Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

CONTROL DESCENTRALIZADO PARA LA RESOLUCIÓN DE CONFLICTOS EN LA NAVEGACIÓN CON MÚLTIPLES ROBOTS MÓVILES

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autora: Sandra Moreno Olivares

Tutor: Martín Mellado Arteche

Curso 2019-2020

Resumen

El Instituto de Automática e Informática Industrial (ai2) de la *Universitat Politècnica de València* (UPV) se encuentra desarrollando el proyecto europeo *Safe, Efficient and Integrated Indoor Robotic Fleet for Logistic Applications in Healthcare and Commercial Spaces*, denominado ENDORSE.

El proyecto ENDORSE aspira a la introducción de robots móviles colaborativos en entornos de interior sanitarios a fin de ayudar los profesionales del sector en la elaboración de las tareas más repetitivas como, por ejemplo, en la medición básica de constantes vitales o el transporte de alimentos, medicinas u otros objetos de unas zonas a otras.

Por otro lado, dicho proyecto se ocupa de varias áreas de investigación diferenciadas en las que participan diversas universidades y empresas europeas, entre ellas la UPV y la empresa valenciana *Robotnik*. El instituto ai2 se encarga, entre otras actividades, de resolver la tarea de *Human-aware path planning algorithms*, en la que se plantea la necesidad de implementar algoritmos que mejoren el sistema de navegación controlado por la aplicación central para el control de la flota robótica móvil a fin de evitar colisiones de los robots móviles con obstáculos y personas, así como que detecten y resuelvan bloqueos eventuales.

Este Trabajo Final de Máster (TFM) trata de abordar la tarea gestionada por el instituto ai2 a partir del desarrollo de un algoritmo para la resolución de los problemas en la navegación entre múltiples robots mediante un control descentralizado que permita a los robots comunicarse entre sí.

Los robots móviles colaborativos con los que se trabaja en el proyecto son los *RB-1 Base* de la empresa *Robotnik* y, para el desarrollo del control descentralizado, se utilizará ROS (*Robot Operating System*); el lenguaje de programación Python se emplea para obtener las lecturas de los diversos sensores que contienen los *RB-1 Base*, interpretar dichos datos y establecer las comunicaciones necesarias con la finalidad de conocer la localización de los robots y resolver los conflictos ocasionados durante la navegación.

Palabras clave: Robots móviles, robots colaborativos, control descentralizado, ROS, navegación de robots, evitación colisiones.

Resum

El Instituto de Automática e Informática Industrial (ai2) de la Universitat Politècnica de València (UPV) es troba desenvolupant el projecte europeu *Safe, Efficient and Integrated Indoor Robotic Fleet for Logistic Applications in Healthcare and Commercial Spaces*, denominat ENDORSE.

El projecte ENDORSE aspira a la introducció de robots mòbils col·laboratius en entorns d'interior sanitaris per tal d'ajudar als professionals del sector en l'elaboració de les tasques més repetitives com, per exemple, en el mesurament bàsic de constants vitals o el transport d'aliments, medicines o altres objectes d'unes zones a unes altres.

D'altra banda, aquest projecte s'ocupa de diverses àrees d'investigació diferenciades en les quals participen vàries universitats i empreses europees, entre elles la UPV i l'empresa valenciana *Robotnik*. L'institut ai2 s'encarrega, entre altres activitats, de resoldre la tasca de *Human-aware path planning algorithms*, en la qual es planteja la necessitat d'implementar algoritmes que milloren el sistema de navegació controlat per l'aplicació central per al control de la flota robòtica mòbil a fi d'evitar col·lisions dels robots mòbils amb obstacles i persones, així com que detecten i resolen bloquejos eventuals.

Aquest Treball Final de Màster (TFM) tracta d'abordar la tasca gestionada per l'institut ai2 a partir del desenvolupament d'un algoritme per a la resolució dels problemes en la navegació entre múltiples robots mitjançant un control descentralitzat que permeta als robots comunicar-se entre ells.

Els robots mòbils col·laboratius amb els quals es treballa en el projecte són els *RB-1 Base* de l'empresa *Robotnik* i, per al desenvolupament del control descentralitzat, s'utilitzarà ROS (*Robot Operating System*); el llenguatge de programació Python s'empra per obtenir les lectures dels diversos sensors que contenen els *RB-1 Base*, interpretar aquestes dades i establir les comunicacions necessàries amb la finalitat de conèixer la localització dels robots i resoldre els conflictes ocasionats durant la navegació.

Paraules clau: Robots mòbils, robots col·laboratius, control descentralitzat, ROS, navegació de robots, evitació col·lisions.

Abstract

The Institute of *Automática e Informática Industrial* (ai2) of the *Universitat Politècnica de València* (UPV), is developing the European project *Safe, Efficient and Integrated Indoor Robotic Fleet for Logistic Applications in Healthcare and Commercial Spaces*, called ENDORSE Project.

The ENDORSE project aims to introduce collaborative mobile robots in indoor healthcare environments. The purpose of the robots is to assist healthcare professionals in performing the most repetitive tasks, such as the basic measurement of vital signs or the transport of food, medicines or other objects from one area to another.

On the other hand, this project deals with several differentiated research areas in which several European universities and companies are involved, including the UPV and the Valencian company Robotnik. The ai2 Institute is in charge, among other activities, of solving the task of *Human-aware path planning algorithms*. To solve this task, it is necessary to implement algorithms that improve the navigation system controlled by the central application for the control of the mobile robotic fleet. The goal is to avoid collisions of the mobile robots with obstacles and people, as well as to detect and resolve eventual deadlocks.

This Masters dissertation tries to address the task managed by the ai2 Institute by developing an algorithm in order to solve problems in navigation system among multiple mobile robots through a decentralized control that allows the robots to communicate with each other.

The collaborative mobile robots working on the project are RB-1 Base of the Robotnik company. Furthermore, for the development of decentralized control will be used ROS (Robot Operating System). The Python programming language is used to obtain the readings of the different sensors that contain RB-1 Base as well as to interpret these data and to establish the necessary communications for knowing the location of the robots and to solve the conflicts caused during the navigation.

Keywords: Mobile robots, collaborative robots, decentralized control, ROS, robot navigation, collision avoidance.

Índice general

Índice de ilustraciones

Agradecimientos

A mi tutor y responsable de equipo de trabajo en el Instituto de Automática e Informática Industrial (ai2) de la *Universitat Politècnica de València* (UPV), don Martín Mellado Arteche por su guía durante este proyecto.

A los colaboradores del proyecto ENDORSE, dentro del cual se ha desarrollado este Trabajo Fin de Máster, especialmente a don Marc Bosch Jorge, de la empresa *Robotnik*, a don Francisco Fraile Gil, del Centro de Investigación Gestión e Ingeniería de Producción (CIGIP) de la UPV y a mi compañera de equipo Marta Payá Tormo; estoy muy agradecida de haberles tenido como apoyo técnico en el diseño e implementación de este trabajo.

A mis amigos José Buzón García y Daria Corina Bec por su apoyo incondicional, incluso durante la cuarentena del COVID-19, y por esos largos paseos y charlas para motivarme. Me siento muy afortunada de tenerles en mi vida.

Y a mis padres y hermanos, que siempre habéis estado a mi lado, impulsándome para poder llegar hasta aquí: os quiero mucho, gracias por estar ahí.

CAPÍTULO 1
Introducción

El Instituto de Automática e Informática Industrial (ai2) de la *Universitat Politècnica de València* (UPV) participa en el desarrollo del proyecto europeo ENDORSE (*Safe, Efficient and Integrated Indoor Robotic Fleet for Logistic Applications in Healthcare and Commercial Spaces*), ideado para investigar las posibilidades de mejorar las condiciones de trabajo del personal sanitario en las labores diarias automatizables mediante la introducción de robots móviles colaborativos en entornos sanitarios de interior.

El equipo del instituto ai2 en el que estoy contratada participa en dicho proyecto desarrollando la tarea *Human-aware path planning algorithms*, entre otras actividades. Dicha tarea consiste en introducir mejoras en el sistema de navegación de los robots móviles colaborativos *RB-1 Base* de la empresa *Robotnik* de los que dispone el proyecto ENDORSE que consisten, en esencia, en crear algoritmos para la detección y resolución de conflictos con humanos y otro tipo de obstáculos en determinadas situaciones durante el desplazamiento de los robots móviles por el entorno de trabajo.

Este Trabajo Fin de Máster (TFM) nace de la necesidad de resolver la tarea *Human-aware path planning algorithms* y se centra en la resolución de los eventuales conflictos de navegación entre varios robots móviles en áreas potencialmente conflictivas (como por ejemplo zonas estrechas o limitadas a dos robots móviles circulando simultáneamente) sin recurrir al sistema central de gestión de la flota robótica del proyecto ENDORSE. Con ese fin, este TFM aglutina dos objetivos:

- Diseñar un sistema de comunicaciones entre los robots móviles del proyecto, así como definir los mensajes que deben intercambiarse.

- Desarrollar un algoritmo para la resolución de los conflictos en el sistema de navegación de los robots móviles mediante un control descentralizado.

Por lo que respecta a esta memoria, su estructura se divide en cinco capítulos. En el primer capítulo se presenta el contenido y objetivos del Trabajo Final de Máster propuesto. El segundo, introduce el mundo de la robótica y el entorno laboral en el que se desarrolla el proyecto ENDORSE a fin de contextualizar la solución propuesta en el presente trabajo. Seguidamente, se describen las tecnologías y herramientas utilizadas a lo largo del proyecto que se mencionan en los capítulos posteriores. En el cuarto capítulo, se presenta inicialmente el plan de trabajo

acordado y, a continuación, se explica tanto el diseño como el desarrollo la solución propuesta. Durante el quinto capítulo, se detallan las pruebas elaboradas al trabajo realizado y se exponen los resultados de los análisis de las mismas. Finalmente, en el último capítulo, se presentan las conclusiones alcanzadas una vez finalizado dicho Trabajo Final de Máster y se proponen futuras mejoras realizables.

Por último, hay que señalar que la empresa valenciana *Robotnik* ha colaborado en este TFM en la resolución de dudas originadas al utilizar los robots móviles *RB-1 Base*, así como en la guía y verificación del sistema de comunicaciones propuesto entre dichos robots móviles. Además, el equipo del Centro de Investigación Gestión e Ingeniería de Producción (CIGIP) de la UPV responsable del desarrollo de la aplicación central para el control de la flota ha participado en dicho TFM durante el diseño de los mensajes intercambiados entre los robots y el sistema central del control de flotas.

CAPÍTULO 2
Contexto y entorno de trabajo

En este capítulo se describirá el contexto tecnológico que rodea a este proyecto, el ámbito en el que se elabora y la necesidad o espacio que trata de ser cubierto con la realización de este Trabajo Fin de Máster.

2.1 Robótica

La robótica es una rama multidisciplinar de la ciencia que combina diversas áreas de conocimiento (como la mecánica, la electrónica y la informática) con el propósito de estudiar, diseñar y construir robots para distintas finalidades.

Según la aplicación práctica para la que están destinados los robots se puede distinguir entre:

- **Robótica industrial:** se ocupa de los robots que cumplen con la ISO 8373 [1] utilizados en el mundo manufacturero, que comprende tanto las fábricas como los almacenes industriales.

- **Robótica de servicio:** comprende los robots diseñados para ofrecer servicios al ser humano, ya sea de forma parcial o totalmente autónoma, procurando su bienestar. Estos robots no están habilitados para realizar operaciones manufactureras, es decir, no están diseñados para llevar a cabo aplicaciones industriales. Dentro de este grupo, se puede distinguir entre los robots de uso personal o doméstico (aspiradoras robóticas, robots de vigilancia, etc.) y los robots para trabajos profesionales (robots para rescate y emergencias, sanitarios, militares, educativos, etc.).

Por otro lado, desde el punto de vista de su estructura, los robots se pueden clasificar de múltiples formas. Podemos encontrar los siguientes tipos:

- **Articulados:** agrupa a los robots estructurados con capacidad para mover sus elementos terminales (herramientas, pinzas de sujeción, ventosas de vacío…) en un espacio de trabajo determinado por uno o varios sistemas de coordenadas, con un número específico de grados de libertad. Dichos robots, a su vez, se suelen destinar fundamentalmente a un uso sedentario; no obstante, se les puede guiar excepcionalmente para realizar desplazamientos limitados. Los robots articulados comprenden los robots manipuladores, industriales y cartesianos.

- **Móviles:** son los robots capaces de desplazarse mediante un sistema locomoción (aéreo, terrestre o marino). Para llegar a su destino, los robots móviles se guían con los datos de entorno recibidos a través de sus sensores o bien son controlados mediante el telemando. Por otra parte, algunos robots de esta clase necesitan una infraestructura externa para desplazarse (como bandas detectadas fotoeléctricamente, pistas materializadas mediante radiación electromagnética, etc.), mientras que otros, dotados de un mayor nivel de inteligencia, se desplazan sin necesidad de infraestructura de ningún tipo.

- **Androides:** son robots que tratan de emular (total o parcialmente) la forma y/o conducta del ser humano. Este grupo de robots no ha evolucionado demasiado debido a la enorme dificultad que representa la resolución del problema principal de locomoción bípeda y de la conservación del equilibrio en tiempo real.

- **Zoomórficos:** caracterizados por intentar reproducir la morfología o conducta de diversos seres vivos del mundo animal adaptando la estructura de su sistema de locomoción, la mayoría de los robots zoomórficos, a pesar de la diversidad de forma, se ha desarrollado con la capacidad de desplazarse para cumplir diversos objetivos como, por ejemplo, moverse en entornos adversos, poco homogéneos o accidentados (tales como medios montañosos, volcanes, el espacio, etc.). También existen robots aéreos y acuáticos semejantes a ciertas aves o peces.

- **Híbridos:** robots con características que combinan algunas propias de las anteriores clasificaciones, bien sea por yuxtaposición o por conjunción.

En los últimos años, se ha producido una espectacular evolución de la robótica como consecuencia, sobre todo, del proceso global de digitalización, de la producción inteligente y de la industria 4.0, que ha mostrado mejor que ninguna otra área la necesidad de mejorar la interacción humano-robot (o *Human-Robot Interaction* HRI). Producto de esta evolución son los denominados “robots colaborativos” o *cobots*, diseñados para efectuar tareas de forma segura en colaboración con humanos preservando su integridad. Con el fin de garantizar la seguridad humana, se siguen en la fabricación de estos *cobots* ciertas pautas, como la utilización de materiales ligeros, sensores en base o articulaciones del robot para el control y medición tanto de la fuerza como de la velocidad, contornos redondeados etc. Dichas características se encuentran definidas en la ISO recogida por la Federación Internacional de Robótica o *International Federation of Robotics* (IFR), que a su vez distingue entre dos tipos de robots colaborativos:

- Aquellos que están diseñados para un uso colaborativo y acatan la ISO 10218/2 [2], que detalla los requerimientos (o pautas) para un diseño seguro, medidas de protección y manuales o instrucciones de uso.

- Los robots concebidos para una utilización colaborativa que no siguen la ISO 10218/2, pero que cumplen con otros estándares de seguridad, como podrían ser estándares específicos nacionales.

Por tanto, a pesar de que surgió para satisfacer los nuevos métodos de producción de la industria 4.0, en la actualidad la robótica colaborativa se está investigando e introduciendo cada vez más en el mundo comercial mediante la adaptación de los distintos tipos de robots a los estándares que la rigen.

2.1.1 Robótica móvil

El mundo de la robótica móvil [3] agrupa robots de diversas áreas, tales como drones, robots humanoides, zoomórficos y distintos tipos de vehículos, es decir, aquellos que son capaces de desplazarse por el entorno, en algunos casos de forma completamente autónoma.

Los robots móviles, dependiendo del área de trabajo a la que se destinan, deben enfrentarse a entornos que pueden ser de “interior” o “exterior”, lo cual condiciona, lógicamente, sus morfologías. En un medio de interior, la luz suele ser artificial, la temperatura estable, las superficies uniformes y el área de trabajo delimitada por paredes, mientras que en el entorno exterior la iluminación suele ser principalmente natural y puede variar mucho, al igual que la temperatura, las superficies suelen ser heterogéneas, el tipo de medio puede ser diverso (terrestre, acuático o aéreo) y el área de trabajo puede no estar claramente definida.

Al mismo tiempo, dependiendo del tipo de objetos presentes en el entorno [4] se habla de entorno “estructurado” o “no estructurado”. Si los objetos presentes son fijos (su forma y posición no varía) y cuentan con particularidades físicas (color, forma, etc.) asociables a figuras geométricas u objetos conocidos (mesas de trabajo, puertas, etc.) se habla de un entorno estructurado. Por el contrario, si los objetos del entorno son dinámicos o sus formas no son asociables a figuras geométricas u objetos conocidos, se trata de un entorno no estructurado.

Por otro lado, todos los robots móviles (sean colaborativos o no) contienen un sistema de locomoción condicionado y adaptado al entorno para el que van destinados (por ejemplo, los terrestres pueden tener uno basado en extremidades articuladas, cadenas, ruedas, etc.). Entre los robots de locomoción mediante ruedas para el medio terrestre, existe una amplia gama que se diferencian por el grado de maniobrabilidad con el que cuentan gracias a la disposición y tipo de

sus ruedas [5][6], conjuntamente con el sistema de tracción y dirección (incluye tracción y dirección en ejes independientes, un mismo eje o sobre todos los ejes). Teniendo todo esto en cuenta, la movilidad de los robots móviles se caracteriza por:

- **Tipo de ruedas:** depende de la estructura mecánica sobre la que están instaladas las ruedas.
    - **Rueda fija:** la orientación de la rueda no varía ya que su eje se encuentra fijo a la estructura del robot.
    - **Rueda orientable centrada:** permite un movimiento de rotación sobre el eje vertical que pasa por el centro de la rueda y que le otorga a la rueda la posibilidad de dirección y tracción.
    - **Rueda orientable descentrada:** conocida como rueda loca o castor, se parece a la rueda centrada orientable con la diferencia de que el eje vertical (sobre el que sucede el movimiento de rotación) no pasa a través del centro de la rueda.
    - **Rueda sueca:** consiste en una rueda omnidireccional compuesta por una serie de rodillos o cilindros con un eje de rotación de 45 grados respecto al plano de rotación de la rueda.
- **Configuración de las ruedas:** cada robot cuenta con un sistema de tracción y dirección relacionado con la mecánica, la disposición de las ruedas y los algoritmos de control de los motores de dichas ruedas. Combinando varios sistemas se pueden realizar diversas configuraciones [7]; las básicas son las siguientes:
    - **Diferencial:** es un sistema sencillo en el que se ubican dos ruedas motrices de forma simétrica en el mismo eje. Esta configuración le otorga al robot la posibilidad de girar sobre sí mismo, avanzar o retroceder e incluso seguir un recorrido parabólico. Puesto que el principal problema al que se enfrenta esta disposición es el de mantener el equilibrio, muchos robots incorporan ruedas de apoyo omnidireccionales para no limitar o interferir en el movimiento.
    - **Triciclo:** esta disposición suele contener dos ruedas motrices sobre un mismo eje y una tercera (orientable centrada) que actúa como directriz, responsable del movimiento y dirección.
    - **Ackerman:** consiste en un sistema más estable y complejo basado en la geometría de Ackermann para una distribución de cuatro ruedas (dos frontales y dos traseras), que se emplea, sobre todo, en automoción. Dicho sistema permite realizar diversas

configuraciones, como el diseño de ruedas frontales directrices y traseras motrices o el de ruedas frontales motrices y directrices (o 4x4).

Los robots móviles requieren percibir el entorno y/o su propio estado para navegar por el área de trabajo. Con ese objetivo se dotan de un sistema de sensorización [8], es decir, de un conjunto de sensores externos o "exteroceptivos" mediante los cuales reciben datos del mundo físico que les rodea (sensores de contacto, infrarrojos, ultrasonidos, cámaras de visión artificial...) y de otro grupo de sensores internos o "propioceptivos" con los que capta información sobre sí mismo (brújulas, giróscopos, *Inertial Measurement Unit* o IMU, sensores de medición de temperatura, nivel de baterías, etc.).

Asimismo, el sistema de navegación con el que se les dota utiliza la información recibida por el sistema de sensorización para llevar a cabo tres tareas esenciales: actualizar el mapa del entorno de trabajo, estimar tanto su posición como su orientación en dicho entorno y detectar los obstáculos potenciales en su camino. Estas tres tareas permiten, entre otros, que el robot genere trayectorias y las siga mientras evita los posibles obstáculos.

Los sistemas de navegación, con el objetivo de estimar la posición del propio robot en el espacio de trabajo según un sistema de coordenadas absoluto, utilizan diversos estimadores de posicionamiento [9], que se pueden agrupar en las siguientes categorías:

- **Estimadores explícitos:** son aquellos que realizan los cálculos de posicionamiento y orientación mediante medidas exactas obtenidas de las lecturas de los sensores, en vez de requerir procesos para la interpretación de la información del entorno. A su vez, los estimadores explícitos pueden dividirse en dos subgrupos con respecto a los datos en los que se basen:
  - **Medidas internas:** utilizan los sensores propioceptivos (giroscopios, acelerómetros, codificadores, etc.) para obtener la pose del robot. Un ejemplo de ello son los sistemas odométricos que obtienen el posicionamiento calculando el número de vueltas realizas por las ruedas del robot; otro ejemplo son los sistemas de navegación inercial, que la obtienen de las lecturas de los ángulos de orientación y las aceleraciones.
  - **Estaciones de transmisión:** empleadas por robots móviles diseñados para entornos de exterior, se componen con dos disposiciones distintas: la primera consiste en un receptor integrado en el robot y la segunda, que es la estación transmisora (o transmisoras, si son varias) se localiza en posiciones conocidas del entorno. Dichas

estaciones pueden ser fijas o móviles y suelen incorporar transmisiones de ultrasonido, láser o transmisores de radio-frecuencia (RF).

- **Estimadores con respecto a la percepción del entorno:** utilizan sensores tanto activos (sensores láser o de ultrasonido) como pasivos (cámaras o sensores de infrarrojos) para interpretar el entorno. Se diferencian los siguientes dos subgrupos:

  - **Estimadores mediante balizas:** se usan balizas o marcas en determinadas posiciones del área de trabajo del robot para que este, al reconocerlas mediante sus sensores, pueda determinar su posición y orientación en el entorno con respecto a dichas marcas. Las balizas pueden ser elementos naturales característicos de una zona, o bien pueden ser artificiales (como, por ejemplo, formas geométricas). Al mismo tiempo, se les puede asignar más información a dichas marcas aparte de su posición exacta en el entorno (según el sistema de referencia utilizado para obtener el mapa que lo compone), por ejemplo, mediante códigos de barras o códigos QR (*Quick Response*).

  - **Estimadores mediante mapas:** se utiliza el sistema sensorizado del robot móvil para crear mapas locales del entorno de trabajo, que se compara mediante otros procesos con el mapa global o de referencia almacenado en memoria. Esta técnica se conoce como *map matching*, y consiste en que el robot calcula su posición y orientación en el entorno real cuando los algoritmos encargados de la comparación entre el mapa local creado por el robot y el mapa de referencia encuentran una coincidencia, tal como muestra la ilustración 1.

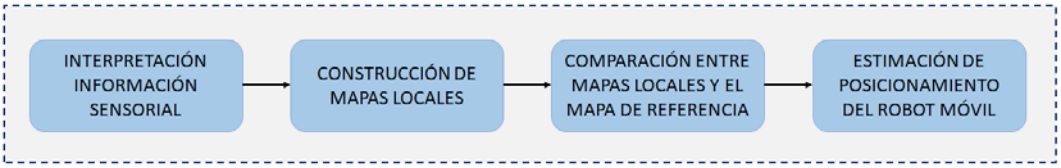

Ilustración 1: Estimación de la posición del robot móvil mediante mapas

El problema principal al que se enfrentan los robots móviles es, por un lado, generar de forma óptima las trayectorias o caminos para alcanzar sus puntos de destino y, por tanto, seguir con éxito dicha trayectoria teniendo en cuenta también la información del entorno recogida por sus sensores.

Para solventar dicho problema, los robots móviles contienen sistemas de comunicación (especialmente basados en conexiones inalámbricas, como *Wi-Fi* o *Bluethooth*, entre otros) y sistemas de control que interpretan la información recogida por el sistema de navegación y que los diferencian de otros robots y vehículos. Adicionalmente, algunos sistemas de control más

avanzados les confieren mayor capacidad de razonamiento o inteligencia, lo que les otorga una mayor autonomía del ser humano o de plataformas externas para su control. La arquitectura general del sistema de control [10] se caracteriza por organizarse en niveles que contienen los siguientes componentes:

- **Planificador global de trayectorias:** se encuentra en el nivel superior y se encarga de decidir las coordenadas de la posición final del robot y otros puntos intermedios (calcula la ruta o camino a seguir por el robot) dependiendo de la tarea asignada. En caso de que el camino calculado sea imposible de seguir (debido, por ejemplo, a una obstrucción) dicho componente permite redefinir la ruta. El mapa del entorno de trabajo o la información que utiliza este componente para calcular la trayectoria puede generarse fuera del robot y ser enviada a este o puede obtenerse de forma dinámica mediante la información recogida por los sensores y procesada por algoritmos para el mapeo y navegación simultáneos, o SLAM (*Simultaneous Localization and Mapping*).

- **Planificador local de trayectorias:** se localiza en el nivel intermedio. Este nivel trata de adecuar la velocidad, maniobrar o calcular pequeñas modificaciones en la ruta global (es decir, calcula una ruta local) para evitar los obstáculos (tanto estáticos como dinámicos) que se va encontrando el robot a medida que avanza por la ruta global calculada en el nivel superior, así como de avisar de las modificaciones al planificador global de trayectorias. Por tanto, el planificador local se encarga de controlar el robot dinámicamente con la información recibida por sus sensores, que a su vez le dan la información necesaria para tomar decisiones.

- **Sistema de control motriz:** se halla en el nivel inferior. Este componente se encarga tanto de interpretar los datos enviados desde el planificador local de trayectorias, como de generar y ejecutar las acciones de control sobre los distintos actuadores o motores que controlan los movimientos del robot.

El proceso general descrito que se realiza para controlar un robot móvil puede observarse en la ilustración 2.

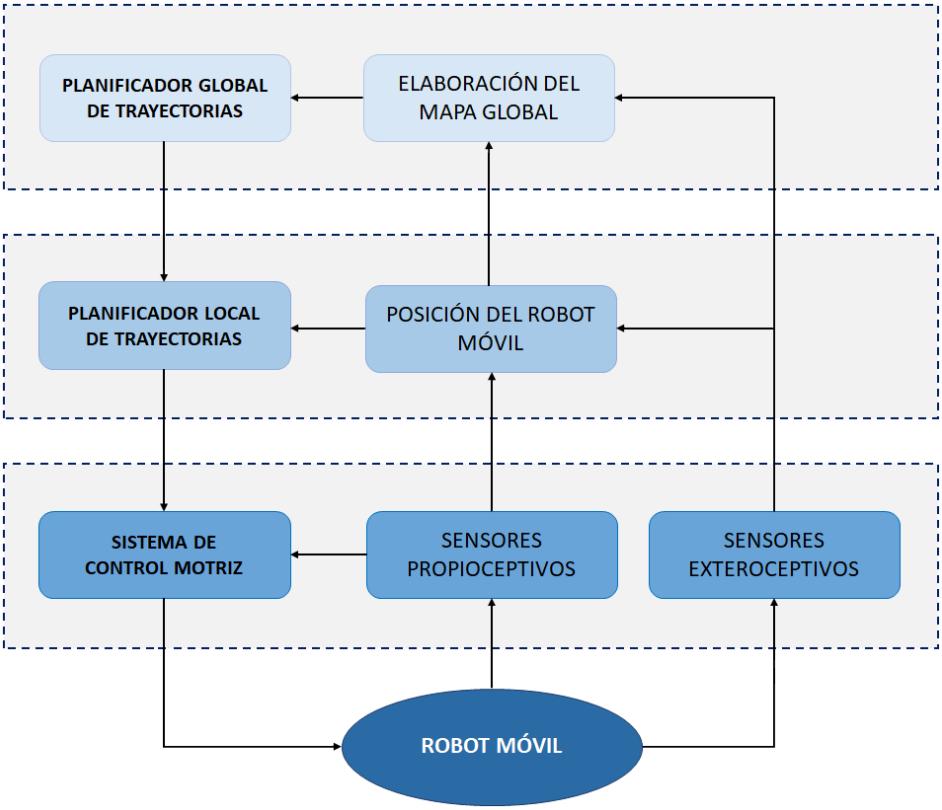

Ilustración 2: Arquitectura general del sistema de control de un robot móvil

El sector de la robótica móvil está evolucionando cada vez más al diseño de robots móviles capaces de desplazarse de forma autónoma por entornos no estructurados, así como a su adaptación para permitir que sean colaborativos, ya sea de forma completa o mediante la combinación de una base móvil con un robot colaborativo.

2.1.2 Robótica móvil sanitaria

Debido a las exigentes condiciones de trabajo, el cambio demográfico provocado por el aumento de la esperanza de vida y el envejecimiento de la población del primer mundo, cada vez se hace más patente la escasez de personal de enfermería cualificado y la necesidad de un mayor número de trabajadores de dicho sector para cubrir la demanda de las distintas áreas que requieren servicios sanitarios.

Algunas empresas entendieron estas carencias como una oportunidad de mercado y empezaron a tratar de cubrirlo mediante el desarrollo de una gran variedad de robots móviles sanitarios. Los robots sanitarios incluyen tanto los utilizados en cirugía como las máquinas para manipulación de personas y vehículos guiados automatizados. Dentro de la robótica sanitara destacan los robots móviles para asistencia sanitaria [11], clasificables en las siguientes categorías (combinables) según su finalidad:

- **Robots de asistencia:** en algunos casos realizan actividades similares a las del personal de enfermería y son destinados a la asistencia de personas, ya sea alimentaria, social o como apoyo en los hogares, centros de salud, residencias de la tercera edad, hospitales, etc.

- **Robots autónomos:** este grupo cuenta con robots que se desplazan de forma inteligente que se emplean generalmente para transportar objetos de diversa índole o para monitorizar.

- **Robots terapéuticos:** son capaces de servir de apoyo a los pacientes mediante acciones tales como el reconocimiento de emociones, la rehabilitación cognitiva o la prestación de compañía (ya sean humanoides o los zoomórficos –mascotas– como los *Hasbro's Joy for All* [12]).

En 1997 se diseñó un robot móvil de asistencia sanitaria para entornos de interior denominado *ROMAN* [13]. Dicho robot tenía inicialmente como objetivo proporcionar ayuda en la realización de las tareas más rutinarias y repetitivas en entornos sanitarios con el fin de que los humanos pudieran invertir ese tiempo en otras tareas. Sin embargo, la interacción humano-robot (HRI) de la época no se encontraba muy desarrollada y, por tanto, quedaba muy limitada. Como consecuencia se rediseñó para que, mediante una interfaz de control, pudiera asumir tareas o actividades básicas tales como buscar y traer objetos, distribuir comidas, limpieza de superficies y desinfección, etc.

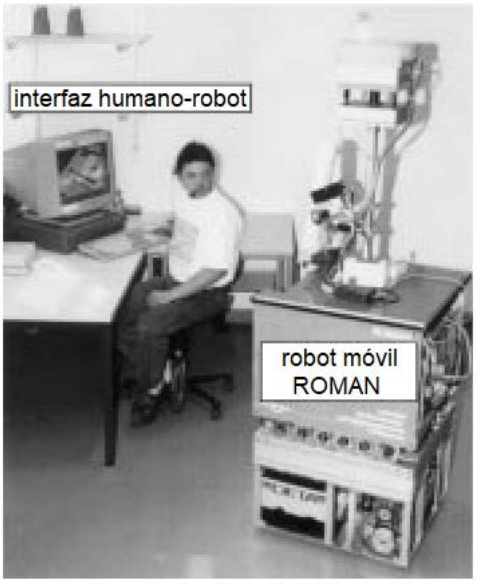

Ilustración 3: Robot móvil ROMAN

Fuente: *Design Issues of a SemiAutonomous Robotic Assistant for the Health Care Environment. Figure 15 (a)* [14]

Para que fuera capaz de llevar a cabo dichas actividades se añadieron a la base móvil del robot, o la plataforma de locomoción, un sistema de localización basado en balizas, para que fuera capaz de orientarse y navegar; los sensores necesarios propioceptivos y exteroceptivos para

percibir e interactuar consigo mismo y el entorno de trabajo (tales como un sensor de ultrasonido, reconocimiento de voz, una cámara para el reconocimiento de objetos y balizas, etc.); un brazo articulado con unas pinzas (con la función de "mano") con el objetivo de facilitar la manipulación de los objetos, así como la interfaz instalada en un ordenador central desde la que los humanos podían controlar al robot. Dichos componentes se pueden observar en la ilustración 4.

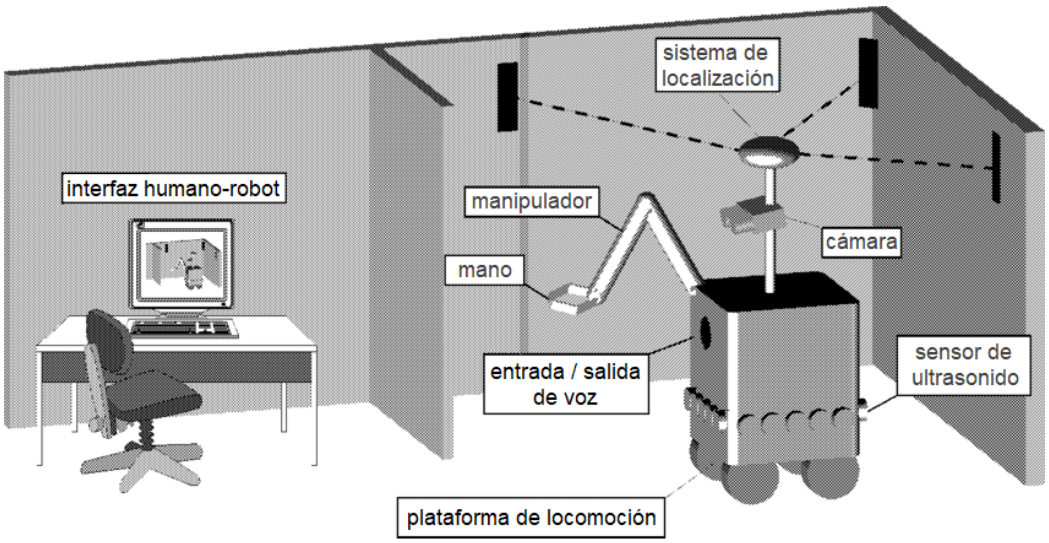

Ilustración 4: Componentes del robot móvil ROMAN

Fuente: *Design Issues of a SemiAutonomous Robotic Assistant for the Health Care Environment. Figure 1* [14]

Pocos años después, en 2002, el proyecto *Nursebot* [15], concebido en 1998 por un equipo de investigación multidisciplinario de tres universidades distintas, desarrolló en Estados Unidos el robot móvil *PEARL*. Este robot se diseñó para asistencia a personas de la tercera edad con un leve deterioro cognitivo en viviendas particulares, centros de día o residencias.

Ilustración 5: Robot móvil PEARL

Fuente: *Pearl - A mobile robotic assistant for the elderly. Figure 1* [15]

En 2018 se presentó el robot móvil de asistencia *Robot Service Assistant* dentro del proyecto *Service Robotics for Support with Personal Services* (SeRoDi) [16] de Alemania. Dicho robot se diseñó para cuidar la alimentación e hidratación de los residentes sanitarios en residencias, hospitales, etc. El robot cuenta con una plataforma sobre la base móvil diseñada con la finalidad de proporcionar alimentos o bebidas y su sistema de almacenaje dispone de una capacidad de hasta 28 bebidas o bocadillos, siendo además sencillo de reponer.

Ilustración 6: Robotic Service Assistant

Fuente: *The Robotic Service Assistant - Relieving the Nursing Staff of Workload. Figure 1 y 4* [16]

En 2019 un grupo de investigadores de Perú [17] desarrollaron un robot móvil de asistencia a la movilidad para ayudar a personas que sufren discapacidades físicas basándose en las interfaces diseñadas para la interacción entre el cerebro y las máquinas conocidas como *Brain Machine Interface* (BMI). Dicho robot consiste en una silla de ruedas eléctrica con un manipulador robótico 5DOF (es decir, un brazo articulado con cinco grados de libertad) que se controla mediante una combinación de señales enviadas por las manos derecha e izquierda y los datos recibidos de la interfaz gráfica con la que interactúa el paciente. Este robot móvil de asistencia ayuda a los pacientes a cumplir con sus actividades diarias, ya sean en su residencia particular o en el centro sanitario en el que se hallen.

Ilustración 7: Brazo robótico 5DOF en silla de ruedas eléctrica

Fuente: *Motor Imagery based Brain Machine Interface for a Mobile Robotic Assistant. Figure 1* [17]

Por otro lado, a los entornos sanitarios les resulta cada vez más factible y económico realizar inversiones en robots móviles colaborativos debido a que más empresas del sector de la robótica (como *Mobile Industrial Robots*, *Universal Robots*, *OnRobot*, *Omron* y *Robotnik*, entre otras) están creando robots móviles colaborativos y complementos (pinzas, garras, manos artificiales, etc.), económicos y eficaces para entornos de interior. Un ejemplo reciente de robótica móvil colaborativa introducida en un entorno hospitalario podría ser el Hospital de Jutlandia en Dinamarca [18]. En dicho hospital se emplea una flota de robots móviles *MIR100* de la empresa *Mobile Industrial Robots* para transportar medicamentos contra el cáncer y muestras de sangre.

2.1.3 Robótica móvil sanitaria para pandemias

Durante el brote de Ébola de 2015, las entidades *White House Office of Science and Technology Policy* y *National Science Foundation* [19] identificaron tres grandes áreas en las que la robótica podía marcar la diferencia con el objetivo de proteger al personal profesional expuesto a los patógenos en primera línea. La aparición en 2019 del brote de Coronavirus (COVID-19), convertido ya en una pandemia, se ha identificado una nueva área que se suma a las anteriores. En caso de pandemia, las áreas que puede cubrir el sector de la robótica son las siguientes:

- **Atención clínica:** agrupa la prevención, diagnóstico, detección de las enfermedades, atención a los pacientes y la gestión de la enfermedad. Por ejemplo, facilitando la atención médica no presencial, descontaminación del entorno de trabajo con luz ultravioleta (UV-C).

- **Logística:** entregas de objetos o materiales, gestión de desechos contaminados

- **Monitorización:** vigilancia de las constantes de los pacientes, de su entorno o del cumplimiento de cuarentenas.

- **Continuidad del trabajo:** mantenimiento de las funciones socioeconómicas y los trabajos, mediante fabricación y manipulación de objetos de forma remota, gestión de suministro eléctrico, tratamiento de aguas o residuos a distancia, etc.

Actualmente, el COVID-19 ha afectado a casi todos los continentes y ha producido un impacto en la economía global que ha evidenciado la necesidad de investigar en múltiples ámbitos, como combatir patógenos y epidemias. Por estas razones, algunos hospitales han empezado a utilizar robots móviles sanitarios al tiempo que se han iniciado diversas líneas de investigación para la atención de pacientes, logística de alimentos, suministros médicos, limpieza y desinfección de entornos sanitarios (hospitales, centros de salud, residencias, etc.).

En México, el Hospital San José en Monterrey [20], que es un hospital de TecSalud (Sistema de Salud Tecnológico), ha integrado el robot móvil *A-NÍMO* en su área de terapia para la asistencia

y consulta con pacientes de forma remota, evitando así el riesgo al contagio del personal médico. El robot colaborativo cuenta con una autonomía de ocho horas, un monitor desde el que los pacientes pueden ver al médico/a en cuestión, dos cámaras móviles que permiten observar al paciente y su entorno, seis micrófonos y un altavoz para las comunicaciones, así como una interfaz fácilmente instalable para el control de dicho robot.

Ilustración 8: Robot móvil A-NÍMO

Fuente: *Tecnología vs COVID-19 - TecSalud usa robot para tratar pacientes* [20]

En el Hospital Universitario de Burgos (España) se acaba de probar el robot móvil colaborativo denominado *ZenZoe* [21] desarrollado por las empresas *ASTI Mobile Robotics* y *BOOS Technical Lighting* para la desinfección mediante luz ultravioleta (UV-C) de los gérmenes y patógenos del COVID-19 (entre otros), tanto de las superficies y objetos como del ambiente. A pesar de que el *ZenZoe* se podría utilizar en múltiples entornos de interior no sanitarios (hoteles, comerciales, centros deportivos, etc.) se ideó inicialmente para ser utilizado en entornos sanitarios (buena muestra de ello es el propio nombre del robot, que homenajea a Isabel Zendal, enfermera que participó en la erradicación de la viruela en Latinoamérica y Filipinas, y Zoe Rosinach Pedrol, primera mujer doctora farmacéutica en España).

Ilustración 9: Robot móvil ZenZoe

Fuente: *ASTI Mobile Robotics y BOOS Technical Lighting desarrollan robot móvil para combatir el COVID19* [22]

El robot móvil *ZenZoe Robot* dispone de una plataforma sobre la base móvil que alberga la herramienta utilizada para desinfectar, una lámpara de radiación ultravioleta respetuosa con el medio ambiente que logra una reducción de los microorganismos en una habitación del 99.9 %.

Por otro lado, en el clúster español de innovación tecnológica *Secpho* [23], que agrupa 150 entidades interrelacionadas en el ámbito de la innovación científica y tecnológica, se encuentran recogidas diversas investigaciones actuales de España para la lucha contra el COVID-19 [24] que abarcan desde el desarrollo de robots como el de componentes (cámaras, sensores fotónicos, térmicos, infrarrojos, etc.).

En Secpho se realizan investigaciones como la llevada a cabo por la empresa *BCB* (con bases en España, Portugal y México) que, focalizada en los sistemas control y reconocimiento termográfico, ha diseñado el software *bcbTempScan* [25] que controla la temperatura de los seres humanos mediante la utilización de cámaras termográficas; a su vez, es capaz de detectar y generar una alarma sonora o visual en caso de identificación de contagio, lo cual lo convierte en un recurso especialmente útil en entornos sanitarios, aeropuertos, transporte, etc. Por otra parte, la investigación de la compañía emergente catalana *MTS Tech* desarrolla un robot móvil autónomo con luz UV-C [26] para la desinfección de diversos agentes patógenos en superficies y ambientes.

Recientemente la GVAInnova (Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital) ha decidido financiar un proyecto del instituto de investigación de Automática e Informática Industrial (ai2) de la *Universitat Politècnica de València* (UPV) [27] que trabaja sobre un robot móvil colaborativo para la desinfección de entornos sanitarios mediante luz ultravioleta que únicamente requiere de 30 segundos para desinfectar un radio de dos metros. Dicho robot es ligero, transportable gracias a su estructura de aluminio y resistente al deterioro y corrosión. Además, cuenta con un sistema remoto sobre el control de encendido y apagado.

Ilustración 10: Robot móvil del ai2 escogido por GVAInnova para combatir el COVID-19

Adicionalmente, el instituto ai2, junto con la empresa española *Robotnik* y otras instituciones europeas involucradas, están desarrollando el proyecto europeo ENDORSE, que investiga con robots móviles colaborativos para, entre otras tareas, reducir el contagio del personal sanitario con el COVID-19 mediante la atención remota de pacientes, tal como señala el periódico El Mundo [24].

Debido a las líneas de investigación abiertas este año en la robótica móvil para combatir al COVID-19, en los próximos meses presenciaremos seguramente un aumento exponencial de la aplicación e integración de componentes y robots móviles colaborativos tanto a nivel nacional como internacional, no solo en el sector sanitario sino también en multitud de sectores comerciales e industriales.

2.2 Ámbito de trabajo

Me encuentro trabajando en el Instituto de Automática e Informática Industrial (ai2) de la *Universitat Politècnica de València* (UPV) contratada en el proyecto "Aplicaciones industriales y servicio con robots colaborativos" de las ayudas para la promoción de empleo joven e implantación de la Garantía Juvenil en I+D+i (Investigación, Desarrollo e innovación) en el Subprograma Estatal de Incorporación, del Programa Estatal de Promoción del Talento y su Empleabilidad en I+D+i, en el marco del Plan Estatal de Investigación Científica y Técnica y de Innovación 2017-2020, del Plan Estratégico de Subvenciones 2018-2020 del MINECO (Ministerio de Economía, Industria y Competitividad).

El instituto ai2 [28] agrupa las siguientes cinco áreas de investigación: control de procesos; informática gráfica y multimedia; informática industrial; robótica; y visión por computador. Dichas áreas de investigación abarcan trabajos en el sector de la alimentación, la salud y la calidad funcional de vida, la sostenibilidad y la energía, los procesos industriales (incluida la industria 4.0), y, por último, la movilidad y la logística.

Dentro del área de investigación de robótica, el instituto ai2 participa en el desarrollo del proyecto europeo ENDORSE (*Safe, Efficient and Integrated Indoor Robotic Fleet for Logistic Applications in Healthcare and Commercial Spaces*) [29] financiado por el programa *Research and Innovation Staff Exchange* (RISE) *Horizon 2020* de la Unión Europea bajo el acuerdo de subvención de *Marie Skłodowska-Curie Action* (MSCA).

Ilustración 11: Robots móviles sanitarios ENDORSE Proyect

Fuente: Web *ENDORSE Project* [29]

La idea del proyecto ENDORSE [30] surgió como consecuencia de la escasez de servicios logísticos de robótica que se adaptara a los entornos de interior comerciales como, por ejemplo, oficinas, hoteles, hospitales, centros de salud, etc. Dicha carencia se debe en gran medida a que los robots diseñados para el mundo industrial no son aplicables a los espacios comerciales, ya que cuentan con diferentes limitaciones y especificaciones. Los entornos industriales (incluidos los almacenes industriales) se caracterizan por estar muy estructurados, lo que los presenta como medios predecibles y, por tanto, ideales para predefinir de forma sencilla caminos por los que se puedan mover los robots de forma eficiente reduciendo o evitando la interacción humana. Sin embargo, en los entornos comerciales los espacios son menos estructurados, las especificaciones arquitectónicas pueden variar mucho dependiendo del edificio o planta y la interacción humana es mucho más frecuente.

La propuesta del proyecto ENDORSE se centra en el diseño y desarrollo de un sistema robótico logístico integrado que permita servicios logísticos para entornos interiores hospitalarios eficientes, rentables y seguros. Una vez finalizado el desarrollo dicho sistema se pondrá a prueba en el centro para la tercera edad *Fundació Ave María* en Sitges, Barcelona (España).

Los objetivos principales del proyecto son los siguientes:

- Desarrollar un sistema de robótica logística rentable para entornos sanitarios de interior, es decir, con navegación multi robot en un espacio de interior sin infraestructura. Un punto clave de esta meta es minimizar el tiempo de instalación y el coste del sistema.

- Mejorar la interacción humano-robot (HRI), incluyendo los casos especiales en los que los robots deben resolver puntos muertos que involucren a humanos, o navegar con mayor seguridad y haciendo un uso eficiente de los recursos espaciales compartidos (por ejemplo, cuando un robot deba cruzar un pasillo abarrotado de gente).

- Desarrollar métodos robustos de localización sin infraestructura, para la localización, supervisión y control de la flota robótica, que minimice el tiempo de instalación.

- Crear un nuevo servicio basado en la nube (*cloud-base service*) a partir del software desarrollado en ENDORSE con el fin de facilitar su integración con soluciones de software corporativas (ERP, CRM, etc.) cumpliendo los requisitos de seguridad de datos GDPR (*General Data Protection Regulation*).

- Generar arquitecturas de hardware reconfigurables y modulares para que diversos módulos puedan añadirse o modificarse fácilmente.

- Desarrollar un módulo de diagnóstico electrónico móvil integrado (*E-diagnostic Module*), equipado con sensores y dispositivos no invasivos, conjuntamente con su interfaz de registros electrónicos de salud o *Electronic Health Records* (EHR), para facilitar un servicio de diagnóstico electrónico móvil.

Las entidades colaboradoras en el proyecto ENDORSE que se encargan de desarrollar y lograr los objetivos anteriormente descritos aprovechando las competencias complementarias en los campos de robótica, visión artificial, logística, IT (*Information Technology*) y diseño de productos, son las siguientes:

- **Universidades:** *Univeristat Politècnica de València* (UPV), por parte de España; *Université d'Orléans* (UORL) y el *Laboratoire* PRISME, de Francia; *Institute of Communication and Computer System* (ICCS), de Grecia; y, por último, *University of Cyprus* (UCY), de Chipre.

- **Empresas:** *Robotnik* (ROBO, España), *StreamVision* (STR, Francia), *SingularLogic* (SLG, Grecia) y *Citard Services Ltd* (CITARD, Chipre).

Por otra parte, el proyecto está divido en los siguientes paquetes de trabajo o *Work Packages* (WP):

- **WP1:** *User requirements, Use Cases and Technical Specifications*.
  Comprende el análisis de las partes interesadas o *stakeholders* de ENDORSE, el diseño de los casos de uso, los indicadores clave de rendimiento o *Key Performance Indicators* (KPIs)

específicos para la asistencia sanitaria, la definición de requisitos y especificaciones técnicas del sistema y el diseño de la arquitectura general del mismo.

- **WP2:** *Fleet Management System: Task Programming and Dynamic Vehicle Routing*.
  Incluye el desarrollo, verificación, validación e implementación de algoritmos diseñados para asignar tareas a los robots, calcular rutas dinámicamente, controlar el de tráfico de los robots y supervisar la ejecución de tareas de la flota.

- **WP3:** *Strategies for Safe and Efficient Navigation and Human Robot Interaction*.
  Agrupa tanto el desarrollo de algoritmos capaces de reconocer humanos y predecir sus movimientos para la mejora de la navegación de los robots, como los algoritmos para mejorar la interacción del sistema de navegación del robot con los humanos, incluida la comunicación de mensajes y el reconocimiento facial. Por otra parte, también forma parte de este paquete de trabajo el diseño e implementación de algoritmos de control de los robots para situaciones especiales y de localización y mapeo simultáneos o SLAM (*Simultaneous Localization and Mapping*), para obtener una localización local y global más precisa y estable.

- **WP4:** *API framework, standards and cloud environment for integrating healthcare and logistic robots*.
  Se centra en el diseño y desarrollo de interfaces de programación de aplicaciones, API (*Application Programming Interface*), para cubrir todos los posibles casos de uso; el diseño y desarrollo de una aplicación de interfaz gráfica de usuario basada en la nube para la programación y supervisión del sistema robótico propuesto en el entorno sanitario; y el diseño y desarrollo de una estructura de soporte (para el módulo de diagnóstico) implementado en una base robótica.

- **WP5:** *Development of Hardware Modules and Robotic System Integration.*
  Se ocupa de la adaptación de la base del robot móvil *RB-1 Base* para aplicaciones de logística en interiores y del desarrollo del módulo de soporte de *e-Diagnostic*, así como de la integración y de la prueba del sistema.

- **WP6:** *Pilot Operation and Evaluation.*
  En este paquete se debe realizar la configuración de la prueba piloto, conjuntamente con la planificación y ejecución de pruebas de validación del sistema.

- **WP7:** *Dissemination, Training and Explotation.*
  Consiste en la elaboración y ejecución de pruebas, demostraciones y divulgación de los resultados (por ejemplo, mediante artículos).

- **WP8:** Project Management

  Comprende las actuaciones y decisiones para la gestión efectiva de todos los aspectos del proyecto.

Asimismo, cada paquete de trabajo está compuesto por varias tareas y entregables que llevan a cabo los diferentes participantes del proyecto. Parte de estos paquetes de trabajo y tareas están concebidos para el desarrollo de las aplicaciones y módulos que darán servicio en la nube y harán posible el control sobre los robots y los diagnósticos desde un ordenador central del centro sanitario, mientras que el resto se implementarán en cada uno de los robots móviles *RB-1 Base*. La arquitectura funcional que propone la solución ENDORSE al finalizar los diversos paquetes de trabajo se puede observar en la ilustración 12.

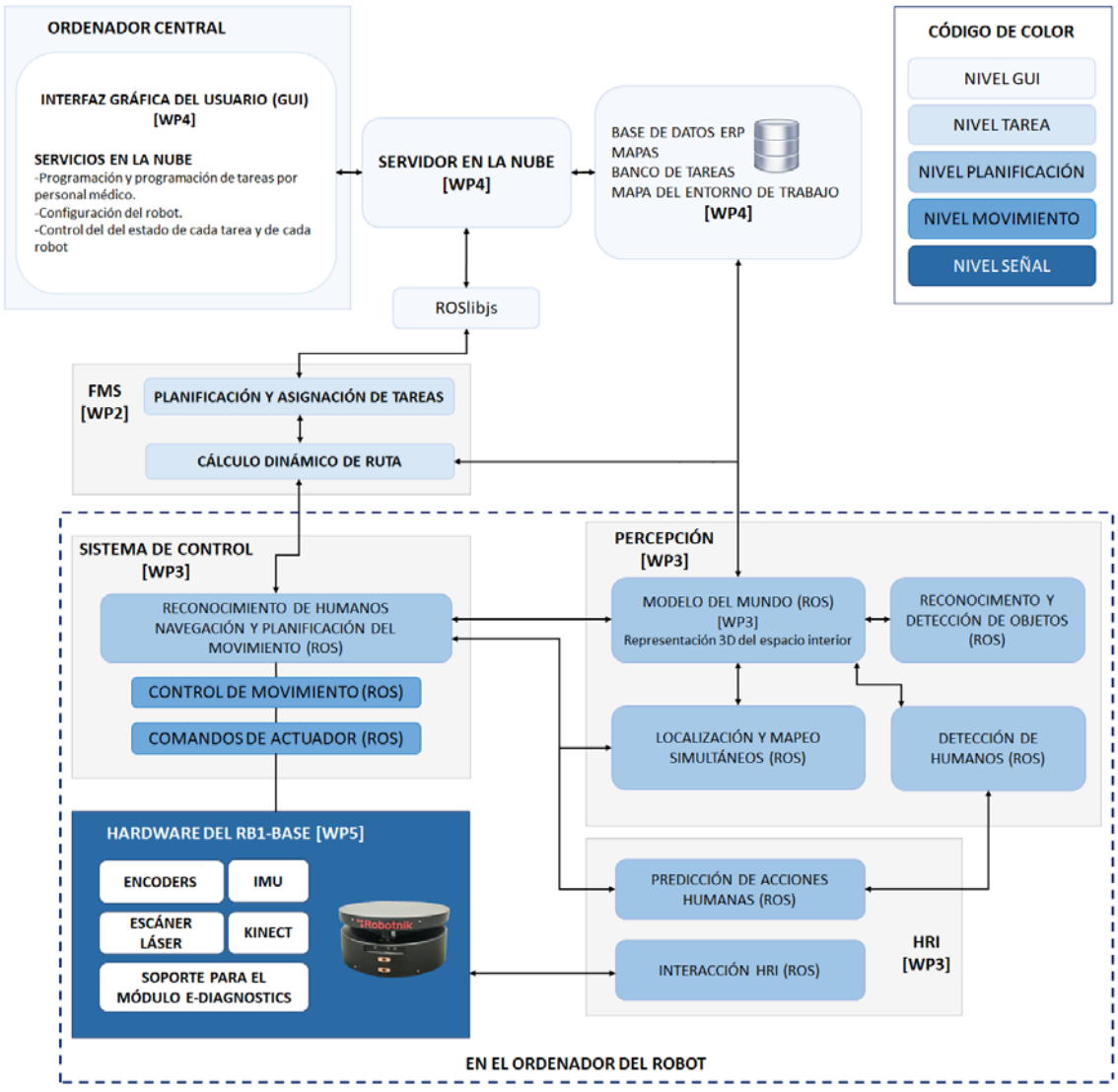

Ilustración 12: ENDORSE Arquitectura funcional

A día de hoy, el paquete de trabajo WP1 ha sido completado con las especificaciones de los requerimientos y casos de uso del proyecto. El WP2, con el módulo para el sistema de gestión

de flotas o *Fleet Management System* (FMS) para los robots móviles utilizados en el proyecto, se encuentra en la última fase de desarrollo. Lo mismo ocurre con el WP5, que ya ha realizado la adaptación de la base del robot móvil para el módulo de soporte de *e-Diagnostic*.

El instituto ai2 de la UPV se encarga, entre otras labores, de resolver la tarea de *Human-aware path planning algorithms* perteneciente al WP3, en la que se deben implementar algoritmos que mejoren la evitación de colisiones de los robots móviles con obstáculos y personas, así como la detección o resolución de bloqueos y situaciones especiales. Es en este punto en el que se centra el presente Trabajo Fin de Máster.

2.3 Análisis del problema con los robots móviles

Actualmente en el proyecto ENDORSE, y tal como se ha mencionado con anterioridad, el paquete de trabajo relativo a la aplicación central para el control de la flota de robots denominada *Fleet Management System* (FMS) ya está prácticamente desarrollado (si bien sigue abierto a mejoras y modificaciones). Entre muchas otras tareas, el FMS se encarga de la restricción de áreas y limitación de robots en determinadas zonas del mapa del entorno de trabajo, de la generación de trayectorias, de la asignación de tareas a los diversos robots y de las prioridades de dichas tareas.

Por tanto, el FMS es responsable del control del tráfico de robots móviles, de asegurarse de que llegan a sus destinos y verificar que cumplen así sus respectivas tareas de forma óptima. Sin embargo, en ocasiones pueden darse situaciones anómalas no contempladas por el FMS que causen un recálculo de algoritmos e incremento en el intercambio de mensajes que o bien pueda llegar a saturar dicha aplicación de control central o bien provoque una respuesta fuera de tiempo; en ambos casos, los robots implicados pueden acabar con problemas en su navegación si están desplazándose por el entorno, ya que se vuelven susceptibles de desembocar en una situación de conflicto (ya sea por bloqueo o colisión).

Un ejemplo de una situación anómala que alteraría el sistema se da cuando en un área restringida a un número determinado de robots se produce una aglomeración inesperada de gente, o un paciente se desmaya en medio de un pasillo e impide la circulación segura de los robots por esa área. Los robots implicados detectarían los obstáculos que les bloquean el paso y todos comunicarían la situación al FMS, que debería reajustar los desplazamientos de todos los robots implicados con la desventaja de que las nuevas trayectorias no serían dinámicas, de manera que si los robots siguen sin resolver el conflicto provocado por los obstáculos y, por tanto, sin poder cruzar o llegar al destino marcado por la aplicación central, seguirían consultando sin cesar al FMS.

Estas situaciones impredecibles que pueden darse en su entorno natural de trabajo y que pueden llegar a afectar gravemente al FMS podrían llegar a tener consecuencias también en algunos de los procesos necesarios para realizar la tarea de *Human-aware path planning algorithms* (de la que el instituto ai2 es responsable), especialmente en lo que respecta a los algoritmos relacionados tanto con la evitación de obstáculos como con la detección y resolución de bloqueos.

Por otro lado, las comunicaciones desarrolladas hasta este momento entre la aplicación de control de los robots móviles y los robots están centralizadas, es decir: el FMS envía a los robots toda la información que deben conocer, mientras ellos a su vez le devuelven (con una determinada frecuencia) la actualización de sus respectivos estados, con lo que los robots actualmente no se comunican entre sí; toda la comunicación pasa por la aplicación central.

2.4 Propuesta de solución

Como ha podido observarse es necesario tomar medidas con respecto a los conflictos que pueden producirse en la navegación de los robots, ya que tal como está planteada actualmente la solución del proyecto ENDORSE los robots dependen enteramente para cumplir sus objetivos del correcto funcionamiento y de la respuesta a tiempo frente a cualquier escenario por parte de la aplicación central de control de flota.

La propuesta de solución que ofrece este proyecto consiste en abordar de forma descentralizada al FMS, dentro de las áreas potencialmente conflictivas del entorno de trabajo, los problemas que pueden darse en la navegación (tales como, evitación de obstáculos, detección y resolución de bloqueos) provocados por circunstancias eventuales. Es decir, en lugar de que se produzca una sincronización global (con todas las comunicaciones y cálculos a través de la aplicación central) serán los propios robots móviles los que se encarguen de forma autónoma de hacer frente al escenario adverso mediante la coordinación e intercambio de información u órdenes entre sí. En esta solución, la aplicación central intervendrá en un primer momento para avisar y enviar la información necesaria a los robots implicados que están entrando en una zona potencialmente conflictiva y, posteriormente, en el momento en que cada robot implicado abandone dicha área.

Para llevar a cabo dicha solución se requiere establecer una comunicación sin sincronización global (descentralizada) con el objetivo de que los robots conectados a una misma red sean capaces de comunicarse entre sí para coordinarse. Además, se necesita implementar un algoritmo basado en el de Markus Jäger y Bernhard Nebel [31] para coordinar y resolver los conflictos en la navegación de los robots de forma descentralizada.

CAPÍTULO 3
Herramientas de trabajo

En este capítulo, se describirán los elementos y tecnologías utilizados para realizar este trabajo. En primer lugar, se explicarán los componentes físicos empleados y, a continuación, las tecnologías empleadas para la modificación y desarrollo del sistema interno del robot.

3.1 Elementos hardware

En este apartado, se detallarán las características de los robots móviles empleados en el proyecto.

3.1.1 RB-1 Base

El *RB-1 Base* es una plataforma móvil creada por la empresa valenciana *Robotnik* [32] para la robótica de servicio. Su objetivo es centrarse en aplicaciones para entornos de interior. La base permite transportar diversos materiales o cargas y ofrece la integración con otros sistemas como, por ejemplo, un brazo robótico, una plataforma para fines médicos o una unidad de elevación para cargar o arrastrar objetos o cargas.

Ilustración 13: RB-1 Base

El *RB-1 Base* cuenta con dos tipos de sensores que le permiten la detección de obstáculos. El primero es un sensor RGBD, es decir, una cámara. El segundo, un láser ubicado en la zona frontal del robot móvil utilizado para la navegación y localización. Externamente, el robot se compone de los elementos que se detallan a continuación:

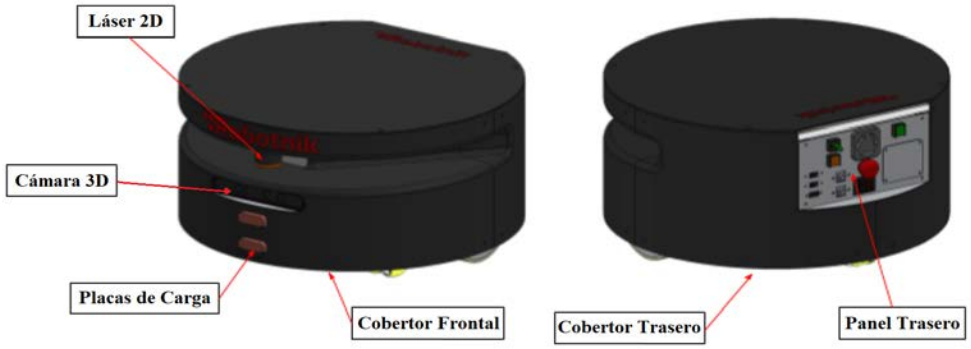

Ilustración 14: Componentes externos del RB-1 Base

- **Cobertor Frontal y Trasero:** Consisten en planchas de plástico que protegen el interior del robot de la entrada de polvo y agua. Se sujetan al robot mediante tornillos.

- **Placas de Carga:** Sirven para cargar las baterías del robot cuando este se conecta desde una estación de carga.

- **Láser 2D:** Se ha incorporado en los robots el sensor *Hokuyo UST-10LX*, que permite la observación del entorno con distancias de 10 a 20 m, con un rango de visión de 270 grados.

- **Cámara 3D:** El sensor *RGBD Orbbec Astra S* permite visualizar en tres dimensiones el entorno del robot en un rango de 0.4 a 2 m, dispone de dos micrófonos y cuenta con un ángulo de visión en horizontal de 60 grados y 49.5 grados en vertical. A su vez este dispositivo admite la realización de varios tipos de funciones tales como: lectura de códigos QR, el reconocimiento del cuerpo humano, medición tridimensional, percepción del entorno y reconstrucción de mapas tridimensionales, etc.

- **Panel trasero:** Cuenta con diversos elementos tal como puede observarse en la ilustración 15, que son: el botón de emergencia, que desconecta la energía y para el robot en caso de peligro o de que dicho robot se descontrole; el selector de energía principal, encargado de suministrar energía (o no) al sistema del robot; un botón de encendido o apagado de la unidad central de procesamiento (CPU) del robot, que inicia el ordenador interno del robot; una cubierta de cables; el conector de carga trasero, que permite cargar el robot de forma manual si no se dispone de la estación de carga; dos puertos USB 2.0, para conectar dispositivos con la CPU del robot; dos puertos de Ethernet conectado al rúter interno del robot, el de la parte superior para una red WAN y el segundo para una red LAN; un puerto HDMI, que admite la conexión de un monitor al robot en caso de ser necesario; el cobertor de la caja de fusibles; y por último, el ventilador que ayuda a regular la temperatura interna del robot expulsando el aire caliente.

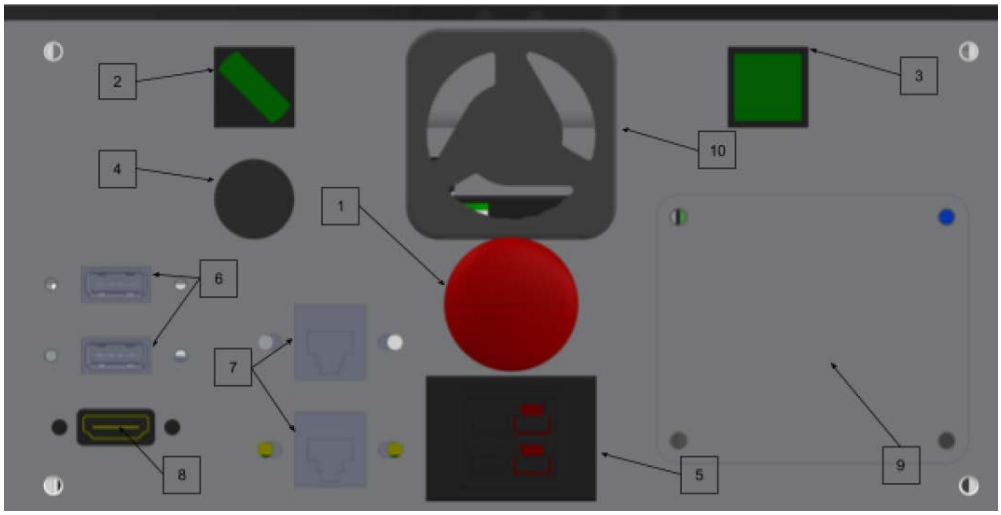

Ilustración 15: Panel trasero del RB-1 Base

- **Ruedas:** El *RB-1 Base* dispone de una configuración diferencial con dos ruedas motrices simétricas y centradas en el eje, además de tres ruedas de apoyo omnidireccionales para mantener el equilibrio de la plataforma del robot.

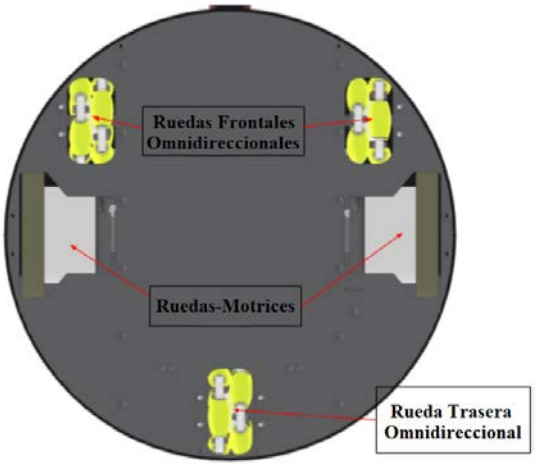

Ilustración 16: Ruedas del RB-1 Base

Las especificaciones de la base de este robot móvil son las siguientes:

- Mide 251 mm de altura y el perímetro de su base circular es de 500 mm.
- Pesa 30 kg y la carga máxima que puede soportar es de 50 kg.
- La velocidad máxima a la que puede desplazarse es de 1.5 m/s con una pendiente máxima del 8 %.
- La autonomía de la carga de la batería es de diez horas de movimiento continuo sin carga en la plataforma.

- Cuenta con protección IP51, que le proporciona cierta protección contra el polvo: no aísla el interior completamente pero la cantidad que entra no interfiere en su correcto funcionamiento. Además, también le confiere protección frente al goteo de agua.

- Su sistema de tracción es diferencial y dispone de dos servomotores de 250 W para mover las ruedas de la base.

3.2 Elementos software

En este apartado se describen las herramientas y lenguajes utilizados para el desarrollo del proyecto.

3.2.1 ROS

ROS (*Robot Operating System*) o Sistema Operativo Robótico [33], consiste en una plataforma software, es decir un marco de referencia o *framework*, que permite el desarrollo de software para robots. Al mismo tiempo, es de código abierto y compatible con diversas plataformas robóticas. ROS surgió en 2007, con el nombre *switchyard* en el Laboratorio de Inteligencia Artificial del instituto de investigación *Willow Garage* de la Universidad de Stanford bajo la licencia BSD, con el propósito de dar soporte al proyecto del Robot con Inteligencia Ambiental de Standford (STAIR2).

Al contrario de lo que podría pensarse por su nombre, ROS no constituye un sistema operativo en sí sino que proporciona los servicios de uno mediante un conjunto de convenciones, librerías y herramientas que proveen la abstracción del hardware, controladores de dispositivos de bajo nivel, la gestión de paquetes, implementación de funcionalidades de uso común y la comunicación entre procesos por medio de mensajes, tanto de forma síncrona como asíncrona. Además, provee de herramientas y librerías para crear, consultar, escribir y ejecutar múltiples nodos en paralelo.

La filosofía de ROS consiste en que sus programas estén constituidos por código abierto, libres, escalables, reutilizables e integrables con el software existente de robótica actual y futuro. Por estas razones, su sistema se desarrolló y creó paquetes para el uso de plataformas de software libre como Player, OpenCV, Gazebo y otras.

ROS [34] presenta otros rasgos interesantes, como su carácter multilenguaje (ya que se puede programar mediante varios lenguajes de programación, tales como, Python, C++ y Lisp) o que no siendo una plataforma software de tiempo real sí admite integrar ROS en código de tiempo real. Además, es ligero, ya que el código es empaquetado en ejecutables independientes de

tamaño reducido dentro del marco de referencia de ROS y puede usarse en diversos robots sin ser modificado, es decir, permite ser reutilizado.

Por otro lado, el sistema de archivos de ROS se descompone en dos niveles.

- **Packages:** Los paquetes son el nivel inferior dentro de la estructura de ROS y pueden contener diversos elementos, como ejecutables, herramientas, librerías, mensajes, servicios o modelos.

- **Stacks:** Las pilas consisten en agrupaciones de paquetes complementarios con el fin de crear una librería de alto nivel para diferentes cálculos o comportamientos en los robots.

ROS se basa en una arquitectura de grafos, en la cual son los nodos los que realizan el procesamiento al recibir, enviar y multiplexar los mensajes recibidos por otros nodos, planificadores, estados y dispositivos de bajo nivel, entre otros. Dicha arquitectura de grafos se encuentra conectada mediante una topología de red Punto-a-Punto (P2P), que permite a un nodo del sistema del robot comunicarse con cualquier otro nodo, síncrona o asíncronamente, dependiendo de la necesidad. Dicha topología de red se denomina *Computation Graph* o Grafo de Computación.

Los conceptos de ROS que intervienen en el Grafo de Computación de diversas maneras, son los siguientes:

- **Nodes:** Los nodos son los procesos ejecutados en ROS que realizan algún cálculo, publicación o suscripción. Para crear un nodo en ROS debe utilizarse la librería *rospy*.

- **Master:** El ROS Master proporciona el servicio de registro y declaración del nombre de los diversos componentes del *Computation Graph* y permite la búsqueda de estos. Sin él, los nodos no podrían comunicarse, intercambiar mensajes o invocar servicios o acciones.

- **Parameter Server:** El servidor de parámetros forma parte del máster y se encarga de que los datos puedan ser almacenados mediante clave en una ubicación determinada.

- **Host:** Ordenador o máquina virtual conectado a la red en la que se está ejecutando ROS. Se identifica mediante una dirección IP (*Internet Protocol*).

- **Messages:** Los nodos se comunican mediante el intercambio de mensajes. Un mensaje consiste en una estructura de datos concreta (enteros, booleanos, cadenas de caracteres, etc.). En ROS se pueden utilizar estructuras predefinidas o crear nuevas.

- **Topics:** Un tema consiste en un canal de comunicación asíncrono, basado en el protocolo publicador-suscritor. Un nodo o varios pueden publicar información en un mismo tema, a la vez que uno o más nodos pueden suscribirse y leer dicha información.

- **Services:** Un servicio proporciona un método de comunicación síncrona basado en el protocolo de petición-respuesta (*Request-Response*) entre un nodo cliente que realiza la solicitud a un nodo servidor, que ofrece el servicio o respuesta a dicha petición.

- **Actions:** es un tipo servicio que utiliza un método de comunicación asíncrono entre un nodo cliente (que realiza la solicitud de la acción) a un nodo servidor (que ofrece el servicio o respuesta a dicha petición). Sin embargo, mientras el nodo servidor de la acción obtiene la respuesta a la petición del nodo cliente, el nodo servidor envía mensajes al nodo cliente comunicándole la situación. Por tanto, en las acciones intervienen tres partes: el *goal* u objetivo, que envía el nodo cliente de la acción, al nodo servidor; el *feedback* o retroalimentación, y el *result* o resultado, enviados por el nodo servidor de la acción, al nodo cliente.

Por otra parte, el funcionamiento de ROS se basa en la arquitectura de comunicación publicador-suscriptor. Existe un primer elemento que actúa como punto o canal de entrada al sistema y se denomina *master*. Dicho *master* actúa como un servidor de nombres (DNS) pues almacena información de registro de los nodos. Cada nodo del sistema notifica al *master* la información sobre dónde publica o a quién está suscrito. A medida que los nodos se van comunicando con el *master*, se van creando las conexiones del *Computation Graph* o Grafo de Computación del sistema y dichos nodos ya pueden empezar a recibir o intercambiar información sobre otros nodos registrados. Asimismo, cuando los nodos cambian la información de registro notifican al master dichos cambios, lo que permite crear conexiones dinámicamente a medida que se ejecutan nuevos nodos. El protocolo más común utilizado en ROS para realizar dichas comunicaciones se denomina TCPROS, que utiliza sockets TCP/IP estándar.

Por tanto, la arquitectura que utiliza el funcionamiento de ROS permite la operación desacoplada, según la cual los nombres son los medios principales por los cuales se pueden construir sistemas más grandes y complejos. Los nombres cuentan en consecuencia con un papel muy importante en ROS: los nodos, los temas, los servicios, las acciones y los parámetros ostentan nombres. Cada librería cliente de ROS admite la reasignación de nombres a través de la línea de comandos, lo que significa que un programa compilado puede reconfigurarse en tiempo de ejecución para operar en una topología de *Computation Graph* diferente. Con lo que si fuera necesario realizar alguna modificación en el sistema o añadir elementos nuevos (como, por

ejemplo, añadir un láser nuevo) tan solo habría que reasignar los nombres involucrados en ese cambio.

En el desarrollo de este trabajo final de máster, se ha utilizado ROS para el desarrollo del software introducido en los robots móviles *RB-1 Base*, ya que el sistema de estos funciona mediante la versión *Kinetic* de ROS. El sistema ROS de los robots se organiza principalmente mediante los siguientes paquetes:

- **rb1_base_common:** Está formado por los paquetes que contienen los modelos de la plataforma del *RB-1 Base* (URDF, *Unified Robot Description Format*), mensajes, los algoritmos, librerías y ejecutables para la localización, navegación y simulación.
    - **rb1_base_description:** El URDF, mallas y otros elementos necesarios en la descripción del robot se encuentran contenidos aquí.
    - **rb1_base_control:** Este paquete contiene ejecutables y archivos de configuración para los controladores de las articulaciones del robot.
    - **rb1_base_pad:** Contiene los archivos necesarios para poder conectar al robot dispositivos externos como, por ejemplo, mandos de la PS4, PS3, entre otros. Este paquete proporciona también los ejecutables que posibilitan el control del robot con dichos dispositivos externos.
    - **rb1_base_localization:** Alberga diversos algoritmos y programas utilizados para conocer y manipular la localización del robot.
    - **rb1_base_navigation:** Agrupa los archivos de configuración necesarios para que sea posible la navegación del robot.
- **rb1_base_robot:** Engloba los paquetes encargados de ejecutar el sistema y los motores del robot.
    - **rb1_base_bringup:** En este paquete se encuentran todos los lanzadores, archivos ejecutables, que inician el sistema del *RB-1 Base*.
    - **rb1_base_controller:** Este componente es responsable de la manipulación y lectura de las articulaciones de los motores del robot.

3.2.2 Python

Python [35] es un lenguaje de programación multiplataforma, interpretado y dinámico de libre distribución. Simultáneamente, se trata de un lenguaje multiparadigma, pues entre otras características admite la programación orientada a objetos y la programación imperativa. Dichas características hacen de Python un lenguaje fácil de aprender y utilizar, así como ideal para el desarrollo de aplicaciones de distintas áreas.

Por otra parte, Python dispone de un paquete o librería cliente denominado *rospy* que permite a los desarrolladores interactuar rápidamente con ROS, ya sea mediante la interacción con temas, servicios, acciones o parámetros. El diseño de dicho paquete favorece la velocidad de implementación, lo que reduce el tiempo de desarrollo sobre el rendimiento en tiempo de ejecución. Por tanto, *rospy* favorece la utilización de Python para el desarrollo de programas que deban usar ROS.

En los programas realizados en este TFM, se ha utilizado Python (versión 2.7.17) para trabajar con ROS debido a la eficiencia de la implementación del paquete *rospy*, además de su facilidad de uso y la notable reducción del tiempo necesario de desarrollo de código que aporta Python en comparación con, por ejemplo, C++.

3.2.3 MQTT

MQTT (*Message Queue Telemetry Transport*) [36] es un protocolo de transporte de mensajes en la red mediante la estructura publicador-subscriptor que se emplea para comunicar diversos dispositivos y se basa en el protocolo TCP/IP. Asimismo, es un estándar abierto de OASIS (*Organization for the Advancement of Structured Information Standards*) que sigue conjuntamente el estándar de la ISO (*International Organization for Standardization*) ISO/IEC 20922.

Las características principales de MQTT son las siguientes:

- Funciona sobre otros protocolos de red que faciliten conexiones bidireccionales y sin pérdida de datos.
- Ofrece independencia de la aplicación mediante la utilización de mensajes *broadcast* para publicar y suscribir datos.
- Dispone de una estructura sencilla que permite enviar el menor número de bytes por mensaje, disminuyendo el ancho de banda necesario en la red para el transporte de dichos mensajes.

- Es fiable, pues ofrece la función QoS (*Quality Of Service*) dedicada a informar del estado de la comunicación. Esta función permite seleccionar entre tres modalidades de calidad de servicio.

    - **Como máximo una vez:** el mensaje se envía solo una vez pero se puede producir pérdida de datos ya que ni el remitente ni el receptor toman medidas para asegurarse de que el mensaje llega a su destino. Esta opción es útil en aplicaciones en las que la recepción de los mensajes no es crítica (por ejemplo, la lectura de la temperatura ambiente del sensor del móvil) y en las que se envían mensajes con mucha frecuencia.

    - **Al menos una vez:** el remitente sigue enviando el mensaje hasta que recibe la confirmación de que el mensaje ha llegado a su destino. Por tanto, puede haber duplicidades.

    - **Exactamente una vez:** se asegura que el mensaje llegará a su destino, con lo que se envía una única vez.

- Alberga un mecanismo para el aviso de las desconexiones inesperadas.

- Existen diversos servidores o *brokers* MQTT para facilitar la implementación de las comunicaciones.

Por otra parte, la utilización de este protocolo destaca en las comunicaciones de dispositivos IoT (*Internet of Things*) debido a que permite enviar pequeñas cantidades datos que no requieren que se disponga de un gran ancho de banda en la red, así como por la sencillez que ofrece MQTT en la lectura de los sensores, el envío de los valores obtenidos y la configuración de los dispositivos pertenecientes a la red de forma remota.

Se ha utilizado MQTT y el *broker* MQTT *Mosquitto* en las comunicaciones entre los robots móviles *RB-1 Base* y con diversos módulos externos a ellos con el fin de obtener y enviar la información necesaria para realizar las tareas requeridas.

3.2.4 Plataforma *The Construct*

La plataforma *The Construct* [37] ofrece cursos de robótica y ROS con el propósito de aprender a crear y desarrollar proyectos de ROS, cuyos cursos abarcan desde los niveles más básicos hasta los más especializados. Asimismo, cuenta con un foro para la comunidad de desarrolladores de ROS con el objetivo de que los usuarios puedan informarse de las

actualidades del *framework* y preguntar o responder a consultas realizadas por otros desarrolladores.

Con fines académicos, *The Construct* alberga la página web *Robot Ignite Academy* [38] en la que se hallan los distintos cursos de ROS, algunos agrupados por nivel o por contenidos como, por ejemplo, los que se encargan de la enseñanza de la navegación, *machine learning*, etc.

En el instituto ai2 y en la asignatura de Automatización y Robótica (ARO) del Máster Universitario de Ingeniería Informática (MUIInf) de la UPV, se utiliza esta plataforma para el aprendizaje de ROS.

3.2.5 Visual Studio Code

Visual Studio Code es una herramienta para la edición de código fuente creada por Microsoft para diversos sistemas operativos, entre ellos Linux y sus distintas distribuciones. Dicho editor admite varios lenguajes de programación como Python, C, C++, JavaScrip y JSON entre otros.

Por otra parte, esta herramienta es personalizable, intuitiva y fácil de utilizar ya que cuenta con finalización inteligente de código, librerías que ayudan a visualizar la sangría del texto como *ident-rainbow*, la inclusión de control de versiones y la posibilidad de depuración de código.

En este proyecto, se ha utilizado la versión 1.46.1 de Visual Studio Code en el sistema operativo Ubuntu 18.04 para el desarrollo del código introducido en los robots que ha posibilitado la comunicación y ejecución de los programas necesarios para realizar un control descentralizado de los robots.

3.2.6 Rviz

Rviz [39] consiste en una herramienta para aplicaciones de ROS con la cual es posible observar el estado del robot en tiempo real por medio de la utilización de diversos componentes o *displays* con representación gráfica. Dichos módulos permiten ver el modelo robótico, el mapa, la posición del robot, las lecturas de reciben los sensores, etc.

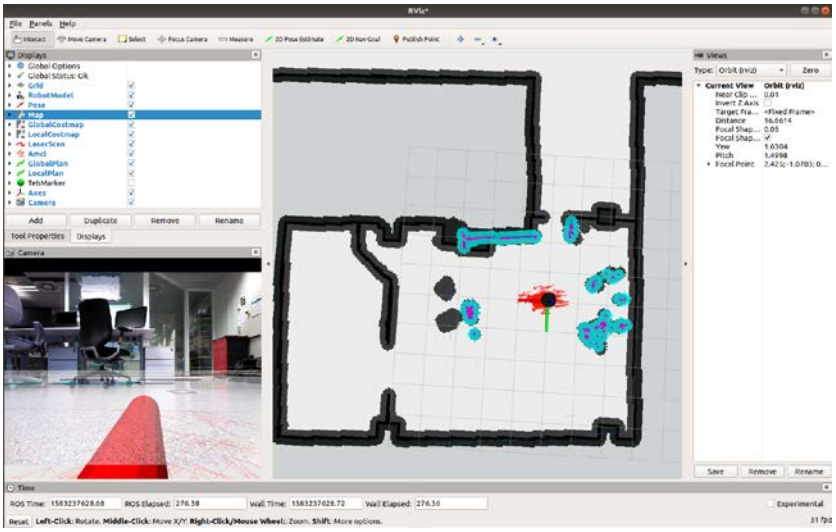

Ilustración 17: Rviz

A lo largo de este proyecto, los componentes que se han utilizado con mayor frecuencia en Rviz, son los siguientes:

- **Mapa:** encargado de visualizar la matriz de ocupación (*Occupancy Grid Map*) en Rviz. A las zonas ocupadas (es decir, las zonas en las que los escáneres del robot han detectado algún obstáculo) se les asigna un coste de valor 100 dentro de la matriz y el color negro en el mapa. Sin embargo, las áreas libres cuentan con un coste de valor 0 en la matriz y se les asigna el color blanco en el mapa. No obstante, las áreas desconocidas, es decir, las zonas por las cuales el robot no ha pasado y, por tanto, no ha obtenido lecturas de sus escáneres tienen asignado el valor -1 en el coste y el color gris. En este último tipo de áreas, por tanto, el robot desconoce si hay obstáculos.

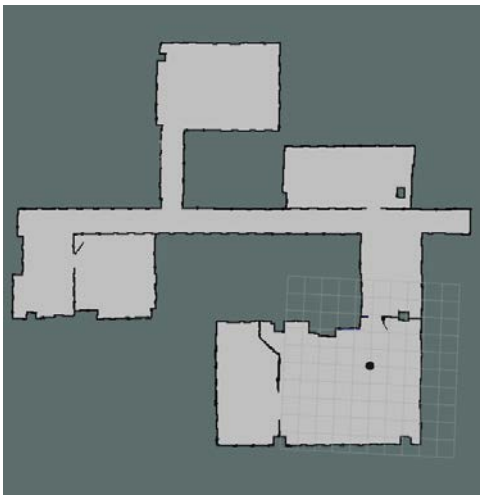

Ilustración 18: Mapa representado en Rviz

- **AMCL:** se trata de un algoritmo probabilístico cuyas siglas significan *Adaptive Monte Carlo Localization*. Este algoritmo calcula mediante unos filtros específicos la localización

de las posiciones más probables en las que podrían ubicarse los robots móviles dentro del mapa. El AMCL crea en el mapa que puede visualizarse en Rviz un área roja en dos dimensiones formada por flechas rojas. El robot podría hallarse en cualquier zona o punto de dicha área con la orientación marcada por las flechas, lo cual significa que si el área dibujada es pequeña el robot estudiado conoce con mayor precisión la posición en la que se encuentra dentro del mapa. Sin embargo, si el área mostrada es grande podría significar que el robot está perdido.

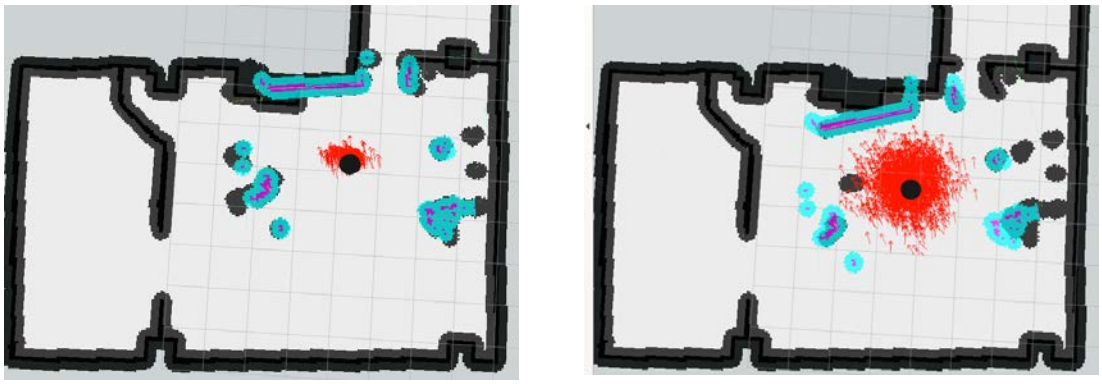

Ilustración 19: Localizado vs perdido en el AMCL

- **GlobalCostmap y LocalCostmap:** estos elementos muestran el mapa del entorno de trabajo del robot móvil visualizando también la inflación de los obstáculos reales. La inflación en realidad es un parámetro dentro de los cálculos que realiza la pila encargada en la navegación del robot, que aumenta el área ocupada por un obstáculo real en el mapa y crea así una distancia de seguridad alrededor dicho obstáculo. La diferencia entre el *GlobalCostmap* y el *LocalCostmap* consiste en que el componente *GlobalCostmap* muestra la inflación en todos los obstáculos del mapa (incluidas las paredes), mientras que el *LocalCostmap* únicamente visualiza la inflación en los obstáculos que el robot alcanza a detectar en tiempo real.

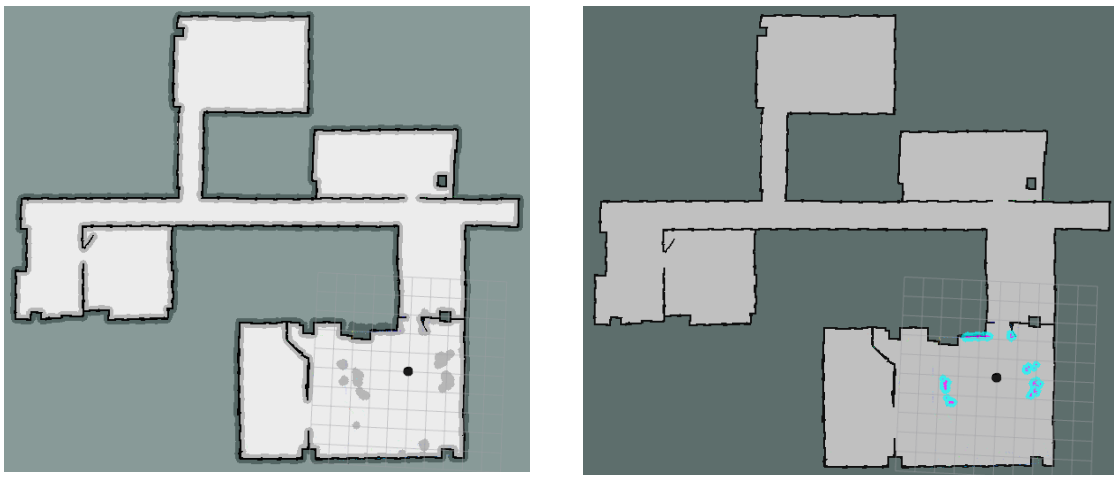

Ilustración 20: Mapas de coste Global y Local en Rviz

- **GlobalPlan y LocalPlan:** el componente *GlobalPlan* muestra en el mapa de Rviz el camino fijo calculado por el planificador de navegación, la ruta fija o global que seguirá el robot móvil con la finalidad de llegar a una determinada posición. Mientras que el *LocalPlan* visualiza la ruta que va calculando el robot en tiempo real intentando seguir la ruta global de la forma más eficientemente posible, sin colisionar con obstáculos que puedan aparecer mientras se dirige hacia su destino final. En la ilustración 21 el *GlobalPlan* y el *LocalPlan* se representan mediante líneas verde y azul respectivamente.

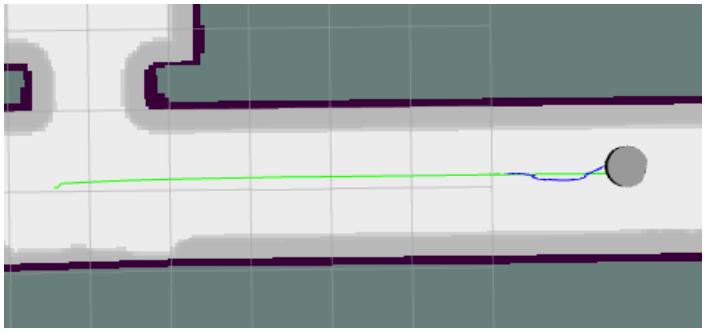

Ilustración 21: Trayectorias Global y Local visualizadas en Rviz

- **Camera y LaserScan:** se encargan de mostrar la información recibida por los respectivos sensores del robot (la cámara y el láser).

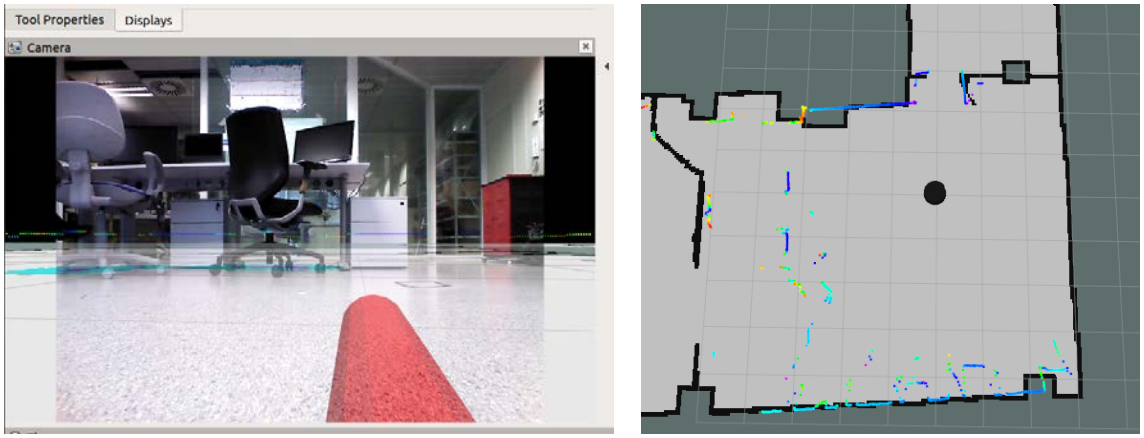

Ilustración 22: Cámara y Láser visualizados en Rviz

Por tanto, gracias a los diversos elementos visualizados por Rviz en tiempo real, se ha podido observar el comportamiento y el estado de los robots a lo largo del desarrollo del proyecto, lo que ha facilitado a su vez la realización y revisión de los experimentos y pruebas llevadas a cabo.

3.2.7 Gazebo

Gazebo [39] consiste en un simulador de tres dimensiones, dinámico, cinemático y multi robot, que admite la simulación de varios tipos de robots en entornos realistas, tridimensionales y

complejos, ya sean interiores o exteriores. Asimismo, permite recrear los entornos en los que se moverán los robots reales para testear sus comportamientos de una forma segura. Gazebo utiliza internamente el motor de física *Open Dynamics Engine* (ODE) y el motor de renderizado para escenas gráficas *OGRE*.

Por otra parte, Gazebo ofrece la ventaja de ser un software libre, compatible con ROS y Player, permitir cargar, simular y crear modelos de robots propios (URDF) y que los robots simulados puedan interactuar en el entorno recreado, ya que pueden desplazarse por el entorno, coger objetos, empujarlos, etc. Al mismo tiempo, este simulador se puede ejecutar desde ROS a fin de enviarles instrucciones a los robots simulados y recibir datos de estos en tiempo real. Por estas características, es una excelente herramienta para probar el comportamiento que tendrán los robots en el entorno real de forma controlada y segura.

Este simulador se ha utilizado en el proyecto antes de realizar los experimentos con los robots móviles en el entorno real, con la finalidad de depurar fallos y corregir errores, ya que se cuenta con el modelo en Gazebo del edificio de la *Fundació Ave María*.

CAPÍTULO 4
Diseño y desarrollo de la solución propuesta

En este capítulo se detallará, en primer lugar, la organización del tiempo invertido en el proyecto, así como las fases en las que está dividido. Seguidamente, se explicará el diseño y arquitectura empleados para la solución propuesta. Y, por último, se expondrá la estructura utilizada para el desarrollo del proyecto.

4.1 Plan de trabajo

El contrato de trabajo en el instituto ai2 dio comienzo a principios de noviembre del 2019. En la asignatura de Automatización y Robótica (ARO) del Máster Universitario de Ingeniería Informática (MUIInf) de la UPV se estudió el sector de la robótica y la plataforma software ROS. Posteriormente, al entrar en el ai2 disfruté de un mes de formación para profundizar en robótica y, especialmente, en ROS. Seguidamente, fui introducida en el proyecto ENDORSE para empezar a llevar a la práctica lo aprendido, para poder familiarizarme con las herramientas de trabajo y las tecnologías utilizadas en el proyecto (mencionadas en el capítulo anterior) así como para empezar a trabajar con la navegación de múltiples robots móviles con el objetivo de analizar e investigar cómo podían mejorarse, lo que constituiría el plan fundamental de este Trabajo Final de Máster (TFM).

Una vez definido el proyecto que se llevaría a cabo en el TFM, se elaboró un plan de trabajo inicial para establecer las etapas del proyecto y distribuir adecuadamente el tiempo dedicado a cada una de ellas. Dicho plan se diseñó a finales de febrero y estimó la necesidad de dedicar siete horas diarias a este proyecto en el horario laboral del instituto ai2 (es decir, de lunes a viernes).

En primer lugar, se valoró que tanto para la investigación como para la redacción del contexto tecnológico y de la situación actual de este trabajo serían necesarias al menos dos semanas. Seguidamente, para la etapa del diseño de la solución se plantearon también dos semanas, incluyendo en esta fase las comunicaciones con los diversos miembros del proyecto ENDORSE y entre ellos, especialmente, con los dos equipos implicados en el desarrollo de esta solución. Dichos equipos de trabajo son: la empresa valenciana *Robotnik*, para revisión y validación del sistema de comunicaciones escogido y, dentro de la UPV, el equipo del Centro de Investigación Gestión e Ingeniería de Producción (CIGIP) responsable del desarrollo de la aplicación central

para el control de la flota de robots o FMS, para definir el conjunto de órdenes e información requerida por ambas partes y las modificaciones necesarias en dicha aplicación.

Cuando el diseño estuviese completado y validado por todos los miembros del proyecto ENDORSE, se procedería a la etapa del desarrollo e implementación de la solución en los robots. Para dicha fase se estimó que serían necesarias seis semanas, ya que constituye la etapa con mayor dedicación de tiempo del proyecto, de las cuales cuatro se preveía que las dedicase durante la estancia en la empresa *SingularLogic* de Atenas (Grecia). Por otra parte, para la etapa de pruebas y resultados obtenidos se estimaron necesarias tres semanas adicionales, lo cual habría de coincidir en tiempo con las últimas semanas de la implementación con el objetivo de comprobar y corregir los errores para asegurar el correcto funcionamiento de la solución. Por lo tanto, se evaluó que el desarrollo completo de este Trabajo Final de Máster supondría unos 84 días.

TAREAS	INICIO	FIN	DURACIÓN
Formación	04/11/2019	29/11/2019	20 días
Estado del arte	02/03/2020	13/03/2020	10 días
Diseño	06/04/2020	17/04/2020	10 días
Implementación	20/04/2020	29/05/2020	29 días
Resultados	25/05/2020	19/06/2020	20 días

Ilustración 23: Plan de trabajo inicial

Sin embargo, como el plan inicial de trabajo se diseñó a finales de febrero, no pudieron tenerse en cuenta los efectos que el confinamiento provocado por la pandemia de COVID-19 iban a tener sobre el mismo.

La semana antes de que se decretara el primer estado de alarma para tratar de paliar los efectos que estaba provocando el COVID-19 se estuvo preparando el entorno de trabajo del instituto ai2 para dar cobertura al entonces ya inminente trabajo no presencial. Además, se requirieron dos semanas más para habilitar el entorno de simulación con varios robots móviles *RB-1 Base* con el propósito de desarrollar y probar los algoritmos sin necesidad de acceder a los robots reales.

Una vez el entorno de simulación y teletrabajo estuvo listo, se realizaron varias reuniones con los miembros del proyecto ENDORSE para organizar el trabajo. Estas reuniones permitieron que a principios de abril se pudiera volver a estimar un nuevo plan de trabajo corregido.

En el nuevo plan se amplió el plazo previsto para todas las etapas afectadas por este particular escenario, debido sobre todo a la necesidad de un aumento del número de reuniones necesarias para organizar los distintos paquetes de trabajo del proyecto ENDORSE y la adaptación a la nueva situación. Por todo ello, se determinó que la fase de diseño recibiría una semana más,

para la etapa de implementación se estimaron como necesarias dos semanas más de las previstas inicialmente y se destinó una semana más para la fase de experimentación y resultados. Además, en este plan corregido las dos últimas etapas se calcularon para llevarse a término en el entorno de simulación en vez de en el entorno real. Por otro lado, la estancia en *SigularLogic* se aplaza indefinidamente debido a la situación de alerta en Europa por esta pandemia.

TAREAS	INICIO	FIN	DURACIÓN
Formación	04/11/2019	29/11/2019	20 días
Estado del arte	02/03/2020	13/03/2020	10 días
Diseño	06/04/2020	24/04/2020	15 días
Implementación	20/04/2020	12/06/2020	39 días
Resultados	25/05/2020	19/06/2020	20 días

Ilustración 24: Plan de trabajo final

Finalmente, el total de días invertido en el proyecto ha sido de 104, veinte más de los previstos en la estimación inicial. Cabe destacar que únicamente se ha podido utilizar el entorno de simulación para el desarrollo y las pruebas, por lo que aunque se ha seguido el último plan de trabajo, se espera poder finalizar realmente la implementación y experimentación en los robots reales en julio o, como muy tarde, en septiembre, dependiendo de los permisos de la UPV, el instituto ai2 y, sobre todo, de la evolución de los rebrotes de esta pandemia.

4.2 Diseño de la solución

En este apartado se describirá la situación para la que se requiere la solución que plantea este Trabajo Fin de Máster (TFM), así como el diseño de las comunicaciones y la arquitectura utilizada en este trabajo.

4.2.1 Contexto de la solución

El equipo de investigación del CIGIP de la UPV colaborador del proyecto ENDORSE, ha implementado un servicio en la nube que consiste en una aplicación para el control de la flota de robots móviles o *Fleet Management System* (FMS), tal como se mencionó en el capítulo dos.

Esta aplicación cuenta con diversos algoritmos para el cálculo dinámico de trayectorias óptimas, la asignación de tareas y prioridades de estas, la supervisión y gestión del tráfico de los robots móviles, etc., así como con múltiples funciones de usuario para la gestión del entorno como, por ejemplo, aplicaciones para dividir el entorno de trabajo en zonas (habitaciones, salas de espera, quirófanos, farmacia, etc.) permitiéndoles asignar a cada zona un número máximo de robots circulando simultáneamente.

Este servicio utiliza el protocolo creado en el proyecto ENDORSE, *ENDORSE Protocol*, para intercambiar información con otros servicios, módulos y robots de forma segura mediante el protocolo de comunicación publicador-subscriptor, que permite enviar y recibir mensajes siguiendo una determinada estructura.

A través de este tipo de mensajes, el FMS puede enviarles órdenes o un conjunto de instrucciones a todos los robots móviles de la flota, al mismo tiempo que puede recibir retroalimentación o *feedback* del entorno y del estado de los robots, así como el informe de errores acontecidos, ya que tanto el FMS como los robots únicamente deben suscribirse o publicar en el *topic* correspondiente (nombre del canal de comunicación que recibe los mensajes) según la necesidad.

El funcionamiento normal del sistema se inicia introduciéndose en el FMS un mapa del entorno de trabajo. Para que el plano sea lo más fiable posible se utiliza uno de los que están empleando robots móviles de la flota para mapear el área de trabajo, ya que así se obtiene una representación más fiable con el sistema de referencia y escala utilizado por los robots móviles *RB-1 Base*. En la ilustración 25 se visualiza un ejemplo de un mapa de entorno creado por un robot móvil *RB-1 Base*; en este caso se trata del área de trabajo del instituto ai2.

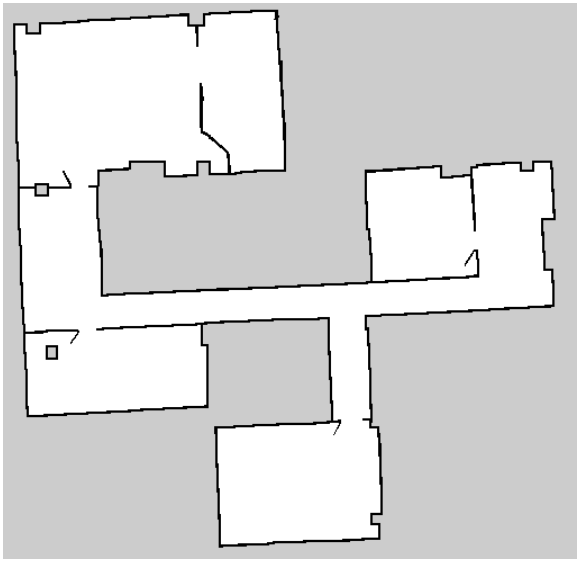

Ilustración 25: Mapa del entorno de trabajo del instituto ai2

Una vez el FMS dispone del mapa, y mediante interfaces de usuario, este se puede dividir a su vez en múltiples áreas de trabajo con distintos costes y se pueden restringir algunas zonas a un determinado número de robots móviles circulando simultáneamente para garantizar la seguridad y la optimización de compartición de los recursos del espacio, entre otras funcionalidades.

Por otro lado, esta aplicación central utiliza un grafo de Voronoi [40], que convierte los píxeles del plano en los diversos nodos o coordenadas concretas a los que los robots podrían desplazarse, para obtener las posibles trayectorias de los robots. La distancia entre los puntos también la determina el FMS: cuanto menor sea la distancia entre puntos mejor seguirá el robot la ruta planificada sin perderse (la distancia máxima entre puntos es de tres metros). Un ejemplo de posibles trayectorias obtenidas mediante un grafo de Voronoi se puede observar en la ilustración 26.

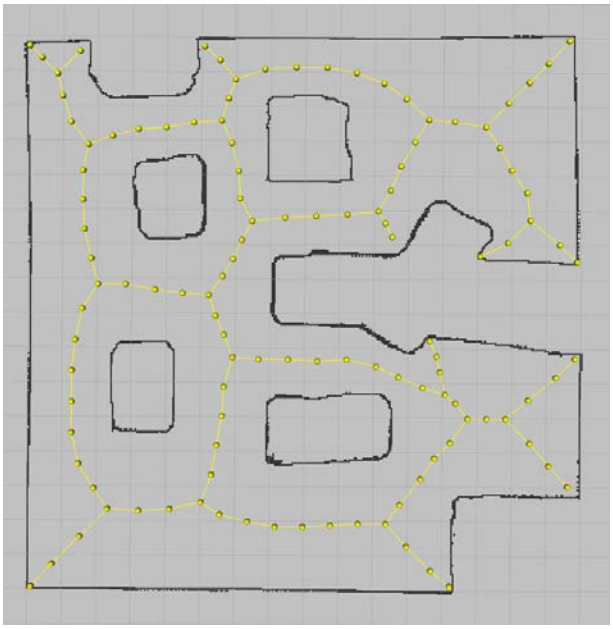

Ilustración 26: Ejemplo de grafo de Voronoi

Fuente: Web http://wiki.ros.org/tuw_voronoi_graph

El algoritmo encargado de la navegación del FMS escoge una determinada ruta para cada robot de entre todas las posibles según una serie de indicadores o criterios (distancia mínima recorrida, tiempo de trabajo, etc.) que se le suministra como datos de entrada a este y otros algoritmos. Las trayectorias obtenidas se les envía a los respectivos robots mediante el *ENDORSE Protocol* en forma de un conjunto de coordenadas. Tal como se ha mencionado anteriormente, junto con dichos puntos se envía otro tipo de información, como las órdenes que debe seguir el robot en cuestión y la información de la zona a la que pertenecen.

El funcionamiento habitual de los robots consiste en seguir las instrucciones, conjunto de tareas, órdenes y coordenadas recibidas del FMS para llegar a los puntos de destino. Al mismo tiempo, los robots mantienen informado al FMS (a una frecuencia determinada) para que pueda actualizar sus estados, ubicaciones, información recopilada del entorno, estado de las tareas, etc.

La solución propuesta en este Trabajo Final de Máster (TFM) se centra en las áreas particularmente expuestas a que se produzcan eventos anómalos que impidan al FMS actuar a

tiempo, como por ejemplo un espacio reducido, en el que es más fácil que se dé una situación de obstrucción del camino planificado para los robots y/o zonas limitadas a un número máximo de dos robots circulando simultáneamente.

Por tanto, este TFM se centra en la resolución de conflictos de navegación con múltiples robots móviles en áreas potencialmente conflictivas. En dichas zonas, los robots móviles son susceptibles a que se produzcan los siguientes problemas de navegación:

- **Colisiones:** si se originan alteraciones en el entorno del robot (por ejemplo, el desmayo de un paciente en una estancia pequeña) la ejecución de las trayectorias planificadas puede causar colisiones entre robots o con objetos del entorno.

- **Bloqueos:** los bloqueos o *deadlocks* entre dos o más robots se producen si alguno de ellos o varios se bloquean entre sí de tal manera que ninguno puede continuar con su trayectoria sin provocar una colisión. Los bloqueos pueden originarse si alguno de los robots es obligado a detenerse durante un tiempo.

Con el objetivo de solucionar estos problemas de navegación es necesario que exista una coordinación entre los propios robots, a parte de la que estos mantienen de por sí con la aplicación central. Por ende, se requiere implementar una comunicación descentralizada del FMS para la detección y evitación de colisiones y bloqueos.

Para el desarrollo de este proyecto se ha emulado un plano mapeado de un entorno sanitario en las instalaciones del instituto ai2.

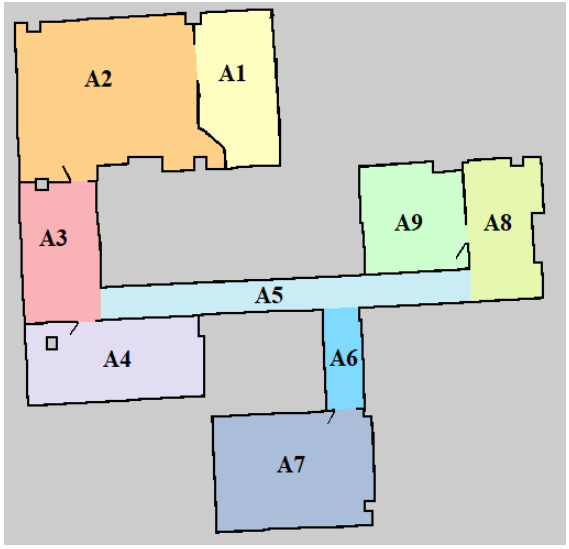

Ilustración 27: Mapa de entorno divido en áreas

Tal como se observa en la ilustración 27, el plano se ha divido en múltiples áreas con la finalidad de facilitar la visualización de las zonas potencialmente conflictivas que pueden crearse en un entorno de trabajo real. En este caso particular, serían ejemplo de zonas potencialmente conflictivas las áreas A5 y A6, ya que comprenden pasillos estrechos en los que únicamente pueden circular a la vez dos robots móviles para asegurar que cumplen con la distancia mínima de seguridad hacia los humanos y los objetos del entorno (tanto estáticos como dinámicos) a fin de cumplir la normativa de robots colaborativos.

4.2.2 Diseño de las comunicaciones del sistema

Con el objetivo de realizar un control descentralizado del FMS sobre los robots móviles en las zonas potencialmente conflictivas se necesita crear un sistema de comunicación entre los robots. Por esta razón, se ha planteado una arquitectura de cliente-servidor que permite la publicación y suscripción a los *topics* correspondientes según el tipo de mensaje con el propósito de mantener la concordancia con la arquitectura general del proyecto (ya que el sistema de los robots se basa en una arquitectura cliente-servidor, así como el FMS y otros módulos del proyecto ENDORSE).

En un principio se planteó utilizar el protocolo de transporte REST en el diseño de las comunicaciones por el hecho de a que, entre otras ventajas, REST permite la conexión de una gran cantidad de dispositivos. Sin embargo, finalmente se optó por emplear el protocolo MQTT ya que ofrecía más ventajas que REST como, por ejemplo, el hecho de constituir un protocolo más ligero y el de ofrecer una conexión estable entre el cliente y el servidor (es decir, permite que el cliente esté conectado siempre al servidor).

En la realidad, la razón principal por la que se prefirió utilizar MQTT fue el hecho de que la empresa valenciana *Robotnik* ha diseñado un módulo para el proyecto ENDORSE basado en la arquitectura cliente-servidor para la comunicación entre varios módulos empleando ese mismo protocolo. Se ha dado prioridad, por tanto, a la idea de mantener una concordancia tecnológica en el conjunto del proyecto ENDORSE.

Una vez tomada la decisión de emplear el protocolo MQTT en el diseño del sistema de comunicaciones ha habido que abordar el hecho de que, a su vez, este contempla dos tipos diferentes de intercambio de información: la comunicación entre los propios robots y la comunicación con el FMS.

Por lo que respecto a el intercambio de información entre robots móviles conectados a una misma red, se precisa de un cliente MQTT para el envío y recepción de los diversos tipos de mensajes necesarios en el funcionamiento del sistema. Conjuntamente, se va a utilizar el

servidor o *broker* MQTT *Mosquitto*, que contendrá los datos necesarios para el algoritmo. En la ilustración 28 se visualizan dichos componentes.

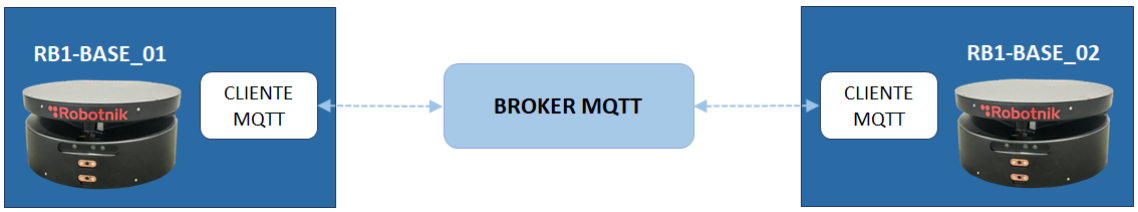

Ilustración 28: Diseño del sistema de comunicación entre robots móviles

Por otro lado, se ha utilizado el mismo diseño en la comunicación entre los robots y la aplicación central de control de flota, como puede comprobarse en la ilustración 29 en la que se visualiza el diseño del sistema de comunicaciones de la solución propuesta.

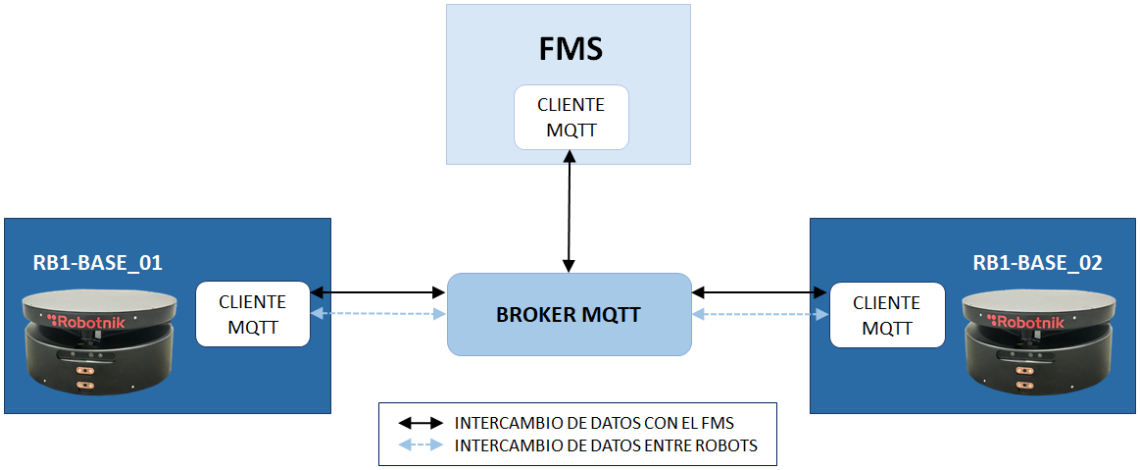

Ilustración 29: Diseño del sistema de comunicaciones

Una vez diseñado el sistema de comunicaciones se debe definir la estructura o arquitectura con la que contará el algoritmo para el control descentralizado.

4.2.3 Arquitectura de la solución propuesta

En la solución propuesta para el control descentralizado del FMS sobre los robots móviles que se encuentren en zonas potencialmente conflictivas se van a asumir, tanto para el diseño como para su posterior desarrollo, las siguientes de indicaciones generales:

- Las zonas potencialmente conflictivas para las que se plantea esta solución son aquellas que permiten circular como máximo dos robots móviles simultáneamente.

- Los robots móviles son alertados de que acaban de entrar en una zona potencialmente conflictiva por la aplicación central.

- El FMS envía a los robots que se encuentran en dichas áreas los datos del otro robot (tales como identificador del robot, las tareas que realiza, las prioridades de sus respectivas tareas, etc.).

- Todas las tareas asignadas a cada uno de los robots móviles cuentan con una determinada prioridad.

Para gestionar y resolver los conflictos que puedan producirse en la navegación de múltiples robots móviles colaborativos se necesita coordinación entre dichos robots. Con el objetivo de llevar a cabo esta coordinación, se ha planteado una arquitectura de tres capas comunicadas entre sí.

- **Capa de coordinación:** como cada robot dispone de los datos del otro robot móvil que se encuentra dentro de la zona conflictiva y todos los robots cuentan con un mismo criterio para seleccionar cuál de ellos goza de mayor prioridad, esta capa se encarga de seleccionar al robot prioritario o máster, que se encargará de coordinar a los demás robots (no prioritarios o esclavos) que se hallen en la zona de conflicto.

- **Capa de planificación:** en esta capa, cada uno de los robots planifica su ruta local a medida que detecta obstáculos o recibe instrucciones del otro robot siempre tratando de ceñirse lo máximo posible a la trayectoria global calculada en el momento de la asignación de la tarea que ejecuta cada robot. Esta capa será la encargada de detectar y evitar que se produzcan colisiones o bloqueos cuando los robots se ubiquen a una distancia determinada el uno del otro menor de lo estipulado.

- **Capa de ejecución:** es la responsable de ejecutar la trayectoria local planificada, así como de cumplir las instrucciones recibidas por el robot prioritario o máster. Además, puesto que los robots reciben información de su entorno constantemente a través de su sistema sensorial, en caso de no poder continuar con la ejecución de la ruta planificada se volvería a la capa anterior para recalcular o volver a planificar una nueva trayectoria local (o en caso del robot no prioritario o esclavo, esperar nuevas instrucciones).

En la ilustración 30 se visualiza la arquitectura aplicada para obtener un control descentralizado.

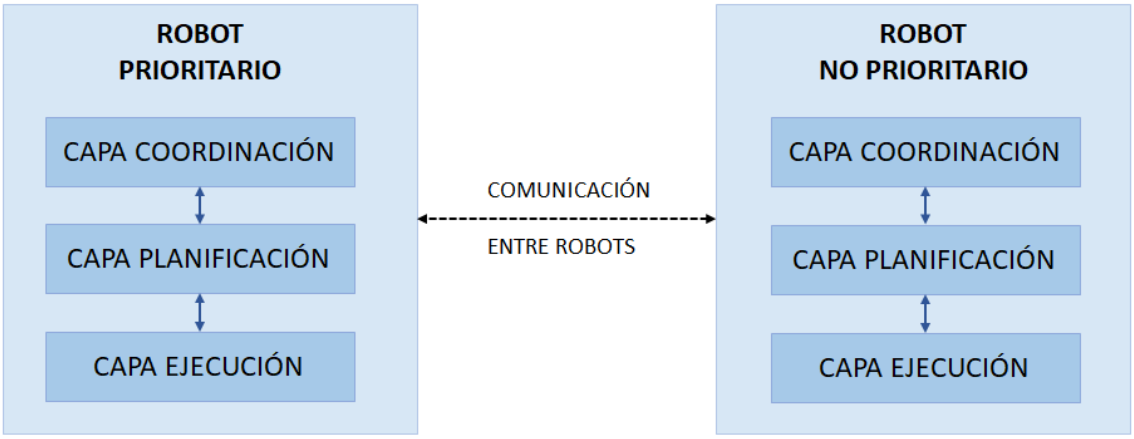

Ilustración 30: Arquitectura de la solución

Por otro lado, el funcionamiento general del algoritmo para el control descentralizado que contendrá cada uno de los robots móviles cuenta con una estructura dividida en las siguientes fases:

- **Fase inicial:** el algoritmo comienza cuando un robot móvil entra en un área codificada como potencialmente conflictiva al recibir del FMS la alerta correspondiente. Adicionalmente, si en dicha zona se localiza también otro robot móvil, el FMS le envía también los datos del primer robot y viceversa.

- **Fase intermedia:** se ejecuta el programa que indica al robot el papel que debe asumir mientras se encuentre en dicha área. Si resulta ser el robot prioritario ejecutará el algoritmo de máster; en caso contrario, ejecutará el algoritmo de esclavo y acatará las peticiones o instrucciones que se le envíen.

- **Fase final:** el algoritmo finaliza en cada robot una vez que abandona la zona potencialmente conflictiva. Cuando el robot se percata de que se encuentra fuera de dicha zona avisa al FMS para que retome el control sobre él.

Adicionalmente, en la ejecución de dichas fases también interviene el sistema de comunicaciones (explicado en el subapartado anterior) para se puedan llevar a cabo tanto la comunicación entre la aplicación central de control de flota y los robots móviles (en las fases inicial y final) como el intercambio de datos entre los robots móviles ubicados en la misma zona (durante la fase intermedia).

El diseño del funcionamiento de la solución propuesta para realizar un control descentralizado en múltiples robots se puede observar en la ilustración 31.

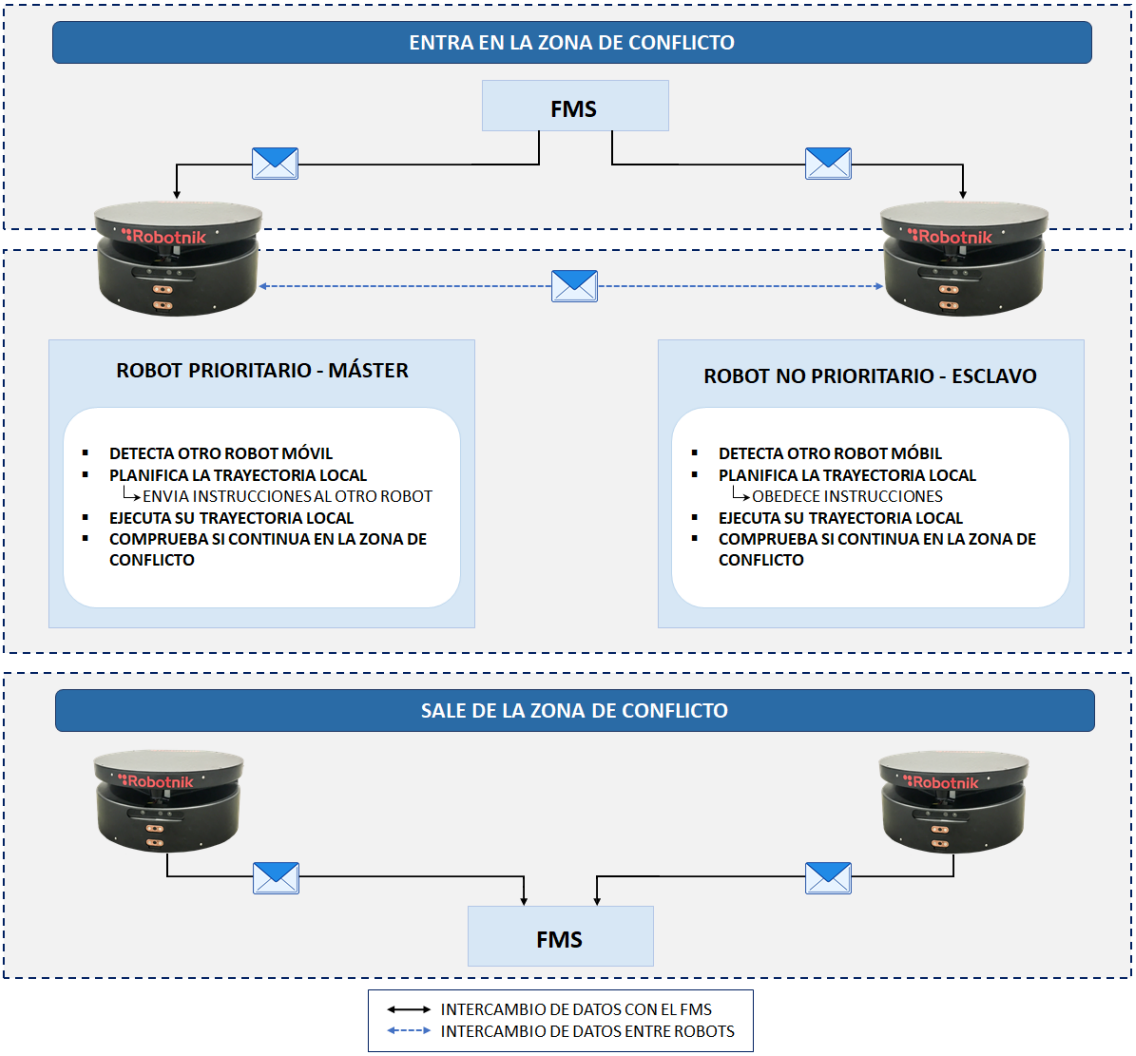

Ilustración 31: Diseño del funcionamiento general de la solución propuesta

Una vez definidos el funcionamiento general del algoritmo para el control descentralizado y el sistema de comunicaciones, pasamos a continuación a especificar los detalles y particularidades de su implementación y funcionamiento.

4.3 Desarrollo de la solución

En este apartado, se van a explicar tanto la implementación del sistema de comunicaciones como el desarrollo del algoritmo para el control descentralizado de múltiples robots móviles.

4.3.1 Implementación de las comunicaciones del sistema

En primer lugar, es conveniente tener en cuenta que la empresa *Robotnik*, tal como se ha explicado en el apartado anterior, ha creado un módulo en el proyecto ENDORSE para la comunicación entre las diversas áreas del proyecto que emplea la estructura cliente-servidor mediante el protocolo MQTT utilizando el *broker* de MQTT *Mosquitto*.

Por otro lado, los robots móviles *RB-1 Base* de la empresa *Robotnik* utilizan el sistema ROS que, entre otras funcionalidades, permite enviar o publicar mensajes en un determinado *topic* y recibir información de este u otros *topics* mediante una suscripción a ellos. Por tanto, para todas las comunicaciones internas entre dichos robots se emplea únicamente ROS.

Cabe señalar que los mensajes utilizados en el intercambio interno de los distintos nodos que conforman el sistema ROS de los robots móviles son a su vez un "tipo" de datos que consiste en una estructura de datos concreta predefinida (como los enteros, booleanos, cadenas de caracteres, etc.) o bien puede ser creada a propósito. Dicha estructura puede dividirse también en varias secciones (separadas mediante tres guiones) dependiendo del método de comunicación que se emplee:

- Los *topics* utilizan tipos de mensaje ".msg", que contienen tantos datos como requiera el mensaje agrupados en una única sección. Por ejemplo, si se quisiera enviar a través de un *topic* un mensaje de tipo "fecha" con estructura: *día-mes-año* junto a su tipo de datos respectivo, el mensaje "fecha.msg" contendría la siguiente información:

```
float32 día

float32 mes

float32 año
```

- Los servicios emplean la estructura de mensajes ".srv" que dispone de los datos necesarios divididos en dos secciones. La primera parte es la correspondiente al *request* que es la información que necesita solicitar el nodo cliente del servicio al nodo servidor. La segunda parte corresponde al *response* y cuenta con los datos con los que debe contestar el nodo servidor al nodo cliente. Por ejemplo, si se quisiera crear un servicio que permitiera dibujar un cuadrado del que se pudiera cambiar el tamaño, el contenido del mensaje "dibujarCuadrado.srv" podría ser:

```
float64 ladoCuadrado

---

bool finalizadoConExito
```

- Las acciones usan el tipo de mensajes ".action" cuya estructura se divide en tres partes. La primera es la misión u objetivo de la acción enviada por el nodo cliente de la acción al nodo servidor; la segunda es el resultado de la acción enviada por el nodo servidor de la acción y la tercera es el *feedback* enviado también por este último nodo al nodo cliente de la acción.

```
string objetivo

---

string resultado

---

string feedback
```

Las diversas estructuras que utilizan los múltiples mensajes que pueden enviar y recibir los robots del proyecto ENDORSE junto con los añadidos para hacer funcionar la solución propuesta en este TFM se localizan en el paquete *robot_simple_command_manager_msgs* creado por *Robotnik*. Dicho paquete se divide en tres carpetas denominadas *msg*, *srv* y *action* respectivamente y albergan la estructura que utilizan los mensajes del proyecto correspondiente.

Los mensajes más importantes en el sistema de comunicaciones de este TFM utilizados por el equipo del CIGIP para intercambiar datos o comandos entre robots o con el FMS son los siguientes:

- **Command:** engloba las diversas instrucciones que se le pueden enviar al robot y sus respectivos argumentos; los más destacables para el desarrollo de este trabajo son:

  - **GOTO:** Le indica al robot dónde debe ir, es decir, le especifica su misión u objetivo. La acción finaliza cuando el robot alcanza dicho objetivo. Adicionalmente, en el contenido del mensaje se especifica la posición y orientación en la que debe concluir; por tanto, los argumentos que necesita este comando son: la posición "x" e "y" del plano y el ángulo de orientación "theta".

    ```
    { "command" : "GOTO 0.0 0.0 0.0" }
    ```

  - **MOVE:** el robot se mueve una distancia determinada y la acción finaliza una vez el robot realiza el movimiento. El mensaje se compone de dos parámetros: el primero le especifica al robot cuánta distancia ha de avanzar (número positivo) o retroceder (número negativo); el segundo sirve para indicar el desplazamiento lateral; en el caso de los robots *RB-1 Base* siempre será cero ya que son robots diferenciales y no pueden desplazarse lateralmente.

    ```
    { "command" : "MOVE 1.0 0.0" }
    ```

- **TURN:** el robot rota un número de radianes determinado. En caso de que los radianes se especifiquen con números positivos el robot gira a la izquierda, en caso contrario gira a la derecha. La acción finaliza cuando el robot completa la rotación.

  ```
  { “command” : “TURN 1.0” }
  ```

- **STOP:** le indica al robot que recibe este mensaje que detenga su movimiento. Este mensaje no tiene argumentos, ya que no es necesario enviar información adicional.

  ```
  { “command” : “STOP” }
  ```

- **CONTINUE:** el robot receptor del mensaje procede a continuar con la ejecución de su trayectoria. La acción finaliza cuando retoma su ruta. Este mensaje también carece de argumentos.

  ```
  { “command” : “CONTINUE” }
  ```

- **Feedback:** se utiliza para mantener informado al nodo cliente del estado del último comando enviado. Por tanto, este tipo de mensaje contiene dos componentes: el primero consiste en la cadena *command*, que es la instrucción que se está ejecutando y el segundo es otra cadena, *message*, que dispone del código con el estado actual de la instrucción. Los códigos de los estados se encuentran en el mensaje *StatusCodes.msg* del paquete *robot_simple_command_manager_msgs* y son los siguientes:

  - 1, ACTIVE; 2, CANCELLED; 3, SUCCEEDED; 4, FAILED; 5, REJECTED.

- **Cancel:** este mensaje permite cancelar el comando que se está procesando o ejecutando en ese momento. Por esta razón, el mensaje no contiene argumentos, ya que para su ejecución no es necesaria más información.

- **Errors:** este mensaje permite que el FMS o los robots puedan realizar un informe de los errores ocurridos durante el procesamiento de una instrucción o la ejecución de una acción. Por tanto, el mensaje contiene la cadena *data* que recopila los errores aparecidos en caso de que se produzcan.

- **Map:** es una lista que contiene las coordenadas de navegación del mapa obtenidas mediante Voronoi junto con las áreas del mapa a las que corresponde cada coordenada.

- **Trajectory_map:** contiene parte de la ruta que debe seguir el robot con el fin de alcanzar su destino final. La ruta se envía en secciones para asegurar que el robot no se quede detenido en una determinada zona en caso de que no pueda situarse en algún punto intermedio

durante la ejecución de la trayectoria. Una vez ejecutada una determinada sección de la ruta, se le envía al robot la siguiente con un mensaje que contiene una lista denominada *trajectory_map* con cuatro objetivos o puntos que debe alcanzar (la distancia entre los puntos es menor que medio metro). Conjuntamente, cada uno de los puntos contiene un *source* o pose inicial en la que se encuentra el robot y un *target* o pose final que debe alcanzar, de forma que el primer conjunto de las variables [x, y, theta] del *source* corresponderá al primer punto del *target*. Un ejemplo del contenido de este mensaje es el que se muestra a continuación:

```
{ “trajectory_map” : {

  “source”: [0.0, 0.0, 0.1], [0.0, 1.0, 0.0], [1.0, 1.0, 0.0], [1.0, 0.0, 0.0]

  “targer”: [0.0, 0.1, 0.0], [1.0, 1.0, 0.0], [1.0, 0.0, 0.0], [0.0, 0.0, 0.0]

  } }
```

- **Alert_zone:** dispone de cuatro argumentos. El primero consiste en una cadena denominada *robotId* que contiene el identificador del robot móvil requerido; el segundo es la cadena *area* con el código de la zona del entorno de trabajo en el que se encuentra el robot identificado en el primer argumento; el tercer argumento *kind* es otra cadena que indica el tipo de zona en la que se encuentra dicho robot; y, por último, el argumento *localization* consiste en una cadena para indicar si se encuentra o no en la zona (mediante *in* o *out*).

    - El mensaje *ZoneMapCodes.msg* contiene una lista con las codificaciones del tipo de áreas; para el propósito que aquí nos ocupa destacan la codificación como zona libre (1, FREE) y la de zona potencialmente conflictiva (2, CONFLICTIVE), creada especialmente para este trabajo.

```
{ “alert_zone” : {

  “robotId”: “rb1_base_01”

  “area”: “A5”

  “kind”: “2”

  “localization”: “in”

  } }
```

Adicionalmente, como en el sistema de comunicaciones no solo intervienen las que se producen en ROS dentro de cada robot sino que también se requiere de comunicaciones externas (ya sea

de robots con el FMS o entre ellos) se necesita poder transformar los mensajes enviados en ROS desde los robots a MQTT en el *broker* de *Mosquitto*, y viceversa. Con esa finalidad, *Robotnik* ha desarrollado un puente o *bridge* en el servidor para transformar mensajes de ROS a MQTT y de MQTT a ROS.

El *topic* o la dirección de los mensajes que se envían desde MQTT disponen de la siguiente estructura predefinida por el protocolo creado en el proyecto ENDORSE, *ENDORSE Protocol*:

```
/{namespace}/{robotId}/{criterion}
```

En dicha estructura, *namespace* indica la ubicación del robot que, en este caso, se encuentra en el proyecto ENDORSE (por eso se utiliza *endorse*). El *robotId* es el identificador del robot móvil receptor del mensaje y *criterion* es el tipo de instrucción enviada (*command*, *feedback*, *cancel*, *errors*, etc.).

Un ejemplo de la estructura que siguen las instrucciones enviadas a través de MQTT al robot de identificador *rb1_base_01* sería el siguiente:

```
/endorse/rb1_base_01/command

/endorse/rb1_base_01/feedback

/endorse/rb1_base_01/cancel

/endorse/rb1_base_01/errors

/endorse/rb1_base_01/trajectory_map

/endorse/rb1_base_01/alert_zone
```

Sin embargo, cuando los mensajes se envían desde ROS se sigue la siguiente estructura:

```
/{robotId}/{command_manager}/{criterion}
```

Aquí, *command_manager* es el programa encargado de gestionar las órdenes o instrucciones del proyecto en el puente o *bridge* creado por Robotnik en el *broker* MQTT.

Como consecuencia de dicha estructura, si se envían órdenes al mismo robot móvil *rb1_base_01* a través de ROS, la dirección del mensaje será:

```
/rb1_base_01/command_manager/command

/rb1_base_01/command_manager/feedback

/rb1_base_01/command_manager/cancel
```

```
/rb1_base_01/command_manager/errors

/rb1_base_01/command_manager/trajectory_map

/rb1_base_01/command_manager/alert_zone
```

Una vez definida la estructura de la dirección en la que se deben publicar o suscribir los mensajes en MQTT o ROS (dependiendo de la tecnología que se emplee en su envío o recepción) los robots ya están en disposición de publicar mensajes en dichas direcciones.

Para publicar en un *topic* o suscribirse a uno o varios *topics* mediante MQTT es necesario crear previamente un cliente MQTT en el dispositivo que se quiere conectar al servidor correspondiente. Por esta razón, en el diseño de comunicaciones del sistema detallado en el apartado anterior tanto el FMS como los robots móviles *RB-1 Base* disponían de un cliente MQTT. Además, una determinada entidad (FMS o robot) puede publicar en un *topic* mediante MQTT a través de consola; por ejemplo, si se quiere publicar el mensaje *command GOTO* para que el robot *rb1_base_01* se desplace a la posición del mapa [-2.0, 0.0, 1.0], se debe publicar en el topic */endorse/rb1_base_01/command* de la siguiente forma:

```
mosquitto_pub -t /endorse/rb1_base_01/command -m "{\"command\": \"GOTO -2.0
0.0 1.0\"}"
```

Para publicar un mensaje mediante ROS o suscribirse a uno o varios *topics* se necesita conocer el nombre del *topic* o la dirección del mensaje, el tipo o la estructura que utiliza dicho mensaje y los datos que se quieren enviar o leer.

En ROS se utilizan los siguientes comandos para inicializar un nodo, crear un publicador y un suscriptor:

```
rospy.init_node('nombre_del_nodo')

rospy.Publisher('/nombre_topic', tipo_mensaje)

rospy.Subscriber('/nombre_topic', tipo_mensaje, funcionCallback_obtenerDatos)
```

Además, ROS también permite publicar mediante consola siguiendo la estructura:

```
rostopic pub </nombre_topic> <paquete/tipo_mensaje> <datos_del_mensaje>
```

Por ejemplo, si se quiere dirigir el mensaje *command GOTO* al robot *rb1_base_01* para que se desplace a la misma posición del mapa que en el ejemplo visto antes con MQTT se debe publicar en la dirección o *topic* para los mensajes enviados a través de ROS siguiente: */rb1_base_01 /command_manager/command*. El tipo de mensaje de *command GOTO* es

CommandString.msg, que se almacena en el paquete *robot_simple_command_manager_msgs*. Por tanto, para publicar este mensaje desde consola se debe introducir los siguientes datos:

```
rostopic pub /rb1_base_01/command_manager/command robot_simple_command_manager
_msgs/CommandString "command: 'GOTO 2.0 0.0 1.0'"
```

En resumen, la implementación explicada para el envío y recepción de mensajes, su estructura y direcciones de publicación y lectura en función de la tecnología empleada es la que posibilita la implementación de la solución que se propone en este TFM y que se especifica en el siguiente apartado.

4.3.2 Implementación de la solución propuesta

Con el objetivo de controlar a los robots del proyecto de forma descentralizada de la aplicación central o sistema de gestión de flotas, se han creado un conjunto de ejecutables para elaborar el algoritmo que conforma la solución propuesta.

El programa principal o *main* del algoritmo se inicia en cada robot móvil cuando este recibe la alerta enviada por la aplicación central a través del mensaje MQTT *alert_zone* de que acaba de acceder a un área del entorno de trabajo codificada como “zona potencialmente peligrosa”. Como en estas zonas se permite la circulación de dos robots móviles de forma simultánea, si se da el caso de que previamente otro robot móvil se encontraba dentro de dicha área, el FMS le enviará al robot que acaba de acceder y en el que se ha iniciado el algoritmo los datos del otro robot móvil (identificador del robot *robotId*, tarea que está ejecutando y prioridad de dicha misión).

El siguiente paso del *main* consiste en escoger cuál de los robots móviles ubicados en la misma zona es prioritario. Para ello, todos los robots siguen la misma pauta: el robot móvil responsable de ejecutar la tarea de mayor prioridad será el líder o máster. Este criterio es aplicable porque, tal como se explica en las asunciones generales detalladas en el apartado anterior, cada una de las misiones o tareas asignadas a un robot dispone conjuntamente de una prioridad determinada.

En el caso de que se encuentre únicamente un robot móvil en la zona potencialmente conflictiva, el programa principal escogerá a dicho robot como prioritario y, seguidamente, ejecutará el algoritmo máster. Sin embargo, para dar lugar a la explicación del flujo del programa aquí se va a suponer que hay dos robots en la zona potencialmente conflictiva indicada por el FMS. Como consecuencia y ya que ambos robots cuentan con el mismo criterio de selección, uno de los robots ejecutará el algoritmo máster y el otro el esclavo.

Los archivos responsables de ejecutar los algoritmos máster y esclavo son muy parecidos, ya que inicialmente ambos algoritmos se comportan exactamente igual. Tanto si el robot móvil en cuestión ejecuta el programa máster como el esclavo, inicialmente continúa ejecutando la ruta calculada por el FMS mientras comprueba, a una determinada frecuencia, si ha entrado o salido algún otro robot móvil de la zona potencialmente conflictiva o si continúa desplazándose por ella.

En ambos programas, en caso de que se detecte que otro robot ha entrado o salido de la zona potencialmente conflictiva se vuelve al programa principal, el *main*, para que vuelva a iniciar el proceso de evaluación del robot prioritario. Si, por otro lado, el algoritmo que está ejecutando el robot (ya sea el máster o el esclavo) averigua que dicho robot móvil ha abandonado la zona indicada por el FMS, vuelve al programa principal encargado de notificar la actualización del estado de dicho robot tanto al otro robot móvil como a la aplicación central mediante la publicación del mensaje MQTT *alert_zone*, aportando su identificador como primer argumento.

Por otro lado, el algoritmo máster se encarga de analizar la distancia entre ambos robots. Cuando detecta que dicha distancia es inferior a un determinado valor y, que por tanto, existe el riesgo de que se produzca un problema de colisión en la navegación, le envía la instrucción de *STOP* al otro robot para que cese su movimiento. Seguidamente, observa el entorno de trabajo de ambos robots a través de los datos recibidos por los sensores exteroceptivos para enviarle el resto de órdenes al otro robot (tales como, *avanza*, *retrocede* o *gira* a la *derecha* o *izquierda* un determinado número de radianes). Posteriormente, recalcula su propia trayectoria local para alcanzar su siguiente objetivo (ya sea una posición intermedia o la pose final) y la ejecuta. En el momento en que detecta que ha sorteado al otro robot le envía la instrucción *CONTINUE* mientras él mismo sigue ejecutando su trayectoria y realiza las evaluaciones requeridas del entorno.

A diferencia del algoritmo máster, el esclavo comprueba si ha recibido órdenes por parte del robot prioritario durante la ejecución de su trayectoria mediante la suscripción a la dirección de mensaje *command* correspondiente. Cuando recibe una instrucción la ejecuta y queda a la espera de nuevas órdenes hasta que obtiene el comando *CONTINUE*. Al recibir *CONTINUE*, el algoritmo esclavo recalcula o replanifica su trayectoria local para cumplir su misión y ejecuta dicha ruta para llegar a su siguiente posición objetivo.

El flujo descrito del algoritmo para el control descentralizado se observa en la ilustración 32.

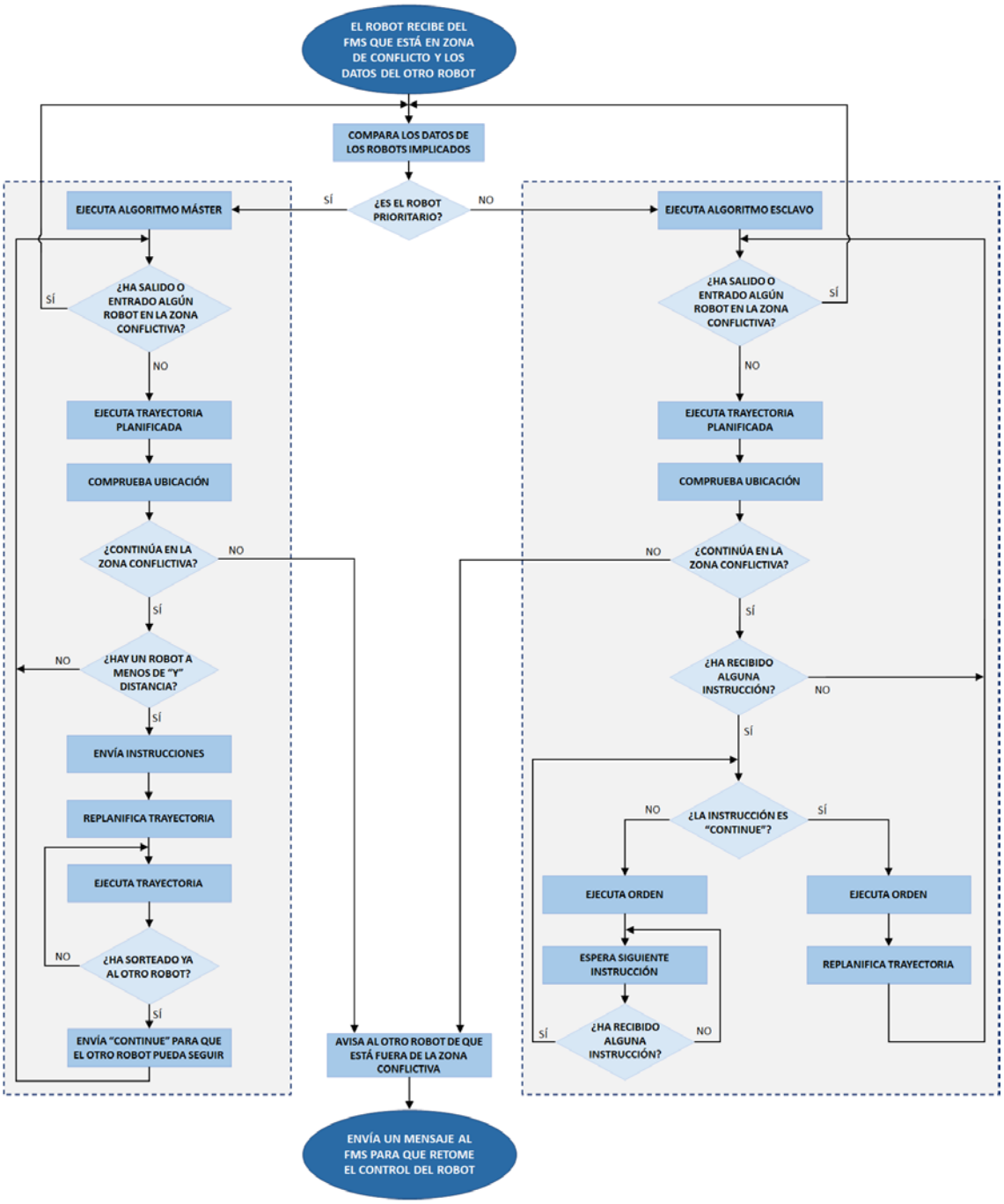

Ilustración 32: Diagrama de flujo de la solución propuesta

Los programas y algoritmos utilizados para desarrollar el flujo del algoritmo para el control descentralizado descrito anteriormente se agrupan en el paquete creado específicamente para dicha solución denominado *decentralized_control*.

Todos los *RB-1 Base* disponen de una CPU en la que está instalado el sistema operativo de los robots (Ubuntu 16.04) y la plataforma software ROS para su manipulación. Por tanto, cada robot móvil cuenta con un directorio de trabajo o *workspace*, denominado *catkin_ws*, en el que se ubican los archivos y paquetes de ROS para su control.

Adicionalmente, cada robot móvil en el directorio *src* dentro de *catkin_ws* contiene el paquete *move_base* [41] que implementa una acción (recogida en la librería *actionlib*) para desplazar el robot por el entorno de trabajo enviándole un objetivo o *goal* con la posición de destino. Conjuntamente, *move_base* enlaza con el paquete *nav_core*, que se encarga de planificar tanto la trayectoria global como la local para alcanzar la posición enviada. Con ese fin, el paquete *nav_core* alberga un planificador global junto al mapa de costes o *costmap* global, un planificador local (en el caso de este proyecto, *teb_local_planner*) con su mapa de costes y, por último, los comportamientos de recuperación frente a eventos que impidan seguir el trayecto planificado.

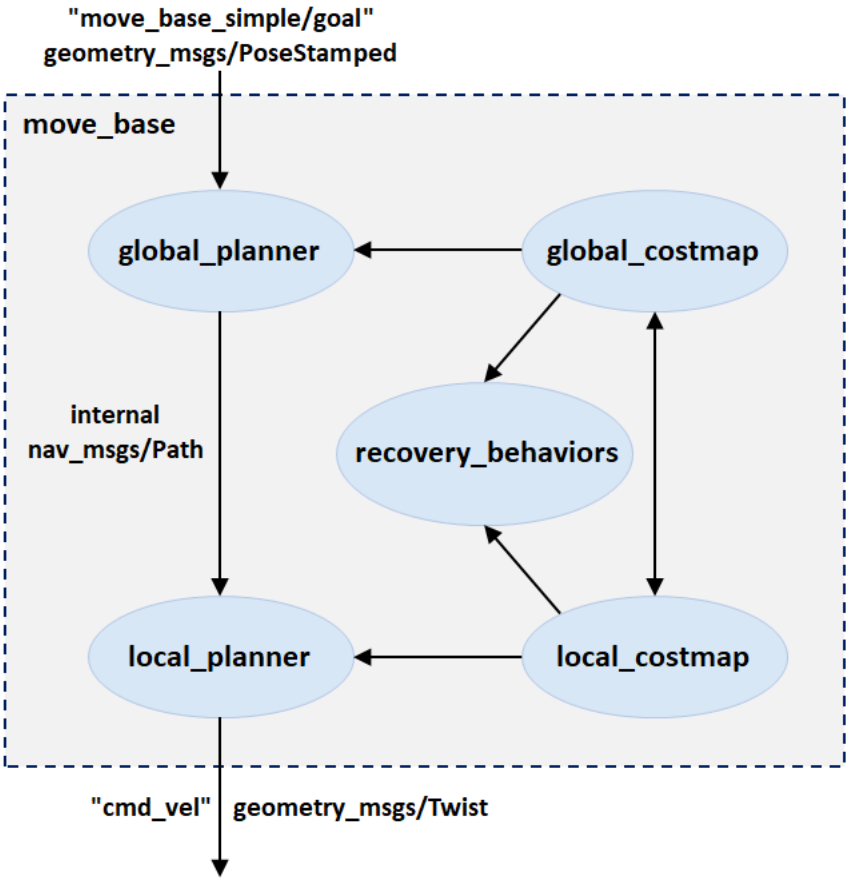

Ilustración 33: Paquete de ROS move_base

El paquete *decentralized_control* creado para la solución propuesta se añadirá al directorio *src* ubicado dentro de *catkin_ws* de cada robot móvil del proyecto ENDORSE. Asimismo, su contenido se visualiza en la ilustración 34 y será responsable de enlazar con los paquetes *move_base* y *nav_core*.

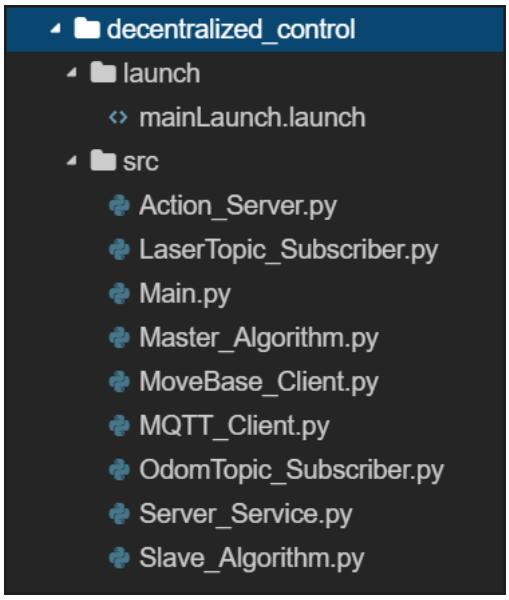

Ilustración 34: Estructura de la solución propuesta

Tal como puede observarse en la ilustración anterior, el paquete *decentralized_control* está compuesto por dos directorios: *launch* y *src*. El directorio *launch* alberga el archivo *mainLaunch.launch* encargado de iniciar los nodos específicos del paquete en un orden determinado, mientras que *src* agrupa los programas y nodos responsables de la elaboración y ejecución de la solución propuesta tal y como se explica a continuación:

- **Main:** inicializa el nodo *decentralized_main_node* y utiliza *MQTT_Client* para recibir del FMS los mensajes *alert_zone*. Cuando el argumento *kind* del mensaje indica que la zona es de tipo conflictiva inicializa la ejecución del proceso de selección de robot prioritario comparando los datos correspondientes. Si obtiene el resultado de que el robot que ejecuta este fichero es el prioritario llama al fichero *Master_Algorithm*, en caso contrario, ejecuta el programa *Slave_Algorithm*. Asimismo, cuando finaliza la ejecución del *Master_Algorithm* o el *Slave_Algorithm* porque el robot ha abandonado dicha zona, se encarga de notificar tanto al otro robot como al FMS esta situación mediante las funciones de *MQTT_Client*.

- **Master_Algorithm:** utiliza las funciones de *LaserTopic_Subscriber* para determinar si hay otro robot *RB-1 Base* demasiado cerca o si ya lo ha sorteado. En caso de que otro robot se encuentre a menor distancia del valor predefinido, este algoritmo es responsable de enviar las órdenes adecuadas al no prioritario; con ese objetivo, consulta la función proporcionada por el *Server_Service* para obtener la dirección a la que se debe mover el robot esclavo. Por otro lado, llama al *MoveBase_Client* cuando debe desplazarse a su siguiente posición objetivo, para que replanifique y ejecute la ruta local. Conjuntamente, llama a *Action_Server* con el fin de verificar si sigue en la zona de conflicto. En este programa se inicializa el nodo *decentralized_master_node*.

- **Slave_Algorithm:** llama a *Action_Server* y comprueba si sigue en la zona de conflicto. Asimismo, ejecuta *MoveBase_Client* cuando debe ir a una nueva posición para cumplir las instrucciones enviadas por el robot máster o para seguir con su trayectoria y alcanzar su posición final. Además, al iniciarse este fichero se inicializa el nodo *decentralized_slave_node*.

- **Action_Server:** inicializa el nodo *decentralized_action_server*. Asimismo, este fichero proporciona la acción que comprueba la ubicación del robot y averigua si sigue dentro de la zona potencialmente conflictiva (indicada por el FMS) mediante la consulta a la odometría del robot proporcionada por *OdomTopic_Subscriber*. En caso de detectar que se encuentra fuera de la zona de conflicto el mensaje de *result* de la acción será *true*. En caso contrario, no se publica el resultado de la acción.

- **Server_Service:** inicializa el nodo *decentralized_server_service*. Este servicio cuenta con la función *direction_to_move* para decidir el conjunto de movimientos que debe realizar el robot esclavo (*avanzar*, *retroceder*, *girar a la izquierda* o la *derecha*), dependiendo de los datos detectados por el sensor láser y la odometría, con lo que consulta tanto a *LaserTopic_Subscriber* como a *OdomTopic_Subscriber*.

- **MoveBase_Client:** inicializa el nodo *decentralized_movebase_client*. Este programa es un nodo cliente de la acción *MoveBaseAction* que proporciona la librería *actionlib* del paquete *move_base* contenido en cada robot móvil. Dicha acción permite enviarle al robot la nueva posición a la que debe desplazarse mediante la publicación del mensaje de acción *MoveBaseGoal*.

- **LaserTopic_Subscriber:** este ejecutable se suscribe al *topic* en el que publica el sensor láser del robot e implementa las funciones: *get_laser_data*, para obtener los datos visualizados por el robot y *sector_detection* que devuelve la posición de los obstáculos (delante, derecha o izquierda) a fin de facilitar el acceso a dichos datos desde otros programas. En este archivo se inicia el nodo *decentralized_subscriber_LaserScan*.

- **OdomTopic_Subscriber:** inicia el nodo *decentralized_subscriber_Odom*. Este programa se suscribe al *topic* en el que recoge los mensajes publicados por la odometría. Además, cuenta con la función *get_odom_data* para facilitar a los demás programas acceder a los datos publicados por la odometría.

- **MQTT_Client:** se inicializa el nodo *decentralized_client_mqtt*. Conjuntamente, este archivo alberga todas las publicaciones y suscriciones con la aplicación central del control

de flota. Cada publicación y suscripción adopta la forma de función con el fin de que se puedan llamar desde otros ejecutables.

CAPÍTULO 5
Ensayos y resultados

En este capítulo se presentarán las pruebas realizadas sobre el proyecto ENDORSE antes de la realización del TFM y, a continuación, se someterá a pruebas al sistema de comunicaciones y se realizarán varios ensayos del entorno de simulación para observar el comportamiento de los robots y comprobar que se ajustan a lo esperable.

5.1 Testeos y resultados previos al desarrollo del TFM

A finales de febrero, la *Project Officer* Aikaterina Lazaridou llevó a cabo una revisión y auditoría del proyecto ENDORSE [42] denominada *Mid Term Review* en las instalaciones del instituto ai2 con el fin de constatar los avances realizados.

Mi compañera de equipo Marta Payá Tormo y yo habíamos iniciado el trabajo de mejora del sistema de navegación por el entorno de los robots móviles durante los meses previos a esta revisión. Para ello, habíamos ajustado dinámicamente la distancia de seguridad que debían respetar los robots con respecto a las paredes, los obstáculos estáticos y los dinámicos, mediante la manipulación del parámetro de inflación del que disponen los mapas creados por ellos. Al permitir que los robots pudieran acercarse más a los objetos (excepto cuando se trata de personas) se dio paso a nuevos escenarios en los que se disponía de más espacio para resolver posibles conflictos, lo que a su vez redundaba en una mejor gestión del espacio de trabajo por parte de los robots. Durante la revisión del proyecto se mostró la aplicación del control de la flota robótica y de nuestro trabajo, cuyo objetivo consistía en lograr que varios robots se desplazaran por el entorno del instituto ai2 (cuya distribución simulaba instalaciones sanitarias) de forma que consiguieran llegar a sus puestos de trabajo finales sin que se produjeran colisiones o bloqueos durante su navegación

Ilustración 35: Demostración de los robots móviles circulando por el Instituto ai2

Se observó que las mejoras implementadas optimizaban la capacidad de los robots móviles para esquivarse de forma efectiva cada vez que sus trayectorias se cruzaban. Sin embargo, en un momento dado en que dos robots se cruzaron por un pasillo estrecho en el cual se encontraban presentes algunas personas, aparecieron algunas dificultades evidentes para evitarse, seguramente porque el FMS tardaba en responder más de lo habitual y ellos no eran capaces por sí mismos de coordinar sus movimientos para resolver el conflicto. Durante la auditoría consiguieron sortearse pero en algunos ensayos posteriores los robots entraban en parada tras intentar esquivarse repetidas veces sin éxito. Estos escenarios fueron los que inspiraron la necesidad de desarrollar un sistema de comunicación y coordinación entre los distintos robots móviles (idea que, a su vez, originó este Trabajo Final de Máster). Además, se aspiraba a poner a prueba a los robots en un entorno real con el fin de analizar los resultados y compararlos con los obtenidos durante la revisión y verificar, de ese modo, el nivel de mejora alcanzado al aplicar la solución propuesta en este TFM. Por desgracia, y como consecuencia de la pandemia sufrida durante los últimos meses, únicamente se ha podido probar en un entorno simulado.

Por otro lado, durante la auditoría el equipo de la empresa *StreamVision* encargada del desarrollo del paquete WP4 mostró el funcionamiento de la aplicación sanitaria *e-Diagnostic*. Esta aplicación sirve para analizar los datos obtenidos de los sensores y aparatos médicos que realizan la medición básica de constantes vitales en los pacientes. El equipo necesario para obtener dichas medidas se encuentra instalado en el soporte creado por la empresa *Robotnik* (como parte del paquete de trabajo WP5) que se visualiza en la ilustración 36.

Ilustración 36: Demostración de la aplicación sanitaria e-Diagnostic en el Instituto ai2

Una vez el equipo de *StreamVision* finalizó su demostración, *Robotnik* mostró el funcionamiento del sistema de elevación diseñado para el traslado de objetos que se incorporó en la plataforma de un *RB-1 Base*. La demostración realizada trataba de ilustrar el modo como la plataforma elevable incorporada al robot móvil podía alzar una mesa de trabajo (diseñada

para el entorno sanitario de la *Fundació Ave María*) y desplazarla a otra habitación con movimientos suaves y sin colisionar con paredes, puertas u obstáculos.

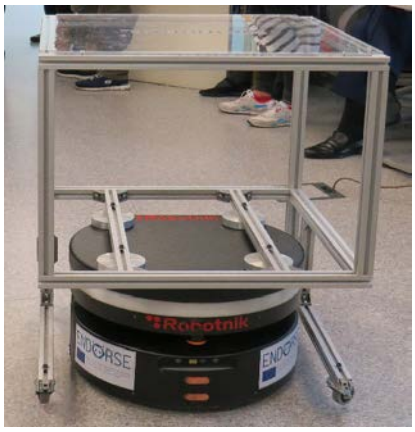

Ilustración 37: Demostración del sistema de elevación en el Instituto ai2

Adicionalmente, se presentaron dos nuevos proyectos desarrollados por el instituto ai2 en la demostración del *Mid Term Review* que fueron ideados para mejorar la interacción humano-robot (HRI) de la tarea *Human-aware path planning algorithms*. El primero consiste en un sensor de guiado instalado en un *RB-1 Base* para desplazar o aparcar los robots móviles. Este sensor ha sido diseñado para escenarios en los que se deba realizar mantenimiento o actualizar los robots, pero es útil también en caso de que algún robot se pierda o deje de responder correctamente, ya que facilita el trasladarlo a su área de mantenimiento sin necesidad de conocimientos técnicos. El segundo proyecto es un brazo robótico UR3 de la empresa *Universal Robots* para la manipulación de objetos. Este robot se instaló sobre una plataforma diseñada específicamente para la base de un robot móvil *RB-1 Base*. En la ilustración 38 se observan tanto el brazo robótico UR3 como el sensor de guiado situado sobre el soporte del UR3. Tal y como se puede visualizar el sensor de guiado va protegido mediante un cobertor de plástico blanco impreso en una impresora 3D del instituto ai2.

Ilustración 38: Sensor de guiado y brazo robótico UR3 del Instituto ai2

5.2 Testeo de la solución propuesta

El proyecto ENDORSE dispone de un repositorio en GitLab [43] en el que cada equipo sube y actualiza el trabajo que ha desarrollado en el proyecto. Además, los colaboradores de la empresa *Citard Services Ltd* junto con los de *Robotnik* han creado un entorno de simulación de la *Fundació Ave María* que dispone de varios robots móviles con el objetivo de contar con un entorno virtual en el que poder comprobar y verificar el funcionamiento de los distintos elementos que componen el proyecto ENDORSE.

Para iniciar el entorno de simulación se requiere estar dada de alta en el repositorio de Docker Hub [44] y descargar la imagen *docker* del proyecto ubicada en el repositorio de GitLab (al que únicamente los miembros del proyecto ENDORSE pueden acceder). Una vez descargada la imagen en nuestro ordenador, se accede a la ubicación *~/Docker/rb_log_sim/.docker* y se abren varias terminales.

En la primera consola se inicia sesión en el registro de Docker Hub con el nombre de usuario *docker* y a continuación se lanza el primer ejecutable que contiene el entorno de simulación mediante las siguientes instrucciones:

```
docker login -u username

./docker_run_rb_log_sim.sh
```

Una vez se han iniciado los programas Rviz y Gazebo con los que se puedan observar los robots en el entorno simulado mientras la simulación se ejecuta, se lanzan en terminales diferentes tres ejecutables responsables de poner en funcionamiento, en primer lugar, el servidor de mapas multinivel (*Multi Level Map server*) para manejar los distintos niveles de la simulación; en segundo lugar, el servidor de Transición Multinivel (*Multi Level Transition server*) con el que se atienden tanto las solicitudes de transición como el desplazamiento de los robots y, por último, la aplicación de la Interfaz de Comandos (*Command Interface API*) con la que se da cobertura al envío y recepción de los mensajes MQTT necesarios en el intercambio de datos del sistema (ya sea entre robots o entre ellos y el FMS). Por tanto, en cada terminal se ejecuta uno de los siguientes comandos:

```
./docker_run_mlm_server.sh

./docker_run_mlt_server.sh

./docker_run_command_interface_server.sh
```

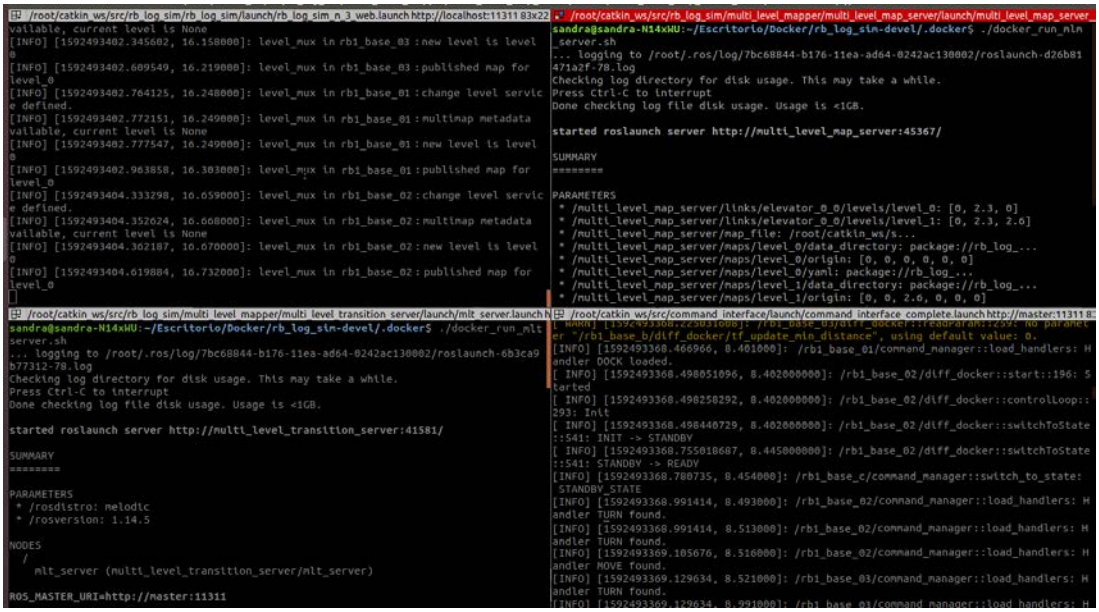

Ilustración 39: Los cuatro ejecutables necesarios para preparar el entorno de simulación

Por otro lado, en Rviz y Gazebo se simulará el comportamiento de dos robots móviles *RB-1 Base*, de manera que el robot *rb1_base_01* desempeñará el papel de prioritario o máster y el *rb1_base_02* el de esclavo. La ubicación y orientación que seguirán los robots inicialmente en cada uno de los ensayos realizados en el entorno de simulación de la *Fundació Ave María* será la siguiente:

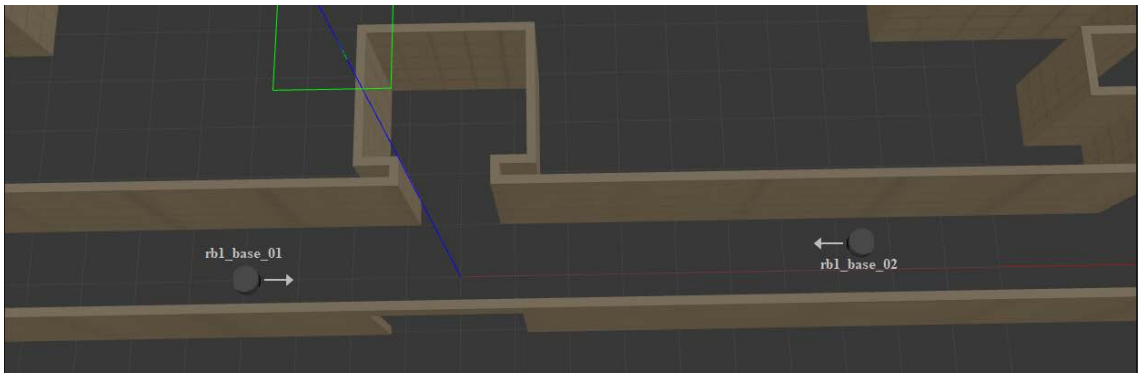

Ilustración 40: Posicionamiento de los robots móviles en el entorno de simulación

Con los cuatro archivos responsables de iniciar la simulación ejecutándose y el entorno de simulación preparado, ya se está en disposición de dar inicio a los ensayos para comprobar la eficacia del trabajo realizado. Sin embargo, se comprobará previamente el correcto funcionamiento del sistema de comunicaciones como paso previo al inicio de la simulación.

5.2.1 Ensayos del sistema de comunicaciones

Para la comprobación del sistema de comunicaciones se deben testear los envíos y recepciones de las instrucciones publicadas tanto por parte del FMS como de otros robots móviles.

Con la finalidad de testear los comandos enviados desde fuera del sistema de los robots (es decir, mediante mensajes MQTT simulando la aplicación central) se procede a abrir una terminal en el ordenador en la que se publicarán los mensajes deseados. Además, como el archivo *docker_run_command_interface_server.sh* se encuentra en ejecución, los mensajes deberían llegar a los robots implicados a través del servidor de MQTT, *Mosquitto*, que mediante su puente o *bridge* traduce dichos mensajes de MQTT a ROS.

En primer lugar, se comprueba la recepción del mensaje que inicia la solución propuesta; para ello, se le envía al robot *rb1_base_01* el mensaje de que ha entrado en el área potencialmente conflictiva "A5" mediante la siguiente orden:

```
mosquitto_pub -t /endorse/rb1_base_01/alert_zone -m "{\"robotId\": \"rb1_base_01\", \"area\": \"A5\", \"kind\": \"2\", \"localization\": \"in\"}"
```

Tal como se observa en la ilustración 41 el robot *rb1_base_01* ha recibido correctamente el mensaje.

```
[INFO] [1592487123.007000, 12.260000]: /endorse/rb1_base_01/command_manager::send_cmd: AlertZone 'rb1_base_01' 'A5' '2' 'in' sent!
[ INFO] [1592487123.007000, 12.340000000]: /rb1_base_01/alertZone::readyState::225: Goal has been reached
```

Ilustración 41: Recepción del mensaje inicial

A continuación, se envía al robot *rb1_base_01* la orden de que gire un radián a la derecha y al robot *rb1_base_02* la orden *MOVE* para que avance en línea recta tres metros; los comandos necesarios son los siguientes:

```
mosquitto_pub -t /endorse/rb1_base_01/command -m "{\"command\": \"TURN -1 0\"}"

mosquitto_pub -t /endorse/rb1_base_02/command -m "{\"command\": \"MOVE 3 0\"}"
```

Tal como se visualiza en la ilustración 42, ambos robots han recibido correctamente las instrucciones; las tres primeras líneas de la imagen corresponden al primer comando y las tres siguientes al último.

```
[INFO] [1592401272.460737, 34.640000]: /rb1_base_01/command_manager::send_cmd: Command TURN -1 sent!
[ INFO] [1592401272.460737, 34.716000]: /rb1_base_01/move::moveGoalCB::401: Received new goal
[ INFO] [1592401272.460737, 34.716000]: /rb1_base_01/move::moveGoalCB::509: Accepted goal to move: (0, 0, -1)
[INFO] [1592401275.111371, 36.006000]: /rb1_base_02/command_manager::send_cmd: Command MOVE 3 0 sent!
[ INFO] [1592401275.111371, 36.012000]: /rb1_base_02/move::moveGoalCB::401: Received new goal
[ INFO] [1592401275.111371, 36.012000]: /rb1_base_02/move::moveGoalCB::509: Accepted goal to move: (3, 0, 0)
```

Ilustración 42: Recepción de instrucciones de movimiento desde MQTT

Por otro lado, con el objetivo de verificar si los mensajes enviados desde otros robots llegan a su destino, se envían órdenes desde ROS al *broker* MQTT para que los traslade a sus respectivos destinos.

Por tanto, en una terminal nueva se lanza el ejecutable *docker_run_bash.sh*, que permite abrir una consola en el interior del *docker* y así tener acceso al sistema y *topics* de ROS del que disponen los robots. Seguidamente, el robot *rb1_base_01* se envía la instrucción *GOTO* a sí mismo para desplazarse a la posición (2, 0, 0) y, a continuación, este mismo robot procede al envío de las instrucciones *STOP* y *CONTINUE* al robot móvil *rb1_base_02*.

```
rostopic pub /rb1_base_01/command_manager/command robot_simple_command_manager
_msgs/CommandString "command: 'GOTO 2 0 0'"

rostopic pub /rb1_base_02/command_manager/command robot_simple_command_manager
_msgs/CommandString "command: 'STOP'"

rostopic pub /rb1_base_02/command_manager/command robot_simple_command_manager
_msgs/CommandString "command: 'CONTINUE'"
```

```
[INFO] [1592487124.102000, 13.340000000]: /rb1_base_01/command_manager::send_cmd: Command GOTO 2 0 0 sent!
[ INFO] [1592487124.102000, 13.579000000]: /rb1_base_01/move::readyState::355: Goal has been reached
[INFO] [1592487124.102000, 14.347000]: /rb1_base_01/command_manager::send_cmd: Command STOP sent!
[INFO] [1592487124.102000, 14.4020000000]: /rb1_base_02/move::moveGoalCB::401: Received new goal
[ INFO] [1592487124.102000, 14.402000000]: /rb1_base_02/move::moveGoalCB::509: Accepted goal to move: STOP
[INFO] [1592487124.102000, 15.019000]: /rb1_base_01/command_manager::send_cmd: Command CONTINUE sent!
[ INFO] [1592487124.102000, 15.023000000]: /rb1_base_02/move::moveGoalCB::401: Received new goal
[ INFO] [1592487124.102000, 15.023000000]: /rb1_base_02/move::moveGoalCB::509: Accepted goal to move: CONTINUE
```

Ilustración 43: Recepción de instrucciones de movimiento desde ROS

Tal y como se observa en la ilustración 43 los mensajes enviados a otros robots móviles desde ROS se han recibido correctamente.

En vista de que el sistema de mensajes enviados tanto desde el FMS (o la parte de MQTT del servidor) como desde los robots móviles funciona correctamente, se puede proceder a ejecutar por fin la solución desarrollada en este TFM.

5.2.2 Experimentos en el entorno de simulación sin obstáculos

En el entorno de trabajo de la primera simulación se dispone únicamente de dos robots móviles en una zona declarada potencialmente conflictiva como consecuencia del reducido espacio que se ven obligados a compartir.

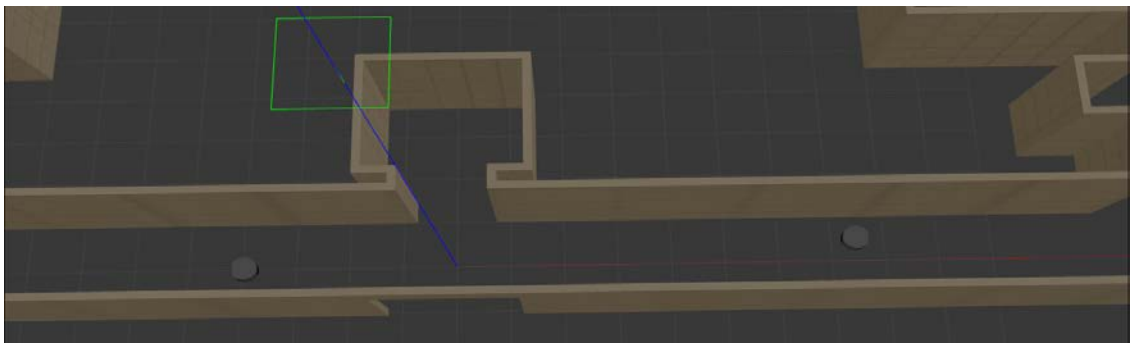

Ilustración 44: Entorno de trabajo de la primera simulación

Tal como se visualiza en la ilustración 44, el primer robot *rb1_base_01* se encuentra inicialmente a la izquierda y debe desplazarse a un punto final próximo a la posición inicial del segundo robot, *rb1_base_02*, ubicado a la derecha, que debe finalizar a su vez en una posición cercana a la posición inicial del primer robot. En consecuencia, ambos robots tendrán que cruzarse en algún punto de sus rutas, tal como muestran las trayectorias globales visualizadas por Rviz en la ilustración 45.

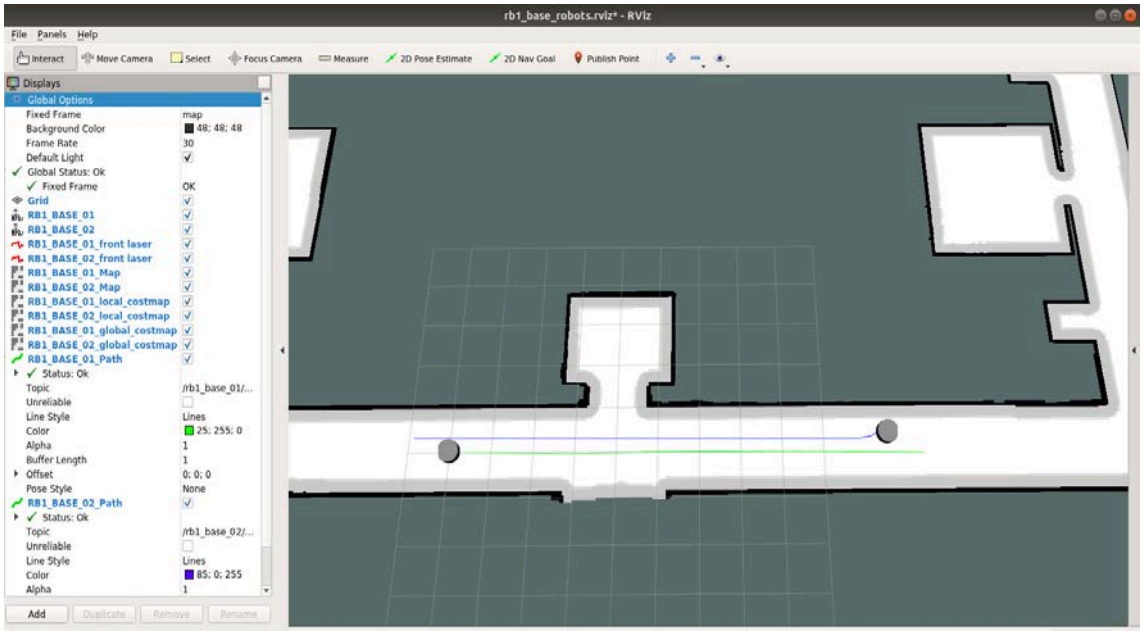

Ilustración 45: Trayectorias iniciales de los robots móviles en el primer entorno de trabajo

Inicialmente, se han realizado tres testeos en este entorno con los sistemas iniciales de los robots sin modificar (es decir, sin utilizar el paquete *decentralized_control* creado en este trabajo) para observar el comportamiento original de los robots y poder comprobar posteriormente si la implementación de dicha solución mejorará o no el sistema de navegación.

Durante la ejecución de estas pruebas se ha observado que los robots reducen la velocidad y detectan que se encuentran muy próximos el uno al otro en una zona ubicada prácticamente a la mitad de sus caminos, lo cual es lógico teniendo en cuenta que el recorrido de ambos robots es similar, no hay obstáculos que alteren sus trayectorias y la velocidad a la que circulan es muy parecida. Una vez inmersos en dicha zona, intentan recalcular las trayectorias locales propias para evitar al otro robot pero como el espacio es reducido y los robots no se coordinan entre sí acaban quedando bloqueados tras varios intentos de replanificar trayectorias y no conseguir esquivar al otro robot (tal como ha ocurrido en dos de los ensayos). En la ilustración 46 se presenta un ejemplo de la posición en la que quedan bloqueados dichos robots.

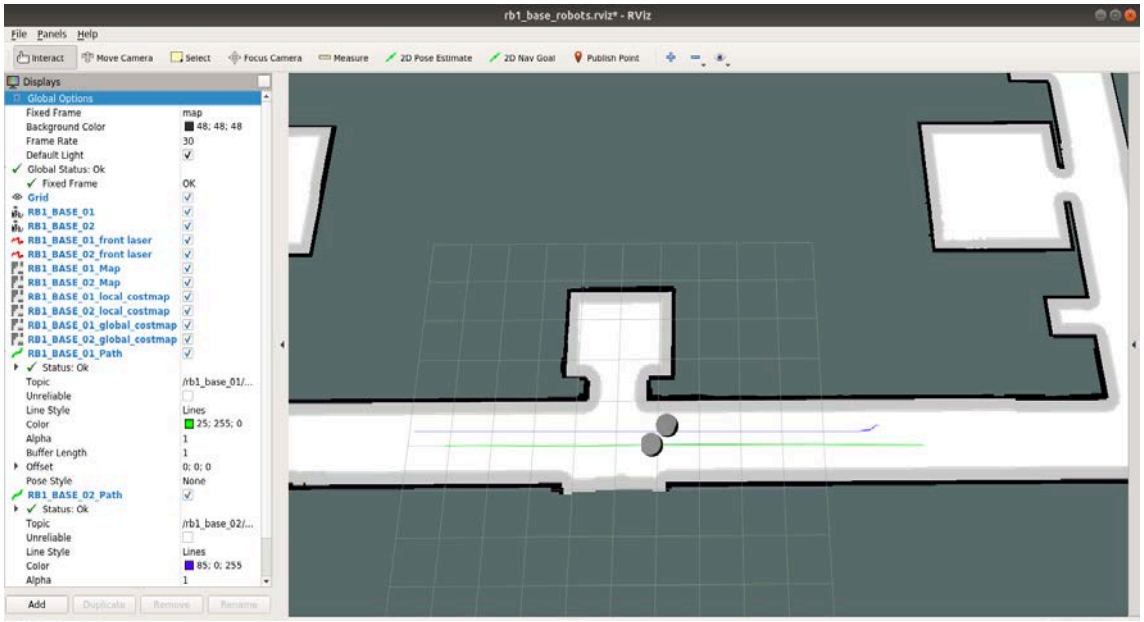

Ilustración 46: Bloqueo de los robots móviles en el primer entorno de trabajo

Una vez efectuadas las pruebas con el sistema de navegación original de los robots, se pasa a realizar otros tres testeos más utilizando ahora el paquete *decentralized_control* diseñado en este TFM.

En estos ensayos se comprueba que los robots siguen las rutas globales calculadas inicialmente hasta que se encuentran separados por menos de medio metro de distancia, momento en el que el robot prioritario *rb1_base_01* le envía al robot esclavo la orden de *STOP*, que provoca que el robot *rb1_base_02* se detenga. Seguidamente, podemos observar que el robot esclavo rota 90 grados en sentido antihorario (es decir, a la izquierda) y retrocede para dar paso al robot prioritario que a su vez, mientras el esclavo realizaba estos movimientos, modificaba ligeramente su propia trayectoria para esquivar al robot *rb1_base_02*. Posteriormente, el robot *rb1_base_01* sortea al esclavo, este replanifica su ruta y vuelve a ponerse en funcionamiento tras recibir la instrucción *CONTINUE*.

Estas acciones se reproducen casi de idéntica forma en los tres testeos efectuados puesto que las posiciones iniciales e instrucciones con que cuentan los robots móviles en estas pruebas no varían prácticamente de una a otra.

La ilustración 47 muestra la posición adoptada por el robot esclavo *rb1_base_02* para dar paso al robot prioritario.

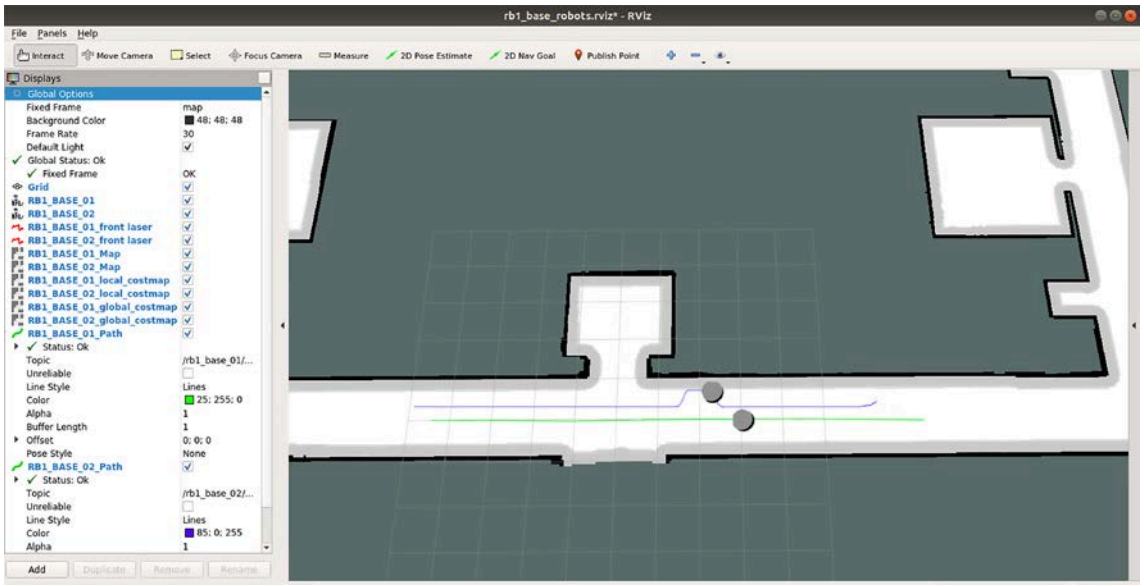

Ilustración 47: Resolución del conflicto en la navegación en el primer entorno de trabajo

En la ilustración 48 se pueden observar los mensajes intercambiados cuando el robot prioritario detecta que el otro se encuentra a poca distancia.

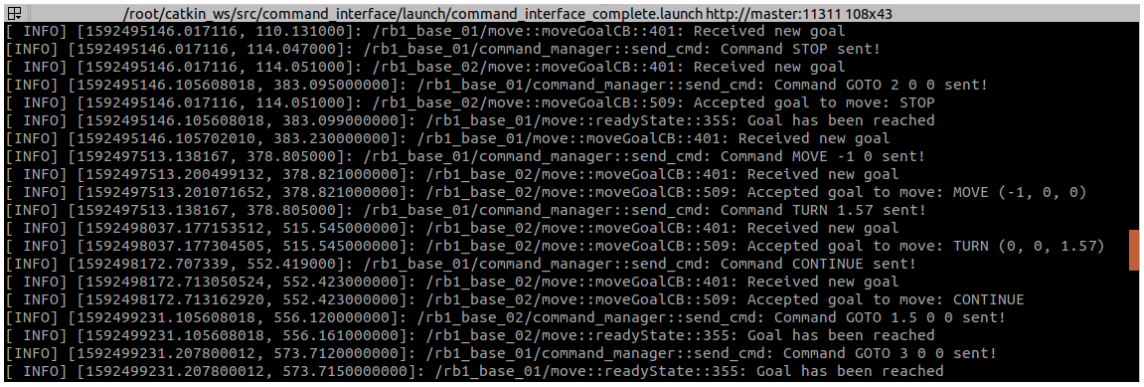

Ilustración 48: Mensajes intercambiados durante el primer entorno de trabajo

Por lo tanto, tras varios ensayos utilizando este entorno de simulación y el paquete *decentralized_control*, se ha observado que los robots móviles consiguen llegar a sus coordenadas finales en poco tiempo y sin bloqueos mientras que con el funcionamiento original del sistema de navegación quedaban bloqueados en todos los ensayos en posiciones similares y sin conseguir cumplir sus objetivos (a excepción de una de las pruebas, en la que sí consiguen evitarse y continuar con sus caminos aunque empleando más tiempo en resolver la situación de conflicto que el utilizado en los ensayos efectuados con el paquete *decentralized_control*).

5.2.3 Experimentos en el entorno de simulación con dos obstáculos

A diferencia del primer entorno de simulación, el entorno de trabajo de la segunda simulación consta de cuatro objetos, dos de los cuales son dinámicos (los propios robots *RB-1 Base*) y dos estáticos para dificultar la navegación de los robots móviles.

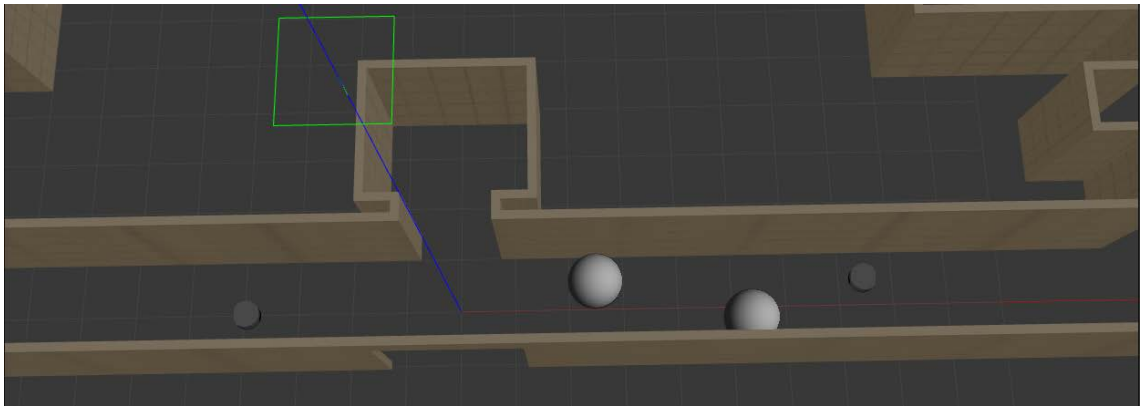

Ilustración 49: Entorno de trabajo de la segunda simulación

Tal como se representa en la ilustración 49, ambos robots deben sortear para alcanzar sus coordenadas finales (que son las mismas que en el primer entorno de simulación) tanto los obstáculos como al otro robot móvil. Las trayectorias mostradas en el mapa de Rviz de la ilustración 50 son las calculadas por los robots móviles considerando los obstáculos estáticos y evitando, por tanto, las áreas ocupadas por ellos.

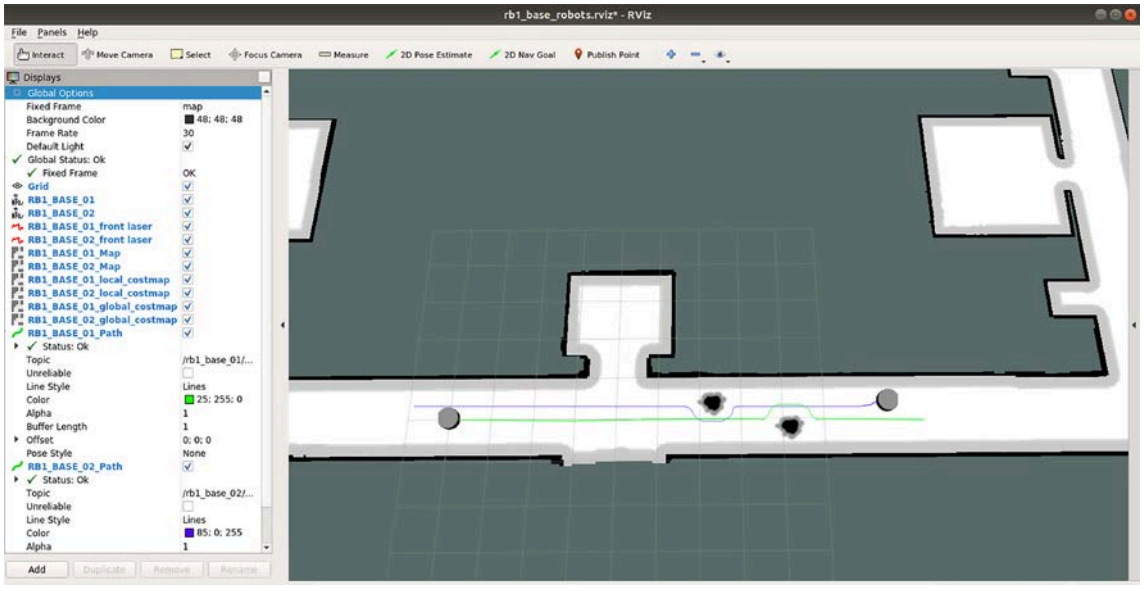

Ilustración 50: Trayectorias iniciales de los robots móviles en el segundo entorno de trabajo

En este entorno de simulación se han realizado un total de diez pruebas. En primer lugar, se efectuaron cuatro ensayos para determinar la diferencia de tiempo con la que se debía iniciar el movimiento de los robots. Para poder comprobar el modo como resuelven el conflicto de navegación que se provoca cuando se cruzan entre los obstáculos (el primero se encuentra prácticamente en el centro, mientras que el segundo obstáculo se sitúa más próximo al *rb1_base_02* ubicado a la derecha de la ilustración) cada robot debe iniciarse en un momento distinto. Una vez determinado que el robot prioritario *rb1_base_01* debe ponerse en movimiento seis segundos antes que el robot móvil *rb1_base_02* para lograr una situación de conflicto entre

obstáculos, se ha procedido a efectuar los seis ensayos restantes: tres con el sistema de navegación original de los robots móviles y otros tres dotados con el paquete *decentralized_control*.

En los tres ensayos realizados con el sistema original se ha observado que en una de las pruebas los robots colisionan pero como consecuencia del retraso en las respuestas de la simulación; como la reacción de los robots en un entorno real no es la misma que en un entorno simulado, se considera que este resultado no es determinante. En los otros dos testeos, sin embargo, tras varios intentos por parte de los robots de replanificar sus rutas locales ambos alcanzan una situación de bloqueo entre los dos obstáculos estáticos.

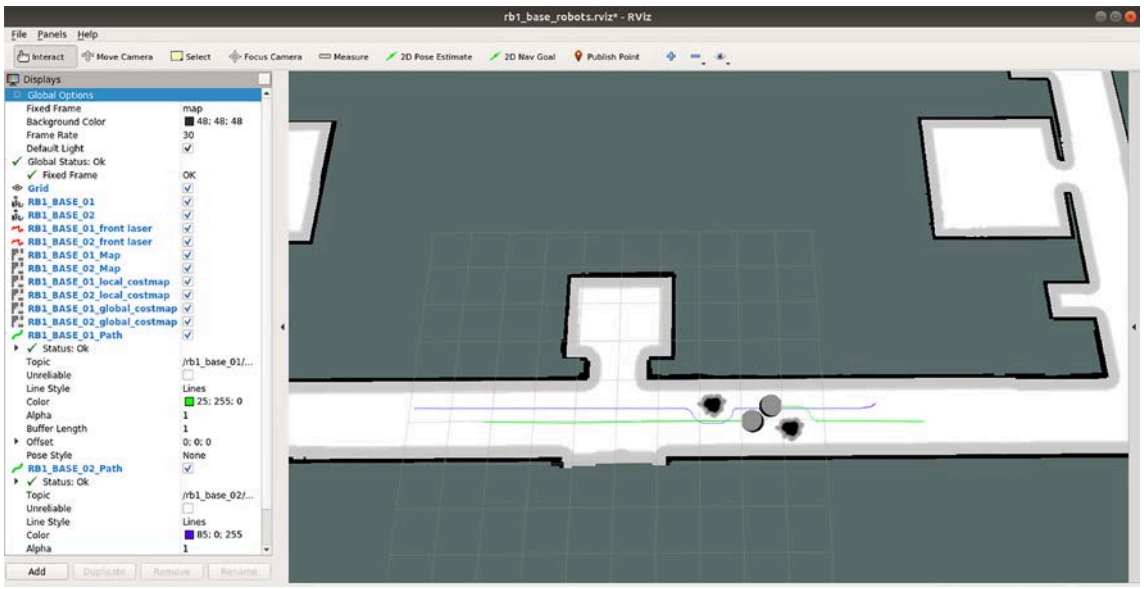

Ilustración 51: Bloqueo de los robots móviles en el segundo entorno de trabajo

Posteriormente, en las pruebas ejecutadas con el paquete *decentralized_control* se alcanzan dos escenarios distintos. En el primer escenario (producido en el primero y en el último ensayo) el robot prioritario *rb1_base_01* le envía al robot esclavo la orden de *STOP* cuando este se encuentra rebasando el segundo obstáculo (a la derecha en el entorno) y eso provoca que el robot *rb1_base_02* se detenga. A continuación, se observa que el robot esclavo avanza hasta el primer obstáculo y gira 90 grados en sentido antihorario (a la izquierda) mientras el robot prioritario modifica su trayectoria para esquivar al robot *rb1_base_02* y el segundo obstáculo. Sin embargo, en el segundo escenario originado durante la segunda prueba, el robot móvil *rb1_base_02* se encuentra junto al primer obstáculo cuando el *rb1_base_01* detecta que ambos robots están demasiado cerca, por lo que el robot prioritario únicamente envía la orden de *STOP* para que el robot esclavo se detenga. De cualquier manera, en todos los ensayos realizados con el paquete *decentralized_control*, en cuanto el robot *rb1_base_01* sortea al esclavo, este último replanifica su ruta y vuelve a ponerse en funcionamiento tras recibir la instrucción *CONTINUE*.

En la ilustración 52 se muestra la posición que adopta el robot esclavo durante el primer escenario con la que resuelve la situación de conflicto.

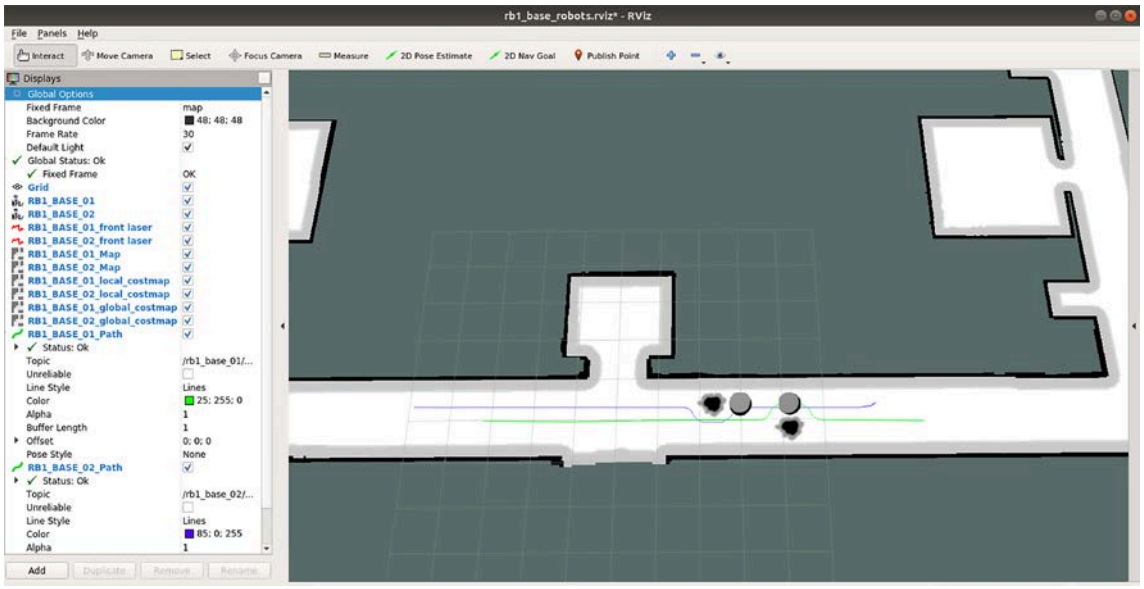

Ilustración 52: Resolución del conflicto en la navegación en el segundo entorno de trabajo

En conclusión, al comparar las pruebas realizadas con el sistema original de navegación y el modificado, se ha observado que los robots móviles alcanzan sus posiciones finales con éxito en el entorno de simulación únicamente mediante la ejecución del sistema de navegación con el paquete *decentralized_control*.

5.3 Conclusiones de las pruebas realizadas

Una vez efectuados los ensayos sobre el sistema de comunicaciones desarrollado y llevados a cabo los diversos testeos en los entornos de simulación de la *Fundació Ave María* preparados con y sin obstáculos, se observa que la solución desarrollada para el control descentralizado de múltiples robots móviles funciona correctamente en simulación y que recibe y envía mensajes tanto de la aplicación central del control de flota FMS como de otros robots. Asimismo, se ha comprobado que dicha solución es capaz de evitar y resolver conflictos en el sistema de navegación (bloqueos o colisiones) antes de que se produzcan.

CAPÍTULO 6
Conclusiones

El propósito de este proyecto consistía en diseñar un sistema de comunicaciones y un algoritmo para la resolución de conflictos en el sistema de navegación de múltiples robots móviles mediante un control descentralizado de la aplicación central para el control de la flota robótica, dentro de la tarea *Human-aware path planning algorithms* de la que es responsable el instituto ai2 de la UPV en el proyecto europeo ENDORSE.

Como se ha podido observar en el capítulo de ensayos y resultados, el sistema de comunicaciones implementado para el intercambio de datos entre los robots móviles y los mensajes definidos para dicho sistema funcionan adecuadamente, ya que estos mensajes se publican y reciben correctamente. Además, se ha verificado que el algoritmo desarrollado cumple satisfactoria y eficientemente con las expectativas de resolver los conflictos en el sistema de navegación de múltiples robots móviles de forma descentralizada que se encuentren en zonas potencialmente conflictivas empleando poco tiempo en su resolución.

Dentro del proyecto ENDORSE, la implementación de esta solución en los robots móviles supondrá una mejora en el rendimiento de la aplicación central de flota; el algoritmo desarrollado hará que esté menos saturada con las peticiones de la navegación de los robots y esto le permitirá atender las solicitudes del resto de funcionalidades de forma óptima y en un tiempo menor. Además, otra ventaja que aportará esta solución será la optimización de la carga de sus baterías, dado que los robots móviles utilizarán menos tiempo y movimientos en resolver situaciones de conflicto y emplearán, por tanto, menos energía que la que se emplea sin la aplicación de esta solución. Esto les permitirá realizar un mayor número de tareas antes de precisar volver a la unidad de carga.

Por otro lado, debido al impacto de la pandemia del COVID-19 no se ha podido cumplir la planificación de trabajo estimada inicialmente ni la estancia programada en la empresa *SingularLogic* (Atenas, Grecia) colaboradora del proyecto ENDORSE. Sin embargo, se ha seguido satisfactoriamente el plan de trabajo corregido a principios de abril. No obstante, las fases de implementación y resultados, aunque han finalizado en el entorno de simulación (que es lo que se estimó para este Trabajo Final de Máster) siguen abiertas en el proyecto ENDORSE hasta que acabe de verificarse el correcto funcionamiento del trabajo desarrollado en los robots móviles desplazándose por el entorno de real. El cierre de dichas fases se estima que se

produzca durante el mes de julio, pero dependerá de los permisos del instituto ai2, de la UPV y, sobre todo, de la evolución de los rebrotes de esta pandemia.

6.1 Valoración personal

La tecnología empleada en este trabajo, me ha permitido adquirir nuevos conocimientos ya que esta ha sido mi primera ocasión para trabajar con robots móviles para uso real (no educativo) con un sistema de control mucho más complejo, así como con el lenguaje de programación Python y la herramienta de edición de código *Visual Studio Code*. En cuanto a la plataforma software ROS (*Robot Operating System*) y las herramientas Rviz y Gazebo debo decir que ya las conocía; sin embargo, este trabajo y el mes de formación en el instituto ai2 me han permitido profundizar en ellas, lo que me ha permitido culminar este proyecto con los objetivos alcanzados y me ha ayudado a interesarme aún más por el sector de la robótica. Además, cuando me encontraba en situaciones de bloqueo o incertidumbre ante problemas complejos, la colaboración con otros investigadores y equipos me han permitido superarlas con rapidez y determinación.

Durante el desarrollo de este proyecto me he apoyado, sobre todo, en los conocimientos adquiridos de las asignaturas cursadas en el máster, tales como la asignatura de Inteligencia Ambiental (me proporcionó los conocimientos necesarios para elaborar el sistema de comunicaciones mediante el protocolo MQTT); empleé y mejoré los conocimientos robóticos adquiridos en la asignatura de Automatización y Robótica; la asignatura de Aplicaciones Gráficas y Multimedia y la de Computación de Altas Prestaciones me sirvieron para las simulaciones y el tratamiento de las imágenes y datos obtenidos de los sensores; y, por último, la asignatura Servicios y Aplicaciones Distribuidas así como su antecesora la asignatura Tecnología de Sistemas de Información en la Red del Grado en Ingeniería Informática de la UPV me proporcionaron los conocimientos básicos en *docker* necesarios como consecuencia de la pandemia para la creación del entorno de simulación en el que se han ejecutado las diversas partes del proyecto ENDORSE.

Adicionalmente, la realización de este Trabajo Fin de Máster y la participación en el proyecto europeo ENDORSE ha supuesto un nuevo reto en mi vida, no solo por enfrentarme a nuevas tecnologías sino también por la necesidad de adaptarme a trabajar en colaboración con mis compañeros de equipo y otros equipos participantes de ENDORSE. Este trabajo me ha mostrado un mundo que era desconocido para mí hasta ahora y me ha ayudado hacer frente a nuevos retos y situaciones (como la colaboración y preparación de reuniones, auditorías y videoconferencias con todos los miembros del proyecto en inglés), a la vez que me ha dado la

confianza necesaria para hacer frente a proyectos de esta envergadura con los objetivos cumplidos e intercambiar conocimientos con otros profesionales de diversas especialidades.

6.2 Líneas futuras

Por último, dado que la tarea *Human-aware path planning algorithms* y el proyecto ENDORSE continúan y el resultado final del algoritmo para la resolución de conflictos en la navegación de múltiples robots móviles mediante control descentralizado de la aplicación central para el control de flotas ha sido satisfactorio, en un futuro se quiere implementar para áreas limitadas a otro valor determinado de robots móviles circulando simultáneamente y escenarios más complejos que los contemplados en este trabajo, como por ejemplo en entornos con la circulación incrementada o con niveles más altos de interacción con humanos (salas de espera, cafeterías, etc. en los que se puede llegar a concentrar grandes grupos de personas). Además, ahora que se ha verificado el correcto funcionamiento de la solución propuesta en este TFM en los robots simulados, solo cabe esperar a que sea posible realizar las estancias requeridas por el proyecto ENDORSE, momento en el que se procederá a unificar e implantar el trabajo realizado por los diversos equipos colaboradores en este proyecto.

CAPÍTULO 7
Glosario de términos

- **Industria 4.0:** la cuarta revolución industrial utiliza nuevas técnicas y tecnologías inteligentes (como la inteligencia artificial, la robótica, la IoT, la analítica, …) para la mejora de los procesos de fabricación otorgándoles mayor flexibilidad e individualización.

- **LAN:** se trata de una red de área local, *Local Area Network*, que cuenta con la propiedad de transmitir rápidamente grandes cantidades de datos. Se utiliza para conectar los dispositivos de áreas normales como, por ejemplo, los ordenadores de una vivienda privada, los de un laboratorio o incluso los aparatos de una empresa.

- **Socket TCP/IP:** un socket consiste en un punto de acceso que puede crear una aplicación mediante la arquitectura cliente-servidor, a fin de acceder a servicios de comunicación mediante protocolos de internet y así conectar dos dispositivos para que puedan intercambiar datos. El protocolo TCP/IP, identifica las direcciones de red IP de los dispositivos a conectar y ofrece un formato estándar para establecer dicha comunicación.

- **WAN:** cuyas siglas significan *Wide Area Network*, es decir, una red de área amplia. Esta red contiene conexiones de gran envergadura, ya que se extiende por grandes zonas geográficas, tales como países o continentes.

CAPÍTULO 8
Referencias bibliográficas

[1] "ISO 8373:2012, Robots and robotic devices — Vocabulary," *Technical Committee ISO/TC 184*, Mar. 2012. https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en (accessed Mar. 04, 2020).

[2] "UNE-EN ISO 10218-2:2011 Robots para entornos industriales." https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0048668 (accessed Mar. 04, 2020).

[3] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, pp. 1–22, 2019, doi: 10.1177/1729881419839596.

[4] J. P. N. Cruz, M. L. Dimaala, L. G. L. Francisco, E. J. S. Franco, A. A. Bandala, and E. P. Dadios, "Object recognition and detection by shape and color pattern recognition utilizing Artificial Neural Networks," *2013 Int. Conf. Inf. Commun. Technol. ICoICT 2013*, pp. 140–144, 2013, doi: 10.1109/ICoICT.2013.6574562.

[5] L. Ferriere, B. Raucent, and G. Campion, "Design of omnimobile robot wheels," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 4, no. April, pp. 3664–3670, 1996, doi: 10.1109/robot.1996.509271.

[6] J. D. La Fuente, J. Santiago, A. Román, C. Dumitrache, and D. Casasanto, *Handbook on Robotics*, vol. 25, no. 9. 2014.

[7] N. Shiroma, Y. H. Chiu, Z. Min, I. Kawabuchi, and F. Matsuno, "Development and control of a high maneuverability wheeled robot with variable-structure functionality," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4000–4005, 2006, doi: 10.1109/IROS.2006.281839.

[8] S. Xingyong, H. K. Lee, and H. Cho, "A sensor fusion method for mobile robot navigation," *2006 SICE-ICASE Int. Jt. Conf.*, vol. 85, no. 1, pp. 5310–5316, 2006, doi: 10.1109/SICE.2006.315317.

[9] I. Bambino, "Una Introducción a los Robots Móviles," *Asoc. Argentina Control Automático - AADECA*, p. 86, 2008.

[10] T. Lozano-Pérez, "Autonomous Robot Vehicles," *Springer-Verlag New York Inc.*, p. 478, 1990, doi: 10.1007/9781461389972.

[11] N. Maalouf, A. Sidaoui, I. H. Elhajj, and D. Asmar, "Robotics in Nursing: A Scoping Review," *J. Nurs. Scholarsh.*, vol. 50, no. 6, pp. 590–600, 2018, doi: 10.1111/jnu.12424.

[12] M. Bates, "Robotic Pets: A Senior's Best Friend?," *IEEE Pulse*, vol. 10, no. 4, pp. 17–

20, 2019, doi: 10.1109/MPULS.2019.2922565.

[13] U. D. Hanebeck, C. Fischer, and G. Schmidt, “ROMAN: A mobile robotic assistant for indoor service applications,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2, pp. 518–525, 1997, doi: 10.1109/iros.1997.655061.

[14] E. Ettelr, R. Furtwängler, U. D. Hanebeck, and G. Schmidt, “Design Issues of a SemiAutonomous Robotic Assistant for the Health Care Environment,” *J. Intell. Robot. Syst. - JIRS*, vol. 22, pp. 191–209, 1998, doi: 10.1023/A:1008082024638.

[15] M. E. Pollack, S. Engberg, J. T. Matthews, J. Dunbar-jacob, C. E. Mccarthy, and S. Thrun, “Pearl: A mobile robotic assistant for the elderly,” *Architecture*, vol. 2002, pp. 85–91, 2002, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Pearl+:+A+Mobile+Robotic+Assistant+for+the+Elderly#0.

[16] S. Baumgarten, T. Jacobs, and B. Graf, “The robotic service assistant – Relieving the nursing staff of workload,” *50th Int. Symp. Robot. ISR 2018*, pp. 119–122, 2018.

[17] S. Mallikarachchi, D. Chinthaka, J. Sandaruwan, I. Ruhunage, and T. D. Lalitharatne, “Motor Imagery EEG-EOG Signals Based Brain Machine Interface (BMI) for a Mobile Robotic Assistant (MRA),” *Proc. - 2019 IEEE 19th Int. Conf. Bioinforma. Bioeng. BIBE 2019*, no. December, pp. 812–816, 2019, doi: 10.1109/BIBE.2019.00151.

[18] AER Automation, “Los robots móviles de MIR entregan quimioterapia,” *AER Automation*, Oct. 30, 2018. https://www.aer-automation.com/los-robots-moviles-de-mir-entregan-quimioterapia/ (accessed Mar. 09, 2020).

[19] G. Z. Yang *et al.*, “Combating COVID-19-The role of robotics in managing public health and infectious diseases,” *Sci. Robot.*, vol. 5, no. 40, pp. 1–3, 2020, doi: 10.1126/scirobotics.abb5589.

[20] R. Treviño and A. Villanueva, “Tecnología vs COVID-19: TecSalud usa robot para tratar pacientes,” *Tecnológico de Monterrey*, Apr. 13, 2020. https://tec.mx/es/noticias/nacional/salud/tecnologia-vs-covid-19-tecsalud-usa-robot-para-tratar-pacientes (accessed Mar. 09, 2020).

[21] “ZenZoe Robot, un robot móvil que combate el COVID-19,” *Distribución Actualidad*, May 08, 2020. https://www.distribucionactualidad.com/zenzeo-robot-movil-covid-19/ (accessed May 10, 2020).

[22] Novologistica, “ASTI Mobile Robotics y BOOS Technical Lighting desarrollan un robot móvil para combatir el COVID19,” *Novologistica.com*, May 11, 2020. https://novologistica.com/manutencion-y-almacenaje/asti-mobile-robotics-y-boos-technical-lighting-desarrollan-un-robot-movil-para-combatir-el-covid19/ (accessed May 12, 2020).

[23] “secpho - Collaborate to innovate.” http://www.secpho.org/ (accessed Mar. 10, 2020).

[24] I. Hernández Velasco, “Made in Spain contra la covid: la pandemia aflora el músculo tecnológico español | Empresas,” *El Mundo*, May 24, 2020. https://www.elmundo.es/economia/empresas/2020/05/24/5ec8f85421efa078048b45db.html (accessed May 26, 2020).

[25] K. Pretz, “Thermal Cameras Are Being Outfitted to Detect Fever and Conduct Contact Tracing for COVID-19 - IEEE Spectrum,” *IEEE Spectrum*, May 04, 2020. https://spectrum.ieee.org/news-from-around-ieee/the-institute/ieee-member-news/thermal-cameras-are-being-outfitted-to-detect-fever-and-conduct-contact-tracing-for-covid19 (accessed May 14, 2020).

[26] “MTS UVC - Desinfección con luz ultravioleta.” https://www.mtsuvc.com/ (accessed May 14, 2020).

[27] “La Conselleria de Innovación elige un proyecto del Instituto ai2 para luchar contra el COVID-19,” *ai2 UPV*, May 13, 2020. https://www.ai2.upv.es/la-conselleria-de-innovacion-elige-un-proyecto-del-instituto-ai2-para-luchar-contra-el-covid-19/ (accessed May 14, 2020).

[28] “Instituto Universitario de Automática e Informática Industrial.” https://www.ai2.upv.es/ (accessed Mar. 08, 2020).

[29] “Home - Endorse Project.” http://www.endorse-project.eu/ (accessed Mar. 08, 2020).

[30] N. Ramdani *et al.*, “A safe, efficient and integrated indoor robotic fleet for logistic applications in healthcare and commercial spaces: The endorse concept,” *Proc. - IEEE Int. Conf. Mob. Data Manag.*, vol. 2019-June, no. Mdm, pp. 425–430, 2019, doi: 10.1109/MDM.2019.000-8.

[31] M. Jäger and B. Nebel, “Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 3, pp. 1213–1219, 2001, doi: 10.1109/IROS.2001.977148.

[32] “RB-1 BASE — Robotnik.” https://www.robotnik.es/logistics/portfolio/rb-1-base-2/ (accessed Apr. 15, 2020).

[33] “ROS/Tutorials - ROS Wiki.” http://wiki.ros.org/ROS/Tutorials (accessed Apr. 15, 2020).

[34] R. Téllez, A. Ezquerro, and M. A. Rodríguez, “ROS Basics in 5 days,” p. 219, 2018, [Online]. Available: www.theconstructsim.com.

[35] G. van Rossum, *An Introduction to Python*. Network Theory Ltd, 2018.

[36] G. C. Hillar, *MQTT Programming with Python: Work with the lightweight IoT protocol in Python*. Packt Publishing, 2018.

[37] “The Construct.” https://www.theconstructsim.com/ (accessed Apr. 17, 2020).

[38] “Robot Ignite Academy.” https://www.robotigniteacademy.com/en/ (accessed Apr. 17,

2020).

[39] M. Alajlan and A. Koubâa, *Robot Operating Systems (ROS)*, vol. 625, no. Volume 1. 2016.

[40] F. Anton, M. L. Gavrilova, and C. J. Kenneth Than, *Transactions on Computational Science IX: Special Issue on Voronoi Diagrams in Science and Engineering*. 2010.

[41] "move_base - ROS Wiki." http://wiki.ros.org/move_base (accessed May 17, 2020).

[42] "Robots 'enfermeros' para hospitales y centros de mayores," *Instituto ai2 - UPV*, Feb. 28, 2020. https://www.ai2.upv.es/robots-enfermeros-para-hospitales-y-centros-de-mayores/?fbclid=IwAR2OAm2XLIsE_84XHUu5NSsg5vbULrqZct1izg4dKLk3A90LPlW3W5sa3RA (accessed Jun. 10, 2020).

[43] E. Project, "RB-LOG / rb_log_sim · GitLab." https://gitlab.com/RB-LOG/rb_log_sim (accessed Jun. 16, 2020).

[44] "Docker Hub." https://hub.docker.com/ (accessed Jun. 16, 2020).