TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Sistema de localización entre radios de banda ultraancha

Autor
José Hinojosa Hidalgo

Directores
Héctor García de Marina Peinado

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2024

Sistema de localización entre radios de banda ultraancha

José Hinojosa Hidalgo

Palabras clave: localización, robótica, redes, descentralización

Resumen

Las soluciones maduras y comerciales de localización en la actualidad se basan en la centralización para la obtención de medidas de posicionamiento; por ejemplo, los sistemas de navegación por satélite o la captura de movimiento. No obstante, en campos como la robótica móvil hay necesidad de un sistema de posicionamiento entre robots preferentemente distribuido, es decir, que no dependa de una infraestructura externa ni de un sistema de referencia proporcionado por anclajes conocidos o satélites.

Este proyecto propone una prueba de concepto para llevar la localización entre robots de manera distribuida a entornos anteriormente no contemplados, conocidos como entornos no estructurados o desconocidos. Proponemos un prototipo que no necesita de infraestructura externa o conocimiento previo. Además permite a los elementos localizados ser independientes e iguales, de forma que un robot pueda obtener su posición relativa a los demás robots sin necesidad de conocer características globales de la red.

La tecnología usada en este proyecto para permite obtener posiciones relativas entre robots está basada en la comunicación radio de banda ultraancha o UWB, por sus siglas en inglés. Esta tecnología permite realizar una medida del tiempo de vuelo de los mensajes entre radios para obtener una noción de distancia.

Este proyecto desarrollará una plataforma basada en un sistema operativo de tiempo real que permita explotar los UWB de manera descentralizada.

Los resultados de este proyecto esperan tener un impacto en el sector del control y la robótica ya que actualmente estos sistemas de localización relativa no están lo suficientemente explorados ni por tanto estandarizados.

Relative localization system based on UWB radio communication

Hinojosa Hidalgo, Jose

Keywords: localization, UWB, ranging, mobile robotics, distributed

Abstract

Commercial localization solutions today rely heavily on centralized systems like satellite navigation or motion capture for obtaining positioning measurements. However, in fields like mobile robotics, there is often a need for a distributed positioning system, due to the absence of external infrastructure or reference systems provided by anchors or satellites.

This project proposes a proof of concept implementing distributed localization among robots in previously unexplored environments, which are known as unstructured or unknown environments. Our prototype does not require any external infrastructure or prior knowledge, enabling localized devices to operate independently with identical behavior. This allows a robot to determine its relative position to other robots without needing any global information about the network.

Ultra-Wideband (UWB) radio communication technology is used to measure the time of flight of messages between radios, which helps estimate the distance between sender and receiver. This project will develop a software platform based on a real-time operating system to leverage UWB technology in a decentralized manner.

The anticipated results of this project aim to significantly impact the control and robotics fields, as currently avaliable relative localization systems are underexplored and lack widespread standardization.

Yo, **José Hinojosa Hidalgo**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 51216129M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: José Hinojosa Hidalgo

Granada a 24 de Junio de 2024.

D. **Héctor García de Marina Peinado**, Profesor del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Sistema de localización entre radios de banda ultraancha***, ha sido realizado bajo su supervisión por **José Hinojosa Hidalgo**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 24 de Junio de 2024.

Los directores:

Héctor García de Marina Peinado

Agradecimientos

A Héctor García de Marina Peinado, por guiarme y ayudarme durante todo el proyecto, gracias por confiar en mí.

A Jesús Bautista por formar parte del equipo y resolver muchas de mis dudas.

A Ángel por ser un gran compañero y amigo apoyándome durante este proyecto.

A Swarm System Labs y sus colaboradores que han sentado las bases y han hecho posible todo este desarrollo.

A mi familia que en todo momento me ha apoyado y me ha mostrado su cariño.

A mis amigos, sin los cuáles no hubiera podido superar los momentos más difíciles.

A mis profesores gracias a los cuáles tengo los conocimientos y herramientas necesarias para realizar este trabajo.

A la escuela y universidad que me han proporcionado la oportunidad de formarme y desarrollarme académicamente.

Índice general

Índice de figuras

Siglas

ARQ Automatic repeat request. 18, 29, 43

CRC Cyclic redundancy check. 14, 18

DS-TWR Double-sided two-way ranging. 59

FEC Forward error correction. 18

HAL Hardware abstraction layer. III, 7, 21, 25, 35, 36, 42

HDLC High-level data link control. III, 30

IOT Internet of things. 15

IVT Interrupt vector table. 26

KPI Key performance indicator. 7, 56

LDE Leading edge detection. 13, 14

LLC Logical link control. 29, 43

LOS Line of sight. III, 12

LR-WPAN Low-rate wireless personal area network. 15

MAC Media access control. 14, 15, 22, 28, 43

MHR MAC header. 28

NLOS Non Line of sight. III, 12

OSI Open systems interconnection. III, 29

OTA Over-the-air. 59

RTLS Real time location system. 13

RTOS Real-time operating system. 4, 20

SPI Serial peripheral interface. III, 24, 25, 28, 35, 38, 39

SS-TWR Single-sided two-way ranging. III, 17, 29, 31, 32, 40, 43–45

TDOA Time differnce of arrival. 17

TFG Trabajo de fin de Grado. 2, 12

TOA Time of arrival. 17

TOF Time of flight. 17

TWR Two-way ranging. III, 8, 17, 21, 29, 31, 46

UART Universal asynchronous receiver/transmitter. 16, 35

UWB Ultra-wideband. III, 2, 3, 8, 12

Capítulo 1

Introducción

1.1. Motivación

En el campo de la **robótica** y teoría de control, los sistemas de localización y posicionamiento han sido una fuente de datos e información vital para gran parte de aplicaciones reales e investigación. Desde reconocimiento, operaciones no supervisadas, rescate o exploración hasta sistemas multi-robot y enjambres, la información de localización es uno de los datos más importantes entre toda la información proporcionada por los sensores disponibles.

En la mayoría de casos, sistemas de posicionamiento global como **GPS** proporcionan esta información. Sin embargo, debido a diversas limitaciones como la necesidad de conexión con satélites, precisión, o fiabilidad, en ocasiones se requieren de sistemas de localización relativa.

Estos sistemas, o al menos los basados en comunicaciones inalámbricas, también cuentan con sus propias limitaciones, al necesitar de dispositivos con posiciones conocidas entre sí para obtener información útil. Existen múltiples aplicaciones a las que estas limitaciones les hacen imposible implementar sistemas de localización, por ejemplo, en enjambres de robots **distribuidos**.

En este caso, es posible que no exista una infraestructura externa y los sistemas de posicionamiento global no estén disponibles ,o no proporcionen datos adecuados, ya sea debido a la naturaleza de estos o limitaciones de exactitud o precisión. Además en este tipo de aplicaciones la información de localización relativa proporciona un enorme conocimiento a cada agente del estado del los demás y su entorno.

Para solventar estas limitaciones y permitir la obtención de esta información sin infraestructura externa, es necesario un sistema de localización distribuido, en el que cada agente obtiene los datos de localización necesarios sin necesidad de una entidad central o cualquier infraestructura, únicamente dependiendo de los demás robots en el enjambre.

1.2. Alcance del proyecto

Este proyecto apunta a desarrollar un **sistema de localización distribuido**, proporcionando información de posicionamiento. Las aplicaciones previstas con unas necesidades similares a la plataforma proporcionada incluyen sistemas como un enjambres de robots móviles.

Para obtener las medidas necesarias se ha decidido usar comunicaciones radio de banda ultraancha o **UWB**. Esta tecnología es usada por la mayoría de sistemas de localización basados en radio comerciales debido a las numerosas ventajas que presenta. Debido a la naturaleza de un **TFG**, se limitará el desarrollo de los componentes del proyecto a las necesidades más básicas de las aplicaciones mencionadas, abriendo una puerta a la expansión del proyecto en un futuro donde más funcionalidades y mejoras puedan añadirse.

1.3. Fundamentos y tecnologías

1.3.1. Fundamentos y objetivos

El objetivo fundamental del proyecto es poder proporcionar información de localización relativa a un enjambre de robots equipados con radios UWB, la localización proporcionada será de únicamente el entorno cercano a cada agente, es decir, información sobre la localización de sus vecinos.

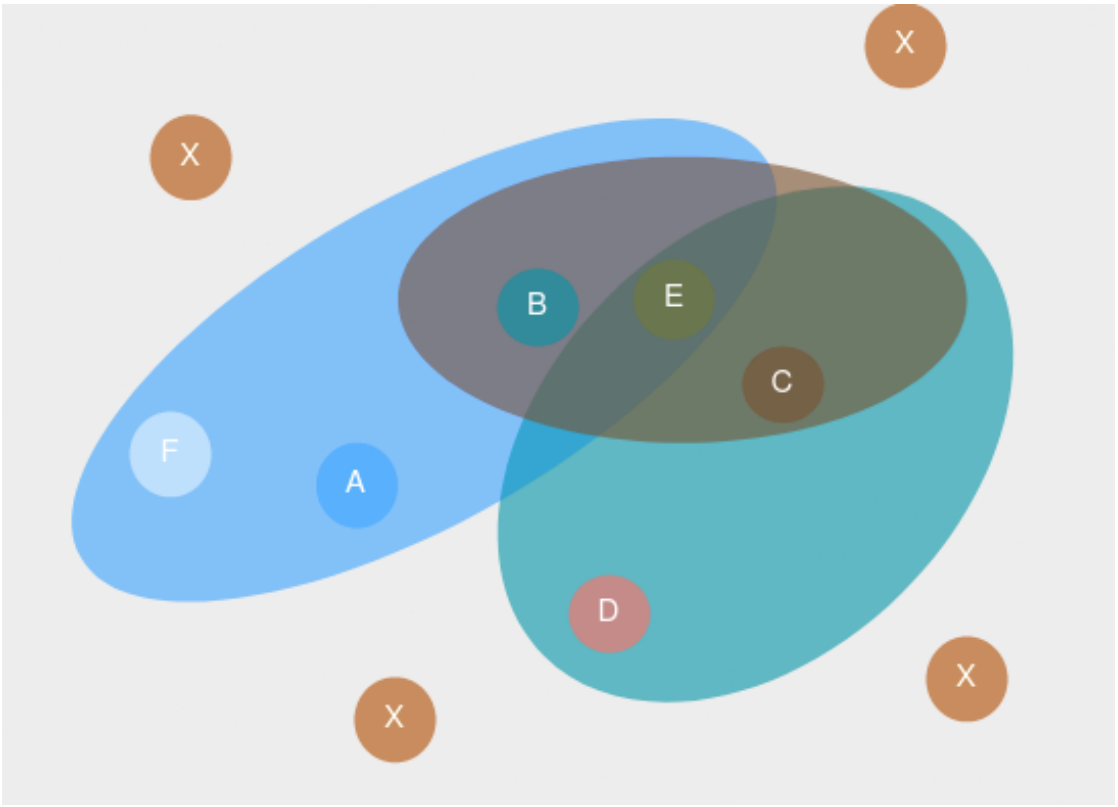

Figura 1.1: Representación de entornos locales, cada dispositivo puede formar un entorno con otros 2 dispositivos

Esta información deberá obtenerse de manera distribuida y sin depender de ninguna estructura o agente específico, recogiendo los datos únicamente mediante intercambio de mensajes dentro del entorno del agente en el enjambre. De esta forma se garantiza que el sistema sea funcional en un gran número de entornos y situaciones distintas, pudiendo

variar tanto el entorno físico como la propia organización y composición del enjambre de robots.

1.3.2. Tecnologías principales

Como se ha mencionado anteriormente el sistema se basa en las comunicaciones UWB, esta tecnología es usada en la mayoría de sistemas comerciales debido a que proporciona una relación entre la complejidad, precio, escalabilidad, precisión y exactitud muy deseables. Esto se debe a que como su nombre indica su gran ancho de banda permite obtener una serie de propiedades deseables:

- **Exactitud en la medidas de tiempo**: Debido al gran rango de frecuencias se pueden producir pulsos más cortos lo que aumenta la resolución temporal.
- **Penetración de la señal**: Las distintas frecuencias del rango son capaces de penetrar materiales, con las frecuencias bajas penetrando más fácilmente.
- **Bajo coste del transmisor**: Las señales usadas no cuentan con una señal portadora, por lo que la fabricación del transmisor se puede simplificar abaratando costes.
- **Resistencia a interferencias**: La potencia de la señal de UWB puede ser menor debido a su gran ancho de banda, esto evita interferencias, además la resolución temporal permite distinguir las señal recibida desde múltiples caminos y así mejorarla, en lugar de degradarla.

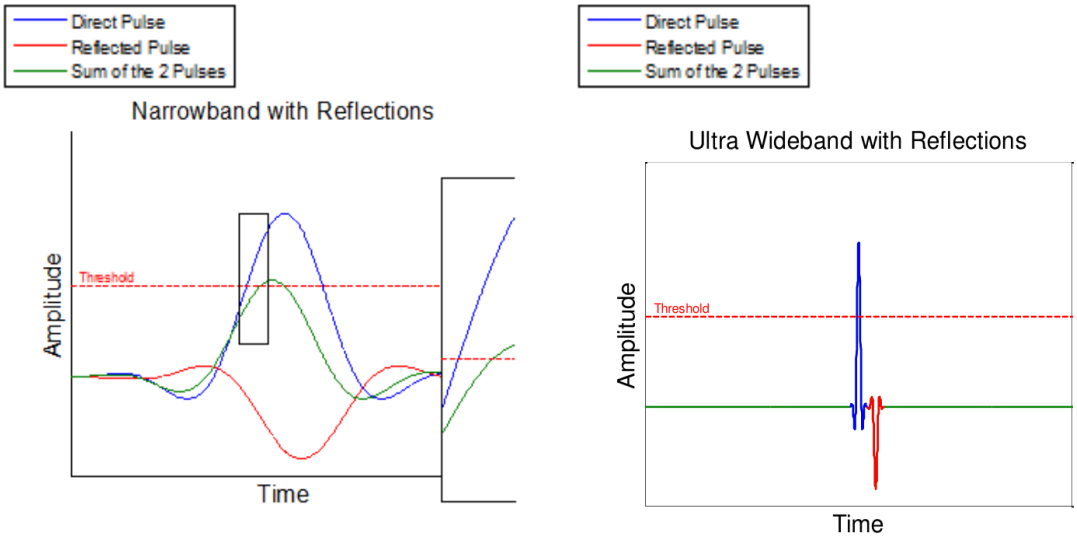

Figura 1.2: Comportamiento de comunicaciones UWB con múltiples caminos[21]

Debido a las aplicaciones a las sirve esta tecnología como la robótica, el hardware suele ser de baja potencia, normalmente estando basado en microcontroladores. El desarrollo en estos dispositivos debe cumplir una serie de condiciones como pueden ser una limitada huella en memoria, un consumo reducido o una alta fiabilidad.

Teniendo en cuenta también los objetivos del proyecto, las utilidades proporcionadas por un sistema operativo de tiempo real (RTOS) como pueden ser las organización por prioridades para poder gestionar la recepción y envío de mensajes a la vez que el procesamiento o las garantías temporales para asegurar las correcta medidas de tiempo necesarias.

1.4. Notación

Los instantes de tiempo, así como los intervalos de tiempo, se han denominado con el símbolo T, normalmente con un subíndice descriptivo, por ejemplo, el intervalo de tiempo del tiempo de vuelo de un mensaje: T_{tof}

La distancias medida, como puede ser d_{ij}, denota la distancia en metros medida desde el dispositivo i con el dispositivo j.

El uso de la notación de norma $||x||$, denota la **norma euclídea**, definida en $\mathbb{R}^n$ como $\sqrt[2]{x_1^2 + ... + x_n^2}$

1.5. Aclaraciones

Durante el resto del proyecto, cuando se haga referencia a la distancia medida se tiene en cuenta que esta es una *noción de distancia*, debido a que por errores de medida, la distancia medida puede no representar la distancia real de forma directa.

Se entiendo noción de distancia como un valor relacionado de alguna forma con la distancia real y que por tanto puede ser usado de forma útil en sistemas de localización y otras aplicaciones.

Capítulo 2

Especificación del problema

Este proyecto, como se ha mencionado anteriormente esta enfocado principalmente a una serie de aplicaciones en robótica multi-robot, debido a esto la especificación del problema presentado se dividirá en tres secciones.

1. Requisitos, requisitos generales del sistema y sus propiedades
2. Formalización, muchos de los requisitos y casos de uso se pueden formalizar para definir su alcance
3. Propuestas, recomendaciones y requisitos específicos relacionados con las características de las principales aplicaciones

2.1. Requisitos

El sistema propuesto puede describirse mediante los siguientes requisitos de desarrollo:

- Desarrollo de un controlador básico para el transceptor UWB DW1000[25]. Este deberá proporcionar una interfaz mínima que permita el intercambio de mensajes entre dispositivos y el acceso a la información de las medidas de tiempo.
- Obtención de una noción de distancia entre una pareja de dispositivos.
- Obtención de las diferentes distancias a un número de dispositivos cercanos determinado formando parejas.
- Cálculo de la localización relativa de los dispositivos considerados respecto a sus vecinos.
- Capacidad para los agentes de unirse al enjambre dinámicamente, en cualquier momento del proceso.
- Estructura distribuida, donde todos los dispositivos deben tener el mismo comportamiento e información sobre su entorno.
- Descentralización del sistema, no dependiendo en ningún dispositivo específico con un conocimiento o responsabilidad mayor a los demás integrantes del sistema.

2.2. Formalización

La especificación de los requisitos en lenguaje natural permite comenzar el proceso de análisis para buscar una solución al problema presentado. Adicionalmente se formalizarán algunos aspectos del problema, de forma que se delimiten y definan claramente los componentes y requisitos con una descripción matemática.

Partimos de un conjunto de n dispositivos localizados $S = \{e_1, e_2, ..., e_n\}, \quad n \in \mathbb{N}$, con capacidad hardware para intercambiar mensajes y obtener la distancia estimada a partir del tiempo de vuelo de estos. Asumimos que distancias reales entre dispositivos se encuentran en un rango finito definido.

Para cada dispositivo definimos T_i como el conjunto de sus vecinos, definiendo m como el número de vecinos.

$$\{\forall e_i \quad \exists T_i \subset S : \quad e_i \in T_i, \quad |T_i| = m\}$$

Para cada vecino en el entorno de un dispositivo, se define una posición relativa.

$$\{\forall a_j \in T_i \quad \exists r_j : \quad r_j \in \mathbb{R} \times \mathbb{R}, \quad 0 \leq ||r_j|| < K, \quad K \in \mathbb{R}^+\}$$

En un conjunto de vecinos el dispositivo desde el cual se define el entorno se encuentra en el origen como posición de referencia.

$$r_i = (0, 0)$$

Se define una matriz D_{EDM} para cada entorno T_i con todas las distancias entre los dispositivos.

$$D_{EDM} = (d_{ij})$$

$$d_{ij} = ||r_j - r_k||$$

$$d_{ii} = 0$$

$$D_{EDM} = \begin{bmatrix} 0 & d_{12}^2 & d_{03}^2 & \dots & d_{0n}^2 \\ d_{21}^2 & 0 & d_{23}^2 & \dots & d_{2n}^2 \\ d_{31}^2 & d_{32}^2 & 0 & \dots & d_{3n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1}^2 & d_{n2}^2 & d_{n3}^2 & \dots & 0 \end{bmatrix}$$

Para considerar el problema resuelto se debe encontrar un conjunto W para cada dispositivo e_i tal que:

$$W = \{r_1, r_2, ...r_m\}$$

Esto se puede conseguir obteniendo las distancias entre dispositivos mediante medidas del tiempo de vuelo de los mensajes intercambiados.

2.3. Propuestas

Las siguientes propuestas describen elementos relacionados al proyecto cuyo uso o implementación es recomendado o requerido.

- Uso de la placa **DWM1001-DEV**[26], una placa de desarrollo que cuenta con el módulo transceptor[25]
- Uso de la plataforma **ChibiOS**[16], en especifico el uso de su HAL para facilitar la interacción con el hardware y su sistema operativo de tiempo real para permitir la implementación de las comunicaciones.
- Definir el **entorno cercano** de los dispositivos como un número de vecinos fijo, esto consigue limitar el alcance del sistema y aumentar la escalabilidad, ya que el entorno local se mantiene independientemente de la escala global.

2.4. Métricas e indicadores

Para evaluar el desarrollo y rendimiento del proyecto es conveniente tener en cuenta métricas e indicadores que cuantifiquen los resultados obtenidos.

Las principales métricas a tener en cuentan son las siguientes variables cuantitativas:

- **Exactitud**: diferencia en valor absoluto entre la distancia real y la distancia medida media.
- **Precisión**: desviación típica en una serie de medidas en las mismas condiciones.
- **Frecuencia** de medida: número de medidas de distancia válidas tomadas por segundo.

También se elegirán una serie de indicadores cualitativos para facilitar la validación de las principales propiedades que el sistema debe tener. Como medida de rendimiento se define un indicador de rendimiento clave o, en inglés, *key performance indicator*, que habilita la posibilidad de validar si el rendimiento del sistema es satisfactorio.

Como KPI se selecciona la capacidad de obtener posiciones representativas de la formación de polígonos. Este indicador representa de forma contenida una gran cantidad de información relevante para la robótica móvil y otras aplicaciones. como distancias, ángulos, posiciones relativas etc.

Otros indicadores relevantes para el sistema son:

- **Fiabilidad**, resistencia a errores, perduración en el tiempo de las comunicaciones.
- **Escalabilidad**, posibilidad de aumentar número de nodos total arbitrariamente sin limitaciones debidas a la solución desarrollada.
- **Descentralización**, falta de dependencias en determinados dispositivos centrales para el funcionamiento del sistema.

Capítulo 3

Planificación

En este capítulo se presentará la planificación temporal del proyecto, además de una estimación del presupuesto necesario para su completo desarrollo. Adicionalmente se mencionarán otros recursos que no presentan una impacto temporal o económico pero son vitales para el desarrollo del proyecto.

3.1. Fases de desarrollo

La planificación temporal del proyecto se dividirá en fases de desarrollo en las que se diseñarán e implementaran los distintos módulos que conforman el sistema. Estas fases han sido elegidas de forma previa al comienzo del desarrollo y por tanto pueden sufrir cambios durante el transcurso del mismo, los cuáles se verán reflejados en la sección 8.1. Se han decidido incluir en la planificación las siguientes fases del desarrollo:

- Período inicial de familiarización con las tecnologías(1 semana)
- Desarrollo del controlador para DW1000(3 semanas)
- Pruebas del controlador y comunicación por UWB(1 semana)
- Desarrollo de la comunicación básica y intercambio de mensajes para TWR entre dos dispositivos(1 semana)
- Desarrollo del protocolo de comunicación (3 semanas)
- Desarrollo de las toma de medidas mediante TWR (2 semanas)
- Validación, calibración, toma de medidas y realización de experimentos (1 semana)
- Evaluación de los resultados y finalización de la memoria (2 semanas)

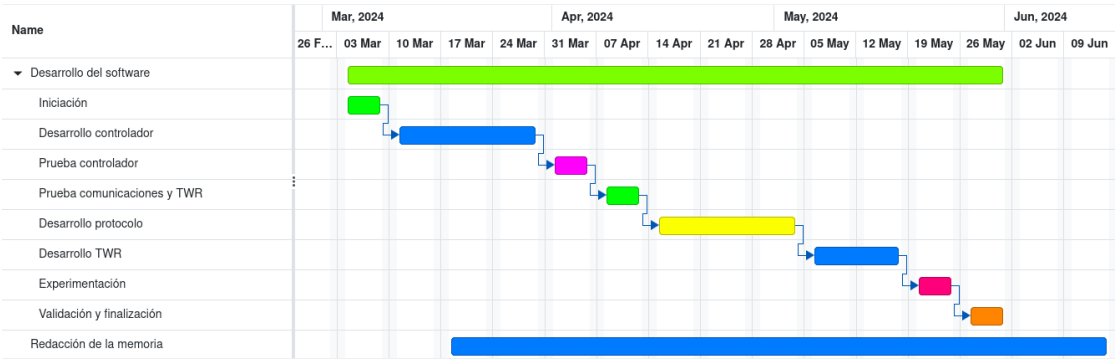

Figura 3.1: Diagrama de Gantt de la planificación inicial

3.2. Presupuesto

En el presupuestos se incluirán principalmente dos gastos: recursos hardware y recursos humanos. No se incluyen los gastos relacionados a los recursos software ya que el software usado ha sido mayoritariamente software libre como se detallará a continuación.

El coste de los recursos hardware se ha estimado usando los precios oficiales disponibles al consumidor en el momento de la redacción de esta memoria. Para estimar el conste de la estación de trabajo se ha tomado el precio de un ordenador portátil actual de gama media-alta (1000€) y se ha calculado el coste usando un período de amortización de 3 años y un período de uso de 3 meses.

Para el cálculo del coste de los recursos humanos se ha tenido en cuenta como personal un ingeniero de software embebido, considerando el sueldo mensual medio en la profesión[34] y un período de 3 meses de desarrollo. Para calcular el coste no se debe tomar el salario que recibe el empleado sino el coste que supone a la empresa, este valor se ha estimado mediante una herramienta online[14].

Recursos	**Coste**
Ing. Sistemas Embebidos	12.368,22€
Estación de trabajo	83,34€
DWM1001-DEV(4)	127,44€
Baterías(4)	22,50€
Otros recursos materiales	12,50€
Coste Total	**12.614€**

3.3. Otros recursos

Los recursos mencionados en esta sección no se han incluido en el presupuesto debido a que, o bien no tiene coste, o bien su coste no debe ser repercutido a este proyecto. Entre estos recursos, los que merecen mención explícita son:

- Los recursos software usados, como puede ser la herramienta de programación openOCD[31] o el sistema operativo ChibiOS[16], son completamente gratuitos. La mayoría de este software es software libre, aunque si algún programa usado requiere una licencia comercial, en todo caso se le ha dado un uso en este proyecto donde la licencia requerida también es gratuita.
- Espacios e instalaciones para el desarrollo y realización de los experimentos, son espacios de la Universidad de Granadaque son usado por otros proyectos.
- Instrumentación de medida, es también compartida con otros proyectos.

Capítulo 4

Análisis del problema

En este capítulo se pondrán en evidencia los procesos de análisis utilizados durante el desarrollo del proyecto, además de delimitar las condiciones supuestas y justificar las decisiones tomadas en aspectos de diseño, implementación y validación.

4.1. Características del sistema

En esta sección se detallan las características del sistema a desarrollar, ya sea debido a suposiciones asumidas o realidades de la tecnología o el hardware usado.

4.1.1. Realidad física y efectos del canal

Al tratar con un sistema físico en condiciones reales, cabe esperar que diversos efectos y fenómenos condicionen las posibilidades del hardware, esto proporciona un marco donde analizar las capacidades y limitaciones del sistema desde el punto de vista más elemental.

Analizando las capacidades del sistema respecto a las limitaciones debidas a efectos físicos y fabricación del hardware se pueden observar las siguientes características:

- **Densidad de mensajes baja**, debido al canal compartido por todos los dispositivos de comunicación inalámbrica y a la implementación del hardware que usa ALOHA como protocolo de acceso al canal, la densidad temporal de mensajes es reducida y depende de la velocidad de transmisión seleccionada, se especifica en [6]
- **Exactitud y precision decimétrica** en condiciones ideales, las posibilidades del hardware en cuanto a exactitud y precision se encuentran especificadas en el manual del transceptor [6] y documentación de los efectos del canal [21] sobre las comunicaciones respectivamente.
- **Rangos** de comunicación entre 1 y 100 metros, el rango de distancias en los que las comunicaciones se pueden mantener de forma fiable en la mayoría de configuraciones, aplicaciones y experimentos. El rango máximo del transceptor DW1000 es de alrededor de **290m** según las especificaciones[9].

Debido a la naturaleza de las aplicaciones propuestas se requiere de la capacidad de obtener resultados similares en condiciones similares o repetibilidad, además de esto también es deseable que el entorno no afecte de forma disruptiva al sistema.

A pesar de que cambios físicos en el entorno, como temperatura, presión o incluso el voltaje de alimentación, pueden afectar a los resultados, se busca que estos sigan siendo de utilidad de forma que el sistema no quede confinado a un entorno predeterminado. El hecho de reducir la operación del sistema a unas condiciones específicas terminaría socavando el propósito del proyecto basado en no depender de una infraestructura externa.

La mayoría de entornos previstos son entornos en los que existe una líneas de visión (entorno LOS), estos entornos son ideales para la propagación de las señales y para medir los tiempos de envío y recepción.

Esto no elimina la posibilidad de que el sistema opere correctamente en entornos NLOS, ya que como se ha comentado anteriormente las comunicaciones UWB son resistentes a las reflexiones e interferencias y tienen un buen comportamiento sin línea de visión.

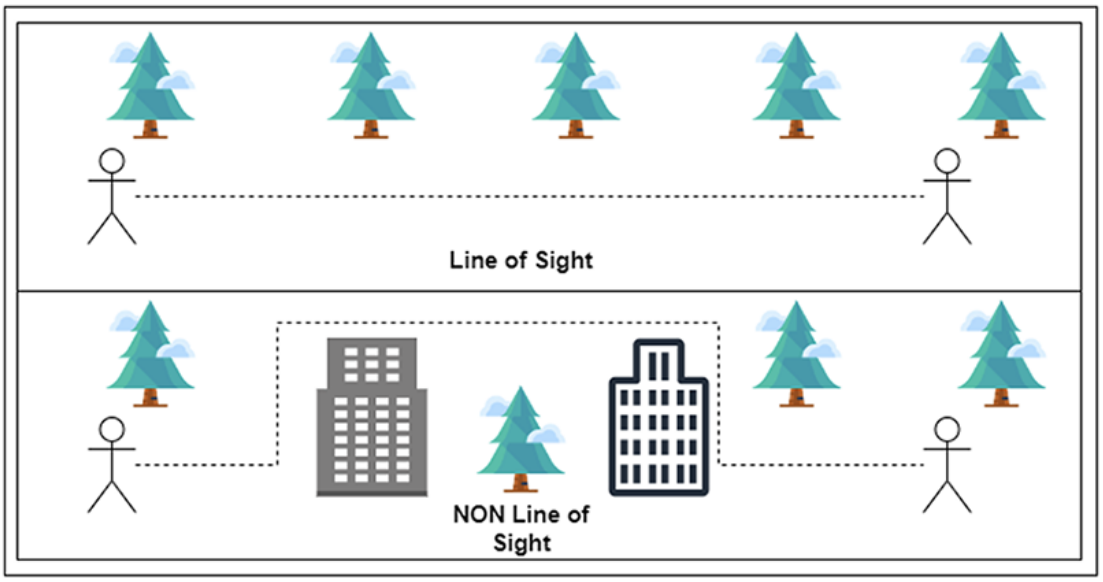

Figura 4.1: Comunicación LOS frente a NLOS[30]

4.1.2. Hardware

Debido al alcance de este TFG y los recursos disponibles, la calibración del hardware, específicamente de los distintos parámetros del transceptor será mínima.

Los parámetros usados durante el desarrollo del proyecto serán los indicados en la configuración por defecto de este[6], realizando una calibración suficiente para la funcionalidad de las comunicaciones y la obtención de resultados coherentes en las medidas de distancia.

Sin embargo no se realizarán estudios de las diferentes fuentes de error indicadas en la documentación oficial[21][22], simplemente implementando la configuración por defecto recomendada y realizando una calibración del retardo de la antena simple para la validación final según las especificaciones del fabricante[20].

La configuración del transceptor puede tener un gran efecto en las características y capacidades del sistema, al ser altamente configurable el circuito integrado DW1000 permite modificar diversos parámetros como:

- Potencia de transmisión
- Velocidad de transmisión
- Longitud del preámbulo
- Canal(frecuencia, ancho de banda...)
- Frecuencia del preámbulo

La optimización de estos parámetros puede resultar en sistemas muy diferentes, con un mayor rango o una mayor velocidad de transmisión. Hay que tener en cuenta que maximizar una característica normalmente resulta en en el sacrificio del rendimiento de otra característica, por ejemplo, aumentar la velocidad de transmisión implica una reducción del rango como se muestra en la figura 4.2. Dicho esto, la configuración por defecto es suficiente para cumplir con los requisitos presentados por lo que se asumirá su uso.

Podemos diferenciar los procesos y efectos más significativos que pueden afectar a las medidas de distancia añadiendo error, algunos son descritos en la documentación del fabricante[22].

- Para obtener el tiempo de recepción de un mensaje, el transceptor debe identificar el momento exacto en el que llega el primer flanco de la señal recibida. Este proceso se realiza mediante el algoritmo LDE y ajusta el tiempo de recepción del mensaje para que pueda reflejar correctamente el tiempo de vuelo de este.

  Como es lógico el correcto funcionamiento de este algoritmo depende de varios factores, como las condiciones de la señal recibida, el entorno o la configuración del receptor, por lo que puede ser una fuente de error en caso de fallo.

- Una de las principales preocupaciones en este tipo de sistemas es la deriva de reloj, al necesitar tiempos medidos en dos dispositivos distintos es imperativo que los relojes de estos se encuentren sincronizados para poder interpretar la medida.

  Sin embargo esto no sucede en la realidad debido a efectos de deriva del reloj y discrepancias en la fase de la señal, el hardware del DW1000 implementa diversos registros con información sobre la deriva de reloj que permiten corregir los tiempos tomados, aunque esta sigue siendo una fuente de error considerable.

- En cualquier comunicación inalámbrica se requiere de una antena para transmitir el mensaje por el medio mediante radiación electromagnética, las antenas físicas, como es esperable, cuentan con un retardo para transmitir y recibir.

  Este retardo altera en gran medida el tiempo de vuelo calculado y por tanto supone una gran fuente de error tanto en precisión como en exactitud para los sistemas RTLS. A pesar de esto, mediante procesos de calibración se puede estimar el valor de estos retardos en distintas condiciones y mitigar los errores que producen en gran medida.

- Otro parámetro que provoca errores en la medida es la potencia de la señal, al recibir una señal el algoritmo LDE debería identificar el momento del primer flanco recibido independientemente de la potencia de la señal, sin embargo, existe una relación entre la potencia de la señal recibida y el tiempo de recepción lo que introduce un error no lineal.

Todas estas fuentes de error descritas, así como otro efectos o propiedades del canal pueden verse afectadas tanto positiva como negativamente por las configuraciones de transmisión y recepción usadas.

La programación del hardware será únicamente de la memoria flash de programa, existe una memoria ROM OTP que puede ser grabada para especificar valores pertinentes a la calibración y otros datos distintivos en cada dispositivo, sin embargo esta memoria no se grabará para permitir su futuro uso realizándose las configuraciones mediante software.

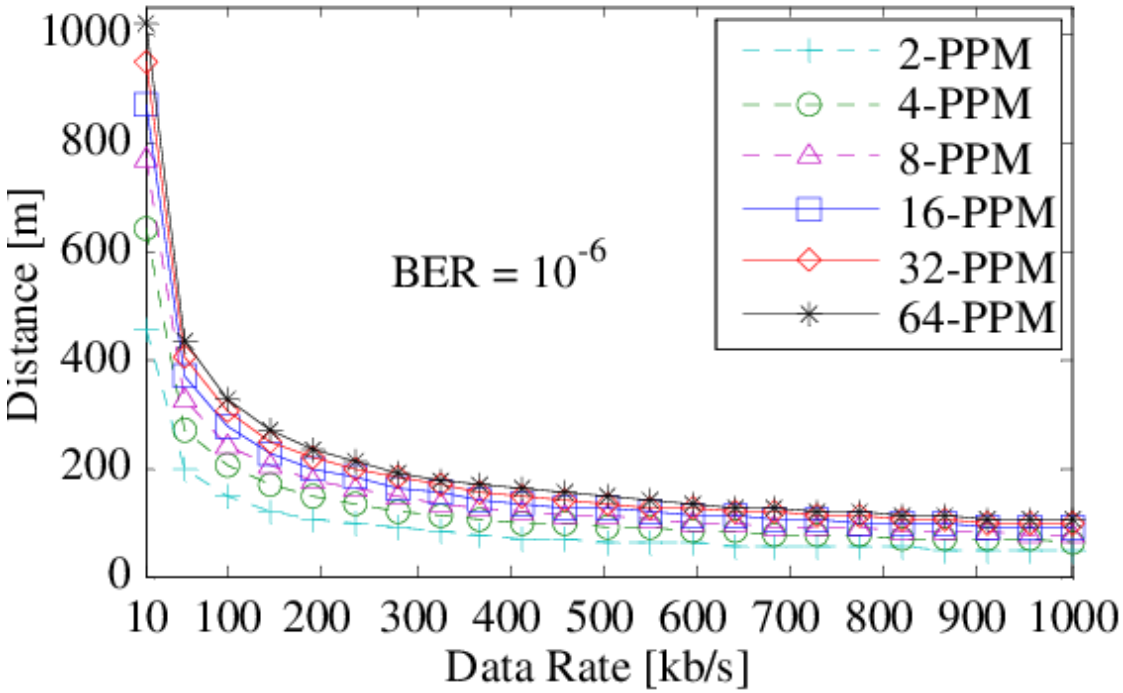

Figura 4.2: Rango máximo frente a la velocidad de transmisión[1]

4.1.3. Comunicaciones

El hardware usado [27] dispone de una serie de funcionalidades que realizan tareas correspondientes a la capa MAC definida en el estándar **IEEE 802.15.4-2011**[2], estas funciones son especificadas en el manual de usuario del DW1000[6] y proporcionan una serie de herramientas para facilitar la implementación software de la capa MAC.

Las funcionalidades de la capa MAC implementadas directamente por el hardware incluyen:

- Generación y comprobado del CRC

- Filtrado de tramas
- Envío de ACK automático
- Recepción de respuesta o ACK automática

El estándar 802.15.4 define la capa física y capa MAC para redes inalámbricas personales de baja velocidad o LR-WPAN, los dispositivos normalmente usados para formar estas redes se basan en microcontroladores de bajo consumo y pueden tratarse desde dispositivos IOT para domótica hasta dispositivos para medida de distancia o *ranging* como los usados en este proyecto.

La capa física y algunas funciones de la capa MAC están implementadas en el hardware, como se ha mencionado anteriormente, debido a esto este estándar solo será relevante para este proyecto en cuanto al formato y uso de las tramas MAC, para asegurar su compatibilidad con la implementación del transceptor DW1000.

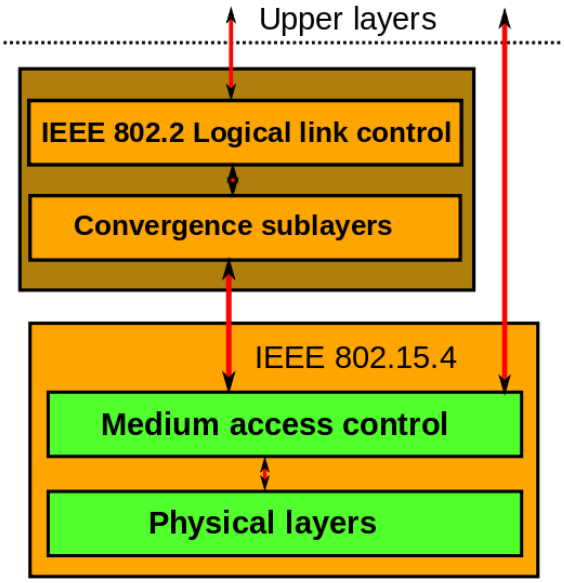

Figura 4.3: Arquitectura por capas del estándar[29]

En cuanto a las características respectivas a capas de red superiores, se pueden considerar los siguientes parámetros para simplificar el problema y delimitar el diseño de la red:

- No existirá comunicación simultanea con más de 10 dispositivos.
- La comunicación será directa, es decir, no existirá comunicación entre dos nodos no conectados.
- Las comunicaciones seguirán una filosofía *best effort* sin garantizar características como la justicia, la fiabilidad o disponibilidad.

4.1.4. Aplicación

La aplicación propuesta, que consiste en el cálculo de posiciones relativas de forma distribuida, posee una serie de características fijas debido a el objetivo del proyecto. Entre las características más relevantes que la aplicación tiene, ya sea de forma natural o por limitaciones impuestas en el proyecto, se encuentran las siguientes:

- Se puede suponer una frecuencia de medida del orden de decenas de medidas por segundo, esta es una variable muy dependiente de la configuración del hardware, la aplicación y las condiciones del entorno y el canal.

  Para simplificar el desarrollo de la aplicación no se fijará ninguna restricción absoluta a la frecuencia de medida

- La aplicación presentará los datos por un puerto serie mediante la interfaz UART de la placa de desarrollo[26]. Los datos, ya sean distancias o posiciones relativas se presentarán como texto para ser usados o procesados por las distintas aplicaciones que los recojan mediante el puerto UART.

- El cálculo de la distancia esta basado en medir el tiempo de vuelo de los mensajes intercambiados entre dos nodos, debido a varios efectos físicos como los mencionados anteriormente y fenómenos como se han comentado anteriormente podemos esperar que la distancia medida sea distinta dependiendo del sentido es decir: $d_{ij} \neq d_{ji}$.

- La representación de las posiciones se realizará en 2 dimensiones como se ha razonado anteriormente, pese a esto el diseño de la aplicación debe permitir y reflejar la representación de posiciones en 3D, ya que esta es la realidad física. Esto facilita futuras expansiones y detecciones de errores o situaciones no contempladas.

- Las pruebas y experimentos se realizarán con 3 nodos, que conforman un plano. Esto no significa que el sistema debe ser diseñado para un número de vecinos menor a 3, se elige este número para simplificar la visualización y realización de los experimentos.

4.2. Deducciones y justificación del diseño

Al conocer los requisitos y suposiciones realizadas se pueden deducir varias conclusiones respecto al sistema en sí, su diseño y su implementación en el hardware proporcionado.

4.2.1. Realidad física y efectos del canal

- Debido a los diversos efectos mencionados existen claras limitaciones cuantificables del hardware, estas se tendrán en cuenta para el diseño del sistema, por ejemplo pueden ser las limitaciones de densidad debidas al protocolo ALOHA, o los errores inevitables de exactitud y precisión.

- También debe considerarse el entorno donde se valide el sistema, debido al gran efecto que puede llegar a tener el canal en las características de las comunicaciones. Por tanto los entornos en los que se realicen pruebas deben componer condiciones favorables e idealmente similares.

4.2.2. Hardware

- Para obtener medidas con una exactitud y precisión representativas se debe realizar una calibración del retardo de la antena, esto permitirá que la noción de distancia obtenida puede llegar a ser más fiel a la distancia real.
- La comunicación con el transceptor deber gestionarse de forma correcta y segura, con el objetivo de evitar errores y posibles fallos de configuración, incluyendo comportamiento no definidos o inesperados.
- La información del fabricante se considera de confianza y por tanto la configuración por defecto puede ser utilizada durante la totalidad de este proyecto.
- Debido al objetivo del proyecto y la naturaleza distribuida de la aplicación se usará un esquema SS-TWR con medidas del TOF, esto se contrapone a los sistemas de TDOA o TOA en los que se usan diferencias entre los tiempos de recepción respecto a varios puntos de referencia realizar la localización. Como es evidente al no contar con ningún sistema de referencia externo este tipo de modalidades quedan fuera del alcance del proyecto.

4.2.3. Comunicaciones

- Debido al concepto de enjambre de robots la red no debe depender de elementos centrales, es decir, ningún nodo de esta debe tener más responsabilidades o mayor importancia que cualquier otro, evitando así la degradación total del sistema frente al fallo de un nodo.
- De la afirmación anterior también podemos deducir que la red debe permitir cualquier comunicación entre cualquier pareja de nodos sin requerir de un nodo intermediario que podría ser un punto de fallo.
- Debido a esto y al objetivo de simplificar el desarrollo no se contará con algoritmos de enrutamiento, es decir dos nodos que no se pueden comunicar directamente, tampoco lo pueden hacer indirectamente a través de otro nodo.
- Con el objetivo de facilitar la escalabilidad y flexibilidad del sistema, los nodos deberán descubrirse e identificarse mutuamente, pudiendo comenzar la comunicación sin necesidad de conocimiento previo de la existencia de sus vecinos.
- El intercambio de mensajes requerido para el cálculo de la distancia mediante TWR depende fuertemente de los tiempos de envío y recepción de estos. Esto provoca un problema a nivel de implementación de la capa de red ya que la respuesta a un mensaje de este tipo debe ser lo más rápida posible para asegurar una medida del tiempo correcta y evitar fuentes de error como las mencionadas anteriormente.

  Esto provoca que se deban tener en consideración dos aspectos importantes de la comunicación entre dos nodos, la disponibilidad y el tiempo de respuesta acotado.

  Para asegurar que el intercambio de mensajes se produce en su totalidad y dentro de los límites temporales, la red debe soportar comunicación orientada a conexión que facilita el consenso entre nodos para asegurar la disponibilidad. A la vez debe permitir la respuesta automática a mensajes de SS-TWR para asegurar su funcionalidad.

- Al tratarse de una red distribuida sin capa de red y debido al alcance del proyecto, el diseño de esta no requiere proporcionar garantías de justicia o fiabilidad, siguiendo una filosofía *best-effort*.
- Al no contar con enrutamiento, podemos asumir que el entorno cercano de un nodo en la red debe ser totalmente conexo, esto presenta un número de comunicaciones posibles que escala de forma cuadrática $\frac{n(n-1)}{2}$, por este motivo se debe limitar el entorno de un nodo a un número de nodos pequeño.
- Al tener un cierto requerimiento de fiabilidad, es un protocolo con ARQ, que asegure la comunicación correcta entre dos nodos. Este sistema es adicional al sistema de detección y corrección de errores FEC mediante CRC implementado por el hardware.

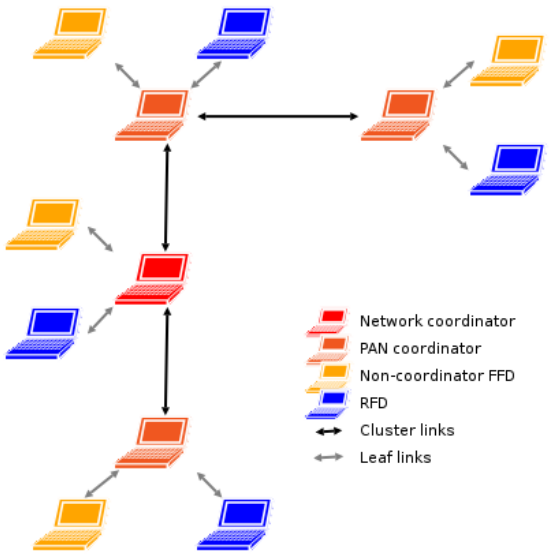

Figura 4.4: Estructura de las PAN[29]

El diseño de las topologías de red contempladas en el estándar **IEEE 802.15.4**[2], conlleva a la dependencia de una serie de elementos centrales o coordinadores de la red como podemos ver en la figura 4.4. Esto es independiente de la topología usada ya que también sucede en la topología *peer-to-peer* contemplada en el estándar.

El objetivo del sistema es ser completamente descentralizado, sin elementos distintos en cuanto a funcionalidad, por lo tanto cumplir con el estándar en estas materias resultaría contraproducente.

4.2.4. Aplicación

- Las medidas de distancia, como consecuencia de las fuentes de error, e inconsistencias propias de cualquier sistema físico se deben someter a algún tipo de filtrado y posprocesamiento estadístico. Este tratamiento se realiza con el objetivo de acercar la medida de distancia a la distancia real para su posterior uso.

Además se debe también tener en cuenta que las medidas pueden tomar distintos valores dependiendo del sentido de la comunicación, esto puede ser útil para algunas aplicaciones o durante el proceso de calibración. Mientras que en otros casos este efecto es una fuente más de error y debe evitarse en la medida de lo posible.

- En cuanto al cálculo de las posiciones de los dispositivos con distancias conocidas, estás serán relativas al dispositivo que realiza el cálculo, de forma que se situará en el origen con los demás agentes en el plano 2D, fijando uno de los agentes al eje de abscisas, eliminando así un grado de libertad y pudiendo determinar unas coordenadas fijas.
- Como método de cálculo de la posición se utilizará la trilateración, esta técnica se basa en las distancias que forman las aristas de los triángulos formados por los dispositivos en el plano. Frente a la triangulación, una opción mas popular, esta técnica es más adecuada en este caso debido a las distancias son obtenidas de forma directa, evitando el procesamiento adicional que sería necesario para operar con ángulos.

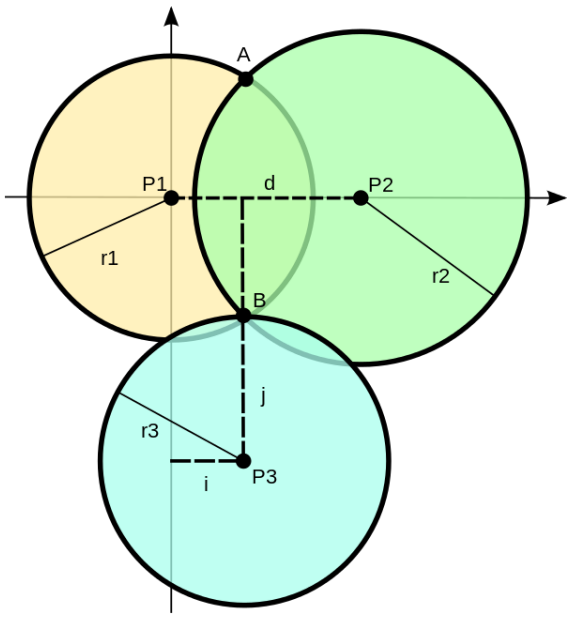

Figura 4.5: Trilateración en dos dimensiones[11]

Basándose en los distintos módulos identificados durante el análisis, una arquitectura basada en capas proporcionaría un número de ventajas al diseño del sistema, las cuáles se detallaran en los siguientes capítulos.

Capítulo 5

Diseño y arquitectura del sistema

Los principales conceptos del diseño del proyecto son una arquitectura por capas y un uso de un **RTOS** basado en **C** en el microcontrolador de los dispositivos.

5.1. Arquitectura por capas

La arquitectura general del sistema se puede describir como una arquitectura por capas, este diseño permite desacoplar los diferentes elementos del sistema de una forma lógica, dejando lugar a su desarrollo aislado. Esto facilita tanto el diseño como la implementación de los distintos módulos del sistema.

Además de esto permite definir más fácilmente los requisitos e interfaces de cada capa o módulo, permitiendo una mayor descriptividad y modularidad del sistema especificando las interfaces de este fuertemente.

Se pueden resumir las distintas capas del sistema con el siguiente esquema en el que se ilustran las interacciones entre las capas, debido a las funcionalidades específicas del hardware, es conveniente no cumplir el requisito de tener interfaces solo con la capa superior y capa inferior.

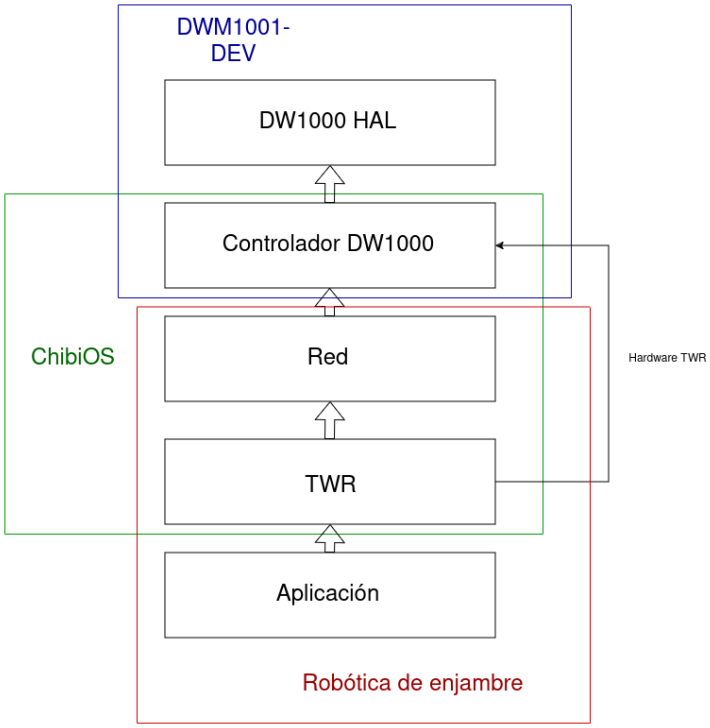

Figura 5.1: Arquitectura por capas del proyecto

Como ejemplo podemos ver que la capa TWR(Two-way ranging) donde se encuentra la lógica correspondiente a la realización de medidas interacciona de manera directa con el controlador para obtener información pertinente a la medida de los tiempos de los mensajes y otra información del hardware. Esto es necesario para la realización de los cálculos necesarios y no incumbe a la capa de Red.

5.1.1. Capa del Controlador

Toda la responsabilidad de interactuar con el hardware de forma directa recae sobre esta capa del sistema, esta se puede dividir en dos estructuras las definiciones y estructuras necesarias para interactuar con el dispositivo, y por otro lado la interfaz y funcionalidades proporcionadas a las capas superiores.

Como se puede apreciar en el esquema la biblioteca DW1000 HAL, contiene las definiciones de constantes, configuraciones válidas, representación de estructuras del hardware y otros elementos necesarios para abstraer la interacción directa con el dispositivo y simplificar la implementación de la lógica del controlador.

La lógica del controlador y su interfaz se representa como una hebra en ChibiOS, a la que se realizan peticiones. El alcance del controlador es básico como se ha descrito en el análisis por lo que solo se contemplan operaciones básica de comunicación.

5.1.2. Capa de Red

La capa de red o comunicaciones se ocupa de especificar el protocolo de comunicación, así como definir la estructuras necesarias para modelar su funcionamiento. Siguiendo las deducciones y limitaciones indicadas en el análisis se puede llegar a los siguientes adjetivos para describir la red diseñada:

- Distribuida
- Dinámica
- Descentralizada
- Homogénea
- *Peer-to-peer*
- A nivel de enlace de datos
- Topología totalmente conexa en el entorno cercano
- Comunicación orientada a conexión
- Salto a salto (sin enrutamiento)
- Direccionamiento estático

Para mantener el funcionamiento de la red se depende de varias estructuras que contengan la información pertinente a las comunicaciones, estas estructuras entre otras deben contener:

- La conexión con otro dispositivo, que mantendrá alguna forma de identificación, además de las variables necesarias para modelar el estado de la conexión.
- La cabecera de los mensajes, tanto en capa MAC como para capas superiores, donde se debe indicar el tipo de mensaje otra información relevante.
- Información propia de la aplicación que concierne o es utilizada por la red, como puede ser el estado de una medida de distancia.

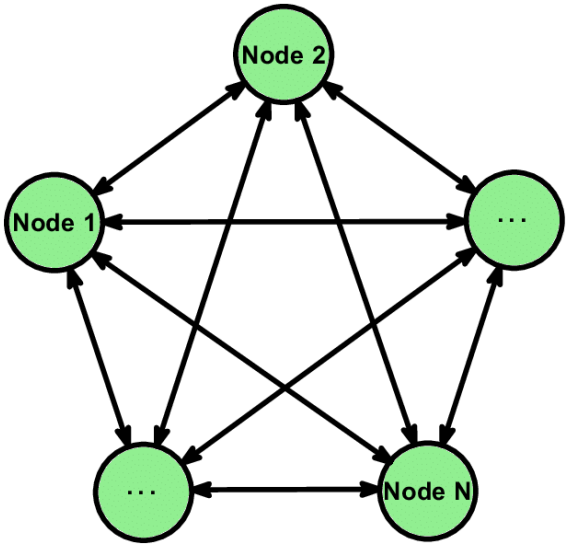

Figura 5.2: La topología de red es totalmente conexa[8] 4.2.3

5.1.3. Capa de Aplicación

En la capa de aplicación se deben mantener dos estructuras principales, una responsable de representar las posiciones relativas de los nodos del entorno y otra representando todas las distancias posibles entre los nodos. Ambas estructuras dependen del número de nodos que se quieran considerar para el entorno y por tanto su huella en memoria también depende de este parámetro, en el caso de las distancias de forma cuadrática, como se describe en el apartado 4.2.3.

Para seguir con el objetivo del proyecto, todos los nodos contarán con estas estructuras para sus respectivos vecinos, esto genera redundancia de la información lo que hace el sistema más resistente a fallo y permite un sistema distribuido, al cualquier nodo poseer un mapa completo de su posición relativa.

Una desventaja clara de esto es el mayor consumo de memoria frente a un sistema centralizado, aunque este es un compromiso necesario para cumplir con las propiedades de homogeneidad y descentralización del sistema.

A pesar de la delimitación de este trabajo a 2 dimensiones, las estructuras que describen las posiciones relativas cuentan con 3 coordenadas para permitir una futura expansión de la funcionalidad.

5.2. Metodología

En esta sección se encuentran los patrones de diseño y metodología usada para la construcción del sistema y sus componentes.

5.2.1. Capa del Controlador

La capa más baja de este sistema se ocupa de interactuar con los dispositivos periféricos al microcontrolador directamente, para el dispositivo usado, el transceptor DW1000, como muchos otros periféricos, permite su uso mediante el acceso a registros.

La comunicación con el DW1000 se produce mediante un bus SPI, que detallaremos posteriormente, esta comunicación simplemente nos proporciona acceso para leer y escribir en el mapa de registros del dispositivo. Mediante la modificación de estos, se puede instruir al transceptor para realizar cualquiera de sus funciones o configurar sus parámetros.

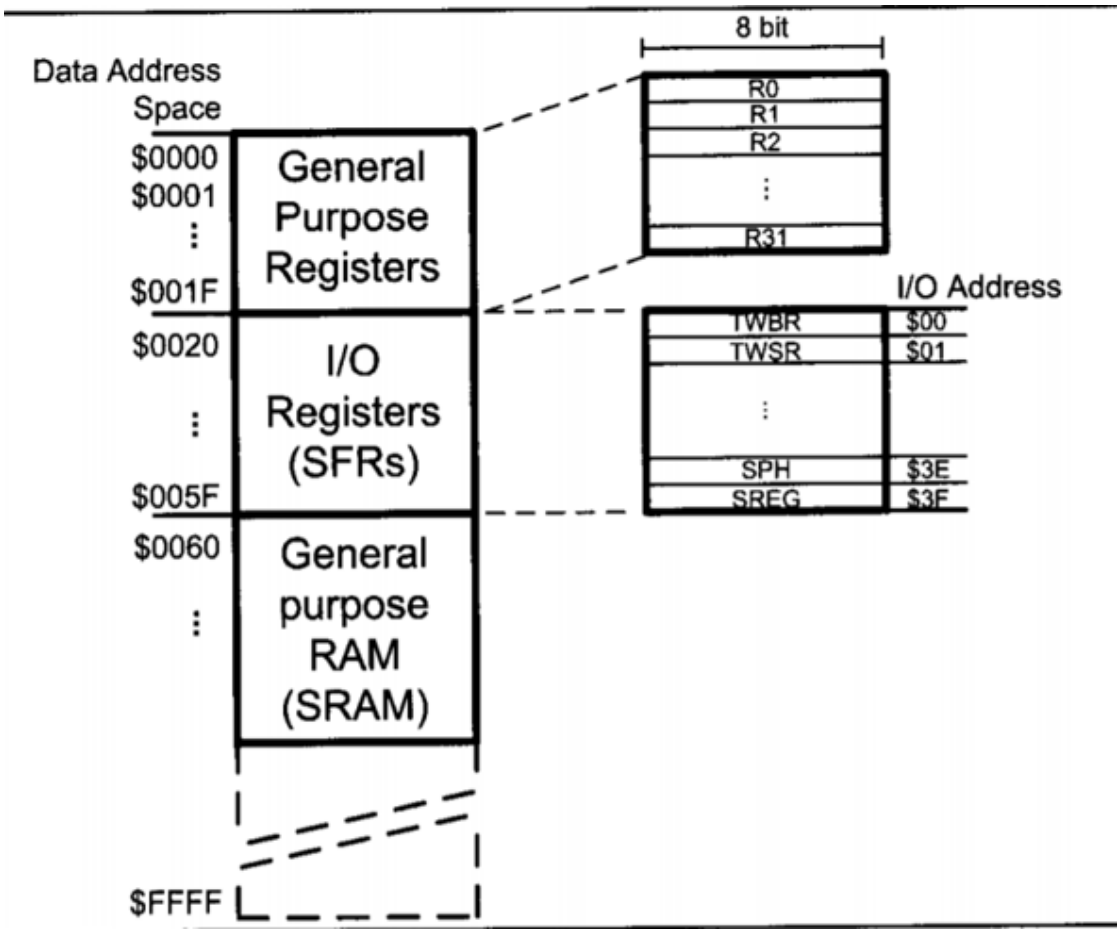

Figura 5.3: Ejemplo de un mapa de memoria de un microcontrolador con registros mapeados a memoria[3]

Este mapa de registros o registros individuales pueden mapearse a la memoria del microcontrolador para así facilitar su modificación. La biblioteca desarrollada define un tipo de dato para representar cada registro del dispositivo, este tipo de dato es una estructura que posee la misma huella en memoria que el registro del dispositivo.

Esto permite trabajar con los registros del dispositivo en memoria de manera sencilla usando los propios tipos de dato definidos para estos y a la vez comunicar estas modificaciones al dispositivo simplemente indicando la zona de memoria donde se encuentra la estructura.

Como se ha mencionado para leer o escribir estos registros del dispositivo a la memoria del procesador y viceversa se hace uso de un bus SPI, como se muestra en la figura 5.4,

la comunicación se basa en indicar la operación: lectura o escritura, y el registro con el que se desea operar, seguido del propio contenido del registro.

Gracias al diseño la comunicación se simplifica en gran medida, ya que al definir el tipo de dato representando cada registro, el formato del registro y su estructura en memoria se organizan correctamente y de forma automática, esto permite obtener el contenido del registro simplemente desde la dirección de memoria de su estructura sin ningún procesamiento adicional.

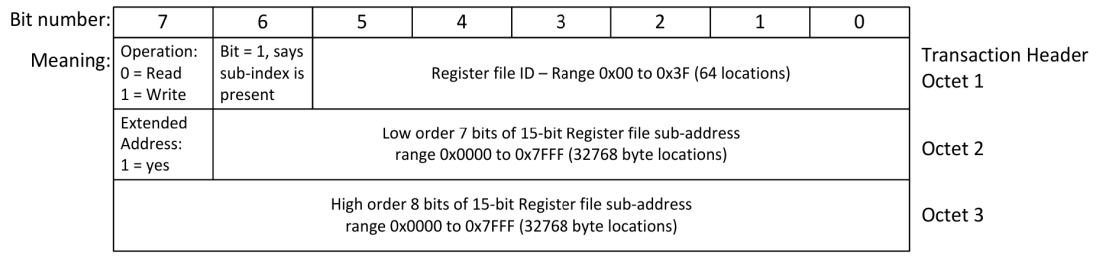

Figura 5.4: Formato de la cabecera SPI para la comunicación con el DW1000[6]

El HAL proporcionado por ChibiOS presenta una interfaz para interactuar con el bus SPI del microcontrolador[18], no obstante usar estas funciones directamente en la implementación del controlador provocaría una dependencia prácticamente total con el sistema operativo.

Por esto se usa un sistema de *wrappers* que definen las funciones necesarias para la comunicación con el dispositivo por el bus y a la vez pueden ser configuradas para llamar a las funciones proporcionadas por ChibiOS o cualquier otra, permitiendo al controlador ser usado con cualquier tipo de procesador anfitrión y sistema operativo.

```
1 void dw_set_spi_set_cs(void (*spi_set_cs_func)(void))
2 {
3     _dw_spi_hal_set._dw_spi_set_cs = spi_set_cs_func;
4 }
5
6 void dw_set_spi_send(void (*spi_send_func)(size_t, const uint8_t*))
7 {
8     _dw_spi_hal_set._dw_spi_send = spi_send_func;
9 }
```

Figura 5.5: Implementación de funciones wrapper para comunicación SPI

La interfaz que proporciona el controlador a las capas superiores contiene las funciones básicas para el uso del dispositivo, entre las más importantes se encuentran:

- Lectura de un registro
- Escritura de un registro

- Inicio de la transmisión
- Inicio de la recepción
- Funciones de configuración

Para gestionar el envío y recepción de mensajes se proporciona un sistema de manejo de **interrupciones**, la propia biblioteca es agnóstica del sistema operativo o microcontrolador en la que se ejecuta por lo tanto depende de este para la gestión de las interrupciones generadas por el dispositivo.

No obstante la biblioteca proporciona un vector de interrupciones o IVT y un manejador para que los usuarios puedan abstraerse de la configuración del tratamiento de interrupciones del hardware.

El vector de interrupciones contiene los manejadores para cada tipo de interrupción que puede generar el hardware, estos manejadores pueden ser configurados por el usuario de la biblioteca.

```
typedef struct dw_irq_vector
{
    union
    {
        struct
        {
            void (*_reserved_irq0)(void);
            void (*_dw_CPLOCK_handler)(void);
            void (*_dw_ESYNCR_handler)(void);
            void (*_dw_AAT_handler)(void);
            void (*_dw_TXFRB_handler)(void);
            void (*_dw_TXPRS_handler)(void);
            void (*_dw_TXPHS_handler)(void);
            void (*_dw_TXFRS_handler)(void);
            void (*_dw_RXPRD_handler)(void);
            void (*_dw_RXFSDD_handler)(void);
            void (*_dw_LDEDONE_handler)(void);
            void (*_dw_RXPHD_handler)(void);
            void (*_dw_RXPHE_handler)(void);
            void (*_dw_RXDFR_handler)(void);
            void (*_dw_RXFCG_handler)(void);
            void (*_dw_RXFCE_handler)(void);
            void (*_dw_RXRFSL_handler)(void);
            void (*_dw_RXRFTO_handler)(void);
            void (*_dw_LDEERR_handler)(void);
            void (*_reserved_irq19)(void);
            void (*_dw_RXOVRR_handler)(void);
            void (*_dw_RXPTO_handler)(void);
            void (*_dw_GPIOIRQ_handler)(void);
            void (*_dw_SLP2INIT_handler)(void);
            void (*_dw_RFPLL_LL_handler)(void);
            void (*_dw_CLKPLL_LL_handler)(void);
            void (*_dw_RXSFDTO_handler)(void);
            void (*_dw_HPDWARN_handler)(void);
            void (*_dw_TXBERR_handler)(void);
            void (*_dw_AFFREJ_handler)(void);
            void (*_reserved_irq30)(void);
            void (*_reserved_irq31)(void);
        };
        void (*vector[32])(void);
    };
} irq_vector_t;
```

Figura 5.6: Vector de interrupciones implementado para DW1000

Un aspecto importante del diseño de las funciones es el concepto de *thread safety*, esta propiedad indica que una función puede ser llamada por varias hebras sin provocar fallos

o condiciones de carrera. Al implementarse el controlador en un sistema basado en un microcontrolador mononúcleo se podría concluir que este problema puede solucionarse fácilmente usando una única hebra que sea responsable del controlador.

Esta suposición no es errónea y de hecho la lógica del controlador se implementará en una hebra aislada, sin embargo, se debe tener en cuenta el uso de las interrupciones. Una interrupción tendrá siempre mayor prioridad que cualquier hebra del sistema y por tanto expulsará del procesador a cualquier ejecución de una función del controlador para dar servicio a la interrupción.

Esto presenta un problema ya que para dar servicio a la interrupción y reconocer esta se debe interactuar con el dispositivo mediante SPI, por lo que podría generar una condición de carrera si la interrupción se produce en medio de una lectura o escritura.

Para evitar esta situación se usa la funcionalidad de exclusión mutua ya implementada por ChibiOS[32] para el bus SPI, evitando condiciones de carrera en cualquier caso.

La lógica del controlador se encuentra en una hebra estática, para interactuar con el dispositivo desde otras hebras, se usan un sistema de señalado entre hebras basado en eventos del sistema operativo[17]. Estos son eventos asíncronos de forma que cuando una hebra desea que el controlador realice cualquier acción señala a la hebra del controlador, cuando esta hebra deja de ejecutarse, la hebra del controlador le da servicio pasando la orden al dispositivo. El paso de datos entre hebras se basa en variables compartidas para ofrecer el máximo rendimiento y simplicidad.

5.2.2. Capa de Red

Al contar con una implementación de funciones MAC 4.1.3 se debe cumplir de forma obligatoria con el formato de trama MAC definido en el estándar usado[2]. De esta forma el hardware será capaz de interpretar la cabecera o MHR y realizar por el ejemplo el filtrado de tramas al interpretar la dirección de destino.

Como se ha mencionado anteriormente y se aprecia en la figura 4.4 este estándar no será seguido para los demás aspectos del sistema, ya que es incompatible con las aplicaciones esperadas y la naturaleza distribuida de este.

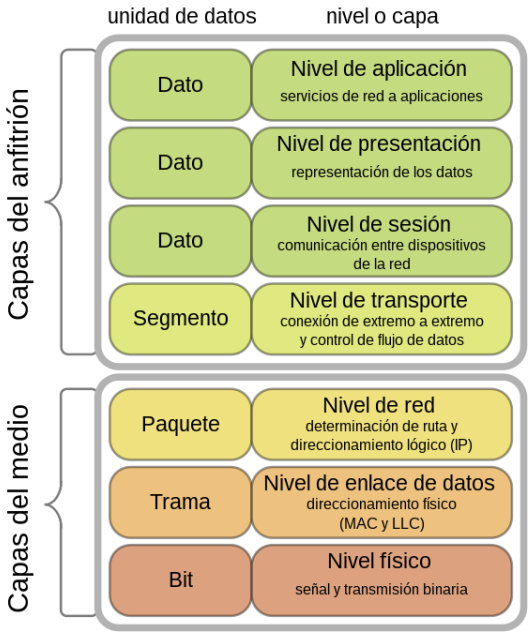

Figura 5.7: Capas de red del modelo OSI[12]

El diseño de la red se concentra en la capa de enlace de datos del modelo OSI, las funciones de acceso al medio están mayormente cubiertas por el hardware o definidas en el estándar IEEE 802.15.4. Esto supone que el diseño se centrará en el control del enlace lógico o LLC.

Esta subcapa se ocupa del control de flujo, corrección de errores, y la multiplexación de los protocolos en capas superiores.

Esta decisión se debe a que las comunicaciones son punto a punto y la realización de TWR requiere una cierta fiabilidad de las comunicaciones. Estas afirmaciones describen el ámbito y responsabilidad de la capa LLC.

El control de flujo y errores se realizará mediante ARQ, debido a que la cantidad de datos transmitida será baja en este tipo de dispositivos y aplicaciones, se usará parada y espera. Esta decisión implica una solución mucho más simple y con menor sobrecarga sin un gran sacrificio en la eficiencia de la comunicación frente a otras técnicas de ventana deslizante.

El tamaño de trama será fijo para representar la realidad del hardware y eliminar el problema de la fragmentación de tramas.

Durante un intercambio de mensajes SS-TWR la respuesta del receptor debe ser instantánea y asegurada, para tener una mayor certeza de la recepción de la respuesta es conveniente basar las comunicaciones en un sistema de conexión. Al realizar una conexión cada nodo en la red, que incialmente no tiene ningún conocimiento de los demás, puede asegurarse de que el nodo con el que ha realizado la conexión esta disponible y además es capaz de mantener una comunicación bidireccional.

Para establecer la conexión se usará una técnica de *three-way handshake* como en muchos protocolos a varios niveles de la estructura OSI, esta técnica se basa en el intercambio de tres mensajes especiales después de los cuáles se inicializa la conexión.

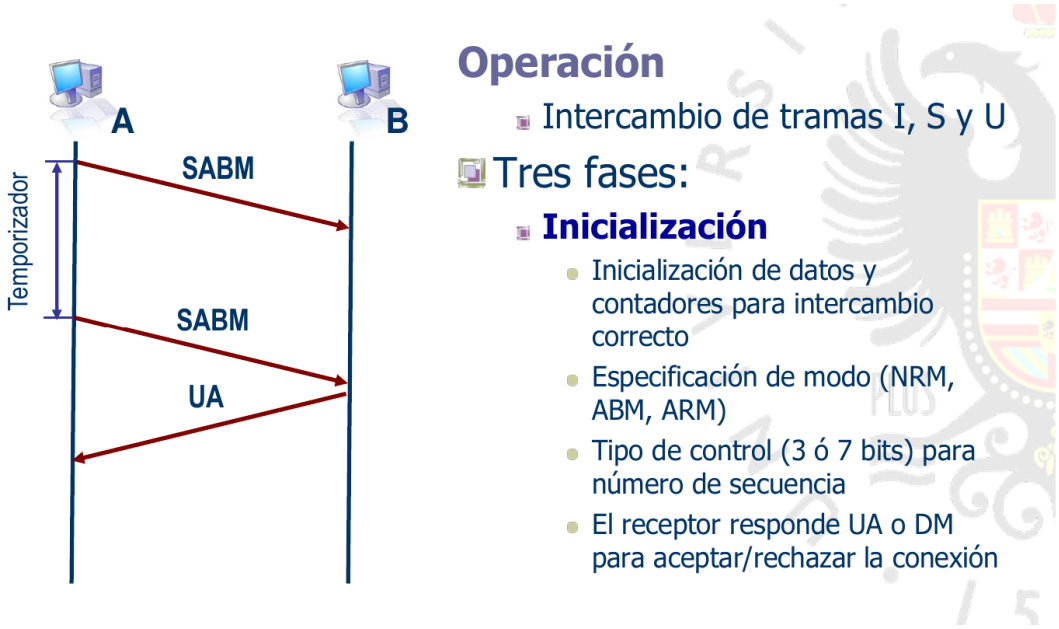

Figura 5.8: Three-way handshake realizado en HDLC, extracto de diapositiva de tecnología de redes impartida por Jesús E. Díaz Verdejo[24]

La red descrita tiene amplias similitudes con redes **Ad hoc**, ya que estas se describen como redes descentralizadas que no dependen de infraestructura externa. No obstante una gran diferencia causa que no se pueda considerar a esta red parte de esta categoría. Las redes Ad hoc se identifican en que los propios nodos finales o *hosts* actúan como *routers* encaminando los paquetes en la red mediante algoritmos de encaminamiento.

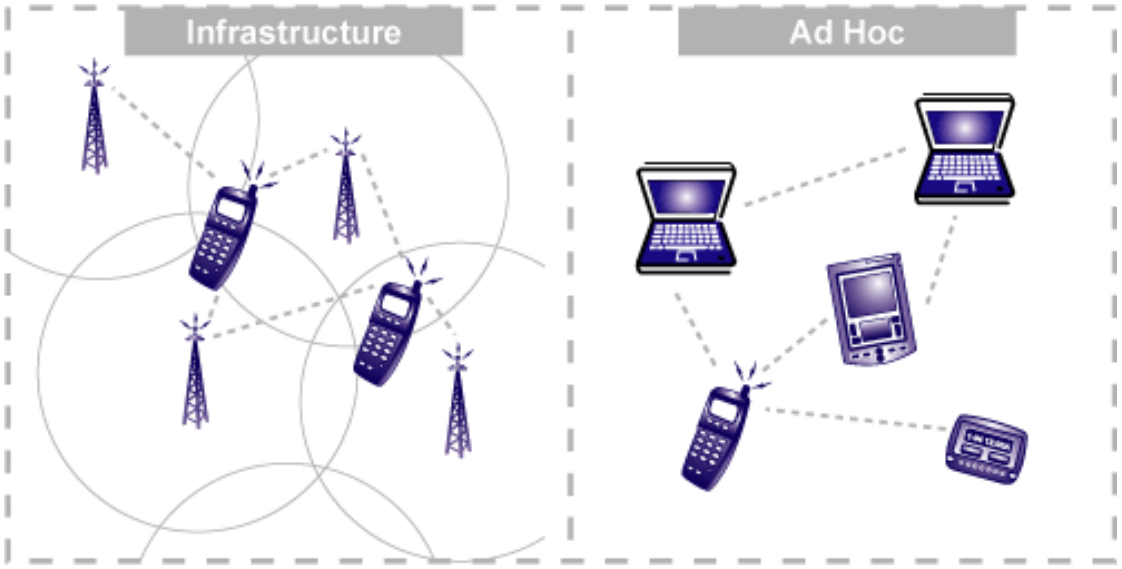

Figura 5.9: Estructura de red Ad Hoc[19]

5.2.3. Capa de Aplicación

Para cualquier uso de las capas inferiores, incluido el cálculo de posiciones relativas en 2D, uno de los aspectos a tener en cuenta es el desacoplamiento de las aplicaciones con las capas inferiores, en concreto con la capa de comunicaciones. En este proyecto se consigue disminuir la dependencia entre capas y el acoplamiento realizando los cálculo de las posiciones relativas en otra hebra, que no realiza comunicación síncrona con las demás.

De esta forma errores en cálculo o la propia medida de distancia no tendrán efecto en el comportamiento de las comunicaciones de forma directa, así como la determinación de la posición contemplará todos los casos posibles en cuanto a datos de distancia, ya que no puede asumir un efecto sobre estos directamente.

En el análisis se llegó a la conclusión de que el esquema de TWR más conveniente para el proyecto es el esquema SS-TWR(Single-sided two-way ranging), este intercambio se basa en dos mensajes en el mensaje de iniciación no se envía información sobre las medidas de tiempo, cuando otro transceptor recibe el mensaje, este extrae el tiempo de recepción y cálculo un tiempo de envío de la respuesta futuro.

Con estos dos tiempos conocidos, se envía el mensaje de respuesta exactamente en el momento de envío precalculado, conteniendo el tiempo de recepción del mensaje de iniciación y el tiempo de envío del mensaje de respuesta. Al recibir el primer nodo el mensaje de respuesta, esta recoge el tiempo de envío del mensaje de iniciación, el que posee ya que fue el emisor, el tiempo de recepción del mensaje de respuesta, que acaba de recibir y por último los dos tiempos descritos anteriormente contenidos en el mensaje de respuesta.

Se puede ver como el intercambio obtiene 4 instantes de tiempo, que se definen como:
$T_{\mathrm{IS}} \in \mathbb{R}^+$, el tiempo de envío del mensaje de iniciación
$T_{\mathrm{IR}} \in \mathbb{R}^+$, el tiempo de recepción del mensaje de iniciación
$T_{\mathrm{RS}} \in \mathbb{R}^+$, el tiempo de envío del mensaje de respuesta
$T_{\mathrm{RR}} \in \mathbb{R}^+$, el tiempo de recepción del mensaje de respuesta

Además:
$T_{\mathrm{round}} = T_{\mathrm{IS}} - T_{\mathrm{IR}}$
$T_{\mathrm{reply}} = T_{\mathrm{RS}} - T_{\mathrm{RR}}$

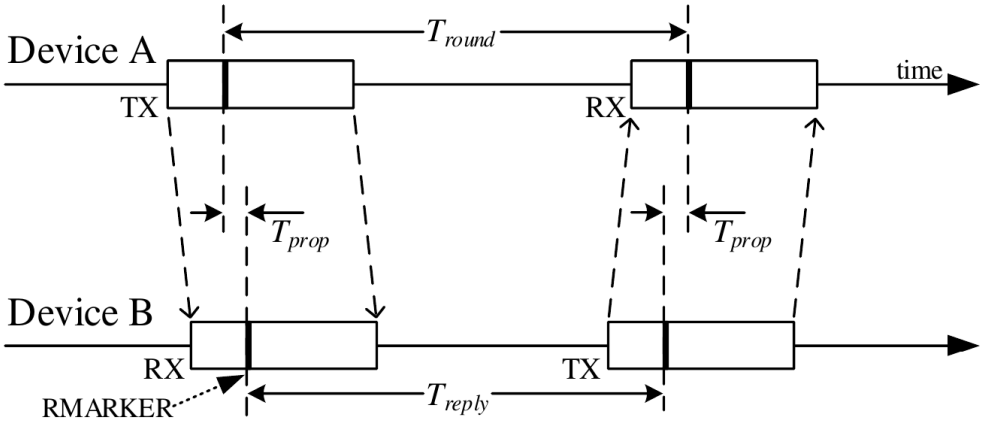

Figura 5.10: Esquema SS-TWR[6]

Con estos cuatro tiempos se puede calcular el tiempo de vuelo o tiempo de propagación de los mensajes, que es proporcional a la distancia:

$$\hat{T}_{\text{prop}} = \frac{1}{2}(T_{\text{round}} - T_{\text{reply}})$$

De esta forma se obtendrá el tiempo de vuelo de un mensaje entre la pareja de dispositivos y de este se deducirá una noción de distancia entre ellos, a partir de los cuatro instantes de tiempo medidos por el hardware.

La distancia D, debido a que la comunicación se basa en señales electromagnéticas viene dada por:

$$D = cT_{tofr}$$

donde c es la velocidad de la luz en el medio, en este caso la atmósfera, y T_{tofr} es el tiempo de vuelo real.

Sin embargo el tiempo de vuelo medido T_{prop} es afectado por los errores descritos, por lo que:

Sea $f : \mathbb{R}^+ \to \mathbb{R}^+$ una función desconocida que modela el error de la medición, entonces:

$$T_{tofr} = f(T_{prop})$$

Por lo que sin determinar la función f de forma exacta no se puede obtener la distancia real, por esto se refiere al valor obtenido como una noción de distancia.

Habiendo obtenido un número sensible a la distancia entre dos dispositivos, se debe interpretar y procesar este dato, debido a fuentes de error y implementaciones de las capas inferiores, este número puede presentarse en un gran rango de posibles valores y distribuciones que no tienen porque reflejar la realidad en todos los casos como se detalla en el análisis 4.2.4.

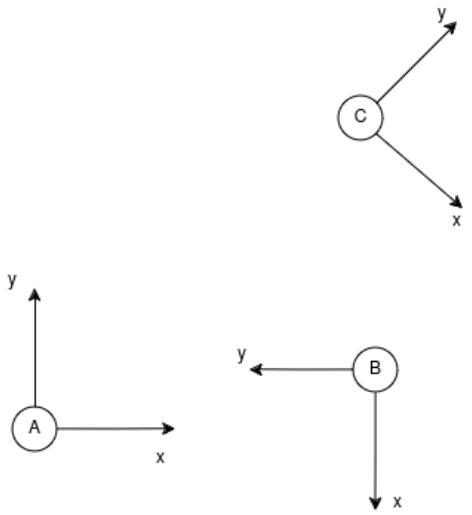

Figura 5.11: Posicionamiento relativo en 2D

Conociendo las distancias medidas entre los dispositivos, se realiza un cálculo de las posiciones de los nodos conocidos en el entorno cercano, estas posiciones son relativas a la posición del propio dispositivo que realiza los cálculos que asumirá como posición el origen en todos los casos. Al asumir un plano como se detalla en el análisis, las coordenadas correspondientes a las tres posiciones se actualizarán mediante un algoritmo de trilateración que se detalla en la implementación.

Como se aprecia en la figura 5.11 desde el punto de vista del dispositivo A, situado en el origen, su sistema de referencia es conocido así como las posiciones de sus vecino. Sin embargo los sistemas de referencia de los demás dispositivos son desconocidos y pueden encontrarse configurados en cualquier disposición posible.

Por tanto desde un dispositivo nunca se podrá asumir que localización perciben los demás, salvo que se realice una comunicación expresa con estos. Este detalle puede parecer una desventaja a simple vista, no obstante el objetivo del proyecto se centra en una organización distribuida en los que los integrantes del sistema son independientes. Esto significa que la percepción de los demás no debe afectar al comportamiento de cualquier integrante, ya que esto aumentaría su dependencia.

De esta forma el hecho de no conocer los sistemas de referencia ajenos de forma nativa no conforma una desventaja, sino que es un resultado lógico del diseño del sistema descentralizado.

Capítulo 6

Implementación del software

Una vez descritos los elementos principales del diseño del sistema, en este capítulo se ilustrará la implementación de estos patrones de diseño y los componentes más importantes del código desarrollado durante el proyecto.

6.1. Resultados técnicos esperados

De la implementación que se desarrollará a continuación se esperan unos ciertos resultados técnicos que sean de utilidad en las aplicaciones mencionadas. Esto resultados serán comparados con los indicadores definidos en la sección 2.4

Una vez realizadas las tareas de análisis y diseño se pueden definir una serie de resultados cuantitativos y cualitativos que la implementación del sistema muestre. Estos resultados técnicos incluyen:

- Una precisión de la distancia medida del orden de **30cm**, este número es una estimación de la posible precisión después de analizar los datos presentados en la documentación del fabricante[9][6][20].
- La exactitud, como se ha detallado en 4.2.3, es altamente dependiente del retardo de la antena, potencia de la señal y otros parámetros de configuración. Al decidir usar una configuración por defecto, no se espera unos valores de distancia con una exactitud definida.

  A pesar de esto, como se mostrará en la validación se realizará una calibración rudimentaria de los retardos de antena para obtener valores representativos de la realidad.
- En específico, las aplicaciones como la robótica móvil requieren una frecuencia de actualizado del posicionamiento elevada, proporcional a las velocidades y capacidades cinéticas de los robots.

  La frecuencia de medida del sistema rondará el orden de **10 medidas por pareja por segundo**, lo que cumple los requisitos de frecuencia de la mayoría de aplicaciones en robótica.
- De forma cualitativa, se espera conseguir una serie de trío de posiciones relativas extraídas de uno de los dispositivos participantes, en la que las posiciones y dis-

tancias representadas se hayan obtenido de forma descentralizada y cuyos valores puedan considerarse usables para una aplicación simple en la robótica móvil.

6.2. Entorno y prerrequisitos

Como es de esperar al realizar el desarrollo en un microcontrolador[36], se usan las herramientas de desarrollo en C para ARM, en concreto el *tool-chain* de GNU.

El depurado del proyecto se basa en dos herramientas principales, GDB[28] y **Cortex-Debug**[13] una interfaz para GDB , disponible para el entorno de desarrollo *Visual Studio Code* de Microsoft.

Al tratarse de un proyecto software de código abierto es indispensable el uso de un sistema de control de versiones, Git es el estándar en la industria de desarrollo. Complementariamente este repositorio de **Git** se hará visible en GitHub[38] de forma que cualquier desarrollador pueda acceder a él de forma sencilla.

ChibiOS proporciona una serie de ejemplos o *demos*, para los distintos microcontroladores, de forma que sirven como plantilla para comenzar el desarrollo. Cabe destacar que previamente al desarrollo de este proyecto el repositorio mencionado ya existía, conteniendo una serie de ejemplos basados en esta plantilla con los que se confirmo el funcionamiento correcto del hardware.

El estado del proyecto cuando comienza el desarrollo puede comprobarse desde la historia disponible, y únicamente implementaba un programa de ejemplo para la placa de desarrollo. Esto es posible debido a que adicionalmente ya se había desarrollado una serie de definiciones que se detallarán en la siguiente sección y permiten la interacción con los elementos de la placa como LEDs o botones.

Al necesitar acceso a diversos periféricos respecto al microcontrolador varias configuraciones de ChibiOS y el proceso de generación del ejecutable necesitan modificaciones, estas modificaciones se realizan en archivos de configuración propios del sistema operativo y están descritas en los distintos tutoriales oficiales[32], así como en el foro para usuarios[15]

6.3. Desarrollo

Durante el curso de esta sección se intenta ilustrar la operación general del sistema detallando el funcionamiento de los componentes de este, mostrando el código que los implementa.

6.3.1. Capa del Controlador

La primera tarea de desarrollo realizada, obviando la instalación de dependencias y pruebas, es la determinación de la configuración adecuada para ChibiOS, así como las definiciones que su HAL usa para identificar la placa de desarrollo y permitir el uso de su interfaz con las características de la placa[26].

Para indicar a ChibiOS en que pines del microcontrolador se encuentran los distintos periféricos se deben definir ficheros específicos para la placa, en el desarrollo se han añadido las definiciones para diversos periféricos como SPI o UART. Estas definiciones

se encuentran en producto separado del sistema operativo en un repositorio mantenido por la comunidad donde se da soporte a un gran rango de placas[37].

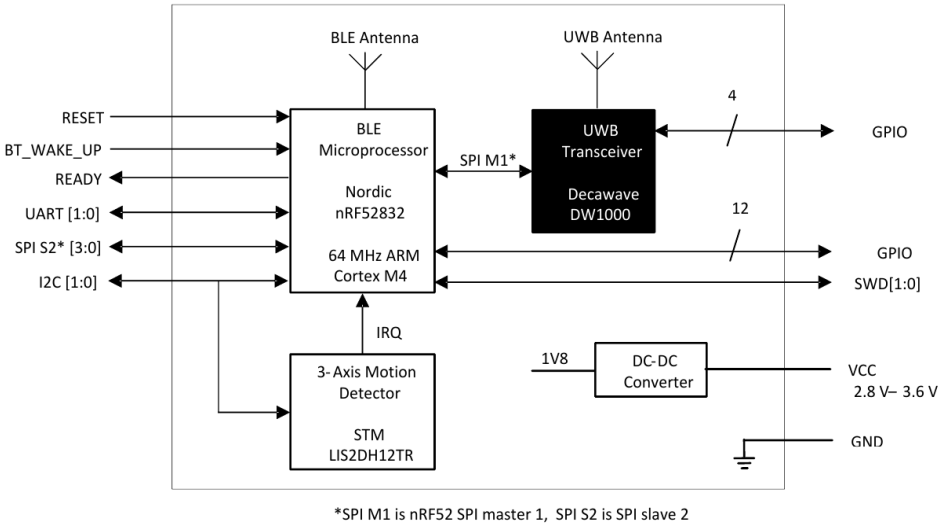

Figura 6.1: Periféricos del microcontrolador en DWM1001[7]

Con la información sobre los pines del microprocesador y los periféricos disponibles[5][4], se ha definido los pines necesarios en la placa para proporcionar al HAL el número de pin correspondiente a cada componente y así poder realizar funciones como leer de este o llevarlo al estado alto.

Una vez definidas las configuraciones necesarias para interactuar con el hardware, cabe crear las estructuras que representan los registros del transceptor [25] tal y como se indicó en el diseño.

En la figura 6.2 se puede observar la definición de un tipo de dato en C para el registro de configuración del dispositivo *sys_cfg*. Con el fin de facilitar tanto el acceso por parte del usuario como ser fiel a la representación real en memoria, se implementa con una unión, esto permite mantener el registro en memoria con la misma huella que tiene en el dispositivo, además dejando al usuario de la biblioteca acceder a los distintos bits de configuración de forma aislada, así como a la máscara total.

Esta solución se implementa para la mayoría de registros y consigue una configurabilidad total sin necesidad de conocer las máscaras de bits y a la vez expone la máxima eficiencia tanto en espacio como en procesamiento, ya que las tareas de estructuración de la memoria recaen en el compilador.

```
typedef struct dw_sys_cfg
{
    union
    {
        struct
        {
            uint32_t FFEN       :1;
            uint32_t FFBC       :1;
            uint32_t FFAB       :1;
            uint32_t FFAD       :1;
            uint32_t FFAA       :1;
            uint32_t FFAM       :1;
            uint32_t FFAR       :1;
            uint32_t FFA4       :1;
            uint32_t FFA5       :1;
            uint32_t HIRQ_POL   :1;
            uint32_t SPI_EDGE   :1;
            uint32_t DIS_FCE    :1;
            uint32_t DIS_DRXB   :1;
            uint32_t DIS_PHE    :1;
            uint32_t DIS_RSDE   :1;
            uint32_t FCS_INT2F  :1;
            uint32_t PHR_MODE   :2;
            uint32_t DIS_STXP   :1;
            uint32_t            :3;
            uint32_t RXM110K    :1;
            uint32_t            :5;
            uint32_t RXWTOE     :1;
            uint32_t RXAUTR     :1;
            uint32_t AUTOACK    :1;
            uint32_t AACKPEND   :1;
        };
        uint8_t reg[4];
        uint32_t mask;
    };
} sys_cfg_t;
```

Figura 6.2: Representación de un registro mediante una estructura

Para realizar la operaciones con estos registros se necesitan comunicaciones SPI con el dispositivo, esta implementación usa las funciones síncronas proporcionadas por el controlador incluido en ChibiOS. No obstante, al dar posibilidad en la biblioteca de elegir cualquier sistema operativo, estas funciones se llaman mediante una serie de *wrappers* que son usado por las funciones internas de la biblioteca.

```
typedef struct dw_spi_hal
{
    void (*_dw_spi_lock)(void);
    void (*_dw_spi_unlock)(void);
    void (*_dw_spi_set_cs)(void);
    void (*_dw_spi_clear_cs)(void);
    void (*_dw_spi_send)(size_t, const uint8_t*);
    void (*_dw_spi_recv)(size_t, const uint8_t*);
} spi_hal_t;
```

Figura 6.3: Abstracción de la comunicación mediante SPI

De esta forma se consigue estandarizar la comunicación con el dispositivo y aumentar la compatibilidad con distintos sistemas, incluso con distintos microcontroladores.

Las interrupciones generadas por el dispositivo son gestionadas por una hebra especializada, esta hebra se encarga de preparar el sistema para el servicio de la interrupción y llamar al manejador de interrupciones.

Como se aprecia en la figura 6.4 lo primero que se asegura antes de interactuar con el dispositivo es la exclusión mutua, se detecta la fuente de interrupción desde el registro de estado de dispositivo y se llama al correspondiente manejador.

Estos manejadores son configurables y se puede cambiar para ajustarse a los deseos del usuario, en este caso además de dar servicio a la interrupción correspondiente, los manejadores señalan a la hebra del controlador para avisar de la interrupción recibida.

```
void _dw_irq_handler(void)
{
    sys_status_t current_status;
    current_status.mask = 1;

    sys_mask_t current_mask;

    _dw_spi_hal_set._dw_spi_lock();
    _dw_spi_transaction(1, DW_REG_INFO.SYS_MASK.id, current_mask.reg,
                        DW_REG_INFO.SYS_MASK.size, 0);

    while (current_status.mask)
    {
        _dw_spi_transaction(1, DW_REG_INFO.SYS_STATUS.id, current_status.reg,
                            DW_REG_INFO.SYS_STATUS.size, 0);

        current_status.mask &= current_mask.mask;

        // Clearing reserved bits to avoid calling a reserved address
        current_status.mask &= 0x3FF7FFFE;

        uint8_t pos = __builtin_ffs(current_status.mask);

        if (pos)
        {
            irq_vector.vector[pos-1]();

            sys_status_t handled_mask;
            handled_mask.mask = 1 << (pos-1);
            // Disable unsupported interrupts
            handled_mask.mask &= 0x376FFFFF;
            _dw_spi_transaction(0, DW_REG_INFO.SYS_STATUS.id, handled_mask.reg,
                                DW_REG_INFO.SYS_STATUS.size, 0);
        }
    }
    _dw_spi_hal_set._dw_spi_unlock();
}
```

Figura 6.4: Manejador de interrupciones para DW1000

Al contar con tipos de datos para cada registro la realización de la transacción SPI solo copia de la dirección de memoria pertinente al buffer del envío y viceversa. En la figura 6.5 se procesa la cabecera SPI 5.4 en la que simplemente se indica la id del registro, y posteriormente según la operación de lectura o escritura se lee o escribe del buffer y se llama al wrapper mencionado para SPI.

```
void _dw_spi_transaction(uint8_t is_read_op,  uint8_t reg_id, uint8_t* buf,
                         size_t count, uint16_t offset)
{
    spi_header_t spi_header = _build_header(is_read_op, reg_id, offset);

    _dw_spi_hal_set._dw_spi_set_cs();

    if (is_read_op)
    {
        _dw_spi_hal_set._dw_spi_send(spi_header.size,
                                     (uint8_t*)&(spi_header.header));
        _dw_spi_hal_set._dw_spi_recv(count, buf);
    }
    else
    {
        size_t spi_trans_size = spi_header.size + count;
        uint8_t spi_transaction[spi_trans_size];
        memcpy(spi_transaction, (uint8_t*)&(spi_header.header), spi_header.size);
        memcpy(spi_transaction + spi_header.size, buf, count);
        _dw_spi_hal_set._dw_spi_send(spi_trans_size, spi_transaction);
    }

    _dw_spi_hal_set._dw_spi_clear_cs();
}
```

Figura 6.5: Transacción SPI

Esta función no garantiza la exclusión mutua ya que esta se gestiona en las funciones *dw_read* y *dw_write*, ya que esta es la interfaz que debe ser usada por el usuario, mientras que la propia función se llama de forma interna.

La funcionalidad más importante para un transceptor es el envío y recepción, siguiendo el diseño se implementan dos funciones asíncronas que se encargan de habilitar el transmisor o el receptor y por tanto enviar o recibir un mensaje.

En la figura 6.6 se muestra la función para el envío de mensajes, se puede apreciar como recibe como parámetro un registro de configuración, así como la dirección al buffer de envío y los registros de envío retardado y espera de respuesta.

Esta función escribe en el dispositivo el registro de configuración y el buffer de envió y posteriormente gestiona las funciones especiales usadas, por ejemplo, en el intercambio SS-TWR. Permitiendo configurar el tiempo de envío y la espera de la respuesta en el momento de activar el transmisor, evitando un uso incorrecto de configuraciones anteriores.

```
void dw_start_tx(tx_fctrl_t tx_fctrl, uint8_t* tx_buf, dx_time_t dly_time,
                 ack_resp_t_t w4r_time)
{
    _dw_spi_hal_set._dw_spi_lock();
    validate_spi_transaction(DW_REG_INFO.TX_BUFFER, tx_fctrl.TFLEN, 0);
    _dw_spi_transaction(0, DW_REG_INFO.TX_BUFFER.id, tx_buf, tx_fctrl.TFLEN, 0);

    _dw_spi_transaction(0, DW_REG_INFO.TX_FCTRL.id, tx_fctrl.reg,
                        DW_REG_INFO.TX_FCTRL.size, 0);

    sys_ctrl_t ctrl_tx_start;
    ctrl_tx_start.mask = 0;
    ctrl_tx_start.TXSTRT = 1;

    if(dly_time.time32)
    {
        ctrl_tx_start.TXDLYS = 1;
        _dw_spi_transaction(0, DW_REG_INFO.DX_TIME.id, dly_time.reg,
                            DW_REG_INFO.DX_TIME.size, 0);
    }

    if (w4r_time.ACK_TIM)
    {
        ctrl_tx_start.WAIT4RESP = 1;
        w4r_time.reg[2] &= 0x0F; // Reserved bits must be written as 0
        _dw_spi_transaction(0, DW_REG_INFO.ACK_RESP_T.id, w4r_time.reg, 3, 0);
    }

    _dw_spi_transaction(0, DW_REG_INFO.SYS_CTRL.id, ctrl_tx_start.reg,
                        DW_REG_INFO.SYS_CTRL.size, 0);
    _dw_spi_hal_set._dw_spi_unlock();
}
```

Figura 6.6: Función para comenzar la transmisión

La biblioteca implementada es usada por una hebra controlador, que se ocupa de toda la interacción con el dispositivo y cuya comunicación con las demás hebras se basa en eventos de ChibiOS[16]. En la figura 6.7 se muestra un extracto del código de esta hebra, en el que se puede ver una implementación mediante un *switch* de una máquina de estados que gestiona las distintas acciones realizables. Entre las cuáles se encuentran:

- Recepción
- Envío
- Envío y espera de la respuesta
- Envío retardado

```
THD_FUNCTION(DW_CONTROLLER, arg)
{
    (void)arg;
    dw_setup();
    dx_time_t dx_time;
    tx_fctrl_t tx_ctrl;
    ack_resp_t_t w4r;
    eventmask_t evt = 0;
    // Inicialización de variables omitida
    chMtxObjectInit(&dw_mutex);
    dw_thread = chThdGetSelfX();
    set_irq_vector();
    chMtxLock(&dw_mutex);

    barrier();

    while (true)
    {
        tx_ctrl.TFLEN = send_size+2;
        w4r.mask = 0;
        memset(dx_time.reg, 0, sizeof(dx_time.reg));

        current_state = dw_ctrl_req;

        switch (dw_ctrl_req)
        {
            case DW_RECV:
                dw_start_rx(dx_time);
                evt = chEvtWaitOneTimeout(MRXFCG_E | MRXERR_E,
                                          TIME_US2I(recv_tmo_usec));
                if (evt == MRXFCG_E)
                    dw_ctrl_req = DW_CTRL_YIELD;
                else if (evt == 0)
                    dw_ctrl_req = DW_RECV_TMO;
                else
                {
                    err_cnt++;
                    dw_soft_reset_rx();
                }
                break;
            case DW_SEND:
                dw_start_tx(tx_ctrl, send_buf, dx_time, w4r);
                evt = chEvtWaitOneTimeout(MTXFRS_E, TX_TIMEOUT);
                if (evt == MTXFRS_E)
                    dw_ctrl_req = DW_CTRL_YIELD;
                else
                    dw_ctrl_req = DW_TRX_ERR;
                clean_send();
                break;
```

Figura 6.7: Implementación del controlador en ChibiOS usando el HAL

El controlador se encuentra en estado ocioso hasta que la hebra de comunicaciones lo señala mediante un evento del sistema operativo, en ese momento realiza la acción indicada, ordenando al dispositivo mediante la biblioteca desarrollada y vuelve a devolver el control al planificador del sistema operativo.

6.3.2. Capa de Red

El nivel más bajo de implementación en las comunicaciones consiste en definir una cabecera MAC que cumpla con el estándar[2]. Para esto se ha optado por una cabecera de longitud fija para todas las comunicaciones, con direccionamiento de 16 bits y sin implementaciones de procesamiento adicional para la seguridad.

La implementación no puede seguir el estándar muchos más allá de esta convención de trama MAC, como se expuso en el análisis 4.2.3 la misma topología centralizada impide que se aplique este diseño al sistema distribuido que se tiene como objetivo.

Ya que la mayoría de responsabilidades MAC están gestionadas por el hardware, la implementación se centra en la capa de enlace lógica o LLC, se disciernen dos conceptos principales para la implementación de las funciones de esta capa:

- **Comunicación orientada a conexión**, este concepto es el que permite representar en la red las parejas de dispositivos con conexión directa necesarias para la realización de medidas mediante SS-TWR. La conexión se gestiona mediante la estructura mostrada en la figura 6.8 donde se definen todos los elementos necesarios para mantener una conexión con un determinado vecino. Estos elementos son:

  - La dirección MAC de la pareja
  - El número de secuencia y ACK de la conexión
  - El *time-to-live*, que representa el número de mensajes fallidos
  - Un temporizador para desconectar a la pareja cuando esta no da respuesta
  - Las estructuras necesarias para gestionar errores y reenvíos

  Como se describe en el diseño la conexión se realiza e inicializa mediante un *three-way-handshake*, después del cuál se espera la comunicación fiable entre los dispositivos.

- **ARQ** Como se sistema de control de errores y flujo se implementa parada y espera, este mecanismo se basa en que el número de secuencia y ACK tiene un tamaño de un bit, por tanto solo se puede enviar un mensaje una vez se ha recibido confirmación del anterior. Esta implementación no es la más eficiente como se detalla en el diseño sin embargo es ideal para la realización de mediciones donde la cantidad de datos transmitidos es baja y sin embargo, recibir la confirmación de los mensajes para asegurar la fiabilidad de la operación es muy deseable.

Este sistema de comunicaciones se puede expandir fácilmente en trabajos futuros para incluir otras aplicaciones en capas superiores, con mensajes diferentes y que se beneficien de esta estructuras de comunicación generales. A la vez permite disponer a la aplicación de un método de comunicación fiable para la toma de medidas.

```
typedef struct connection_peer
{
    uint16_t peer_addr;
    uint8_t seq_ack_n;                  // 0b    000     ack    000    seq
    uint8_t ttl;
    virtual_timer_t tmo_timer;
    uint8_t last_message[120];
    uint8_t last_message_size;
    message_t last_message_type;
    message_t last_cmd_type;
} peer_connection_t;
```

Figura 6.8: Estructura que representa la conexión a un vecino

6.3.3. Capa de Aplicación

La principal responsabilidad de la aplicación es el intercambio de mensajes para SS-TWR, este intercambio se implementa de una forma similar al ejemplo proporcionado por el fabricante[35][10][23], incluyendo algunas estructuras y mensajes adicionales que se detallarán a continuación para facilitar los cálculos de posiciones.

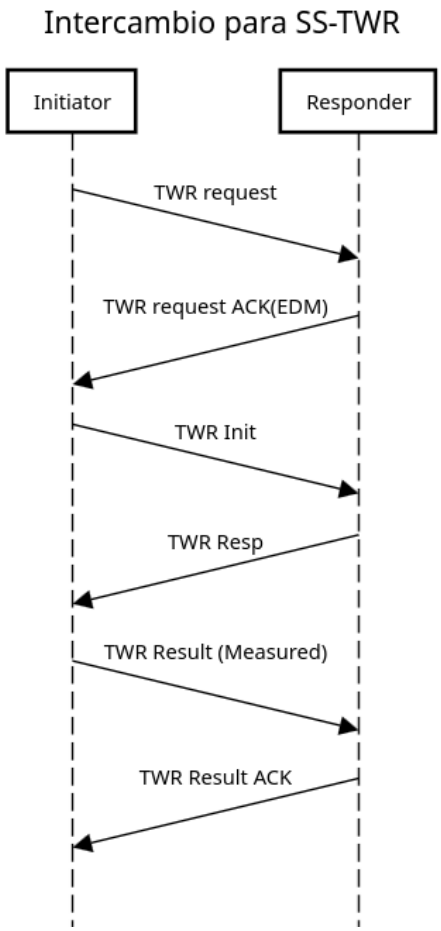

Figura 6.9: Intercambio de mensajes para SS-TWR

En la figura 6.9, cada medición de distancia implementada requiere 6 mensajes. Los primeros dos mensajes se utilizan para preparar a ambos dispositivos para realizar las medidas de tiempo y sincronizarlos. Además como estos mensajes no van a afectar a la medida independientemente de su tamaño o contenido, se usan para compartir información sobre las distancias medidas desde cada dispositivo desde su propio punto de vista.

Esta distribución de la información es vital ya que como se verá a continuación para la realización de la trilateración cada dispositivo necesita distancias que no puede medir directamente. Y por tanto debe obtener ya calculadas mediante la comunicación con su entorno.

Los siguientes mensajes son los que se usan para extraer los instantes temporales y calcular la distancia medida, como se detalla en el diseño, una vez se calcula esta distancia, los últimos mensajes informan a la pareja de la medida y finalizan el intercambio permitiendo al dispositivo comenzar otra medida, ya sea con el mismo dispositivo u otro distinto.

```
void twr_handle(peer_connection_t* peer)
{
    float distance = 0.0;
    if (twr_state != TWR_NO_TWR && peer != twr_peer)
        recvd_type = 0;

    loc_action = LOC_RESP_NOW;
    distance = MIN_DIST;
    switch (recvd_type)
    {
        case MT_D_REQ:
            if (twr_state == TWR_NO_TWR || twr_state == TWR_REQ_SENT)
            {
                if (twr_state == TWR_REQ_SENT &&
                    peer->peer_addr < panadr_own.short_addr)
                {
                    loc_action = LOC_NO_RESP;
                }
                else
                {
                    loc_state = LOC_TWR;
                    twr_state = TWR_REQ_RECVD;
                    twr_peer = peer;
                    twr_peer_seq = peer->seq_ack_n;
                    send_d_req_ack(peer);
                }
            }
            else
            {
                handle_twr_fail();
            }
            break;
        case MT_D_REQ_ACK:
            if (peer->last_message_type == MT_D_REQ &&
                twr_state == TWR_REQ_SENT)
            {
                twr_state = TWR_REQ_ACK_RECVD;
                send_w4r_msg(peer, 1, MT_D_INIT);
            }
            else
            {
                handle_twr_fail();
            }
            break;
        case MT_D_INIT:
            if (!(peer == twr_peer && twr_state == TWR_REQ_RECVD))
            {
                handle_twr_fail();
            }
            twr_state = TWR_INIT_RECVD;
            loc_action = LOC_NO_RESP;
```

Figura 6.10: Función ocupada de la interpretación y gestión de los mensajes TWR

La gestión de estos mensajes es responsabilidad de la hebra de comunicaciones, en esta una máquina de estados identifica el mensaje recibido clasificándolo y comprobando que el estado del sistema sea válido y se cumpla con el ordenamiento mostrado en el diagrama de secuencia 6.9.

```
void compute_distance(void)
{
    uint64_t m_tx_time, m_rx_time;
    uint64_t tx_time = dw_get_tx_time();
    uint64_t rx_time = 0;
    memcpy(&rx_time, recv_info.dw_rx_time.RX_STAMP,
           sizeof(recv_info.dw_rx_time.RX_STAMP));

    uint8_t lde_stat = 0;
    memcpy(&lde_stat, recv_buf+sizeof(m_tx_time)+sizeof(m_rx_time),
           sizeof(lde_stat));
    if (!rx_time || lde_stat)
    {
        memcpy(&rx_time, recv_info.dw_rx_time.RX_RAWST,
               sizeof(recv_info.dw_rx_time.RX_RAWST));
        rx_time -= rx_ant_d;

        rx_time = tx_time;
    }

    memcpy(&m_tx_time, recv_buf, sizeof(m_tx_time));
    memcpy(&m_rx_time, recv_buf+sizeof(m_tx_time), sizeof(m_rx_time));

    int64_t rt_init = (rx_time) - (tx_time);
    int64_t rt_resp = (m_tx_time) - (m_rx_time);

    float clock_offset_r = dw_get_car_int() * CAR_INT_CTE;
    rt_resp *= (1.0f - clock_offset_r);

    double tof = (rt_init - rt_resp)/2.0f;
    tof *= (1.0f/(float)DW_TIME_U);

    double distance = tof * C_ATM;

    peer_info_t* peer_info = get_peer_info(recvd_header.src_addr);

    if (peer_info)
        update_peer_distance(peer_info, (float)distance);
}
```

Figura 6.11: Función para el cálculo de la distancia dados los tiempos de envío y recepción

El cálculo de la distancia a partir de los intervalos de tiempo se indica en la sección 5.2.3, la implementación simplemente obtiene estos instantes de tiempo del hardware y los convierte a unidades de tiempo y distancia. Estos instantes son representados en el hardware como unidades del reloj del transceptor, cuyo formato es un entero de 40 bits y cuya unidad es equivalente a 15,65 picosegundos[6].

Previamente a la transformación a segundos de los tiempos se aplica una corrección que intentar mitigar los errores provenientes de la diferencia de relojes entre los dispositivos, este valor es calculado automáticamente por el hardware y se aplica a los intervalos de tiempo medidos.

Por último el tiempo de vuelo se transforma en distancia usando la velocidad de la luz en la atmósfera y esta distancia calculada se añade a la distancia anterior, proceso que se describirá a continuación.

Cuando se completa una medida de distancia esta se somete a una media móvil, donde su valor se añade a la distancia que ya calculada anteriormente. Asimismo se incluye esta distancia en una matriz de distancia euclídea que contiene todas las posibles distancias entre los dispositivos del entorno.

$$EDM = \begin{bmatrix} 0 & d_{01} & d_{02} \\ d_{10} & 0 & d_{12} \\ d_{20} & d_{21} & 0 \end{bmatrix}$$

Figura 6.12: Matriz de distancias euclídeas para 3 dispositivos

Además de un media móvil la distancia calculada también es sometida a un filtrado donde se eliminan valores no válidos, como pueden ser valores negativos o demasiado alejados de la media.

Habiendo construido la matriz de distancias en cada dispositivo, el cálculo de las posiciones relativas se realiza de la siguiente forma:

- La posición del propio dispositivo que realiza el cálculo se asume como referencia y se sitúa en el origen de coordenadas
- La posición del dispositivo representado en la fila 1 de la matriz, que se elige de forma arbitraria, se define en el eje de abscisas con coordenadas $(\frac{d_{01}+d_{10}}{2}, 0)$, en dos dimensiones.
- Por último, la posición del tercer dispositivo viene dada por las ecuaciones de trilateración, que se deducen como:

  Las coordenadas del punto (x, y) pueden expresarse usando las distancias dadas en la matriz de distancias euclídeas. Para simplificar la notación se considera:

  $$d = \frac{d_{01} + d_{10}}{2}, d_1 = \frac{d_{02} + d_{20}}{2}, d_2 = \frac{d_{12} + d_{21}}{2}$$

  Esto se realiza debido a que los errores de medida pueden ocasionar que:

  $$d_i j \neq d_j i$$

Obtenemos un sistema de dos ecuaciones con dos incógnitas:

$$\sqrt{x^2 + y^2} = d_1$$
$$\sqrt{(x-d)^2 + y^2} = d_2$$

Elevamos al cuadrado ambas ecuaciones:

$$x^2 + y^2 = d_1^2$$
$$(x-d)^2 + y^2 = d_2^2$$

Este sistema se puede resolver obteniendo las siguientes soluciones:

$$x = \frac{d^2 + d_1^2 - d_2^2}{2d}$$
$$y = \pm\sqrt{d_1^2 - \left(\frac{d^2 + d_1^2 - d_2^2}{2d}\right)^2}$$

Consideramos que todas las posiciones se encuentran en valores del eje de ordenadas positivos por lo que descartamos una de las soluciones. Por lo tanto las coordenadas del tercer dispositivo vienen dadas por:

$$\left(\frac{d^2 + d_1^2 - d_2^2}{2d}, \sqrt{d_1^2 - \left(\frac{d^2 + d_1^2 - d_2^2}{2d}\right)^2}\right)$$

6.4. Despliegue

Para ejecutar el código, además de la generación del binario, es necesario copiar este a la memoria de programa del microcontrolador en la placa de desarrollo. Esto se conoce coloquialmente como "flashear" la placa, este proceso es facilitado por el circuito programador J-Link[33] incluido en la placa de desarrollo 6.13.

Este módulo además de la programación de la placa, permite usar el puerto micro USB como puerto serie y permite además el depurado del código sin necesidad de un dispositivo depurador externo.

Para permitir la programación de varias placas conectadas a la estación de trabajo mediante USB, se ha incluido en el archivo Makefile varios objetivos que se apoyan en openOCD[31] para programar varias placas conectadas de forma automática. Como se puede entender la interfaz J-link es también compatible con openOCD, aunque requiere de configuración adicional.

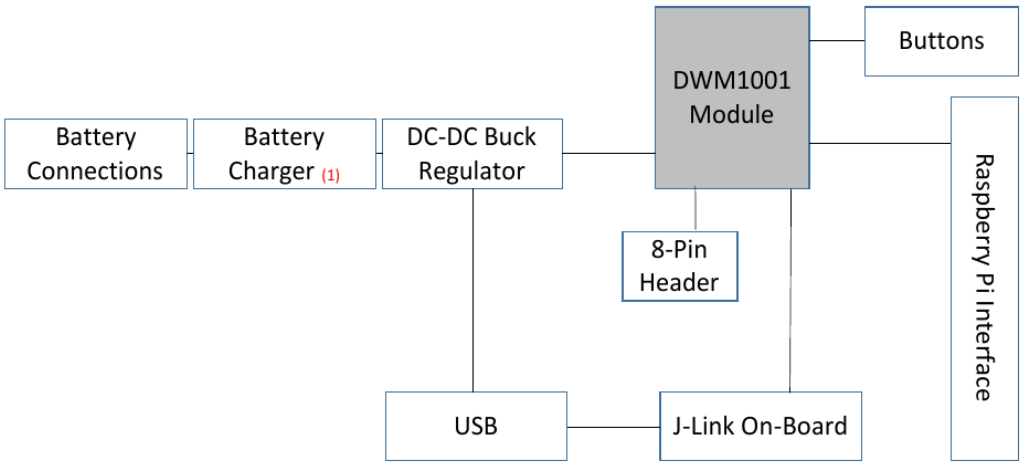

Figura 6.13: Componentes de la placa de desarrollo DWM1001-DEV

Capítulo 7

Experimentos y validación

Finalizada la implementación de los diferentes componentes del sistema, se pasa a validar la funcionalidad del sistema en su totalidad, midiendo distintas variable y observando el comportamiento de los dispositivos para estudiar si verdaderamente se cumplen los requisitos del proyecto. Además se evidenciará si el sistema puede ser usado en las aplicaciones señaladas en la introducción.

7.1. Diseño de los experimentos

La metodología a seguir en el diseño de los experimentos consiste en facilitar la toma de medidas de distancia durante el experimento, así como modelar y recrear situaciones reales de uso del sistema.

Se realizarán dos experimentos principales que se incluirán en este proyecto, el primero tendrá como objetivo caracterizar las medidas tomadas, estudiando los valores que pueden tomar, su frecuencia y su distribución de valores. El segundo se centrará en simular una situación posible en robótica móvil, donde 3 dispositivos se situarán en una formación geométrica y uno de estos se desplazará a lo largo del tiempo.

Se espera que este movimiento y estructura geométrica se vea reflejada en las posiciones relativas calculadas desde cada uno de los dispositivos.

Para la realización de estos experimentos la única calibración adicional realizada es búsqueda rudimentaria del valor de retardo de antena agregado, es decir, retardo de transmisión y recepción juntos. Esta calibración se basa en una minimización del error de medida entre 3 dispositivos, para minimizar este error se debe encontrar una serie de retardos candidato τ de forma que:

$$\min_{\forall \tau} \|(EDM_{\text{Actual}} - EDM_{\text{Measured}})\|$$

Para llegar a este mínimo se usa un algoritmo de enfriamiento que se ejecuta de forma externa sobre la matriz de distancias euclídeas obtenida en la medición inicial con los agentes en una formación de triángulo equilátero. Este proceso se describe en detalle en la documentación oficial[20] por lo que no se entrará en mayor detalle.

7.1.1. Caracterización de las métricas

Cuando se toman medidas, los errores están presentes desde muchos puntos de vista, realidad que se ha evidenciado durante el desarrollo de este proyecto. La mayoría de efectos que causan errores de gran magnitud son conocidos y descritos por el fabricante, a pesar de esto, debido a la escala del proyecto, no se han implementado sistemas para corregir o mitigar la mayoría de estas fuentes de error.

Por ejemplo, en la figura 7.1 se puede visualizar el error relacionado con la potencia de la señal recibida, que no sigue una relación lineal con esta. Esta es una de las fuentes de error que afectará a los resultados medidos entre otras.

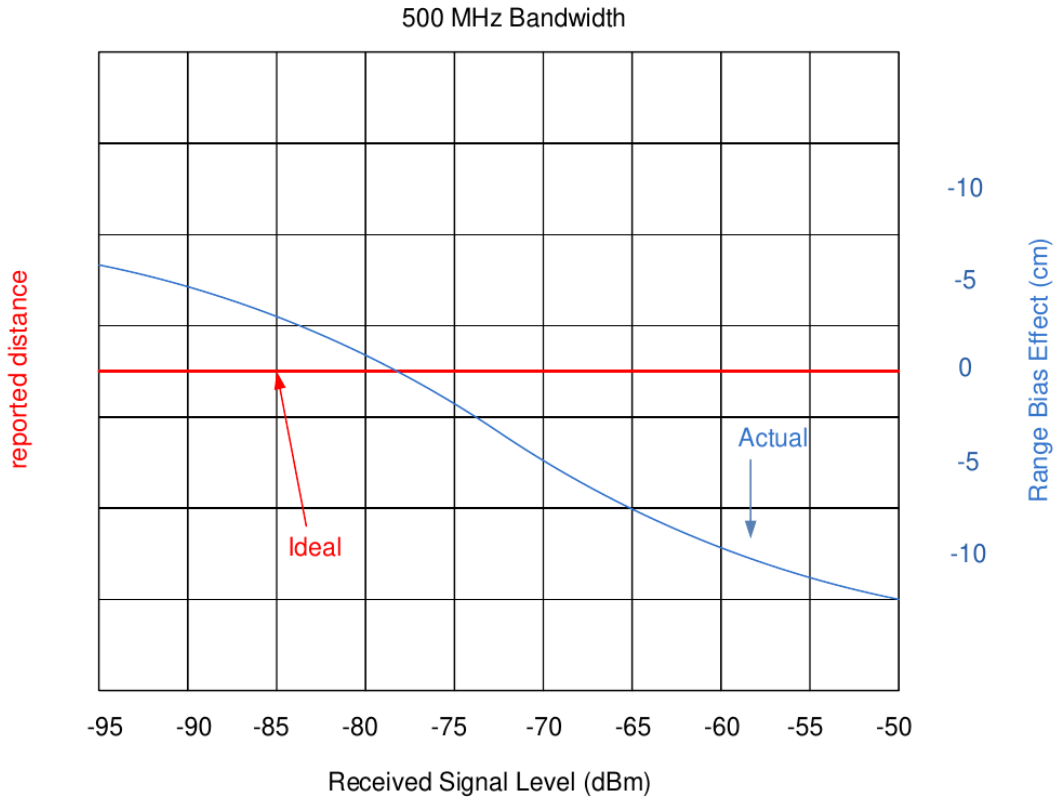

Figura 7.1: Efecto de la distancia a la potencia de la señal y la medida[22]

No obstante los resultados obtenidos siguen aportando datos útiles sobre otras variables como la precisión, frecuencia de medida o distribución.

Las medidas tomadas se usarán para el cálculo de las métricas expuestas en la sección 2.4 además de otros datos de interés.

7.1.2. Simulación de una aplicación en robótica móvil

Este experimento, como se ha descrito anteriormente se realizará con 3 dispositivos situados en una formación inicial de triángulo rectángulo, como se aprecia en la figura 7.2, las posiciones iniciales son A, B y C0. El dispositivo C será desplazado a las posiciones C1

y C2, mientras en todo momento se encuentra tomando medidas de distancia y calculando las posiciones relativas de los demás dispositivos.

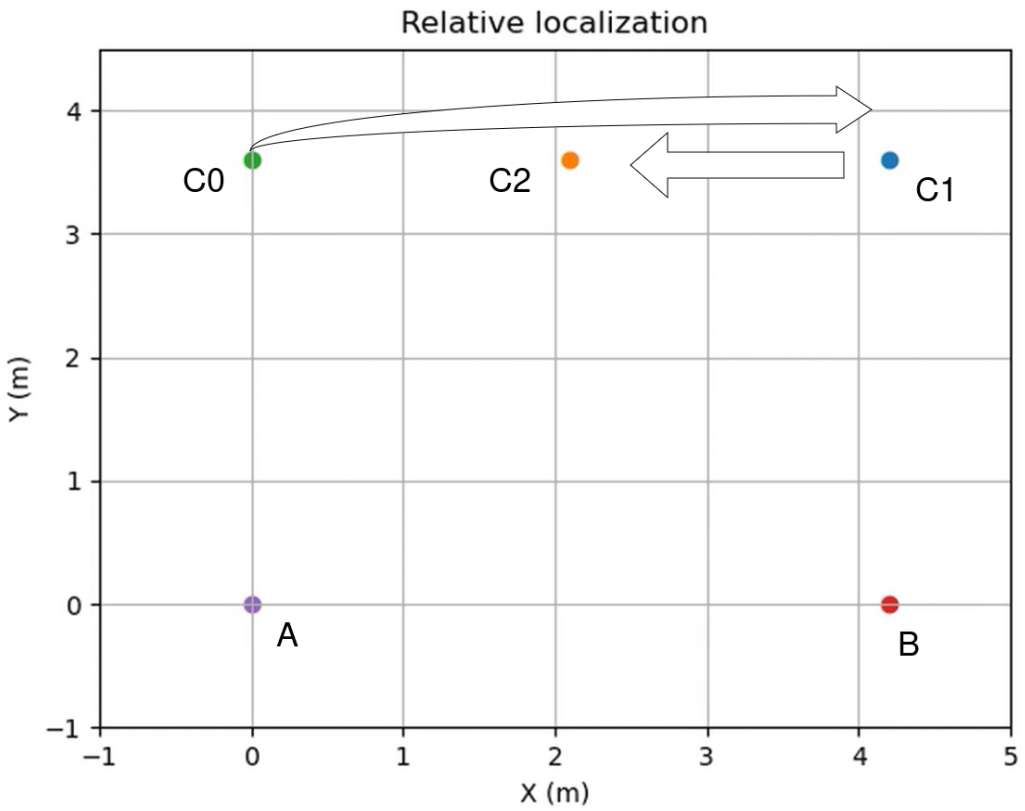

Figura 7.2: Movimiento real que se simula

Al presentar los datos por el puerto serie y con el objetivo de simplificar la interpretación gráfica de las posiciones se mostrarán los datos recogidos desde el dispositivo A, aunque cabe destacar que los 3 dispositivos involucrado calculan su propio mapa de las posiciones relativas de sus vecinos de forma completamente independiente.

7.2. Resultados experimentales

7.2.1. Métricas obtenidas

El primer experimento se realizó durante 2 minutos y 30 segundos, en los que una pareja de dispositivos tomaron medidas de distancia de forma automática a una distancia real de alrededor de **8 metros**.

Los datos de distancia medida obtenidos durante el experimento llegan a un total de **2265**, 1154 medidas en un sentido y 1111 en el otro, contando el intervalo de tiempo en el que transcurre el experimento, la frecuencia de medida o el número de medidas por segundo es:

$$F_m = \frac{N_m}{T} = \frac{2265 medidas}{150 segundos} = 15,1 \frac{medidas}{segundo}$$

Para estudiar los demás resultados consideraremos únicamente los valores medidos en un sentido, esta decisión es debida a que el valor medio de las medidas sufre errores de exactitud debido al retardo de las antenas siendo distinto en la realidad, además de otras fuentes de error mencionadas.

De entre los 1111 valores medidos en un sentido podemos obtener las siguientes características de la distribución, los valores de distancia son medidos en **metros**:

- **Media**: 8.71
- **Mediana**: 8.68
- **Desviación típica**: 0.36
- **Rango**: 2.01

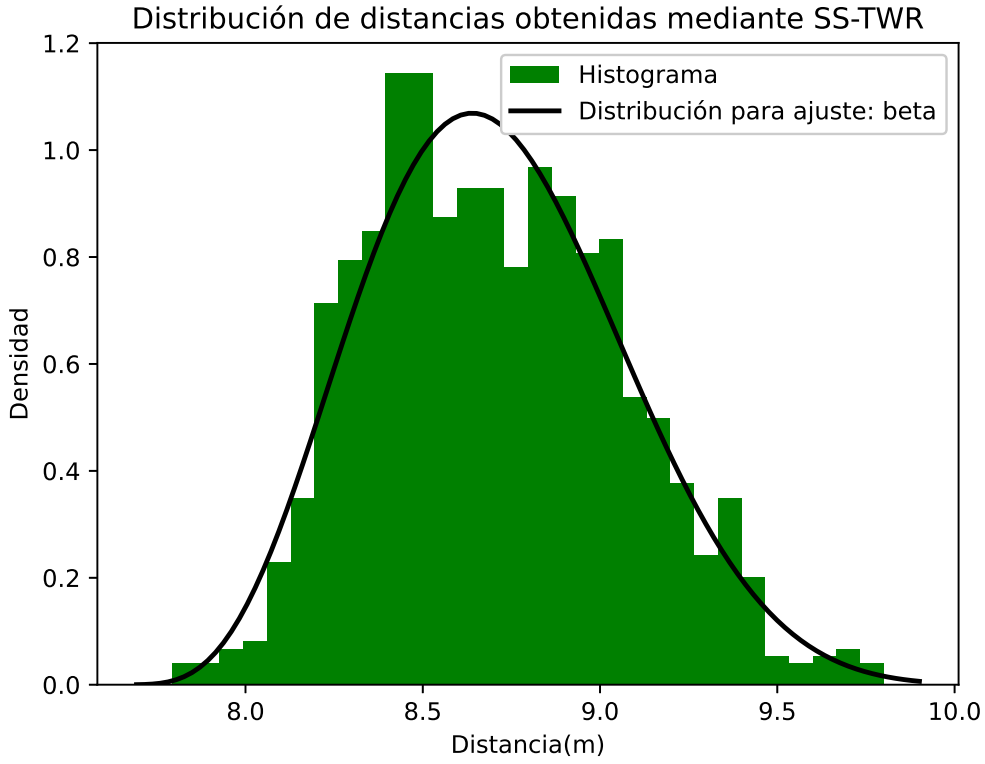

Figura 7.3: Distribución de las medidas tomadas

La distribución a la que se han ajustado los datos es una distribución **beta**, sin embargo se puede observar como la forma de la distribución no se aleja mucho de una distribución normal, lo que es esperable en este tipo de medidas de fenómenos físicos como es la propagación de ondas electromagnéticas.

7.2.2. Simulación de una aplicación en robótica móvil

Los resultados del experimento de localización son prometedores, durante aproximadamente un minuto, se recogieron las posiciones relativas calculadas por el dispositivo en la posición A 7.2, estas posiciones reflejan la realidad del experimento de forma satisfactoria.

En la figura 7.4 podemos observar los puntos de las posiciones reales, en colores, y las posiciones relativas calculadas en azul. La circunferencia discontinua representa el máximo error de posición, obtenido en el proceso de calibración.

Como se indica en la sección 7.1 el proceso de calibración se basa en minimizar el error de la medida, modificando los valores de retardo de la antena. Tras el proceso de calibración, se obtiene la matriz de error para los valores obtenidos, de esta forma se puede acotar el error de exactitud máximo que en este caso era de 3,16 metros.

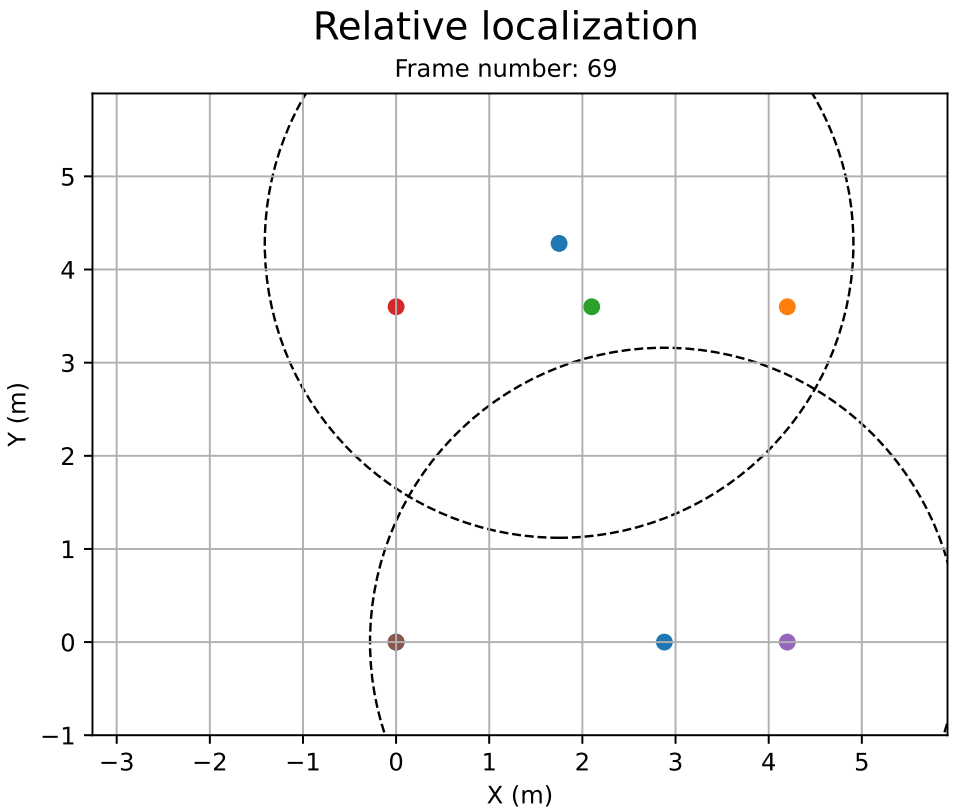

Figura 7.4: Posicionamiento relativo con 3 nodos a 360 cm

En la figura se muestran los valores medidos para la posiciones reales A, B y C2, la situación final del movimiento. Se puede apreciar como las posiciones están sobradamente dentro del rango de error calculado en cuanto a exactitud, representando un polígono con una forma similar al real. Además, se puede visualizar una animación de las posiciones en el tiempo[38] donde comprobar que la precisión de las medidas también es adecuada y entra dentro de los errores esperados.

Este experimento demuestra que, a pesar de no obtener distancias reales, este sistema puede ser usado en aplicaciones de robótica móvil, sobretodo teniendo en cuenta que los

errores de precisión y exactitud no serán muy significativos a escalas de cientos de metros.

7.3. Comprobación de métricas e indicadores y validación

Una vez realizados los experimentos se deben valorar las métricas e indicadores definidos al principio del proyecto 2.4, con el objetivo de validar que los resultados obtenidos concuerdan con lo esperado y son útiles para el cumplimiento de los requisitos propuestos y el desarrollo de las aplicaciones señaladas.

7.3.1. Métricas cuantitativas

Las métricas cuantitativas, exactitud, precisión y frecuencia de medida han mostrado resultados muy prometedores en ambos experimentos. Como se muestra en el primer experimento, a pesar de partir de unas condiciones no ideales y no corregir las diversas fuentes de error conocidas, los resultados no solo se encuentran dentro del error esperado, sino que demuestran ser usables.

Esto es, las métricas calculadas, así como la distribución de las medidas indican que estas distancias obtenidas son útiles y permiten la implementación de localización relativa en sistemas con requisitos razonables de precisión, como es el caso de la robótica móvil.

Esto es evidenciado por los resultados mostrados en la simulación de esta aplicación en el segundo experimento.

7.3.2. Indicadores

Para validar el funcionamiento del sistema y valorar el **rendimiento** mediante el KPI solo hace falta analizar los resultados presentados en la sección 7.2.2, en la figura 7.4 puede apreciarse claramente una formación poligonal que como se expone es extremadamente similar a la formación.

Esta formación representada es actualizada en tiempo real como se puede ver en la animación realizada a partir de los datos obtenidos. Por tanto se puede concluir que el rendimiento del sistema **cumple con creces** las expectativas. Esto significa que el sistema es capaz de proporcionar posiciones relativas en un enjambre de robots que representan de forma leal la formación real que componen los robots.

Los demás indicadores definidos, de la misma forma, muestran que el comportamiento del sistema es correcto y esperado.

- La descentralización forma una parte intrínseca del diseño del sistema, en ninguna prueba o experimento los dispositivos han ejecutado un programa diferente o han sido diferenciados. Esto evidencia que todos los dispositivos son iguales y no diferenciables, lo que resulta en un comportamiento completamente descentralizado y distribuido. Además no se coordinó de forma externa la conexión o comunicación de los dispositivos, partiendo todos ellos del estado inicial y incluyéndose en el sistema de forma dinámica en instantes de tiempo arbitrarios.
- Las medidas tomadas y las posiciones calculadas fueron actualizadas continuamente durante la duración de ambos experimentos, esto indica que el rendimiento del sistema no fue degradado y por tanto las comunicaciones cuentan con unas condiciones de **fiabilidad** suficientes.

- La escalabilidad del sistema no ha sido comprobada de forma directa, no obstante debido a la definición de un entorno local y la validación de este, ningún dato indica que no puedan coexistir en el mismo espacio más de uno de estos entornos locales.

En conclusión, el sistema ha sido validado positivamente mediante los resultados obtenidos experimentalmente y en referencia a los valores cuantitativos y resultados cualitativos esperados. Cabe destacar que pese a la existencia de diversas fuentes de error que afectan negativamente a los resultados, estos errores son conocidos y están dentro de lo esperado.

Esto supone que el sistema, incluso con los inevitables errores de medida, es predecible y además está abierto a trabajos futuros en los que se mitiguen estas fuentes de error.

Capítulo 8

Conclusiones

De forma general, el objetivo del proyecto ha sido cumplido, proporcionando un sistema como prueba de concepto que demuestra resultados reales de localización relativa de forma distribuida.

8.1. Evaluación de la planificación inicial

La planificación inicial propuesta ha sido utilizada durante el desarrollo del proyecto y ha demostrado ser adecuada. Sin embargo, es cierto que uno de los aspectos que se ha pasado por alto en el diseño de esta primera planificación es el desarrollo iterativo. Con esto se da a entender que durante el desarrollo de módulos o componentes situados al final de la planificación ha sido necesario **revisitar** el desarrollo de módulos anteriores.

Esta técnica de desarrollo proporciona diversas ventajas, evitando afrontar todos los problemas de integración a la vez, dividiendo este problema y permitiendo la prueba y validación del sistema previamente a su completado total. Esto asegura la funcionalidad del sistema a lo largo del desarrollo e impide a un gran desafío presentarse de forma repentina en la conclusión del desarrollo.

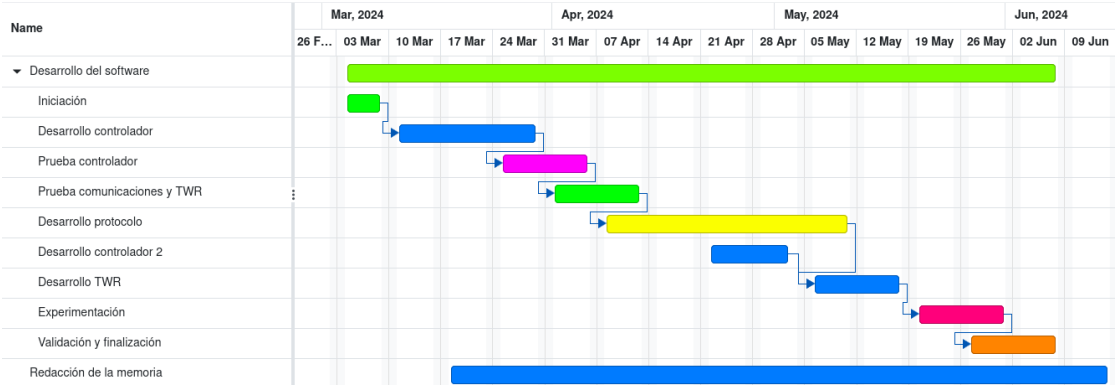

Figura 8.1: Diagrama de Gantt del desarrollo real

8.2. Trabajos Futuros

Al tratarse de una prueba de concepto, este proyecto presenta una gran oportunidad de desarrollo para trabajos futuros. La arquitectura presentada facilita la mejora de los diferentes módulos del proyecto, por ejemplo, introduciendo nuevas aplicaciones a alto nivel, o añadiendo más funcionalidad a nivel de comunicaciones.

Uno de las mejoras más inmediatas se basan en modelar las diferentes fuentes de error conocidas para así mitigar su efecto de forma más efectiva. Entre otras mejoras de bajo nivel, se pueden considerar adiciones como introducir un esquema DS-TWR o añadir actualizaciones OTA.

Además se podría fácilmente introducir modificaciones en la capa de aplicación para permitir el posicionamiento en **3 dimensiones**. Dentro del campo de la robótica, añadir aplicaciones con sistemas de redundancia usando más de un dispositivo para obtener la localización conformaría una mejora de la fiabilidad significativa.

A nivel de red, implementar un sistema de enrutamiento que permitiera la comunicación indirecta entre los distintos entornos locales podría suponer una mejora del rendimiento del sistema a gran escala.

En resumen existen multitud de oportunidades para trabajos futuros, utilizando este sistema como plataforma donde desarrollar nuevas funcionalidades o aplicaciones, así como mejorar sus características o rendimiento.

8.3. Impacto y valoración personal

Se espera que este tipo de sistemas tengan un gran impacto en el sector del control y la robótica móvil, sobretodo en aplicaciones donde otros sistemas de localización no son útiles debido a su naturaleza centralizada, como pueden ser los sistemas de captura de movimiento.

La experimentación en robots móviles usando esta plataforma ya es posible abriendo una nueva avenida de posibilidades para el control de enjambres de robots imperfectos de forma distribuida.

Como valoración personal, he de decir que este proyecto ha supuesto una de las oportunidades más grandes en mi vida para desarrollarme como persona. He aprendido de una diversidad de ramas del conocimiento al tratarse de un proyecto multidisciplinario, mejorando mis habilidades tanto técnicas como sociales.

Este proyecto marca para mi una introducción al mundo de la investigación y administración de proyectos científicos, algo que me apasiona y espero continuar desarrollando en el futuro. Además quiero destacar mi satisfacción con el resultado del proyecto, que incluso no siendo perfecto, ha demostrado resultados de gran valor.

Por último, quiero enfatizar la importancia del software libre y animar a cualquier lector a participar y expandir el proyecto de cualquier forma que le sea posible.

Bibliografía

[1] J. Nascimento, "A Study on Range - Data Rate Performance of UWB Wireless Communications," 2007. dirección: https://api.semanticscholar.org/CorpusID:107015342.

[2] *Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Standard 802.15.4-2011, 2011.

[3] M. A. Mazidi, S. Naimi y S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*. Upper Saddle River, N.J: Prentice Hall, 2011, 776 págs., ISBN: 978-0-13-800331-9.

[4] Decawave, *DWM1001 Squematic*, ver. 1.21, Qorvo, Inc, 2018. dirección: https://www.qorvo.com/products/p/DWM1001C.

[5] Decawave, *DWM1001-DEV Squematic*, ver. 4, Qorvo, Inc, 2018. dirección: https://www.qorvo.com/products/p/DWM1001-DEV.

[6] Decawave, *DW1000 User Manual*, ver. 2.17, Qorvo, Inc, 2019. dirección: https://www.qorvo.com/products/p/DW1000.

[7] Decawave, *DWM1001-DEV Datasheet*, ver. 2.17, Qorvo, Inc, 2019. dirección: https://www.qorvo.com/products/p/DWM1001-DEV.

[8] A. Golubkov, A. Tsyganov, J. Tsyganova e I. Petrishchev, "Decentralized multisensor estimation of motion parameters of an object moving along a complex trajectory," *Journal of Physics: Conference Series*, vol. 1368, pág. 042 041, nov. de 2019. DOI: 10.1088/1742-6596/1368/4/042041.

[9] Decawave, *DW1000 Datasheet*, ver. 2.22, Qorvo, Inc, 2020. dirección: https://www.qorvo.com/products/p/DW1000.

[10] Á. V. García, *Abvg9/Distributed_localization_DWM1001*, 26 de nov. de 2022. dirección: https://github.com/abvg9/Distributed_localization_DWM1001.

[11] "Trilateración," en *Wikipedia, la enciclopedia libre*, 8 de oct. de 2022. dirección: https://es.wikipedia.org/w/index.php?title=Trilateraci%C3%B3n&oldid=146465977.

[12] "Capa de enlace de datos," en *Wikipedia, la enciclopedia libre*, 9 de dic. de 2023. dirección: https://es.wikipedia.org/w/index.php?title=Capa_de_enlace_de_datos&oldid=155913351.

[13] M. Ball, *Marus/Cortex-Debug*, 12 de jun. de 2024. dirección: https://github.com/Marus/cortex-debug.

[14] "Calculadora coste empresa contratar un trabajador — Factorial." (2024), dirección: https://factorialhr.es/calculadora-coste-trabajador.

[15] "ChibiOS Forum." (2024), dirección: https://forum.chibios.org/.

[16] "ChibiOS Wiki." (2024), dirección: https://www.chibios.org/dokuwiki/doku.php.

[17] "ChibiOS/RT Guide." (2024), dirección: https://www.chibios.org/dokuwiki/doku.php?id=chibios:documentation:books:rt:start.

[18] "ChibiOS/RT Source." (2024), dirección: https://chibios.sourceforge.net/docs3/rt/index.html.

[19] "Connectivity of the Internet of Things - SparkFun Learn." (2024), dirección: https://learn.sparkfun.com/tutorials/connectivity-of-the-internet-of-things/infrastructure-and-ad-hoc-networks-.

[20] Decawave, *Antenna delay calibration of DW1000-based products and systems*, ver. 1.3, Qorvo, Inc, 2024. dirección: https://www.qorvo.com/products/p/DW1000.

[21] Decawave, *Channel effects on communications, range and time stamp accuracy in DW1000 based systems*, ver. 1.04, Qorvo, Inc, 2024. dirección: https://www.qorvo.com/products/p/DW1000.

[22] Decawave, *Sources of error in DW1000 based two-way ranging(twr) schemes*, ver. 1.2, Qorvo, Inc, 2024. dirección: https://www.qorvo.com/products/p/DW1000.

[23] Decawave, *The implementation of two-way ranging with the DW1000*, ver. 2.4, Qorvo, Inc, 2024. dirección: https://www.qorvo.com/products/p/DW1000.

[24] J. Díaz, "Teoría de Redes - Tema 4 - HDLC," 2024.

[25] "DW1000 - Qorvo." (2024), dirección: https://www.qorvo.com/products/p/DW1000.

[26] "DWM1001-DEV - Qorvo." (2024), dirección: https://www.qorvo.com/products/p/DWM1001-DEV.

[27] "DWM1001C - Qorvo." (2024), dirección: https://www.qorvo.com/products/p/DWM1001C.

[28] "GDB: The GNU Project Debugger." (2024), dirección: https://www.gnu.org/savannah-checkouts/gnu/gdb/index.html.

[29] "IEEE 802.15.4," en *Wikipedia*, 4 de jun. de 2024. dirección: https://en.wikipedia.org/w/index.php?title=IEEE_802.15.4&oldid=1227302237.

[30] A. Jain. "Difference between LOS and NLOS in Wireless LoRa Technology," Engineers Garage. (2024), dirección: https://www.engineersgarage.com/difference-between-los-and-nlos-in-wireless-lora-technology/.

[31] "Open On-Chip Debugger." (2024), dirección: https://openocd.org/.

[32] "PLAY Embedded," PLAY Embedded. (18 de mar. de 2024), dirección: https://www.playembedded.org/blog.

[33] "SEGGER - J-Link / J-Trace." (2024), dirección: https://www.segger.com/downloads/jlink/.

[34] "Sueldo: Ingeniero De Software Embebido en en 2024," Glassdoor. (2024), dirección: https://www.glassdoor.es/Sueldos/ingeniero-de-software-embebido-sueldo-SRCH_KO0,30.htm.

[35] *Decawave/Dwm1001-Examples*. dirección: `https://github.com/Decawave/dwm1001-examples`.

[36] *nRF52832 Product Specification*.

[37] *UCM-237/ChibiOS-Contrib*. dirección: `https://github.com/UCM-237/ChibiOS-Contrib`.

[38] *UCM-237/Distributed_localization_DWM100*. dirección: `https://github.com/UCM-237/Distributed_localization_DWM1001`.