilustración 6 Sistema de Posicionamiento Global Diferencial [6]

La principal desventaja del GPS para la localización es que no funciona muy bien en espacios interiores., puesto que las señales emitidas por los satélites no logran traspasar las estructuras de los edificios.

2.4. LOCALIZACIÓN MEDIANTE UWB

La tecnología UWB es una de las más recientes y prometedora actualmente. Está basada en la transmisión de pulsos cortos de señal (del orden de nanosegundos) en una banda muy ancha de frecuencias [7]. Esto permite que las señales traspasen fácilmente obstáculos y estructuras, reduciendo el efecto Multipath. Además facilita la medida del tiempo de llegada de la señal en el receptor, lo que hace que esta tecnología sea idónea para el cálculo preciso de posicionamiento [8]. Con esta tecnología se logran velocidades de transferencia de datos de hasta 100 Mb/s.

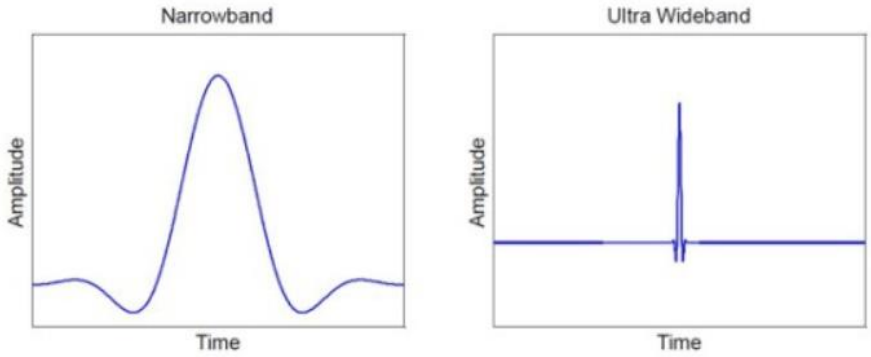


Ilustración 7 Señal común y señal UWB [8]

Por otro lado, los niveles de potencia radiada en los chips UWB son muy bajos, del orden de medio mW frente a las varias centenas de mW de Bluetooth y decenas de mW de WiFi. Esto permite que la autonomía de los dispositivos donde es integrado el chip UWB no se vea afectado prácticamente. Puesto que además esta energía se debe distribuir sobre un ancho de banda muy grande, la densidad espectral de dicha energía es muy pequeña inferior a los 100 nW/MHz, lo que evita interferencias con otras señales que estén utilizando dicha porción del espectro. [9]

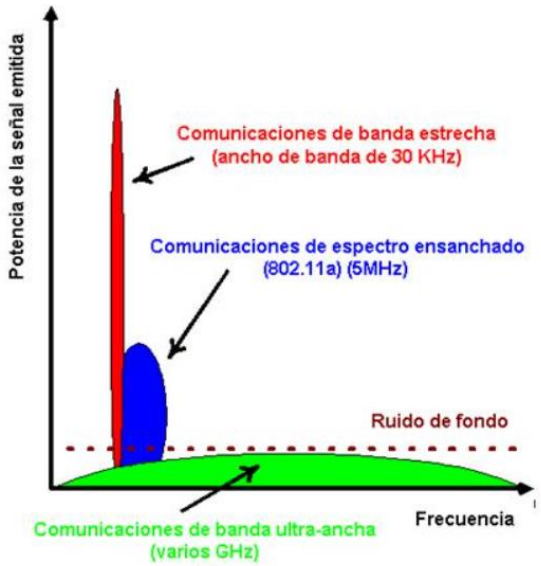


Ilustración 8 Potencia de la señal emitida frente a la frecuencia para varios tipos de señal [9]

Para utilizar esta tecnología se han desarrollado distintos algoritmos de posicionamiento, los cuales se puede clasificar en cinco categorías: hora de llegada (TOA), ángulo de llegada (AOA), intensidad de la señal recibida (RSS), diferencia de tiempo de llegada (TDOA) e híbrido. Estos se van a explicar brevemente en los siguientes subapartados.

2.4.1. Algoritmo TOA

Con este algoritmo la distancia desde el objetivo móvil a la unidad de medida es directamente proporcional al tiempo de propagación. Como se puede observar en la figura 9, la distancia se calcula a partir de la intersección de circunferencias de señal de múltiples transmisores, midiendo el tiempo

de propagación de la señal de cada transmisor en la dirección de la intersección. Para ello, es necesario sincronizar minuciosamente los relojes de los transmisores, no siendo necesaria la sincronización en el receptor. [7]

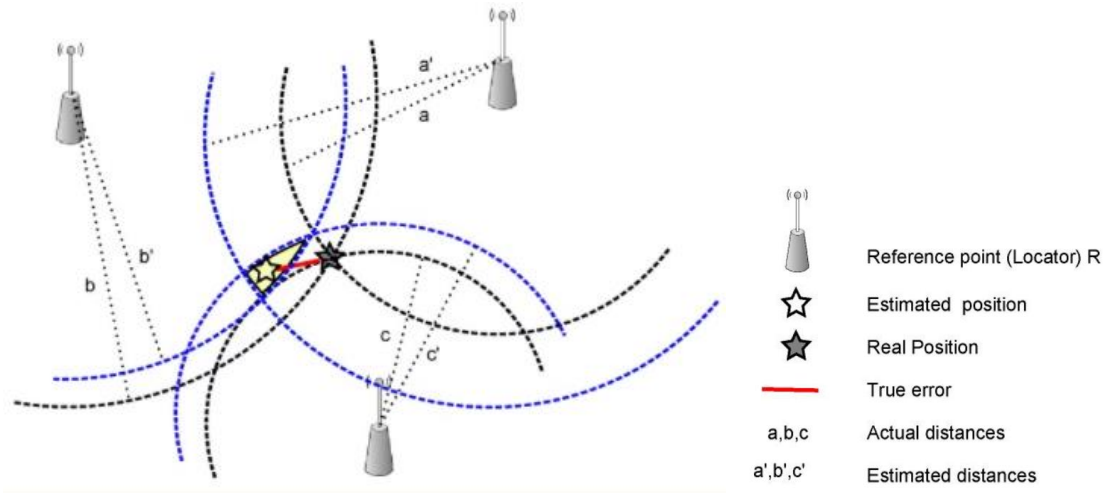


Ilustración 9 Algoritmo Time of Arrival (TOA) [7]

2.4.2. Algoritmo AOA

Este algoritmo consiste en estimar la posición a partir de la intersección de los ángulos de recepción de la señal de cada transmisor, siendo necesarios dos como mínimo. Estos ángulos se obtienen de comparar la amplitud de la dicha señal o la fase en los diferentes transmisores. Este algoritmo se suele usar de manera conjunta con otros métodos para mejorar su precisión, que puede verse alterada por muchos factores.

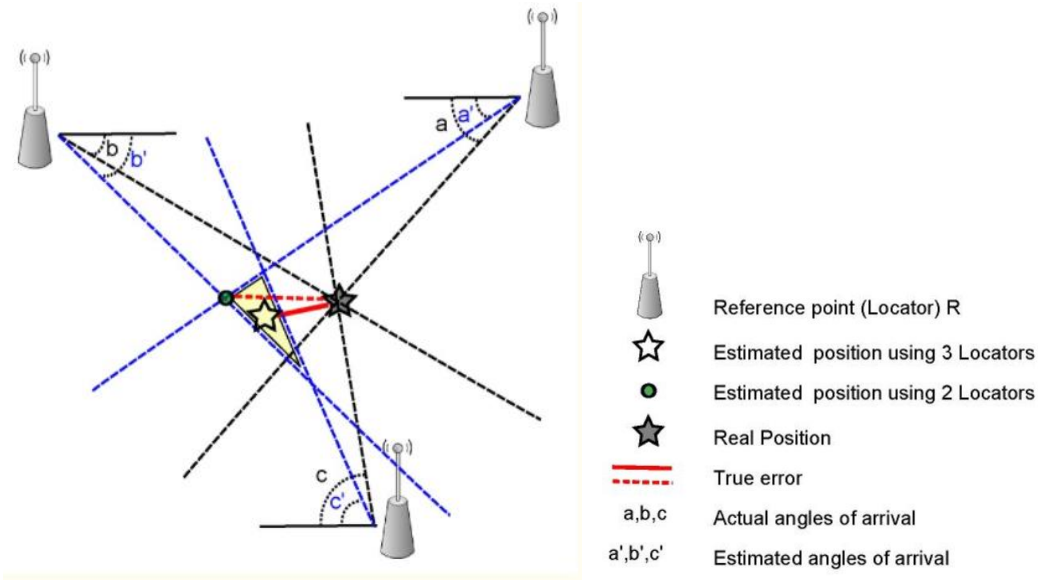


Ilustración 10 Algoritmo Angle of Arrival (AOA) [7]

2.4.3. Algoritmo RSS

En los algoritmos basados en RSS, el dispositivo receptor mide la intensidad de la señal para las señales recibidas de múltiples transmisores. De esta manera, el receptor podrá estimar su posición en relación con los nodos transmisores en base a dicha intensidad de la señal [7].

Este método es muy sensible al efecto Multipath, y consta de baja precisión en entornos donde no se puede lograr la línea directa de visión (Line of Sight), luego no es del todo adecuado para el cálculo de posicionamiento en interiores.

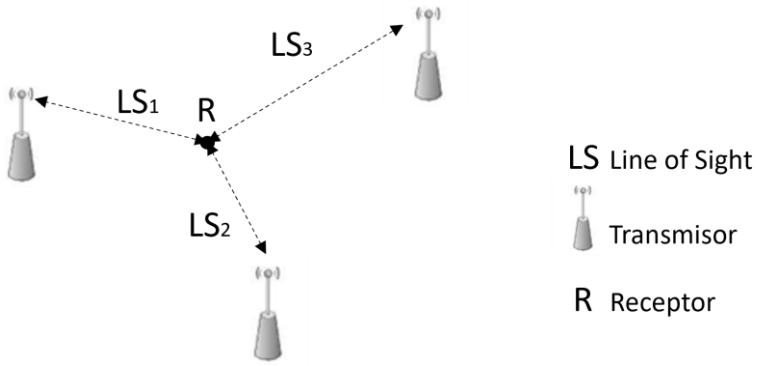


Ilustración 11 Algoritmo RSS (Received Signal Strength)

2.4.4. Algoritmo TDOA

Este algoritmo calcula la distancia al objeto o tag midiendo la diferencia en los tiempos de llegada de las señales a las antenas. Así, el tag envía un mensaje de sondeo que es recibido por las antenas cercanas. Estas registran el instante de llegada del mensaje, siendo distinto para cada una de ellas dado que su posición es diferente y a cada una le llega en un instante distinto. Estas diferencias de tiempo se emplean para determinar la ubicación del tag [10].

Para ello, hay que garantizar la cooperación entre los receptores, pues se requiere la compartición de los datos para determinar la ubicación del objeto. Esta cooperación requiere un ancho de banda superior al resto de algoritmos. [7]

Este método es óptimo en espacios amplios libres de obstáculos, puesto que con una medida única es posible determinar la posición del objeto. También para aplicaciones que requieran una alta tasa de actualización de coordenadas, como, por ejemplo, en deportes de competición.

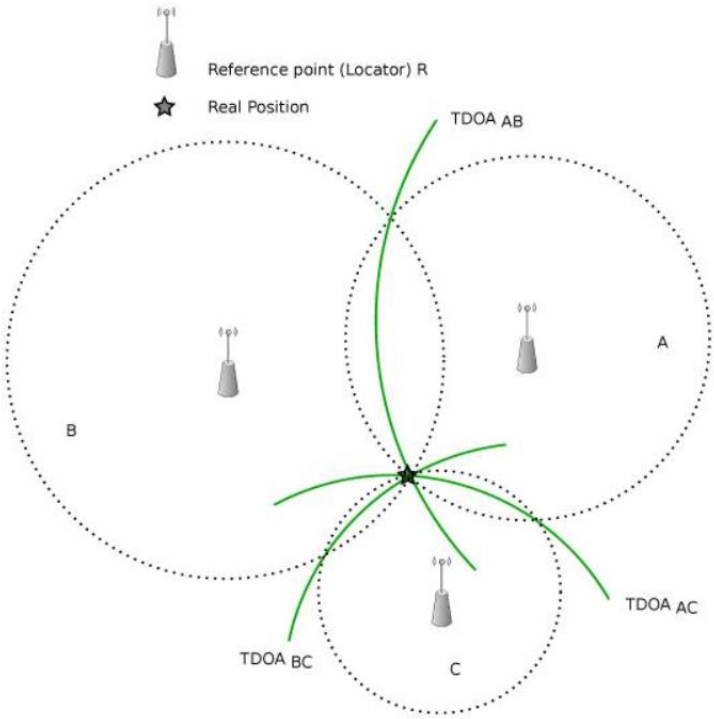


Ilustración 12 Algoritmo Time Difference of Arrival (TDOA) [7]

2.4.5. Algoritmo TWR

En este algoritmo se intercambian varios mensajes entre el dispositivo a localizar y el transmisor, en este caso, tag y antena ('anchor'). Por tanto, se le conoce como algoritmo de dos vías ('Two Way Ranging'). Según se muestra en la ilustración 13, el tag envía un paquete en el que se registra el tiempo de salida como T_{SP}, que es recibido y registrado por la antena en el instante T_{RP}. El tiempo que tarda la antena en procesar el paquete, registrar la marca de tiempo y generar un paquete de respuesta se define como T_{RSP}. Una vez generado, la antena vuelve a enviar el paquete al tag registrando el nuevo tiempo de salida como T_{SR}. El tag recibe la respuesta en el instante T_{RR}, y tarda en procesarlo y generar un último mensaje un tiempo T_{RSP}. Por último, el tag envía el mensaje final en el instante T_{SF}, que es recibido por la antena en el instante T_{RF}.

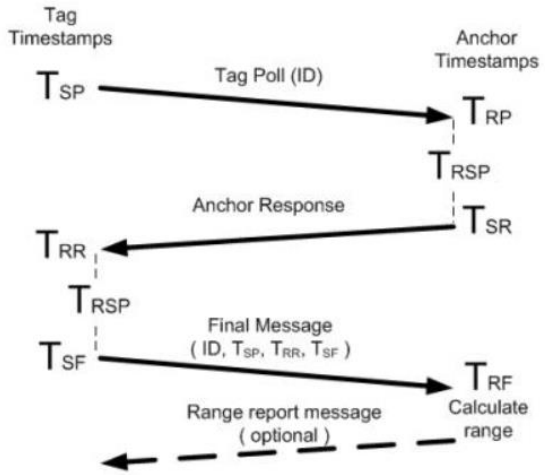


Ilustración 13 Algoritmo Two Way Ranging (TWR). Tiempo de vuelo de la señal [10]

A partir de estos tiempos, la antena puede calcular el tiempo total de vuelo ('Time of Flight' o ToF) mediante la siguiente fórmula:

$$\mathrm{ToF} = \frac{(T_{RR}-T_{SP})-(T_{SR}-T_{RP})+(T_{RF}-T_{SR})-(T_{SF}-T_{RR})}{4} = \frac{(T_{RR}-T_{SP})+(T_{RF}-T_{SR})-2T_{RSP}}{4} \quad (Ec.\ 1)$$

El tag debe realizar este proceso con cada una de las antenas, calculando para cada antena una distancia determinada (producto de la velocidad de la luz por el tiempo de vuelo). En el plano basta con tener tres antenas, pero en el caso tridimensional serían necesarias cuatro antenas. Por último, se traza una circunferencia por cada antena con el radio la distancia calculada, y el punto donde intersectan dichas circunferencias es la posición estimada del tag.

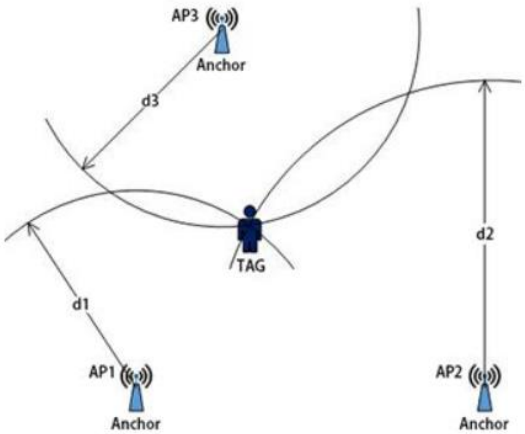


Ilustración 14 Algoritmo TWR (Two Way Ranging). Determinación de la posición [10]

Este método es muy preciso en interiores, ya que al enviar varios mensajes se pueden corregir las desviaciones producidas por el efecto Multipath y las interferencias. También es especialmente útil en aplicaciones en las que la posición debe estar disponible en el propio tag, o donde la duración de la batería no tiene tanta importancia, y en aplicaciones donde existe la posibilidad de que el tag se salga del área de posicionamiento de las antenas.

2.5. COMPARATIVA

En este apartado se realiza una comparación de los sistemas de localización descritos en los apartados anteriores en base a unas determinadas características. Para extraer los datos que se muestran en la tabla siguiente se ha tomado como referencia bibliográfica la [18].

	Bluetooth	**WiFi**	**GPS**	**UWB**
Entorno	Exterior / Interior	Exterior / Interior	Exterior	Exterior / Interior
Precisión	1-5 m	1-15 m	5-20 m	5-30 cm
Banda de frecuencia	2.4 GHz	2.4 GHz - 5 GHz	1.6 GHz	3.1-10.6 GHz
Cobertura	<100 m	<150 m	--	<250 m
Transferencia de datos	Hasta 20 Mb/s	Hasta 1 Gb/s	--	Hasta 100 Mb/s
Latencia	>3 seg	>3 seg	100 mseg	< 1 mseg
Escalabilidad	<1000 tags	<1000 tags	Ilimitada	>10000 tags
Consumo de batería	Bajo	Alto	--	Bajo
Fiabilidad	Muy sensible al Multipath, obstáculos e interferencias	Muy sensible al Multipath, obstáculos e interferencias	Muy sensible a los obstáculos	Poco sensible al Multipath e interferencias

Tabla 1 Tabla comparativa de los sistemas de localización descritos

Por tanto, la mejor opción para la localización de activos en un entorno de trabajo cerrado con numerosos obstáculos y materiales reflectantes es el sistema UWB. Este sistema cuenta con gran precisión en las medidas de posicionamiento en tiempo real en comparación con las otras tecnologías. Además, tiene suficiente cobertura y el consumo de batería de los tags es bajo.

Todas estas características hacen que este sistema sea el óptimo para las aplicaciones de la industria 4.0, y en concreto, para los objetivos que se persiguen en este proyecto.

CAPÍTULO 3. SISTEMA DE LOCALIZACIÓN IMPLEMENTADO

Como se ha visto, la mejor opción para medir el posicionamiento de activos en espacios cerrados es un sistema que emplee la tecnología UWB. Por tanto, en este apartado se va a describir el sistema de localización por UWB del que se dispone en el laboratorio de trabajo, así como el entorno de trabajo.

3.1. ENTORNO DE PRUEBAS

Este trabajo se ha realizado en el Laboratorio de industria 4.0 perteneciente al Instituto Universitario de Automática e Informática industrial (Ai2) de la Universidad Politécnica de Valencia.

El laboratorio tiene una superficie de 125 m^2 y alberga soluciones para la industria actual informatizada, como son robots industriales, robots colaborativos o células de producción automatizada. En este espacio de trabajo se instaló el sistema de localización de Pozyx para el seguimiento en tiempo real de la localización de activos. Este entorno es ideal para realizar los experimentos del presente trabajo, puesto que simula un entorno real industrializado de una fábrica de producción, donde puede ser interesante monitorizar la posición de los activos por temas de seguridad y/o optimización de sus actividades.

Ilustración 15 Laboratorio de Industria 4.0 del Ai2

3.2. SISTEMA DE LOCALIZACIÓN UWB DE POZYX

En este apartado se va a describir el sistema de localización mediante Ultra-Wide-Band (UWB) de Pozyx instalado en el laboratorio de industria descrito, así como las pautas para su correcta instalación.

3.2.1. Componentes

El sistema está constituido por tres partes diferenciadas.

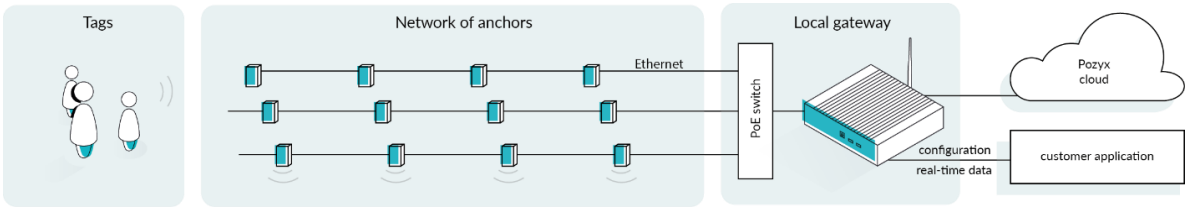


Ilustración 16 Componentes del sistema de posicionamiento de Pozyx [14]

- En primer lugar, **una red de antenas y tags** que conforman la estructura principal del sistema de comunicación para obtener el posicionamiento en tiempo real de personas u objetos móviles. Los tags son dispositivos móviles de los cuales se desea conocer la posición, mientras que las antenas son dispositivos que se fijan en una posición determinada y reciben la información emitida por los tags.
- En segundo lugar, **un Gateway**. Este dispositivo es el servidor de posicionamiento y conecta las antenas mediante Ethernet. Asimismo, recopila los datos que recibe de las antenas y los convierte en posiciones. Permite la conexión con la aplicación web de POZYX.
- Por último, **la interfaz web**. En ella se pueden configurar los dispositivos y visualizar en tiempo real las trayectorias seguidas por los tags que estén conectados al servidor.

3.2.2. Métodos de cálculo del posicionamiento

Los dos métodos de cálculo de posicionamiento disponibles en este sistema de localización de Pozyx son el método TDOA y el método TWR, descritos en los apartados 2.4.4 y 2.4.5. Cabe destacar que Pozyx está actualmente desarrollando un tercer método, aún no disponible, llamado reverse TDOA, que no será comentado en este apartado ya que no se han podido realizar ningún tipo de pruebas.

A continuación, se exponen las ventajas y las desventajas de ambos métodos en una tabla.

	TDOA	**TWR**
Capacidad del sistema	Muy alta	Limitada
Consumo de energía	Muy bajo	Alto
Requisitos del entorno	Altos	Bajos
Requisitos de sincronización	Altos	Bajos
Disponibilidad de la información	Servidor	Servidor y tag

Tabla 2 Tabla comparativa de los métodos de cálculo de posicionamiento disponibles en el sistema de localización de Pozyx (TDOA y TWR)

La capacidad del sistema hace referencia al número de tags activos permitidos para la determinación de la posición. Puesto que el método TWR requiere una comunicación bidireccional, debe tener un número limitado de tags activos. Asimismo, el consumo de batería en este método es considerablemente mayor que en el método TDOA, debido a que se envían y se reciben impulsos de manera periódica mientras que en el segundo método solamente se envían. Por esta misma razón, la información de la posición también está disponible en el tag para el método TWR, no siendo posible con el otro método.

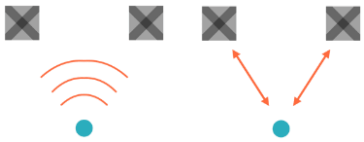


Ilustración 17 Comunicación unidireccional (TDOA) y bidireccional (TWR) [14]

Por otro lado, para obtener buena precisión con el método TDOA es necesario que el tag esté dentro del área que forman las antenas, además empeora con la aparición de obstáculos que puedan generar interferencias en las señales emitidas por el tag, por ejemplo, objetos metálicos. Esto no ocurre con el método TWR, que gracias a su sistema de comunicación bidireccional puede compensar estos errores manteniendo una buena precisión en el posicionamiento.

Por último, hay que tener en cuenta que el método TDOA requiere que las antenas estén perfectamente sincronizadas para lograr buena precisión en los cálculos, sin embargo, no es determinante con el método TWR.

3.2.3. Tipos de tags

El sistema cuenta con varios tipos de tags, de los cuales los que se van a emplear son el tag de desarrollo y el tag portable. En la siguiente tabla se muestran sus características:

	MINITAG	**TAG PORTABLE**	**TAG DE DESARROLLO**
			
Conexión	NFC	NFC	Micro USB 12c
Sensores	Acelerómetro	Acelerómetro, botón pulsador integrado (en la versión nueva)	Acelerómetro, giroscopio, magnetómetro, y sensor de presión
Alimentación	Cr2450 battery (3V)	Cr2450 battery (3V)	Micro USB (5V) dc jack (4.5V – 12V)
Autonomía	18 meses 0.5 Hz 2 semanas a 24 Hz	Hasta 20 meses a 0.5 Hz 10 meses a 1 Hz 1 mes a 10 Hz	En función de la batería utilizada

Recubrimiento	IP65	IP65 IP20	No tiene
Dimensiones	45 × 30 × 23 mm	50 × 42 × 15 mm	60 × 53 × 26 mm
Masa	26 g	21 g	12 g

Tabla 3 Tabla comparativa de los tipos de tags disponibles para el sistema de localización de Pozyx

3.2.4. Interfaz web

Pozyx pone a disposición una interfaz web, utilizada para la instalación, que permite también la visualización y configuración de múltiples elementos del cálculo de posicionamiento.

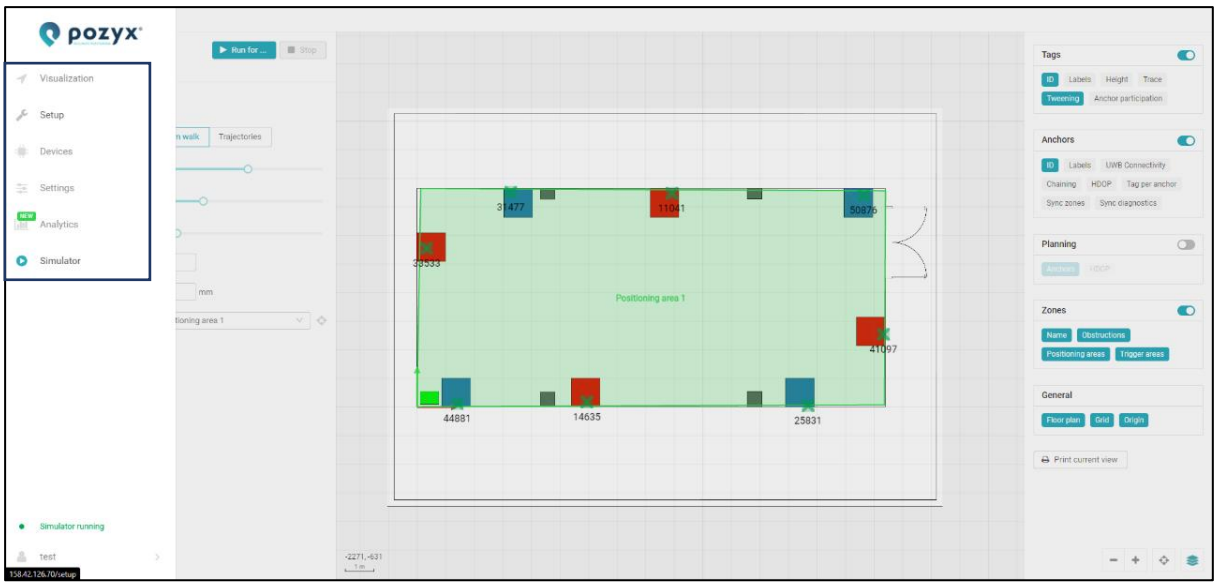


Ilustración 18. Menú principal de la interfaz web

Consta de varios apartados:

- **Visualization:** permite visualizar la posición de los tags asi como también su conexión y recepción con los anchors (antenas). La diferente información disponible se puede visualizar activando los diferentes topics disponibles en el menú desplegable al pulsar el botón de la esquina inferior derecha que permite gestionar la visualización de diferentes capas de información.
- **Setup**: permite acotar zonas de desplazamiento de los tags, cargar mapa del espacio de trabajo, ubicar las antenas y observar su conectividad.
- **Devices**: permite ver el estado de configuración de los anchors, tags y gateway, así como hacer un backup de su configuración.
- **Settings**: permite configurar el método de cálculo de posición comentado anteriormente, elegir los canales a utilizar, la interacción con el cloud y la API MQTT.
- **Analytics**: adquiriendo la licencia correspondiente, permitirá obtener información adicional en cuanto al posicionamiento de los tags.

- **Simulator**: permite simular tags virtuales en la interfaz para realización de pruebas.

3.2.5. Instalación y configuración

En este apartado se describen los pasos que hay que seguir para configurar el sistema de localización en el entorno de trabajo.

3.2.5.1. Instalación y configuración del Gateway

Para la instalación del Gateway, es necesario alimentarlo a una tensión de 12 V y conectarlo a internet mediante un cable ethernet al puerto de red "UPLINK". Una vez conectado, se debe acceder a la web https://app.pozyx.io e iniciar sesión. Por último, se añade un nuevo dispositivo del tipo Enterprise, introduciendo el ID del Gateway.

Ilustración 19 Gateway [14]

Se genera una dirección IP pública que es la que servirá para dar acceso a la interfaz. Para acceder a los datos generados por el sistema de posicionamiento para su tratamiento, es necesario realizar un puente MQTT con los siguientes datos de conexión generados por el sistema:

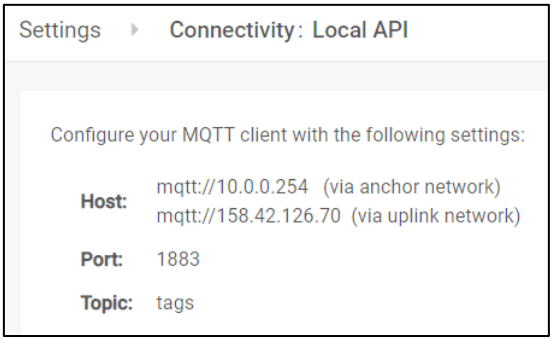


Ilustración 20 Datos de conexión por MQTT al Gateway de Pozyx

Una vez se ha configurado la conexión del Gateway, se puede proceder a la instalación de las antenas.

3.2.5.2. Instalación y configuración de las antenas

Una vez esté en funcionamiento el Gateway, se puede comenzar con la instalación de las antenas. Para ello, lo primero es configurar el mapa donde se van a ubicar las antenas para poder visualizar correctamente la posición de los tags.

3.2.5.2.1. Configuración del mapa

Este paso requiere de un plano escalado y con la posición de origen de coordenadas marcado para posteriormente hacerlo coincidir con el origen que se tiene en la interfaz web. En esta interfaz también se habrá de marcar la dirección del eje X y del eje Y en el plano subido.

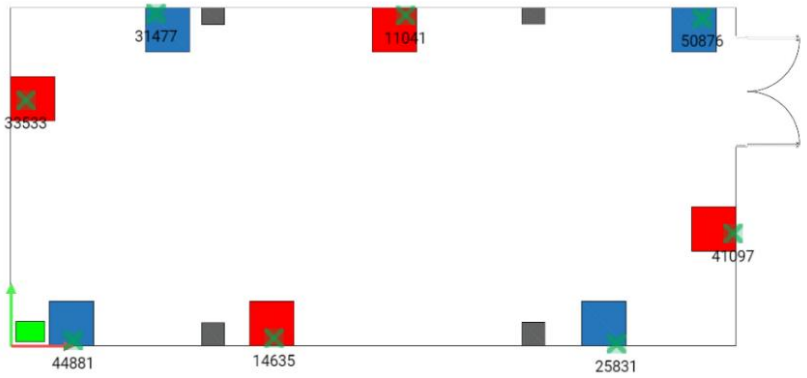


Ilustración 21 Configuración del mapa en la aplicación web de Pozyx

Una vez cargado el mapa, se procede a la instalación de las antenas. En este caso, las antenas se han de ubicar de distintas formas en función de si se desea obtener las posiciones de los tags en dos o en tres dimensiones.

3.2.5.2.2. Ubicación de las antenas para posicionamiento en 2D

Para ubicar correctamente las antenas en el emplazamiento deseado, es conveniente seguir las instrucciones que se listan a continuación:

- Situar las antenas en las paredes a la altura de la línea de visión de los usuarios. Así se reducen las posibilidades de que la señal se vea afectada por la presencia de obstáculos. Con esto se logra mejorar la precisión de las medidas de posicionamiento.
- Las antenas han de envolver al usuario o al objeto en cuestión, evitando situar las antenas en una misma línea o totalmente enfrentadas. El objetivo es cubrir todas las direcciones de medida, logrando minimizar el error en el posicionamiento.
- Cada antena debe visualizar al menos a otras dos antenas a una distancia inferior a 20 o 30 metros. Esta distancia puede disminuir en entornos donde existen muchos obstáculos.
- Situar las antenas verticalmente con los conectores en la parte superior y el receptor en la parte inferior. Es recomendable disponer de una distancia de separación de 20 cm entre la antena y la pared.
- Para evitar interferencias es recomendable no situar objetos metálicos, cables o depósitos que contengan fluidos cerca de las antenas.

3.2.5.2.3. Ubicación de las antenas para posicionamiento en 3D

Si se desea obtener el posicionamiento en tres dimensiones, se han de seguir las mismas instrucciones que para el posicionamiento en 2D pero ubicando las antenas a diferentes alturas. En este caso, lo ideal sería maximizar la distancia entre las antenas situadas en la parte baja y las situadas en la parte alta para optimizar la precisión de la coordenada z. Además, igual que en el caso

anterior, hay que tratar de ubicar las antenas desfasadas para cubrir todas las direcciones de medida posibles.

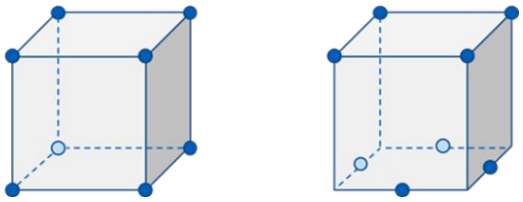


Ilustración 22 Disposición ideal de las antenas para el posicionamiento en 3D [14]

3.2.5.2.4. Conexión de las antenas

Cada una de las antenas cuenta con dos puertos ethernet (RJ45). Esto es así para conectar las antenas en cadena y reducir la cantidad de cables empleados. El número máximo de antenas por cadena depende de cómo se alimentan y la calidad de los cables empleados. Normalmente, se recomienda disponer de un máximo de cuatro antenas por cadena, con una distancia máxima de separación entre antenas de 50 metros.

Ilustración 23 Antenas de Pozyx [14]

Para esta configuración sólo es necesario alimentar la primera antena de la cadena, pudiéndose realizar de dos formas distintas:

1. Alimentando la primera antena de cada cadena a través de su conector de potencia.

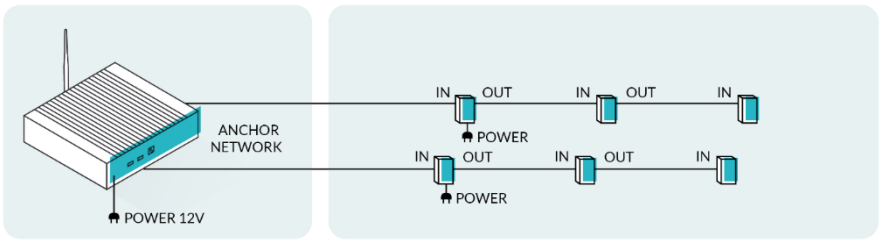


Ilustración 24 Alimentación de las antenas en cadena mediante el conector de potencia [14]

2. Alimentando la primera antena de cada cadena mediante el cable de Ethernet, si se tiene un switch con PoE+(802.3at).

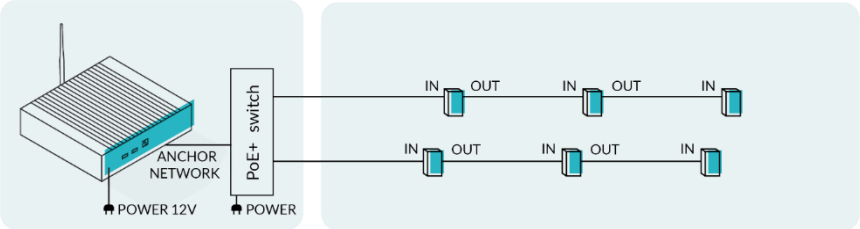


Ilustración 25 Alimentación de las antenas en cadena mediante el cable de Ethernet y un switch [14]

Por otro lado, es recomendable emplear cables de red de cobre ,sin revestimientos de aluminio, con alta protección contra las interferencias electromagnéticas (mínimo CAT6 AWG24). También cables de red trenzados S/FTP.

Las antenas cuentan con un LED que sirve para obtener información acerca del estado de funcionamiento de estas. Para poder visualizar el LED y extraer información valiosa, es necesario que las antenas se encuentren conectadas al Gateway. En la siguiente tabla se muestran los posibles estados de funcionamiento de las antenas en función del LED indicador.

ESTADO DEL LED	SIGNIFICADO
Apagado	La antena se encuentra apagada
Parpadeo rojo	La antena no está funcionando bien
Rojo	La antena no está conectada a la red
Parpadeo naranja	La antena está conectada a la red pero no recibe la dirección IP
Naranja	La antena ha recibido la dirección IP
Azul	La antena está en modo mantenimiento
Parpadeo verde	La antena está iniciándose
Verde	La antena está en el modo normal de operación

Tabla 4 Tabla informativa del estado de la antena en función del estado del LED

Una vez instaladas las antenas, se puede acceder desde la aplicación web de Pozyx para ver su disposición y el grado de conexión establecido entre ellas. Se puede apreciar en la imagen siguiente las conexiones con UWB que pueden establecerse entre cada una de las antenas con sus vecinas.

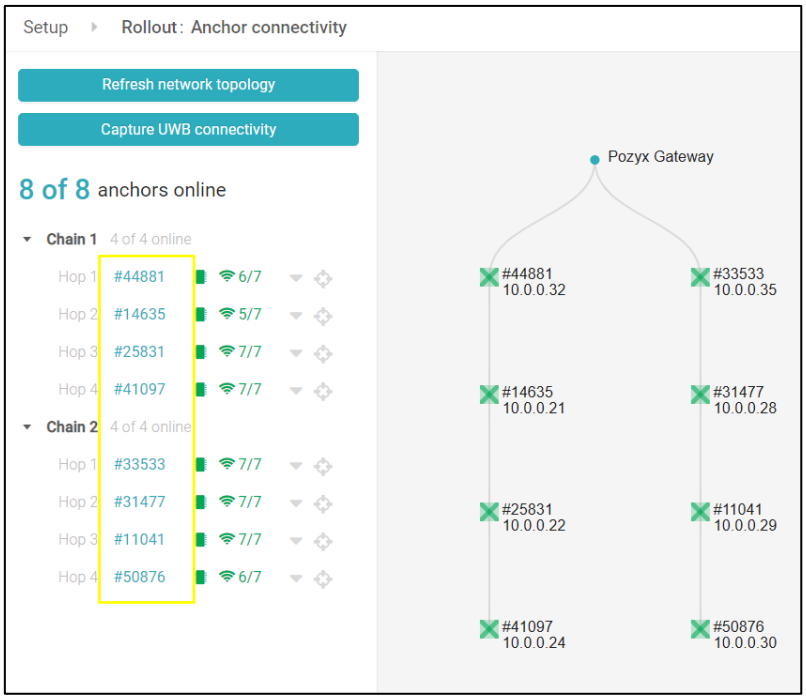


Ilustración 26 Esquema de conexión de las antenas en la interfaz web de Pozyx

Por último, es necesario indicar las coordenadas de cada una de las antenas en la aplicación web de Pozyx para partir de un sistema de coordenadas adecuado y preciso para el correcto posicionamiento de los tags.

3.2.5.3. Configuración de los tags

La configuración de los tags se puede modificar a través de la aplicación **Pozyx Device Configurator**. Mediante ella, se puede renombrar el ID del tag, como se muestra en la siguiente imagen.

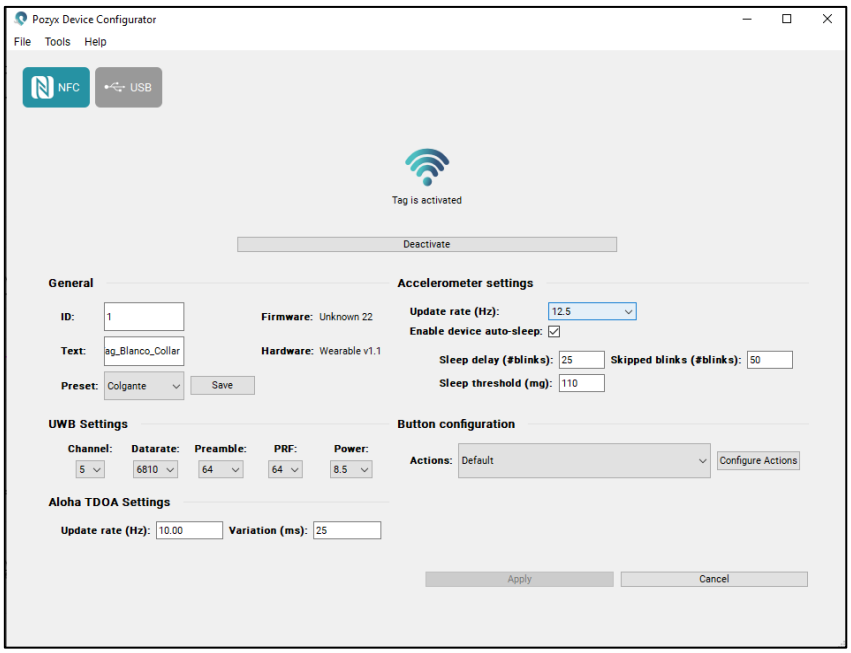


Ilustración 27. Menú del tag habilitado para renombrar el ID del tag

Desde esta App se puede elegir el protocolo de comunicación de los tags (pudiendo ser TDOA o TWR), su ID, la configuración de UWB y los parámetros de comunicación, como se muestra en la siguiente imagen.

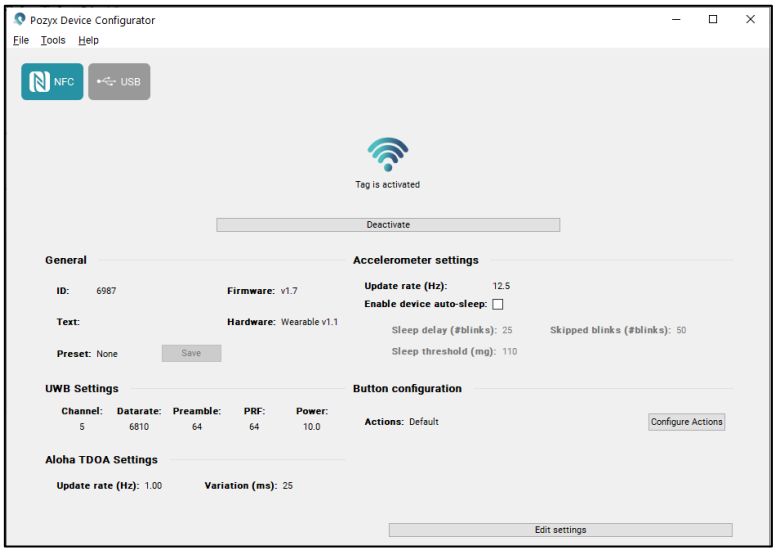


Ilustración 28 Interfaz gráfica de Pozyx Device Configurator para un tag portable

Las características que aparecen en la interfaz dependen del tag que se desee configurar. Se tienen las siguientes características:

- **General.** En esta sección se puede cambiar la ID del tag. Además, es posible guardar o cargar una configuración de otro tag en la opción de "Preset".
- **UWB settings.** En esta sección se puede modificar el canal de UWB, la velocidad de intercambio de datos, la frecuencia de repetición de pulso (PRF) y la potencia de transmisión. Hay que tener en cuenta que la potencia del tag debe ser lo suficientemente alta para que las antenas puedan captar los paquetes UWB emitidos. Sin embargo, aumentar la potencia puede aumentar la reflexión de las señales y por tanto, se tendrá mayor error de posicionamiento.
- **Aloha TDOA Settings.** En esta sección se puede modificar el número de veces por segundo que el tag emite una nueva posición, así como el parámetro de variación que reduce la posibilidad de que se den colisiones entre diversos tags.
- **Accelerometer settings.** En esta sección se puede configurar el tag para que pase a estar inactivo en función de unos determinados parámetros del acelerómetro, para prolongar la duración de la batería.
- **Button configuration.** En esta sección se puede configurar la función deseada para el botón del tag.

Para los tags de desarrollo se tiene que configurar, además, el método de localización. Esto se realiza dentro de la sección general, en el parámetro "Mode". Hay que indicar si va a trabajar con el método TDOA o el método TWR, que se explicarán en el siguiente apartado. Sin embargo, los tags portables únicamente pueden trabajar con el método TDOA y por tanto, no presentan la opción de modificar dicho parámetro en la interfaz de Pozyx Device Configurator.

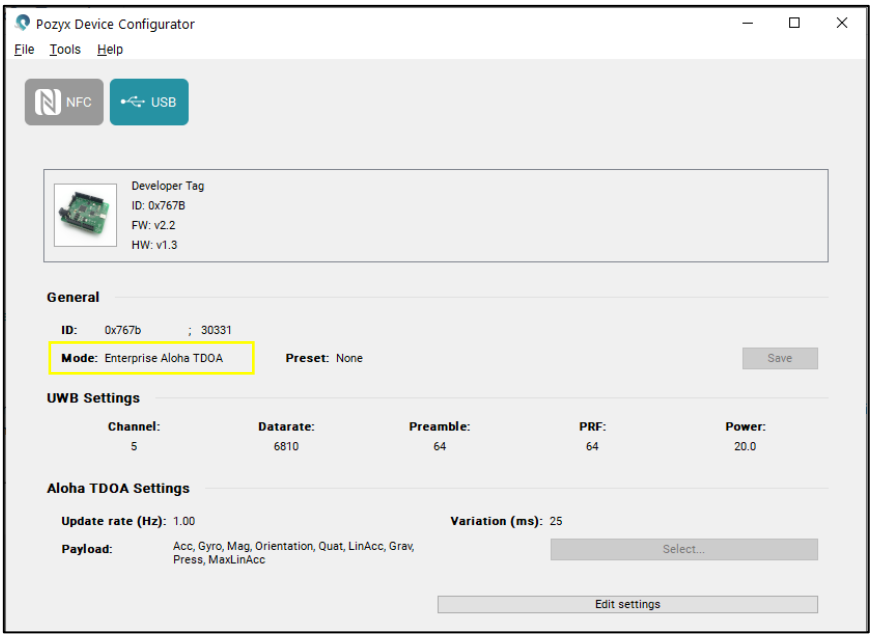


Ilustración 29 Interfaz gráfica de Pozyx Device Configurator para un tag de desarrollo

Finalmente, también se pueden configurar los eventos mostrados en las pruebas.

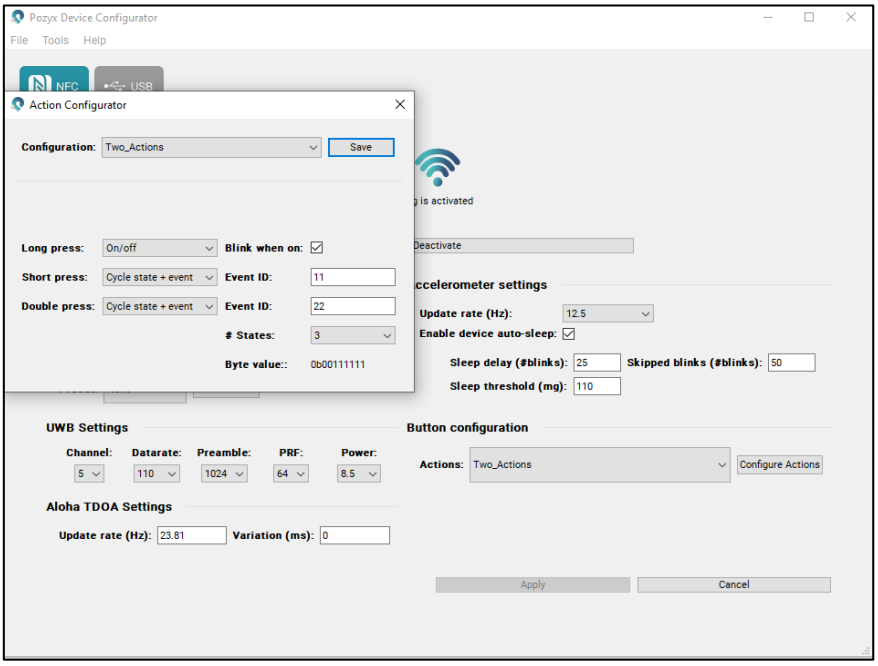


Ilustración 30. Configuración de acciones y eventos

3.2.5.4. Configuración del modo de localización

Una vez se ha realizado toda la instalación del hardware del sistema de posicionamiento, hay que establecer el protocolo de comunicación con el que se desea trabajar para la obtención de las coordenadas de los tags. Esta configuración se deberá indicar en la aplicación web de Pozyx para el correcto funcionamiento de las antenas, dentro del menú “Settings”>”Positioning”>”Positioning settings”>”Mode”.

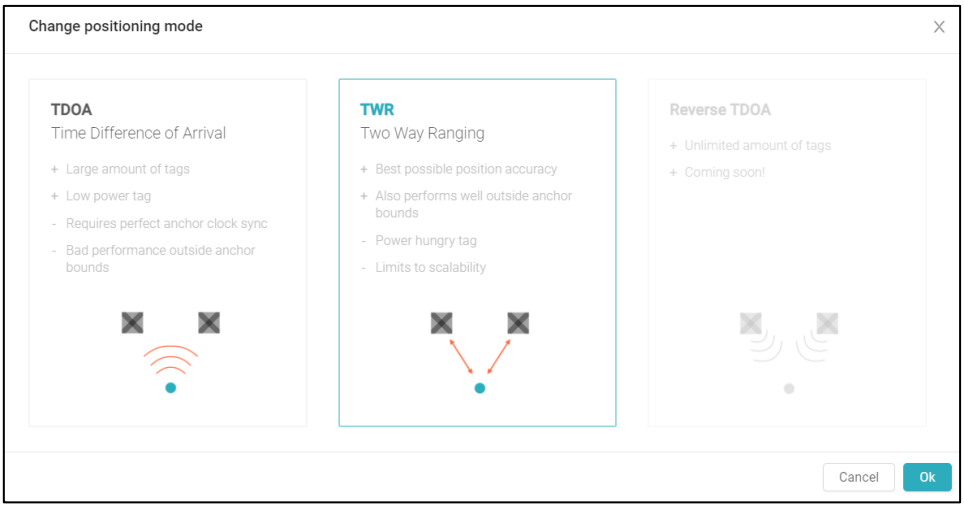


Ilustración 31. Configuración del método de posicionamiento

También se puede configurar el canal de funcionamiento accediendo al menú “Settings”>”UWB”

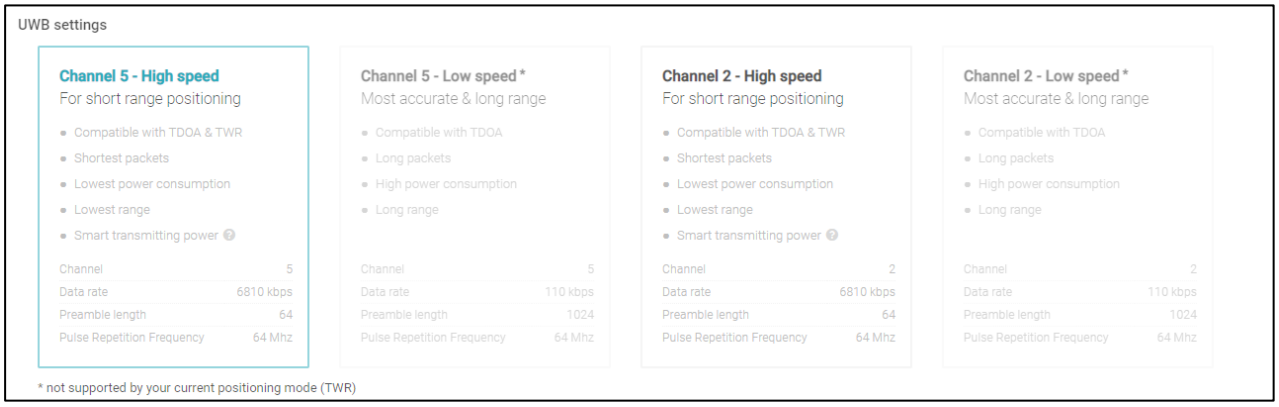


Ilustración 32. Configuración del canal y modo

3.3. ROBOT DE TRABAJO

Para poder medir la precisión del sistema de localización descrito y poder utilizarlo en el desarrollo de aplicaciones industriales se ha empleado el robot RB-1 base de la empresa Robotnik. Este es un robot móvil autónomo para el transporte de cargas en interiores que soporta hasta 50 kg de carga y tiene hasta 10 horas de autonomía [13]. Como se observa en la siguiente imagen, cuenta con un sensor láser empleado para la navegación y la localización, así como un sensor RGBD para la detección de obstáculos.

Ilustración 33 Robot RB-1 base de la empresa Robotnik

El RB-1 está basado en ROS, que es un framework de código abierto para el desarrollo de software para robots que ofrece los servicios de un sistema operativo. Tales como la abstracción del hardware, el control de dispositivos de bajo nivel, la implementación de funcionalidad de uso común, el paso de mensajes entre procesos y el mantenimiento de paquetes.

Este framework está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros [12]. Estos nodos se comunican entre sí bajo la jerarquía del nodo Máster, como se observa en el esquema siguiente. Así, cada nodo envía los mensajes a un determinado tema o topic. Enviar mensajes a un topic se conoce como "publicar" y leer los datos de un topic se le denomina "suscribirse". Los mensajes pueden ser del tipo entero, de coma flotante o booleanos, pero un determinado topic sólo puede enviar o recibir un tipo de mensaje cada vez.

Por otro lado están los servicios, cuando se requiere algún tipo de interacción entre los nodos. Los servicios tienen una definición del tipo de mensaje, así podemos enviar una petición de servicio a otro nodo que lo realice. El resultado del servicio se envía como una respuesta. El nodo debe esperar hasta que el resultado sea recibido en el otro nodo. Esto también se muestra en el esquema inferior.

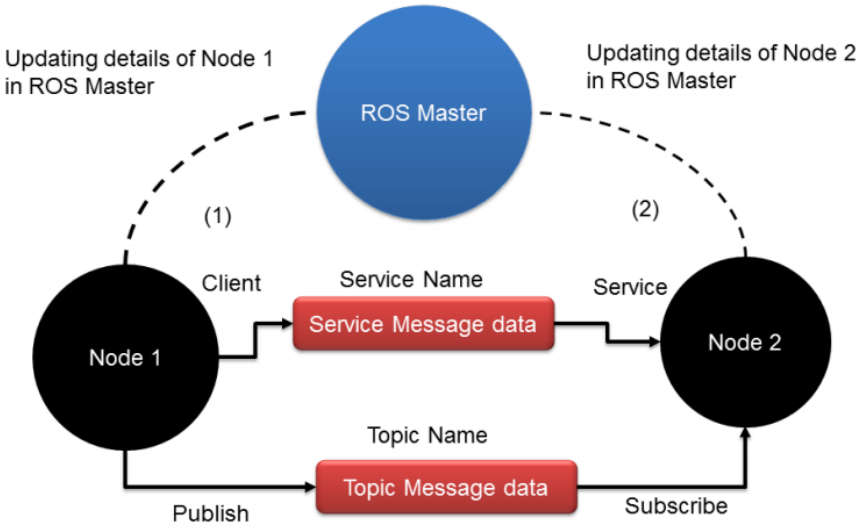


Ilustración 34 Esquema de funcionamiento básico de ROS [11]

La librería está orientada para un sistema UNIX (Ubuntu -Linux) aunque también se está adaptando a otros sistemas operativos.

ROS servirá de puente entre el sistema de localización de activos de Pozyx y el robot, permitiendo la implementación de algunas mejoras en determinadas aplicaciones de la industria 4.0, como se verá más adelante.

CAPÍTULO 4. INTERFAZ GRÁFICA DESARROLLADA

Con el objetivo de recopilar datos de posicionamiento de los diversos tags para los experimentos realizados, se ha programado una interfaz gráfica en Python con la librería de PyQt5. En este capítulo se va a presentar el diseño y sus prestaciones, así como la arquitectura software implementada.

4.1. DISEÑO Y FUNCIONALIDAD

La interfaz está compuesta por dos partes diferenciadas, en la parte inferior se carga el mapa del laboratorio en el cual se tiene instalado el sistema de localización de Pozyx, y en la parte superior se tienen los botones que proporcionan la parte lógica de la interfaz junto con la leyenda con los tags que haya activos en el laboratorio durante la monitorización. Esta parte segunda parte está compuesta por varias secciones, como se observa en la imagen inferior. La sección de monitorización, la sección de comunicación con el robot, la de grabación de los datos, a su derecha la de simulación, la de comunicación con el sistema de localización de Pozyx, y una última sección donde poder cargar un mapa de intensidad de señal para realizar un análisis de zonas oscuras del laboratorio de estudio.

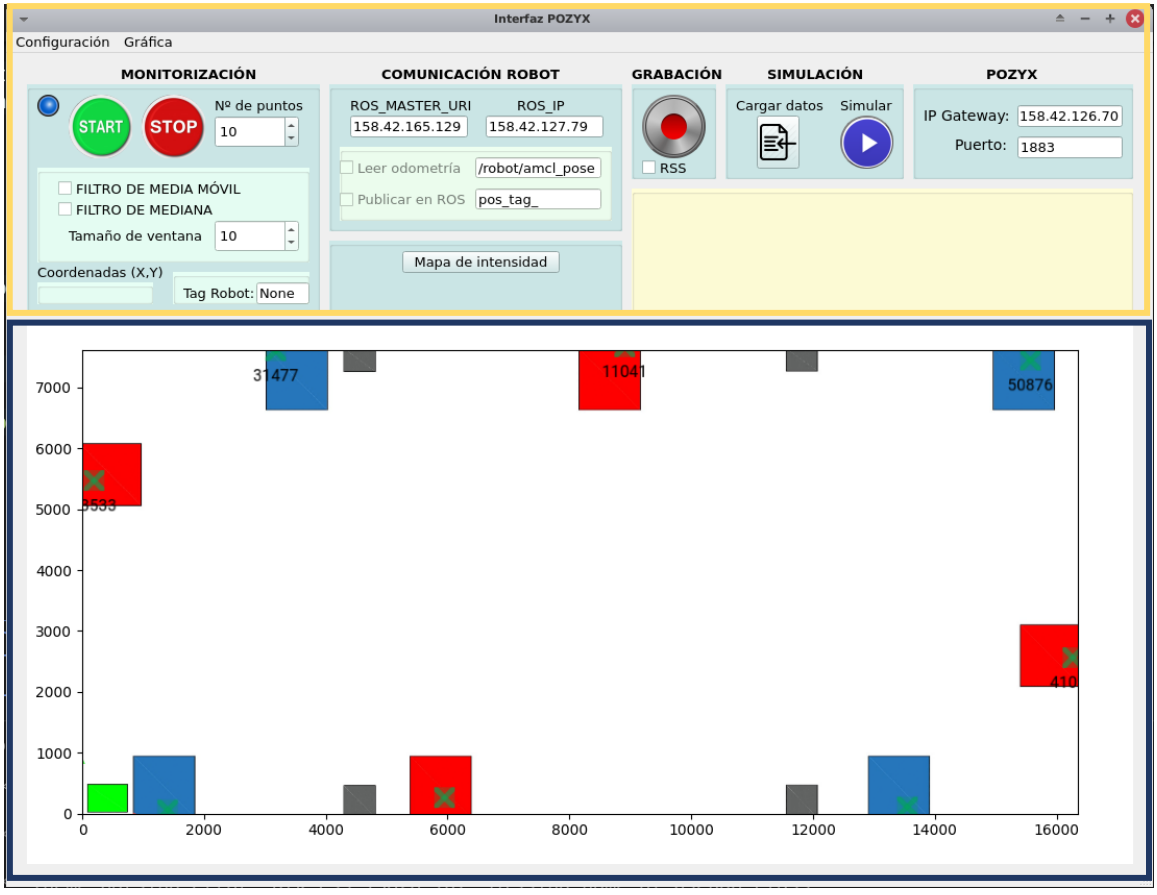


Ilustración 35 Diseño de la interfaz propia programada con la librería PyQt5

Al iniciar la interfaz, se carga por defecto un fichero de configuración en formato JSON, situado en la misma carpeta, donde se extrae una IP y un puerto por defecto para establecer la conexión puente MQTT, así como el mapa del laboratorio y sus medidas de ancho y largo en milímetros para establecer los ejes de la gráfica. También el número de puntos que se van a visualizar en un principio al iniciar la monitorización, como se detallará más adelante.

```
{
    "IP_DATA":"158.42.126.70",
    "PORT_DATA":1883,
    "MAP":"index.png",
    "XLIMIT":16350,
    "YLIMIT":7610,
    "NPOINTS":10
}
```

Ilustración 36 Fichero de configuración

Se va a realizar a continuación una descripción detallada de las funciones contenidas en cada una de las secciones mostradas en la interfaz.

4.1.1. Comunicación con el sistema de localización de POZYX

Arriba en la parte derecha se introducen los dos campos necesarios para poder realizar la conexión por puente MQTT con el sistema de localización de Pozyx que haya disponible en el laboratorio, es decir, la IP y el puerto de conexión.

Ilustración 37 Menú de comunicación con el sistema de localización de Pozyx

4.1.2. Monitorización

A través de esta sección se pueden reproducir en tiempo real las trayectorias seguidas por los tags que estén activos. Cuando se pulsa el botón de “Start” se realiza la conexión por MQTT con el sistema de localización de Pozyx, poniéndose el led en verde si ésta se realiza correctamente. Asimismo, se puede especificar el número de puntos que se desea ver en el mapa, y aplicar a los datos en bruto un filtro de media móvil o de mediana, en función de la aplicación.

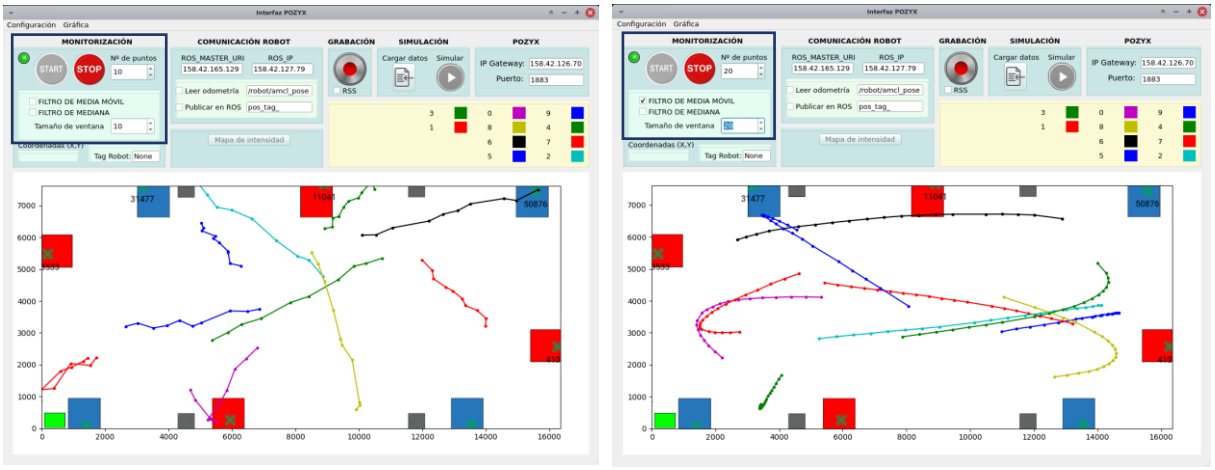


Ilustración 38 Menú de monitorización de la interfaz

Por otro lado, si se están monitorizando muchos tags simultáneamente, puede ser deseable sólo visualizar unos determinados tags en algún momento. Para deshabilitar la monitorización de dichos tags sin tener que parar el experimento para configurarlos, se puede pinchar sobre el nombre del tag en la leyenda y éste dejará de mostrarse en la gráfica. No obstante, permanecerán inactivos en la leyenda por si se desea reactivar su monitorización en cualquier momento.

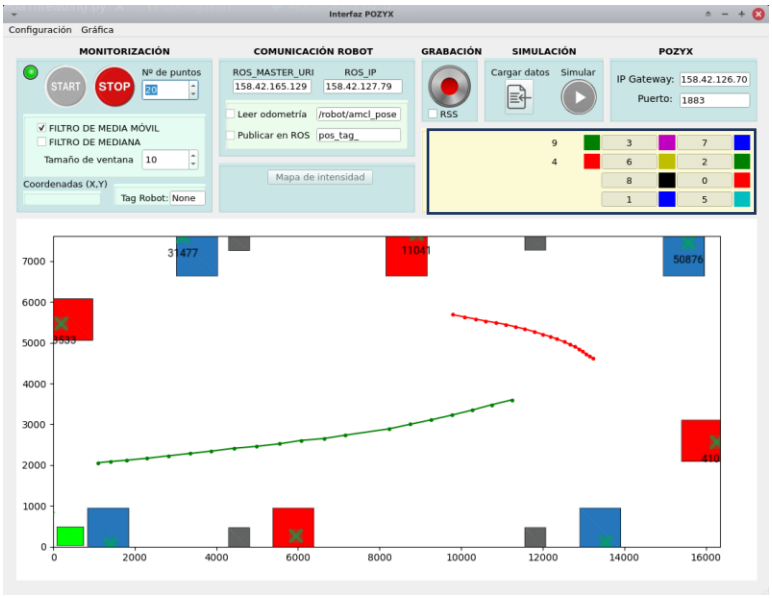


Ilustración 39 Leyenda de la interfaz

También se muestran en un recuadro las coordenadas del mapa cuando el puntero del ratón se sitúa sobre el área del mapa cargado, esto puede ser interesante en algunos experimentos donde se desee conocer la posición del mapa en la que se tiene la lectura del tag. Además, es posible especificar en esta sección si hay algún tag que se identifique con un robot y se graficará con un margen de seguridad, como se muestra en la imagen siguiente.

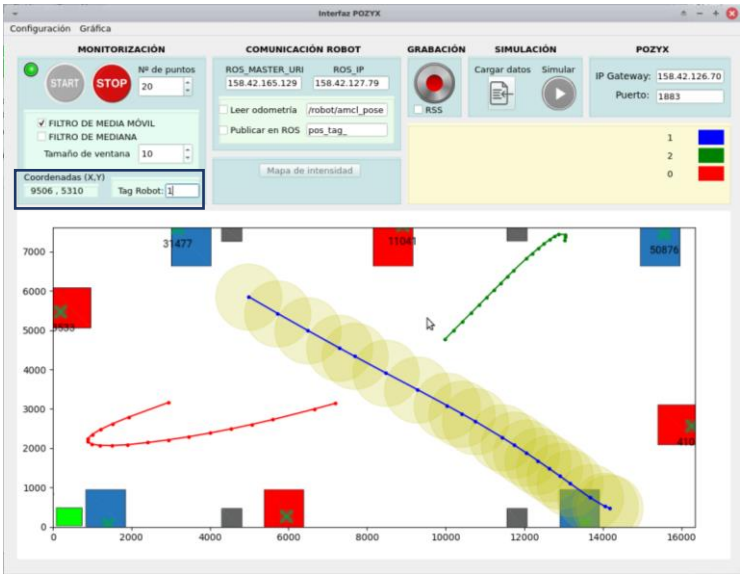


Ilustración 40 Coordenadas del ratón y selección del tipo de tag a mostrar en la interfaz

La monitorización finaliza cuando se pulsa el botón “Stop”.

4.1.3. Grabación

En ocasiones es necesario extraer información acerca de las trayectorias seguidas por determinados tags durante un experimento concreto, o medir la intensidad de la señal que se tiene en determinados puntos del mapa. Para poder realizar análisis posteriores con datos extraídos de la monitorización, es posible grabar los datos en tiempo real. Durante la monitorización de los tags, se puede pinchar sobre el botón de “Grabar” y se abrirán dos ficheros por cada tag que guardarán las coordenadas x e y del tag, así como la intensidad de la señal que tiene en ese punto del mapa, si es que se marca la opción de “RSS” (“Received Signal Strength”).

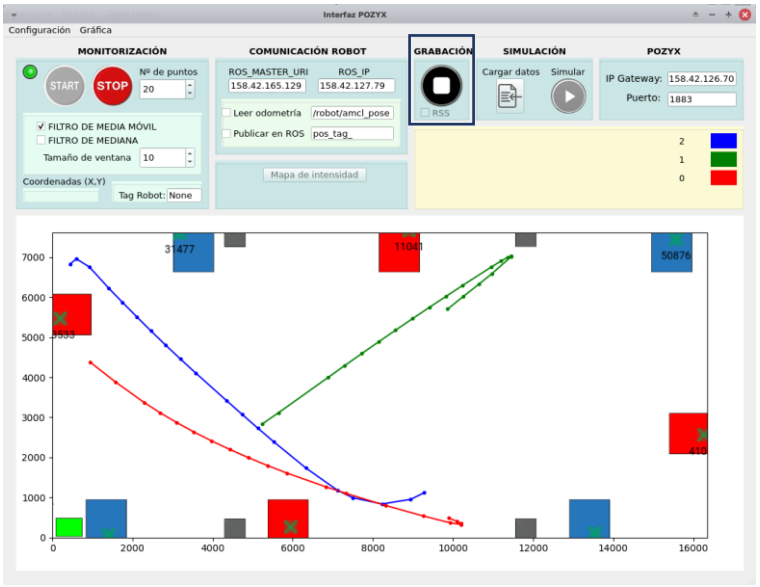


Ilustración 41 Menú de grabación de la interfaz

Al detener la grabación, se cierran los dos archivos con la información grabada. Uno de los ficheros tiene formato de texto ".txt" y otro formato de codificación binaria ".pickle", en este caso. Estos ficheros se guardan en la carpeta desde la cual se ejecuta la interfaz, con el nombre que contiene el identificador del tag y la fecha de grabación. En la imagen siguiente se muestra un ejemplo de la estructura de datos recogida en el fichero de texto para un tag del que se desea recoger además de las coordenadas, la intensidad de la señal.

tag27191_7-7.txt

```
coord_x coord_y ['11041', '50876', '41097', '33533', '14635', '31477', '25831', '44881']
9845 3812 -81.46  -84.64  -85.21  -83.02  -81.53  -82.06  -82.31  -83.36
9836 3799 -81.46  -84.25  -85.06  -83.02  -81.27  -82.52  -82.25  -83.23
9839 3780 -80.93  -84.32  -84.61  -82.05  -81.39  -82.08  -82.03  -83.27
9843 3760 -81.27  -84.82  -84.99  -82.05  -81.19  -81.99  -82.1  -82.97
9836 3735 -81.48  -84.49  -85.26  -82.05  -81.19  -82.13  -82.29  -83.23
9835 3711 -82.51  -84.14  -84.53  -82.05  -81.58  -82.1  -82.38  -83.5
9830 3694 -82.51  -84.09  -84.53  -82.05  -81.16  -82.08  -81.94  -83.12
9853 3717 -82.24  -84.21  -85.15  -82.05  -81.06  -82.15  -82.49  -83.57
9869 3733 -82.1  -84.48  -84.64  -82.76  -81.97  -82.08  -82.03  -83.18
9880 3747 -82.27  -84.11  -84.54  -82.58  -81.42  -82.1  -82.34  -83.07
9896 3744 -82.39  -84.53  -85.72  -82.45  -81.21  -82.16  -82.06  -83.34
9899 3735 -81.61  -84.09  -84.82  -82.07  -81.77  -82.31  -82.31  -83.16
9902 3748 -82.13  -84.03  -85.56  -83.28  -81.34  -82.18  -82.31  -83.25
9896 3756 -81.98  -84.31  -85.15  -81.95  -81.41  -81.86  -81.3  -83.36
9887 3767 -81.77  -83.87  -84.54  -82.45  -81.79  -82.22  -81.04  -83.16
9868 3772 -81.86  -84.34  -85.09  -83.05  -81.28  -81.94  -80.76  -83.16
9853 3846 -82.17  -84.13  -85.04  -83.05  -81.67  -82.27  -80.76  -83.12
9840 3854 -82.05  -84.06  -85.48  -82.12  -82.49  -82.47  -82.45  -83.21
9824 3866 -82.03  -84.03  -85.39  -82.52  -82.9  -82.15  -81.72  -83.21
9816 3878 -81.98  -84.4  -86.07  -82.54  -82.91  -82.47  -82.11  -83.58
9808 3890 -81.83  -84.03  -84.17  -82.96  -82.97  -82.4  -82.8  -83.54
9802 3896 -81.85  -84.59  -85.19  -83.8  -83.23  -82.06  -81.93  -83.55
9829 3872 -81.91  -84.17  -84.7  -83.8  -83.23  -82.33  -82.17  -83.23
9849 3850 -81.88  -84.24  -84.92  -82.01  -82.93  -82.52  -81.93  -83.37
9859 3837 -81.81  -84.54  -85.52  -83.21  -83.09  -82.13  -82.81  -83.32
9861 3836 -81.7  -84.22  -85.02  -82.35  -83.12  -81.99  -82.1  -83.05
9866 3835 -82.06  -84.64  -84.76  -82.2  -83.26  -82.4  -82.1  -83.41
9874 3797 -81.95  -84.4  -85.81  -82.2  -83.26  -81.93  -82.52  -83.55
9882 3762 -82.01  -84.43  -84.64  -83.4  -82.61  -82.42  -82.31  -83.14
9892 3742 -82.03  -84.06  -85.48  -82.31  -82.73  -82.06  -82.2  -84.01
9901 3713 -82.2  -84.47  -84.86  -82.7  -83.56  -81.98  -82.86  -83.1
```

Ilustración 42 Fichero de texto generado por la grabación

Como se puede observar, en la primera línea se indica el identificador de cada una de las antenas que están recibiendo señales del tag activo, y en sus respectivas columnas se muestra la RSS en decibelios.

4.1.4. Simulación

Esta sección permite cargar los datos recogidos de un experimento y simular las trayectorias que siguieron los tags. Al pinchar sobre el botón de "Cargar datos", se abre una ventana que permite escoger el directorio deseado y seleccionar el fichero que se quiere mostrar en la gráfica, como se muestra en la siguiente imagen.

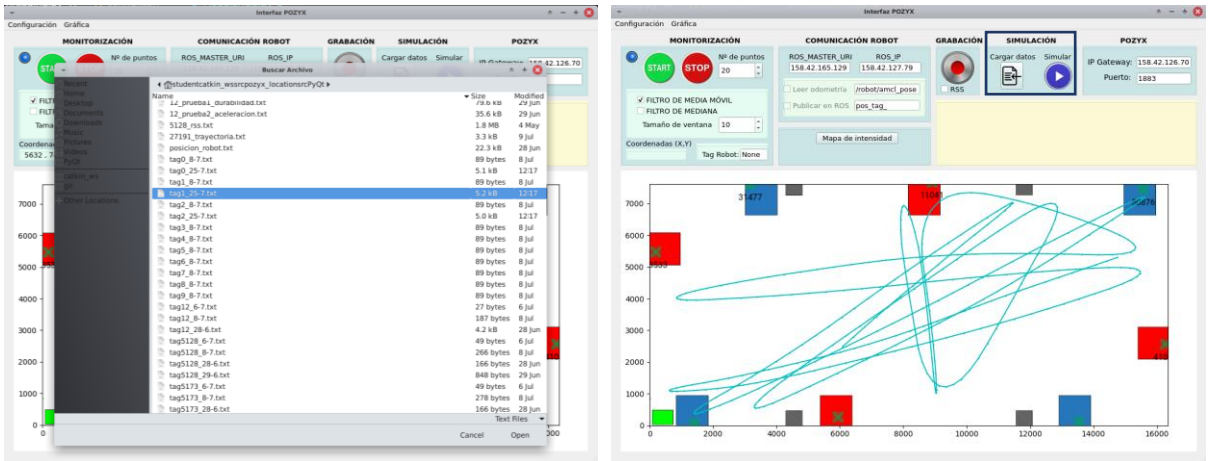


Ilustración 43 Menú de simulación de trayectorias en la interfaz y carga de ficheros

Una vez se ha cargado un fichero de datos de un determinado tag, es posible realizar una simulación de la trayectoria a una velocidad más rápida de la que se obtuvieron los datos, pudiendo detener la simulación en cualquier momento (ilustración 44. Izquierda). También es posible cargar la trayectoria de varios tags y representarlos sobre el mapa (Ilustración 44. Derecha).

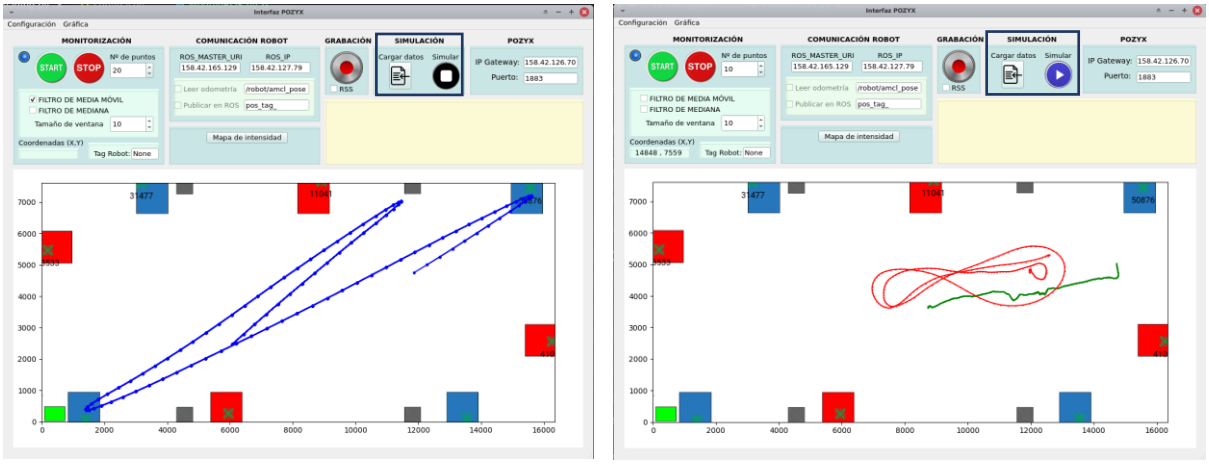


Ilustración 44 Simulación de trayectorias cargadas en la interfaz

Por otro lado, también puede resultar interesante realizar un posprocesado de los datos fuera de la interfaz, y luego cargarlos en esta para realizar el análisis, siempre respetando la estructura del fichero de datos.

4.1.5. Mapa de intensidad de la señal

Si se recogen datos de la intensidad de señal recibida por las antenas para cada tag, es posible realizar un mapa de intensidad cargando los datos que hayan sido grabados con la opción “RSS” marcada. Con estos datos, para cada posición del mapa se calcula el sumatorio de las intensidades de señal recibidas en cada antena, y se divide entre en número de antenas total que reciben esta señal, obteniéndose la intensidad media de señal en esa posición. Así, como se muestra en la siguiente imagen, se genera una leyenda de color que indica el valor en decibelios de la señal media para cada zona muestreada del mapa. Esto puede resultar muy interesante para conocer las “zonas oscuras” del mapa, donde probablemente se obtengan peores datos de posicionamiento.

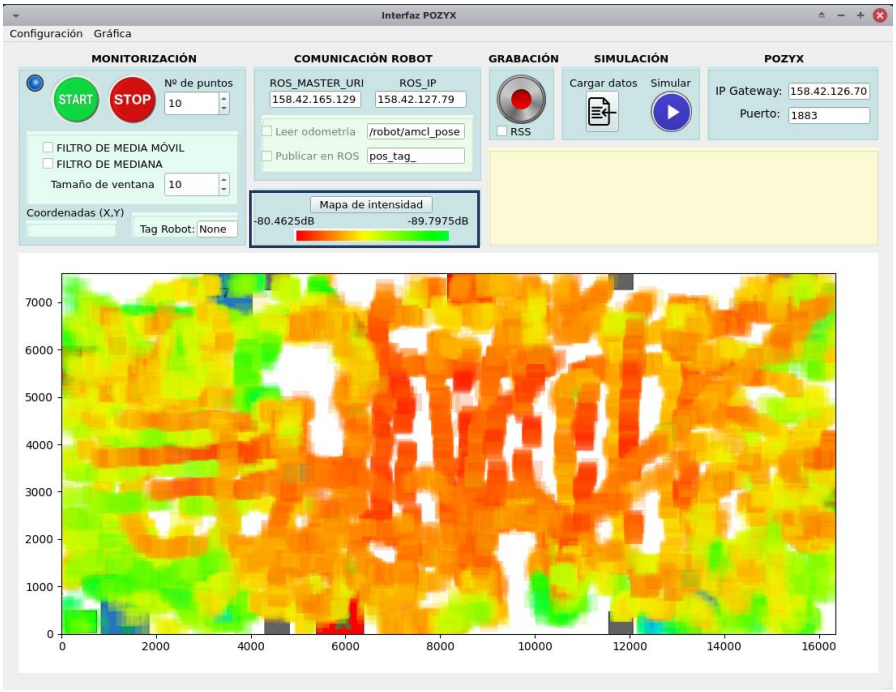


Ilustración 45 Menú de mapeo de intensidad de la señal (RSS)

En el caso de la imagen, la leyenda muestra el mínimo y el máximo valor de intensidad de señal en decibelios para el conjunto de datos representado. Por tanto, la zona que tiene mejor cobertura de señal es la zona central, graficada en color rojo, mientras que las zonas en color verde pueden proporcionar peor calidad de los datos de posicionamiento debido a la aparición de interferencias o lejanía del tag con respecto a ciertas antenas.

4.1.6. Comunicación con el robot

En esta sección se puede establecer una comunicación con el robot mediante la arquitectura de ROS. Para ello, se debe especificar la dirección IP del robot que contiene el nodo Master y la dirección IP del ordenador local, siempre que estén en la misma red Wifi. Una vez se haya establecido la conexión, es posible publicar en ROS durante la monitorización las posiciones de cada uno de los tags activos seleccionando la opción de "Publicar en ROS" y escribiendo el nombre del topic al que se desean publicar las posiciones.

En el caso de la imagen que se muestra a continuación, las coordenadas x e y del tag 27191 se están publicando en el topic "pos_tag_27191", esto es, la interfaz inserta para cada tag activo durante la monitorización su número de identificación al nombre escogido para el topic. Esto supone que las posiciones de cada tag activo se publican en topics distintos.

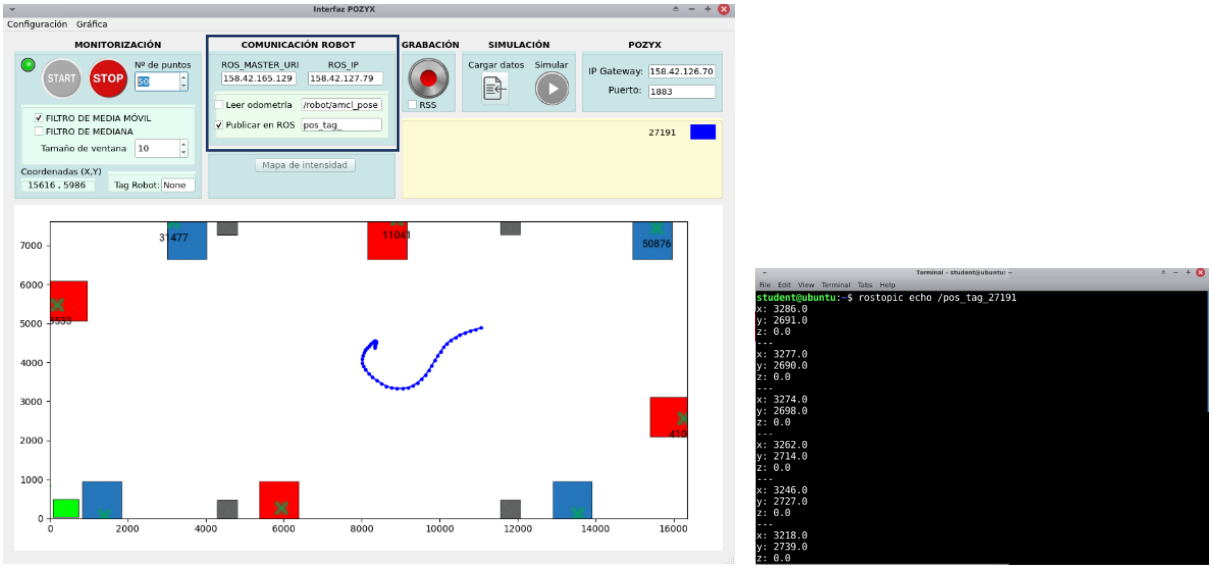


Ilustración 46 Menú de comunicación con el robot

Otra opción posible es la lectura de los datos de la odometría del robot, que puede ser interesante para contrastar los datos leídos por las antenas de Pozyx y las trayectorias que el robot calcula que puede estar realizando. Para ver en la gráfica la trayectoria seguida por el robot, hay que especificar el topic donde publica el robot los datos de la odometría y seleccionar la opción de "Leer odometría". En la siguiente imagen se muestra un ejemplo de esta aplicación, donde en azul se observa la trayectoria que sigue el tag 27191 según la lectura del sistema de posicionamiento de Pozyx, y en negro se visualiza la odometría que calcula el robot. En la aplicación real se ha situado el tag encima del robot, y se ha ordenado al robot que realice una trayectoria recta.

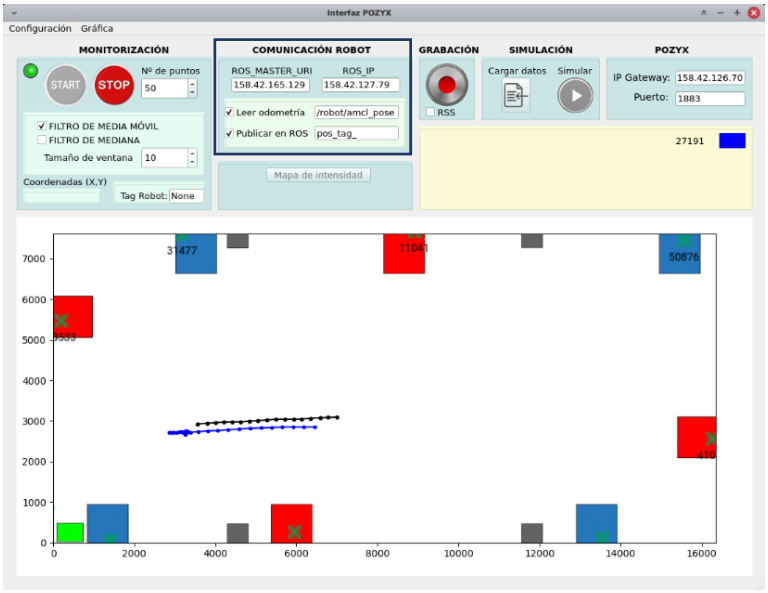


Ilustración 47 Lectura de la odometría del robot en la interfaz

4.1.7. Funciones adicionales

Además de las funciones explicadas en los apartados anteriores, se ha implementado un menú de configuración en la barra de estado de la interfaz, así como un menú de gráfica.

Ilustración 48 Funciones adicionales de la barra de configuración de la interfaz

El menú de configuración permite cargar un fichero de configuración como el que se carga por defecto al ejecutar la interfaz, en formato JSON. También es posible exportar la configuración actual de los parámetros escogidos. Por otro lado, está la opción de cargar un mapa distinto al que se carga por defecto, como se muestra en la imagen de abajo. Si se escoge la opción de "Editar ejes" aparece una ventana emergente donde se puede modificar el tamaño de los ejes (en milímetros), ajustando la escala de la gráfica.

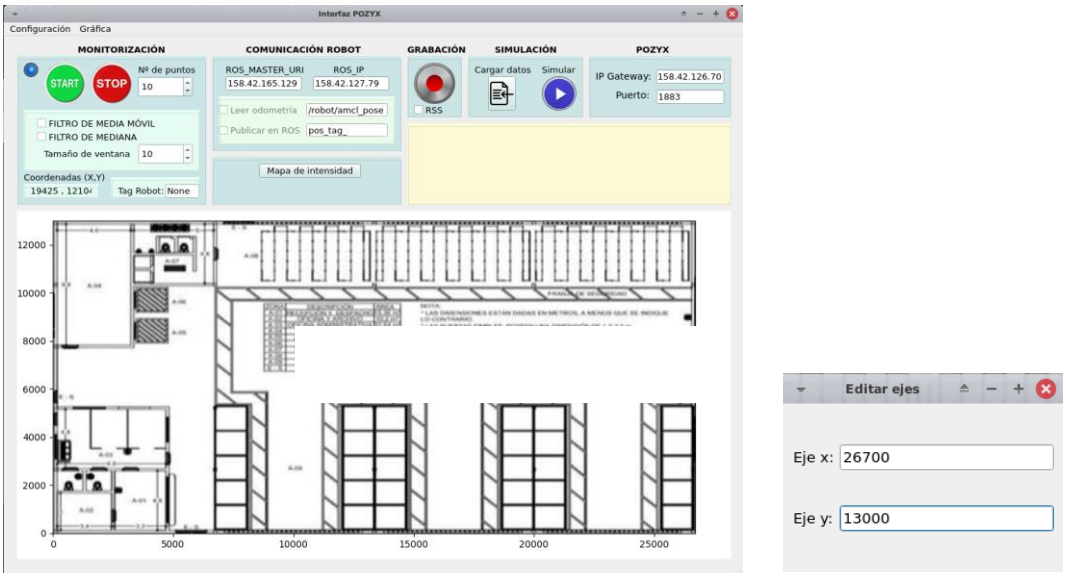


Ilustración 49 Mapa [15] cargado y ajustado en la interfaz

El menú de gráfica incluye únicamente la función "limpiar", que permite en cualquier momento limpiar los datos que haya en el mapa. Este menú podría incluir en un futuro nuevas funcionalidades, aunque no haya sido necesario en este trabajo.

4.2. ESTRUCTURA DE CÓDIGO DE LA INTERFAZ

El código de la interfaz se ha realizado en Python con clases y objetos. La estructura de clases programada se muestra en el siguiente esquema.

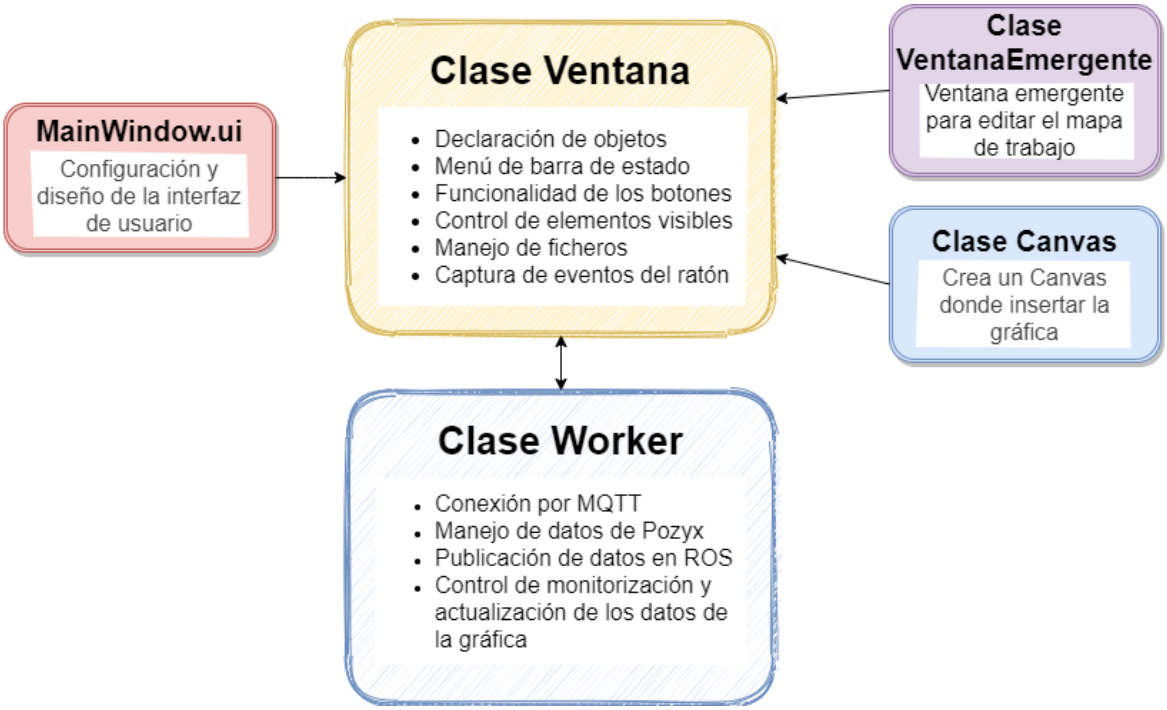


Ilustración 50 Esquema de la estructura del código de la interfaz

Para que la interfaz pueda funcionar correctamente se ha implementado una estructura multihilo, donde el hilo principal alberga la clase Ventana. Dentro de este objeto de la clase Ventana se crean los objetos VentanaEmergente, Canvas y Worker. Este último objeto pertenece a otro hilo de ejecución, pues se encarga de realizar la conexión con el sistema de localización así como el manejo de los datos. Al estar en dos hilos de ejecución distintos, la interfaz mantiene su funcionalidad independientemente del proceso llevado a cabo en el objeto Worker. Asimismo, también se crean hilos dentro de los dos hilos principales para las funciones que tienen un ritmo de ejecución independiente del programa. Por ejemplo, dentro del objeto Worker se crean dos hilos, uno de ellos para el cliente MQTT y el otro para actualizar los datos que aparecerán en la gráfica de la interfaz. Este último hilo controla la velocidad de actualización de las coordenadas a representar, puesto que la librería de Matplotlib empleada para construir la gráfica tiene un límite en frecuencia de actualización y si esto no se tiene en cuenta, genera problemas. Por otro lado, el hilo que crea el cliente MQTT sigue la siguiente estructura.

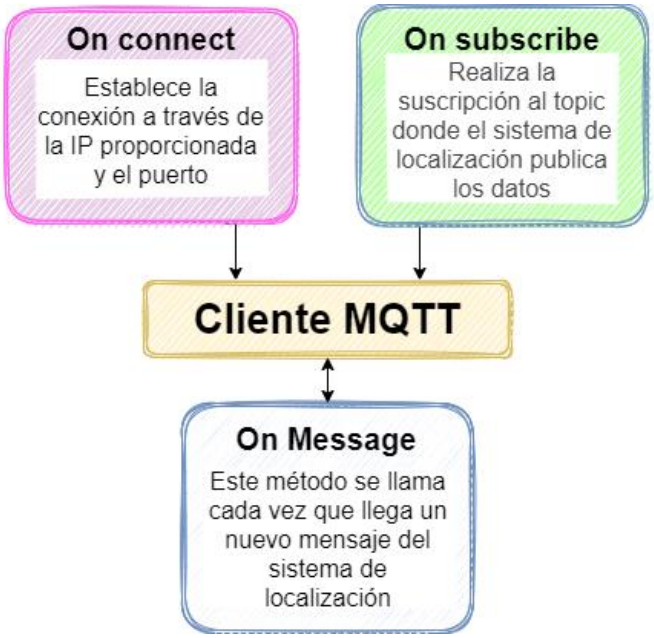


Ilustración 51 Esquema del cliente MQTT implementado en la interfaz

Para la gestión de las variables compartidas entre los distintos hilos activos, se emplean semáforos. Esto son las variables que regulan el bloqueo de las variables compartidas si están siendo manipuladas en un determinado instante en uno de los hilos, por temas de seguridad.

En el primer apartado del Anexo I de este documento se muestra todo el código desarrollado para la interfaz.

CAPÍTULO 5. EXPERIMENTOS DE MEDIDA DE PRECISIÓN

Para poder acotar el margen de error que tiene el sistema de localización por UWB empleado en este trabajo, se han realizado diversas pruebas de precisión. En primer lugar, pruebas de precisión con los tags estáticos, que en una situación real podrían asemejarse a recursos humanos en su puesto de trabajo, por ejemplo. En segundo lugar, también se han realizado pruebas de precisión con los tags en movimiento. Este último caso podría tratarse de robots móviles en aplicaciones de logística, o humanos en sus tareas de desplazamiento.

5.1. MEDIDAS DE PRECISIÓN PARA TAGS EN ESTÁTICO

Para estas primeras pruebas se han tomado cuatro posiciones del laboratorio distintas, tratando de aumentar la casuística con relación a la cobertura de la señal y las interferencias producidas por el entorno. A continuación se muestran las posiciones escogidas para la medida de precisión de los distintos tags.

Ilustración 52 Plano con las posiciones escogidas para situar los tags

Para conocer a priori la cobertura de señal que se tiene en el laboratorio, se ha realizado un mapa de intensidad de la señal, como se ha explicado en el apartado 3.3.1.5. En la gráfica siguiente se tiene que para los colores más cálidos hay mayor cobertura de señal, coincidiendo con la zona central del laboratorio. Mientras que por las zonas laterales hay peor cobertura, puesto que son menos las antenas que pueden leer el tag. En negro se muestran las posiciones escogidas, luego cabe esperar que en la posición dos se tendrán datos más precisos que en la posición uno.

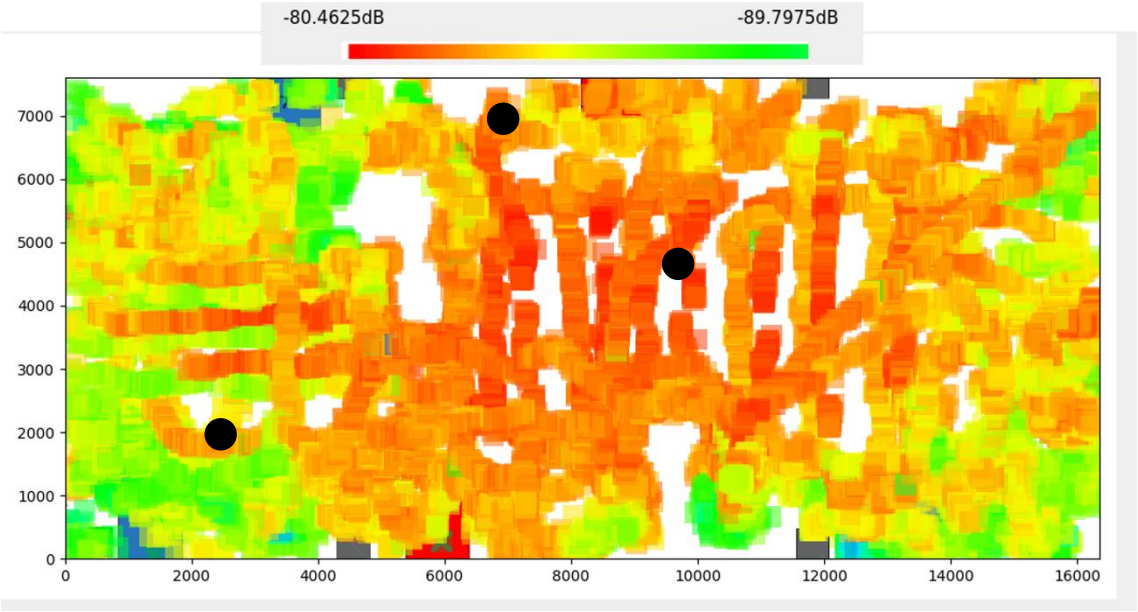


Ilustración 53 Mapa de intensidad de la señal del laboratorio

En las pruebas realizadas en el laboratorio se recogen los datos en crudo y se aplica un filtro de media móvil en Matlab a posteriori. Para medir la calidad de los datos de posicionamiento obtenidos, se calcula el error normalizado medio cuadrático con la fórmula siguiente.

$$NMSE = \frac{1}{N}\sum_{i}\frac{(M_i - P_i)^2}{\overline{MP}} \quad (Ec.2)$$

Donde:

N: número de datos grabados

M: datos medidos

P: datos reales

Así pues, probando para cada posición los tres tipos de tags disponibles (minitag, tag portable y tag de desarrollo) con el método de posicionamiento TDOA e introduciendo los datos en Matlab, se tienen los siguientes resultados en función del tiempo t.

- Posición 1. Minitag:

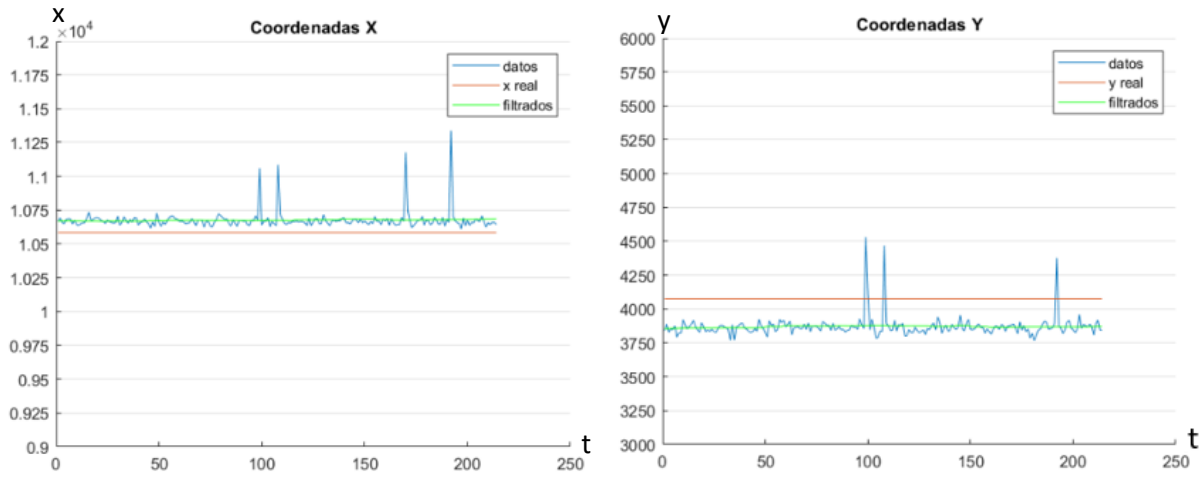


Ilustración 54 Valor de las coordenadas X e Y en función del tiempo para el Minitag (Pos. 1)

- Posición 1. Tag portable:

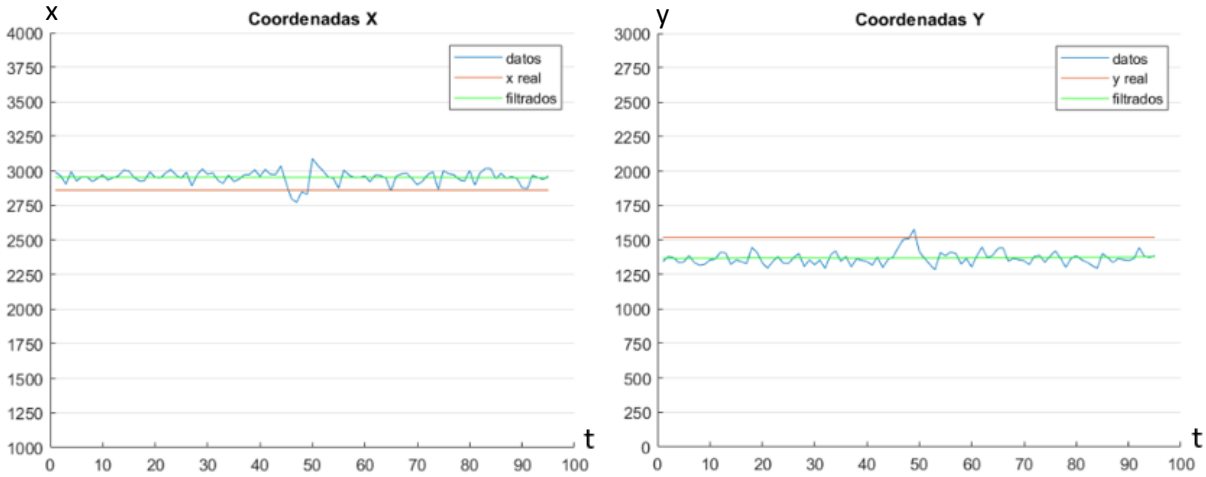


Ilustración 55 Valor de las coordenadas X e Y en función del tiempo para el Tag portable (Pos. 1)

- Posición 1. Tag de desarrollo:

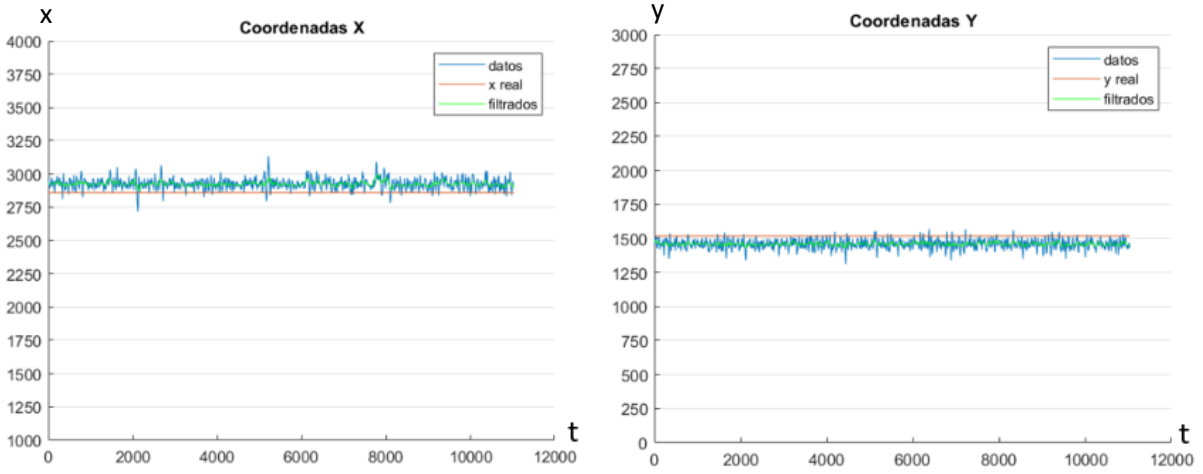


Ilustración 56 Valor de las coordenadas X e Y en función del tiempo para el Tag de desarrollo (Pos. 1)

- Posición 2. Minitag:

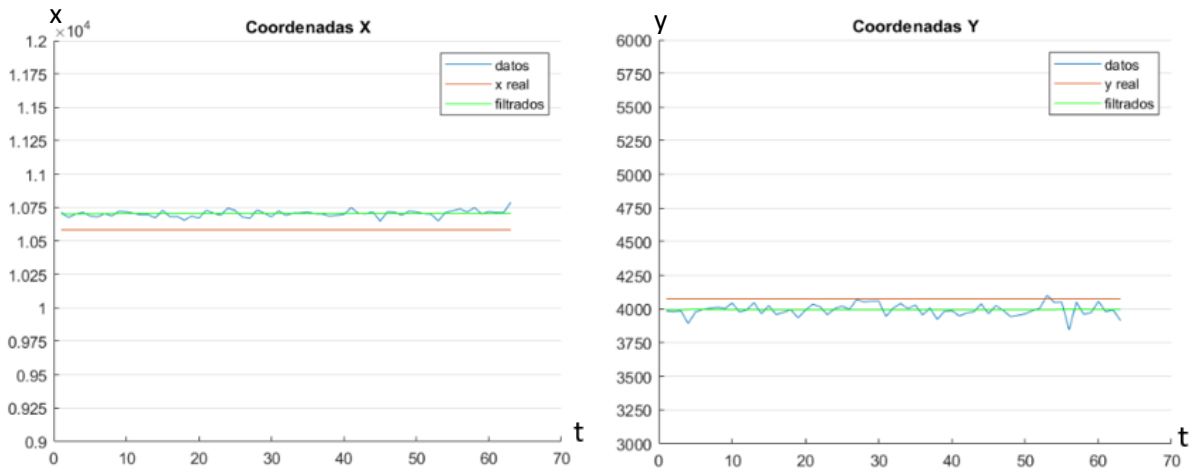


Ilustración 57 Valor de las coordenadas X e Y en función del tiempo para el Minitag (Pos. 2)

- Posición 2. Tag portable:

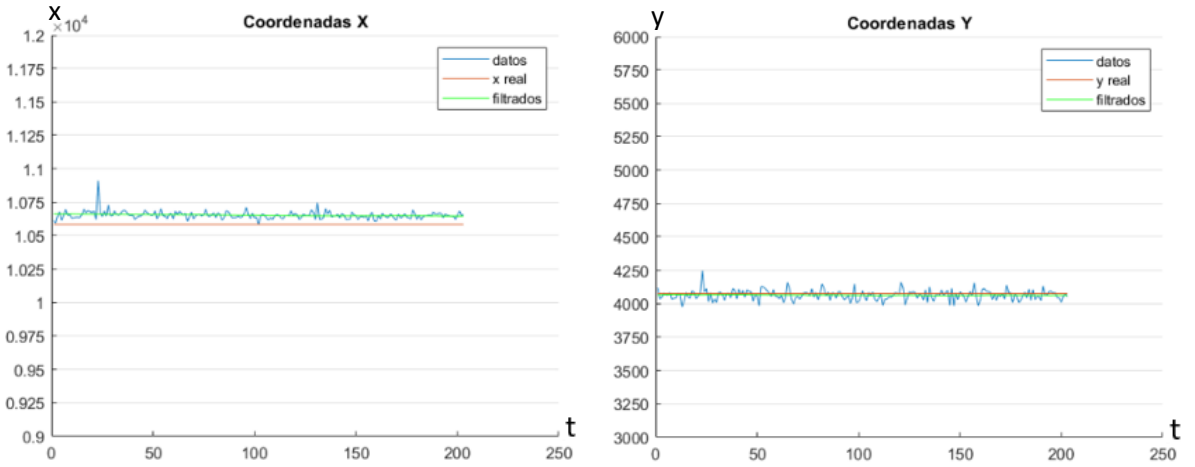


Ilustración 58 Valor de las coordenadas X e Y en función del tiempo para el Tag portable (Pos. 2)

- Posición 2. Tag de desarrollo:

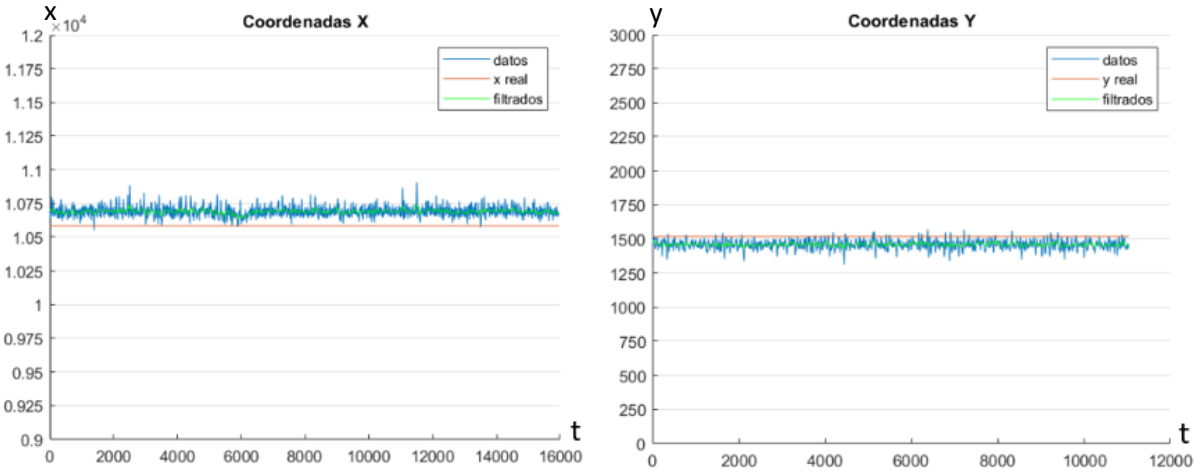


Ilustración 59 Valor de las coordenadas X e Y en función del tiempo para el Tag de desarrollo (Pos. 2)

- Posición 3. Minitag:

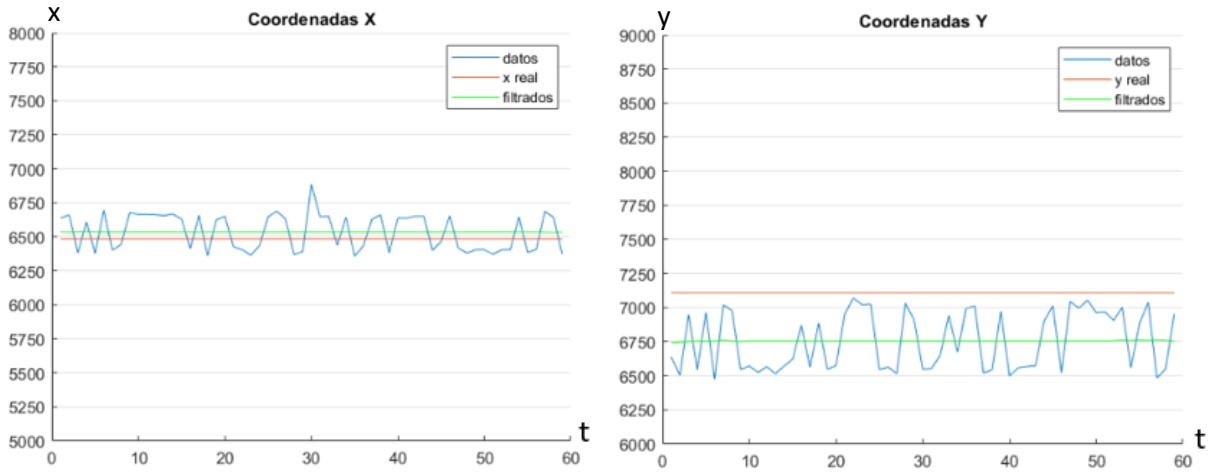


Ilustración 60 Valor de las coordenadas X e Y en función del tiempo para el Minitag (Pos. 3)

- Posición 3. Tag portable:

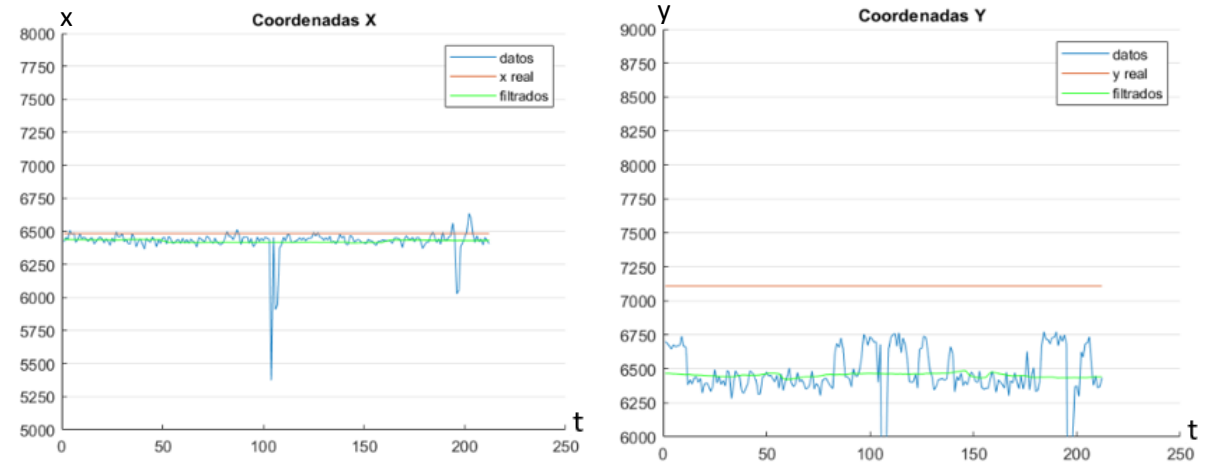


Ilustración 61 Valor de las coordenadas X e Y en función del tiempo para el Tag portable (Pos. 3)

- Posición 3. Tag de desarrollo:

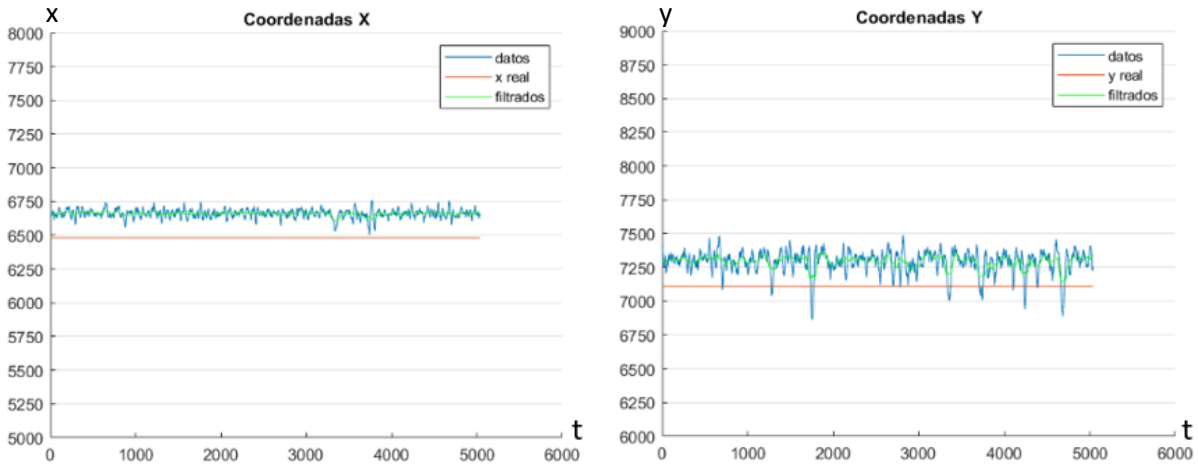


Ilustración 62 Valor de las coordenadas X e Y en función del tiempo para el Tag de desarrollo (Pos. 3)

Se observa cómo en la posición tres se tienen los datos menos precisos, puesto que en este punto del laboratorio hay una estantería metálica cercana a una célula robotizada que puede estar generando interferencias en la señal emitida por el tag. También se aprecia que en la posición dos se dan menos picos de ruido de medida que en la posición uno, pues como se ha mostrado anteriormente, en la posición uno hay menos cobertura de señal que en la posición dos. Al filtrar los datos a posteriori con un filtro de media móvil es posible eliminar el ruido de medida, aunque permanece el offset en el error de medida.

A continuación, se muestran estos datos filtrados (en verde) en su respectiva porción del mapa del laboratorio junto con los datos en bruto (en azul), así como el error cuadrático medio normalizado calculado para los datos en bruto y los datos filtrados.

Posición 1.

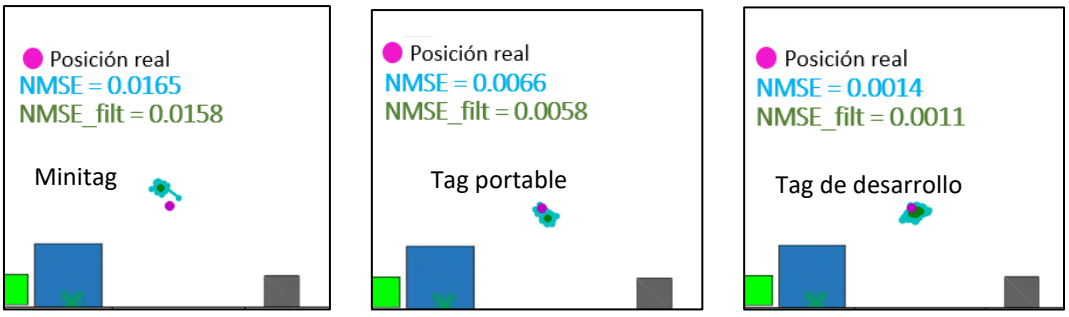


Ilustración 63 Datos de la posición filtrada y la posición obtenida por el sistema de Pozyx en el mapa (Pos. 1)

Posición 2.

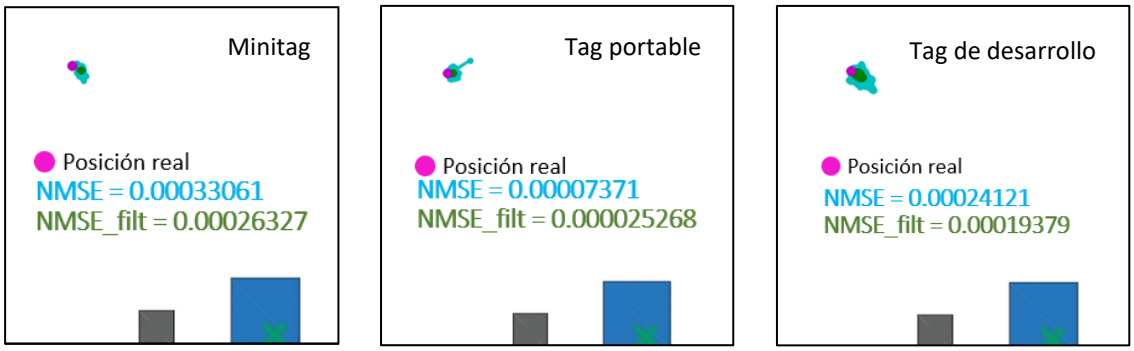


Ilustración 64 Datos de la posición filtrada y la posición obtenida por el sistema de Pozyx en el mapa (Pos. 2)

Posición 3.

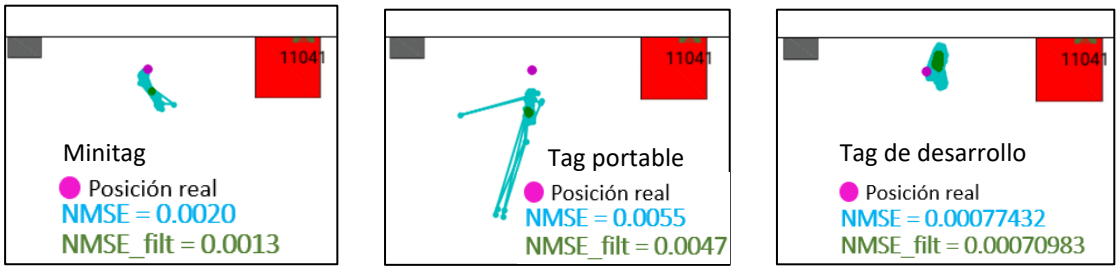


Ilustración 65 Datos de la posición filtrada y la posición obtenida por el sistema de Pozyx en el mapa (Pos. 3)

A partir de los datos extraídos se pueden tomar las siguientes conclusiones:

- En las zonas oscuras del plano, es más fiable emplear tags de desarrollo. Los tags portables y los minitags son muy sensibles a las interferencias.
- Los tags portables presentan errores más aleatorios que el resto de tags, pero éstos se pueden controlar con el filtrado posterior.
- Con un filtro de media móvil aplicado a posteriori se puede reducir más de diez veces el error de los datos en crudo.
- En el caso más desfavorable, una zona oscura o con interferencias, se tendrían errores máximos de 20-25 cm, lo que hace de la tecnología UWB un método fiable para la localización de activos en interiores.

En el caso de los tags de desarrollo, también es posible calcular la posición con el método TWR, como ya se ha visto. Por tanto, se han realizado diversas pruebas para poder comparar ambos métodos en las mismas condiciones de adquisición de datos. Además, Pozyx ofrece la posibilidad para este tipo de tags de prefiltrar los datos, lo que puede ser una gran ventaja respecto del resto de tags. A continuación se muestran los resultados obtenidos para los tags de desarrollo con los métodos TDOA, TWR y TWR con filtro de media móvil.

- Posición 1.

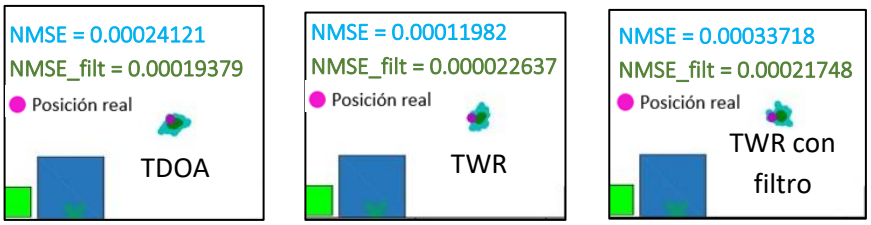


Ilustración 66 Comparación de los métodos TDOA, TWR y TWR con filtro en la posición 1

- Posición 2.

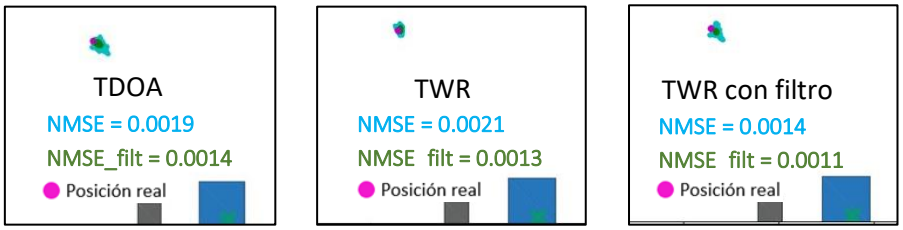


Ilustración 67 Comparación de los métodos TDOA, TWR y TWR con filtro en la posición 2

- Posición 3.

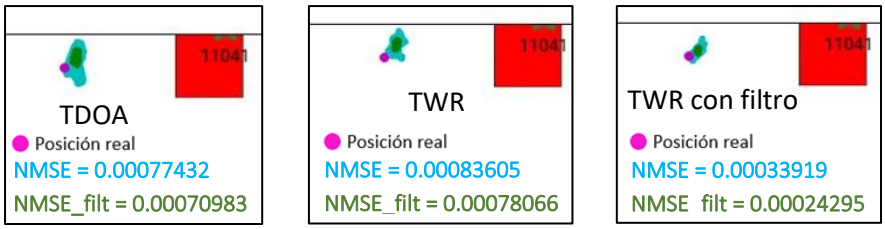


Ilustración 68 Comparación de los métodos TDOA, TWR y TWR con filtro en la posición 3

De estos experimentos realizados se puede concluir que el tag de desarrollo presenta gran precisión en los datos obtenidos, independientemente del método de posicionamiento empleado. Normalmente, el método TWR presenta mejores resultados (con o sin filtro), aunque no supone una gran mejora respecto al método TDOA.

Además, se ha observado que con el método TWR empeora la cobertura de la señal. Esto se debe a que al ser un método en el que se envía y se recibe la señal, se gasta más energía en el posicionamiento. Para limitar este aumento en el consumo de batería, se reduce la intensidad de la señal enviada, reduciendo así la precisión en el posicionamiento. Por este motivo se tienen resultados tan similares en el método TDOA y el método TWR.

En las imágenes mostradas a continuación se observa la cobertura en cada una de las posiciones estudiadas para los tags de desarrollo con el método TWR. Con una “X” se marcan las antenas que reciben una señal inferior a -120 decibelios. El software rechaza los datos recogidos por estas antenas. Asimismo, se han marcado en amarillo las antenas que reciben la señal con baja intensidad y en verde las que la reciben con una intensidad media.

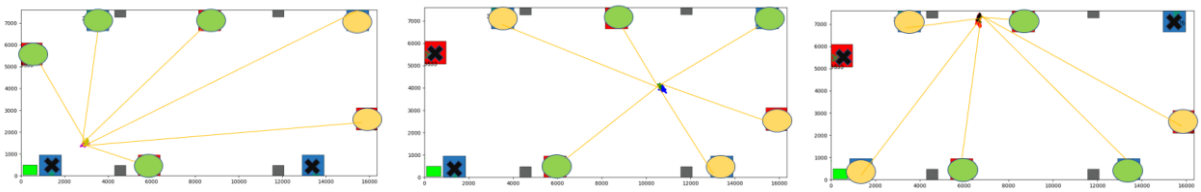


Ilustración 69 Cobertura de la señal con el método TWR

Por tanto, para obtener la posición de activos en estático es conveniente extraer los datos en crudo y aplicar un filtro a posteriori. El tag que presenta mejores resultados en condiciones adversas es el tag de desarrollo, aunque en condiciones normales puede ser preferible el empleo de tags portables de muy bajo consumo, asumiendo un error de posición ligeramente superior.

Por otro lado, para el tag de desarrollo no se han obtenido diferencias significativas entre los dos métodos de posicionamiento disponibles, debido a la limitación en el consumo de batería. Habría que valorar en cada caso lo que es deseable, aumentar la precisión en los datos de posición o la autonomía.

5.2. MEDIDAS DE PRECISIÓN PARA TAGS EN MOVIMIENTO

Además de las pruebas mencionadas en el apartado anterior, se han realizado también pruebas con los tags en movimiento. Para ello se ha situado cada tag encima del robot móvil RB1 de Robotnik, descrito en el apartado 3.4, con el objetivo de realizar la misma trayectoria en cada prueba. Se analizan dos tipos de trayectoria, una rectilínea y otra curvilínea. Para medir la precisión que se tiene de los datos de posicionamiento de los tags en movimiento, se mide con un láser la posición inicial y la posición final real del robot en su trayectoria, además de recopilar los datos que calcula el robot con la odometría. Así, tomando como la posición real los datos de la odometría del robot y las posiciones inicial y final, se calcula nuevamente el error normalizado medio cuadrático con la ecuación 2. Los resultados obtenidos se muestran en las siguientes gráficas en función del tipo de tag empleado, incluyendo los datos filtrados en Matlab a posteriori para cada caso.

En primer lugar, con los datos obtenidos para una trayectoria rectilínea con el método TDOA:

- Minitag:

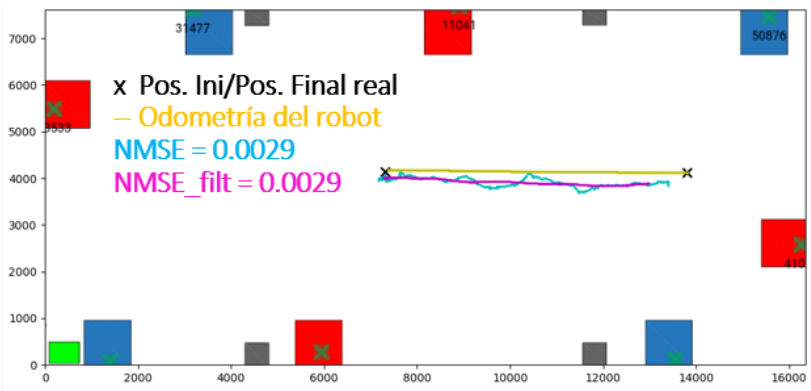


Ilustración 70 Datos filtrados y sin filtrar obtenidos para una trayectoria rectilínea con el minitag

- Tag portable:

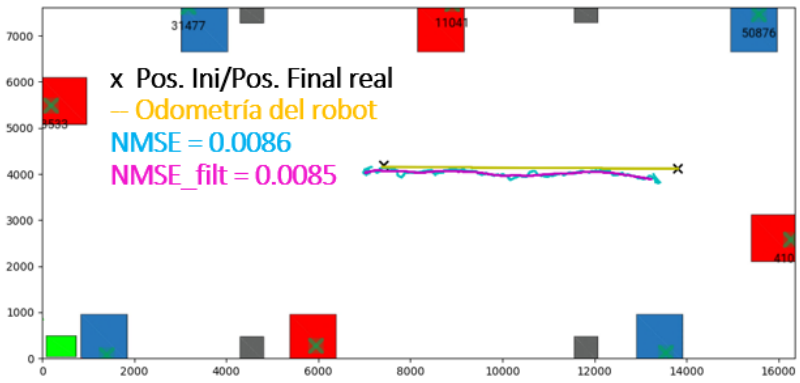


Ilustración 71 Datos filtrados y sin filtrar obtenidos para una trayectoria rectilínea con el tag portable

- Tag de desarrollo:

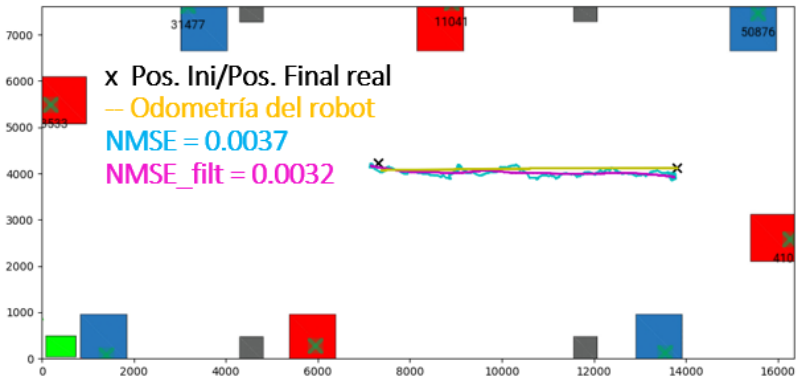


Ilustración 72 Datos filtrados y sin filtrar obtenidos para una trayectoria rectilínea con el tag de desarrollo

Realizando las mismas pruebas con una trayectoria curvilínea se obtienen los siguientes resultados.

- Minitag:

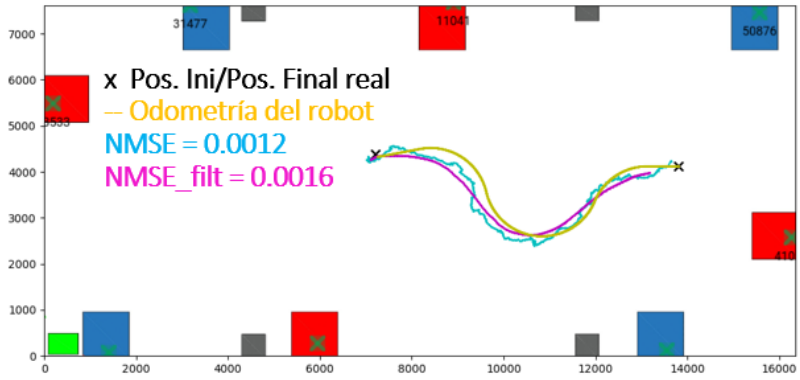


Ilustración 73 Datos filtrados y sin filtrar obtenidos para una trayectoria curvilínea con el minitag

- Tag portable:

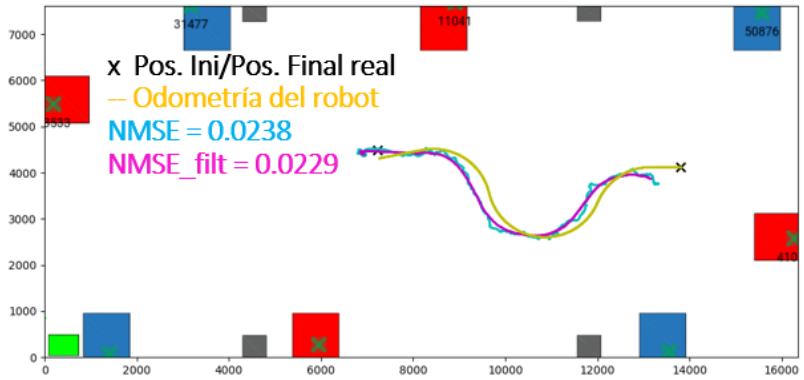


Ilustración 74 Datos filtrados y sin filtrar obtenidos para una trayectoria curvilínea con el tag portable

- Tag de desarrollo:

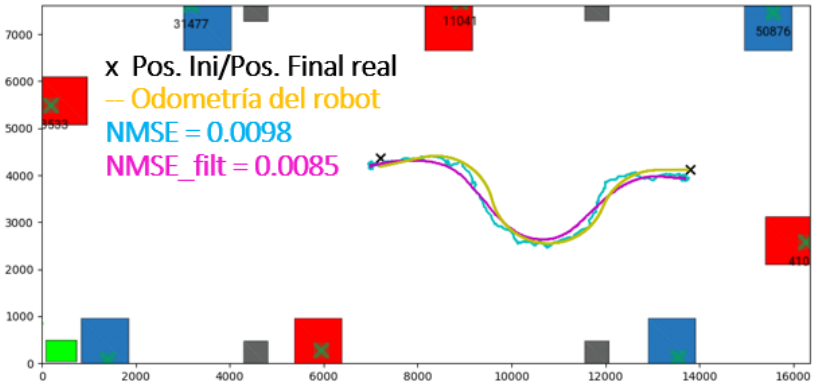
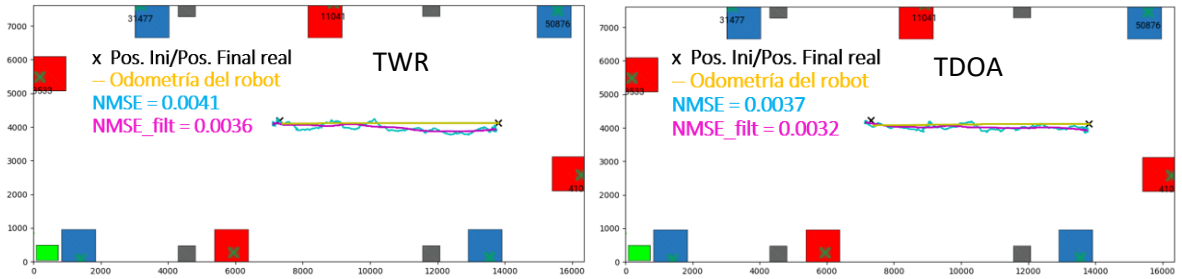


Ilustración 75 Datos filtrados y sin filtrar obtenidos para una trayectoria curvilínea con el tag de desarrollo

A partir de estas gráficas se puede concluir que los datos que se obtienen con los tags en movimiento son suficientemente buenos para aplicaciones de seguridad en la industria, con trayectorias muy aproximadas a las teóricas. Sin embargo, se observa que la odometría del robot no siempre coincide exactamente con las posiciones de inicio y final reales del robot, se da un pequeño error. Esto puede deberse a errores de redondeo en el radio de las ruedas del robot, o pequeños deslizamientos.

Nuevamente, el tag que presenta mayor precisión en el posicionamiento es el tag de desarrollo, que será el que se emplee en lograr los objetivos prácticos de este trabajo. Por tanto, se ha probado para este tipo de tag cómo afecta el emplear el método TDOA o el método TWR en los datos de posicionamiento para las mismas trayectorias del caso anterior.



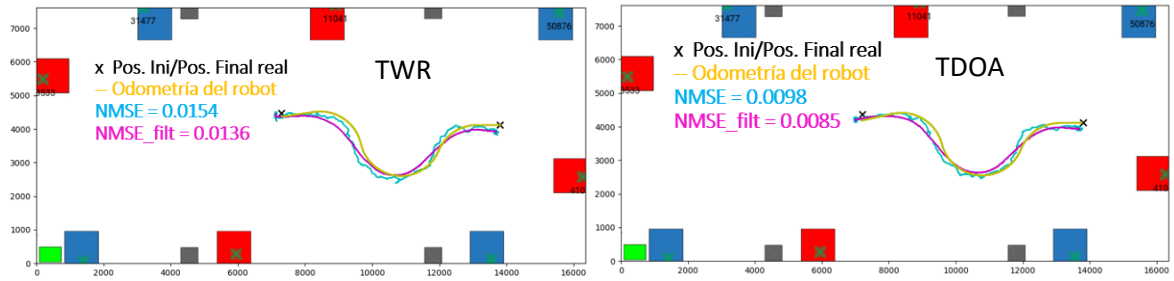


Ilustración 76 Comparación de los métodos TWR y TDOA para el tag de desarrollo con dos tipos de trayectoria

De nuevo no se dan diferencias significativas entre los dos métodos, seguramente por los motivos que se comentaron en el apartado anterior acerca de la duración de la batería del tag. No obstante, con los datos en bruto o con los datos filtrados, se tiene una buena aproximación a la trayectoria real seguida por el robot, la cuestión ahora sería determinar si es interesante emplear este sistema de localización de activos por UWB en los robots o podría bastar con leer la odometría del robot para poder implementar mejoras en determinadas aplicaciones de la industria.

Para resolver esta cuestión se han realizado más pruebas, esta vez para comprobar hasta qué punto es fiable la lectura de la odometría del robot.

5.3. MEDIDAS DE PRECISIÓN DE LA ODOMETRÍA DEL ROBOT

En robótica móvil, la odometría se refiere al sistema de detección de bajo nivel de un robot para medir su desplazamiento sobre una trayectoria determinada. Para medirla se suelen emplear sensores como encoders, potenciómetros, giroscopios, acelerómetros y similares.

Si bien es cierto que la odometría del robot cuando trabaja sin carga en un entorno industrial con suelo antideslizante, y con movimientos que no presenten cambios de velocidad puede resultar fiable. No obstante, dado que el diámetro teórico de las ruedas del robot no suele coincidir exactamente con el diámetro real, se va acumulando cierto error en la medida de la odometría conforme pasa el tiempo.

Para medir el error que se da en la odometría del robot en este caso, se ha llevado al robot a una velocidad constante durante diez minutos, recorriendo un total de 290 metros. La trayectoria realizada comienza y termina exactamente en el mismo punto del laboratorio, luego leyendo la posición que calcula el robot al principio y al final del recorrido se podrá determinar el error que tiene en la odometría al final del experimento. En este caso:

$$\Delta x = 7.63 \text{ cm}$$

$$\Delta y = 14.65 \text{ cm}$$

Es el error de posición que se tiene al cabo de diez minutos, por tanto, si se tiene al robot activo durante horas, el error será mayor.

Por otro lado, cuando se dan cambios en la aceleración del robot se pueden producir deslizamientos. Este caso también se ha probado en el laboratorio obteniéndose el siguiente error de posición.

$$\Delta x = 6.8 \text{ cm}$$

$$\Delta y = 27.76 \text{ cm}$$

Teniendo en cuenta que el suelo del laboratorio es bastante bueno para este caso, puesto que es rugoso y sin desniveles, se tiene un error de casi 30 cm. Si se pretende tener trabajando al robot durante la producción con sus arranques y sus paradas en un periodo prolongado, es probable que este error sea mucho mayor.

Con todo ello, se puede concluir que sí que resulta conveniente la utilización del sistema de localización por UWB para obtener el posicionamiento en tiempo real de robots en la industria.

CAPÍTULO 6. DESARROLLO DE APLICACIONES INDUSTRIALES BASADAS EN EL SISTEMA DE LOCALIZACIÓN

Una vez analizadas las ventajas que puede proporcionar el sistema de localización de Pozyx para el cálculo del posicionamiento de activos en entornos industriales, se han desarrollado algunas aplicaciones que podrían facilitar y/o optimizar el funcionamiento de los robots móviles en espacios cerrados con interacción humano-robot. Así, se ha integrado la información recogida por el sistema de localización en un módulo ROS que permite conectarse a través de un puente MQTT al Gateway de Pozyx o mediante puerto serie al propio tag situado en el robot móvil. Con esta información se puede calcular la posición inicial absoluta del robot así como su orientación, también se puede conocer la posición de determinados obstáculos (ya sean humanos, objetos o robots) para realizar la evitación u optimizar las rutas conociendo la ubicación previa de barreras, mejorando así la eficiencia de las trayectorias. También se puede emplear este sistema para el seguimiento de tags en aplicaciones de logística, lo que puede ser muy interesante para transportar pequeñas cargas de manera puntual sin que el operario tenga que cargar con estas en su itinerario de producción.

Además, este sistema de localización se puede emplear para aplicaciones con robots industriales en el ámbito de la seguridad, por ejemplo, disminuir la actividad del robot cuando detecte que se acerca un humano a su zona de trabajo.

Estas aplicaciones se describen en este apartado, siendo el objetivo práctico principal de este trabajo. Para las aplicaciones 6.1, 6.2, 6.3 y 6.4 se ha desarrollado el código que se muestra en el apartado 3 del documento Anexo I.

6.1. MODOS DE CONEXIÓN

Existen diversas posibilidades de conexión en el módulo ROS desarrollado. Para entender los distintos modos, hay que conocer los datos que se pueden extraer de los tags de desarrollo en sus dos métodos de funcionamiento TDOA o TWR. Con el método TWR el tag de desarrollo sólo devuelve la posición, mientras que con el método TDOA se pueden escoger más datos de lectura, por ejemplo, la orientación del tag en forma de ángulos de Euler.

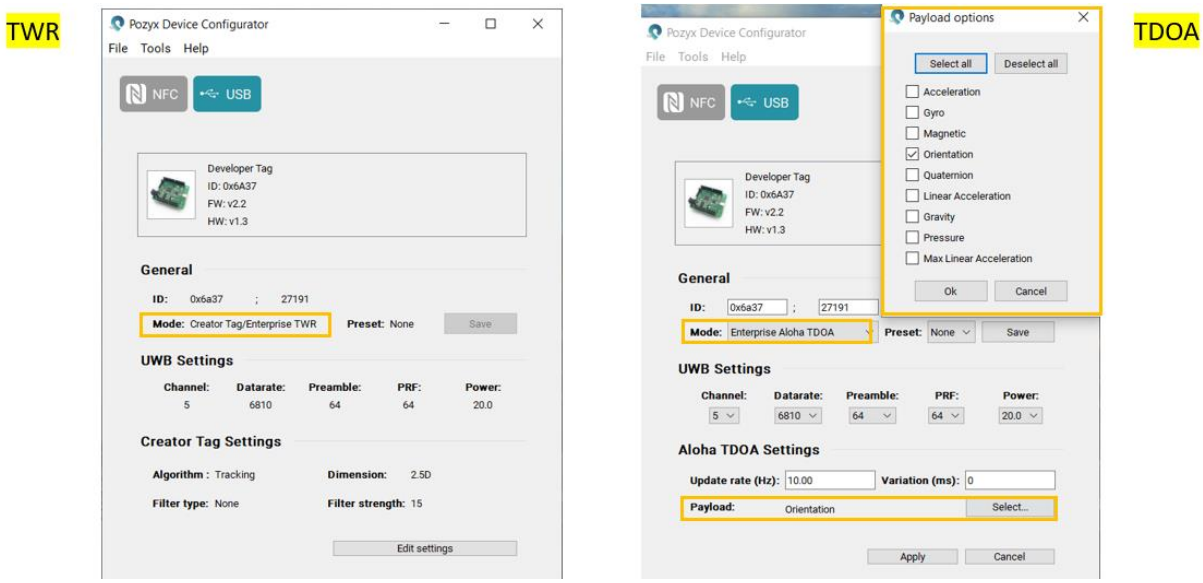


Ilustración 77 Menú de configuración del tag de desarrollo en la aplicación Pozyx Device Configurator

Con estas opciones se tienen las tres opciones de funcionamiento mostradas en el siguiente esquema. Por un lado, creando un hilo de conexión por MQTT se puede conocer tanto la posición del robot con su correspondiente tag, como la posición del resto de tags. Si además se emplea el método de comunicación por TDOA se puede conocer también la orientación del robot (o del resto de tags). Por otro lado, es posible conectar el tag por puerto serie al robot sólo con el método TWR si se desea extraer la información directamente desde el tag. Conectando por puerto serie es posible extraer los mismos datos que permite el método TDOA, en este caso, la orientación del tag. Sin embargo, no es posible conocer la posición del resto de los tags.

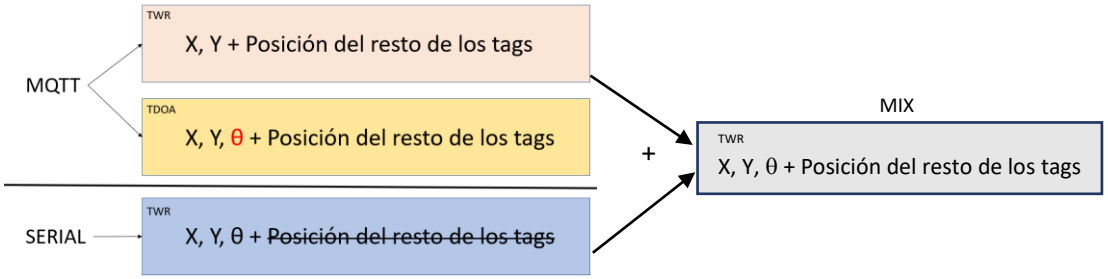


Ilustración 78 Esquema de posibilidades de obtención de datos en función de los métodos de cálculo de la posición disponibles

Por tanto, un modo de conexión interesante sería realizar una mezcla entre los descritos con el método TWR. De cualquier forma, en el módulo desarrollado se carga un fichero de configuración en formato JSON donde se especifican las preferencias de funcionamiento. En este fichero se puede elegir si aplicar un filtro a los datos de posición leídos, y de aplicarlo, se puede escoger un filtro de media móvil o un filtro de mediana.

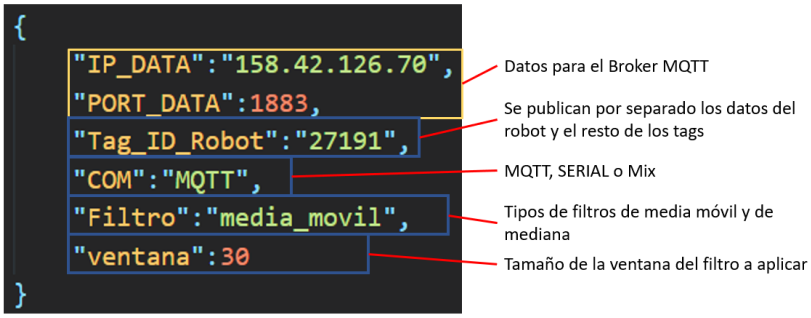


Ilustración 79 Fichero de configuración para el módulo ROS

6.2. CÁLCULO DE LA POSICIÓN Y ORIENTACIÓN INICIALES

Para poder iniciar la navegación con el robot RB-1 es necesario disponer de un mapa del entorno. Esto se puede realizar a través de la interfaz que provee Robotnik para facilitar el proceso. En primer lugar, es necesario realizar el mapeado del laboratorio, a continuación, se carga este mapa en el robot.

Ilustración 80 Mapa del laboratorio cargado en el robot

Una vez cargado el mapa, hay que indicarle al robot en qué posición y con qué orientación se encuentra ubicado de forma aproximada, ya que el robot por defecto se carga en la posición del eje de coordenadas del mapa como se muestra en la imagen superior. El robot podrá iniciar la autolocalización y la autonavegación cuando conozca su posición en el mapa, aumentando la precisión mediante el algoritmo probabilístico de localización AMCL (Adaptive Monte-Carlo Localizer). Este algoritmo emplea la odometría, el sensor láser y la cámara para generar un mapa de puntos y poder realizar una imagen del entorno. El robot continuamente trata de hacer coincidir este mapa de puntos con el mapa cargado, hasta tener una posición lo bastante precisa.

Ilustración 81 Robot ubicado en su posición real en el mapa

Mediante el sistema de localización de Pozyx es posible establecer la posición y la orientación iniciales del robot en el mapa de forma inmediata al encender el robot, sin necesidad de indicarlo en la interfaz. Para ello, se ha de fijar un tag en la superficie del robot y leer el ángulo de cabeceo que ofrece el sensor de brújula respecto del norte. El cálculo de la orientación absoluta inicial del robot se realiza mediante la fórmula del producto escalar entre dos vectores, en este caso, entre un vector arbitrario y el vector de la orientación del robot respecto al norte.

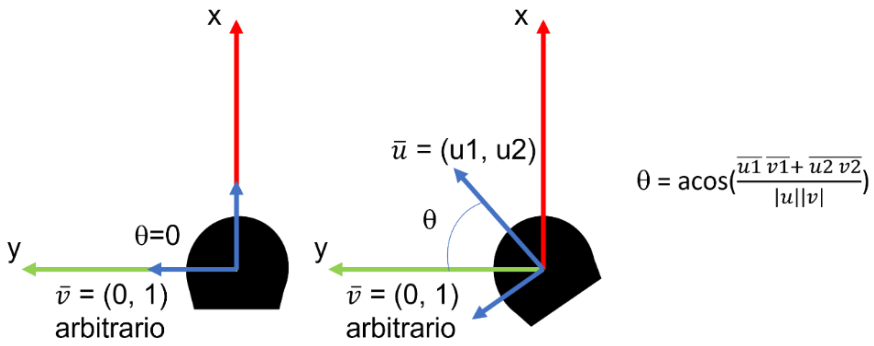


Ilustración 82 Esquema de cálculo de la orientación relativa del robot respecto de un eje arbitrario

Esto dota de mayor autonomía al robot y supone una gran ventaja a la hora de iniciar la autolocalización del robot, pues disminuye la carga computacional del algoritmo AMCL del robot. Además, en temas de seguridad industrial, esto supone que no dependa de un usuario la localización inicial del robot y su orientación. En ocasiones esto puede desencadenar un accidente cuando se trata de enviar al robot a una determinada posición y el robot no sabe dónde está situado en el mapa.

6.3. EVITACIÓN DE OBSTÁCULOS

En espacios de trabajo donde la actividad humana se complementa con la actividad sistemática de equipos robotizados, es primordial guardar la seguridad de los activos involucrando a tal propósito todos los medios que sean necesarios. Concretamente, en aquellas áreas donde se disponga de robots móviles autónomos conviene asegurar el trazado de trayectorias seguras, libres de colisiones. Esta aplicación es muy importante en el ámbito de la robótica móvil, se trabaja continuamente en la mejora de los algoritmos encargados de realizar la evitación de obstáculos con el objetivo de

optimizar los tiempos de cálculo de la replanificación de las trayectorias a realizar en presencia de obstáculos, así como para prevenir los riesgos inherentes a la actividad desarrollada en el entorno de trabajo.

El éxito de esta aplicación requiere mucha precisión en la localización de los obstáculos en tiempo real. Para mejorar la precisión de estas medidas se emplean sensores actualizados y muy competentes en el mercado que dotan al robot móvil de autonomía suficiente para poder decidir sus movimientos dentro del área de trabajo mientras se están realizando otras aplicaciones que puedan interferir en su ruta normal. Los sensores más comúnmente utilizados para la detección de obstáculos son los sensores láser, sensores de visión y los sonar o sensores de ultrasonidos.

Ilustración 83 Sensor láser [20], sonar [21] y cámara RGBD [22]

En primer lugar, los sensores láser calculan la distancia a la que se encuentra el robot de un obstáculo emitiendo rayos láser que se reflejan en los objetos. Esta técnica se conoce como "Tiempo de Vuelo" (TOF), que consiste en medir el tiempo que tarda el rayo láser emitido en alcanzar un obstáculo y volver a su origen. Así, la distancia es la mitad del valor que se obtiene de multiplicar este tiempo por la velocidad de la luz. Estos sensores son muy precisos y tienen un alcance superior a los 20 metros de longitud, lo que suele ser suficiente en las aplicaciones de la industria. Tienen tiempos de respuesta muy rápidos y abarcan rangos de circunferencia bastante amplios (pudiendo alcanzar rangos de 300º), dependiendo de cada sensor. Además, no presentan problemas cuando en el entorno hay objetos de materiales de diversas texturas y/o colores. Sin embargo, sí que se ven afectados por aquellos cuyas superficies sean de material reflectante, disminuyendo la precisión del sensor. A continuación se puede observar cómo genera la imagen del entorno un robot móvil con un sensor láser instalado.

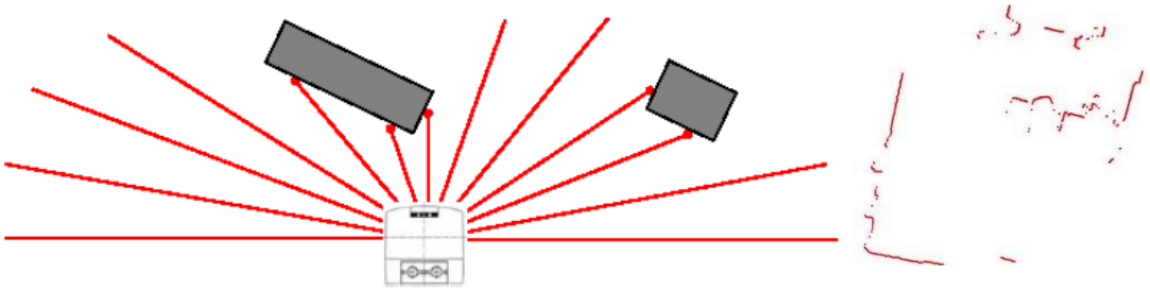


Ilustración 84 Detección de obstáculos mediante un sensor láser

En segundo lugar, los sensores ultrasónicos miden las distancias mediante el uso de membranas. La membrana emisora se excita con una señal eléctrica creando un tren de pulsos (ondas de ultrasonidos). Estas ondas rebotan en los obstáculos y vuelven a la membrana receptora que con la

vibración genera una señal eléctrica que se traduce en un valor de distancia empleando la técnica de TOF o midiendo la diferencia de fases entre la señal emitida y la recibida. Estos sensores suelen tener un alcance de 5 a 10 metros de distancia, lo que puede resultar insuficiente en determinadas aplicaciones. Además, las ondas de ultrasonidos pueden traspasar ciertos objetos o reflejar, produciendo ecos y disminuyendo por tanto la precisión de la medida de distancia.

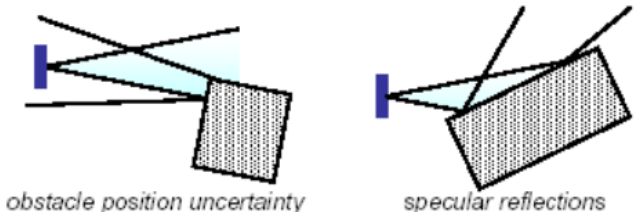


Ilustración 85 Reflexión de las ondas ultrasónicas en un objeto

En tercer lugar, los sensores de visión capturan la luz del entorno y la traducen en una imagen donde cada píxel se calcula a partir de la señal eléctrica generada por la luz. En robótica son comunes las cámaras RGBD (Red-Green-Blue-Depth). Estos sensores, además de suministrar información visual (imágenes en color), aportan información de profundidad con un rango de precisión aceptable. Con estas cámaras se puede elaborar un mapa del entorno a través de una nube de puntos, con un rango de hasta 8 metros, en función de la cámara empleada.

El robot RB-1 disponible en el laboratorio cuenta con un sensor láser y una cámara RGBD para detectar los obstáculos y generar un mapa del entorno. Sin embargo, pueden darse situaciones donde estos sensores no cubran en su totalidad el área de trabajo y el robot puede verse expuesto a colisiones. Con esta aplicación se pretende cubrir un rango de 360º de visión, pudiendo anticipar incluso aquellos obstáculos que se encuentren en la parte trasera del robot, fuera de su rango de visión normal. Además, no afecta el material de los objetos del entorno ni la distancia a la que se encuentran situados.

Para generar el mapeado del entorno de trabajo, se distinguen dos tipos de mapas empleados en la planificación de trayectorias y evitación de obstáculos: el mapa de costes global y el mapa de costes local. El mapa de costes global es una Matriz de Ocupación, es decir, el robot durante el mapeado genera una matriz o imagen donde cada celda o píxel tiene un valor de ocupación, tomando el valor de objeto o zona de libre acceso. De esta forma, cuando se desea planificar una trayectoria con una meta, el robot calcula la ruta óptima global a partir de los datos cargados de la matriz de ocupación evitando los obstáculos que conoce a priori, sin necesidad de usar los sensores que tiene disponibles. Por otro lado, el mapa local utiliza los sensores disponibles en el robot (láser y cámara RGBD en este caso) para localizar obstáculos puntuales que no son fijos, o no estaban cuando se realizó el mapeado. Este mapa de costes local se va generando a medida que el robot sigue la trayectoria planificada en base al mapa global, calculando mediante un determinado algoritmo la ruta óptima para sortear los obstáculos que aparecen en la ruta. El esquema del planificador es el siguiente.

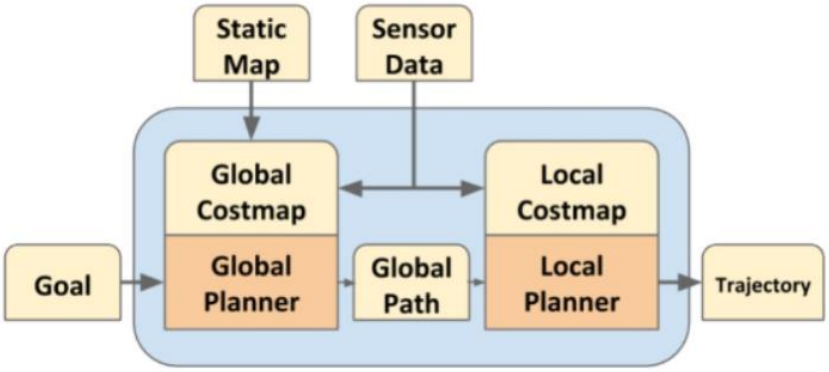


Ilustración 86 Esquema de funcionamiento interno del mapa de costes del robot [16]

El objetivo de esta aplicación es generar una capa que contenga los tags leídos por el sistema de localización de Pozyx en el mapa de costes local del robot para realizar la evitación de obstáculos, una capa adicional que sea independiente de la lectura de los sensores disponibles en el robot. Es decir, un sensor virtual que podría implementarse en cualquier robot móvil independientemente de su estructura y funcionalidades.

Para realizar la evitación de obstáculos ROS recoge los datos que proveen los sensores láser, sonar o cámaras. Cada uno de estos sensores genera los datos con una estructura diferente, por ejemplo, la cámara genera los datos en forma de nube de puntos o PointCloud mientras que el láser sigue una estructura bidimensional angular denominada LaserScan. Esta será la escogida para la inserción de los tags como obstáculos en el mapa local del robot RB1 en este trabajo. Este tipo de mensaje tiene la siguiente estructura de datos:

```
Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

Ilustración 87 Estructura del mensaje tipo LaserScan de ROS [17]

Por tanto, se ha definido este tipo de mensaje para la capa de los tags leídos por el sistema de localización del laboratorio. Como se trata de una capa ficticia y no hay límites físicos, se ha escogido un rango de 360 grados de visión, lo que dotaría al robot de visión multidireccional incluyendo la zona que no puede cubrir el sensor láser o la cámara frontal. La resolución angular o "angle_increment" es de un grado, pues en base a las pruebas realizadas, resultó ser suficiente. El

vector "ranges" contiene las distancias en metros de los obstáculos que rodean al robot. Para generar este vector, es necesario conocer con qué ángulo relativo observa el robot a cada tag u obstáculo. En el siguiente esquema se muestra cómo se ha calculado dicho ángulo β, siendo α el ángulo de cada tag en base a un sistema de coordenadas absoluto (el del sistema de localización de Pozyx) y θ el ángulo que mide la orientación relativa del robot respecto de dicho sistema de coordenadas, calculado como se explicó en el apartado anterior.

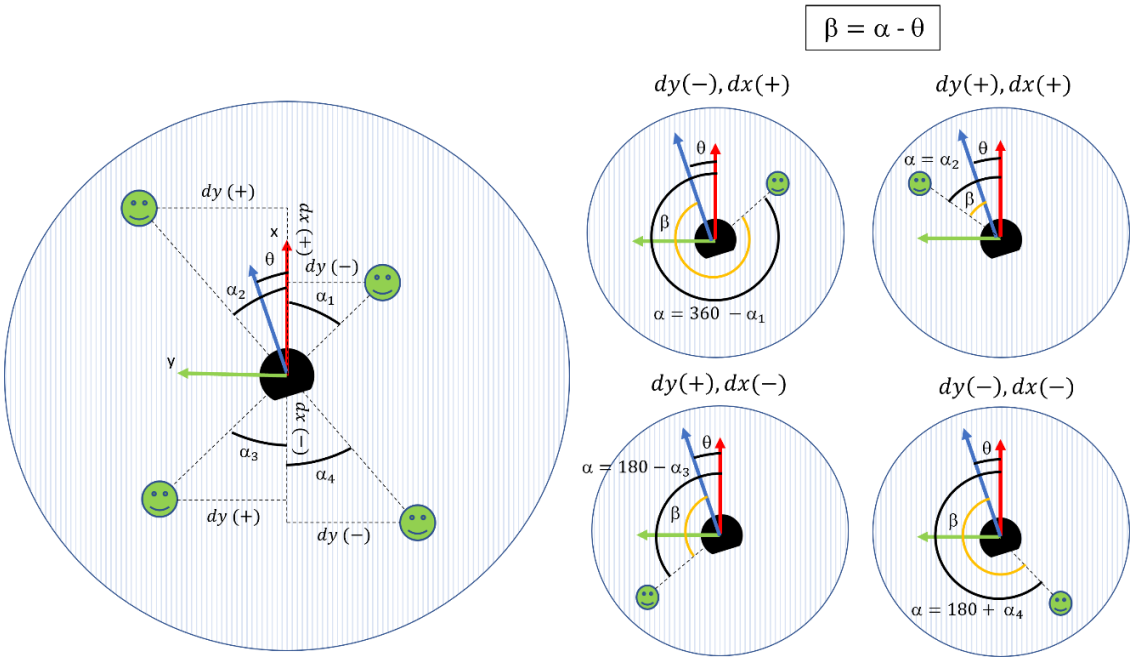
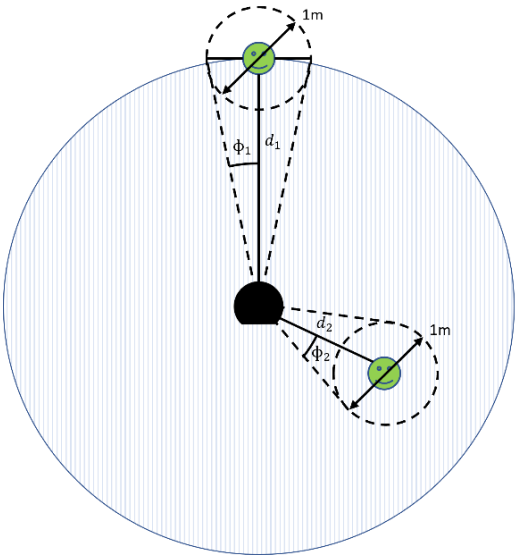


Ilustración 88 Esquema de cálculo del ángulo β en función del cuadrante

Una vez calculado β, se añade un margen de seguridad de un metro de diámetro alrededor del tag. Este margen de seguridad se ha calculado como un ángulo ϕ según el esquema que se muestra a continuación, variando en función de la distancia entre el robot y el tag para mantener siempre el mismo diámetro.



$$\phi = \operatorname{atan}\left(\frac{0{,}5}{d}\right)$$

$$d = \sqrt{(x_{tag} - x_{robot})^2 + (y_{tag} - y_{robot})^2}$$

Ilustración 89 Esquema de cálculo del ángulo ϕ en función de la distancia del tag al robot

Conocidos los ángulos de β y ϕ, es posible generar el mensaje del tipo LaserScan. Cuando se crea el mensaje, se inicializan todas las celdas a un número elevado (por ejemplo, 20 metros) y sólo se modifican las celdas cuya posición esté en el rango de [β-ϕ, β+ϕ] rellenándolas con la distancia d, la distancia entre el robot móvil y el tag. Cuando se dé el caso en el que haya dos tags en la misma línea de visión pero a distintas distancias, se rellena el vector con la distancia más próxima.

Tras realizar los ajustes adecuados en el ángulo β para ajustarlo al rango de [-180º, 180º] ([angle_min, angle_max]), se tiene el siguiente esquema donde se observa que el ángulo cero quedaría al frente del robot, coincidiendo con la mitad del vector de "ranges".

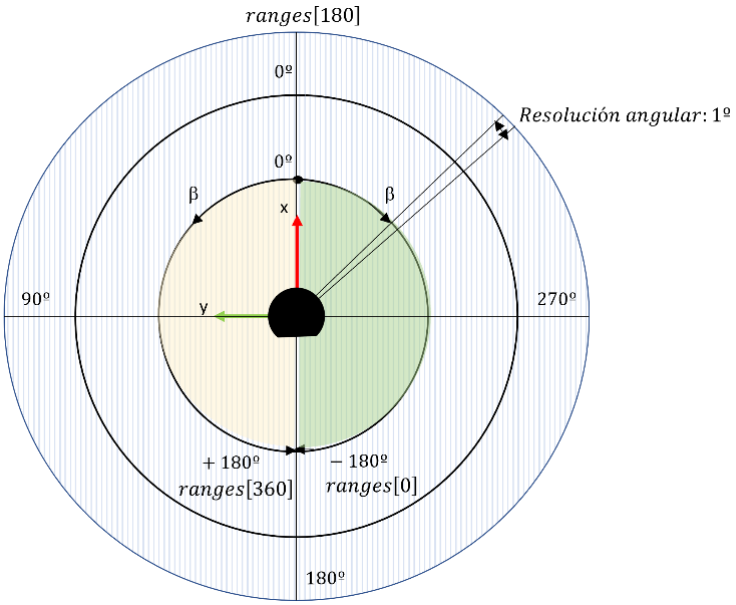


Ilustración 90 Equivalencia de los ángulos para generar el mensaje LaserScan

La posición del robot (x_{robot}, y_{robot}) en esta aplicación se toma directamente del algoritmo AMCL del robot. Para ello, en un hilo de ejecución independiente en el código, se realiza la suscripción al topic donde el robot publica su posición de forma continua.

Una vez generado el mensaje, se publica en un topic personalizado "/my_laser" de forma ininterrumpida. Para incluir esta capa en el cálculo del mapa de costes local hay que indicar en su fichero de configuración que debe leer la información que se publique en dicho topic, tal y como se muestra en la siguiente porción de código.

```
obstacle_layer:
  obstacle_range: 2.5
  raytrace_range: 5.5
  observation_sources: tags front_laser
  front_laser:
        sensor_frame: robot_front_laser_link
        data_type: LaserScan
        topic: front_laser/scan
        marking: true
        clearing: true
```

```
        observation_persistence: 0.0
tags:
        sensor_frame: robot_front_laser_link
        data_type: LaserScan
        topic: my_laser
        marking: true
        clearing: true
        observation_persistence: 0.0
```

Una vez realizado este paso, ya se incluyen los tags leídos por el sistema de localización en el mapa de costes local del robot. Esto se puede comprobar en el visor tridimensional de ROS denominado RVIZ, marcando la opción de visualización del mapa de costes local. En la siguiente imagen se puede observar la información del laser real del robot (en colores cálidos) así como de la capa creada a partir de la lectura de los tags (en azul). El área gris que rodea los datos provenientes del sensor láser y del sensor virtual creado es la capa de inflación de los objetos que aparecen en el mapa de costes local. Esta capa se encarga de aumentar el área de seguridad de todos aquellos objetos que deben ser evitados por el robot en su ruta, facilitando el control del robot a su origen de coordenadas.

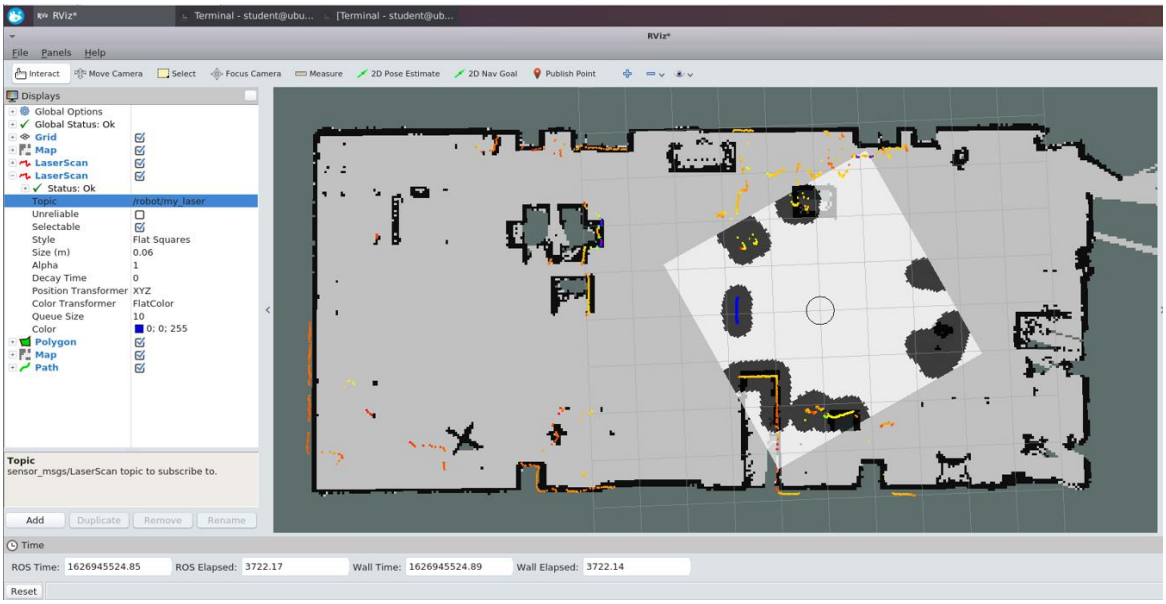


Ilustración 91 Visualización de la capa creada en el costmap en RVIZ

Al incluir esta información en el mapa de costes local, el robot sorteará los tags cuando se interpongan en la ruta calculada por el mapa de costes global. En la figura inferior se puede observar la secuencia de evitación del tag suspendido en el aire que se ha situado en el laboratorio. Al estar suspendido, el robot no puede visualizar el tag con el sensor láser y hace evidente el correcto funcionamiento de esta aplicación. El robot sortea un obstáculo que no puede ver.

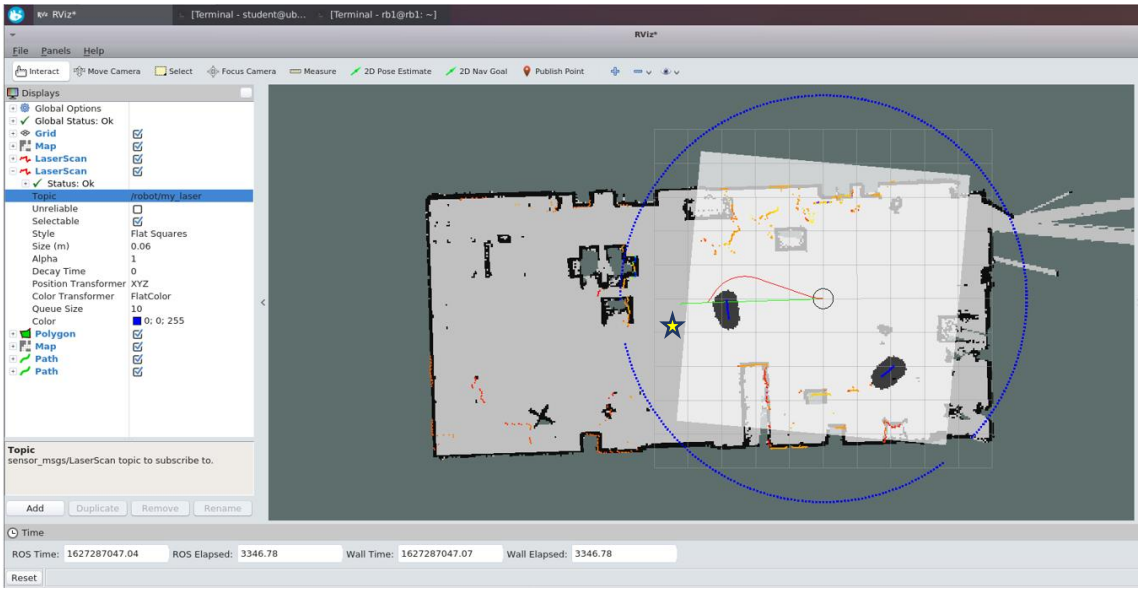


Ilustración 92 Trayectoria generada por el mapa global (verde) y la generada por el mapa de costes local (roja)

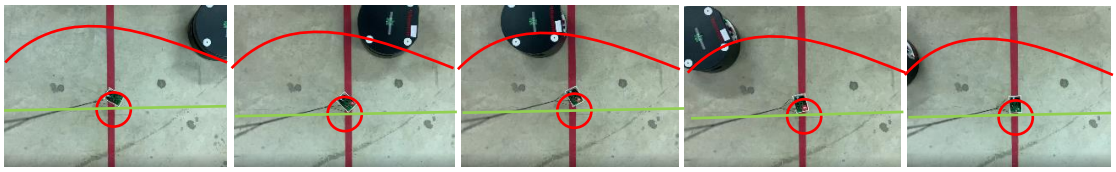


Ilustración 93 Secuencia de evitación del tag por el RB-1

El vídeo de esta aplicación se puede visualizar en https://youtu.be/mNCkvJZcqUA

6.4. OPTIMIZACIÓN DE TRAYECTORIAS BASADA EN INFORMACIÓN DE LOCALIZACIÓN DE BARRERAS

Con el sistema de localización empleado, sería relativamente sencillo estudiar el comportamiento de humanos y/o robots en el entorno productivo. Con los datos de posicionamiento se pueden conocer fácilmente cuáles son las áreas más concurridas, o las trayectorias que se efectúan para realizar determinados procesos. Estos datos se pueden emplear para analizar la productividad derivada del sistema de producción implementado en una empresa, si se realizan recorridos innecesarios, o si por el contrario se podrían optimizar ciertas trayectorias con el uso de nuevos dispositivos. También es posible gestionar de forma automática el stock en almacenes, conociendo la ubicación precisa de determinados materiales, así como la información de cuándo salen o entran en el almacén. Son múltiples las aplicaciones que permite la implementación del sistema de localización en la industria.

En este apartado, en concreto, se propone su utilización para optimizar las trayectorias de un robot móvil con un campo de visión limitado determinado por sus sensores convencionales, como son el sensor láser y la cámara. Así, el hardware del robot limita la optimización de trayectorias al incluir en el mapa de costes local únicamente los obstáculos localizados dentro del campo visual del robot. Luego una aplicación muy interesante puede ser la de añadir obstáculos en el mapa de costes que no se pueden visualizar con los sensores descritos. Esto se realiza de forma análoga a lo realizado en la aplicación anterior, pero en este caso, en vez de realizar la evitación de obstáculos se realizará una

planificación de la ruta que resulte óptima en presencia de objetos o barreras indetectables por los sensores del robot.

Por ejemplo, en la siguiente imagen se introduce el robot en un laberinto cuyo mapa global es conocido. Sin embargo, en el instante en el que se envía al robot a la meta se crea una obstrucción en la ruta más corta que no está dentro del rango visual del robot. Esto provoca que el robot inicie su trayectoria más corta hasta toparse con la barrera, teniendo entonces que retroceder hasta el punto inicial e iniciar nuevamente su recorrido hacia la meta por la segunda ruta posible. En este caso, el robot duplica prácticamente su trayectoria, consumiendo tiempo y energía.

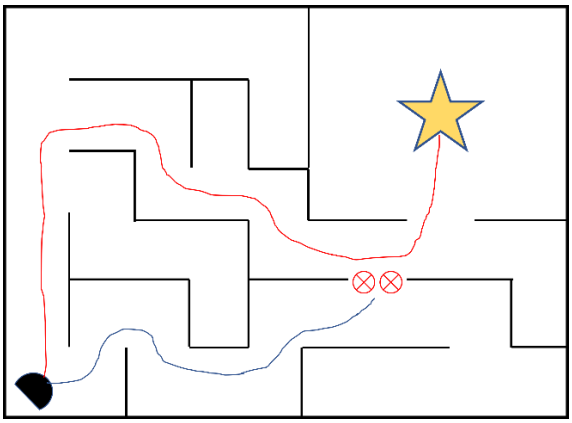


Ilustración 94 Optimización de trayectoria conocida la ubicación previa de barreras

Esta aplicación se ha llevado a cabo en el laboratorio de trabajo, aumentando el tamaño del mapa de costes local del RB-1 para que inicie los cálculos de optimización de ruta con obstáculos en un rango de 10 metros de distancia. Se ha colocado un tag a modo de barrera dentro de la ruta más corta hacia la meta, pero fuera del alcance de la visión del robot. En la siguiente imagen se puede observar cómo la ruta generada por el planificador global no tiene en cuenta esta barrera (línea verde), sin embargo, la ruta generada por el planificador local comienza a modificar esa primera ruta calculada para sortear dicha barrera (línea roja). De esta forma, el robot no tiene que entrar en el pasillo que se observa en el mapa para ver que hay una barrera que no puede cruzar, sino que modificará progresivamente su trayectoria inicial hasta alcanzar la meta por una ruta alternativa aunque no sea en un principio la más óptima.

Ilustración 95 Ruta global inicial y replanificación de la ruta local del robot conocida una barrera

En este caso el robot no mantendrá su trayectoria hasta poder visualizar la barrera, sino que se anticipará a esta y modificará su ruta de forma progresiva sin necesidad de ver con el sensor láser la barrera.

Esta aplicación puede resultar útil en espacios con mucha afluencia de trabajadores. La ruta óptima del robot puede verse obstaculizada por la concentración de un grupo de personas en un habitáculo de paso. O por necesidades productivas, pueden localizarse útiles de trabajo en una determinada zona de la ruta del robot de manera provisional que dificulten su paso o pongan en riesgo su estructura.

Por otro lado, en ocasiones puede ser deseable modificar la ruta que realiza el robot en función de la carga que transporta, por ejemplo, si el robot traslada carga delicada y en la ruta óptima que calcula debe introducirse por un pasillo con un suelo en malas condiciones, se podría insertar una barrera ficticia en el mapa de costes local del robot para que de forma autónoma este evite su paso por dicha zona. El robot recalcularía su trayectoria hasta alcanzar su meta por una vía segura, aunque más larga.

El vídeo de esta aplicación se puede visualizar en https://youtu.be/afMGcox5Ivs

6.5. SEGUIMIENTO DE TAGS

Actualmente existe tendencia al estudio de aplicaciones para el seguimiento de robots a humanos en diversas situaciones. Es aquí donde destaca el campo de la visión artificial en la robótica. En este ámbito se estudian algoritmos que logren el reconocimiento de humanos y sus características para poder establecer unas pautas de funcionamiento en los robots y poder realizar un seguimiento selectivo. Gracias a estos estudios se dan situaciones cotidianas que son más sencillas y pueden servir de gran ayuda a las personas. Un ejemplo de este tipo de aplicaciones, donde la robótica de servicios es la protagonista, es la utilización de un robot móvil para mantener un tanque de oxígeno siempre cerca de un paciente con una enfermedad pulmonar, evitándole realizar el esfuerzo que supone arrastrar consigo el tanque cuando el paciente necesite desplazarse. Otro ejemplo puede ser

el uso de un robot móvil para el transporte del equipaje de los pasajeros en un aeropuerto, realizando el seguimiento de su trayectoria.

Por otro lado, la robótica de servicios está en auge en el sector logístico donde la automatización de los procesos y la optimización de los recursos está en constante crecimiento, tratando de reducir las tareas pesadas o tediosas a los operarios. La industria requiere cada vez más de robots autónomos que puedan desplazarse libremente por el entorno de trabajo, sin necesidad de estar bajo constante supervisión de los operarios.

Sin embargo, para poder realizar el seguimiento del humano, el robot primero necesita identificarlo. Aprovechando las ventajas que proporciona el sistema de localización disponible en este caso, resulta más sencillo realizar el seguimiento mediante tags sin necesidad de integrar la visión artificial en el robot, como es habitual. Este sistema aporta al robot la información de la posición del humano, sin necesidad de procesar los datos obtenidos por sus sensores. Además, como cada tag tiene su identificador, al robot sólo hay que indicarle a qué tag debe seguir.

En esta aplicación desarrollada se busca liberar a los operarios de transportar cargas pequeñas o medianas por las zonas de producción, lo que garantiza mayor comodidad al operario así como mejorar la prevención de riesgos en términos de seguridad industrial. Para ello, el robot realiza el seguimiento del operario (portador de un tag) mediante el sistema de localización UWB, siempre respetando un margen de seguridad entre el robot y el operario. El robot lee continuamente la posición del operario a través del módulo ROS desarrollado y trata de mantenerse próximo a la última posición leída. Así, cuando el operario se detiene para realizar cualquier operación, el robot también se detiene, pudiendo colocar en la plataforma del RB-1 la carga que se desea transportar. Una vez depositada la carga, el operario inicia de nuevo su trayectoria perseguido por el robot hacia un nuevo punto de destino donde la carga será depositada.

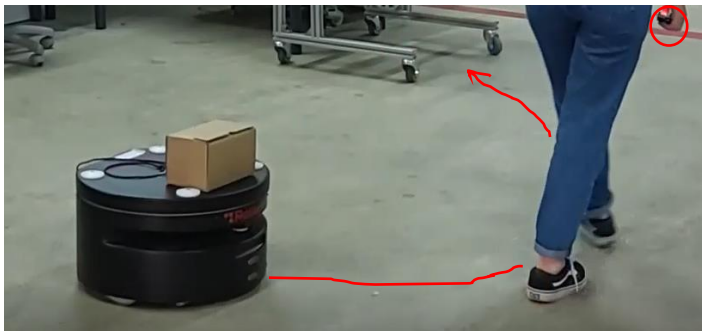


Ilustración 96 Seguimiento de operarios mediante el módulo ROS desarrollado para aplicaciones logísticas

El código desarrollado para esta aplicación se muestra en el apartado 2 del Anexo I. La estructura es similar a la seguida en el resto de las aplicaciones. En primer lugar, se establece el identificador del tag que lleva el robot y el identificador del tag que debe seguir. Se crea un hilo de conexión por MQTT para obtener los datos que recoge el sistema de localización de Pozyx, y se clasifican estos datos en función del identificador del tag que los emite. Así, se publica continuamente la posición del humano en la planificación de trayectorias del robot para que éste, conociendo su propia posición, se desplace en dirección al tag del humano respetando un margen de seguridad alrededor del mismo. En este caso, como el robot recibe la posición del humano a gran frecuencia, puede acortar la trayectoria a seguir si en vez de guardar todas las posiciones recibidas, le da prioridad a la última. Por

tanto, a pesar de la trayectoria seguida por el humano, el robot realizará el seguimiento con un periodo de muestreo superior al que se tiene para la lectura de posicionamiento con el sistema de Pozyx.

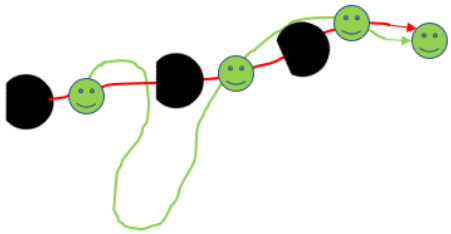


Ilustración 97 Esquema de seguimiento del operario con los últimos datos de posición leídos

De este modo, se logra el seguimiento del humano por el robot mediante un sensor virtual, manteniendo la autonomía del robot para la evitación de aquellos obstáculos que se puedan interponer entre el robot y el humano durante el seguimiento de la trayectoria, en este caso, con los sensores convencionales que tiene instalados el robot.

El vídeo de esta aplicación se puede visualizar en https://youtu.be/MPHHbw6phv8

6.6. APLICACIONES DE SEGURIDAD PARA ROBOTS INDUSTRIALES

El uso del sistema de localización de Pozyx para el ámbito de la seguridad se puede extrapolar también a robots industriales, no sólo con robots móviles. En entornos donde se trabaja con robots industriales pueden ocurrir accidentes causados por errores humanos, errores de control, accesos no autorizados, fallos mecánicos, fuentes ambientales, sistemas de energía e instalaciones inadecuadas. Sin embargo, el error más frecuente es el error humano. Los trabajadores se confían con los equipos y se adaptan a los peligros trabajando en áreas inseguras cuando programan o realizan el mantenimiento de un robot [23]. Las fases para evitar los accidentes en el espacio de trabajo son:

1. **Evaluación de los riesgos**. Entender los peligros que conlleva trabajar con robots industriales. Se han de identificar y analizar los peligros previsibles y las condiciones que los pueden desencadenar.
2. **Minimización de los riesgos**. Con esa información se pueden diseñar protocolos de actuación y establecer de controles de seguridad que reduzcan la probabilidad de exposición al peligro.
3. **Formación en la prevención de riesgos laborales**. Ofrecer a los trabajadores una formación técnica adecuada en materia de seguridad para el conocimiento del riesgo inherente a la actividad que se desarrolla en un entorno con robots industriales.

Una vez realizada la fase de evaluación de los riesgos, se implementan los controles necesarios para minimizar los riesgos detectados en la actividad. Algunos de estos controles pueden ser restringir el acceso a una zona de peligro instalando barreras fijas o dispositivos de detección de presencia como barreras optoelectrónicas, barreras láser, alfombras de presión, etc. Estos sistemas de protección pueden incluir alarmas sonoras, luces de advertencia, o señales diversas para concienciar a los operarios del peligro existente.

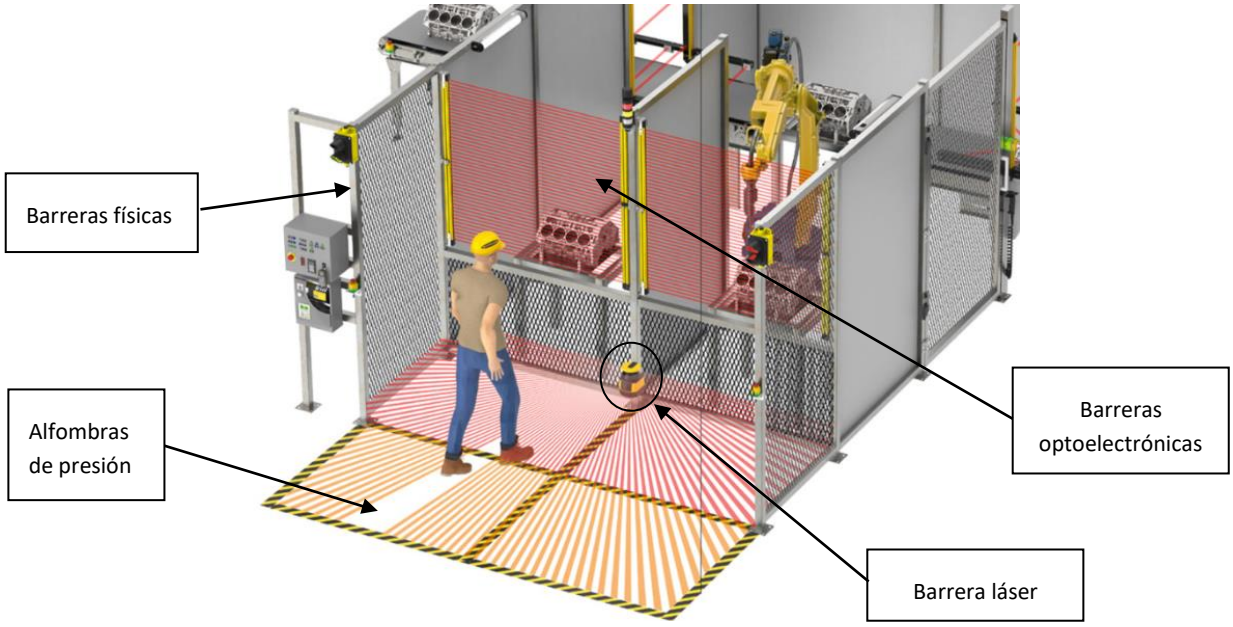


Ilustración 98 Sistemas de seguridad por detección de presencia para robots industriales [24]

En esta última aplicación se propone un nuevo sistema de seguridad para minimizar los riesgos que supone trabajar en un entorno donde conviven humanos y robots industriales, empleando nuevamente el sistema de localización de Pozyx. El objetivo es que cuando el robot industrial detecte la presencia de un operario cercano a su área de trabajo, disminuya su velocidad de operación para reducir los riesgos derivados de su actividad, y en caso de colisionar, aminorar los posibles daños causados. De este modo, se lee continuamente la posición del operario portador de un tag y se calcula la distancia que separa al operario del robot.

Para llevar a cabo el desarrollo de esta última aplicación se ha trabajado con un robot industrial ABB denominado YuMi disponible en el laboratorio. Este robot se programa en lenguaje RAPID mediante el programa de RobotStudio desarrollado por ABB.

Para establecer la conexión del robot con el sistema de localización de Pozyx se genera una conexión cliente-servidor mediante el protocolo de comunicación TCP/IP, donde el servidor en este caso se levanta desde el ordenador que conecta por protocolo MQTT con el sistema de localización y envía al YuMi (cliente) la posición del operario en todo momento. En el programa del robot se calcula la distancia que hay entre el operario y el robot como el módulo del vector que une las coordenadas de un tag fijo situado en el robot y el tag que lleva el operario. Al mismo tiempo, en el programa del robot se han implementado unas trayectorias aleatorias para ambos brazos del robot con una determinada velocidad, que disminuye progresivamente conforme se acorta la distancia calculada entre el operario y el robot evitando situaciones peligrosas en el entorno de trabajo.

De la misma forma, el robot recupera paulatinamente su velocidad nominal de operación a medida que el operario se aleja de su área de trabajo.

Esta aplicación es muy sencilla y puede resultar muy útil para la prevención de accidentes en espacios de producción, puesto que la responsabilidad no recae únicamente en las buenas prácticas de seguridad industrial del operario y en los deberes propios de su puesto de trabajo. Resulta una herramienta más para la detección de presencia en áreas con sistemas robotizados, y si se cuida la

fase de formación en la prevención de riesgos, puede contribuir a la reducción de accidentes en el entorno de trabajo.

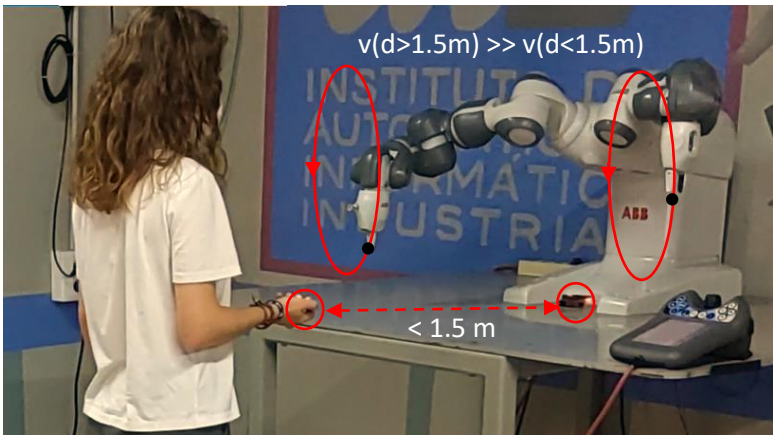


Ilustración 99 Robot industrial YuMi disminuyendo la velocidad de sus trayectorias por el acercamiento de un operario

El código desarrollado para esta aplicación se muestra en el apartado 4 del documento Anexo I, donde se crea el servidor que envía las posiciones calculadas por el sistema de localización al robot industrial.

El vídeo de esta aplicación se puede visualizar en https://youtu.be/2Z2J4aVy61s

CAPÍTULO 7. CONCLUSIONES Y LÍNEAS FUTURAS

En este trabajo se han analizado las tecnologías actuales de localización más comunes, realizando un breve análisis de sus principios de funcionamiento, su alcance y el entorno en el que presentan mayor precisión. Así, se ha podido concluir que la tecnología más adecuada para calcular el posicionamiento en espacios interiores de producción es la tecnología UWB. Esta tecnología ha sido instalada en el laboratorio de trabajo y se ha podido realizar un análisis experimental de sus prestaciones en un entorno real de producción.

Para facilitar el manejo de los datos que se obtienen de este sistema de localización implementado, se ha diseñado y programado una interfaz gráfica de usuario que permite establecer la conexión con el sistema y realizar el procesado de los datos de posicionamiento en tiempo real. Con esta interfaz es posible grabar los datos en ficheros y simularlos posteriormente. También permite la comunicación con robots que trabajen con la arquitectura de ROS, lo cual hace de esta interfaz una aplicación muy versátil y con numerosas posibilidades de mejora. En un futuro se podrían implementar funciones adicionales, como puede ser el disparo de alarmas visuales cuando un tag entre en una zona de trabajo delimitada, o modificar el número identificador de los tags que se visualicen en la gráfica para facilitar el reconocimiento.

Por último, se han desarrollado diversas aplicaciones para la mejora de la seguridad en entornos con sistemas robotizados y también se ha propuesto la optimización de trayectorias en robots móviles empleando el sistema descrito, produciendo resultados muy satisfactorios. La precisión de los datos obtenidos por el sistema de localización junto con el posprocesado de estos en la interfaz desarrollada y/o en los módulos descritos, hace de este sistema una herramienta muy eficaz y potente para mejorar operaciones habituales en el campo de la seguridad industrial y la robótica.

A pesar de que los robots ya cumplen con las medidas de seguridad exigibles y cuentan con sensores que facilitan su operación, este sistema de localización puede ser una utilidad añadida a su funcionamiento, según se ha visto en el capítulo seis de este documento. Especialmente en situaciones donde la seguridad recae totalmente sobre el usuario. Por otra parte, la optimización de trayectorias es uno de los campos clave en la robótica móvil, con este sistema se ha explorado una nueva posibilidad de cálculo “a ciegas” de trayectorias que podría servir en un futuro para implementar posibles mejoras en esta área.

CAPÍTULO 8. BIBLIOGRAFÍA

[1]. Silicon Labs <https://www.silabs.com/wireless/bluetooth/bluetooth-5-1>

[2]. Navixy <https://www.navixy.com/es/docs/academy/location-services/lbs-cell-id-y-wps/>

[3]. Redes Zone <https://www.redeszone.net/2017/03/11/wi-fi-location-funciona-sirve-este-estandar-geoposicionamiento-interiores-wi-fi/>

[4]. E. Soltanaghaei, Avinash Kalyanaraman, K. Whitehouse, (2017). Peripheral WiFi Vision: Exploiting Multipath Reflections for More Sensitive Human Sensing

[5]. Navegar.com <http://www.navegar.com/como-funciona-el-sistema-de-localizacion-por-gps/>

[6]. Azimut Marine <https://www.azimutmarine.es/sistema-posicionamiento-gps>

[7]. Lyudmila Mihaylova, Byung-Gyu Kim, Debi Prosad Dogra, (2016). UltraWideband Indoor Positioning Technologies: Analysis and Recent Advances. Sensors (Basel).

[8]. Sewio <https://www.sewio.net/uwb-technology/>

[9]. RamonMillan.com <https://www.ramonmillan.com/tutoriales/uwb.php>

[10]. Nanjing Woxu Wireless Co.,Ltd. <http://www.uwbaide.com/info/uwb-localization-techniques-tof-and-tdoa-34123155.html>

[11]. Lentin Joseph, Learning robotics using Python

[12]. ROS.org <Wiki.ros.org/ROS/Introduction>

[13]. Web de Robotnik <https://robotnik.eu/es/productos/robots-moviles/rb-1-base/>

[14]. Web de Pozyx: https://docs.pozyx.io/enterprise/latest

[15]. Diseño de un sistema de gestión de control de almacén <https://www.monografias.com>

[16]. iThome <https://ithelp.ithome.com.tw/articles/10222020>

[17]. ROS.org <http://docs.ros.org/en/noetic/api/sensor_msgs>

[18]. Decawave <https://www.decawave.com/technology1/>

[19]. Unigis-Girona <https://www.unigis.es/posicionamiento-indoor/>

[20]. Sensor láser <https://www.robotshop.com/us/es/telemetro-escaneo-laser-hokuyo-ust-10lx.html>

[21]. Sensor de ultrasonidos <https://www.antratek.com/hc-sr04-ultrasonic-sonar-distance-sensor>

[22]. Cámara RGBD <https://www.roscomponents.com/es/camaras/248-orbbec.html>

[23]. Centro de formación técnica para la industria <https://www.cursosaula21.com>

[24]. Web de Elion <https://www.elion.es/productos/seguridad-hombre-maquina/>

PRESUPUESTO

1. INTRODUCCIÓN

En este documento del trabajo se ha realizado una estimación económica del gasto total de realización de este proyecto. Para ello se han tenido en cuenta todos los recursos utilizados para su elaboración, siendo estos la mano de obra, la maquinaria a emplear, los materiales y las licencias de software necesarias.

2. CUADRO DE PRECIOS BÁSICOS

En este apartado se muestran los precios unitarios de los recursos considerados para la elaboración del presupuesto.

2.1. CUADRO DE LA MANO DE OBRA

Una estimación del coste de los recursos humanos por hora se muestra en la siguiente tabla.

CUADRO DE PRECIOS BÁSICOS: MANO DE OBRA			
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (euros/h)**
MO-ING	h	Ingeniero/a Industrial	20,00
MO-PRO	h	Ingeniero/a Senior	50,00
MO-TEC	h	Técnico/a de instalaciones	12,00

Tabla 5 Cuadro de precios básicos de la mano de obra

2.2. CUADRO DE LA MAQUINARIA

Para calcular el coste de la maquinaria por horas, es necesario estimar los años de amortización que tiene. En la siguiente tabla se muestra el precio estimado de adquisición de cada recurso así como su amortización. El cálculo final del precio de la maquinaria por horas se obtiene teniendo en cuenta que el año 2021 tiene un total de 10 días festivos a nivel nacional y 104 días en concepto de fines de semana, resultando 251 días laborales. Sabiendo que la jornada laboral es de 8 horas/día, el precio unitario se calcula como:

$$\text{Precio}\left(\frac{€}{\text{ud}}\right)\text{Amortización}^{-1}\left(\frac{\text{ud}}{\text{años}}\right)*\frac{1\text{ año}}{251\text{ días}}*\frac{1\text{ día}}{8\text{ horas}}=\text{Precio}\ \frac{€}{\text{hora}}$$

Los resultados se muestran en la siguiente tabla.

CUADRO DE PRECIOS BÁSICOS: MAQUINARIA					
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (euros/Ud)**	**Amortización (años/ud)**	**PRECIO (euros/h)**
MQ-UWB	UD	Sistema de localización por UWB de Pozyx	7700	7	0,55
MQ-HP	UD	Ordenador portátil HP	880	5	0,09
MQ-LAS	UD	Medidor láser	27	5	0,00

MQ-RB1	UD	Robot RB-1	20000	10	1,00
MQ-ABB	UD	Robot ABB	30000	10	1,49

Tabla 6 Cuadro de precios básicos de la maquinaria

2.3. CUADRO DE LICENCIAS DE SOFTWARE

En el caso de las licencias de software, el precio unitario se calcula de la misma forma que en el apartado anterior, teniendo en cuenta los días laborables y la jornada laboral. El precio de cada una se muestra en la siguiente tabla en tarifa anual.

CUADRO DE PRECIOS BÁSICOS: LICENCIAS DE SOFTWARE				
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (euros/Año)**	**PRECIO (euros/h)**
LI-MAT	h	Licencia Matlab	500,00	0,25
LI-OFF	h	Licencia Microsoft Office	69,00	0,03
LI-VMW	h	Licencia VMWare Workstation	167,60	0,08
LI-AUT	h	Licencia Autodesk	190,00	0,09
LI-RS	h	Licencia RobotStudio	200,00	0,10

Tabla 7 Cuadro de precios básicos de las licencias de software

2.4. CUADRO DE MATERIALES

Por último, se han tenido en cuenta aquellos materiales fungibles que han sido necesarios para la elaboración del proyecto. Entre ellos destacan los materiales de escritura y los útiles de trabajo empleados, pudiendo ser pequeñas herramientas. En la siguiente tabla se muestra su precio unitario aproximado.

CUADRO DE PRECIOS BÁSICOS: MATERIAL FUNGIBLE			
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (euros/Ud)**
MF-ESC	UD	Material de escritura	0,05
MF-AUX	UD	Herramientas auxiliares	0,07

Tabla 8 Cuadro de precios básicos de materiales

3. CUADRO DE PRECIOS DESCOMPUESTOS POR UNIDAD DE OBRA

En este apartado del presupuesto se exponen por capítulos las unidades de obra de las que consta el proyecto, así como la descomposición de los precios que contiene cada una de ellas. Se ha dividido el proyecto en tres capítulos: trabajo previo, desarrollo del proyecto y formalización de la memoria. En el primer capítulo se estima el precio unitario que supone la formación en las tecnologías a emplear durante el desarrollo, así como del tiempo dedicado a la instalación de los programas necesarios. En el segundo capítulo se realiza una descomposición de tareas para la estimación, en función del núcle

de trabajo. En el tercer capítulo se estima el precio unitario del tiempo dedicado a la elaboración del documento de la memoria. Para los cálculos se ha tenido en cuenta un valor del 2% en concepto de costes directos complementarios, como puede ser el gasto en electricidad para cargar el ordenador, la luz del habitáculo donde se trabaja, etc.

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (euros)	IMPORTE
CAPÍTULO 1. TRABAJO PREVIO					
UD1.0	h	Formación e instalación de programas		21,27	
	Formación en el lenguaje de programación Python (POO y estructuras multihilo), la librería PyQt5, la arquitectura ROS y el entorno de Linux. Instalación y configuración de VMWare Workstation con los paquetes de ROS y Python.				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MQ-HP	h	Ordenador portátil HP	1	0,09	0,09
LI-VMW	h	Licencia VMWare Workstation	1	0,08	0,08
LI-MAT	h	Licencia Matlab	0,5	0,25	0,12
LI-RS	h	Licencia RobotStudio	0,5	0,10	0,05
MQ-RB1	h	Robot RB-1	0,5	1,00	0,50
	%	Costes Directos Complementarios	0,02	21,52	0,43
			Coste Total		21,27

Tabla 9 Cuadro de precios descompuestos por unidad de obra del capítulo 1

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (euros)	IMPORTE
CAPÍTULO 2. DESARROLLO DEL PROYECTO					
UD2.0	h	Organización y seguimiento		71,40	
	Establecimiento de los objetivos del proyecto y el alcance del mismo. Valoración de los recursos disponibles para llevarlo a cabo. Seguimiento de la calidad de las pruebas y la progresión alcanzada.				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MO-PRO	h	Ingeniero/a Senior	1	50,00	50,00
	%	Costes Directos Complementarios	0,02	70,00	1,40
			Coste Total		71,40
UD2.1	h	Instalación del sistema de localización		29,19	
	Instalación y configuración del sistema de localización de Pozyx en el laboratorio de trabajo				
MO-TEC	h	Técnico/a de instalaciones	1	12,00	12,00
MO-ING	h	Ingeniero/a Industrial	0,8	20,00	16,00
MQ-UWB	h	Sistema de localización por UWB de Pozyx	1	0,55	0,55
MQ-HP	h	Ordenador portátil HP	0,8	0,09	0,07
	%	Costes Directos Complementarios	0,02	28,62	0,57
			Coste Total		29,19

UD2.2	h	Diseño y desarrollo de la interfaz gráfica en Python		21,13	
	Diseño y programación de la interfaz de usuario en Python con la librería de PyQt, implementando la conexión por protocolo MQTT con el sistema de localización de Pozyx, manejo de ficheros, opciones de configuración, etc.				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MQ-HP	h	Ordenador portátil HP	1	0,09	0,09
LI-VMW	h	Licencia VMWare Workstation	1	0,08	0,08
LI-RB1	h	Robot RB-1	0,3	1,00	0,30
LI-UWB	h	Sistema de localización por UWB de Pozyx	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	20,72	0,41
			Coste Total		21,13
UD2.3	h	Pruebas de medida de la precisión del sistema de localización		22,53	
	Realización de las pruebas de medida de la precisión del sistema de localización de Pozyx instalado. Implementación de filtros de datos mediante la interfaz gráfica y/o Matlab				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MQ-HP	h	Ordenador portátil HP	1	0,09	0,09
MQ-RB1	h	Robot RB-1	1	1,00	1,00
MQ-UWB	h	Sistema de localización por UWB de Pozyx	1	0,55	0,55
MQ-LAS	h	Medidor láser	1	0,00	0,00
MF-ESC	UD	Material de escritura	1	0,05	0,05
MF-AUX	UD	Herramientas auxiliares	1	0,07	0,07
LI-VMW	h	Licencia VMWare Workstation	1	0,08	0,08
LI-MAT	h	Licencia Matlab	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	22,09	0,44
			Coste Total		22,53
UD2.4	h	Desarrollo e integración de los datos en un módulo ROS		22,27	
	Procesado y adaptación de los datos obtenidos por el sistema de localización para integrarlos en los robots a tiempo real. Estudio y desarrollo de las diversas aplicaciones en el campo de la robótica móvil y robótica industrial para mejorar la seguridad en el entorno de trabajo y/o la optimización de procesos				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MQ-HP	h	Ordenador portátil HP	1	0,09	0,09
MQ-UWB	h	Sistema de localización por UWB de Pozyx	1	0,55	0,55
MQ-RB1	h	Robot RB-1	0,8	1,00	0,80
MQ-ABB	h	Robot ABB	0,2	1,49	0,30
LI-VMW	h	Licencia VMWare Workstation	1	0,08	0,08
LI-RS	h	Licencia RobotStudio	0,2	0,10	0,02
	%	Costes Directos Complementarios	0,02	21,83	0,44
			Coste Total		22,27

Tabla 10 Cuadro de precios descompuestos por unidad de obra del capítulo 2

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (euros)	IMPORTE
CAPÍTULO 3. FORMALIZACIÓN DE LA MEMORIA					
UD3.0	h	Redacción de la memoria		20,55	
	Redacción y maquetación del contenido del proyecto según las normas establecidas por la Universidad				
MO-ING	h	Ingeniero/a Industrial	1	20,00	20,00
MQ-HP	h	Ordenador portátil HP	1	0,09	0,09
LI-AUT	h	Licencia Autodesk	0,3	0,09	0,03
LI-OFF	h	Licencia Microsoft Office	1	0,03	0,03
	%	Costes Directos Complementarios	0,02	20,15	0,40
			Coste Total		20,55
UD3.1	h	Revisión de la memoria y evaluación		51,04	
	Evaluación del profesor acerca del trabajo realizado y corrección de la memoria				
MO-PRO	h	Ingeniero/a Senior	1	50,00	50,00
LI-OFF	h	Licencia Microsoft Office	1	0,03	0,03
	%	Costes Directos Complementarios	0,02	50,03	1,00
			Coste Total		51,04

Tabla 11 Cuadro de precios descompuestos por unidad de obra del capítulo 3

4. ESTADO DE LAS MEDICIONES DE TODAS LAS UNIDADES DEL PROYECTO

Una vez realizada la descomposición por unidades de obra y estimado su precio por hora, se realiza la medición de los tiempos que han sido necesarios para la realización de este proyecto. Estos se muestran en la siguiente tabla.

CÓDIGO	UNIDAD	DESCRIPCIÓN	MEDICIÓN	PRECIO (euros)	IMPORTE
CAPÍTULO 1. TRABAJO PREVIO					
UD1.0	h	Formación e instalación de programas	120	21,27	2552,85
			Coste Total		2552,85
CAPÍTULO 2. REALIZACIÓN DEL PROYECTO					
UD2.0	h	Organización y seguimiento	30	71,40	2142,00
UD2.1	h	Instalación del sistema de localización	10	29,19	291,90
UD2.2	h	Diseño y desarrollo de la interfaz gráfica en Python	200	21,13	4226,86
UD2.3	h	Pruebas de medida de la precisión del sistema de localización	80	22,53	1802,27
UD2.4	h	Desarrollo e integración de los datos en un módulo ROS	160	22,27	3563,38
			Coste Total		12026,42

CAPÍTULO 3. FORMALIZACIÓN DE LA MEMORIA					
UD3.0	h	Redacción de la memoria	30	20,55	616,60
UD3.1	h	Revisión de la memoria y evaluación	16	51,04	816,56
			Coste Total		1433,16

Tabla 12 Estado de mediciones de todas las unidades de obra del proyecto

5. PRESUPUESTO DE INVERSIÓN TOTAL

Para finalizar se calcula el presupuesto de ejecución material como la suma de los precios obtenidos en el apartado anterior, y se calcula el presupuesto de inversión total añadiendo los impuestos pertinentes.

PRESUPUESTO DE INVERSIÓN		
CÓDIGO	**DESCRIPCIÓN**	**IMPORTE (€)**
1	TRABAJO PREVIO	2552,85
2	REALIZACIÓN DEL PROYECTO	12026,42
3	FORMALIZACIÓN DE LA MEMORIA	1433,16
Presupuesto Ejecución Material (PEM)		16012,43
Gastos Generales (0,13 x PEM)		2081,62
Beneficio Industrial (0,06 x BI)		960,75
Subtotal (PEC)		19054,79
IVA (0,21 x PEC)		4001,51
Presupuesto de Inversión Total		**23056,30**

Tabla 13 Presupuesto de inversión total

El presupuesto asciende a:

VEINTITRÉS MIL CINCUENTA Y SEIS EUROS CON TREINTA CÉNTIMOS

ANEXO I.

CÓDIGO DESARROLLADO

1. CÓDIGO PYTHON INTERFAZ GRÁFICA

```
#!/usr/bin/env python
import sys
import os
import threading
from PyQt5.QtWidgets import QApplication, QMainWindow, QLineEdit, QLabel, QFormLayout, QMes
sageBox, QGroupBox, QGridLayout, QHBoxLayout, QWidget, QFileDialog
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5 import uic
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import matplotlib as mpl
import numpy as np
import paho.mqtt.client as mqtt
from datetime import datetime
import time
import ssl
import json
import pickle
import random
import math
from copy import deepcopy
try:
   import rospy
   from geometry_msgs.msg import Point
except:
   print('No estan habilitados los recursos de ROS')
pass

class VentanaEmergente(QWidget):
   def __init__(self):
       global xlimit, ylimit
       self.setWindowTitle('Editar ejes')
       layout = QGridLayout()
       self.labelejex = QLabel('Eje x:')
       self.labelejey = QLabel('Eje y:')
      self.editejex = QLineEdit()
      self.editejey = QLineEdit()
      layout.addWidget(self.labelejex,1,1)
      layout.addWidget(self.labelejey,2,1)
      layout.addWidget(self.editejex,1,2)
      layout.addWidget(self.editejey,2,2)
      self.setLayout(layout)
      self.valorx=xlimit
      self.valory=ylimit
      self.editejex.setText(str(self.valorx))
      self.editejey.setText(str(self.valory))
      self.editejex.returnPressed.connect(self.valor_ejex)
```

```
        self.editejey.returnPressed.connect(self.valor_ejey)

    def valor_ejex(self):
        global xlimit
        xlimit=int(self.editejex.text())
        _ventana.mapagrafica()
        _ventana.graphic.draw()

    def valor_ejey(self):
        global ylimit
        ylimit=int(self.editejey.text())
        _ventana.mapagrafica()
        _ventana.graphic.draw()

class Ventana(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        uic.loadUi("MainWindow.ui",self)
        self.setMinimumSize(1100,820)
        self.setMaximumSize(1100,820)
        self.setWindowTitle('Interfaz POZYX')

        #Llamadas a funciones
        self.podom.clicked.connect(self.subscriber)
        self.fmedia.clicked.connect(self.func_fmedia)
        self.fmediana.clicked.connect(self.func_fmediana)
        self.pmonitorizar.clicked.connect(self.monitorizar)
        self.line_IP.returnPressed.connect(self.lee_IP)
        self.line_port.returnPressed.connect(self.lee_port)
        self.line_tag_robot.returnPressed.connect(self.robot_tag)
        self.line_odom.returnPressed.connect(self.fcn_odom_topic)
        self.line_topic.returnPressed.connect(self.fcn_pub_topic)
        self.pstop.clicked.connect(self.parar)
        self.pgrabar.clicked.connect(self.grabar)
        self.pcargar.clicked.connect(self.cargar_datos)
        self.psimular.clicked.connect(self.simular)
        self.rosbutton.clicked.connect(self.rosactivate)
        self.mapa_intensidad.clicked.connect(self.mapa_rss)
        self.ppuntos.valueChanged.connect(self.valuechange)
        self.ptamano_ventana.valueChanged.connect(self.filt_ventana)
        self.oimportarconf.triggered.connect(lambda: self.import_config(file_name="None"))
        self.oexportarconf.triggered.connect(self.export_config)
        self.ocargarmapa.triggered.connect(self.cargar_mapa)
        self.oeditejes.triggered.connect(self.editar_ejes)
        self.olimpiar.triggered.connect(self.limpiar)
        self.a.clicked.connect(lambda: self.quitar_label(0))
        self.b.clicked.connect(lambda: self.quitar_label(1))
        self.c.clicked.connect(lambda: self.quitar_label(2))
        self.d.clicked.connect(lambda: self.quitar_label(3))
        self.e.clicked.connect(lambda: self.quitar_label(4))
```

```
        self.f.clicked.connect(lambda: self.quitar_label(5))
        self.g.clicked.connect(lambda: self.quitar_label(6))
        self.h.clicked.connect(lambda: self.quitar_label(7))
        self.i.clicked.connect(lambda: self.quitar_label(8))
        self.j.clicked.connect(lambda: self.quitar_label(9))
        self.k.clicked.connect(lambda: self.quitar_label(10))
        self.l.clicked.connect(lambda: self.quitar_label(11))
        self.m.clicked.connect(lambda: self.quitar_label(12))
        self.n.clicked.connect(lambda: self.quitar_label(13))
        self.o.clicked.connect(lambda: self.quitar_label(14))
        self.p.clicked.connect(lambda: self.quitar_label(15))
        self.q.clicked.connect(lambda: self.quitar_label(16))
        self.r.clicked.connect(lambda: self.quitar_label(17))
        self.s.clicked.connect(lambda: self.quitar_label(18))
        self.t.clicked.connect(lambda: self.quitar_label(19))
        self.vgrabar = False
        self.ros = False
        self.ros_is_new = True
        self.simulacion_on = False
        self.npuntos=self.ppuntos.value()
        self.tamano_ventana=self.ptamano_ventana.value()
        self.import_config(file_name='config.json')
        self.ventanaejes=VentanaEmergente()
        self.worker=None
        self.tag_robot="None"
        self.topic_odom=self.line_odom.text()
        self.topic_pub=self.line_topic.text()
        self.podom.setEnabled(False)
        self.rosbutton.setEnabled(False)
        self.ocultar_leyenda_rss()
        self.num_pos_leyenda=[]
        global c
        c=0

        #Cargar la gráfica
        self.graphic = Canvas(12, 8, 100)
        self.grafica.addWidget(self.graphic)
        self.graphic.setMouseTracking(True)
        self.graphic.mouseMoveEvent=self.mouseMoveEvent
        self.mapagrafica()

    def mouseMoveEvent(self, event):
        if event.pos().x()>53 and event.pos().x()<1011 and event.pos().y()>24 and event.pos().y()<474:
            x=int(((event.pos().x()-53)/(1011-53))*xlimit)
            y=int(ylimit-((event.pos().y()-24)/(474-24))*ylimit)
        self.label_coord.setText(str(x)+' , '+str(y))
    else:
        self.label_coord.clear()

    def filt_ventana(self):
```

```
        self.tamano_ventana=self.ptamano_ventana.value()

    def func_fmedia(self):
        if self.fmediana.isChecked():
            self.fmediana.setChecked(False)

    def func_fmediana(self):
        if self.fmedia.isChecked():
            self.fmedia.setChecked(False)

    def quitar_label(self,num_pos):
        if not num_pos in self.num_pos_leyenda:
            self.num_pos_leyenda.append(num_pos)
        else:
            self.num_pos_leyenda.pop(self.num_pos_leyenda.index(num_pos))

    def ocultar_leyenda_rss(self):
        self.colormap_rss.hide()
        self.value_rss1.hide()
        self.value_rss2.hide()

    def mapa_rss(self):
        color_rss=[]
        lista_medias=[]
        if self.worker!=None:
            self.parar()
            self.limpiar()
            dir=os.getcwd()
        file, _ = QFileDialog.getOpenFileName(self, 'Buscar Archivo', dir, "Text Files (*.txt);;Pickle Files (*.
        pickle)")

        if file.endswith('.txt'):
            datos_leidos = np.loadtxt(file,skiprows=1,dtype='float')
            self.datos_x=datos_leidos[:,0]
            self.datos_y=datos_leidos[:,1]
            minrss=9999
            maxrss=-9999
            for i in range(len(self.datos_x)):
                datos_rss=datos_leidos[i,2:]
                for j in range(len(datos_rss)):
                    if datos_rss[j]==-999:
                        datos_rss=np.delete(datos_rss,j)

                media_rss=sum(datos_rss)/len(datos_rss)
                if media_rss>maxrss:
                    maxrss=media_rss
                if media_rss<minrss:
                    minrss=media_rss
            for i in range(len(self.datos_x)):
                datos_rss=datos_leidos[i,2:]
```

```
            media_rss=sum(datos_rss)/len(datos_rss)
            c=1-(media_rss-minrss)/(maxrss-minrss)
            color_rss.append(c)
            lista_medias.append(media_rss)

        greyscale_map = plt.get_cmap('hsv')
        self.graphic.axes.scatter(self.datos_x,self.datos_y,s=250,marker='s',c=color_rss,cmap=greyscale_
        map,alpha=0.2,edgecolor='none')
        self.graphic.draw()

        #Leyenda
        self.colormap_rss.show()
        self.value_rss1.show()
        self.value_rss2.show()
        self.value_rss1.setText(str((math.trunc(maxrss*pow(10.0,4)))/pow(10.0,4))+'dB')
        self.value_rss2.setText(str((math.trunc(minrss*pow(10.0,4)))/pow(10.0,4))+'dB')


    def editar_ejes(self):
        if self.ventanaejes.isVisible():
            self.ventanaejes.hide()
        else:
            self.ventanaejes.show()

    def cargar_mapa(self):
        dir=os.getcwd()
        image, _ = QFileDialog.getOpenFileName(self, 'Buscar Archivo', dir, "Image Files (*.png *.jpg)")
        img=image.split("/")
        self.mapa=img[-1]
        self.limpiar()
        self.mapagrafica()
        self.graphic.draw()

    def mapagrafica(self):
        global xlimit, ylimit
        mapaimg = plt.imread(self.mapa)
        self.graphic.axes.imshow(mapaimg, extent=[1,xlimit,0,ylimit])
        self.graphic.axes.set_xlim([0, xlimit])
        self.graphic.axes.set_ylim([0, ylimit])

    def callback(self,data):
        if len(self.datax_odom)>self.npuntos:
            self.datax_odom.pop(0)
            self.datay_odom.pop(0)
        self.datax_odom.append(data.pose.pose.position.x*1000)
        self.datay_odom.append(data.pose.pose.position.y*1000)

    def subscriber(self):
        try:
            import rospy
```

```
            from geometry_msgs.msg import PoseWithCovarianceStamped
            self.datax_odom=[]
            self.datay_odom=[]
            if self.topic_odom!=None:
                rospy.Subscriber(self.topic_odom, PoseWithCovarianceStamped, self.callback)
            else:
                print("Es necesario especificar el topic")
        except:
            print("No se pueden importar las librerias, o no se ha cargado el mapa en el robot")

    def rosactivate(self):
        if not self.ros:
            try:
                if self.ros_is_new:
                    self.ros_is_new=False
                    self.roscore_init=threading.Thread(target=self.roscore_def)
                    self.roscore_init.start()
                rospy.init_node('talker',anonymous=True)
                self.ros=True
            except:
                print('No esta instalada la libreria de ROS, o no se puede realizar la conexion')
        else:
            self.ros=False

    def roscore_def(self):
        import subprocess
        import rosgraph
        self.rosmaster_ip=self.line_rosmaster.text()
        self.ros_ip=self.line_rosip.text()

        if rosgraph.is_master_online():
            os.putenv('ROS_MASTER_URI',"http://"+self.rosmaster_ip+":11311")
            os.putenv('ROS_IP',self.ros_ip)
        else:
            os.putenv('ROS_MASTER_URI',"http://"+self.rosmaster_ip+":11311")
            os.putenv('ROS_IP',self.ros_ip)
            self.roscore = subprocess.Popen(['roscore'])
        time.sleep(1)

    def fcn_odom_topic(self):
        self.topic_odom=self.line_odom.text()

    def fcn_pub_topic(self):
        self.topic_pub=self.line_topic.text()

    def robot_tag(self):
        self.tag_robot=self.line_tag_robot.text()

    def lee_IP(self):
        self.IP_data=self.line_IP.text()
```

```
def lee_port(self):
    port=self.line_port.text()
    self.port_data=int(port)

def monitorizar(self):
   if self.worker==None:
      self.worker = Worker(self.IP_data,self.port_data)
      self.worker.start()
  else:
      self.num_pos_leyenda.clear()
      self.limpiar()
      self.worker.init_variables(self.IP_data,self.port_data)
      self.worker.run()
  self.pmonitorizar.setEnabled(False)
  self.pcargar.setEnabled(False)
  self.psimular.setEnabled(False)
  self.mapa_intensidad.setEnabled(False)
  self.podom.setEnabled(True)
  self.rosbutton.setEnabled(True)

def parar(self):
  self.pmonitorizar.setEnabled(True)
  self.pcargar.setEnabled(True)
  self.psimular.setEnabled(True)
  self.mapa_intensidad.setEnabled(True)
  self.podom.setEnabled(False)
  self.rosbutton.setEnabled(False)
  self.worker.stop()
  if self.vgrabar==True:
     self.grabar()

def grabar(self):
   if self.vgrabar == False and not self.pmonitorizar.isEnabled():
      self.vgrabar = True
      self.pgrabar.setIcon(QIcon("parar.png"))
      self.prss.setEnabled(False)

  elif self.vgrabar == False and self.pmonitorizar.isEnabled():
      self.monitorizar()
      self.vgrabar = True
      self.pgrabar.setIcon(QIcon("parar.png"))
      self.prss.setEnabled(False)

  else:
      self.vgrabar = False
      self.pgrabar.setIcon(QIcon("grabar.png"))
      self.prss.setEnabled(True)
      data={}
      dt=datetime.now()
```

```
        day = str(dt.day)
        month = str(dt.month)

        fileName, _ = QFileDialog.getSaveFileName(self,"Guardar archivo",os.getcwd(),"JSON (*.json)")
        if fileName == "":
            print("Cancelar seleccion")

        #Escribir en PIKLE
        for tag in self.worker.xgrab:
            pickled_file = open('tag'+tag+'_'+day+'-'+month+'.pickle','wb')
            data[tag]={'x':self.worker.xgrab[tag],'y':self.worker.ygrab[tag],'rss':self.worker.rssgrab[tag]}
            pickle.dump(data[tag], pickled_file)

        #Escribir en TXT
        for tag in self.worker.xgrab:
            with open('tag'+tag+'_'+day+'-'+month+'.txt','w') as file_txt:
                datos_x = [str(i) for i in self.worker.xgrab[tag]]
                datos_y = [str(i) for i in self.worker.ygrab[tag]]
                if self.prss.isChecked():
                    file_txt.write('coord_x'+' '+'coord_y'+' '+str(self.worker.anchors)+'\n')
                    for i in range(len(datos_x)):
                        datos_rss=self.worker.rssgrab[tag][i].values()
                        if len(datos_rss)!=len(self.worker.anchors):
                            dif=len(self.worker.anchors)-len(datos_rss)
                            for j in range(dif):
                                self.worker.rssgrab[tag][i].append(-150)
                        datos_rss=str(self.worker.rssgrab[tag][i].values()).lstrip('dict_values([')
                        rss1=datos_rss.rstrip('])')
                        rss2=rss1.replace(',',' ')
                        file_txt.write(datos_x[i]+' '+datos_y[i]+' '+rss2+'\n')
                else:
                    file_txt.write('coord_x'+' '+'coord_y'+'\n')
                    for i in range(len(datos_x)):
                        file_txt.write(datos_x[i]+' '+datos_y[i]+'\n')

                self.worker.xgrab[tag]=[]
                self.worker.ygrab[tag]=[]
                self.worker.rssgrab[tag]=[]

    def cargar_datos(self):
        global c
        self.ocultar_leyenda_rss()
        if self.worker!=None:
            self.parar()

        dir=os.getcwd()
        file, _ = QFileDialog.getOpenFileName(self, 'Buscar Archivo', dir, "Text Files (*.txt);;Pickle Files (*.
        pickle)")

        if file.endswith('.txt'):
```

```
            datos_leidos = np.loadtxt(file,skiprows=1,dtype='float', usecols=[0,1])
            self.datos_x=datos_leidos[:,0]
            self.datos_y=datos_leidos[:,1]

        else:
            with open(file,'rb') as pickled_file:
                data = pickle.load(pickled_file)
                self.datos_x=data['x']
                self.datos_y=data['y']

        color=["c", "m", "y", "b", "g", "r", "k"]
        self.graphic.axes.plot(self.datos_x, self.datos_y,color[c]+'o-',ms=1)
        self.graphic.draw()
        if c<len(color)-1:
            c=c+1
        else:
            c=0

    def simular(self):
        if self.simulacion_on==False:
            self.simulacion_on=True
            self.limpiar()
            self.simular=threading.Thread(target=self.simular_grafica)
            self.simular.start()
        else:
            self.simulacion_on=False

    def simular_grafica(self):
        try:
            for i in range(1,len(self.datos_x)):
                if self.simulacion_on:
                    x=self.datos_x[0:i]
                    y=self.datos_y[0:i]
                    self.graphic.axes.plot(x, y, 'bo-',ms=3)
                    self.graphic.draw()
                    time.sleep(0.005)
                else:
                    break
        except:
            print('Se ha de cargar un fichero de datos')
            pass

    def import_config(self,file_name):
        global xlimit, ylimit

        if file_name=="None":
            dir=os.getcwd()
            file, _ = QFileDialog.getOpenFileName(self, 'Buscar Archivo', dir, "JSON files (*.json)")
            file=file.split("/")
            file_name=file.pop(-1)
```

```
        try:
            with open(file_name,'r') as file:
                config=json.load(file)
            self.IP_data=config['IP_DATA']
            self.line_IP.setText(self.IP_data)
            self.port_data=config['PORT_DATA']
            self.line_port.setText(str(self.port_data))
            self.mapa=config['MAP']
            xlimit=config['XLIMIT']
            ylimit=config['YLIMIT']
            npuntos=config['NPOINTS']
            self.ppuntos.setValue(npuntos)

        except:
            print('El archivo de configuracion no tiene los parametros suficientes o no se ha cargado')

    def export_config(self):
        global xlimit, ylimit
        fileName, _ = QFileDialog.getSaveFileName(self,"Guardar archivo",os.getcwd(),"JSON (*.json)")

        if fileName == "":
            print("Cancelar seleccion")
            pass

        data={}
        data['IP_DATA']=self.IP_data
        data['PORT_DATA']=self.port_data
        data['MAP']=self.mapa
        data['XLIMIT']=xlimit
        data['YLIMIT']=ylimit
        data['NPOINTS']=self.ppuntos.value()
        with open(fileName+'.json','w') as outfile:
            json.dump(data,outfile,indent=4)

    def limpiar(self):
        self.graphic.axes.clear()
        self.mapagrafica()
        self.graphic.draw()
        if self.worker!=None:
            self.worker.clear_legend()

    def valuechange(self):
        self.npuntos=self.ppuntos.value()

class Worker(threading.Thread):
    def __init__(self,host,port):
        threading.Thread.__init__(self)
        self.lock = threading.Lock()
        self.colors = ["b", "g", "r", "c", "m", "y", "k"]
```

```
        self.leyenda=[_ventana.a,_ventana.b,_ventana.c,_ventana.d,_ventana.e,_ventana.f,_ventana.g
        ,_ventana.h,_ventana.i,_ventana.j,_ventana.k,_ventana.l,_ventana.m,_ventana.n,_ventana.o,_
        ventana.p,_ventana.q,_ventana.r,_ventana.s,_ventana.t]
        self.leycolor=[_ventana.pi1,_ventana.pi2,_ventana.pi3,_ventana.pi4,_ventana.pi5,_ventana.pi
        6,_ventana.pi7,_ventana.pi8,_ventana.pi9,_ventana.pi10,_ventana.pi11,_ventana.pi12,_ventana
        na.pi13,_ventana.pi14,_ventana.pi15,_ventana.pi16,_ventana.pi17,_ventana.pi18,_ventana.pi
        19,_ventana.pi20]
        self.init_variables(host,port)

    def init_variables(self,host,port):
        self.host = host
        self.port = port
        self.xdata={}
        self.ydata={}
        self.xdata_filt={}
        self.ydata_filt={}
        self.anchors=[]
        self.rss={}
        self.xgrab={}
        self.ygrab={}
        self.c={}
        self.xplot={}
        self.yplot={}
        self.pubs ={}
        self.index=-1
        self.rssgrab={}
        self.datos_grab=[]
        self.tags=[]

    def run(self):
        _ventana.ocultar_leyenda_rss()
        self.client = mqtt.Client()
        self.client.on_connect = self.on_connect
        self.client.connect(self.host, port=self.port)
        self.client.on_message = self.on_message
        self.client.on_subscribe = self.on_subscribe
        self.client.subscribe('tags')
        self.client.loop_start()
        self.monitorizar=True
        self.plot=threading.Thread(target=self.update_plot)
        self.plot.start()

    def stop(self):
        self.client.loop_stop()
        self.monitorizar=False
        pixmap=QPixmap('led-blue-black.png')
        _ventana.led.setPixmap(pixmap)
        if self.lock.locked():
            self.lock.release()
```

```
def on_connect(self, client, userdata, flags, rc):
    print(mqtt.connack_string(rc))
    pixmap=QPixmap('ledverdee.png')
    _ventana.led.setPixmap(pixmap)

def on_message(self, client, userdata, msg):
    try:
        msg_utf8 = msg.payload.decode()[1:-1]
        dic = json.loads(msg_utf8)
        self.TagID=dic["tagId"]
        x=dic["data"]["coordinates"]["x"]
        y=dic["data"]["coordinates"]["y"]
        numanchors=len(dic["data"]["anchorData"])
        pos=[]
        num=0

        # ROS
        if _ventana.ros:
        try:
            self.message = Point()
            if self.TagID not in self.pubs.keys():
                self.pubs[self.TagID]=rospy.Publisher(_ventana.topic_pub+self.TagID,Point,queue_size=10)

            # Genero el mensaje
            self.message.x=x
            self.message.y=y
            self.pubs[self.TagID].publish(self.message)
            print(self.message)
        except:
            print('No esta instalada la libreria de ROS')
            _ventana.rosbutton.setChecked(False)

        # GUARDAR VECTORES DE DATOS
        self.lock.acquire()
        if self.TagID not in self.xdata.keys():
            # Leyenda ------------------------------------------------------
            self.tags.append(self.TagID)
            self.index=self.index+1
            self.leyenda[self.index].setText(self.TagID)
            self.c[self.TagID] = self.colors[len(self.xdata.keys()) % len(self.colors)]
            pixmap=QPixmap(self.c[self.TagID])
            self.leycolor[self.index].setPixmap(pixmap)
            # --------------------------------------------------------------
            self.xdata[self.TagID]=[]
            self.ydata[self.TagID]=[]
            self.xdata_filt[self.TagID]=[]
            self.ydata_filt[self.TagID]=[]

        if _ventana.fmedia.isChecked():
            if len(self.xdata_filt[self.TagID])>=_ventana.tamano_ventana:
```

```
            self.xdata_filt[self.TagID].pop(0)
            self.ydata_filt[self.TagID].pop(0)
        self.xdata_filt[self.TagID].append(x)
        self.ydata_filt[self.TagID].append(y)
        x=np.mean(self.xdata_filt[self.TagID])
        y=np.mean(self.ydata_filt[self.TagID])

    elif _ventana.fmediana.isChecked():
        if len(self.xdata_filt[self.TagID])>=_ventana.tamano_ventana:
            self.xdata_filt[self.TagID].pop(0)
            self.ydata_filt[self.TagID].pop(0)
        self.xdata_filt[self.TagID].append(x)
        self.ydata_filt[self.TagID].append(y)
        x=np.median(self.xdata_filt[self.TagID])
        y=np.median(self.ydata_filt[self.TagID])

    self.xdata[self.TagID].append(x)
    self.ydata[self.TagID].append(y)
    self.lock.release()

    # GRABAR DATOS-------------------------------------------------------------
    if _ventana.vgrabar:
        if self.TagID not in self.xgrab.keys():
            self.xgrab[self.TagID]=[]
            self.ygrab[self.TagID]=[]
            self.rssgrab[self.TagID]=[]
            for i in range(numanchors):
                if dic["data"]["anchorData"][i]["anchorId"] not in self.anchors:
                    self.anchors.append(dic["data"]["anchorData"][i]["anchorId"])

        self.xgrab[self.TagID].append(x)
        self.ygrab[self.TagID].append(y)

        if _ventana.prss.isChecked():
            for i in range(numanchors):
                pos.append(self.anchors.index(dic["data"]["anchorData"][i]["anchorId"]))

            for i in pos:
                self.rss[self.anchors[i]]= dic["data"]["anchorData"][num]["rss"]
                num=num+1

            rss_copy=self.rss.copy()
            self.rssgrab[self.TagID].append(rss_copy)
            #-----------------------------------------------------------------------
    except:
        if self.lock.locked():
            self.lock.release()
        pass

def update_plot(self):
```

```
try:
   while self.monitorizar:
      _ventana.graphic.axes.clear()
      _ventana.mapagrafica()
      self.lock.acquire()
      xdata = deepcopy(self.xdata)
      ydata = deepcopy(self.ydata)
      if self.lock.locked():
         self.lock.release()

      for tagId in xdata.keys():
         if not self.tags.index(tagId) in _ventana.num_pos_leyenda:
            if tagId not in self.xplot.keys():
               self.xplot[tagId] = []
               self.yplot[tagId] = []
            elif len(self.xplot[tagId]) < _ventana.npuntos:
               if len(self.xplot[tagId]) > 0:
                  if self.xplot[tagId][-1] != xdata[tagId][-1]:
                     self.xplot[tagId].append(xdata[tagId][-1])
                     self.yplot[tagId].append(ydata[tagId][-1])
               else:
                  self.xplot[tagId].append(xdata[tagId][-1])
                  self.yplot[tagId].append(ydata[tagId][-1])
            elif len(self.xplot[tagId]) > _ventana.npuntos:
               self.xplot[tagId].pop(0)
               self.yplot[tagId].pop(0)
               self.xplot[tagId].pop(0)
               self.yplot[tagId].pop(0)
               self.xplot[tagId].append(xdata[tagId][-1])
               self.yplot[tagId].append(ydata[tagId][-1])
            else:
               if self.xplot[tagId][-1] != xdata[tagId][-1]:
                  if len(self.xplot[tagId]) > 0:
                     self.xplot[tagId].pop(0)
                     self.yplot[tagId].pop(0)
                     self.xplot[tagId].append(xdata[tagId][-1])
                     self.yplot[tagId].append(ydata[tagId][-1])
         else:
            self.xplot[tagId].clear()
            self.yplot[tagId].clear()

         if tagId != _ventana.tag_robot:
            _ventana.graphic.axes.plot(self.xplot[tagId], self.yplot[tagId], self.c[tagId]+'o-', ms=3)
         else:
            _ventana.graphic.axes.plot(self.xplot[tagId], self.yplot[tagId], self.c[tagId]+'o-', ms=3)
            _ventana.graphic.axes.scatter(self.xplot[tagId], self.yplot[tagId], marker='o',s=5000,color
            ='y',alpha=0.2)
         if _ventana.podom.isChecked():
            _ventana.graphic.axes.plot(_ventana.datax_odom , _ventana.datay_odom , 'ko-', ms=3)
```

```
            _ventana.graphic.draw()
        except:
            pass

    def on_subscribe(self, client, userdata, mid, granted_qos):
        print("Subscribed to topic!")

    def clear_legend(self):
        for label in self.leyenda:
            label.setText('')
        for color in self.leycolor:
            color.clear()

class Canvas(FigureCanvas):
    def __init__(self, width, height, dpi):
        fig = plt.figure(figsize=(width,height))
        self.axes = fig.add_axes([0.05,0.05,0.90,0.95])
        FigureCanvas.__init__(self,fig)

if __name__=='__main__':
    app=QApplication(sys.argv)
    _ventana=Ventana()
    _ventana.show()
    sys.exit(app.exec_())
```

2. CÓDIGO PYTHON SEGUIMIENTO DE TAGS

```
#!/usr/bin/env python
from pozyx_location.msg import custom_msg
import paho.mqtt.client as mqtt
import ssl
import json
import rospy
import os
import actionlib
from time import sleep
from move_base_msgs.msg import MoveBaseAction
from threading import Thread, Lock
from math import sin, cos, pi
import numpy as np

host = "158.42.126.70"
port = 1883
pubs={}
id = "27191"
target = "27252"
x = 0
y = 0
```

```
lock = Lock()

def on_connect(client, userdata, flags, rc):
    print(mqtt.connack_string(rc))

def on_message(client, userdata, msg):
    global id
    global client_move
    global target
    global x
    global y
    global lock
    try:
        msg_utf8 = msg.payload.decode()[1:-1]
        dic = json.loads(msg_utf8)
        TagID=dic["tagId"]
        if TagID not in pubs.keys():
            if TagID == id:
                pubs[TagID]=rospy.Publisher('tag_pose_robot',custom_msg,queue_size=10)
            else:
                pubs[TagID]=rospy.Publisher('tag_pose_'+TagID,custom_msg,queue_size=10)

#Enviando nuevo goal a robot
        if TagID == target:
            x = float(dic["data"]["coordinates"]["x"]) / 1000
            y = float(dic["data"]["coordinates"]["y"]) / 1000

        message=custom_msg()
        message.TagID.data=dic["tagId"]
        message.position.x=dic["data"]["coordinates"]["x"]
        message.position.y=dic["data"]["coordinates"]["y"]
        message.position.z=dic["data"]["coordinates"]["z"]
        pubs[TagID].publish(message)

        except:
            pass

def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed to topic!")

def mqtt_thread():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_subscribe = on_subscribe
    client.connect(host, port=port)
    client.subscribe('tags')
    client.loop_forever()

def move():
```

```
    global x
    global y
    global lock
    xant = 0
    yant = 0
    tolx = 0.3
    toly = 0.3
    client_move = actionlib.SimpleActionClient('/robot/move_base', MoveBaseAction)
    client_move.wait_for_server()
    cont = 0
    while True:
        lock.acquire()
        xlocal = x
        ylocal = y
        lock.release()
        diffx = abs(xlocal - xant)
        diffy = abs(ylocal - yant)
        xant = xlocal
        yant = ylocal

        if diffx > tolx or diffy > toly:
            pose = MoveBaseAction()
            pose.action_goal.goal.target_pose.header.frame_id = "robot_map"
            pose.action_goal.goal.target_pose.pose.position.x = xlocal + 0.7
            pose.action_goal.goal.target_pose.pose.position.y = ylocal - 0.4
            pose.action_goal.goal.target_pose.pose.orientation.x = 0.0
            pose.action_goal.goal.target_pose.pose.orientation.y = 0.0
            pose.action_goal.goal.target_pose.pose.orientation.z = 0.0
            pose.action_goal.goal.target_pose.pose.orientation.w = 1.0
            client_move.send_goal(pose.action_goal.goal)
        sleep(1)
        cont+=1

if __name__ == "__main__":
    pathConfig ='config.txt'
    if os.path.isfile(pathConfig):
        with open(pathConfig, "r") as configFile:
            id = configFile.readline().strip().rstrip()
            host = configFile.readline().strip().rstrip()
            port = int(configFile.readline().strip().rstrip())
    else:
        with open(pathConfig, "w") as configFile:
            configFile.write(id+"\n")
            configFile.write(host+"\n")
            configFile.write(str(port)+"\n")

    rospy.init_node('talker',anonymous=True)
    threads = []
    threads.append(Thread(target=mqtt_thread))
    threads[0].start()
```

```
    threads.append(Thread(target=move))
    threads[1].start()
```

3. MÓDULO ROS

```
#!/usr/bin/env python
import paho.mqtt.client as mqtt
from math import pi, cos, sin, acos, atan, sqrt, degrees, radians
from numpy import sign
import ssl
import json
import os
import rospy
from geometry_msgs.msg import Pose, PoseWithCovarianceStamped
from sensor_msgs.msg import LaserScan
import threading
from pypozyx import (get_first_pozyx_serial_port, PozyxSerial, Coordinates, Acceleration, AngularVel
ocity, EulerAngles, LinearAcceleration, Magnetic, MaxLinearAcceleration, Pressure, Quaternion, Tem
perature)
from time import sleep
import numpy as np
from tf.transformations import quaternion_from_euler, euler_from_quaternion
import signal
import sys

#---HILO MQTT-----------------------------------------------------------------------------------------------------------
def on_connect(client, userdata, flags, rc):
    print(mqtt.connack_string(rc))

def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed to topic!")

def on_message(client, userdata, msg):
    global TagID, x, y
    try:
        # Parsear datos
        msg_utf8 = msg.payload.decode()[1:-1]
        dic = json.loads(msg_utf8)
        lock.acquire()
        TagID=dic["tagId"]
        x=dic["data"]["coordinates"]["x"]
        y=dic["data"]["coordinates"]["y"]
        lock.release()
        # Nuevo tag
        if TagID not in pubs.keys():
            if TagID == robot_id:
                if COM=="mix":
```

```
                pass
            else:
                pubs[TagID]=rospy.Publisher('tag_pose_robot',Pose,queue_size=10)
        else:
            distances[TagID]=[]
            pubs[TagID]=rospy.Publisher('tag_pose_'+TagID,Pose,queue_size=10)
        # Filtrar datos
        if TagID not in xdata_filt.keys():
            xdata_filt[TagID]=[]
            ydata_filt[TagID]=[]
        if filt_type!="None":
            if len(xdata_filt[TagID])>=tamano_ventana:
                xdata_filt[TagID].pop(0)
                ydata_filt[TagID].pop(0)
            xdata_filt[TagID].append(x)
            ydata_filt[TagID].append(y)
            if filt_type=="media_movil":
                x=np.mean(xdata_filt[TagID])
                y=np.mean(ydata_filt[TagID])
            elif filt_type=="mediana":
                x=np.median(xdata_filt[TagID])
                y=np.median(ydata_filt[TagID])
        if COM=="mix" and TagID == robot_id:
            pass
        else:
            pose=Pose()
            pose.position.x=x
            pose.position.y=y
            pubs[TagID].publish(pose)
            laser_reader()

    except:
        print("pass")
        if lock.locked():
            lock.release()
            pass

def mqtt_thread():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_subscribe = on_subscribe
    client.connect(host, port=port)
    client.subscribe('tags')
    client.loop_start()
    while end != True:
        sleep(1)
```

```
    client.loop_stop()

#-------------------------------------------------------------------------------------------------------------------
#---PUBLICADOR LASER-----------------------------------------------------------------------------------------------
def adjust_angle(angle,dx,dy):
    if dx >= 0 and dy >= 0:
        return angle
    elif dx < 0 and dy >= 0:
        return 180 - angle
    elif dx < 0 and dy < 0:
        return angle + 180
    elif dx >= 0 and dy < 0:
        return 360 - angle

def adjust_beta(beta):
    if beta < 0:
        beta = 360 + beta
    if beta > 180:
        beta = beta - 360
    return beta

def laser_reader():
    global scan
    if TagID != robot_id:
        scan.ranges = np.ones(360) * 15
        lock.acquire()
        dx = (x/1000-xrobot)
        dy = (y/1000-yrobot)
        d = sqrt(dx*dx+dy*dy)
        phi = degrees(atan(0.6/d)) # angulo del obstaculo con 1 m de diametro
        alpha = adjust_angle((degrees(atan(abs(dy/dx)))),dx,dy) #alpha esta entre 0-
        360 y es el angulo del tag en coordenadas absolutas
        beta = alpha - degrees(theta)
        lock.release()

        beta=adjust_beta(beta)
        distances[TagID]=[d,int(beta)+180,int(phi)]

        for tag in distances.keys():
            for i in range((distances[tag][1]-distances[tag][2]),(distances[tag][1]+distances[tag][2])):
                if i>=360:
                    i=i-360
                if distances[tag][0]<scan.ranges[i]:
                    scan.ranges[i]=distances[tag][0]

    scan.header.stamp=rospy.Time.now()
    pub.publish(scan)
```

```
#-------------------------------------------------------------------------------------------------------------------
#---LECTURA DE ODOMETRIA DEL ROBOT-----------------------------------------------------------------------------
def callback_amcl(data):
    global theta, xrobot, yrobot
    lock.acquire()
    theta=euler_from_quaternion([data.pose.pose.orientation.x,data.pose.pose.orientation.y,data.pos
    e.pose.orientation.z,data.pose.pose.orientation.w])[2]
    if theta<0:
        theta=2*pi+theta
    xrobot=data.pose.pose.position.x
    yrobot=data.pose.pose.position.y
    lock.release()

def amcl_reader():
    rospy.Subscriber('/robot/amcl_pose',PoseWithCovarianceStamped, callback_amcl)
    while end != True:
        sleep(1)

#-------------------------------------------------------------------------------------------------------------------
def getFirstAngle():
    magnetic_uT = Magnetic()
    pozyx.getMagnetic_uT(magnetic_uT)
    norm = sqrt(magnetic_uT.x*magnetic_uT.x + magnetic_uT.y*magnetic_uT.y)
    angle = degrees(acos((magnetic_uT.y/norm)))
    if magnetic_uT.x < 0:
        angle = 360 -angle
    angle = 360 - angle
    return angle

def getEulerAngles():
    euler_angles_deg = EulerAngles()
    pozyx.getEulerAngles_deg(euler_angles_deg)
    euler_angles_deg.heading=360-euler_angles_deg.heading
    return euler_angles_deg.heading

def serial_tag():
    global pozyx
    serial_port = get_first_pozyx_serial_port()
    pub_robot=rospy.Publisher('/tag_pose_robot',Pose,queue_size=10)
    pub_init=rospy.Publisher('/robot/initialpose',PoseWithCovarianceStamped,queue_size=1)
    xdata_filt=[]
    ydata_filt=[]

    if serial_port is not None:
        pozyx = PozyxSerial(serial_port)
        print("Connection success!")
        pozyx.setHeight(1000)
```

```
init_angle = getFirstAngle()
init_pose=True

while not rospy.is_shutdown():
  coordinates = Coordinates()
  pozyx.getCoordinates(coordinates)
  x=coordinates.x
  y=coordinates.y
  theta=(getEulerAngles()+init_angle+320) % 360
  if theta>180:
    theta=theta-360
  quat=quaternion_from_euler(0,0,radians(theta))

  if init_pose==True:
    init_pos=PoseWithCovarianceStamped()
    init_pos.pose.pose.position.x=x/1000
    init_pos.pose.pose.position.y=y/1000
    init_pos.pose.pose.orientation.x=quat[0]
     init_pos.pose.pose.orientation.y=quat[1]
    init_pos.pose.pose.orientation.z=quat[2]
    init_pos.pose.pose.orientation.w=quat[3]
    pub_init.publish(init_pos)
    init_pose=False

   if filt_type!="None":
     if len(xdata_filt)>=tamano_ventana:
       xdata_filt.pop(0)
       ydata_filt.pop(0)
     xdata_filt.append(x)
     ydata_filt.append(y)

     if filt_type=="media_movil":
       x=np.mean(xdata_filt)
       y=np.mean(ydata_filt)

     elif filt_type=="mediana":
       x=np.median(xdata_filt)
       y=np.median(ydata_filt)

  pose=Pose()
  pose.position.x=x/1000
  pose.position.y=y/1000
  pose.orientation.x=quat[0]
  pose.orientation.y=quat[1]
  pose.orientation.z=quat[2]
  pose.orientation.w=quat[3]
  pub_robot.publish(pose)
```

```
            sleep(0.1)
    else:
        print("No Pozyx port was found")

def signal_handler(sig, frame):
    global end
    end = True
    sleep(2)
    sys.exit(0)

if __name__ == "__main__":
    pubs={}
    xdata_filt={}
    ydata_filt={}
    distances={}
    end=False

    #Inicio el nodo publicador
    rospy.init_node('tag_laser_avoidance',anonymous=True)

    #Creo el publicador
    pub = rospy.Publisher('/robot/my_laser', LaserScan, queue_size=50)

    #Creo el mensaje del laser
    scan = LaserScan()
    scan.header.frame_id = "robot_base_link"
    scan.angle_min = -3.14
    scan.angle_max = 3.14
    scan.angle_increment = (3.14/180)
    scan.range_min = 0
    scan.range_max = 20
    scan.ranges = []
    scan.intensities = []

    #Cargar los datos
    file_name='robot_config.json'
    if os.path.isfile(file_name):
        with open(file_name,'r') as file:
            config=json.load(file)
        host=config['IP_DATA']
        port=config['PORT_DATA']
        robot_id=config['Tag_ID_Robot']
        COM=config['COM'] #MQTT, serial, mix
        filt_type=config['Filtro'] #None, media_movil, mediana
        tag_type=config['Tag_type'] #0: wearable tag, 1:minitag, 2:developer tag
        tamano_ventana=config['ventana']
    else:
```

```
    data={}
    data['IP_DATA']="158.42.126.70"
    data['PORT_DATA']=1883
    data['Tag_ID_Robot']="27191"
    data['COM']="MQTT"
    data['Filtro']="media_movil"
    data['Tag_type']=2
    data['ventana']=30
    with open(file_name,'w') as configFile:
        json.dump(data,configFile,indent=4)

#Creo el Mutex
lock=threading.Lock()

#Lanzo los hilos en funcion del modo de funcionamiento
if COM=="MQTT":
    threads = []
    threads.append(threading.Thread(target=mqtt_thread))
    threads[0].start()
    threads.append(threading.Thread(target=amcl_reader))
    threads[1].start()
elif COM=="mix":
    threads = []
    threads.append(threading.Thread(target=mqtt_thread))
    threads[0].start()
    threads.append(threading.Thread(target=serial_tag))
    threads[1].start()
    threads.append(threading.Thread(target=amcl_reader))
    threads[2].start()
else:
    threads = []
    threads.append(threading.Thread(target=serial_tag))
    threads[0].start()
    threads.append(threading.Thread(target=amcl_reader))
    threads[1].start()

#Defino una senal para matar los hilos cuando se pulse Ctrl+C
signal.signal(signal.SIGINT, signal_handler)
signal.pause()
```

4. MÓDULO ABB

```
#!/usr/bin/env python
import socket
import time
import os
```

```
import sys
import paho.mqtt.client as mqtt
import ssl
import signal
import json

def on_connect(client, userdata, flags, rc):
    print(mqtt.connack_string(rc))

def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed to topic!")

def on_message(client, userdata, msg):
    global connection
    try:
        msg_utf8 = msg.payload.decode()[1:-1]
        dic = json.loads(msg_utf8)
        TagID=dic["tagId"]
        x=dic["data"]["coordinates"]["x"] #X
        y=dic["data"]["coordinates"]["y"] #Y
        z=dic["data"]["coordinates"]["z"]
        connection.sendall((TagID+":"+str(x)+":"+str(y)+":"+str(z)+";").encode("utf-8"))

    except:
        pass

def mqtt_thread():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_subscribe = on_subscribe
    client.connect(pozyxhost, port=pozyxport)
    client.subscribe(topic)
    client.loop_start()
    while end != True:
        time.sleep(1)
    client.loop_stop()

def signal_handler(sig, frame):
    global end
    end = True
    server.close()
    time.sleep(2)
    sys.exit(0)

def server():
    global server, success, connection
```

```
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
    try:
        server.bind((localhost, localport))
        server.listen()
        connection, address = server.accept()

    except:
        print('Puerto no disponible')
        sys.exit(0)

if __name__=='__main__':
    end=False
    pozyxhost = "158.42.126.70"
    localhost = "192.168.174.128"
    pozyxport = 1883
    localport = 7654
    topic = "tags"
    server()
    mqtt_thread()

    #Defino una señal para matar los hilos cuando se pulse Ctrl+C
    signal.signal(signal.SIGINT, signal_handler)
    signal.pause()
```