Desarrollo de un sistema de localización en interior para robots móviles basado en UWB

Trabajo Fin de Máster

Máster Universitario en Informática Industrial y Robótica

Autor
Darío Cabarcos Novo

Tutores
Francisco Javier Bellas Bouza
Félix Orjales Saavedra

14 de septiembre de 2021

Resumen

El objetivo concreto de este TFM será el ajuste de un sistema de localización basado en Ultra Wideband (UWB) para ser utilizado en el robot móvil Robobo, en un entorno real de conducción autónoma, pero con el objetivo global de que el sistema quede preparado para poder ser equipado en otros robots móviles que se muevan por interior.

Este TFM se enmarca en la línea de investigación de conducción autónoma que se lleva a cabo en el Grupo Integrado de Ingeniería (GII) de la UDC. En esta linea, se utiliza el robot educativo Robobo para realizar estudios a escala de diversos aspectos de la conducción autónoma, como la navegación basada en imagen o la coordinación de varios vehículos. Para ello, el GII dispone de una maqueta de una ciudad de 4x4 metros, en la que se llevan a cabo los experimentos. De cara a poder replicar el funcionamiento de los vehículos reales, es necesario disponer de un sistema de localización en interior equivalente al GPS, de modo que Robobo pueda estar localizado en todo momento sobre el mapa del entorno, y pueda realizar así una planificación de trayectorias adecuada.

El GII dispone de un sistema de localización comercial en interior basado en UWB. Este sistema se puede configurar con diferentes modos, números variables de puntos de referencia (o anclas) y se deben procesar los resultados para obtener una medida de posición fiable y precisa.

El ajuste de este sistema deberá realizarse con el objetivo de que el robot se pueda localizar con precisión en un mapa interno del entorno de conducción autónoma.

Las distintas fases de desarrollo serán:

1. Estudio de configuración óptima del sistema UWB para el espacio de trabajo.
2. Filtrado y tratamiento de los datos para mejora de precisión y fiabilidad.
3. Pruebas de precisión estáticas y dinámicas del sistema propuesto.
4. Integración del sistema en el Robobo para su uso posterior.

Índice

Índice de figuras

1. Introducción

Entre los temas más recurrentes a la hora de mencionar avances tecnológicos, cabe destacar la conducción autónoma. Ésta consta de varios factores, como pueden ser la capacidad de interpretar el entorno y realizar acciones en consecuencia o, uno de los más importante, la localización; es decir, la capacidad de conocer la posición actual respecto a un sistema global, ya que de esto depende el poder ir de un sitio a otro, proceso conocido como navegación.

En la actualidad, la conducción autónoma va muy ligada a su implementación en vehículos de carretera, como pueden ser automóviles o camiones. En estos casos, su localización se realiza gracias a los sistemas GPS, que permiten ubicar al vehículo en un mapa y de esta manera trazar las rutas pertinentes.

Los sistemas GPS son capaces de obtener las coordenadas del vehículo por triangulación utilizando satélites que orbitan al planeta Tierra. Esta señal de los satélites se recibe correctamente en exteriores, viéndose claramente atenuada cuando se intenta recibir dentro de un edificio.

Por otra parte, en la actualidad se están llevando a cabo grandes avances en robótica, más concretamente en sistemas robóticos móviles y autónomos. Actualmente, lo más común en la industria es ver robots que se desplazan por la planta transportando elementos sin nadie que los opere, pero utilizando sistemas que los guían, como pueden ser líneas en el suelo, lo cual condiciona posibles modificaciones a sus trayectos en el futuro. Por lo tanto, un gran paso hacia la flexibilidad y mejora de sus capacidades sería que por ellos mismos se localizasen dentro de la planta y obtuvieran sus rutas de forma similar a los vehículos autónomos de exterior.

Claramente, existe un paralelismo entre la conducción autónoma de los robots móviles de interior y la de los vehículos, salvo que, como se mencionó anteriormente, el sistema GPS es deficiente o inasumible en interiores, lo cual lleva a buscar una alternativa.

En este trabajo se implementa un sistema comercial llamado Pozyx, que basa su funcionamiento en la tecnología *Ultra Wide Band* o banda ultra ancha, en castellano. Este sistema guarda muchos similitudes con los sistemas GPS, ya que consiste en un dispositivo colocado sobre el robot que calcula su posición triangulando con otro tipo de dispositivos llamados “anclas”, que se distribuyen de manera adecuada por la planta.

En resumen, este trabajo buscará implementar y optimizar el sistema Pozyx en robots móviles, ajustando sus diferentes configuraciones e incorporando un procesamiento adecuado de los datos. Además, se buscará la forma más adecuada para que el sistema montado sobre el robot funcione de manera inalámbrica y comunique la localización calculada a un ordenador sobre el que se pueda implementar el algoritmo de navegación.

2. Objetivos

1. Implementar un sistema de localización en interiores basado en Pozyx.
2. Procesar de manera adecuada la localización obtenida.
3. Comunicar el resultado inalámbricamente a un ordenador.
4. Crear un programa capaz de interpretar y visualizar los datos.

3. Antecedentes

Para poner en contexto el uso de la tecnología UWB y de Pozyx como sistema de localización en interiores, se van a citar tres artículos que aportan puntos muy interesantes acerca del tema.

3.1. Análisis de la tecnología UWB

El primero de los artículos que se van a mencionar en esta sección realiza un análisis sobre la tecnología UWB para el posicionamiento en interiores [1].

Comienza mencionando que la necesidad de una tecnología de posicionamiento para interiores ha aumentado durante los últimos años, ya que es vital para múltiples aplicaciones, entre ellas la industria. En comparación con los entornos de exterior, los entornos de interior necesitan una mayor precisión, además de que el reto es mayor por culpa de los reflejos y dispersión de la señal que provocan los obstáculos. La tecnología UWB presentaría un mejor rendimiento en estos aspectos que otras tecnologías destinadas al posicionamiento en interiores.

Este artículo realiza un análisis SWOT (*Strengths Weaknesses Opportunities Threats*), que es una herramienta de análisis muy potente para evaluar y entender una tecnología.

Como fortalezas se destacan el hecho de que es una tecnología de licencia libre debido a que no se considera equipamiento de radio, ya que la potencia de la señal es baja. Además, al consumir poco comparado con otras tecnologías, lo convierte en una opción más eficiente. Por otra parte, el utilizar un gran ancho de banda la hace resistente a los problemas derivados de que la señal pueda tomar múltiples caminos o las interferencias.

Respecto a las debilidades, se destaca el hecho de que puedan existir sistema cercanos a donde se implementa el UWB que produzcan interferencias, ya que estos también podrían usar frecuencias en un espectro similar. Además, al viajar los pulsos a mucha velocidad, el recptor de UWB debe calcular la posición a un ritmo muy alto, lo que podría presentar también una debilidad.

En el apartado de oportunidades, se menciona que UWB se está convirtiendo en la elección para sistemas que requieren gran precisión, como el guiado de robots dentro de edificios, ya que los reflejos de los objetos no afectan en esta tecnología.

Para terminar, las amenazas mencionadas son que los equipos que se comercializan con esta tecnología pueden ser caros y que la implementación de las antenas en el entorno puede llegar a ser complicada y delicada.

3.2. Uso de UWB para localización de AGV

El siguiente artículo introduce un sistema de posicionamiento basado en UWB para vehículos guiados automáticamente (AGV) [2].

Debido a que las tecnologías basadas en inteligencia artificial avanzan rápidamente y que las tareas que desempeñan los AGV son complejas y sofisticadas, conocer las coordenadas es súmamente importante y, además, se requiere una precisión alta.

La solución más popular es usar líneas en el suelo, pero limita al AGV a la hora de realizar tareas complejas. Una solución basada en láser puede ser cara y no apta para grandes áreas y otras soluciones como los ultrasonidos o las señales de radio pueden tener una precisón reducida. Por lo tanto, la mejor idea podría ser implementar un sistema basado en UWB.

Las pruebas realizadas con el sistema implementado en el artículo determinan que la alta precisión y la alta tasa de refresco de las medidas son claras ventajas frente a otros sistemas, además de que sería más barato implementar UWB que, por ejemplo, una solución basada en un sistema óptico.

3.3. Uso de Pozyx en entorno industrial

Como último antecedente se va a mencionar un proyecto desarrollado en la Universidad de A Coruña basado en el sistema Pozyx [3], al igual que el que se presenta en este documento. El proyecto consiste en utilizar Pozyx para la localización y seguimiento de una carretilla elevadora en un entorno industrial de interior.

La necesidad de localización y seguimiento en entornos industriales está aumentando hoy en día. Además, un sistema de seguimiento puede utilizarse en la optimización, trazabilidad y seguridad de los procesos para incrementar la productividad en una fábrica. De entre todas las posibles opciones, UWB

emerge como la mejor opción para obtener gran preción en interiores. Sin embargo, las interferencias que puedan causar los trabajadores o la maquinaria, son un reto para este tipo de sistemas. El hecho de implementar Pozyx sobre una carretilla elevadora [Fig. 1], constituye una manera de evaluar las capacidades de este sistema en un entorno industrial real.

Para realizar las pruebas, se ha modelado un entorno en el simulador Gazebo, considerando múltiples sensores para realizar el seguimiento de la carretilla. Además, se simula la tecnología UWB basándose en una serie de mediciones realizadas en un entorno real.

Las conclusiones dictaminan que la precisión de las medidas utilizando únicamente UWB es aceptable cuando el tag tiene visión directa de las anclas. En caso contrario, la precisión puede caer drásticamente. Por lo tanto, se debería considerar el combinar varios sensores para reducir considerablemente el error.

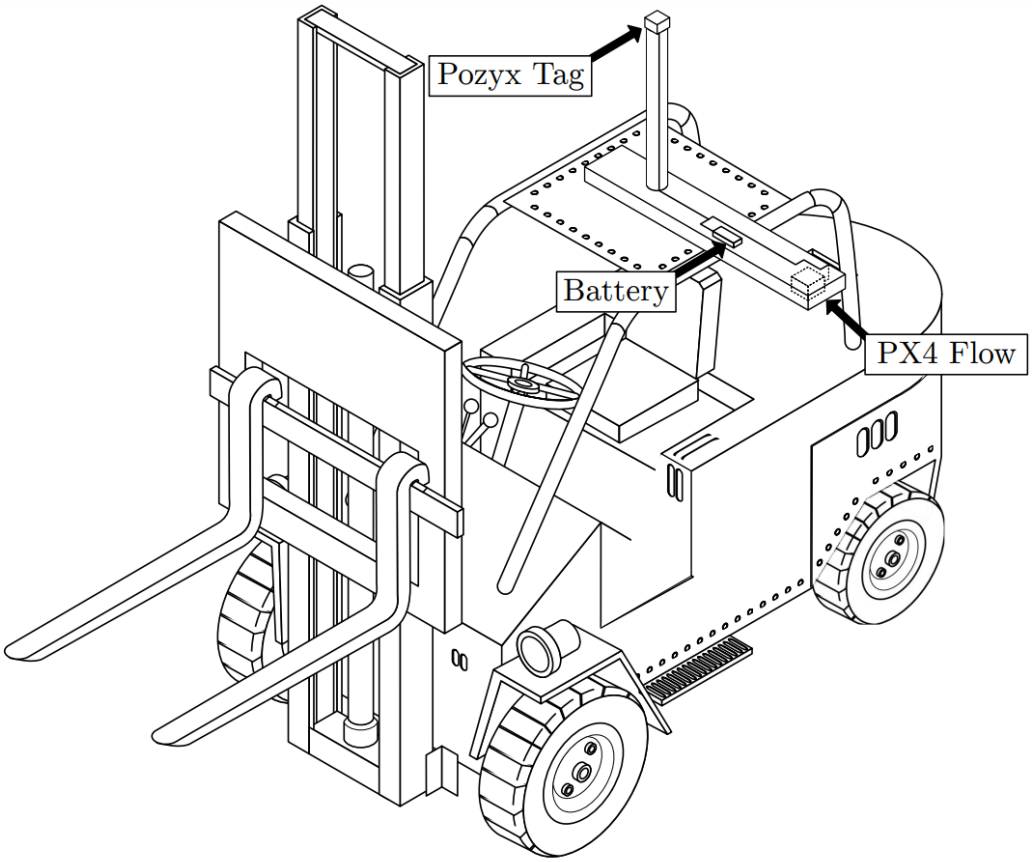

Figura 1: Esquema de la carretilla elevadora.

4. Tecnologías utilizadas

En este apartado se van a enunciar y describir la diferentes tecnologías incorpoadas al sistema de localización, desde cómo se obtiene el posicionamiento hasta la comunicación con el ordenador que controla al robot.

4.1. Pozyx

Para una distancia determinada a un punto de referencia, que a partir de ahora se denominará "ancla", se sabe que existen un número infinito de puntos posibles que se encuentran a esa distancia del ancla, formando una circunferencia de radio igual a dicha distancia.

Teniendo esto en cuenta, se puede obtener la posición de un punto respecto a tres anclas, ya que las tres circunferencias que se forman alrededor de cada ancla con la distancia al punto, se cortarán en dicho punto.

La dificultad de esta aproximación radica en que las medidas no siempre son perfectas y que las circunferencias no tienen por qué cortarse en un único punto.

Por lo tanto, este modo de obtener la posición de un objeto respecto a unas anclas se conoce como triangulación, y es el utilizado por el sistema Pozyx [4], un producto comercial capaz de proporcionar un posicionamiento preciso, del orden de centímetros, gracias al uso de *Ultra Wide Band*.

Esta tecnología [5] usa ondas de radio que se envían de un punto a otro, midiendo el tiempo que tarda la señal en ir y volver, lo que se conoce como tiempo de vuelo. Para obtener la distancia sería tan sencillo cómo dividir el tiempo de vuelo entre la velocidad de las ondas de radio. En el caso concreto del Pozyx, el ancho de banda es de 500 MHz.

Un ancho de banda tan alto permite al receptor distinguir incluso los reflejos en la señal, lo que posibilita obtener la posición incluso en lugares con muchos elementos reflectantes, como son los interiores.

En total, Pozyx cuenta con dos tipos de dispositivos [6]:

- **Tag** [Fig. 2a]. Es el elemento que se posiciona dentro de un sistema Pozyx. Para obtener su posición necesita anclas en rango. Se moverá encima del robot, por lo que es necesario alimentarla con una pequeña batería.
- **Anclas** [Fig .2b]. Estos elementos nunca se mueven y actúan como referencias para el algoritmo de posicionamiento de los tags.

(a) Tag.

(b) Ancla.

Figura 2: Dispositivos de Pozyx.

4.2. Arduino UNO

Para poder obtener los resultados de la localización por parte del tag, es necesario colocarlo sobre un Arduino UNO [Fig. 3], ya que el tag en sí tiene formato de *shield*.

Arduino UNO es un placa basada en el microcontrolador ATmega328P [7]. Entre sus carácterísticas destacan los 14 pines digitales y 6 analógicos, junto con una conexión USB que sirve tanto para alimentar la placa, como para conectarse a un ordenador para programarla.

Esta placa es súmamente popular y es "hardware libre", lo que significa que puede ser utilizada y modificada por cualquiera con total libertad. Todo esto, junto con su bajo coste, la convierte en la opción ideal para el desarrollo de pequeños proyectos.

El lenguaje de programación más común que se utiliza para implementar el código a ejecutar por la placa es C++. Gracias a la gran comunidad que existe alrededor de Arduino, es posible encontrar un gran número de librerías y ejemplos para poder implementar una gran cantidad de sensores y dispositivos de manera relativamente sencilla.

Por lo tanto, recurriendo a la documentación y a la librería que Pozyx pone a disposición para utilizar el tag conectado a un Arduino UNO, se podrá desarrollar un programa que lea el resultado del posicionamiento y lo procese y administre según las necesidades.

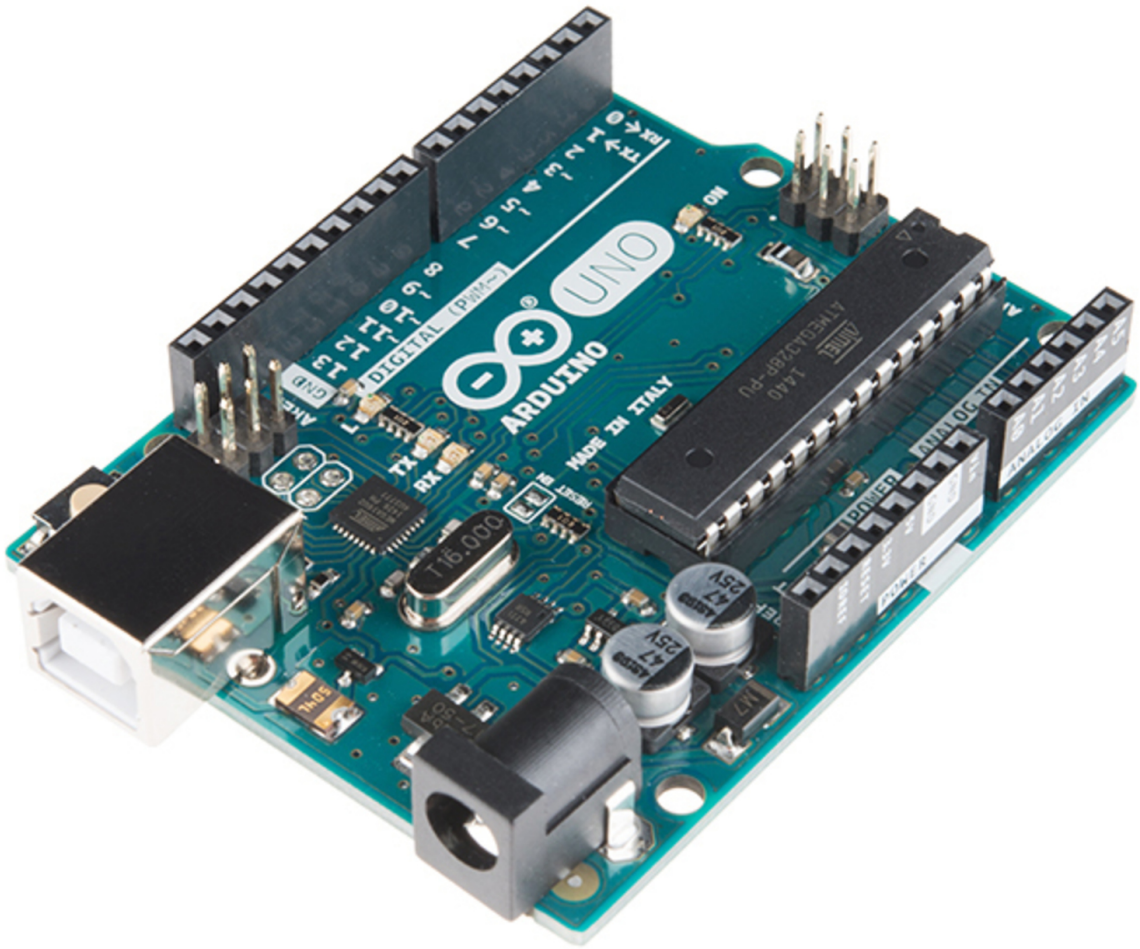

Figura 3: Arduino UNO.

4.3. ESP8266

Uno de los objetivos es que la posición del robot se comunique de manera inalámbrica con el ordenador. Para ello, es necesario incoporar un elemento más al conjunto de tag y Arduino UNO que permita conectarse a una red WiFi y enviar los datos.

La opción escogida es un pequeño módulo que incorpora un microprocesador ESP8266 [Fig. 4]. Este procesador de 32 bit es compacto y con un consumo bajo, lo que permite integrarlo directamente en los pines disponibles en Arduino UNO.

El lenguaje de programación también es C++ y gracias a la librería propia del microprocesador, es sencillo tanto conectarse a una red WiFi como comunicarse con Arduino UNO.

En conclusión, este módulo se utilizará como pasarela inalámbrica para los datos de la posición entre el Arduino UNO y un ordenador y sobre él será necesario implementar un protocolo de comunicación adecuado.

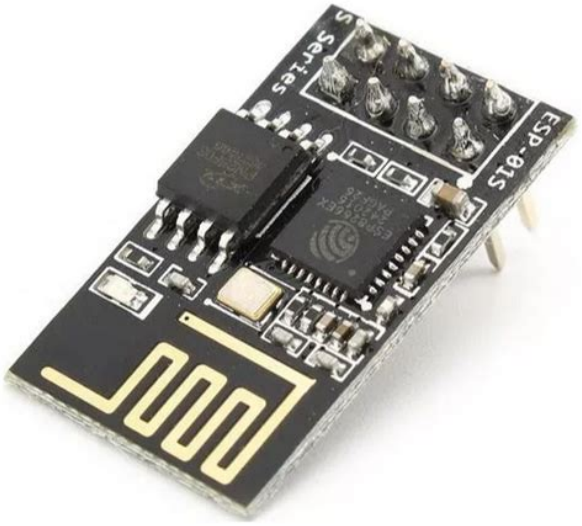

Figura 4: ESP8266.

4.4. WebSocket

Una vez que ya se han seleccionado todos los elementos que formarán parte del dispositivo que se monte sobre el robot, hay que implementar la manera en la que estos elementos se comunican a través de la red WiFi con un ordenador; es decir, escoger protocolo.

Para este proyecto se ha tomado la decisión de utilizar WebSocket. Este protocolo cuenta con un librería que permite implentarlo en el módulo del microprocesador ESP8266 de manera sencilla. También existe una librería de Python, que es el lenguaje de programación que se va a utilizar en el ordenador que reciba los datos. Todo esto, sumado a que es eficiente y ligero, lo convierten en una opción ideal.

WebSocket es un protocolo que proporciona canales de comunicación full-dúplex sobre una sola conexión TCP [8]. Además, WebSocket permite flujos de mensajes sobre TCP, por lo que TCP sólo gestiona flujos de bytes sin necesidad de establecer mensajes por separado. Esto quiere decir que, subscribiéndose a un canal, la comunicación ocurrirá de manera continua. Aunque puede parecer en parte similiar a una conexión HTTP, se encuentra una gran diferencia precisamente en el aspecto de la comunicación continua, ya que una petición HTTP necesita esperar por una respuesta, al contrario que un WebSocket.

En cuanto a la arquitectura, básicamente existe un servidor que comparte el servicio y un conjunto de clientes que se conectan a él. La conexión abre un canal entre los dispositivos pero, para obtener información, los clientes deben subscribirse a un topic.

En este proyecto, el microprocesador ESP8266 llevará implementada la parte de servidor y proporcionará el resultado del posicionamiento a través de un topic. El ordenar ejecutará un código de

Python que implementa un cliente que se conecta con el servidor y se subscribe al topic correspondiente y, de esta manera, poder obtener el dato de la localización del robot siempre que sea necesario.

Esta manera de implementar el protocolo WebSocket permite, en el caso de querer escalar el proyecto a un mayor número de robots, que un mismo equipo se conecte y obtenga la posición de varios robots fácilmente o que un robot proporcione su posición a varios ordenadores de manera simultánea. Un ejemplo sería una fábrica en la que se disponga de varios ordenadores para ejecutar los algoritmos de control de diferentes robots y otro ordenador que se encargue de la visualización y supervisión de la posición de los mismos sobre el mapa de la planta.

5. Desarrollo

En este apartado se van a describir las distintas fases de desarrollo que se han llevado a cabo para poner en funcionamiento todo el sistema de localización [Fig. 5], utilizando las tecnologías anteriormente planteadas, así como las considereciones que se han tenido en cuenta a la hora de implementar el proyecto.

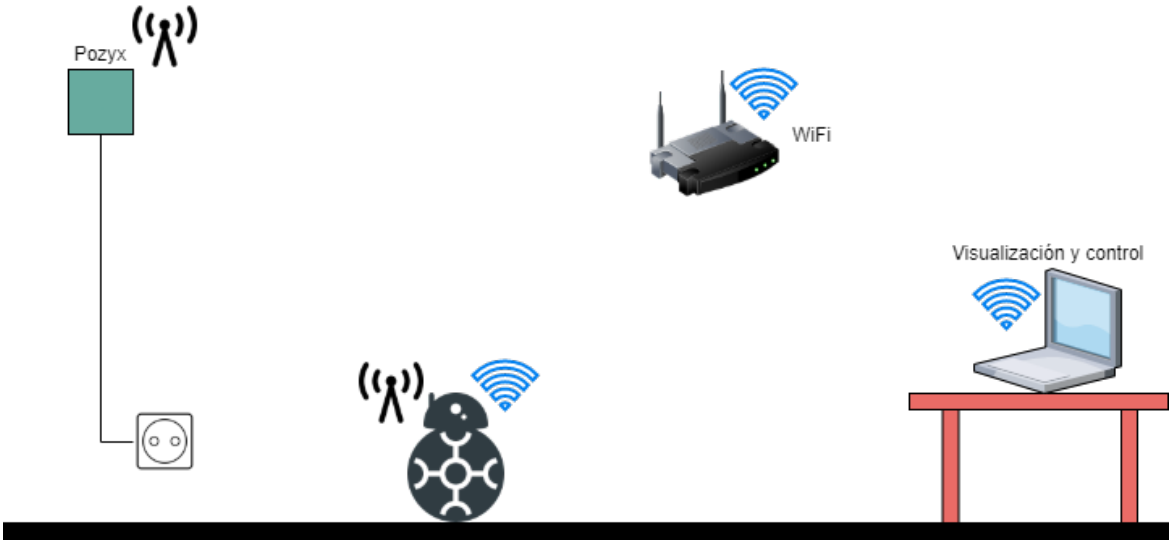

Figura 5: Arquitectura del proyecto.

5.1. Colocación de las anclas

El sistema de posicionamiento Pozyx requiere de por lo menos cuatro anclas colocadas alrededor del área donde se va a realizar la localización. La precisión del sistema depende mucho de la calidad de la instalación, por lo que conviene seguir una serie de reglas [6].

1. Colocar las anclas altas y a la vista del tag. El hecho de colocarlas altas incrementa las posibilidades de recibir la señal con buena calidad y menos obstrucciones. Aunque UWB puede tolerar ciertos obstáculos y reflejos, cualquier inconveniente empeora la precisión de la medida.

2. Esparcir las anclas por el área y nunca en línea recta. Cada medición proporciona información en una única dirección, por lo que es mejor espacir las anclas para cubrir todas las direcciones. Si se colocan en línea recta, un pequeño cambio en la medida de distancia a un ancla, debido por ejemplo a ruido, resultará en un gran error de posición.

3. Mantener la distancia entre dos anclas en un rango de 2 a 20 metros. En una configuración de UWB estándar, un máximo de 20 metros es lo recomendado para que el tag esté siempre en rango de las anclas. Por otra parte, el algoritmo de medida de la distancia funciona mejor cuando las anclas están separadas, por lo que es mejor mantener cierta distancia entre el tag y las anclas para mejorar la precisión.

4. Colocar las anclas verticalmente con la antena en la parte superior o inferior. El sistema Pozyx utiliza antenas, que son necesarias para enviar las señales inalámbricas UWB. Sin embargo, es imposible que una antena funcione bien en todas direcciones. Debido al diseño de las antenas de Pozyx, se recomienda colocarlas verticalmente para asegurarse de que la recepción es la mejor posible. Además, el algoritmo que incorpora el tag está optimizado para esta orientación.

5. Mantener las anclas a 20 centímetros de objetos metálicos. La antena de las anclas es un conductor metálico especialmente diseñado para radiar las frecuencias de UWB. Cualquier objeto metálico cercano reducirá la eficiencia de las antenas o las hará menos omnidireccionales.

6. Colocar las antenas a diferentes alturas para posicionamiento 3D. Esta es la manera de que las antenas puedan discernir la coordenada Z con precisión. Es importante que a parte de estar a

diferentes alturas, las anclas no formen un único plano 3D [Fig. 6], ya que esto provoca que no haya una sola solución para la posición del tag.

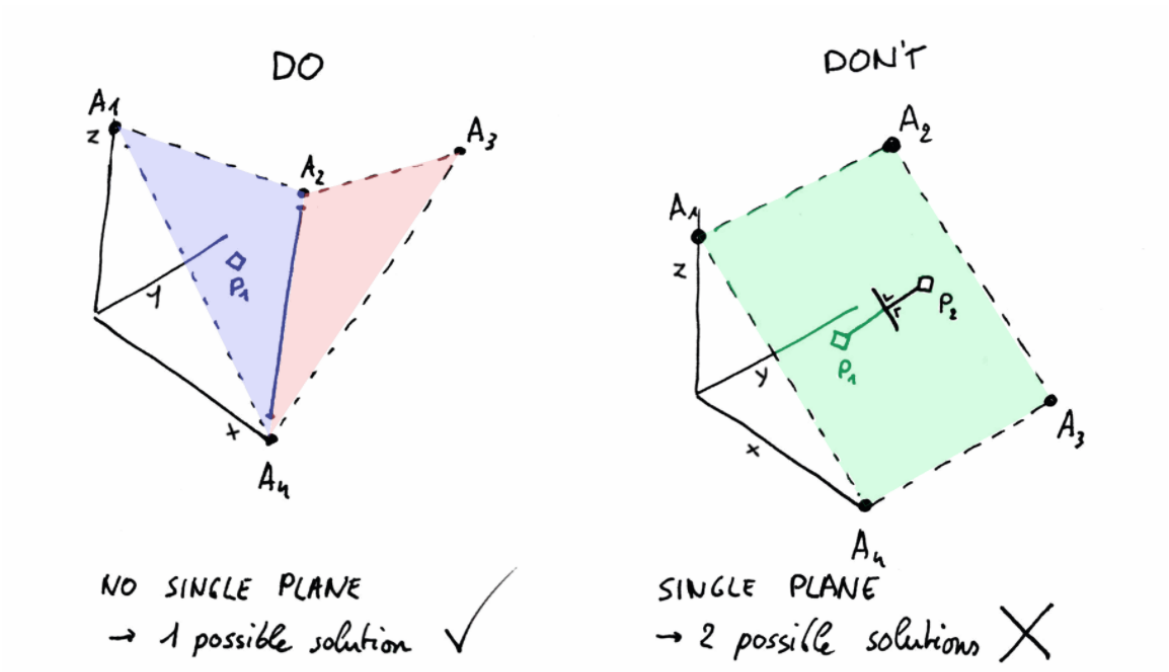

Figura 6: Colocar las anclas no formando un único plano.

5.2. Programar Arduino UNO

Una vez colocadas las anclas teniendo en cuenta las reglas anteriormente descritas, hay que programar el Arduino UNO sobre el que se conecta el tag para leer y procesar la localización realizada por el sistema Pozyx [9].

En concreto, el programa consta de cuatro secciones, la primera de ellas de configuración y las otras tres forman parte del bucle que obtiene la posición y la procesar.

Para configurar y utilizar la información que proporciona el tag se utiliza la librería de Pozyx para Arduino, que proporciona todo lo necesario y es relativamente sencilla de utilizar.

A continuación, a modo de visión general, se encuentra el código que contiene el archivo principal del programa.

```
#include <Arduino.h>
#include <Localizar.h>
#include <Filtrar.h>
#include <Transmitir.h>

void setup()
{
    Serial.begin(115200);
    initPozyx();
}

void loop()
{
    int pos[2];

    getCoordinates(pos);
    processCoordinates(pos);
    printCoordinates(pos);

    delay(20);
}
```

5.2.1. Configuración inicial

Esta etapa se ejecuta al alimentar el Arduino UNO y primeramente establece la comunicación serie a 115200 baudios, que se usará más adelante para comunicarse con el módulo del ESP8266.

Lo siguiente es configurar el sistema Pozyx. Para ello, se empieza estableciendo una serie de parámetros, comenzando por establecer cuántas anclas se van a utilizar, su identificación y qué coordenadas tienen, ya que de esto depende el valor de posición que obtiene el algoritmo.

En cuanto al algoritmo, primeramente se establece que se va a utilizar el algoritmo para localización UWB. Dentro de éste, hay tres modos, que permiten escoger cómo se van a calcular las coordenadas y adaptar esto a la aplicación en la que se utilizará el sistema Pozyx.

- 2D. En este modo se parte de que las anlcas y el tag están en el mismo plano.
- 2.5D. Las anclas se colocan a diferentes alturas, pero se fija el tag a una altura determinada, en previsión de que este sólo se mueva en un plano horizontal.
- 3D. Las anclas se colocan a diferentes alturas y el tag se puede mover en cualquier dirección.

Para este proyecto, la mejor opción es utilizar el modo 2.5D. Aunque el modo 2D podría ser suficiente, es muy suceptible a que existan obstáculos que puedan estorbar fácilemente a la transmisión de las señales UWB, a parte de que se incumplirían varias reglas de las mencionadas a cerca de la colocación de las anclas.

El modo 2.5D se beneficia del poder aplicar todas las reglas, lo que mejora notablemente la precisión del sistema. Además, teniendo en cuenta de que Pozyx no es igual de preciso en el eje Z que en el X e Y, se está evitando tener que obetner un resultado también para esta coordenada. El eliminar esta incógnita también reporta otras ventajas durante el cálculo del algoritmo, como es que, en caso de que el tag sólo pueda transmitir a tres anclas por culpa de alguna obstrucción, sólo hay una posible solución para la posición.

Por lo tanto, teniendo en cuenta que en este proyecto se va a utilizar Pozyx para obtener la posición de un robot móvil que se desplaza por una planta, se configurará tanto el modo 2.5D como la altura a la que está el tag sobre el robot, respecto al suelo.

```
#include <Arduino.h>
#include <Pozyx.h>
#include <Pozyx_definitions.h>
#include <Wire.h>

// ---------- Parametros ---------- //

// Anclas
const uint8_t num_anchors = 4;
uint16_t anchors[num_anchors] = {0x692f, 0x6931, 0x6933, 0x693a};
int32_t anchors_x[num_anchors] = {0, 0, 3550, 5380};
int32_t anchors_y[num_anchors] = {0, 4490, 4490, 0};
int32_t heights[num_anchors] = {105, 1960, 1380, 1610};
// Algoritmo y modo
uint8_t algorithm = POZYX_POS_ALG_UWB_ONLY;
uint8_t dimension = POZYX_2_5D;
int32_t height = 95;
```

Una vez establecidos los parámetros de configuración de las anclas y del algoritmo de Pozyx, hay que iniciarlo.

Se comienza por esperar a que el propio tag comunique que está funcionando correctamente y, a continuación se añaden las anclas al sistema una por una. Esto se hace mediante un bucle que asigna los parámetros de las anclas previamente configurados a los atributos correspondientes.

Finalmente, se le indica al tag el algoritmo y el modo a utilizar y, tras esperar un par de segundos, ya está el sistema listo para locarlizarse.

```
// Configurar manualmente las anclas
void setAnchorsManual()
{
    Pozyx.clearDevices();
    // Anadir las anclas
    for (int i = 0; i < num_anchors; i++)
    {
```

```
            device_coordinates_t anchor;
            anchor.network_id = anchors[i];
            anchor.flag = 0x1;
            anchor.pos.x = anchors_x[i];
            anchor.pos.y = anchors_y[i];
            anchor.pos.z = heights[i];
            Pozyx.addDevice(anchor);
        }
    }

    // Iniciar Pozyx
    void initPozyx()
    {
        // Iniciar sistema
        if (Pozyx.begin() == POZYX_FAILURE)
        {
            delay(100);
            abort();
        }
        // Configurar anclas
        setAnchorsManual();
        // Configurar algoritmo y modo
        Pozyx.setPositionAlgorithm(algorithm, dimension);
        delay(2000);
    }
```

5.2.2. Obtención de las coordenadas

Con Pozyx apropiadamente configurado, se puede empezar a obtener las coordenadas que devuelve el algoritmo cada cierto tiempo, obteniendo así la posición del robot en cada instante.

Para leer la información se le envía una instrucción al tag que incluye el modo 2.5D y la altura ya especificada. Esta instrucción devolverá si ha tenido éxito la operación y, en caso afirmativo, se puede proceder a guardar las coordenadas X e Y para su posterior utilización.

Como primera medida para evitar datos erróneos que afecten al resto del proceso, se incluye una condición para descartar cualquier valor de X e Y que sea negativo, ya que tal y como se dispone la colocación de las anclas respecto al área por la que se desplaza el robot con el tag, sólo son posibles valores positivos.

```
    // Obtener las coordenadas
    void getCoordinates(int pos[2])
    {
        coordinates_t coor;
        int status;
        // Realizar el posicionamiento
        status = Pozyx.doPositioning(&coor, dimension, height);
        // Leer el resultado
        if ((status == POZYX_SUCCESS) and (coor.x > 0 and coor.y > 0))
        {
            pos[0] = (int)coor.x;
            pos[1] = (int)coor.y;
        }
    }
```

5.2.3. Procesado de los datos

El siguiente paso tras obetner las coordenadas calculadas por el algoritmo de Pozyx, es filtar y procesar los datos para intentar eliminar medidas incorrectas e intentar mejorar la precisión.

Para ello se van a probar tres métodos distintos, cuya característica principal es utilizar resultados pasados o un conjunto de los mismos para estimar el nuevo resultado y mejorar su calidad.

- Filtro paso bajo [10]. Es uno de los más empleados en electrónica digital debido a sus buenos resultados y su implementación sencilla y eficiente. El resultado se obtiene a partir de la medición actual y el valor filtrado anterior, en conjunto con un valor entre 0 y 1 que representaría la frecuencia de corte del filtro, por lo que finalmente se obtiene un valor suavizado. Al ser un filtro paso bajo, perimitirá eliminar el ruido de alta frecuencia; es decir, valores muy dispares

a los obtenidos anteriormente, lo cual es improbable en un robot que se desplaza por una superficie.

- Filtro de media móvil [11]. En este filtro se toman los últimos N valores medidos y se calcula su media. El resultado nuevamente es una señal suavizada que elimina parte del ruido de alta frecuencia. El valor de N tiene una gran importancia ya que cuanto mayor sea, más suave es la señal, pero se aumenta el desfase entre la señal original y la señal filtrada, por lo que conviene ajustarlo adecuadamente. Este filtro además es capaz de interporlar valores, lo que se conoce como submuestreo, que aporta continuidad a la señal.
- Filtro de mediana móvil [12]. De forma similar al filtro de media móvil, este filtro utiliza las N últimas mediciones para calular la mediana. De igual manera, aumentar el tamaño de N aumenta el suaviazado de la señal pero también su desfase. Puede presentar mejores resultados ante cierto tipo de señales, pero es computacionalmente más costoso. Además, sólo puede devolver puntos muestreados en algún momento, por lo que se pierde la capacidad de submuestreo, provocando que las señales presenten saltos discretos.

En este proyecto se va a utilizar en cualquier caso el filtro paso bajo y a continuación se probará a añadir o bien el filtro de media móvil o el de mediana móvil.

Por lo tanto, en el código para Arduino UNO se empezará definiendo los parámetros de la configuración de los filtros y, a continuación, se cargarán los filtros de sus librerías correspondientes incluyendo dichos parámetros.

Por último, basta con simplemente introducir las coordenadas obtenidas por Pozyx en el primer filtro, obtener el resultado e introducirlo en el segundo filtro. El resultado de este último será el valor definitivo que se tome como posición actual del robot.

```
#include <Arduino.h>
#include <SingleEMAFilterLib.h>
#include <MeanFilterLib.h>

// ---------- Parametros ---------- //

#define fc 0.6
#define n 50

// ---------- Definiciones ---------- //

SingleEMAFilter<int> lowpass_X(fc), lowpass_Y(fc);
MeanFilter<float> mean_X(n), mean_Y(n);

// ---------- Funciones ---------- //

void processCoordinates(int pos[2])
{
    // Filtro paso bajo
    pos[0] = lowpass_X.AddValue(pos[0]);
    pos[1] = lowpass_Y.AddValue(pos[1]);
    // Media
    pos[0] = mean_X.AddValue(pos[0]);
    pos[1] = mean_Y.AddValue(pos[1]);
}
```

5.2.4. Transmitir la posición

El último paso del bucle que se dedica a calcular la posición del robot de manera continua, es transmitir el valor ya filtrado al módulo ESP8266, que posteriormente se encargará comunicarlo a través de la red WiFi.

Esto se realizará a través de comunicación serie UART. Este protocolo es muy sencillo de implentar entre la placa Arduino UNO y el módulo ESP8266, ya que ambos cuentan con los pines necesarios. En concreto, se necesitan dos pines, uno de transmisión y otro de recepción que se conectarán de manera cruzada entre los dispositivos. Gracias a esto, el protocolo permite transmitir bytes de manera asíncrona con una programación muy sencilla. Simplemente bastará con enviar una cadena de caracteres a través de la conexión UART y esta se encargará de hacerla llegar al otro dispositivo de manera adecuada.

Para no sobrecargar el canal de comunicación, el envío de las coordenadas se realizará cada medio segundo. Esto se indica con una condición que, cuando se cumpla concatenará la coordenada X e Y en una cadena de caracteres que se envía por la comunicación serie.

```
#include <Arduino.h>

unsigned long int time = 0;

// Imprimir las coordenadas
void printCoordinates(int pos[2])
{
    if (millis() - time > 500)
    {
        char buffer[20];
        sprintf(buffer, "%i,%i", pos[0], pos[1]);
        Serial.println(buffer);
        time = millis();
    }
}
```

5.3. Programar ESP8266

La programación del microprocesador ESP8266 es mucho más sencilla, ya que este módulo simplemente se va a encargar de comunicar a través de la red WiFi las coordenadas ya procesadas recibidas de la placa Arduino UNO.

Tras importar tanto la librería propia de ESP8266 como la de WebSocket, se establecen los parámetros para ambas. La primera necesita tanto el nombre como la contraseña de la red WiFi a la que se conectará el módulo. La segunda necesita el puerto que se va a utilizar en la conexión WebSocket y se crea la instancia para gestionar este protocolo, creando así un servidor.

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <WebSocketsServer.h>

// Configuracion de la red WiFi
const char *ssid = "SSID";
const char *password = "password";

// Configuracion de WebSocket
const uint8_t wsPort = 81;
WebSocketsServer webSocket = WebSocketsServer(wsPort);
```

En la fase de configuración, se inicia la comunicación serie a 115200 baudios, al igual que en la placa Arduino UNO, para que ambos puedan comunicarse correctamente.

A continuación, se configura el módulo en modo "estación"; es decir, su objetivo es conectarse a una red WiFi establecida por un punto de acceso [13]. También es necesario iniciar la conexión indicando los parámetros de la red especificados anteriormente y después se espera hasta que la conexión esté establecida. La fase de configuración concluye iniciando la conexión WebSocket.

```
void setup()
{
  Serial.begin(115200);

  // Iniciar conexion WiFi en modo estacion
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  // Esperar a que se establezca la conexion
  while (WiFi.status() != WL_CONNECTED)
    delay(500);

  // Iniciar WebSocket
  webSocket.begin();
}
```

Por su parte, el bucle consistirá en una instrucción que mantiene funcionando la conexión para que los clientes se puedan conectar al servidor. Además, se incluye una condición para que cada vez que existan datos en la comunicación serie provenientes de la placa Arduino UNO, éstes se lean y

se envíen a través de la conexión WebSocket, para que todos los clientes conectados a ella puedan leerlos.

```
void loop()
{
  // Mantener la conexion WebSocket
  webSocket.loop();

  // Cuando se reciba un nuevo dato, comunicarlo por el WebSocket
  if (Serial.available() > 0)
  {
    String data = Serial.readStringUntil('\n');
    webSocket.broadcastTXT(data);
  }
}
```

5.4. Paquete de Python para el ordenador

Los datos que envía el módulo ESP8266 a través de la red WiFi se recibirán en un ordenador y para ello se implementará un cliente WebSocket en Python. Una vez recibida la posición del robot en el ordenador, se puede utilizar tanto para implementar su control como para visualizarlo sobre un mapa.

Teniendo esto en cuenta, el código de Python utilizará dos paquetes. El primero, como es evidente, es el paquete WebSocket, que permitirá conectarse al servidor, y el segundo es el paquete *Multiprocessing*, que permitirá ejecutar el código de Python en varios procesos distintos en paralelo. El código estará estructurado de manera que forme un paquete, lo que permitirá importarlo en otros programas de Python y utilizar el sistema de localización en programas más complejos.

El paquete está formado por una clase pública, que será la que se utilice para crear un objeto en el código donde se utilice el paquete, y una privada, utilizada por el propio paquete para su funcionamiento.

La clase pública cuenta únicamente con un método contructor, encargado de crear un atributo en el que se irán actualizando las coordenadas y también crear e iniciar un proceso con la clase privada, pasándole la información que indique el usuario al crear el objeto.

El atributo de las coordenadas es una variable que se comparte de tal manera que tanto el proceso principal como el paralelo con la clase privada pueden acceder a ella. De esta manera, el proceso paralelo actualiza el valor de las coordenadas y el principal puede leearlas para usarlas.

```
import sys
from multiprocessing import Process, Manager
import websocket

# Importar Matplotlib solo si se encuentra disponible
try:
    import matplotlib.pyplot as plt
except:
    pass


class Pozyx:
    def __init__(self, address, plot=False):
        self.coor = Manager().list([0, 0])
        Process(target=_Localization, args=(self.coor, address, plot)).start()
```

La clase privada que se ejecutará en paralelo contiene un método constructor que crea el cliente WebSocket, conectándolo a la dirección IP y puerto del servidor y especificando la función a ejecutar cuando se reciba un mensaje, que será otro método de la clase privada.

Cuando el usuario crea un objeto con el paquete de este proyecto, ha de especificar la dirección IP del servidor WebSocket y si se desea activar el gráfico de visualización de los datos. Esto último habilita la ejecución de una serie de métodos en la clase privada que permiten al proceso paralelo, además de recibir y actualizar la variable de las coordenadas cada vez que se reciba un mensaje a través del WebSocket, crear un gráfico de Matplotlib, guardar las coordenadas en una lista e ir actualizando en tiempo real el gráfico.

```
class _Localization:
    def __init__(self, coor, address, plot):
```

```
        self.coor = coor
        self.__plotEnable = plot
        # Activar el grafico
        if plot is True and "matplotlib" in sys.modules:
            self.__initPlot()
        # Conectar al servidor WebSocket
        ws = websocket.WebSocketApp(
            "ws://" + address + ":81/", on_message=self.__on_message
        )
        ws.run_forever()

    def __on_message(self, _, message):
        # Leer las coordenadas recibidas
        coor = [int(c) for c in message.split(",")]
        self.coor[0], self.coor[1] = coor[0], coor[1]
        # Si el grafico esta activado, actualizar su informacion
        if self.__plotEnable is True:
            self.__pointsPlot()
            self.__updatePlot()

    def __initPlot(self):
        self.__X, self.__Y = [], []
        plt.ion()
        self.__fig, _ = plt.subplots()
        (self.__plot,) = plt.plot(self.__X, self.__Y)
        plt.axis([0, 4000, 0, 4000])
        plt.tight_layout()
        self.__fig.canvas.manager.show()

    def __pointsPlot(self):
        if len(self.__X) < 1000:
            self.__X.append(self.coor[0])
            self.__Y.append(self.coor[1])
        else:
            self.__X = self.__X[1:]
            self.__Y = self.__Y[1:]
            self.__X.append(self.coor[0])
            self.__Y.append(self.coor[1])

    def __updatePlot(self):
        self.__plot.set_xdata(self.__X)
        self.__plot.set_ydata(self.__Y)
        self.__fig.canvas.draw()
        self.__fig.canvas.flush_events()
```

La razón por la que se utiliza el procesamiento en paralelo es que, cada vez que se recibe un mensaje a través de la conexión WebSocket, se produce una interrupción que ejecuta el método configurado en el cliente. Para evitar que esta interrupción y, en caso de estar activado, el gráfico afecten en la ejecución del supuesto algoritmo que use las coordenadas para controlar el robot, es conveniente que la localización se ejecute en paralelo y simplemente se pueda acceder a la variable actualizada con las coordenadas.

Una vez terminada toda la fase de desarrollo, el flujo de información resultante se puede ver representado en el siguiente esquema [Fig. 7].

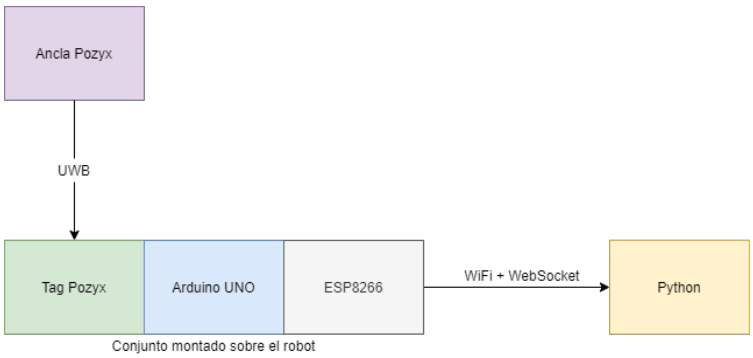

Figura 7: Flujo de información entre los componentes del sistema.

6. Pruebas de funcionamiento

Tras tener claras todas las consideraciones necesarias para implementar el sistema Pozyx y programar todos los componentes necesarios para poner en funcionamiento de manera completa el proyecto, llega el momento de comprobar que el funcionamiento es correcto y de ajustar las configuraciones necesarias.

El entorno de pruebas consistirá en un laboratorio en el que hay una maqueta de una ciudad [Fig. 8]. Esta maqueta mide aproximandamente tres metros por tres metros y medio. Presenta un par de edificios y diversos elementos que servirán como obstáculos para poner a prueba el funcionamiento del sistema y además se tendrán en cuenta al colocar las anclas de Pozyx.

Figura 8: Maqueta de la ciudad.

Una vez visto el entorno, se procede a colocar las anclas adecuadamente. Se colocarán cuatro anlcas. Una de ellas será el origen de coordenadas y a partir de ahí ser formarán unos ejes imaginarios. Se colocará un ancla sobre el eje X y otra sobre el eje Y y, la cuarta, irá en la esquina opuesta. Todas ellas estarán separadas unas de otras la distancia adecuada para cumplir las reglas.

El hecho de tener que colocarlas a distintas alturas hace que se tengan que tener muy en cuenta los edificios de la maqueta, ya que colocar un ancla a baja altura detrás de alguno de los edificios, provocaría que en la mayoría del área el tag no tuviera este ancla a la vista, perjudicando en la precisión de la localización.

Finalmente, las anclas se han dispuesto tal y como se ven en el esquema a continuación [Fig. 9]. Estas coordenas, junto con la identificación que posee cada una de las anclas, se especificará en los parámetros de la librería Pozyx que se implementa en la placa Arduino UNO.

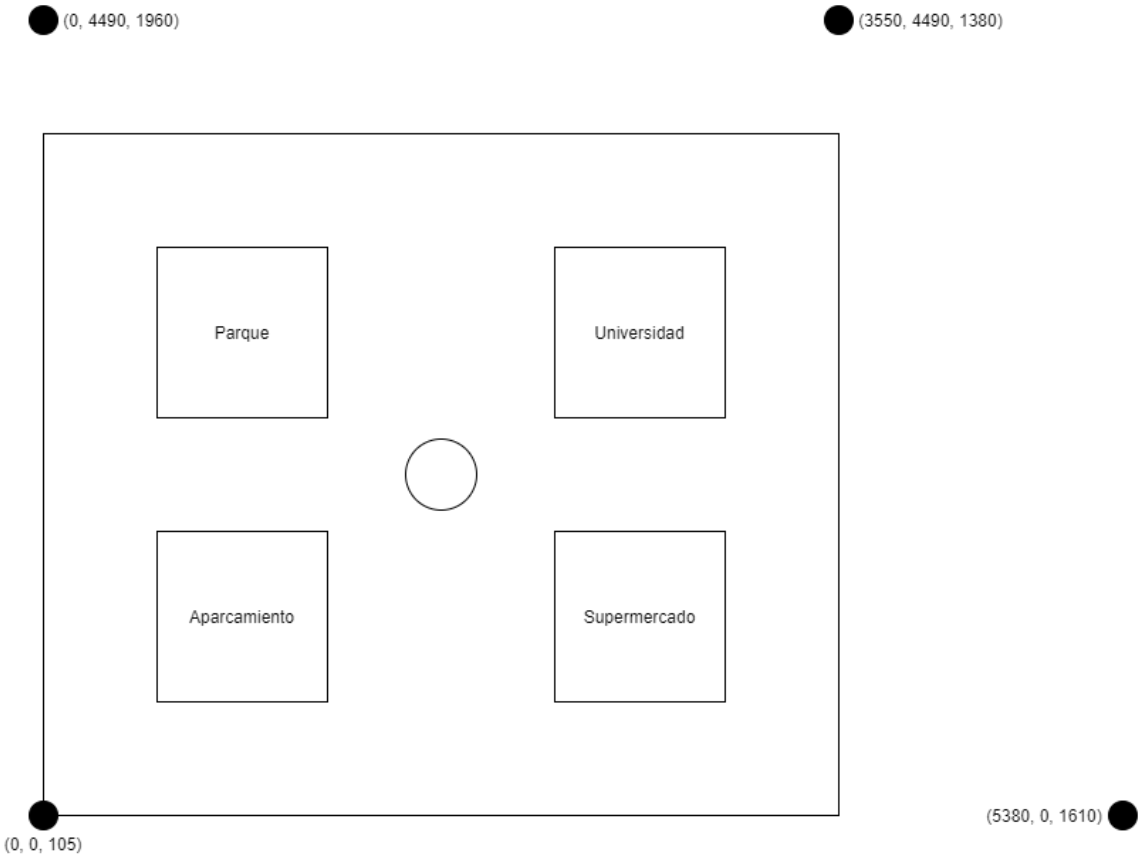

Figura 9: Esquema de la ubicación de las anclas.

Una vez preparado el entorno de pruebas con las anclas correctamente colocadas, hay que colocar el conjunto del tag sobre el Arduino UNO, una pequeña batería para alimentarlo y el módulo ESP8266 sobre un robot.

El robot a utilizar será Robobo [14]. Este robot combina una base móvil con sensores junto con las capacidades de un smartphone; es decir, la base es su cuerpo y el smartphone es su cerebro. Los smartphones actuales contienen la última tecnología en sensores, procesadores y capacidad de comunicación y, gracias a la movilidad, detección de bajo nivel y capacidades gestuales proporcionadas por la base, Robobo se convierte en la herramienta óptima para desarrollar pequeños proyectos de robótica autónoma [Fig. 10].

En cuanto a la configuración del parámetro que indica la altura que tiene el tag colocado sobre Robobo respecto del suelo, se asignarán 95 milimétros tras realizar la medida pertinente.

Figura 10: Robobo.

Con el entorno de pruebas definido y Robobo preparado, se va a probar el funcionamiento del sistema completo. Las pruebas servirán tanto para comprobar qué todos los elementos se comuniquen correctamente y Pozyx obtenga una posición adecuada, como para ver qué filtro se concatenará al filtro paso bajo y comparar la señal filtrada con la señal pura que genera Pozyx.

La primera prueba consiste en que el Robobo se desplaze de manera sencilla de una esquina a otra de la ciudad cruzando por el centro, pasando por la rotonda [Fig. 11].

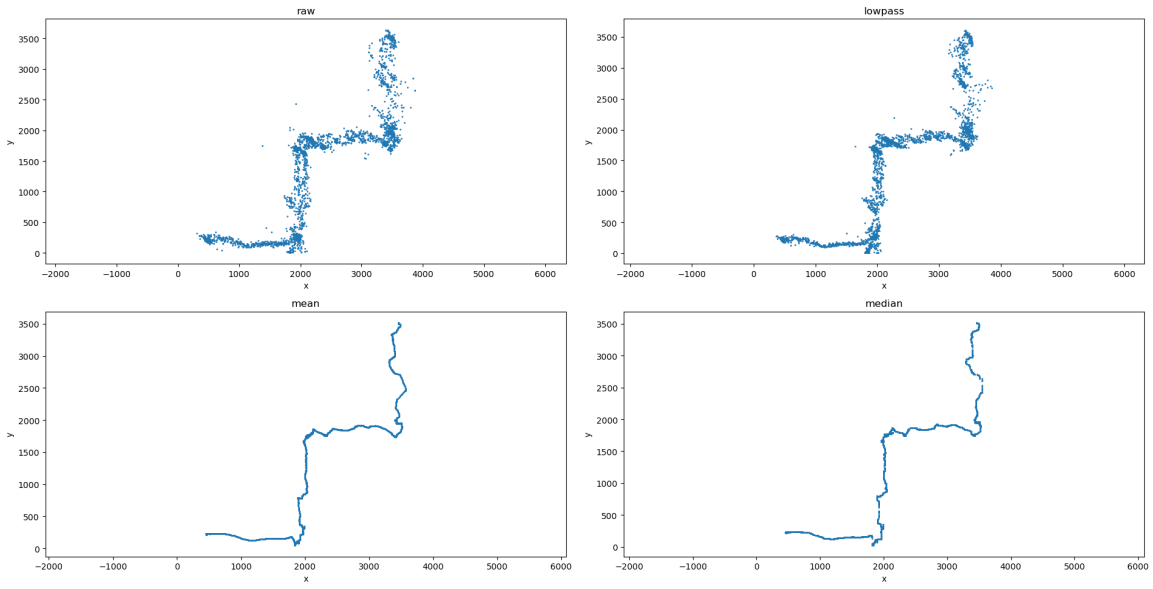

Figura 11: Prueba cruzando la ciudad.

Tal y como se puede apreciar, aunque el Pozyx por si mismo no introduce demasiados errores, produce una nube de puntos que sería difícil de gestionar por un algoritmo de control. Aplicar sobre esta nube un filtro paso bajo con una factor de 0.6, reduce la cantidad de datos dispersos y concentra la nube de tal manera que se mejore el efecto de la siguinte etapa de procesado.

En la comparación del filtro de media móvil y el filtro de mediana móvil, se observan varias diferencias que tienen relación directa con el funcionamiento del filtro. Para ambos casos se utiliza una ventana de 50 puntos, ya que junto con el delay de 20 milisegundos en el bucle que obtiene cada punto, hace que obtengamos un resultado de la posición alrededor del último segundo.

Entre las diferencias se aprecia que el filtro de media móvil suaviza más el trazado y hace un trazo más contínuo, aunque, de todos modos ambos resultados se pueden considerar relativamente buenos.

La segunda prueba consiste en que el Robobo de una vuelta a la ciudad por el carril exterior [Fig. 12], lo que pone a prueba el Pozyx ya que se van a pasar por puntos de la ciudad en la que la corbertura que proporcionan las antenas se puede ver afectada por la obstrucción que los edificios puedan hacer.

Por culpa de esto precisamente, se pueden ver dos pequeños "claros" en la nube de puntos de los datos del Pozyx en la zona de la esquina superior derecha, que permanecen tras la etapa del filtro paso bajo.

Al aplicar los filtros de media móvil o mediana móvil, esos claros pasan mucho más desapercibidos en el trazado. En el filtro de mediana móvil se pueden observar unos pequeños saltos discretos que no se ven en el filtro de media móvil, ya que éste interpola los datos para obtener un trazo contínuo.

Lo que si llama la atención, es que en la esquina superior izquierda, la medición del Pozyx generó una serie de errores, que quizás se puedan apreciar más claramente tras el filtro paso bajo. Estos errores afectan más claramente al filtro de media móvil, produciendo un pequeño desvío. Sin embargo, el filtro de mediana móvil consigue erradicar este error. Esta situación deja en evidencia una clara desventaja del submuestreo, y es que puede amplificar el efecto de ciertos errores, en lugar de eliminarlos.

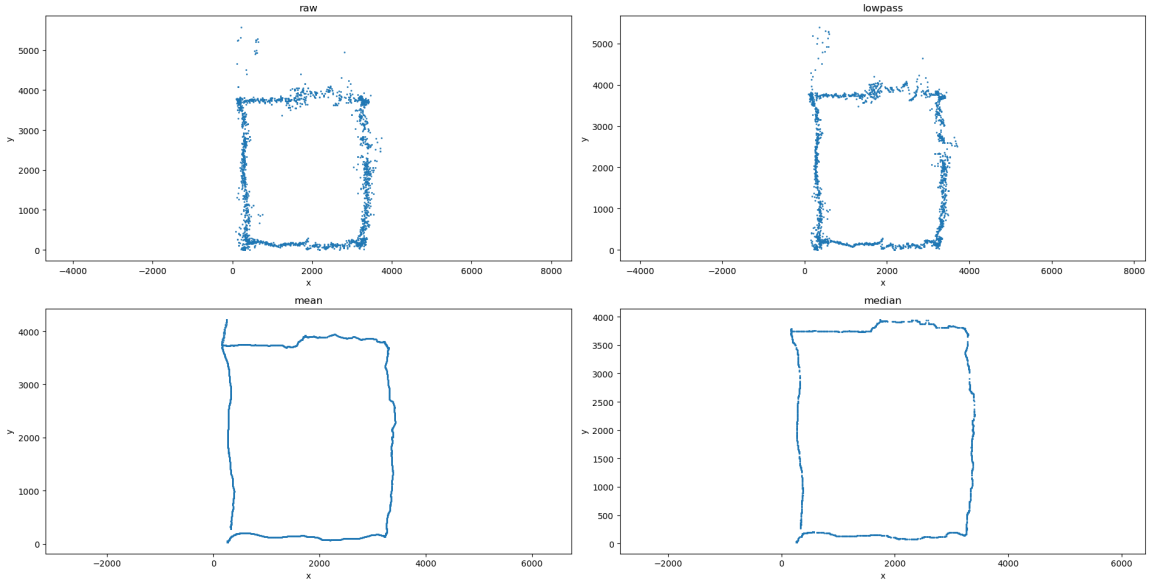

Figura 12: Prueba rodenado la ciudad.

A la vista de estas dos pruebas, se puede determinar que si lo que se busca es un trazado contínuo, la operación evidente es utilizar el filtro de media móvil, ya que gracias al submuestreo, tiene esa capacidad. Sin embargo, puede ser sensible a ciertos errores, por lo que hay que prestar especial atención a todos los elementos susceptibles de provocar defectos en la medición de Pozyx, empezando por una colocación lo más óptima posible de las anclas.

En el caso de que no se pueda evitar que se produzcan errores en determinadas zonas del mapa, la mejor opción es utilizar el filtro de mediana móvil. A costa de sacrificar el submuestreo y la consecuente generación de un trazo contínuo, se pueden llegar a reducir defectos que a priori pueden parecer más grandes de lo esperado.

7. Conclusiones

Una vez realizado por completo el desarrollo para implementar el sistema basado en Pozyx y puesto a prueba, se puede concluir que la opción de utilizar UWB como tecnología para posicionar robots móviles en interiores puede ser buena.

No obstante, hay que tener en cuenta que a lo largo de la implementación pueden aparecer distintas dificultades. El hecho de tener que indicar en el tag la posición de las anclas con precisión milimétrica puede ser súmamente complicado teniendo en cuenta que Pozyx está pensado para trabajar con distancias de entre 2 a 20 metros. Esto sumando a la serie de consideraciones que hay que tener en cuenta, hace que la colocación de las anclas sea un proceso muy delicado.

Una vez salvado este obstáculo, bien porque se cuente con equipamiento que permita realizar mediciones con gran precisión a la hora de colocar las anclas o porque el entorno no presenta demasiados retos en cuanto a obstáculos se refiere, poner Pozyx y el resto del sistema en funcionamiento es relativamente sencillo.

Entre las fortalezas del sistema completo que se ha desarrollado en este proyecto, destaca que tanto Pozyx como la comunicación inalámbrica basada en WebSocket, proprocionan una arquitectura sobre la que resultaría fácil escalar a un sistema con más robots móviles y con más equipos de control y visualización de la actividad de estos.

Además, Pozyx permite operar en áreas más grandes o con más estancias simplemente con añadir más anclas y asegurándose de que el tag mantiene conexión con un mínimo de 4 de ellas para asegurar la mejor precisión. Esto sumado a una conexión WiFi que sea capaz de cubrir todo el área, permite escalar a nivel espacial a parte de a nivel de arquitectura como se mencionó anterioremnte.

Si se quisiera proseguir en el desarrollo del sistema implementado en este proyecto, el autor propone utilizar algún otro sensor que se combine con la medida de posición obtenidad por Pozyx y que permita mejorar la precisión de éste, sobre todo en casos donde se pueda encontrar algún tipo de obstrucción que impida la comunicación directa entre el tag y algún ancla.

Para terminar, mencionar que el sistema correctamente ajustado presenta un gran punto de partida para implementar cualquier algoritmo de navegación autónoma, ya que se dispondría de la posición del robot en todo momento directamente en un ordenador, con gran precisión y una alta tasa de muestreo, que permitiría reaccionar a distintos eventos de manera rápida.

8. Bibliografía

[1] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa, "Ultra wideband indoor positioning technologies: Analysis and recent advances," *Sensors*, vol. 16, no. 5, p. 707, 2016.

[2] A. Chugunov, R. Kulikov, D. Tsaregorodcev, and N. Petukhov, "Ultra-wide band positioning for automatic guided vehicles," in *IOP Conference Series: Materials Science and Engineering*, vol. 537, p. 032093, IOP Publishing, 2019.

[3] V. Barral, P. Suárez-Casal, C. J. Escudero, and J. A. García-Naya, "Multi-sensor accurate forklift location and tracking simulation in industrial indoor environments," *Electronics*, vol. 8, no. 10, p. 1152, 2019.

[4] Pozyx, "How does uwb positioning work?," Mar 2021. Enlace.

[5] Pozyx, "How does ultra-wideband (uwb) work?," Mar 2021. Enlace.

[6] Pozyx, "Hardware setup," Mar 2021. Enlace.

[7] Arduino, "Arduino board uno," May 2021. Enlace.

[8] R. Mischianti, "Websocket on arduino, esp8266 and esp32," Dec 2020. Enlace.

[9] Pozyx, "Tutorial 2: Ready to localize (arduino)," Mar 2021. Enlace.

[10] L. Llamas, "Filtro paso bajo y paso alto exponencial (ema) en arduino," Mar 2017. Enlace.

[11] L. Llamas, "Implementar un filtro de media móvil rápido en arduino," Mar 2017. Enlace.

[12] L. Llamas, "Implementar un filtro de mediana móvil rápido en arduino," Apr 2017. Enlace.

[13] E. C. Forum, "Esp8266wifi library," Apr 2017. Enlace.

[14] R. Project, "¿qué es robobo?," Feb 2021. Enlace.